Mathlab Implementation of Quantum Computation in Searching an Unstructured Database

I.O. Awoyelu and P. Okoh
Department of Computer Science & Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria
E-mail: iawoyelu@oauife.edu.ng

Keywords: qubit, quantum algorithm, classical computing, quantum computing

Received: August 15, 2011

In the classical model of a computer, the most fundamental building block, the bit, can only exist in one of two distinct states, a 0 or a 1. Computations are carried out by logic gates that act on these bits to produce other bits. Unless there is duplicate (parallel) hardware, only one problem instance (i.e. input data set) can be handled at a time. In this classical computing, increasing the number of bits increases the complexity of the problem and the time necessary to arrive at a solution.
A quantum algorithm consists of a sequence of operations on a register, to transform it into a state which, when measured, yields the desired result with high probability. An n-bit quantum register can store an exponential amount of information.
This paper aims at taking advantage of the superiority of quantum computing over classical computing to solve the problem of searching unstructured databases for a particular item or more than one item in good time. The general aim of this work is to establish the correctness and optimality of Grover's quantum database search algorithm compared against classical database search methods in order to investigate the superiority or otherwise of quantum computing over classical computing. This is followed by a simulation of the algorithm using a classical computer, namely through functions that are present in MATLAB, referred to as "Quantum Computing Functions".

Povzetek: Članek predstavlja implementacijo kvantnega iskanja v nestrukturiranih bazah.

1 Introduction

The bit is the fundamental unit of storage in a classical computer; similarly, the basis of quantum computation is a qubit. The qubit is similar to a bit in that when measured, its value will be either 0 or 1. It differs primarily in what it is doing when it is not being measured. In particular, a qubit can exist in any superposition of the 0 and 1 state simultaneously. When a qubit in such a state is measured the superposition will be destroyed. It will be found to be uniquely in the 0 or 1 state with some probability for each, determined by the particulars of the superposition prior to the measurement (Williams and Clearwater, 1998).

The renowned scientist Moore's empirical law, which states that "computing power doubles approximately every 18 months" is believed by just about everyone that is into computing (Williams and Clearwater, 1998). Computer components have been steadily decreasing in size and increasing in speed, tending to follow this prediction. However, what is the fate of this trend in real life? A little extrapolation would suggest that within about 10 years the size of a transistor logic gate element will be only a few atoms. Consequently, computer power will soon reach a limit, unless another approach for computing can be developed. Quantum computing is one possible approach.

A quantum algorithm consists of a sequence of operations on a register, to transform it into a state which, when measured, yields the desired result with high probability. An n-bit quantum register can store an exponential amount of information. The register as a whole can be in an arbitrary superposition of the 2^n base states which it can be measured to be in. While in this superposition, and computation applied to the register will be applied to each component of the superposition, this behavior follows from the linearity of operators on quantum mechanical systems. This behavior, called "quantum parallelism" is the basis for most quantum algorithms. In 1982, the Nobel prize-winning physicist, Richard Feynman, thought up the idea of a 'quantum computer', a computer that uses the effects of quantum mechanics to its advantage (Deutsch and Jozsa, 1992).

In the classical model of a computer, the most fundamental building block, the bit, can only exist in one of two distinct states, a 0 or a 1. Computations are carried out by logic gates that act on these bits to produce other bits. Unless there is duplicate (parallel) hardware, only one problem instance (i.e. input data set) can be handled at a time. In this classical computing, increasing the number of bits increases the complexity of the problem and the time necessary to arrive at a solution.

In a quantum computer, the rules are changed. Not only can a 'quantum bit', usually referred to as a 'qubit',

exist in the classical 0 and 1 states, it can also be in a coherent superposition and all linear combinations of both.

The number of possibilities grows exponentially with the number of qubits. For example, for a 2-qubit system, there are all possible superpositions of the states 00, 01, 10, and 11, including entangled states of the form (01 ±10). If there are N qubits, the vector space required to describe their states has dimension 2^N . Calculations are carried out on vectors by quantum gates that apply unitary transformations to these vectors to produce other vectors. Since quantum computers can process superpositions, they can (at least for some problems) be viewed as devices that can process all possible inputs simultaneously.

When a qubit is in this state, it can be thought of as existing in two universes: as a 0 in one universe and as a 1 in the other. An operation on such a qubit effectively acts on both values at the same time. The significant point being that by performing the single operation on the qubit, we have performed the operation on two different values. Likewise, a two-qubit system would perform the operation on 4 values, and a three-qubit system on eight. Increasing the number of qubits therefore exponentially increases the 'quantum parallelism' we can obtain with the system (Williams and Clearwater, 1998). This is a great leap from the classical computing we are familiar with in which increasing the number of bits increases the complexity of the problem and the time necessary to arrive at a solution.

With the correct type of algorithm, it is possible to use this parallelism to solve certain problems in a fraction of the time taken by a classical computer. A quantum computer would consist of many qubit gates with entangled states. These gates could be addressed in parallel by unitary transformations, which must be carried out reversibly, implying no loss of energy in a gate operation. Quantum computers are “wired” so that they can do many calculations at the same time. This is known as “quantum parallelism” and represents the power of a quantum computer.

Several systems have been proposed for quantum computing including photons in nonlinear optical systems, trapped ions, electron and nuclear spins, quantum dots, and Josephson junctions (Grover, 2000). There are advantages and disadvantages to all these approaches. Some, such as those employing light or trapped ions, have demonstrated that they can provide individual qubits of excellent quality. But, it is not yet known if they can be scaled up to produce systems with many qubits and many possible quantum gate operations.

There is nothing a quantum computer can do that cannot also be done by an ordinary computer. However, for some problems, a quantum computer may be many orders of magnitude faster. There are presently two important problems involving commerce and security for which a quantum computer is believed to be superior. These are finding the factors of a large number and searching an unstructured database. A quantum computer would also be superior for simulating the behavior of quantum systems, and thus have enormous implications for physics.

This paper aims at taking advantage of the superiority of quantum computing over classical computing to solve the problem of searching unstructured databases for a particular item or more than one item in good time. The general aim of this work is to establish the correctness and optimality of Grover’s quantum database search algorithm compared against classical database search methods in order to investigate the superiority or otherwise of quantum computing over classical computing. This is followed by a simulation of the algorithm using a classical computer, namely through functions that are present in MATLAB, referred to as “Quantum Computing Functions”.

2 Existing Works in Quantum Computing

In the early 1980's, physicist Richard Feynman observed that no classical computer could simulate quantum mechanical systems without incurring exponential slowdown (Williams and Clearwater, 1998). At the same time, it seems reasonable that a computer which behaves in a manner consistent with quantum mechanics could, in principle, simulate such systems without exponential slowdown.

For many years the study of quantum computing was primarily an academic curiosity. Shor (1994) developed a polynomial time algorithm for factoring large integers. According to Williams and Clearwater (1998), it is not known if there is a classical algorithm for factoring large integers efficiently, but the best algorithms published thus far are super-polynomial. This algorithm coupled with the prominence of cryptographic systems based on factoring large integers fueled study of quantum computation, both from an algorithmic and a manufacturing point of view. Grover (1996) provided an $O(\sqrt{n})$ time algorithm for finding a single marked element in an unsorted database of n elements. The best possible classical algorithm would run in O (n) time. This search problem was not the first problem for which a quantum computer was shown to be better than any possible classical computer, but it was the first problem of real utility found where a quantum computer outperforms a classical computer in an asymptotic sense.

While Shor's algorithm may be of more immediate utility, Grover's algorithm seems more interesting in a theoretical sense, as it highlights an area of fundamental superiority in quantum computation.

2.1 Grover's Algorithm

Assuming there is a system with $N = 2^n$ states labeled S_1, S_2,…, S_N. These 2^n states are represented by *n* bit strings. Assuming there is a unique marked element S_m that satisfies a condition $C(S_m) = 1$, and for all other states $C(S) = 0$. Suppose that C can be evaluated in unit time. The task is to devise an algorithm which minimizes the number of evaluations of C.

The idea of Grover's algorithm is to place the register

in an equal superposition of all states, and then selectively invert the phase of the marked state, and then perform an inversion about average operation a number of times. The selective inversion of the marked state followed by the inversion about average steps has the effect of increasing the amplitude of the marked state by $O(1/\sqrt{NO(1)})$. Therefore after $O(\sqrt{N})$ operations the probability of measuring the marked state approaches 1 (Grover, 1996).

Grover's algorithm is as follows:

- Prepare a quantum register to be normalized and uniquely in the first state. Then place the register in an equal superposition of all states $\left(\frac{1}{\sqrt{N}},\frac{1}{\sqrt{N}},\ldots,\frac{1}{\sqrt{N}}\right)$ by applying the Walsh-Hadamard operator W. This means simply the state vector will be in an equal superposition of each state.
- Repeat $O(\sqrt{N})$ times the following two steps (the precise number of iterations is important, and discussed below):
  - Let the system be in any state S. If C(S) = 1, rotate the phase by π radians, else leave system unaltered. It is worth noting that this operation has no classical analog. One cannot observe the state of the quantum register, doing so would collapse the superposition. The selective phase rotation gate would be a quantum mechanical operator which would rotate only the amplitude proportional to the marked state within the superposition.
  - Apply the inversion about average operator A, whose matrix representation is: $A_{ij} = 2/N$ if i ≠ j and $A_{ij} = -1 + 2/N$ to the quantum register.
- Measure the quantum register. The measurement will yield the n bit label of the marked state $C(S_M)$ = 1 with probability at least 1/2 (Grover, 1996).

This Grover's algorithm flow chart is as shown in Figure 1.

3 Proposed System

The proposed system is concerned with the simulation of Grover's Algorithm using MATLAB. Quantum computing uses unitary operators acting on discrete state vectors. Matlab is a well-known (classical) matrix computing environment, which makes it well suited for simulating quantum algorithms.

Appendix A contains the Matlab commands to simulate Grover's algorithm using six qubits. The number of database elements is 2^6 = 64. The desired element is randomly generated from among the 64 elements using Matlab's *rand* function and it is the 53rd element. The quantum gates are defined using Matlab's *eye* function. The optimal number of iterations is determined by $\frac{\pi}{4}\sqrt{n}$ as proposed by Grover. Figure 2 shows the probability dynamics generated by the *plot* function in Matlab while Figure 3 shows the result distribution.

Grover demonstrated that quantum computers can perform database search faster than their classical counterparts. In this simple example of Grover's algorithm, a haystack function is used to represent the database. We are searching for a needle in the haystack, i.e. there is one element of the database that we require.

Grover's algorithm works by iteratively applying the inversion about the average operator to the current state. Each iteration amplifies the probability of a measurement collapsing the state to the correct needle value. Grover showed that performing a measurement after $\frac{\pi}{4}\sqrt{n}$ iterations is highly likely to give the correct result.

Appendix B contains the Matlab commands needed to find the needle in a haystack, i.e. to find a particular element among the elements of a database. The Walsh-Hadamard transformation, operator to rotate phase and inversion about average transformation were achieved through matrices of zeros and ones available as Matlab commands. At the end of the program, the output was set to be a movie-like display of the different stages of the iteration in a 3-dimensional plane comprising the axes Amplitude, States and Time. This was achieved by using Matlab's function *movie*. The last stage of the iteration process is shown by Figure 4.

3.1 Open Questions

There are several open questions that prop up from Grover's search algorithm proposal. Foremost among these is how many times exactly we should iterate step 2 of Grover's algorithm. Grover proves the existence of some m ∈ O ($\sqrt{N}$), such that after m iterations of step 2 of the algorithm, the probability of finding the register in the marked state is greater than 1/2. Since the amplitude of the desired state, and hence the probability of measuring the desired state, is not monotonic increasing after m iterations, it is not enough to know the existence of m, its value must be determined.

3.2 Searching for More Than One Item

Grover briefly mentions that his algorithm can work in a situation where there is more than one state, such that C (Si) = 1. In fact, this poses no difficulty whatsoever, and regardless of the number of marked states, its superior performance over classical algorithms is still retained. If there are t marked states, we can find one of the marked states in O ($\sqrt{N/t}$) time. This presumes that we know the number of marked elements in advance (Boyer et al, 1996).

Another interesting special case comes when t = N/4, in this case just as in the special case where N = 4, we will find a solution with unit probability after only one iteration, which is twice as fast as the expected running time for a classical algorithm, and exponentially faster than the worst case classical running time (Boyer et al, 1996).

3.3 Optimality of Grover's Algorithm

It is stated in Grover (1996) that his result was optimal, but it is not directly proved. Bennett *et al* (1996)

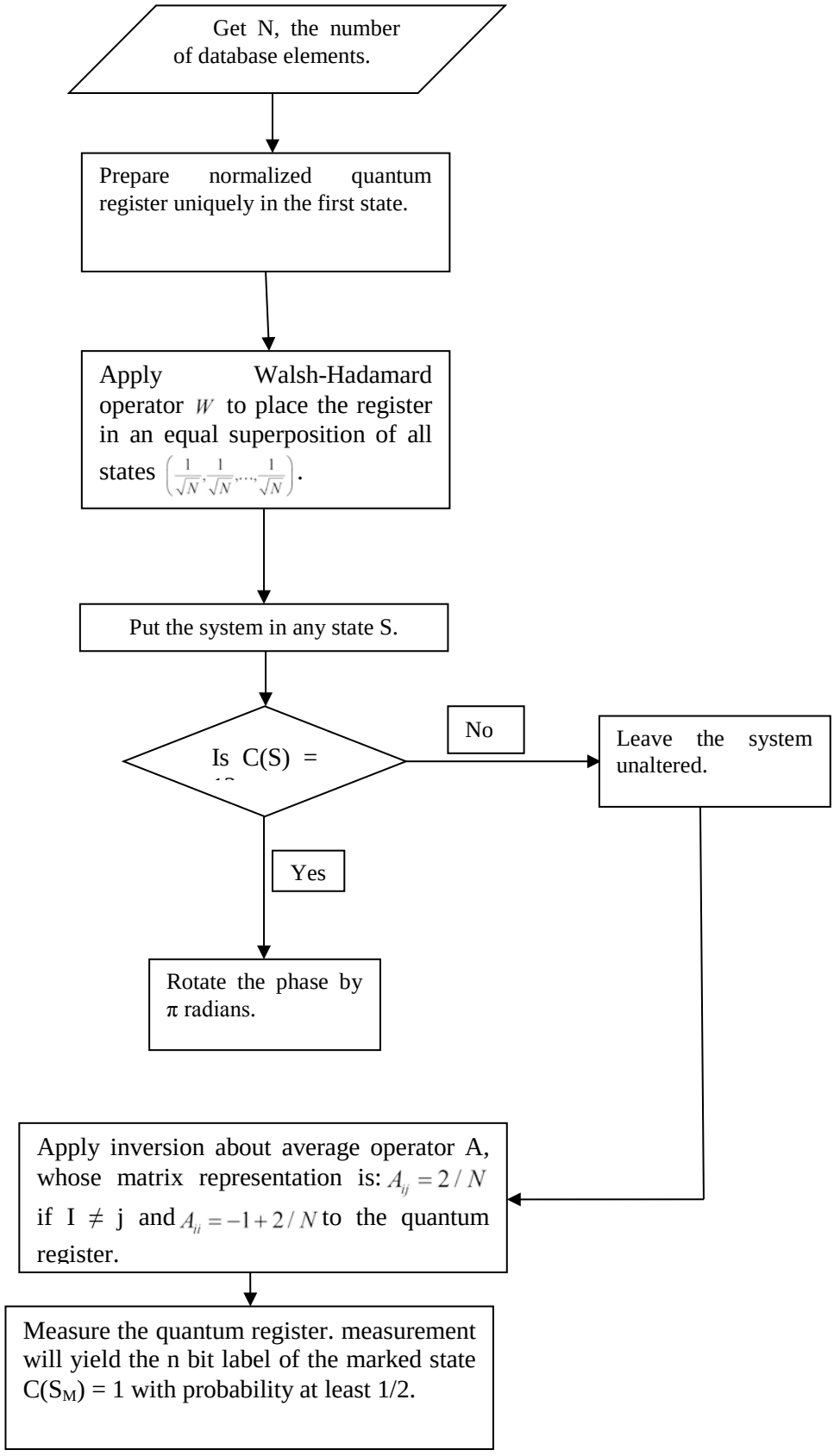

Figure 1: Grover's Algorithm Flowchart.

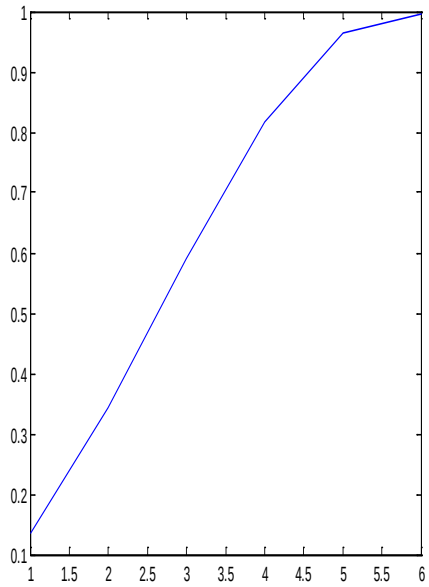

Figure 2: Probability Dynamics.

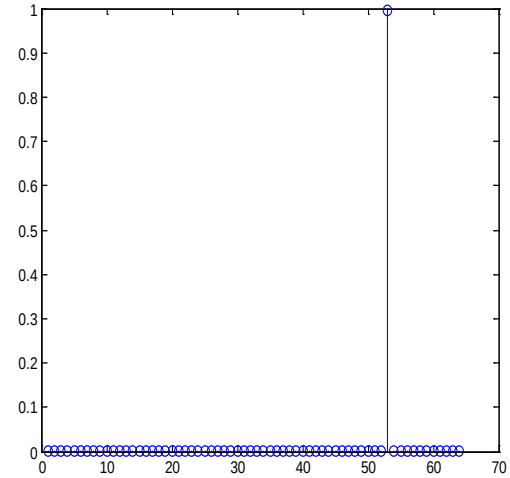

Figure 3: Result Distribution.

established that any quantum algorithm cannot identify a single marked element in fewer than Ω (√N). Grover's algorithm takes O(√n) iterations, and is thus asymptotically optimal. It has been shown since that any quantum algorithm would require at least $\pi/4\sqrt{N}$ queries, which is precisely the number queries required by Grover's algorithm (Grover, 1999).

4 Conclusion

Intriguing breakthroughs occurred in the area of quantum computing in the late 1990s. Quantum computers under development use components of a chloroform molecule (a combination of chlorine and hydrogen atoms) and a variation of a medical procedure called magnetic resonance imaging (MRI) to compute at a molecular level. Scientists use a branch of physics called quantum mechanics, which describes the behavior of subatomic particles (particles that make up atoms), as the basis for quantum computing (Synder, 2008).

Quantum computers may one day be thousands to millions of times faster than current computers, because they take advantage of the laws that govern the behavior

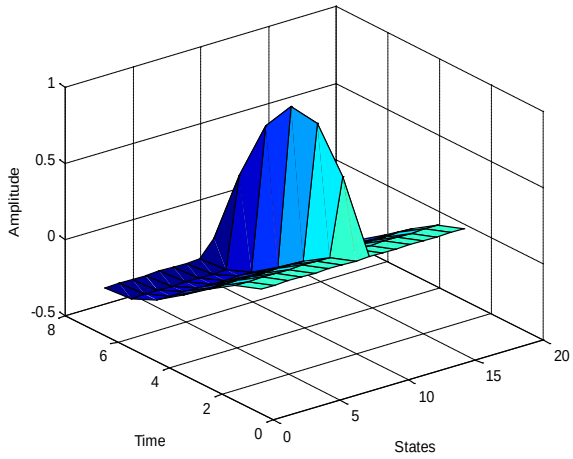

Figure 4: Amplitude-Time Graph for Simulated Grover's Algorithm.

of subatomic particles. These laws allow quantum computers to examine all possible answers to a query simultaneously. Future uses of quantum computers could include code breaking (cryptography) and large database queries. Theorists of chemistry, computer science, mathematics, and physics are now working to determine the possibilities and limitations of quantum computing.

Quantum computation allows for exponential speed up and storage in a quantum register via quantum parallelism. The more basis states represented within the register, the more speed up due to parallelism in the register, and the more improbable it is that a desired state can be measured. Grover's algorithm handles this problem by relying on transformations which cause the amplitude of the marked state to increase at the expense of the non marked states, in a number of ways this is analogous to interference of waves.

Grover's algorithm is unique among quantum algorithms in that it shows a useful calculation that a quantum computer can calculate faster than any classical computer possibly can. At the heart of Grover's algorithm are two unitary transformations, the first is a selective phase inversion, which makes the sign of the amplitude of the target negative. The second unitary transformation is an inversion about average operation. Initially we place the amplitude of all states at the same positive value, each phase switch and inversion about average increases the amplitude of the target state. The exact number of times we perform these transformations is roughly $\pi/4\sqrt{N}$ for sufficiently large N. For a classical algorithm the best time bound is O (N).

5 Appendix A: Grover's Algorithm Simulation

This Matlab codes simulate Lov Grover's quantum database search algorithm by plotting the graph of the probability distribution of finding the marked state as the system undergoes Grover iteration.

```
%This script simulates the Quantum Mechanical Lov Grover's
%Search Alghorithm.
clear all;
%-----parameters-----------
nqubits=6;%number of q-bits
n=2^nqubits;%nnumber of elements in database
findmode=mod(round(n*rand+1),n);%desired element
%-----defining quantum gates
d=-eye(n)+2/n;%diffusion transform
oracle=eye(n);%oracle
oracle(findmode,findmode)=-1;
%--calculate the optimal number of iterations---
finish=round(pi/4*sqrt(n));
%--step(i)--initialization----
psistart=ones(n,1)/sqrt(n);
psi=psistart*exp(i*rand);
%step (ii)--algorithm body----
for steps=1:finish
steps
psi=d*oracle*psi;
probability(steps)=psi(findmode)*conj(psi(findmode));
end
%see the probability dynamics
plot(probability);
%see the result distribution
figure;
stem(psi.*conj(psi));
```

6 Appendix B: Searching for a Needle in a Haystack

This set of Matlab codes simulate the Lov Grover's quantum database search algorithm by using the example of searching for a needle in a haystack.

```
s=[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]';
W=zeros(16,16); %Walsh-Hadamard transformation
for i=1:16
for j=1:16
W(i,j)=2^(-4/2)*(-1)^double(bitand(uint8(i-1),uint8(j-1)));
end
end
s=W*s;
R=zeros(16,16); %Operator to rotate phase
for i=1:16
if i==9
R(i,i)=-1;
else
R(i,i)=1
end
end
A=(2/16)*ones(16,16); %inversion about average transformation
for i=1:16
A(i,i)=-1+2/16;
end
path=zeros(7,16);
path_i=1;
path(path_i,:)=s';
surf(path);
axis([0 20 0 8 -0.5 1]);
xlabel('States');ylabel('Time');zlabel('Amplitude');
F(path_i)=getframe;
path_i=2;
n=1;
while n<(pi/2)*sqrt(16), %needed iteration time is (pi/4)*sqrt(N)
s=R*s;
s=A*s;
path(path_i,:)=s';
surf(path);
xlabel('States');ylabel('Time');zlabel('Amplitude');
F(path_i)=getframe;
path_i=path_i+1;
n=n+1;
end
movie(F,3,3);
```

7 References

[1] Bennett C.H, Bernstein E., Brassard G. and Vazirani U.

[2] (1996). Strengths and Weaknesses of Quantum Computing. *In the proceedings of SIAM Journal of Computing*.

[3] Boyer M., Brassard G., Hoyer P. and Tapp A., Tight Bounds on Quantum Searching. *In the Proceedings of PhysComp.* (lanl e-print quant-ph/9701001).

[4] Deutsch, D., and Jozsa, R.(1992). Rapid Solutions of Problems by Quantum Computation, *Proceedings of the Royal Soc. of London,* 439, 553.

[5] Eppstein D., Irani S., and Dillencthet M. (2000). ICS 260-Fall Class Notes 11: Turing Machines, Non-determinism, P and NP. Available at: http://www1.ics.uci.edu/~eppstein/260/notes/notes11.ps. Accessed on 20th April, 2010.

[6] Grover, L. K. (1996). A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of 28th Annual ACM Symposium on the Theory of Computing*, New York, pp. 212.

[7] Grover L. (2000). Searching with Quantum Computers, lanl e-print quantph/0011118.

[8] Papadimitriou, C (1994). Computational Complexity, Addison-Onesley Publishing Company.

[9] Shor, P. W. (1994). Algorithms for Quantum Computation: Discrete Logs and Factoring. *In Proc. 35th Annual Symposium on Foundations of Computer Science.*

[10] Snyder T. (2008). Law in "Computer." Microsoft® Encarta® 2009 [DVD]. Redmond, WA: Microsoft Corporation.

[11] Williams C. and Clearwater S. (1998). Explorations in Quantum Computing, Springer-Verlag, New York, Inc.