Computational Power of the **Quantum Turing Automata**

Sina Jafarpour, Mohammad Ghodsi, Keyvan Sadri, Zuheir Montazeri

Abstract

Lots of efforts in the last decades have been done to prove or disprove whether the set of polynomially bounded problems is equal to the set of polynomially verifiable problems. This paper will present an overview of the current beliefs of quantum complexity theorists and discussion detail the impacts these beliefs may have on the future of the field. We introduce a new form of Turing machine based on the concepts of quantum mechanics and investigate the domain of this novel definition for computation. The main object is to show the domain of these Computation machines and the new definition of Algorithms using these automata.

In section one we give a brief fundamental overview of classical complexity theory, Turing machine and all the fundamental concepts required for the next chapters. In section two we describe various classes of classical complexity automata. In next chapter we introduce quantum complexity featuring Quantum Turing Machines and discuss its relationship to classical complexity. Section four presents a relationship between Quantum complexity classes based on the quantum Turing machines and Quantum oracles, and their relationship with corresponding classical complexity classes. And section five presents a discussion on the practical phenomena in problem solving using quantum computers. Finally in section five we speculate briefly on the direction we believe the field to be headed, and what might reasonably be expected in the future.

Keywords

Quantum, Computation, Complexity, Turing machine, Oracle, Computability, Grover algorithm

1. Introduction

Early in the 20th century Gödel, Church and Turing proposed 3 different models of computation:

- general recursive functions of Gödel
- lambda expressions
- the Turing machine

Somewhat later it turned out that they were very much the same. But later still, towards the end of the 20th century, it turned out that certain physical assumptions, which may not necessarily correspond to how certain computations can be done, were smuggled into all three models. In particular quantum computation, the subject of this lecture, is not modeled correctly by any of the above. But there are even some aspects of classical computation, which are not adequately accounted for by the Turing machine and equivalent models, e.g., the thermodynamics of computation.

The Turing machine was invented by Alan Turing in order to address Hilbert's Entscheidungsproblem. It is sufficiently simple so that various mathematical theorems can be proven about it. In particular by using this model Turing was able to show that the Entscheidungsproblem must be answered in negative, i.e., there is no mechanical procedure, in general, which can be used to decide a theoremhood of an arbitrary statement in mathematics. This, in combination with Gödel theorem, came as a bit of a surprise to mathematicians. On the other hand, it merely demonstrated what the greatest mathematicians always knew and practiced, namely that mathematics is an art. Also, that some of the best mathematics is constructed by broadening a particular theory that is an object of some assertion and attacking the problem from a higher level. For example, algebraic problems can often be tackled with a surprising efficacy by rolling out the apparatus of complex analysis.

The original Turing machine was deterministic (DTM): the head would be always in a single state, which would uniquely determine which direction it would go into and how far. There is a variant of the Turing machine, which is not deterministic. The head may be in a state, which gives the machine certain choices as to the direction and length of the next traverse. The choices are then made by throwing dice and possibly applying some weights to the outcome. A machine like that is called a probabilistic Turing machine (PTM), and it turns out that it is more powerful than the deterministic Turing machine in the sense that anything computable with DTM is also computable with PTM and usually faster.

But both PTM and DTM are based on classical physics: the states of the tape and of the head are always readable and writable, data can be always copied, and everything

is uniquely defined. As a result, the definition of human computation is deeply affected by the environment we are living in. The main question is that on the grounds that the human brain is not environmental dependent and also new sorts of abstract environments have been innovated, is there a way to define the computation in a more abstract form?

2. Classical Complexity Classes

2.1. The Class P

The set of all languages that can be computed in polynomial time on a deterministic Turing machine is called the class P. Familiar examples of problems in P include sorting a list of elements, computing the maximum flow between two points in a network of pipes, and finding the shortest path between two points in a graph.

2.2. The Class NP

Definition of NP is that it contains all languages that are verifiable in polynomial time on a deterministic machine. This means that given an input and a certificate of the input's membership in a language, a machine can check if the certificate is valid in polynomial time. For example, in determining whether a number k is composite, one of the factors of k could be used as the certificate. A machine could then determine the validity of the certificate by dividing k by it and checking that the remainder is 0.

Since a deterministic Turing machine is just a specific type of nondeterministic machine, $P \subseteq NP$. Perhaps the most famous open question in computer science is whether or not the classes P and NP are equivalent. Does nondeterminism increase the power of a Turing machine, to the point where we can quickly solve problems that we otherwise could not? Although it may appear "obvious" that these two classes are not equivalent, given that a nondeterministic machine can perform parallel computation, no one has been able to prove this distinction.

2.3. The Class BPP

A language L is in the complexity class BPP (Bounded-Error Probabilistic Polynomial-Time), if there exists a probabilistic Turing machine, M, that runs in polynomial time such that:

1. if $x\psi$ is in L, then $M\psi$ accepts $x\psi$ with probability $\top\leftarrow$ 2/3

2..if x is not in L, then M accepts x with probability < 1/3

In other words, BPP is the class of decidable problems solvable by a probabilistic Turing machine in polynomial time with an error probability of at most 1/3. Note, however, that the constant 1/3 is arbitrary. Any constant value in the range (0, 0.5) gives an equivalent definition of the class BPP, since repeated executions of the probabilistic machine M on x can bring the probability of correct behavior arbitrarily close to 1.

It is clear that $P \subseteq BPP$, since a deterministic algorithm that runs in polynomial time, is like a probabilistic algorithm that runs in polynomial time with an error probability of 0, which is less than 1/3. It is still an open question whether $P \subset BPP$ or $P = BPP$, though currently there exist problems which are known to lie in BPP but not known to lie in P, such as the problem of finding square roots modulo a prime number. It is also an open question whether $BPP \subseteq NP$ or $NP \subseteq BPP$.

2.4. The Class PSPACE

The class PSPACE is the set of decision problems that can be solved by a Turing machine using a polynomial amount of tape, given an unlimited amount of time. Analogously, EXPSPACE is the set of decision problems that can be solved by a Turing machine using an exponential amount of tape, given an unlimited amount of time. It is clear that $BPP \subseteq PSPACE$, since a polynomial time machine can only using a polynomial amount of tape. It's an open question whether this inclusion is strict, though it is generally believed that $BPP \subseteq PSPACE$. It has been proven that $NP \subseteq PSPACE$, and that $PSPACE \subseteq EXPSPACE$.

3. Quantum Abstract Machine

The quantum analog to a Boolean circuit that operates on n bits is a quantum circuit that operates on n Qubits. A Qubit, like a classical bit, can be either 0 or 1, but unlike a classical bit, can also be in a normalized superposition of these states. The state of n Qubits, then is a normalized vector in $N = 2^n$-dimensional Hilbert space In the same way a boolean circuit is built from NAND gates, a quantum circuit is built from universal gates, represented by unitary $N \times N$ matrices, that can be composed to produce any unitary $N \times N$ matrix. The result of applying a gate is the product of its matrix and the vector representing the state being operated on. One additional operation on a quantum state is measurement, which probabilistically determines a physical property of the state, and simultaneously removes elements of the initial superposition that are inconsistent with the result of the measurement.*

3.1. Quantum Turing Machines

A mathematical theory of computation that is based on quantum physics is bound to be different. As you move from classical physics to quantum physics there is a

* We have assumed that the reader has the rudimentary familiarity with the Quantum Computation and Quantum Algorithms, [1],[2].

qualitative change in concepts that has profound ramifications.

In the quantum Turing machine read, write, and shift operations are all accomplished by quantum interactions. The tape itself exists in a quantum state as does the head. In particular in place of the Turing cell on the tape, that could hold either 0 or 1, in quantum Turing machine there is a Qubit, which can hold a quantum superposition of 0 and 1. The quantum Turing machine can encode many inputs to a problem simultaneously, and then it can perform calculations on all the inputs at the same time. This is called **quantum parallelism**. The tape of the quantum Turing machine can be drawn as shown in **Fig.1** Which each arrowed circle stands for a Qubit.

The machine evolves in many different directions simultaneously. After some time t its state is a superposition of all states that can be reached from the initial condition in that time. This model, like the classical Turing machine was sufficiently simple and at the same time universal to prove various theorems about quantum computation. In 1990 Christopher Moore from Cornell University showed that a single classical particle moving in a three-dimensional potential well made of a finite number of parabolic mirrors is equivalent to a Turing machine, and hence is capable of universal computation.

3.2. Formal Definition of Quantum Turing Machine

A quantum Turing machine is defined as a triple (Q, Σ, δ) where Q is a finite set of states that includes a start and final state, $\sum$ is the tape alphabet and includes the blank letter **'#'**, and δ is a transition functionψ $\delta : Q * \Sigma \to C^{Q*\Sigma*D}$ where D = {L,R} **i**s the direction in which the tape head moves. The set C is all complex numbers whose kth bit can be computed in time polynomial in k. The configuration of the quantum machine is a superposition of configurations, where a configuration is an element of $\Sigma^z * Q * Z$ the first member of which corresponds to the tape contents, the second the state, and the last the tape head position. The function δ must be unitary. A quantum machine halts when its state is a superposition of only those configurations that are in the final state. The output of the machine is the corresponding superposition of the tape contents. For a decider, the output contains a 0 in the start cell if it rejects, 1 if it accepts. The probability that the decider accepts is the total amplitude of accepting configurations in its output superposition.

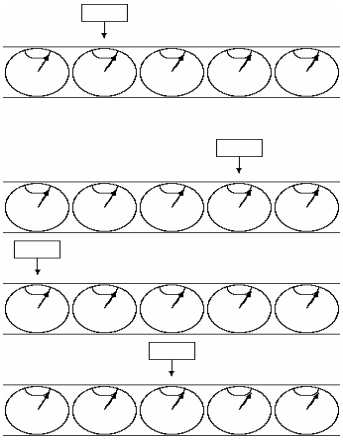

Fig.1 Quantum Turing Machine

3.3. Quantum Computability

Bennet showed that any classical circuit can be converted into an equivalent reversible circuit, and further that this conversion can be done efficiently. Thus, we see immediately that a quantum computer is at least as powerful as a classical computer. To demonstrate that a quantum computer is no more powerful than a classical computer we present a classical Turing machine that simulates an arbitrary quantum circuit. The quantum gates of the circuit are given, suitably encoded, as input to the Turing machine as N × N matrices, along with the measurements made on the circuit, in the order of application. In addition, the states of the Turing machine contain all the information necessary about the physical properties of each of the quantum basis states. The machine also receives an encoding of the N-dimensional initial state in its input. The machine then applies each quantum gate to the current state by doing a matrix multiplication.

One detail that may not be immediately obvious is the manner in which our simulation should handle measurement, since measurement appears to rely entirely on the physical properties of the system and true "randomness." It turns out, however, that a non-deterministic Turing machine will still compute correctly Providing some computation path succeeds. Therefore we appeal to nondeterminism and let our Turing machine decide the outcome of the measurement nondeterministically, and then use the information it knows about each of the quantum basis states to delete inconsistent elements. Finally, it renormalizes the resulting vector. Thus the Turing machine can simulate each of the operations of a quantum circuit, and therefore compute anything that can be computed on a quantum circuit.

4. Complexity of Quantum Turing machines

Having proven that a quantum computer can compute exactly the same set of languages as a classical computer, we now examine quantum complexity.

4.1. The Class BQP

The quantum analog to BPP is the class BQP. This consists of all languages for which a quantum machine gives the right answer at least $\frac{2}{3}$ of the time. It is known that BPP $\subseteq$ BQP, since anything that can be computed on a classical machine can be computed on a quantum machine with little overhead. However, it is not known whether or not this containment is strict. While there are problems such as factoring and computing a discrete log that are in BQP but not known to be in BPP, no one has actually proven that these problems are not in BPP. It is currently known that BQP sits between BPP and PSPACE in the complexity hierarchy. Thus BQP contains all of P and BPP, and potentially some problems in NP but probably none that are NP-complete, and perhaps some problems in PSPACE that are not in NP. The latter two postulations, however, have not been proven.

4.2. Relationship between NP and BQP

Although nothing universal has been proven about the relationship between the relative speeds of quantum and classical computation, several such relativized† results have made progress that suggests the direction that future work will eventually go. The first results discovered appeared to suggest that quantum computation was significantly faster than classical computation. Among these was the discovery by Deutsch, Jozsa, Berthiaume, and Brassard, of oracles under which problems could be found for which a quantum machine computed the answer with certainty in polynomial time, whereas requiring a probabilistic classical machine to solve the same problem with certainty required exponential time for some inputs. Note, however, that relative to the same oracle, these problems were in BPP, indicating that only the requirement of certainty pushed the problem across-the-exponential time barrier.

Bernstein and Vazirani showed that there exist oracles under which there exist problems that are in BQP but not BPP, which is now taken as some of the earliest evidence that QTMs are more powerful than probabilistic Turing Machines. Building upon this, Simon proved the stronger result that there exists an oracle relative to which BQP cannot even be simulated by a probabilistic Turing Machine allowed to run for an exponential number of steps Quantum Turing machine can be used to simulate the classical Turing machine and the probabilistic Turing machine too. But quantum Turing machine can do more than that. For example it can generate truly random numbers, something that classical Turing machines cannot do.

Despite this evidence as to the power of quantum computation, the above results do not appear to break any major ground in terms of computational complexity. That is, although they suggest that quantum computation is at least somewhat more powerful than classical computation, they do not lend any real insight as to whether it will allow us to compute NP-complete problems. However, in the following sections we will demonstrate that class NP can never be solved in less than $\mathbf{p2_n}$ time.

4.3. Oracle Quantum Turing machine

Consider SAT, the prototypical example of an NP-complete problem. An instance of this problem consists of a Boolean function $f(x_1, \ldots, x_n) = c_1 \wedge \ldots \wedge c_m$ the SAT problem asks you to determine whether there exists a satisfying assignment that is, an input $(a_1, \ldots, a_n)$ such that $f(a_1, \ldots, a_n) = 1$**.** UNIQUE-SAT is a variant of SAT that poses the same problem with the restriction that f must have zero or one satisfying assignments, but no more. As it turns out, there is a randomized reduction from SAT to UNIQUE-SAT; thus, the two problems are equally hard. We will use the black box model when considering this problem. In this model, we know that either $f \equiv 0$ or there exists exactly one a such that $f(a) = 1$, where a is chosen uniformly at random. That is, f is treated as a black box; we can make queries to f, but we have no access to the Boolean formula itself. Equivalently we can represent f by a table of $N = 2^n$ entries where either none or exactly one entry is 1. Ideally we want a quantum algorithm that solves this problem in time $O(\text{poly}(n))$.

Can a quantum computer solve this problem by going into a superposition of all exponentially many possible truth assignments? To answer this question precisely, let us define the black box query model:

Here's the problem: You are given a Boolean function $f: \{1, \ldots, N\} \rightarrow \{0,1\}$, and are promised that for exactly one $a \in \{1, \ldots, N\}, f(a) = 1$. Think of this as a table of size N, where exactly one element has value 1, and all the others are 0. Since we assume f can be computed classically in polynomial time, we can also compute it in superposition:

$$\sum_x \alpha_x |x\rangle|0\rangle \rightarrow \sum_x \alpha_x |x\rangle|f(x)\rangle$$

Another way we can implement f is put the answer register in superposition:

† A case can be determined to be true but there is no efficient algorithm for it

$$\sum_x \alpha_x |x\rangle \left(\frac{|0\rangle - |1\rangle}{2} \right) \rightarrow \sum_x \alpha_x \left(\frac{|x\rangle|f(x)\rangle - |x\rangle|\overline{f(x)}\rangle}{2} \right)$$
$$= \sum_x \alpha_x |x\rangle \left(\frac{|f(x)\rangle - |\overline{f(x)}\rangle}{2} \right) = \sum_x \alpha_x |x\rangle (-1)^x \left(\frac{|0\rangle - |1\rangle}{2} \right)$$

Now, we might as well assume f is a black box or oracle. All we need to do is design an algorithm that finds a: $f(a) = 1$

For the purposes of this discussion, we want to separate the quantum algorithm itself from the function f. We assume that the quantum algorithm is infinitely powerful and focus instead on the number of queries it must make to f. All queries to f occur in superposition; that is, a single query on $\sum_x \alpha_x |x\rangle|0\rangle$ yields the output $\alpha_x |x\rangle|f(x)\rangle$.

Theorem: *In the black box model, any quantum algorithm 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 the function $f \equiv 0$. Let T be the number of queries that A makes to f, and let $\alpha_{x,t}$ be the amplitude with which A queries x at time t (that is, the query at time t is $\sum_t \alpha_{x,t}|x\rangle$).

Now, define the query magnitude of x to be $\sum_x |\alpha_{x,t}|^2$. The expectation value of the query magnitude of x is $E\left(\sum_t |\alpha_{x,t}|^2\right) = \frac{T}{N} \Rightarrow \min_x \left(\sum_t |\alpha_{x,t}|^2\right) \leq \frac{T}{N}$.

Let z be the input at which this minimum occurs; then by the Cauchy-Schwarz inequality:

$$\left(\sum_t |\alpha_{x,t}|\right)^2 \leq T \sum_t |\alpha_{x,t}|^2 = \frac{T^2}{N}$$

Let $|\Phi_t\rangle|$ be the states of A_f Af after the t-th step. Now run the algorithm A on the function g such that $g(z) = 1$ and for all $y \neq z$, $g(y) = 0$. Suppose the final state of A_g is $|\omega_t\rangle$. By the lemma proven in follow: $|\Phi_t\rangle - |\omega_t\rangle = |E_0\rangle + \ldots + |E_{t-1}\rangle|$where $\|E_t\| \leq \sqrt{2}\alpha_{z,t}$ .Using the triangle inequality and the inequality proved above, we have:

$$\||\Phi_t\rangle - |\omega_t\rangle\| \leq \left\|\sum_t E_t\right\| \leq \sqrt{2} \sum |\alpha_{x,t}| \leq T\sqrt{\frac{2}{N}}.$$

This implies that the two states can be distinguished with probability at most $O\left(\frac{T}{\sqrt{N}}\right)$ by any measurement. Thus any quantum algorithm that distinguishes f from g with constant probability of success must make $\Omega(\sqrt{N})$ queries.

Lemma: $|\Phi_t\rangle - |\omega_t\rangle = |E_0\rangle + \ldots + |E_{t-1}\rangle$

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. After the first $t - 1$ steps, both runs are in state $|\Phi_t\rangle$. On the t-th step, one run queries f and the other queries g. The outputs of these queries differ only on the amplitude of the two basis vectors $|z\rangle|0\rangle$ and $|z\rangle|1\rangle$, so overall the output vectors differ by at most $\sqrt{2}\alpha_{z,t}$. Thus, 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 $\|F_t\| \leq \sqrt{2}\alpha_{z,t}$. 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 $|E_t\rangle = U(|F_t\rangle)$. Since unitary transformations preserve length, we know that $\|E_t\| \leq \sqrt{2}\alpha_{z,t}$. Thus, the effect of switching the queried function only on the t-th step can be described by an "error" $|E_t|$ in the final state of the algorithm, where $\|E_t\| \leq \sqrt{2}\alpha_{z,t}$.

We can transform the run Af to Ag by a succession of T changes of the kind described above. Overall, the difference between the final states of Af and Ag is $|E_0\rangle + \ldots + |E_{t-1}\rangle$, where $\|E_t\| \leq \sqrt{2}\alpha_{z,t}$.

Finally, it is useful to consider where this factor of $\sqrt{N}$ comes from. In the worst case, we query z with amplitude $\frac{1}{\sqrt{N}}$ at each time step .The vectors that indicate the differences at each step could all be orthogonal, in which case the total distance is the sum of the squares of each vector's length, which is about N. However, if all vectors are in the same direction, the total distance is the sum of the length of each vector, which is approximately $\sqrt{N}$.

5. Quantum Problem Solving

Now that we have come up with the idea of Quantum black box lets have a more practical look at the concept of solving problems using Quantum Computers. The unique features of a quantum computer pose the following paradox: imagine that the computer is used to prove automatically a mathematical theorem. Classical computer programs that do just that exist and have delivered a number of genuine proofs of nontrivial theorems. But in a quantum computer the details of the reasoning cannot be followed. An attempt to do that

converts the quantum computer into a classical computer. The situation is exactly the same as in the Feynman double-slit Gedankenexperiment The moment you insert an apparatus that can tell you which way the particle goes, the quantum interference image vanishes and you're left with a classical distribution and a classical trajectory. This led some authors, e.g., Williams and Clearwater to ponder a situation whereupon a quantum computer would be able to tell you if your theorem is true or false, but it would not be possible to extract the proof.

This may indeed be the case, but it does not imply that a classical proof of that theorem does not exist or that it cannot be found. One can demonstrate easily that the solution of the Schrödinger wave equation that describes the double-slit Gedankenexperiment represents a congruence of classical trajectories. Whether there is a classical particle that follows those trajectories or not is highly debatable. But this is a matter of interpretation. From a strictly mathematical point of view a congruence of trajectories is there in the solution of the Schrödinger equation.

Translating this result onto our quantum theorem prover tells us that if we were able to somehow measure the whole wave function of the computer as it goes through the proof, and it may be possible to do that by running the job repetitively and measuring distributions, then it should be possible to extract a ``classical trajectory'' from that function that represents a classical proof of our theorem. The wave function will, in fact, deliver a whole congruence of proofs of numerous theorems, of which ours will be but one.

5. Conclusion

Perhaps unsurprisingly, quantum complexity theory now faces many of the same big questions that have faced the classical complexity community for many years. And unfortunately there doesn't appear to be a solution on the horizon. Despite a great deal of effort by people hoping to prove both sides of the issues, virtually nothing concrete is known about the true power of quantum computation. Since we are left then to sift through relativized results that, at best offer mere glimpses at what might be the larger picture, we choose to take a cautious position.
We believe that the present evidence suggests that the realistic gains to be reaped from quantum computing are going to be polynomial, and not exponential. Nevertheless we acknowledge that a quadratic speedup in processing time is significant, and as nano-production continues to develop, we expect that practical implementations of quantum computation will become a reality. Initially the applications may be limited to database systems and other specific applications that require enormous problem sizes (with which to exploit Grover's algorithm), but we believe that in time other examples of modest, but substantial, polynomial gains will be discovered.

References

[1] Zdzisaw Meglicki, "Introduction to Quantum Computing",http://beige.ucs.indiana.edu/M743/M743.pdf ,2005.

[2] Riffel, Polak, "An Introduction to Quantum Computer for Non-Physicists", Palo laboratory,2004.

[3] Dasgupta, Papadimitriou ,and Vazirani, "Algorithms", http://www.cse.ucsd.edu/users/dasgupta/mcgrawhill,2006

[4] Steane, M., "A quantum computer only needs one universe," lanl e-print quant-ph/0003084, Mar.2003.

[5] Robinson, S., "Emerging insights on limitations of quantum computing shape quest for fast algorithms," SIAM News, vol. 36, no. 1, Jan./Feb. 2003.

[6] Fortnow, L., "One complexity theorist's view of quantum computing," Theoretical Computer Science, 292(3):597-610, 2003.

[7] Dam, W. V. "Quantum complexity theory," SQuInT Retreat 2003, lecture, June 2003.

[8] Bennet, C., "Logical reversibility of computation," *IBM J. Res. Develop.*, Vol. 17, 1993, pp. 525-532.

[9] Bennet, C., Bernstein, E., Brassard, G., and Vazirani, U., "Strengths and weaknesses of quantum computation," Special issue on Quantum Computation of the Siam Journal of Computing, Oct. 1999.

[10] Bernstein, E. and Vazirani, U., "Quantum complexity theory," Special issue on Quantum Computation of the Siam Journal of Computing, Oct. 1999.

[11] Berthiaume, A. and Brassard, G., "Oracle quantum computing," Journal of Modern Optics, vol. 41,no. 12, Dec. 1994, pp. 2521-2535.

[12] Boyer, M., Brassard, G., Høyer, P., and Tapp, A., "Tight bounds on quantum searching," arXiv e-print quant-ph/9605034, 1996.

[13] Grover, L., "A framework for fast quantum mechanical algorithms," lanl e-print quant-ph/9711043, Nov. 1998.

[14] Nielsen, M. and Chuang, I, Quantum Computation and Quantum Information. Cambridge University Press, 2004.

[15] Simon, D., "On the power of quantum computation," Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, 1996, pp. 116-123.

[16] Sipser, M., Introduction to the Theory of Computation. PWS Publishing Company, 1997.