A fast quantum mechanical algorithm for database search

Lov K. Grover
3C-404A, AT&T Bell Labs
600 Mountain Avenue
Murray Hill NJ 07974
lkg@mhcnet.att.com

Summary

An unsorted database contains N records, of which just one satisfies a particular property. The problem is to identify that one record. Any classical algorithm, deterministic or probabilistic, will clearly take $O(N)$ steps since on the average it will have to examine a large fraction of the N records. Quantum mechanical systems can do several operations simultaneously due to their wave like properties. This paper gives an $O(\sqrt{N})$ step quantum mechanical algorithm for identifying that record. It is within a constant factor of the fastest possible quantum mechanical algorithm.

1. Introduction

1.0 Background Quantum mechanical computers were proposed in the early 1980's [Benioff80] and shown to be at least as powerful as classical computers - an important but not surprising result, since classical computers, at the deepest level, ultimately follow the laws of quantum mechanics. The description of quantum mechanical computers was formalized in the late 80's and early 90's [Deutsch85][BB92] [BV93] [Yao93] and they were shown to be more powerful than classical computers on various specialized problems. In early 1994, [Shor94] demonstrated that a quantum mechanical computer could efficiently solve a well-known problem for which there was no known efficient algorithm using classical computers. This is the problem of integer factorization, i.e. testing whether or not a given integer, N, is prime, in a time which is a finite power of $O(\log N)$.

STOC'96, Philadelphia PA,USA

This paper applies quantum computing to a mundane problem in information processing and presents an algorithm that is significantly faster than any classical algorithm can be. The problem is this: there is an unsorted database containing N items out of which just one item satisfies a given condition - that one item has to be retrieved. Once an item is examined, it is possible to tell whether or not it satisfies the condition in one step. However, there does not exist any sorting on the database that would aid its selection. The most efficient classical algorithm for this is to examine the items in the database one by one. If an item satisfies the required condition stop; if it does not, keep track of this item so that it is not examined again. It is easily seen that this algorithm will need to look at an average of $\frac{N}{2}$ items before finding the desired one.

1.1 Search Problems in Computer Science

Even in theoretical computer science, the typical problem can be looked at as that of examining a number of different possibilities to see which, if any, of them satisfy a given condition. This is analogous to the search problem stated in the summary above, except that usually there exists some structure to the problem, i.e some sorting does exist on the database. Most interesting problems are concerned with the effect of this structure on the speed of the algorithm. For example the SAT problem of NP-completeness asks whether it is possible to find any combination of n binary variables that satisfies a certain set of clauses C, in time polynomial in n. In this case there are $N=2^n$ possible combinations which have to be searched for any that satisfy the specified property and the question is whether we can do that in a time which is a finite power of $O(\log N)$, i.e. $O(n^k)$. Thus if it were possible to reduce the number of steps to a finite power of $O(\log N)$ (instead of $O(\sqrt{N})$ as in this paper), it would yield a polynomial time algorithm for NP-complete problems.

In view of the fundamental nature of the search problem in both theoretical and applied computer science, it is natural to ask - how fast can the basic identifi-

cation problem be solved without assuming anything about the structure of the problem? It is generally assumed that this limit is $O(N)$ since there are N items to be examined and a classical algorithm will clearly take $O(N)$ steps. However, quantum mechanical systems can sometimes be counter-intuitive [BV93] [Shor94] [Simon94]. This paper presents an $O(\sqrt{N})$ step algorithm for the above problem.

There is a matching lower bound on how fast the desired item can be identified. [BBBV94] show in their paper that in order to identify the desired element, without any information about the structure of the database, a quantum mechanical system will need at least $\Omega(\sqrt{N})$ steps. Since the number of steps required by the algorithm of this paper is $O(\sqrt{N})$, it is within a constant factor of the fastest possible quantum mechanical algorithm.

1.2 Quantum mechanical algorithms A good starting point to think of quantum mechanical algorithms is probabilistic algorithms [BV93] (e.g. simulated annealing). In these algorithms, instead of having the system in a specified state, it is in a distribution over various states with a certain probability of being in each state. At each step, there is a certain probability of making a transition from one state to another. The evolution of the system is obtained by premultiplying this probability vector (that describes the distribution of probabilities over various states) by a state transition matrix. Knowing the initial distribution and the state transition matrix, it is possible in principle to calculate the distribution at any instant in time.

Just like classical probabilistic algorithms, quantum mechanical algorithms work with a probability distribution over various states. However, unlike classical systems, the probability vector does not completely describe the system. In order to completely describe the system we need the *amplitude* in each state which is a complex number. The evolution of the system is obtained by premultiplying this amplitude vector (that describes the distribution of amplitudes over various states) by a transition matrix, the entries of which are complex in general. The probabilities in any state are given by the square of the absolute values of the amplitude in that state. It can be shown that in order to conserve probabilities, the state transition matrix has to be unitary [BV93].

The machinery of quantum mechanical algorithms is illustrated by discussing the three operations that are needed in the algorithm of this paper. The first is the creation of a configuration in which the amplitude of the system being in any of the 2^n basic states of the system is equal; the second is the Fourier transformation operation and the third the selective rotation of different states.

A basic operation in quantum computing is that of a "fair coin flip" performed on a single bit whose states are 0 and 1 [Simon93]. This operation is represented by the following matrix: $M = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. A bit in the state 0 is transformed into a superposition in the two states: $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$. Similarly a bit in the state 1 is transformed into $\left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right)$, i.e. the magnitude of the amplitude in each state is $\frac{1}{\sqrt{2}}$ but the *phase* of the amplitude in the state 1 is inverted. The phase does not have an analog in classical probabilistic algorithms. It comes about in quantum mechanics since the amplitudes are in general complex. As mentioned previously, the probabilities are determined by the square of the absolute value of the amplitude in each state; hence the probabilites of the two states in both cases (i.e. when the amplitudes are $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$ and when they are $\left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right)$) are equal. And yet the distributions have very different properties. For example, applying M to the distribution $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$ results in the configuration being in the state 0; applying M to the distribution $\left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right)$ results in the system being in the state 1. Two distributions that had the same probability distributions over all states, now have orthogonal distributions! This illustrates the point that, in quantum mechanics, the complete description of the system requires the amplitudes over all states, just the probabilities is not enough.

In a system in which the states are described by n bits (it has 2^n possible states) we can perform the transformation M on each bit independently in sequence thus changing the state of the system. The state transition matrix representing this operation will be of dimension $2^n \times 2^n$. In case the initial configuration was the configuration with all n bits in the first state, the resultant configuration will have an identical amplitude of $2^{-\frac{n}{2}}$ in each of the 2^n states. This is a way of creating a distribution with the same amplitude in all 2^n states.

Next consider the case when the starting state is another one of the 2^n states, i.e. a state described by an n bit binary string with some 0s and some 1s. The result of performing the transformation M on each bit will be a superposition of states described by all possible n bit binary strings with amplitude of each state having a magnitude equal to $2^{-\frac{n}{2}}$ and sign either + or -. To deduce the sign, observe that from the definition of the matrix M, i.e. $M = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, the phase of the resulting configuration is changed when a bit that was previously a 1 remains a 1 after the transformation is performed. Hence if $\bar{x}$ be the n-bit binary string describing the starting state and $\bar{y}$ the n-bit binary string describing the resulting string, the sign of the amplitude of $\bar{y}$ is determined by the parity of the bitwise dot product of $\bar{x}$ and $\bar{y}$, i.e. $(-1)^{\bar{x}\cdot\bar{y}}$. This transformation is referred to as the Fourier transformation [DJ92]. This operation is one of the things that makes quantum mechanical algorithms more powerful than classical algorithms and forms the basis for most significant quantum mechanical algorithms.

The third transformation that we will need is the selective rotation of the phase of the amplitude in certain states. The transformation describing this for a 4 state system is of the form: $\begin{bmatrix} e^{j\phi_1} & 0 & 0 & 0 \\ 0 & e^{j\phi_2} & 0 & 0 \\ 0 & 0 & e^{j\phi_3} & 0 \\ 0 & 0 & 0 & e^{j\phi_4} \end{bmatrix}$ where $j = \sqrt{-1}$ and $\phi_1, \phi_2, \phi_3, \phi_4$ are arbitrary real numbers. Note that, unlike the Fourier transformation and other state transition matrices, the probability in each state stays the same since the square of the absolute value of the amplitude in each state stays the same.

2. The Abstracted Problem

Let a system have $N = 2^n$ states which are labelled $S_1, S_2, \ldots S_N$. These 2^n states are represented as n bit strings. Let there be a unique state, say S_v, that satisfies the condition $C(S_v) = 1$, whereas for all other states S, $C(S) = 0$ (assume that for any state S, the condition $C(S)$ can be evaluated in unit time). The problem is to identify the state S_v.

3. Algorithm

(i) Initialize the system to the distribution: $\left(\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}} \ldots \frac{1}{\sqrt{N}}\right)$, i.e. there is the same amplitude to be in each of the N states. It is possible to obtain this distribution in $O(\log N)$ steps as discussed in section 1.2.

(ii) Repeat the following unitary operations $O(\sqrt{N})$ times:

(a) Let the system be in any state S:
in case $C(S) = 1$, rotate the phase by π radians,
in case $C(S) = 0$, leave the system unaltered.

(b) Apply the diffusion transform D which is defined by the matrix D as follows:

$D_{ij} = \frac{2}{N}$ if $i \neq j$ & $D_{ii} = -1 + \frac{2}{N}$.

This diffusion transform, D, can be implemented as $D = FRF$, where F the Fourier Transform Matrix and R the rotation matrix are defined as follows:

$R_{ij} = 0$ if $i \neq j$;

$R_{ii} = 1$ if $i = 0$; $R_{ii} = -1$ if $i \neq 0$.

As discussed in section 1.2, the Fourier transformation matrix F is defined as:

$F_{ij} = 2^{-n/2}(-1)^{\dot{i}\cdot\dot{j}}$, where $\dot{i}$ is the binary representation of i, and $\dot{i}\cdot\dot{j}$ denotes the bitwise dot product of the two n bit strings $\dot{i}$ and $\dot{j}$.

(iii) Sample the resulting state. In case there is a unique state S_v such that $C(S_v) = 1$, the final state is S_v with a probability of at least $\frac{1}{2}$.

4. Outline of rest of paper

The following sections prove that the system discussed in section 3 is indeed a valid quantum mechanical system and that it converges to the desired state with a probability O(1). The proofs are divided into two parts. First, it is proved that the system is a valid quantum mechanical system (theorems 1 & 2) and, second, that it converges to the desired state (theorems 3, 4 & 5) . In order to prove that it is a valid quantum mechanical system we need to prove that it is unitary and that it can be implemented as a product of local transition matrices.

The diffusion transform D is defined by the matrix D as follows:

(4.0) $D_{ij} = \frac{2}{N}$, if $i \neq j$ & $D_{ii} = -1 + \frac{2}{N}$.

Theorem 1 - The matrix D is unitary.

The way D is presented above, it is not a local transition matrix since there are transitions from each state to all N states. Using the Fourier transformation matrix as defined in section 3, it can be implemented as a product of three unitary transformations as $D = FRF$, each of F & R is a local transition matrix. R as defined in theorem 2 is a phase rotation matrix and is clearly local. F when implemented as in section 1.2 is a local transition matrix on each bit.

Theorem 2 - D can be expressed as $D = FRF$, where F, the Fourier Transform Matrix and R, the rotation matrix are defined as follows

$R_{ij} = 0$ if $i \neq j$,

$R_{ii} = 1$ if $i = 0$, $R_{ii} = -1$ if $i \neq 0$.

$F_{ij} = 2^{-n/2}(-1)^{i \cdot j}$.

Next it is proved that the algorithm indeed converges to the desired state. In order to follow the evolution of the state when the unitary transformation D of step (ii)(b) of the algorithm is applied, we prove:

Theorem 3 - Let the state vector be as follows - for any one state the amplitude is k, for each of the remaining *(N-1)* states the amplitude is l. Then after applying the diffusion transform D, the amplitude in the one state is $k_1 = \left(\frac{2}{N} - 1\right)k + 2\frac{(N-1)}{N}l$ and the amplitude in each of the remaining *(N-1)* states is $l_1 = \frac{2}{N}k + \frac{(N-2)}{N}l$.

Using this we prove a simple relation between the amplitudes k & l.

Corollary 3.1 - Let the state vector be as follows - for any one state the amplitude is k, for each of the remaining *(N-1)* states the amplitude is l. Let k and l be real numbers (in general the amplitudes can be complex). Let k be negative and l be positive such that $\left|\frac{k}{l}\right| < \sqrt{N}$. Then after applying the diffusion transform both k and l are positive numbers.

As is well known, in a unitary transformation the total probability is conserved - this is proved for the particular case of the diffusion transformation by using theorem 3.

Corollary 3.2 - Let the state vector be as follows - for the state that satisfies C(S) = 1, the amplitude is k, for each of the remaining *(N-1)* states the amplitude is l. Then if after applying the diffusion transformation D, the new amplitudes are respectively k_1 and l_1 as derived in theorem 3, then

$$k_1^2 + (N-1)l_1^2 = k^2 + (N-1)l^2$$

k & l vary with each iteration of the algorithm in section 3. Therefore if we are given k at any step of the algorithm, by using the conservation result of corollary 3.2 we can immediately obtain $|l|$, the sign of l can be deduced by corollary 3.1 - it is thus enough to keep track of one variable k or l, not both. Using this we prove -

Theorem 4 - Let the state vector before step (a) of the algorithm be as follows - for the one state that satisfies $C(S) = 1$, the amplitude is k, for each of the remaining *(N-1)* states the amplitude is l such that $\left(0 < k < \frac{1}{\sqrt{2}}\right)$ and $l > 0$. The change in k (Δk) after steps (a) and (b) of the algorithm is lower bounded by $\Delta k > \frac{1}{2\sqrt{N}}$. Also after steps (a) and (b), $l > 0$.

Using this it immediately follows that there exists a number M less than $\sqrt{2N}$, such that in M repetitions of the loop in step (ii), k will become $\frac{1}{\sqrt{2}}$. Since the probability of the system being found in any particular state is proportional to the square of the amplitude, it follows that the probability of the system being in the desired state when k is $\frac{1}{\sqrt{2}}$, is $k^2 = \frac{1}{2}$. Therefore if the system is now sampled, it will be in the desired state with a probability of $\frac{1}{2}$. Finally it is proved that once we can design an experiment with a finite success probability of ε, in $O\left(\frac{1}{\varepsilon}\right)$ repetitions of the experiment it is possible to have a probability of success very close to unity.

Theorem 5 - Let the probability of a particular outcome in an experiment be ε. Then if the experiment be repeated $\frac{\kappa}{\varepsilon}$ times, the probability that the outcome does not happen even once is less than or equal to $e^{-\kappa}$.

Section 6 quotes the argument from [BBBV94] that it is not possible to identify the desired record in less than $\Omega(\sqrt{N})$ steps.

5. Proofs

As mentioned before (4.0), the diffusion transform D is defined by the matrix D as follows:

(5.0) $D_{ij} = \frac{2}{N}$, if $i \neq j$ & $D_{ii} = -1 + \frac{2}{N}$.

Theorem 1- The matrix D is unitary.

Proof - For a matrix to be unitary it is necessary and sufficient to prove that the columns are orthonormal. The magnitude of each column vector is $\sum_{i=1}^{n} D_{ij}^2, j = 1, 2 \ldots n$. Substituting from (5.0), it follows that this is $\left(1 - \frac{2}{N}\right)^2 + (N-1)\frac{4}{N^2}$ which is equal to 1. The dot product of two distinct column vectors is $\sum_{i=1}^{n} D_{ik} D_{ij}$. This is $N\frac{4}{N^2} - \frac{4}{N} = 0$.

Theorem 2 - D can be expressed as $D = FRF$, where F, the Fourier Transform Matrix and R, the rotation matrix, are defined as follows

$R_{ij} = 0$ if $i \neq j$,

$R_{ii} = 1$ if $i = 0$, $R_{ii} = -1$ if $i \neq 0$.

$F_{ij} = 2^{-n/2}(-1)^{\dot{i} \cdot \dot{j}}$.

Proof - We evaluate FRF and show that it is equal to D. As discussed in section 3, $F_{ij} = 2^{-n/2}(-1)^{\dot{i} \cdot \dot{j}}$, where $\dot{i}$ is the binary representation of i, and $\dot{i} \cdot \dot{j}$ denotes the bitwise dot product of the two n bit strings $\dot{i}$ and $\dot{j}$. R can be written as $R = R_1 + R_2$ where $R_1 = -I$, I is the identity matrix and $R_{2,00} = 2$, $R_{2,ij} = 0$ if $i \neq 0, j \neq 0$. By observing that $MM = I$ where M is the matrix defined in section 1.2, it is easily proved that $FF=I$ and hence $D_1 = FR_1F = -I$. We next evaluate $D_2 = FR_2F$. By standard matrix multiplication: $D_{2,ad} = \sum_{bc} F_{ab} R_{2,bc} F_{cd}$. Using the definition of R_2 and the fact $N = 2^n$, it follows that $D_{2,ad} = 2F_{a0}F_{0d} = \frac{2}{2^n}(-1)^{\bar{a} \cdot \bar{0} + \bar{0} \cdot \bar{a}} = \frac{2}{N}$. Thus all elements of the matrix D_2 equal $\frac{2}{N}$, the sum of the two matrices D_1 and D_2 gives D.

Theorem 3 - Let the state vector be as follows - for any one state the amplitude is k_1, for each of the remaining *(N-1)* states the amplitude is l_1. Then after applying the diffusion transform D, the amplitude in the one state is $k_2 = \left(\frac{2}{N} - 1\right)k_1 + 2\frac{(N-1)}{N}l_1$ and the amplitude in each of the remaining *(N-1)* states is $l_2 = \frac{2}{N}k_1 + \frac{(N-2)}{N}l_1$.

Proof -Using the definition of the diffusion transform (5.0) (at the beginning of this section), it follows that

$$k_2 = \left(\frac{2}{N} - 1\right)k_1 + 2\frac{(N-1)}{N}l_1$$

$$l_2 = \left(\frac{2}{N} - 1\right)l_1 + \frac{2}{N}k_1 + \frac{2(N-2)}{N}l_1$$

Therefore:

$$l_2 = \frac{2}{N}k_1 + \frac{(N-2)}{N}l_1$$

Corollary 3.1 - Let the state vector be as follows - for any one state the amplitude is k, for each of the remaining *(N-1)* states the amplitude is l. Let k and l be real numbers (in general the amplitudes can be complex). Let k be negative and l be positive and $\left|\frac{k}{l}\right| < \sqrt{N}$. Then after applying the diffusion transform both k_1 and l_1 are positive numbers.

Proof - From theorem 3,

$k_1 = \left(\frac{2}{N} - 1\right)k + 2\frac{(N-1)}{N}l$. Assuming $N > 2$, it follows that $\left(\frac{2}{N} - 1\right)$ is negative; by assumption k is negative and $2\frac{(N-1)}{N}l$ is positive and hence $k_1 > 0$. Similarly it follows that since by theorem 3,

$l_1 = \frac{2}{N}k + \frac{(N-2)}{N}l$, and so if the condition

$\left|\frac{k}{l}\right| < \frac{(N-2)}{2}$ is satisfied, then $l_1 > 0$. If $\left|\frac{k}{l}\right| < \sqrt{N}$, then for $N \geq 9$ the condition $\left|\frac{k}{l}\right| < \frac{(N-2)}{2}$ is satisfied and $l_1 > 0$.

Corollary 3.2 - Let the state vector be as follows - for the state that satisfies C(S) = 1, the amplitude is k, for each of the remaining $(N-1)$ states the amplitude is l. Then if after applying the diffusion transformation D, the new amplitudes are respectively k_1 and l_1 as derived in theorem 3, then

$$k_1^2 + (N-1)\,l_1^2 = k^2 + (N-1)\,l^2.$$

Proof - Using theorem 3 it follows that

$$k_1^2 = \frac{(N-2)^2}{N^2}k^2 + 4\frac{(N-1)^2}{N^2}l^2 - \frac{4(N-2)(N-1)}{N^2}kl$$

Similarly

$$(N-1)\,l_1^2 = \frac{4(N-1)^2}{N^2}k^2 + \frac{(N-2)^2}{N^2}(N-1)\,l^2 + \frac{4(N-2)(N-1)}{N^2}kl.$$

Adding the previous two equations the corollary follows.

Theorem 4 - Let the state vector before step (a) of the algorithm be as follows - for the one state that satisfies $C(S) = 1$, the amplitude is k, for each of the remaining $(N-1)$ states the amplitude is l such that $\left(0 < k < \frac{1}{\sqrt{2}}\right)$ and $l > 0$. The change in k (Δk) after steps (a) and (b) of the algorithm is lower bounded by $\Delta k > \frac{1}{2\sqrt{N}}$. Also after steps (a) and (b), $l > 0$.

Proof - Denote the initial amplitudes by k and l, the amplitudes after the phase inversion (step (a)) by k_1 and l_1 and after the diffusion transform (step (b)) by k_2 and l_2. Using theorem 3, it follows that:

$k_2 = \left(1 - \frac{2}{N}\right)k + 2\left(1 - \frac{1}{N}\right)l$. Therefore

(5.1) $\Delta k = k_2 - k = -\frac{2k}{N} + 2\left(1 - \frac{1}{N}\right)l.$

Since $\left(0 < k < \frac{1}{\sqrt{2}}\right)$, it follows from corollary 3.2 that $|l| > \frac{1}{\sqrt{2N}}$ and since by the assumption in this theorem, l is positive, it follows that $l > \frac{1}{\sqrt{2N}}$. Therefore by (5.1), assuming non-trivial N, it follows that $\Delta k > \frac{1}{2\sqrt{N}}$.

In order to prove $l_2 > 0$, observe that after the phase inversion (step (a)), $k_1 < 0$ & $l_1 > 0$. Furthermore it follows from the facts $\left(0 < k < \frac{1}{\sqrt{2}}\right)$ & $|l| > \frac{1}{\sqrt{2N}}$ (discussed in the previous paragraph) that $\left|\frac{k_1}{l_1}\right| < \sqrt{N}$. Therefore by corollary 3.1, l_2 is positive.

In order to prove theorem 5, we first prove a simple lemma:

Lemma 5 - $(1-\gamma)^{\kappa/\gamma} < e^{-\kappa}$ provided $\kappa, \gamma > 0$.

Proof - Observe that $(1-\gamma) < e^{-\gamma}$, if $\gamma > 0$. Now raise both sides to the $\frac{\kappa}{\gamma}$ power.

Theorem 5 - Let the probability of a particular outcome of an experiment be ε. Then if the experiment be repeated $\frac{\kappa}{\varepsilon}$ times, the probability that the outcome does not happen even once is less than or equal to $e^{-\kappa}$.

Proof - The probability that the outcome does not happen even once in $\frac{\kappa}{\varepsilon}$ experiments is $(1-\varepsilon)^{\kappa/\varepsilon}$. By lemma 5 this is less than or equal to $e^{-\kappa}$.

6. How fast is it possible to find the desired element?

There is a matching lower bound from the paper [BBBV94] that suggests that it is not possible to identify the desired element in fewer than $\Omega(\sqrt{N})$ steps. This result states that any quantum mechanical algorithm running for T steps is only sensitive to $O(T^2)$ queries (i.e. if there are more possible queries, then the answer to at least one can be flipped without affecting the behavior of the algorithm). So in order to correctly decide the answer which is sensitive

to N queries will take a running time of $T = \Omega(\sqrt{N})$. To see this, assume that $C(S) = 0$ for all states and the algorithm returns the right result. Then, by [BBBV94] if $T < \Omega(\sqrt{N})$ the answer to at least one of the queries about $C(S)$ for some S can be flipped without affecting the result, thus giving an incorrect result this time.

7. Other observations

1. As mentioned in the introduction, the algorithm of this paper does not use any knowledge about the problem. There exist fast quantum mechanical algorithms that make use of the structure of the problem at hand. For example Shor's primality testing algorithm [Shor94] for a number N can be viewed as a search of the $(N-1)$ numbers smaller than N to see if any of them is a factor of N. This algorithm, by cleverly making use of some features of the structure of the problem, is the only known algorithm that solves the problem in $O(\log N)$ steps. It might be possible to combine the search scheme of this paper with [Shor94] and other quantum mechanical algorithms to design even faster algorithms. Alternatively, it might be possible to combine it with efficient database search algorithms that make use of specific properties of the database.

Associative memories were one of the first applications of synthetic neural networks. There, the motivation was to use as much information as possible about the problem so as to encode and retrieve it most efficiently. It is interesting to speculate about computational properties and limits of *quantum mechanical neural* systems that might achieve both efficient encoding as well as efficient search. [Penrose89] [Penrose94] mention that certain neural sensing systems in the retina and brain operate in a quantum mechanical framework. Quantum mechanical computation is discussed as one possible aspect distinguishing a brain from a traditional computer. Noise is a factor which limits quantum mechanical computation in the brain and elsewhere. Very recent results [Shor95] have shown the existence of quantum mechanical error correcting codes which enable transmission of data even in the presence of noise; design of error free quantum mechanical gates still remains an unsolved problem.

2. The algorithm as discussed here assumes a unique state that satisfies the desired condition. It can be easily modified to take care of the case when there are multiple states satisfying the condition $C(S) = 1$ and it is required to return one of these. Two ways of achieving this are:
(i) The first possibility would be to repeat the experiment so that it checks for a range of degeneracy, i.e. redesign the experiment so that it checks for the degeneracy of the solution being in the range $(k, k+1, \ldots 2k)$ for various k. Then within $\log N$ repetitions of this procedure, one can ascertain whether or not there exists at least one state out of the N states that satisfies the condition.
(ii) The other possibility is to slightly perturb the problem in a random fashion as discussed in [MVV87] so that with a high probability the degeneracy is removed. There is also a scheme discussed in [VV86] by which it is possible to modify any algorithm that solves an NP-search problem with a unique solution and use it to solve an NP-search problem in general.

7. Acknowledgments

The author gratefully acknowledges Peter Shor's and Ethan Bernstein's help in going through this paper in detail and making several extremely constructive suggestions.

8. References

[BBBV94] C.H. Bennett, E. Bernstein, G. Brassard and U.Vazirani, *Strengths and weaknesses of quantum computing*, preprint, 1994.

[BB92] A. Berthiaume and G. Brassard, *The quantum challenge to structural complexity theory*, Proceedings 7th IEEE Conference on Structure in Complexity Theory, 1992, pp. 132-137.

[Benioff80] P. Benioff, *The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines*, Journal of Statistical Physics, 22, pp. 563-591.

[BV93] E. Bernstein and U. Vazirani, *Quantum Complexity Theory*, Proceedings 25th ACM Symposium on Theory of Computing, 1993, pp. 11-20.

[Deutsch85] D. Deutsch, *Quantum Theory, the Church-Turing principle and the universal quantum computer*, Proceedings Royal Society London Ser. A, 400, pp. 96-117.

[DJ92] D. Deutsch and R. Jozsa, *Rapid solution of problems by quantum computation*, Proceedings Royal Society of London, A400, pp. 73-90.

[MVV87] K. Mulmuley, U. Vazirani and V. Vazirani, *Matching is as easy as matrix inversion*, Combinatorica, 7 (1987), pp. 105-131.

[Penrose89] R. Penrose, *The Emperor's New Mind, Concerning Computers, Minds and the Laws of Physics*. Oxford University Press, 1989, pp. 400-402.

[Penrose94] R. Penrose, *Shadows of the mind,*. Oxford University Press, 1994, pp. 348-388.

[Shor94] P. W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring,* Proceedings, 35th Annual Symposium on Fundamentals of Computer Science (FOCS), 1994, pp. 124-134.

[Shor95] P. W. Shor, *Scheme for reducing decoherence in quantum computer memory,* Phys. Rev. A, Vol. 52, October 1995, pp. 2493-2496.

[Simon94] D. Simon, *On the power of quantum computation,* Proceedings, 35th Annual Symposium on Fundamentals of Computer Science (FOCS), 1994, pp.116-123.

[VV86] L. G. Valiant and V. Vazirani, *NP is as easy as detecting unique solutions*, Theoretical Computer Science 47, 1986, pp. 85-93.

[Yao93] A. Yao, *Quantum circuit complexity*, Proceedings 34th Annual Symposium on Foundations of Computer Science (FOCS), 1993, pp. 352-361.