International Journal of Electrical and Computer Engineering (IJECE)
Vol. 12, No. 2, April 2022, pp. 1675~1682
ISSN: 2088-8708, DOI: 10.11591/ijece.v12i2.pp1675-1682

Flight-schedule using Dijkstra's algorithm with comparison of routes findings

Israa Ezzat Salem, Maad M. Mijwil, Alaa Wagih Abdulqader, Marwa M. Ismaeel
Computer Techniques Engineering Department, Baghdad College of Economic Sciences University, Baghdad, Iraq

Article Info

Article history:

Received Sep 28, 2020
Revised Aug 5, 2021
Accepted Sep 4, 2021

Keywords:

Destination nodes
Dijkstra's algorithm
Optimization
Route
Source nodes

ABSTRACT

The Dijkstra algorithm, also termed the shortest-route algorithm, is a model that is categorized within the search algorithms. Its purpose is to discover the shortest-route, from the beginning node (origin node) to any node on the tracks, and is applied to both directional and undirected graphs. However, all edges must have non-negative values. The problem of organizing inter-city flights is one of the most important challenges facing airplanes and how to transport passengers and commercial goods between large cities in less time and at a lower cost. In this paper, the authors implement the Dijkstra algorithm to solve this complex problem and also to update it to see the shortest-route from the origin node (city) to the destination node (other cities) in less time and cost for flights using simulation environment. Such as, when graph nodes describe cities and edge route costs represent driving distances between cities that are linked with the direct road. The experimental results show the ability of the simulation to locate the most cost-effective route in the shortest possible time (seconds), as the test achieved 95% to find the suitable route for flights in the shortest possible time and whatever the number of cities on the tracks application.

Corresponding Author:

Maad M. Mijwil
Computer Techniques Engineering Department, Baghdad College of Economic Sciences University
Baghdad Province, Yarmouk, Nafaq Al-Shurta, Iraq
Email: mr.maad.alnaimiy@baghdadcollege.edu.iq

1. INTRODUCTION

Edsger Dijkstra is a Dutch computer scientist in 1959, He is considered as a developer of Dijkstra's algorithm [1]. It's a dynamic programming (DP) algorithm [2], [3]. Dijkstra is an algorithm applied for search and it provides a shorter-route from single-source for a graph with nonnegative edge route costs [4]. It depends on the close road to explain the problem of the single source-shortest route [5], [6]. It keeps the nearest way to source on the graph form source (s) to destination (v) [7]. We think the problem of the single-source shortest-route problem in a designated graph with positive edge values is a common obstacle which occurs in various apps such as in long road nets: social nets. [8], [9], route information protocol [10], multicast routing [11], wireless networks [12], [13], and internet of things (IoT) applications [14], [15].

Dijkstra's algorithm aims to define the shortest routes for a given starting node [16], [17]. When choosing the node for the shortest-route, the algorithm follows a so-called greedy strategy [18]. The node that is closest to the starting node is always chosen. With the Dijkstra algorithm, all edge weightings must therefore be non-negative [19]. We always face the problem of shortest-route in our ordinary life [20], the widespread of e-commerce, the rapid growth of the logistics industry, and the growing complexity and congestion of road traffic, for example, the efficiency of material distribution [21], [22] and travel efficiency and also road use in the current transport network can be improved to reduce the shortest route problem [23]. The intelligent traffic

systems (ITS) are created to solve this problem [24]. Route planning also plays a vital role in intelligent transport systems [25]. Consequently, it is important to improve route planning skills. Using an appropriate data structure to save the network information is considered to be a very important procedure to ensure that the PC and the route planning algorithm can recognize the network and calculate the shortest route in the network [26]. The Dijkstra algorithm [27] and A* algorithm [28] are most common shortest-route optimization algorithms.

The (v) edges are checked to determine the destination for reaching. Dijkstra algorithm has several forms wherever the Dijkstra algorithm originates from start node to end node. However, the more common type is fixation shorter route from the single node for all nodes which lie on the graph. Dijkstra's algorithm applications are the following: i) it is applied as a subroutine in excellent algorithms; ii) Dijkstra's algorithm is used for reducing hops number from computer to another and for enhancement to transfer the file between the computers inside the network lead to occupy all time [1]; iii) by Dijkstra's algorithm, it could provide both the address of the origin node and that of the destination node for the remote server to determine the shortest route [4]; iv) Dijkstra's algorithm is used as high efficiency in traffic information systems wherever it calculates the shorter road; and v) the algorithm uses Google Maps for determining the shortest route between 2 points [29]. Dijkstra's algorithm disadvantages are: i) It spends extended time in the necessary resources. Therefore, it's blind [1], ii) It does not have a handle with negative edges. Therefore, it doesn't get right shortest route.

Dijkstra's algorithm is applied in network related protocols for determining vertex on the graph, to determine the shorter route with the lowest cost between two nodes on the network. Also, it is used to find lower costs and shortest route between two vertexes [30]. The nodes represent the cities or points, and the edge route costs are distances among the cities that are linked with the same road, Dijkstra's algorithm is applied to fetch the shortest road between any two cities.

The rest of the article's structure is as follows: section 2 illustrates our methodological steps. In section 3, the experimental effects of the Dijkstra algorithm are exhibited. Conclusions and future scope are in the last section.

2. METHOD

A MATLAB program simulation is used in our study to carry out the Dijkstra's algorithm as in Figure 1. MATLAB has an excellent library for graphs called Graph that allows to relate the graphs in nodes and edges in addition to making good graphs of these graphs. It also allows to calculate the time it takes to process easily. This algorithm, like Floyd's theorem, has the ability to explain the shortest path problem, both for cyclic and acyclic networks. In such a way that the loops that a network presents do not restrict the control of the algorithm. In other words, it utilizes for defines marks from the source node and each of the subsequent nodes. These marks include information related to an accumulated value of the node's size and the closest source of the route. Marks correspond to nodes, not arcs. In this algorithm, these marks are temporary and permanent. Temporary marks are those that can be changed while there is the possibility of obtaining a shorter route for themselves; otherwise, the tag becomes hard.

The below-weighted graph consists of five vertices (v1- v2- v3- v4-v5). Edge cost mean value between any two v. [30] Such as edge cost between vertices 1 and vertices 2 is seven. The graph is a di-graph, the vertices are representing the airports, the distances are flights between two points, and the total time is included from (*depart T)* to (*arrived)*.

The distance should be the shorter road to arriving at the target point in a shorter time. Wherever the queue (Q) refers to the priority queue, *T*. is time, *depart T* is start time, *arrived* is arriving time. For all vertices, *w*, adjacent to *v* do:

Make Priority Queue, *P*, of fights f, with $f.departT \geq T[v]$ keyed by $f.arriveT$

```
Time t← ∞ //will need for relaxation.
If P in not empty then t← p.min().arriveT //no need to remove
```

Relaxation is nearly the same:

```
If t < T[w] then
T[w] ← t
Update w in Q
```

But it all together by:

```
FlightAgenda (DiGraph G, Vertex s, Time startT)
//Input: G di-graph, not a simple graph
//Vertices are airports
//di-edges are flights with two weights
//deparT is the departure time at origin airport
```

```
//arrive is the arrival time at the destination airport
//s is the origin airport and starT is the starting time
//inintlize arrival time, T
T[s]←startT
For all vertexes, v≠s do T[v] ←∞
```

Make priority queue, *Q*, of vertices keyed by *T* While *Q* is not empty do

```
V←Q.removeMine()
For all vertices, w, adjacent to v and in Q do
// determine the next flight
Make Priority Queue, P, of fights, f, with f.departT≥T[v]
Time t ←∞
If P in not empty then   t ←P.min⟦()⟧.arrive T // no need to remove
// relaxation
If t < T[w] then
T[w] ← t
Update w in Q
Return T
```

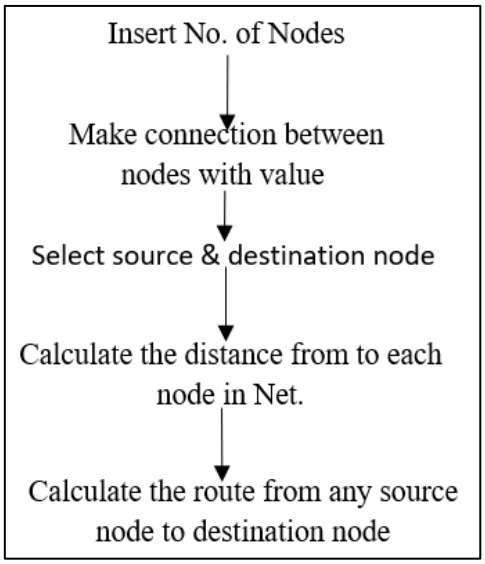


Figure 1. Flow chart of algorithm

All the above mentioned steps are back to the Dijkstra's algorithm with updates added by the authors therefore, this study focuses mainly on the results of this algorithm and its steps in solving the problem of organizing flights between cities.. During the execution of the work, each node will be marked with its least distance to a node that is specified in the implementation where this node is with a value of zero, meaning the length is zero. Still, we do not know the minimum distance. It starts with the value of the infinity. In addition, we will have the current node value. We start by checking the current node's neighbours in action and in no specific order. For example, if the current node is x, we take the nearest neighbours to it and say that it is x1. We add the distance value of the current node (the least distance is zero) with the weight of the edge connecting the current node to x1, and for example, it is 6, and we get the result of 0+6=6 . This value is also compared with the minimum distance x 1, which is the distance that is less than the infinity. Next, we check the other node, for example, to be x2. We add zero minimum distance of x, the current node with a value of 2, edge weight to get 2. We compare this result with the minimum distance x2. Moreover, we have verified all the neighbours of x. we mark it as visited.

Figure 2 displays in paint the mechanism for running from the current node to the neighbour nodes. Figure 2 explains how this algorithm works, where Figure 2(a) is the starting point and is designated *s* and at time 0. Figure 2(b) gets the shortest route between *f* and *y*. Figure 2(c) is the flight going to point y, and then there will be three opportunities available which are *t, x* and *z* routes. In Figure 2(d), the shortest route between *t, x* and *z* points will be defined, which is *t* and *x* (are most beneficial). While route *x* is long, it will be dismissed. Eventually, the *x* point will be reached through the *t* and *z* routes in the shortest time, as shown in Figures 2(e) and 2(f).

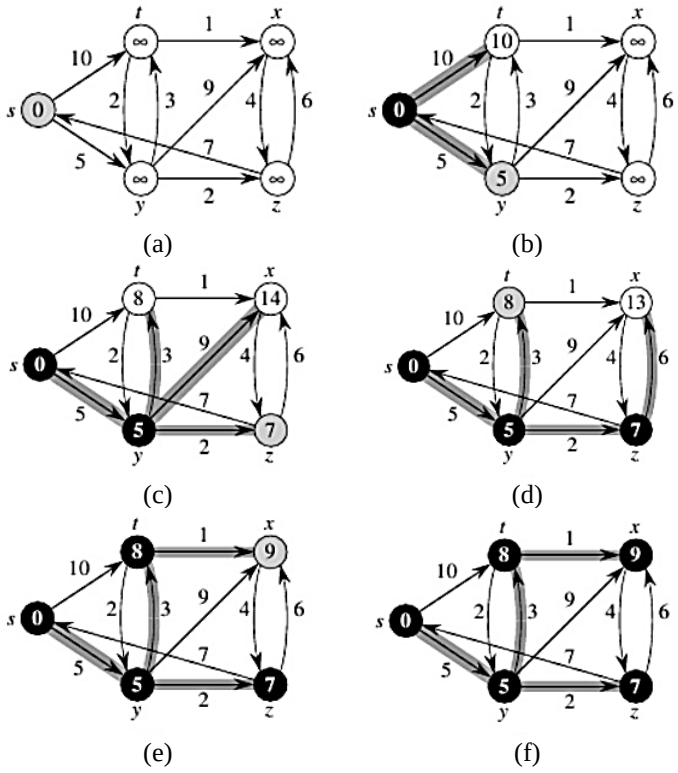


Figure 2. Graph to choose the closest point: (a) it is starting point at time 0, (b) it is gets the shortest route, (c) it is the flight going to point *y*, (d) it is to discover the best route among the three ways, (e) it is to reach point *x* through the t route, and (f) it is to reach point *x* through the *z* route

3. RESULTS AND DISCUSSION

In this section, Dijkstra's algorithm is applied to searching and measuring the route from single-source to another single-source at the shortest route at the nonnegative edge. The working of Dijkstra's algorithm is shown below, where in figures the shorter route appears, start node and final node and shorter distance. Figures 3(a)-(b) to Figures 10(a)-(b) show all the simulated experimental results and the comparison of the results over the route in the NaN case, *num_nodes*, L and *max_seg_length* with their values. These figures accurately show the mechanism of action in drawing directions in a realistic way to determine the smallest paths between cities.

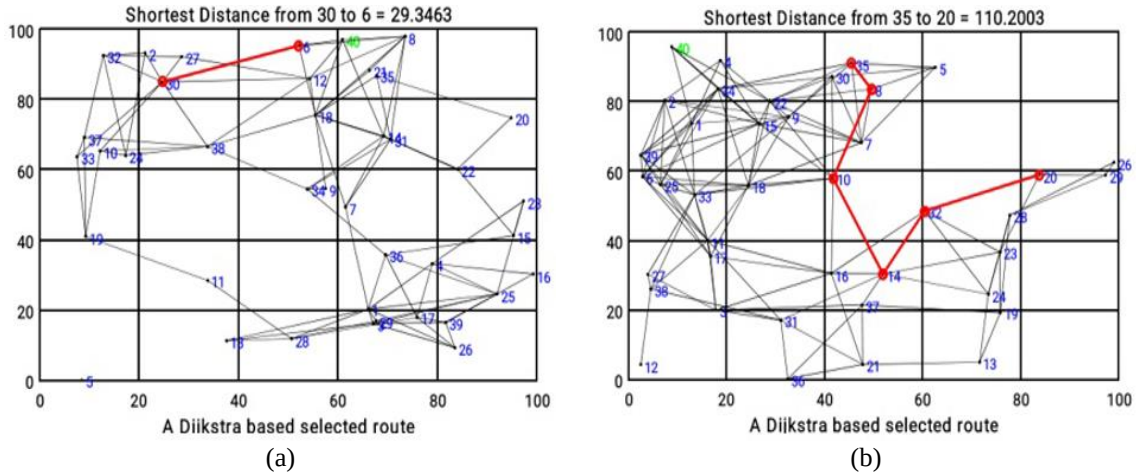
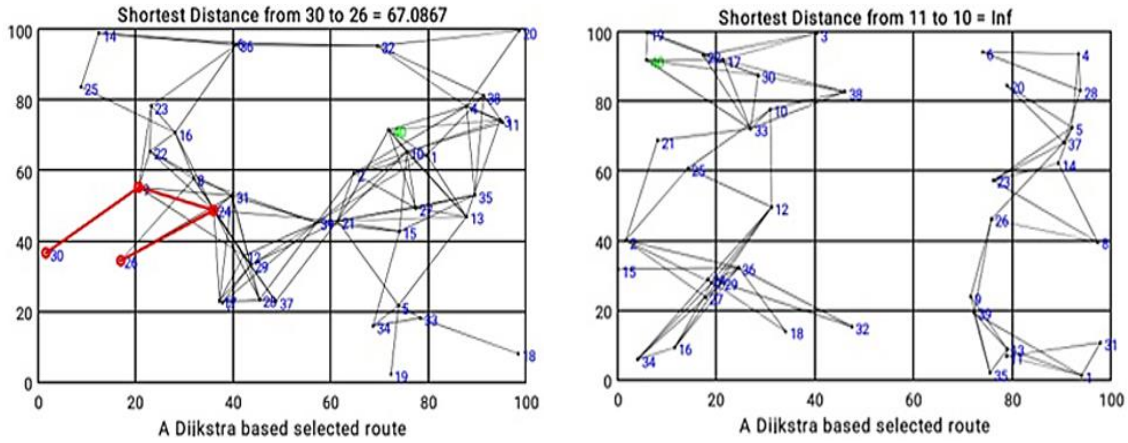
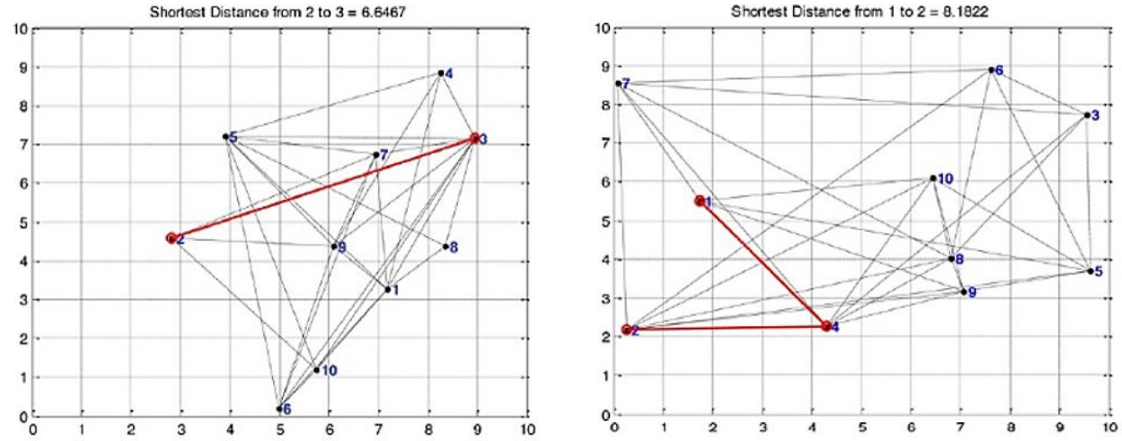


Figure 3. A Dijkstra based selected route, (a) NaN case. no_nodes=40; L=100; max_seg_length=40 and (b) NaN case. no_nodes=40; L=100; max_seg_length=40



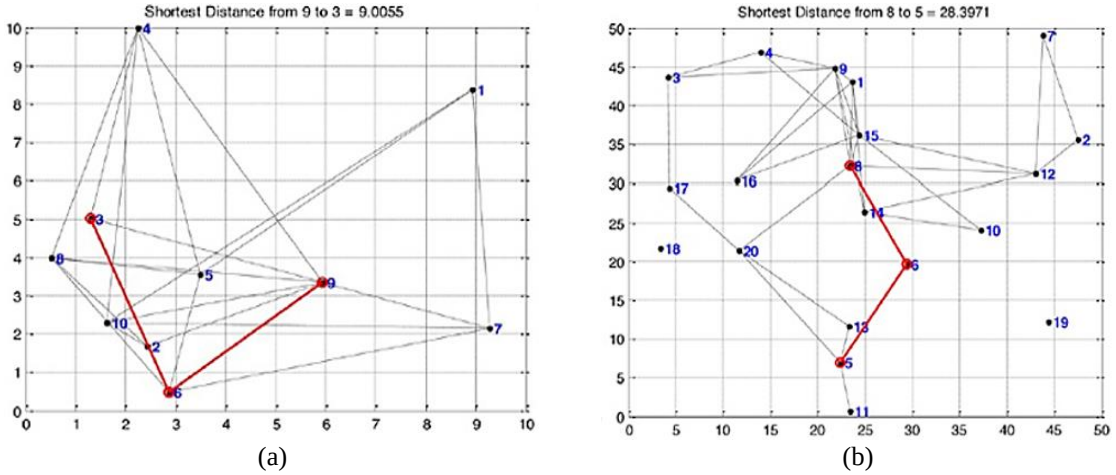
(a) (b)

Figure 4. A Dijkstra based selected route, (a) NaN case. no_nodes=40; L=100; max_seg_length=4 and (b) NaN case. no_nodes=40; L=100; max_seg_length=40



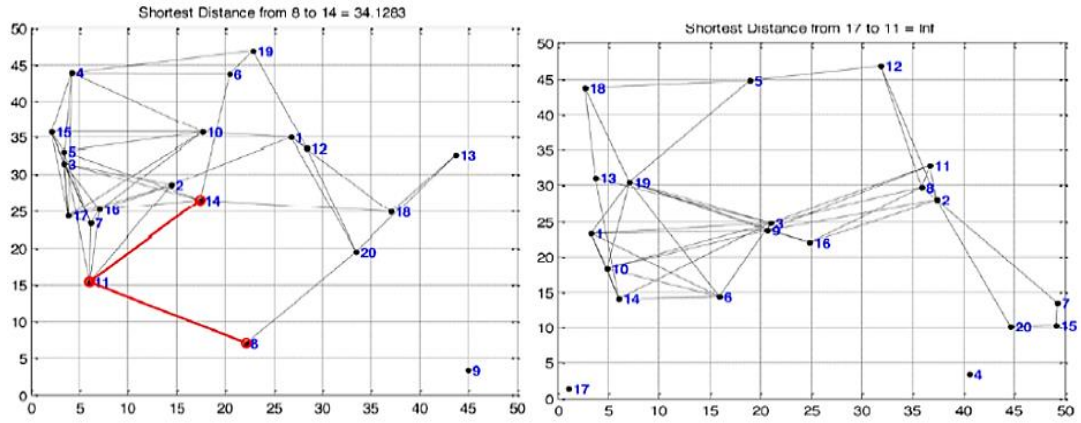
(a) (b)

Figure 5. A Dijkstra based selected route, (a) NaN case. no_nodes=10; L=10; max_seg_length=15 and (b) NaN case. no_nodes=10; L=10; max_seg_length=15



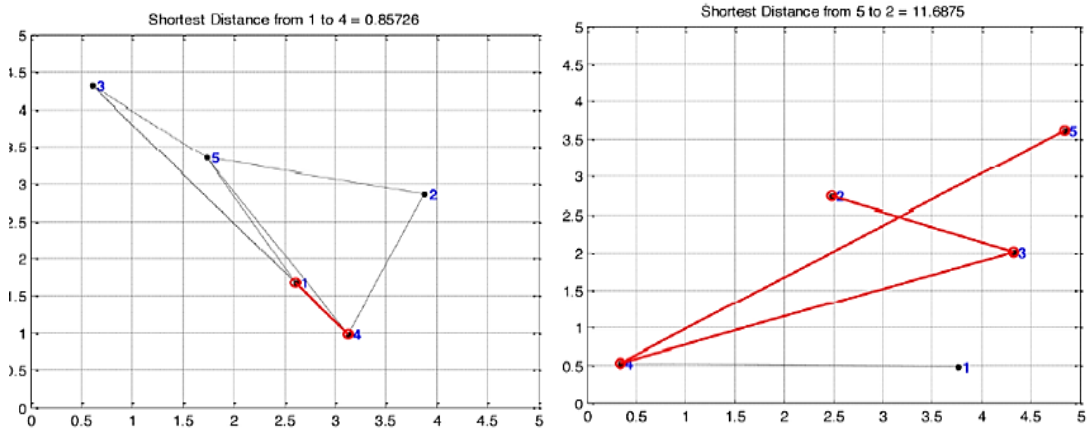
(a) (b)

Figure 6. A Dijkstra based selected route, (a) NaN case. no_nodes=10; L=10; max_seg_length=15 and (b) NaN case. no_nodes=20; L=50; max_seg_length=20



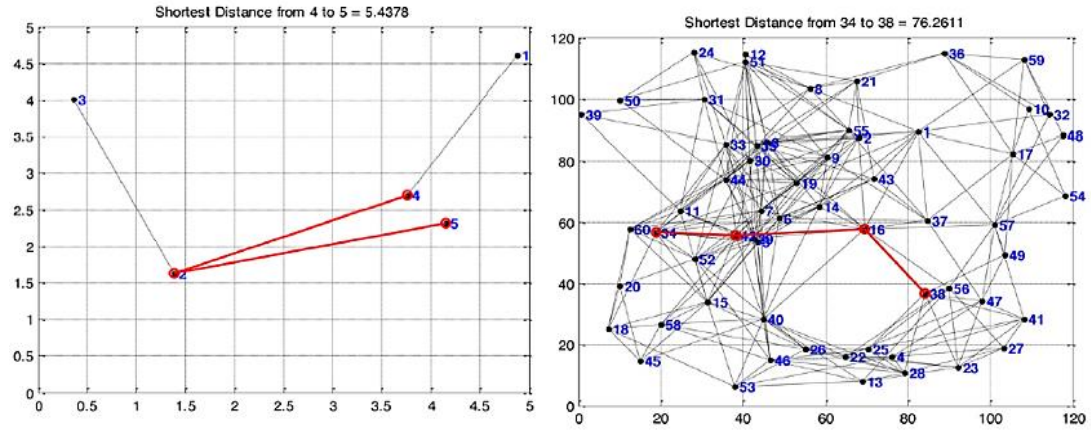
(a) (b)

Figure 7. A Dijkstra based selected route, (a) NaN case. no_nodes=20; L=50; max_seg_length=20 and (b) NaN case. no_nodes=20; L=50; max_seg_length=20



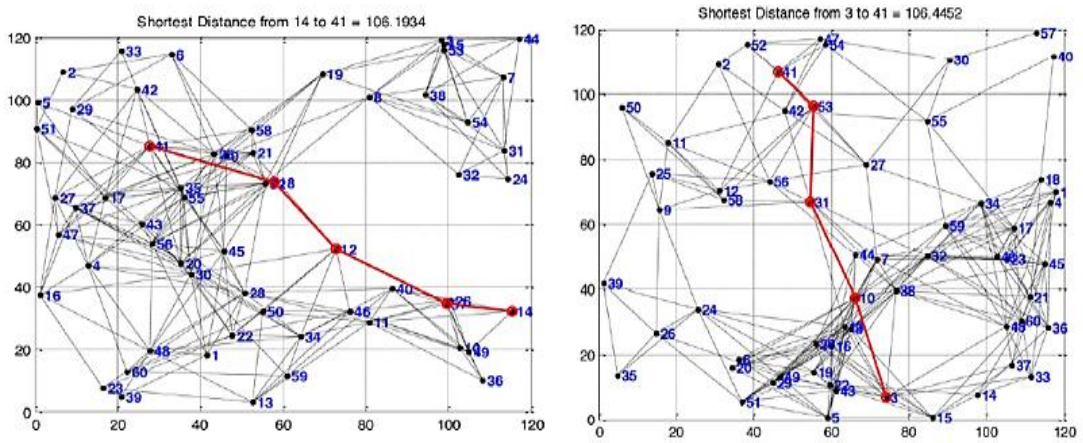
(a) (b)

Figure 8. A Dijkstra based selected route, (a) NaN case. no_nodes=5; L=5; max_seg_length=20 and (b) NaN case. no_nodes=5; L=5; max_seg_length=20



(a) (b)

Figure 9. A Dijkstra based selected route, (a) NaN case. no_nodes=5; L =5; max_seg_length=20 and (b) NaN case. no_nodes=60; L=120; max_seg_length=40



(a) (b)

Figure 10. A Dijkstra based selected route, (a) NaN case. no_nodes=60; L=120; max_seg_length=40 and (b) NaN case. no_nodes=60; L=120; max_seg_length=40

4. CONCLUSION AND FUTURE SCOPE

In this scenario, the Dijkstra's algorithm has been updated to discover the shortest and best route between cities and in less time (seconds) to organize flights, as maps of 10 to 50 cities are used. This work has been applied to the MATLAB 2018R environment with a computer: Windows 10, Intel i7, CPU 3.4-GHz, and 6-GB RAM. Moreover, the results of this scenario show an accuracy of more than 95% in 3.45 seconds for 50 cities, while a result of 97% for 10 cities with a period of fewer than 10 seconds is achieved. As the number of cities is increasing, the results remain best and the updated algorithm is not changed. This proposed and improved algorithm is characterized by the speed of performing the work and covering all the node points to connect the node to be reached accurately and at high speed. In the future, the authors suggest implementing this algorithm on Python environment with using 3D technology to give very accurate results with high-resolution graphics. One of the authors' future plans is to implement this idea on more than one algorithm and to make a comparison between the results that will be obtained from the simulation systems. This study is just the beginning of other works to come in future.

REFERENCES

[1] G. O'Regan, "Edsger Dijkstra," Springer London, 2013, pp. 91–97.
[2] M. Sniedovich, "Dijkstra's algorithm revisited: The dynamic programming connexion," *Control and Cybernetics*, vol. 35, 2006.
[3] D. P. Bertsekas, "Robust shortest path planning and semicontractive dynamic programming," *Naval Research Logistics (NRL)*, vol. 66, no. 1, pp. 15–37, Feb. 2019, doi: 10.1002/nav.21697.
[4] S. Sanan, L. Jain, and B. Kappor, "Shortest path algorithm," *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, vol. 2, no. 7, pp. 316–320, 2013.
[5] Y. Huang, Q. Yi, and M. Shi, "An improved Dijkstra shortest path algorithm," *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), Published by Atlantis Press*, Paris, France, 2013, doi: 10.2991/iccsee.2013.59.
[6] K. Uppalancha, "Optimizing the robot's path using dijkstra algorithm," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 04, pp. 4423–4430, 2015, doi: 10.15680/IJIRSET.2015.0406050.
[7] A. Ebrahimnejad, Z. Karimnejad, and H. Alrezaamiri, "Particle swarm optimisation algorithm for solving shortest path problems with mixed fuzzy arc weights," *International Journal of Applied Decision Sciences*, vol. 8, no. 2, 2015,Art. no. 203, doi: 10.1504/IJADS.2015.069610.
[8] G. Ertl, "Shortest path calculation in large road networks," *OR Spektrum*, vol. 20, no. 1, pp. 15–20, Mar. 1998, doi: 10.1007/BF01545524.
[9] P. Wanda, M. E. Hiswati, and H. J. Jie, "DeepOSN: Bringing deep learning as malicious detection scheme in online social network," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 1, pp. 146–154, Mar. 2020, doi: 10.11591/ijai.v9.i1.pp146-154.
[10] U. Nguyen and J. Xu, "Multicast routing in wireless mesh networks: Minimum cost trees or shortest path trees?," *IEEE Communications Magazine*, vol. 45, no. 11, pp. 72–77, Nov. 2007, doi: 10.1109/MCOM.2007.4378324.
[11] M. Gong, G. Li, Z. Wang, L. Ma, and D. Tian, "An efficient shortest path approach for social networks based on community structure," *CAAI Transactions on Intelligence Technology*, vol. 1, no. 1, pp. 114–123, Jan. 2016, doi: 10.1016/j.trit.2016.03.011.
[12] F. Bauer and A. Varma, "Distributed algorithms for multicast path setup in data networks," in *Proceedings of GLOBECOM '95*, vol. 2, pp. 1374–1378, doi: 10.1109/GLOCOM.1995.502627.
[13] A. Sarkar, "Multilayer neural network synchronized secured session key based encryption in wireless communication," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 1, pp. 44–53, Mar. 2019, doi: 10.11591/ijai.v8.i1.pp44-53.
[14] R. Atiqur and Y. Li, "Automated smart car parking system using raspberry Pi 4 and iOS application," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 9, no. 3, pp. 229–234, Nov. 2020, doi: 10.11591/ijres.v9.i3.pp229-234.

[15] A. D. Savio and V. J. A., "Development of multiple plug-in electric vehicle mobile charging station using bidirectional converter," *International Journal of Power Electronics and Drive Systems (IJPEDS)*, vol. 11, no. 2, pp. 785–791, Jun. 2020, doi: 10.11591/ijpeds.v11.i2.pp785-791.
[16] Huijuan Wang, Yuan Yu, and Quanbo Yuan, "Application of Dijkstra algorithm in robot path-planning," in *2011 Second International Conference on Mechanic Automation and Control Engineering*, Jul. 2011, pp. 1067–1069, doi: 10.1109/MACE.2011.5987118.
[17] C. Jain and K. Jitendra, "Processing Delay Consideration in Dijkstra's Algorithm," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 8, pp. 407–411, 2013.
[18] L. Sujatha and D. Hyacinta, "The shortest path problem on networks with intuitionistic fuzzy edge weights," *Glob J Pure Appl Math*, vol. 13, no. 7, pp. 3285–3300, 2017.
[19] M. A. Qureshi, M. F. Hassan, S. Safdar, and R. Akbar, "AO (E) time shortest path algorithm for non negative weighted undirected graphs," *arXiv preprint arXiv:0911.0174*, 2009.
[20] P. W. Yanliang L., "Optimization model of complicated network and shortest path algorithm," *Journal of Tianjin University of Technology*, vol. 22, pp. 42–44, 2006.
[21] W. Peng, X. Hu, F. Zhao, and J. Su, "A fast algorithm to find all-pairs shortest paths in complex networks," *Procedia Computer Science*, vol. 9, pp. 557–566, 2012, doi: 10.1016/j.procs.2012.04.060.
[22] P. Tirastittam and P. Waiyawuththanapoom, "Public transport planning system by dijkstra algorithm: Case study bangkok metropolitan area," *World Academy of Science, Engineering and Technology International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, vol. 8, no. 1, pp. 54–59, Jan. 2014.
[23] H. Arslan and M. Manguoğlu, "A hybrid single-source shortest path algorithm," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 27, no. 4, pp. 2636–2647, Jul. 2019, doi: 10.3906/elk-1901-23.
[24] I. Chabini and S. Lan, "Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 60–74, Mar. 2002, doi: 10.1109/6979.994796.
[25] O. S. Al Mushayt, W. Gharibi, and N. Armi, "Multicast routing protocol for advanced vehicular ad hoc networks," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, no. 3, pp. 1096–1100, Jun. 2019, doi: 10.12928/telkomnika.v17i3.10240.
[26] M. A. Javaid, "Understanding Dijkstra algorithm," *SSRN Electronic Journal*, 2013, doi: 10.2139/ssrn.2340905.
[27] X. Cui and H. Shi, "A*-based pathfinding in modern computer games," *International Journal of Computer Science and Network Security*, vol. 11, no. 1, pp. 125–130, 2010.
[28] et al. Jiechen W., "United structure of point-Arc for network graph and it's application in GISs shortest path searching," *Acta Geodaetica et Cartographica Sinica*, vol. 29, pp. 47–51, 2000.
[29] Q. Wang, X. Lu, X. Zhang, Y. Deng, and C. Xiao, "An anticipation mechanism for the shortest path problem based on Physarum polycephalum," *International Journal of General Systems*, vol. 44, no. 3, pp. 326–340, Apr. 2015, doi: 10.1080/03081079.2014.997532.
[30] Z. Zhang, C. Gao, Y. Liu, and T. Qian, "A universal optimization strategy for ant colony optimization algorithms based on the Physarum -inspired mathematical model," *Bioinspiration & Biomimetics*, vol. 9, no. 3, Mar. 2014, Art. no. 036006, doi: 10.1088/1748-3182/9/3/036006.