International Journal of Digital Information and Wireless Communications (IJDIWC) 6(3): 213-218
The Society of Digital Information and Wireless Communications, 2016 ISSN: 2225-658X (Online); ISSN 2412-6551 (Print)

An Efficient Algorithm for Shortest Computer Network Path

Muhammad Asif Mansoor, Taimoor Karamat
Department of Computer Science & Information Technology,
Virtual University of Pakistan,
Lahore, Pakistan
m_asifmansoor@hotmail.com, taimoor.karamat@vu.edu.pk

ABSTRACT

To find the shortest path in computer network problem is always a challenge for the network designers and many algorithms have been developed in past to solve this problem. One of these algorithms is the quantum classical algorithm that is the possibly fastest algorithm due to its wave like properties and it works faster with different operations. An algorithm is usually called an efficient algorithm if it takes less computation time than other algorithms. One of the popular algorithms for finding the shortest network path is Dijkstra's algorithm. In this paper an efficient algorithm is presented to find shortest network path and its comparison with already developed algorithms which are Dijkstra's algorithm and Dynamic Dijkstra's algorithm. Along this simulation results are generated and proved that the proposed algorithm has improved the performance for shortest network path.

KEYWORDS

Efficient Algorithm, Dijkstra's Algorithm, Shortest Network Path, Graph Theory, Computation Time

1. INTRODUCTION

Many efficient algorithms have been developed in recent past to find the shortest network path. Some of them were extraordinary fast and best in terms of exponential and some were fast in terms of polynomial. Our Proposed algorithm showed some uniqueness than other algorithms developed in the past in terms of finding the shortest network path.

In this research paper an algorithm is proposed to find the shortest network path. We have used Dijkstra's and Dynamic Dijkstra's algorithms for comparison of the shortest network path in the experimental network graph. Our simulation results prove that our proposed algorithm performs well than the other algorithms.

This paper is organized in following sections; first section is introduction followed by the section related work then there is the section of proposed algorithm & analysis after that there is section of implementation & simulation work which is followed by the sections future work and conclusion.

2. RELATED WORK

To find the shortest path in a network is a key area of research, where an efficient algorithm can increases network efficiency and reduce load in any network. Though researchers are still proposing new dimensions to improve the performance of a network by decreasing the number of computations to find optimum path, fast track updating of new routes, routing table recalculations etc. In [11] the author has proposed an algorithm that reduces calculations by 8% as compared to Dijkstra's algorithm. Instead of working on the whole network for calculations the proposed algorithm is applied on sub graph restricted to upper bound distance between two nodes, experimental results are for 150 nodes with 176 edges. In [12] it was proposed that first we should compare modified matrix with values of shortest path, which makes it a complex solution whereas in [13] the authors introduced a solution where we have feature matrix that have 0-1's, which makes it difficult to read the shortest path lengths. The aforementioned issues were resolved in [14] the path and lengths were found intuitively by increasing shortest path tree.

The proposed algorithm that is presented in this paper is tested against both static and dynamic Dijkstra's algorithm. In past many algorithms have been developed [1, 2] to find the best and efficient network path. One of them is Dijkstra's algorithm

which is very popular network popular network graph search algorithm that was generated to find shortest network path. When the Dijkstra's algorithm is applied [3] in a network graph then it starts from source vertex called ***s*** and it is reachable from ***S*** that pass through the all vertices and a tree structure ***T*** grows up. It follows the rule that at start it goes with first vertex ***S*** then to its closest vertex, and then to the next one closest vertex and so on. As Dijkstra's algorithm is a quantum search algorithm [4] and mostly used in finding the shortest network path in which all the edges have non-negative values. The main advantages which are found [5] are that it is better and faster than other algorithms used for computing the dense graphs. It is also very useful to find the shortest paths of unknown network graph. The main disadvantages which are found are that as this algorithm is from the class of quantum search algorithms so it can be mixed with other classical algorithms. On the basis of the problem found in Dijkstra's algorithm [6, 7] we have proposed a new approach for the solution of such problems which is discussed and proved via simulation work in subsequent sections.

3. PROPOSED ALGORITHM AND ANALYSIS

The proposed algorithm is static routing algorithm which becomes in-effective when the link status of the network is changed. This algorithm requires the updates from smaller part of the shortest path. This increases the load on router's CPU as it requires re-computation of the new shortest path as a whole and not the use old shortest path. As a result more routers' resources are used and it takes more time for the Dijkstra's algorithm to find the shortest path. Packet may be loss due to this reason and the routing tables do not update to show the correct network topology information.

By definition an efficient algorithm is that one which takes less time than other algorithms of that class of algorithms. In this research paper we presented a proposed algorithm which will reduce the total time taken to find the shortest network path. Static and dynamic algorithms are applied to find very important constraint which is the time taken by running this proposed algorithm to find the shortest network path.

Static routing algorithm is used to find the shortest path in the network rather than using the dynamic algorithm. The reason for using static routing algorithm in such a situation is because there are large numbers of nodes for which the weights are to be computed. So in this scenario it is better to use the static routing algorithm rather than dynamic algorithm which takes more computational time for each node in the network.

The pseudo code of the proposed algorithm which is divided into three steps is as follows:

Step 1: Find the depth of the network by explore the structure of network

```
Find the depth of the network      // Use
Depth First Search (DFS) algorithm

D = Depth of entire network for all nodes

Do

Find the link of which link cost is
changed

Use DFS to compute depth of the link

                if depth of the link < D
// Entire network depth  is greater than
depth of the single node link
                then
                                Go to Step 2
                 else
                                Go to Step 3
                endif
endfor
```

Step 2: Perform Static Routing

By using the static routing find the shortest path

```
      G = (V, E)          //  Dijkstra's
algorithm is used

Routing Table information is updated
```

Step 3: Perform the Dynamic Routing

```
Use dynamic routing to find shortest path

 G = (V, E)        // Dijkstra's algorithm
                  is used
```

Update the previous nodes initialization information

```
Depth First Search is used from a node in
shortest        path        time        T
// All previous nodes are updated

From Q, remove edges with ended nodes and
which are from previous nodes

     In Q, old information is updated
   A temporary shortest path time T is
                obtained
          while (previous nodes)
    {previous nodes, inc_node} ← Extract
                  (M)
 if some incoming links hold by v between
             previous nodes
                  then
       if incoming links for D(i)> inside
             nodes for D(j)
                   then D(i) = D(j)
                     endif
                  endif
          Routing table is updated
```

The aforementioned procedure is adopted in our algorithm. At first to make a decision for applying which algorithm on a given network, the entire depth of the network is found by using Depth-first search. Secondly, static routing algorithm is applied for whose links whose weights are new and have depth less than 20%. At last stage, if some links have the weights with depth more than 50% dynamic algorithm is used to compute the shortest network path.

In some cases where the weight of the links is new and is near to end node, dynamic algorithm is applied because there is small number of nodes for which the weight is to be computed. In this case only weight of old and effected nodes is calculated rather than of every node.

4. SIMULATION AND RESULTS

In this section simulation is carried out and results are generated. The proposed algorithm is used for the network depth of about 50% by applying the static and dynamic routing algorithms and simulated them from the depth of 40% to 60%. We have found in results of simulation that the 40% of the network depth is best possible value for this algorithm as its computation time is less than the computation time of other algorithms.

This simulation work is shown in below Figure 1 which is displaying the comparison between different algorithms.

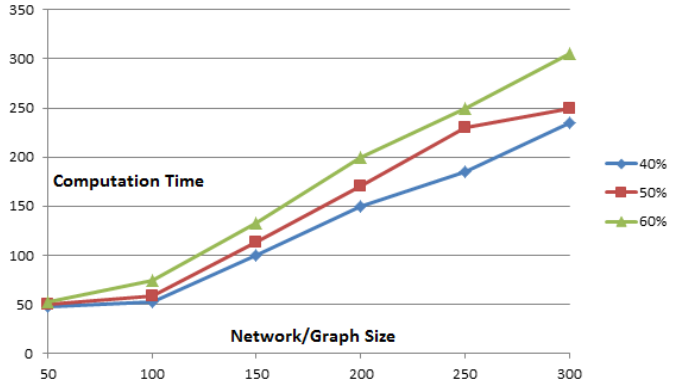

Figure 1: The Simulation Result with respect to Graph Size

Performance and Simulation Results:

We have evaluated the performance of proposed algorithm for shortest network path and compared it with other algorithms. One of them is Dijkstra's algorithm and the other one is Dynamic Dijkstra's algorithm.

For the input parameters of simulation, we have used the parameters including the number of nodes in the network, rate of change for link weights and the deviation of link weights. We have taken the following values to perform simulation work:

- Number of Nodes in Network = 50, 100, 150, 200, 250, 300
- Rate of change for link weights in percentage = 100, 200, 300
- Link weights changing (Deviation) = 5, 10, 15

On the basis of above values the simulation work is performed to compare 3 different algorithms in which one in our proposed algorithm.

Simulation Results:

In simulation for comparison of algorithms the computation time for each algorithm is measured for finding the shortest network path. Images of simulation results of comparison between our proposed efficient algorithm, Dijkstra's algorithm

and Dynamic Dijkstra's algorithm are shown in this research paper.

Rate of Change for Link Weights:

In the Figure 2, the rate of change for link weights on value of 100% is shown. Here in this figure it can be seen that the computation time for the value of 100% rate of change for link weights for our proposed algorithm is less than the other algorithms.

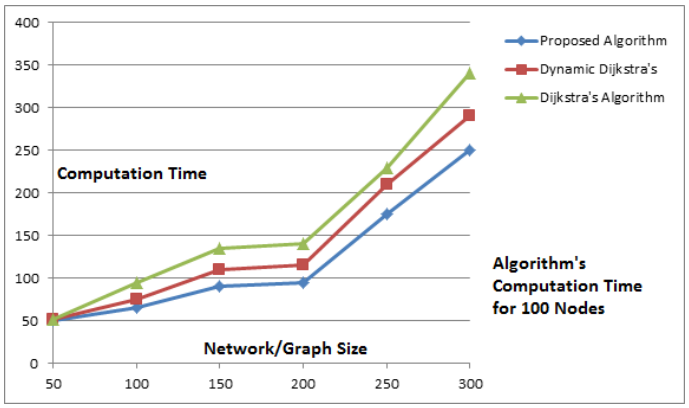

Figure 2: Computation Time against network size of 100 nodes for Rate of Change for Link Weights

In Figure 3, the graphs for rate of change for link weights with value 200 and in Figure 4, the link weights with value 300 are shown.

It is clear from these graphs that are generated via simulation results; our proposed algorithm is taking less computation time than the other algorithms.

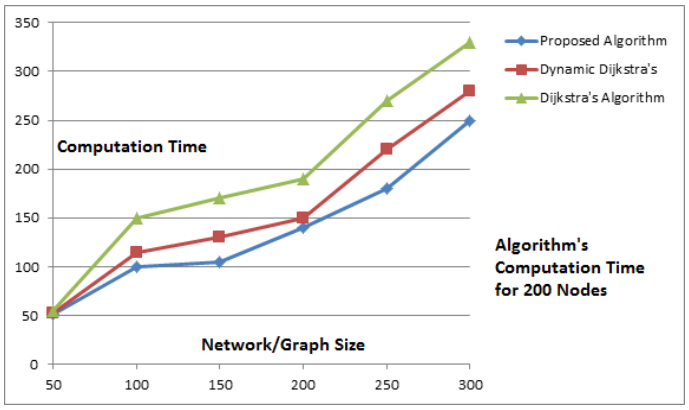

Figure 3: Computation Time against network size of 200 nodes for Rate of Change for Link Weights

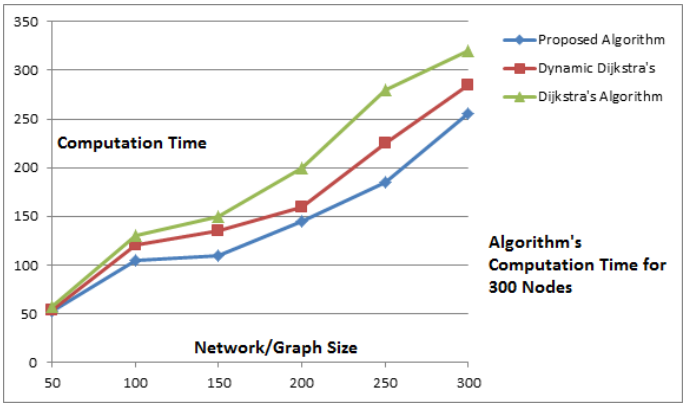

Figure 4: Computation Time against network size of 300 nodes for Rate of Change for Link Weights

Now the simulation is performed to compare the deviation of the link weights against different values for the efficient algorithm, Dijkstra's algorithm and Dynamic Dijkstra's algorithm.

In below Figure 5 the simulation result in forma of graph for the Deviation of Link weights for value 5 is shown and it can be seen that the computation time for our proposed algorithm is less than other algorithm's computation time.

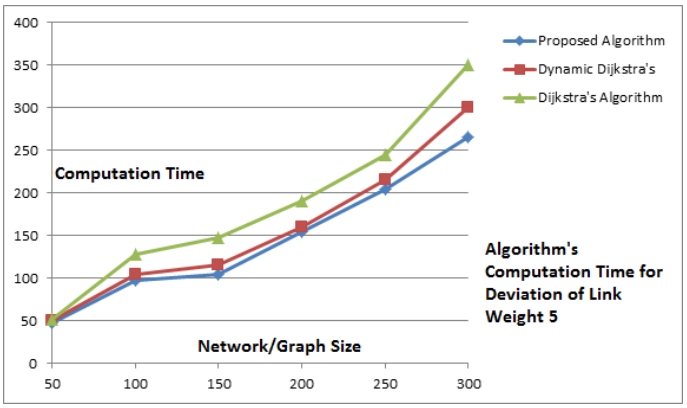

Figure 5: Computation Time against network size for Deviation value of 5 for Link Weights

In next two simulation result graphs the deviation of link weight for value 10 and 15 are shown. The computation time of our proposed algorithm is less than Dijkstra's algorithm and Dynamic Dijkstra's algorithm.

The simulation result graphs for both deviations of link weights values of 10 and 15 are shown in Figure 6 and Figure 7 respectively to find the computation time against the network size.

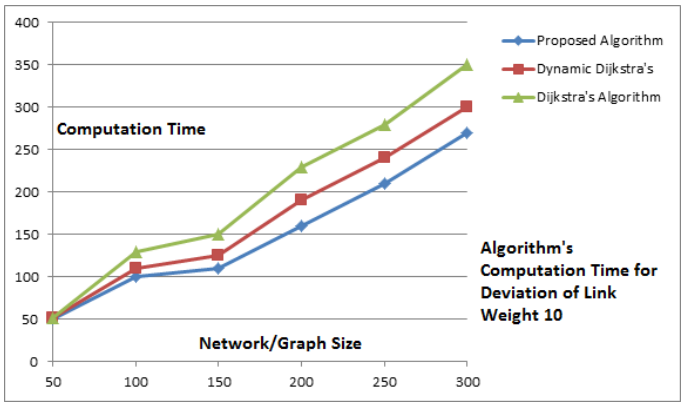

Figure 6: Computation Time against network size for Deviation value of 10 for Link Weights

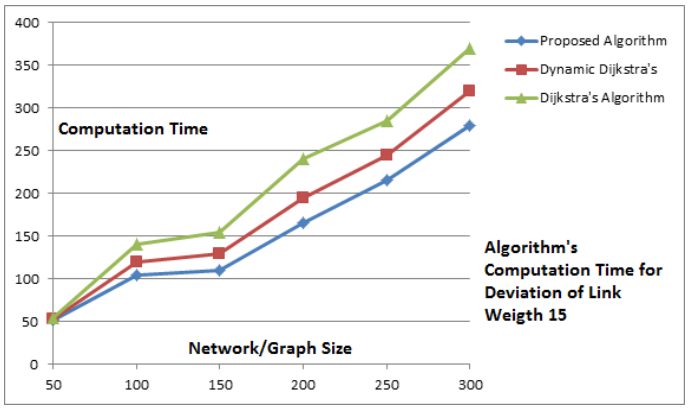

Figure 7: Computation Time against network size for Deviation value of 15 for Link Weights

As shown in all above simulation results, our proposed algorithm for finding the computation time of shortest network path is less than other algorithms including Dijkstra's algorithm and Dynamic Dijkstra's algorithm which were used in the evaluation of an efficient algorithm.

Here it can be noticed that as we increase the number of nodes in every simulation result the computation time of each algorithm also increases and it takes longer computation time to find the shortest network path.

Furthermore if the rate of change for link weights is calculated lower then the computation time is counted for long time on liner base but if the rate of change for link weights is higher then the computation time for finding the shortest network path becomes longer on nonlinear basis.

The results of simulation work which are shown in all above graphs prove that our proposed algorithm is better than already developed algorithms in the past by other researchers. The results are compared for computation time against the network size which includes the number of nodes.

5. Conclusion

By applying this algorithm on an experimental network and measure the effectiveness of total computation time, our developed algorithm is an efficient algorithm to find the shortest network path. This is also proved by the simulation results in form of graphs mentioned in this research paper. The proposed algorithm is compared with Dijkstra's and Dynamic Dijkstra's algorithms and it shown the results better in performance than the other algorithms for the shortest network path in terms of computation time.

6. Future Work

Our proposed algorithm to find the shortest network path in less time than the other algorithms from same class can be extended further to find shortest network path in further less time than current time.

Some more details are also required to work more in our proposed algorithm. More optimized work would be needed to make this algorithm to work more smartly and find the best shortest path in a given network.

Furthermore for the shortest network path problems in network routing, more network topologies like Star, Ring, and tree can be implemented and for these topologies the searching algorithms can be faster than linear array type algorithms. More work can also be done for unknown network paths for which some powerful algorithms are required to develop.

Our proposed algorithm can be further extended in the future works and a generalization form of algorithm can be developed which will work for every type of network path and will find the best network path with best computation time.

7. ACKNOWLEDGMENT

We special thanks to our Advanced Computer Network course teacher Dr. Amir Qayyum who recommended our research topic and guide us to work on this exciting computer network research topic.

8. REFERENCES

[1] Taehwan Cho, Kyeongseob Kim, Wanoh Yoon and Sangbang Choi: A Hybrid Routing Algorithm for an Efficient Shortest Path Decision in Network Routing, July 2013 :1-5

[2] Huseyin Kusetogullari, Md. Haidar Sharif, Mark Leeson, Turgay Celik: A Reduced Uncertainty-Based Hybrid Evolutionary Algorithm for Solving Dynamic Shortest-Path Routing Problem, February 2012 :1-2

[3] AmmarW. Mohemmed, Nirod Chandra Sahoo: Efficient Computation of Shortest Paths in Networks Using Particle Swarm Optimization and Noising Metaheuristics, April 2007 :2-5

[4] Gang Feng, Turgay Korkmaz: A Hybrid Algorithm for Computing Constrained Shortest Paths, April 2013 :1-3

[5] Mohammad Reza Soltan Aghaei, Zuruati Ahmad Zukarnain, Ali Mamat, Hishamuddin Zainuddin: A Hybrid Algorithm for Finding Shortest Path in Network Routing, June 2009 :3-6

[6] Kirill Levchenko, Geoffrey M. Voelker, Ramamohan Paturi, Stefan Savage: XL: An Efficient Network Routing Algorithm, September 2008 :1-7

[7] Sumitha J.: Routing Algorithms in Networks, December 2013 :5-6

[8] K.Rohila, P.Gouthami, Priya M: Dijkstra's Shortest Path Algorithm for Road Network, October 2010. :4-8

[9] Andrew V. Goldberg, Eva Tardos and Robert E. Tarjan: Network Flow Algorithm, August 1990 :9-14

[10] Tayseer S. Atia, Manar Y. Kashmola: A Hybrid Hopfield Neural Network and Tabu Search Algorithm to Solve Routing Problem in Communication Network, June 2012 :13-18

[11] Practical Algorithm for Shortest Path on Transportation Network Zhen Zhang, Wu Jigang, Xinming Duan School of Computer Science and Software Tianjin Polytechnic University, Tianjin 300160, China :7-17

[12] Liu Ping, Bai Cuimei A Improvement of the Shortest Path Based on the Dijkstra Algorithm [J]. Qinghai Normal University, 2008,1 (1) :79-80

[13] Li Guilin A Improvement of the Dijkstra Algorithm [J]. Development and Application of Computer 2009 22 7 13-14

[14] An Improvement of the Shortest Path Algorithm Based on Dijkstra Algorithm Ji-Xian Xiao college of science Hebei Polytechnic University Tangshan, China xiaojix@yahoo.com.cn Fang-Ling Lu college of science Hebei Polytechnic University Tangshan, China :5-9