# A Hybrid Algorithm for the shortest-path problem in the graph

Mohammad Reza Soltan Aghaei[1], Zuriati Ahmad Zukarnain[2], Ali Mamat[3], and Hishamuddin Zainuddin[4]
*msoltanaghaii@yahoo.com[1], zuriati@fsktm.upm.edu.my[2], ali@fsktm.upm.edu.my[3], hisham@fsas.upm.edu.my[4]*

## Abstract

*Quantum algorithms run on quantum computers are qualitatively different from those that run on classical computers. Quantum computing algorithms can be used for several problems in graph theory. Most of the Classical algorithms involve searching over some space for finding the shortest-paths problem between two points in a graph and a minimal weight spanning tree. We modified classical Dijkstra's algorithm and implement quantum search instead of classical search, of which it will lead to more efficient algorithm. 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.*

## 1. Introduction

Science and engineering have benefited from advances in algorithms and software. 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.

Quantum search solves is known as the unstructured search problem, which is important for real world applications, such as searching for cryptographic keys [5].

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 follow. 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 presented.

## 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

Lov Grover, a computer scientist at Lucent Technologies Bell Labs, discovered the quantum search algorithm in 1996. The algorithm solves the unstructured search problem, under the assumption that there exists a computational oracle that can decide

**M.R. Soltan Aghaei** is faculty member in Dep. of Computer Eng., Islamic Azad University of Khorasgan, Isfahan, Iran, and PhD candidate in Faculty of Computer Science and Information Tech., University Putra Malaysia 43400 UPM Serdang, Selangor, Malaysia.

**Zuriati A.Z., and Ali Mamat** are with Faculty of Computer Science and Information Technology, University Putra Malaysia 43400 UPM Serdang, Selangor, Malaysia, www.fsktm.upm.edu.my .

**Hishamuddin Z.** is with Laboratory of Computational Sciences and Informatics, Institute for Mathematical Research, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia.

978-0-7695-3489-3/08 $25.00 © 2008 IEEE
DOI 10.1109/ICACTE.2008.137

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].

### 2.2 Quantum search and NP-hard problems

One of the most interesting open questions in quantum complexity theory is whether there exists a polynomial time quantum algorithm for solving so-called NP-hard problems. No one has yet found a polynomial time quantum algorithm for solving NP-hard problems [7].

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 Shortest Path Algorithm in graph

Dijkstra's algorithm solves the single-source shortest-path problem when all edges have non-negative weights [8]. It is a greedy algorithm. 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. $S \leftarrow \{\}$ // S will ultimately contains vertices of final shortest-path weights from s
3. $Q \leftarrow V[G]$ // Initialize priority queue $Q$
4. **while** $Q \neq \varnothing$ **do** //while priority queue $Q$ is not empty
5. $u \leftarrow$ EXTRACT_MIN($Q$) // Pull out new vertex
6. $S \leftarrow S \cup \{u\}$ // Perform relaxation for each vertex *v* adjacent to u
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 we 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.

**3.1.1 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 $O(V^2 + E) = O(V^2)$.

**3.1.2 When a *Q* is implemented as a binary heap (If G is sparse).** 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(ElgV) if all vertices in the graph is reachable from the source vertices. It is reminding that if Graph G to be sparse then we can use a binary heap.

## 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.

The algorithm that we are going to develop is consists of two parts; the first part is classical algorithm of which can be done on a classical computer. The second part is Quantum algorithm of which can be done on a quantum computer that we simulate on classical computer. Figure 1 show the structure link of classical part and quantum part of algorithm.

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.

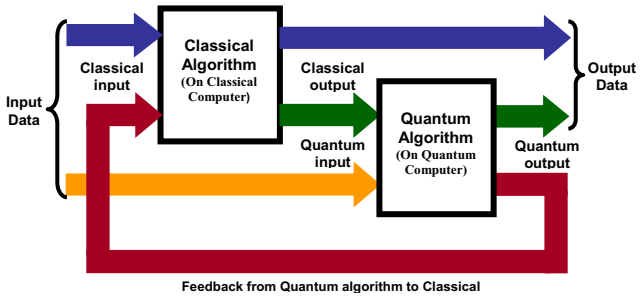



**Figure 1: The structure link of classical part and quantum part of algorithm.**

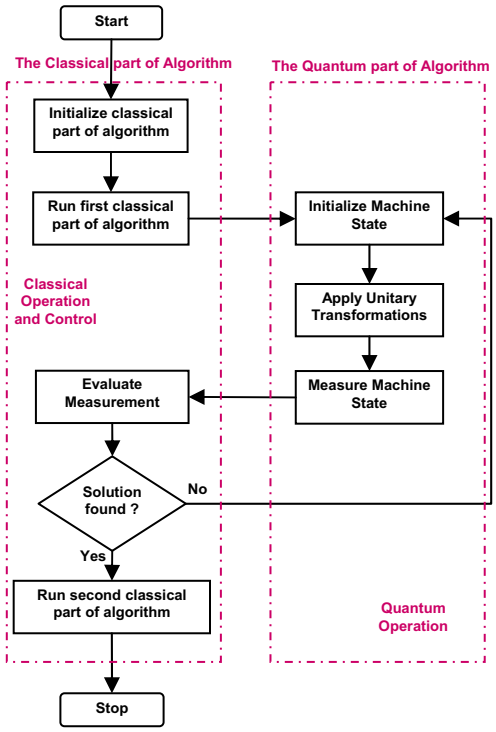



**Figure 2: A probabilistic classical-quantum algorithm**

All lines of Dijkstra's algorithm 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. We implement both parts of algorithm and simulate this quantum part on classical computer.

## 5. Implementation *Q* as a Quantum search

Recently 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.

### 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.

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.

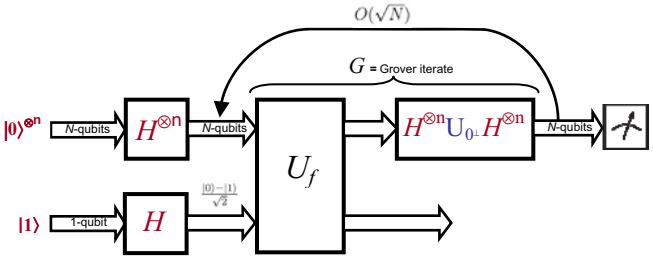



**Figure 3: Grover's quantum searching algorithm.**

Grover's quantum searching algorithm can be written succinctly as follows[15]:

1. Start with the $n$-qubit state $|00\ldots0\rangle$.
2. Apply the $n$-qubit Hadamard gate $H$ to prepare the state $\frac{1}{\sqrt{N}}|\psi\rangle = \sum_{x=0}^{N-1}|x\rangle$ (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 $\boldsymbol{G = HU_0^{\perp}HU_f}$ 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

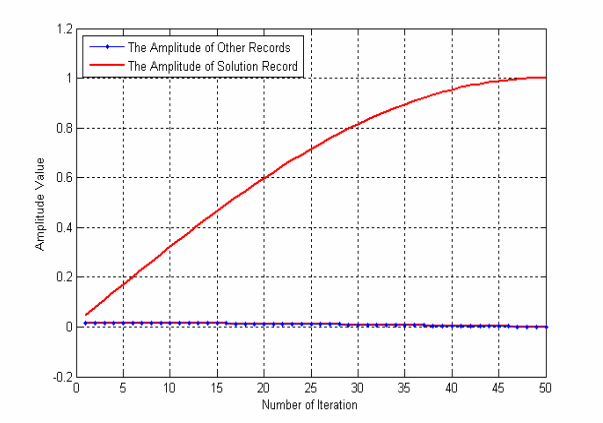
We implemented and simulated Dijkstra's algorithm and the Grover's algorithm with Matlab on classical computer. We have tested this algorithm with $N=2^n$ possible inputs that $n$ is number of qubits. The quantum algorithm use quantum computation and needed more memory. Our computer had 4 GB RAM and 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 4. In these diagrams, number of possible inputs is $N$=4096 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 one after $(\pi/4)\sqrt{N}$=50 iterates and the amplitude value of other data reached to zero.

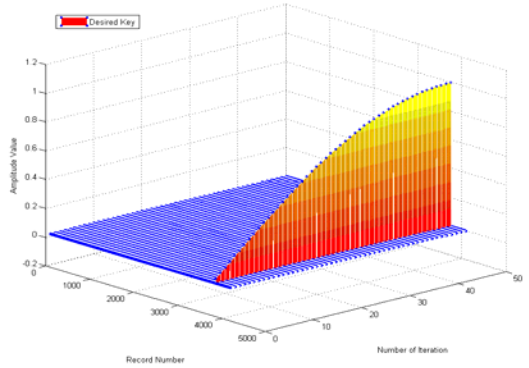
In figure 5 compare speeds of Dijkstra's algorithm in of three states of implementation for finding the shortest path in graph. These states depend on implementation of EXTRACT_MIN procedure as a linear array, or as a binary heap, or as a quantum search. When a $Q$ is implemented as a linear heap or quantum search, the algorithm is more speed up than as a linear array. To use Binary heap have special application and when we can use binary heap that the graph to be sparse.

## 6. Conclusion and future work

With comparing the time complexities of the versions of Dijkstra's algorithms, mentioned 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.




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



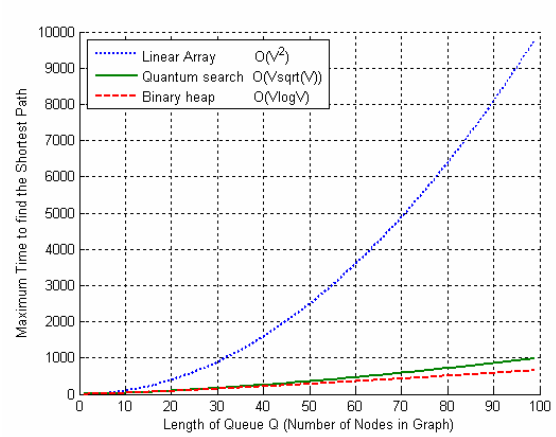

**(b) The amplitude of 4096 elements in 50 iteration that recorded 3750 solution keys in queue.**

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

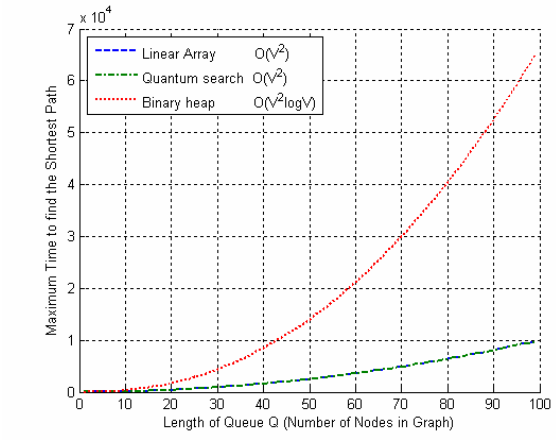
When a $Q$ is implemented as a linear array, EXTRACT_MIN takes O(V) time and there are |V| such operations. Hence, the running time of the algorithm with array implementation is $O(V^2 + E) = O(V^2)$. When a $Q$ is implemented as a binary heap, EXTRACT_MIN operations takes O(lg V) time and there are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

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 $O(V\sqrt{V} + E) = O(V\sqrt{V})$.

It shown the quantum algorithm is faster than linear array for finding the shortest path in graph. Also the quantum algorithm does not need any special conditions for graph and this algorithm can be used for all kinds of graphs.




**(a) Sparse graph.**




**(b) Dense graph.**

**Figure 5. Speeds comparison of Dijkstra's algorithm in three states of implementation to find the shortest path in the graph.**

The quantum search algorithm can be extended to other classical algorithms for 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, as a future work.

## 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.