# A HYBRID ALGORITHM FOR FINDING SHORTEST PATH IN NETWORK ROUTING

**[1,2]MOHAMMAD REZA SOLTAN AGHAEI, [3]ZURIATI AHMAD ZUKARNAIN, [4]ALI MAMAT, [5]HISHAMUDDIN ZAINUDDIN**

[1]PhD Candidate, Dep. of Communication Tech. and Network, Faculty of Computer Science,UPM, Malaysia.
[2]Department of Computer Engineering, Islamic Azad University of Khorasgan, Esfahan, Iran.
[3]Asstt Prof., Department of Communication Technology and Network, University of Putra Malaysia.
[4]Assoc. Prof., Faculty of Computer Science, University of Putra Malaysia (UPM), Malaysia.
[5]Assoc. Prof., Institute for Mathematical Research, University of Putra Malaysia (UPM), Malaysia.
E-mail: msoltanaghaii@yahoo.com[1], zuriati@fsktm.upm.edu.my[3], ali@fsktm.upm.edu.my[4], hisham@fsas.upm.edu.my[5]

**ABSTRACT**

Classical algorithms have been used to search over some space for finding the shortest paths problem between two points in a network and a minimal weight spanning tree for routing. Any classical algorithm deterministic or probabilistic will clearly used O(*N*) steps since on the average it will measure a large fraction of *N* records. Quantum algorithm is the fastest possible algorithm that can do several operations simultaneously due to their wave like properties. This wave gives an O($\sqrt{N}$) steps quantum algorithm for identifying that record, where was used classical Dijkstra's algorithm for finding shortest path problem in the graph of network and implement quantum search. Also we proposed the structure for non-classical algorithms and design the various phases of the probabilistic quantum-classical algorithm for classical and quantum parts. Finally, we represent the result of implementing and simulating Dijkstra's algorithm as the probabilistic quantum-classical algorithm.

**Keywords:** *Graph Theory, Algorithm Design, Quantum Algorithm, Network Routing*

## 1. INTRODUCTION

Over the last few years, several quantum algorithms have emerged. Some are exponentially faster than their best classical counterparts [1, 2]; others are polynomially faster [3,4]. While a polynomial speedup is less than we would like ideally, quantum search has proven to be considerably more versatile than the quantum algorithms exhibiting exponential speedups. Hence, quantum search is likely to find widespread use in future quantum computers.

In this study, a classical-quantum algorithm is proposed to find the shortest path in graph. The Dijkstra's algorithm being used for finding shortest path in a given graph and also use quantum search in this algorithm. Simulation results shown that quantum search algorithm is faster than classical one for finding the shortest path in graph.

The rest of this paper is organized as follows. First we have a review on related work on quantum search algorithm and discuss some of the exciting ways that can be used in science and engineering. Then, in section 3, we consider the Dijkstra's algorithm for finding the shortest path for a given graph. Next, in section 4, a new framework is proposed to improve the analyses and design of non-classical algorithm. After that, in section 5, we did a simulation on a classical-quantum algorithm. Finally, the analysis of the results and conclusion are discussed.

## 2. RELATED WORKS

The related works contain of three parts. The first part explains about the research on quantum search algorithm. The second part is describe the NP-hard problems and followed by the latest research on quantum search.

### 2.1 The Quantum Search Algorithm

The quantum algorithm discovered in 1996 has solved the unstructured search problem, under the

assumption that there exists a computational oracle that can decide whether a candidate solution (such as the index of an entry in the telephone directory) is the true solution the index of the sought-after number [3,4].

Grover's algorithm was called the database search algorithm, but this name was dropped because it misled people into thinking that it could be used to search real databases when, in fact, it cannot [6], at least not without first encoding the database in the quantum state to be amplitude amplified. If this encoding is done naively, the cost of creating the database would be linear in its size that is, $O(N)$. Thus, the cost of encoding followed by quantum search would be $O(N+\sqrt{N})$, whereas the cost of a classical search alone would be just $O(N)$ beating the quantum scheme.

### 2.2 Quantum Search and NP-Hard Problems

NP-hard problems constitute a class of computational problems that arise frequently in science and engineering. If any one NP-hard problem could be solved efficiently, then all of them could be solved efficiently due to polynomial cost reductions from one NP-hard problem to another. No one has yet found a polynomial time quantum algorithm for solving NP-hard problems [7]. NP-hard problem can be solved by exploiting its internal structure to "grow" complete solutions by recursively extending consistent partial solutions.

Grover's quantum search algorithm are used to solve an NP-hard problem, such as graph coloring, by creating a superposition of all $N$ possible colorings of the graph, building a polynomial time quantum circuit for testing candidate colorings, and then creating an amplitude-amplification operator based on this circuit to concentrate amplitude in the solution states in $O(\pi/4\sqrt{N})$ steps [9].

## 3. THE DIJKSTRA'S ALGORITHM

Dijkstra's algorithm solves the single-source shortest-path problem when all edges have non-negative weights [8]. Algorithm starts at the source vertex, **s**, it grows a tree, ***T***, that ultimately spans all vertices reachable from ***S***. Vertices are added to ***T*** in order of distance i.e., first ***S***, then the vertex closest to ***S***, then the next closest, and so on. Following implementation assumes that graph ***G*** is represented by adjacency lists [8].

**DIJKSTRA (G, w, s)**

1. INITIALIZE SINGLE-SOURCE (*G, s*)
2. $S \leftarrow \{\}$
3. $Q \leftarrow V[G]$ *// Initialize priority queue Q*
4. **while** $Q \neq \varnothing$ **do** *//while queue is not empty*
5. $u \leftarrow$ **EXTRACT_MIN**($Q$)
6. $S \leftarrow S \cup \{u\}$ *// Relaxation*
7. **for** each vertex v in *Adj*[$u$] do
8. Relax ($u, v, w$)

**INITIALIZE SINGLE-SOURCE (*G, s*)**

1. **for** each vertex $v \in V[G]$
2. **do** $d[v] \leftarrow \infty$
3. $\pi[v] \leftarrow NIL$
4. $d[s] \leftarrow 0$

### 3.1 Analysis

The performance of Dijkstra's algorithm depends of how being choose to implement the priority queue $Q$ [8].

**Definitions:** Sparse graphs are those for which |E| is much less than $|V|^2$ i.e., $|E| << |V|^2$ we preferred the adjacency-list representation of the graph in this case. On the other hand, dense graphs are those for which |E| is graphs are those for which |E| is close to $|V|^2$. In this case, we like to represent graph with adjacency-matrix representation.

When a $Q$ is implemented as a linear array, EXTRACT_MIN takes O(V) time and there are |V| such operations. Therefore, a total time for EXTRACT_MIN in while-loop is $O(V^2)$. Since the total number of edges in all the adjacency list is |E|. Therefore for-loop iterates |E| times with each iteration taking O(1) time. Hence, the running time of the algorithm with array implementation is $O(V^2 + E) = O(V^2)$.

When a $Q$ is implemented as a binary heap, In this case, EXTRACT_MIN operations takes O(lgV) time and there are |V| such operations. The binary heap can be build in O(V) time. Operation DECREASE (in the RELAX) takes O(lgV) time and there are at most such operations. Hence, the running time of the algorithm with binary heap provided given graph is sparse is O((V + E) lgV). Note that this time becomes O(VlgV) if all vertices in the graph is reachable from the source vertices, and Graph G to be sparse.

## 4. A QUANTUM-CLASSICAL ALGORITHM

A quantum computer is a device that takes advantage of quantum mechanical effects to

perform certain computations faster than a purely classical machine can.

In this work, we are going to do two things. First, we are going to develop a classical algorithm that can be run on classical computer. Then quantum algorithm will be simulated on classical computer. Figure 1 shows the structure link of classical part and quantum part of algorithm.

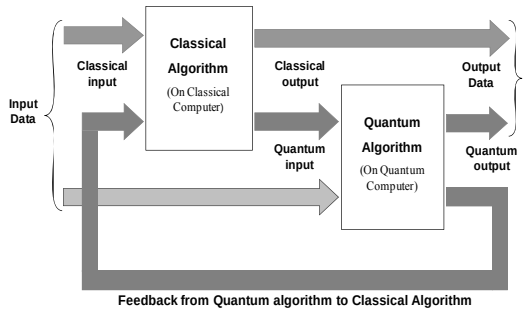



Figure 1: *The structure link of classical part and quantum part of algorithm.*

The probabilistic quantum-classical algorithm can be developed as following steps:

1-Initialize classical part of algorithm.
2-Run first classical part of algorithm.
3-Initialize Machine State.
4-Apply the Unitary Transformation.
5-Measure Machine Stat.
6-Evaluate Measurement.
7-If find solution then go to step 8 else go to step 3.
8-Run second classical part of algorithm.
9-Stop.

Steps 1, 2, 6, 7 and 8 do by classical operations, but steps 3, 4 and 5 do by quantum operations. Figure 2 shows a probabilistic classical-quantum algorithm that can simulated on classical computer and categorized in two parts of classical and quantum. This diagram helps to find a general plan for quantum algorithms and simulation that on classical computer.

Dijkstra's algorithm will be run on the classical part of algorithm except EXTRACT_MIN procedure in line 5. This procedure find the minimum value of a computable function as the set of input arguments ranges over a finite, but unordered list. In this case, if the list is of length $N$, then the quantum cost of finding the minimum is $O(\sqrt{N})$, while the classical cost is $O(N)$. We implement EXTRACT_MIN as a quantum procedure and use quantum search on quantum computer to find minimum value. Using both parts of algorithm, the quantum part is simulated on classical computer.

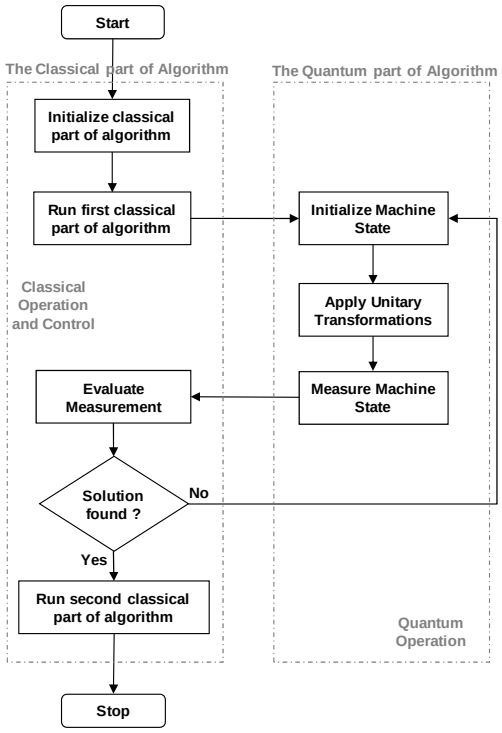


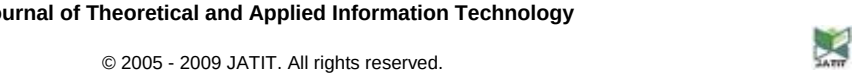

Figure 2: *A probabilistic classical-quantum algorithm*

## 5. IMPLEMENTATION *Q* AS A QUANTUM SEARCH

There are only a few general techniques known in the field of quantum computing and finding new problems that are amenable to quantum speedups is a high priority. Classically, one area of mathematics that is full of interesting algorithms is computational graph theory.

Grover's algorithm is for searching an unsorted list for a specified element. This original idea has been extended to general amplitude amplification that can be applied to any classical algorithm. There are some interesting cases where "Grover-like" techniques do that lead to speedups of classical algorithms. This algorithm is used to find the minimum value of a computable function as the set of input arguments ranges over a finite, but unordered list. In this case, the length of the list is $N$, then the quantum cost of finding the minimum is $O(\sqrt{N})$, while the classical cost is $O(N)$.

### 5.1. Grover's Search Algorithm





The quantum search algorithm performs a generic search for a solution to a very wide range of problems. [3,4,5,6,15]. Quantum searching is a tool for speeding up these sorts of generic searches through a space of potential solutions.

The problem of unstructured search is paradigmatic for any problem where an optimal solution needs to be found in a black box fashion, i.e., without using the possible structure of the problem:

***Problem:*** *Given a Boolean black box function* $f_w : \{0,1\}^n \to \{0,1\}$ *which is equal to* **0** *for all inputs except one ("marked item" w), find the marked item w.*

A black box function is often used to model a subroutine of calculate. We are then interested to know how often this subroutine needs to be performed to solve a problem. Many of the separations between classical and quantum computing power will be formulated in the black box or *oracle* model. In certain problems a quantum algorithm needs to make substantially less calls - or *queries* - to the black box than any classical algorithm. Classically, a black-box function can be simply thought of as a box that evaluates an unknown function *f*. The input is some *n*-bit string $|x\rangle$ and the output is given by an *m*-bit string *f*(*x*). Quantumly, such a box can only exist if it is reversible [10].

Classically, a deterministic algorithm needs to make $2^n$ - 1 queries to identify *w* in the worst case and a probabilistic algorithm still needs $O(2^n)$ queries. Grover gave a quantum algorithm that solves this problem with $O(\sqrt{2^n})$ queries and this is known to be the best possible. Grover's algorithm can hence speed up quadratically any algorithm that uses searching as a subroutine. Grover's quantum algorithm is shown schematically in Figure 3.

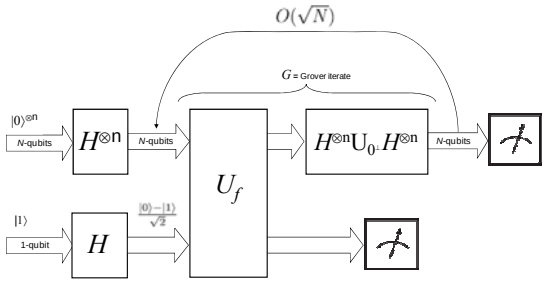



Figure 3: *Grover's quantum searching algorithm.*

Grover's quantum searching algorithm can be written such in the references [15]:

1. Start with the *n*-qubit state $|00\ldots0\rangle$.
2. Apply the *n*-qubit Hadamard gate *H* to prepare the state $|\psi\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle$ (where $N = 2^n$).
3. Apply the Grover iterate ***G*** a total of $\left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$ times.
4. Measure the resulting state.

The operator $\boldsymbol{G} = \boldsymbol{H}\boldsymbol{U}_0^{\perp}\boldsymbol{H}\boldsymbol{U}_f$ is called the *Grover iterate* or *the quantum search iterate*. It is defined by the following sequence of transformations.

1. Apply the oracle $\boldsymbol{U}_f$.
2. Apply the *n*-qubit Hadamard gate $\boldsymbol{H}$.
3. Apply $\boldsymbol{U}_0^{\perp}$.
4. Apply the *n*-qubit Hadamard gate $\boldsymbol{H}$.

The effect $\boldsymbol{U}_f$ on the first register define:

$$U_f : |x\rangle \longmapsto (-1)^{f(x)}|x\rangle$$

The operator $\boldsymbol{U}_0^{\perp}$ is an *n*-qubit phase shift operator $\boldsymbol{U}_0^{\perp}$ that acts as follows:

$$U_{0^\perp} : \begin{cases} |x\rangle \mapsto -|x\rangle, & x \neq 0 \\ |0\rangle \mapsto |0\rangle \end{cases}$$

This operator applies a phase shift of $-1$ to all *n*-qubit states orthogonal to the state $|00\ldots0\rangle$.

### 5.2 Result

We implemented and simulated Dijkstra's algorithm and the Grover's algorithm with Matlab on classical computer. We have tested this algorithm with $N$=$2^n$ possible inputs that *n* is number of qubits.

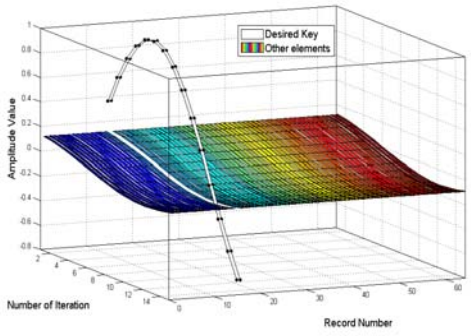
The simulation results for *n*=6 qubits as a data index is shown in the figure 4. In these diagrams, number of possible inputs is *N*=64 and this number is length of queue *Q*. We assumed that there is one solution in queue *Q*. The amplitude value of solution in Grover's algorithm reaches to 1 after (π/4)Sqrt(64)=6.28 iterates and the amplitude value of other data reached to zero. Figure 4(a) and 4(b) show that with 6 iteration we can find solution in queue Q, and if we continue to run the algorithm, the amplitude value of the solution will be far from 1, and lose the solution (figure 4(b)). The maximum iteration of algorithm is (π/4) $\sqrt{N}$ .

The quantum algorithm used quantum computation and needed more memory. Our computer had 4 GB RAM and we could run with maximum 12-qubits. With 12-qubits as input data, we have $2^{12}$ =4096 vertices in queue *Q*.

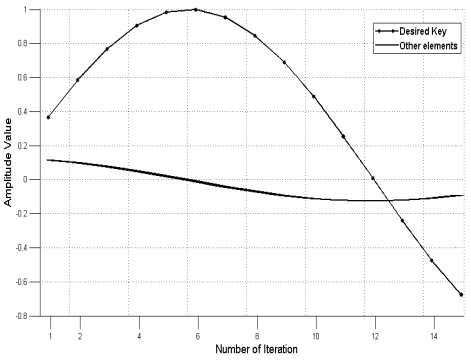
The simulation results for *n*=12 qubits as a data index will be shown in the figure 5(a) and 5(b). In these diagrams, number of possible inputs is *N*=4096 and this number is length of queue *Q*. The

element of 3750 is desired key. The amplitude value of solution in Grover's algorithm reaches to one after $(\pi/4)\sqrt{N}$ =50 iterates and the amplitude value of other data reached to zero.

In figure 6(a) and 6(b) we compare the speeds of Dijkstra's algorithm in three states of implementation for finding the shortest path in graph. These states depend on implementation of EXTRACT_MIN procedure as a linear array, or as a binary heap, or as a quantum search. When a *Q* is implemented as a linear heap or quantum search, the algorithm is more speed up than as a linear array.




*(a) The amplitude value in 15 iterations for 64 elements in queue Q that element of 14 is desired key.*



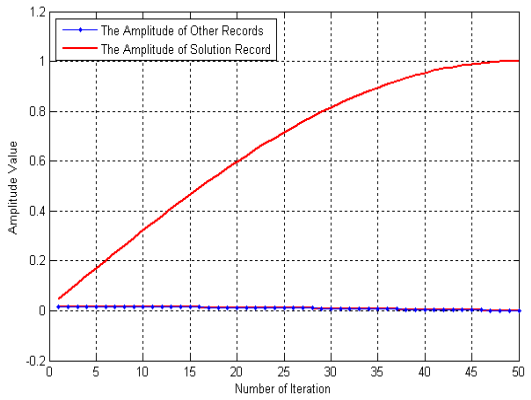

*(b) The amplitude value in 15 iterations.*

Figure 4. *The result of quantum search algorithm with 6 qubits input data and 64 elements in queue with 15 iterations (But 6 iterations are enough for finding record).*

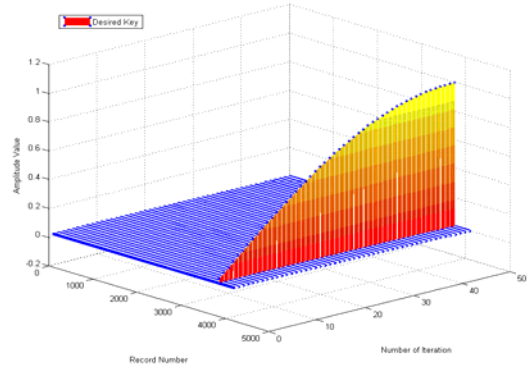
## 6. CONCLUSION AND FUTURE WORK

With comparing the time complexities of the versions of Dijkstra's algorithms, discussed in sections 3 and 5, we can see that the time taken by Dijkstra's algorithm is determined by the speed of the queue operations.

When a *Q* is implemented as a linear array, EXTRACT_MIN takes O(V) time and there are |V| such operations. Hence, the running time of the algorithm with array implementation is $O(V^2 + E) = O(V^2)$. When a *Q* is implemented as a binary heap, EXTRACT_MIN operations takes O(lg V) time and there are |V| such operations. Hence, the running time of the algorithm with binary heap provided given graph is O((V + E) lg V). Note that this time becomes O((V+V)logV)=O(ElgV) if all vertices in the graph is reachable from the source vertices and the graph is sparse. If graph be dense, the running time of the algorithm is $O((V+V^2)\lg V) = O(V^2 \lg V)$.




*(a) Comparison the amplitude of key and other elements in queue Q for 50 iterations.*



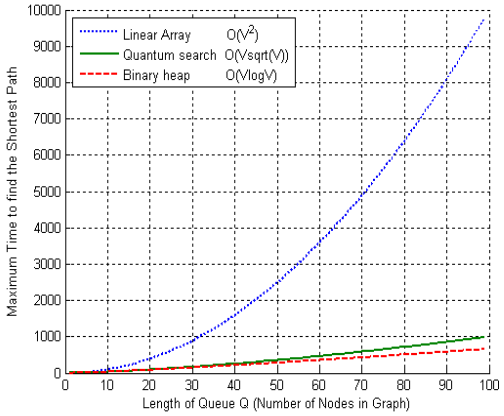

*(b) The amplitude of 4096 elements in 50 iterations that recorded 3750 solution keys in queue.*

Figure 5. *The simulation result of quantum search algorithm with 12 qubits input data and 4096 elements in queue with* $(\pi/4)\sqrt{N}$ *=50 iteration.*

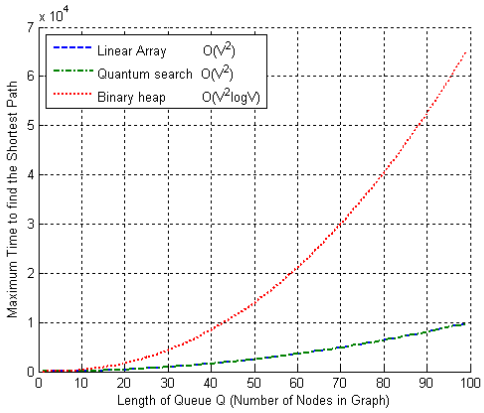
When a *Q* is implemented as a quantum search, EXTRACT_MIN takes $O(\sqrt{V})$ time. Therefore, a total time for EXTRACT_MIN in while-loop is $O(V\sqrt{V})$. Hence, the running time of the algorithm with quantum implementation is $O(V\sqrt{V} + E)$.

When graph is sparse, this time is $O(V\sqrt{V})$ and for dense graph is $O(V\sqrt{V} + V^2)=O(V^2)$.

From the result, we can see that quantum algorithm and binary heap are faster than linear array for finding the shortest path in sparse graph, but for dense graph quantum algorithm is faster than binary heap. Also the quantum algorithm does not need any special conditions for graph and this algorithm can be used for all kinds of graphs with same cost of memory.




*(a) Sparse graph.*




*(b) Dense graph.*

Figure 6. *Speeds comparison of Dijkstra's algorithm in three states of implementation to find the shortest path in the graph.*

We can use the shortest path problem for routing in networks. If the network topology implemented such as star, loop and tree, then this network have sparse graph, so binary heap and quantum algorithm are faster than linear array. The quantum algorithm is faster for the complete topology with dense graph. It's also good for unknown graphs to find the shortest path.

The quantum search algorithm can be extended to other classical algorithms in the future work. Furthermore the quantum algorithms given here can be readily extended to these problems, although the details are yet to be worked out. It is also needs to design a general plan for the implementation and simulation of non-classical algorithms. These ideas can be incorporated into future quantum algorithms.

## 7. REFERENCES

[1] D. Deutsch and R. Jozsa, "Rapid Solution of Problems by Quantum Computation," *Proc. Royal Soc. London*, London, vol. 439, 1992, pp. 553–558.

[2] P.W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *Proc. 35th Ann. Symp. Foundations of Computer Science*, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 124–134.

[3] L.K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," *Proc. 28th Ann. ACM Symp. Theory of Computing*, ACM Press, New York, 1996, pp. 212–219.

[4] L.K. Grover, "Quantum Mechanics Helps in Searching for a Needle in a Haystack," *Physical Rev. Letters*, vol. 79, no. 2, 1997, pp. 325–328.

[5] G. Brassard, "Searching a Quantum Phone Book," *Science,* vol. 275, 1997, p. 627.

[6] C.H. Bennett et al., "Strengths and Weaknesses of Quantum Computing," *SIAM J. Computing*, vol. 26, no. 5, pp. 1510–1523, 2001.

[7] R. Paturi et al., "An Improved Exponential Time Algorithm for k- SAT," *Proc. IEEE 39th Symp. Foundations of Computer Science*, IEEE CS Press, Los Alamitos, Calif., 1998, pp. 628–637.

[8] K. H. Thomas, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction of Algorithms*, MIT press, 2001.

[9] C. Zalka, "Using Grover's Quantum Algorithm for Searching Actual Databases," *Physical Rev. A*, vol. 62, 2000.

[10]. P. Kaya, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing*, Oxford University Press, 2007.