Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization

Article *in* IOP Conference Series Materials Science and Engineering · September 2020

DOI: 10.1088/1757-899X/917/1/012077

5 authors, including:

Samah Abusalim
Universiti Teknologi PETRONAS
4 PUBLICATIONS **12** CITATIONS
SEE PROFILE

Rosziati Ibrahim
Universiti Tun Hussein Onn Malaysia
124 PUBLICATIONS **721** CITATIONS
SEE PROFILE

Mohd Zainuri Saringat
Universiti Tun Hussein Onn Malaysia
27 PUBLICATIONS **139** CITATIONS
SEE PROFILE

Jahari Abdul Wahab
9 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization

IOP Conf. Series: Materials Science and Engineering **917** (2020) 012077 doi:10.1088/1757-899X/917/1/012077

Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization

Samah W.G. AbuSalim[1], Rosziati Ibrahim[1], Mohd Zainuri Saringat[1], Sapiee Jamel[1], Jahari Abdul Wahab[2]
[1]Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400 Johor, Malaysia.
(samahwa.salim@gmail.com)
(rosziati@uthm.edu.my, zainuri@uthm.edu.my, sapiee@uthm.edu.my)
[2]Department of Engineering R&D, SENA Traffic Systems Sdn. Bhd., Kuala Lumpur, Malaysia.
(jahari@senatraffic.com.my)

Abstract. Due to the tremendous development in the field of computer and software sciences, the theory of graphics has spread widely and quickly, even becoming one of the most important sciences that played a large role in solving many problems of many diverse applications. These applications include computer protocols, Google Maps, games and more. Many researches have discussed shortest path algorithms to solve the shortest path problem in these applications. In this study, a very popular algorithms called Dijkstra algorithm and Bellman-Ford algorithm are used to make a comparison between them on the basis of complexity and performance in terms of shortest path optimization. Our results show that Dijkstra is better than the Bellman-Ford interms of execution time and more efficient for solving the shortest path issue, but the algorithm of Dijkstra work with non-negative edge weights.

1. Introduction

The graph theory and network analysis are one of the most important topics dealt with in operations research. It has been launched that any system can be represented by a group of nodes, where some of these pairs of nodes are related to specific relationships that we express with lines that connect these pairs of points. This idea led to the emergence of the so-called graph theory. The reason for the development of this theory and its interest in it is due to its applicability in various and varied fields. And given the wide range of problems and the huge and varied number of issues dealt with in the graph theory, this theory played and is still playing a big role in dealing with many problems. And the issue of the shortest path is considered one of the most important issues in graph theory.

Research for finding the best algorithms for handling the issue of shortest path has been on going. In the real world it is easy to apply the graph theory to different types of scenarios. In shortest path algorithm, the study focuses on two nodes or vertices of the path and find the best solution for the shortest path. Several algorithms are frequently used to discover the shortest path between graph nodes. For example, Bellman-Ford and Dijkstra are the most effective algorithms for single-source shortest path issue. For dense graph, the Floyd-Warshall is mostly used to discover the shortest path for all pairs and the Johnson algorithm is the best for sparse graphs. The process of these algorithms is time-consuming because first, each algorithm needs to explore the entire graph and calculate the shortest path from each node.

This paper will discuss the two algorithms namely Dijkstra and Bellman-Ford for its strength in finding the shortest path. The structure of the paper is as follow. Section 2 covers the area of related work for the algorithms for the shortest path and Section 3presents the methods which include Dijkstra, Bellman-Ford algorithms and the representation of this algorithm in Z notation. Section 4 demonstrates the results for the case study of shortest path optimization system and finally, Section 5give the concluding remarks for the paper.

2. Algorithms for Shortest Path

Shortest path algorithms are mainly separated into two wide groups. The first group is called the single source shortest path (SSSP). Purpose of SSSP is to find out the distinguished from the single supply vertex to all different vertices. The second group is called all pair shortest path (APSP) and the

purpose of this path is to dig out the smallest path among all sets of vertices shown in the graph [1]. The calculation of this smallest path could be exact or roundabout solution. The use of the best algorithm is depending on the features and requirements of the program. Like, the purpose of the shortest path algorithm is to generate the quick reply even in the existence of a huge input graph. From between these two algorithms, Dijkstra and Bellman's ford depend to get the fast and efficiently smallest path between two vertices. Java languages are used to find the shortest path along with requirements. The name of the program is mentioned as shortest path optimization system and to elaborate its functions from the Unified Modelling Language (UML) class diagram java language is used.

Oyola *et al.* [2] proposed an approach called Safe and Short Evacuation Routes (SSER) by using Dijkstra-based algorithm to address the issue of finding the shortest secure paths in houses with severalwithdrawal gates and where distinct types of sensors could change the accessibility of inner areas. In order to check the potency of the recommended strategy, four Dijkstra-based algorithms were regarded, obtaining brief evacuation times to various exits. This strategy makes the algorithm appropriate for use in dynamic settings where distinct types of sensors can change the accessibility status of inner areas.

Dinitz and Itzhak [3] presented a newhybrid algorithm called Bellman-Ford–Dijkstra (BFD) by combining Bellman-Ford and Dijkstra algorithms. This algorithm for locating the shortest paths from a source node s in a graph G with general edge costs.To improve the runtime of Bellman-ford algorithm and a sparse sharing of the destructive cost edges. They suggested A hybrid of Bellman-Ford and Dijkstra algorithms.The principle of the proposed algorithm is to repeat the Dijkstra algorithm multiple times without resetting the temporary value of d (v) to the vertices. They also suggested a new and simple proof that the Bellman–Ford algorithm can generate the shortest paths tree.

Singh and Tripathi [4] made comparative between two algorithms: Bellman-Ford and Dijkstra's and discussed the results based on the number of nodes. The research enables to define and suggest which algorithm is used in the shortest path problems for a specific variant. Their results show that very minimum quantity of nods, Bellman Ford algorithm is superior over Dijkstra's algorithm but Dijkstra is most effective for a massive variety of nodes.

Ryash and Tamimi [5] described and compared between the algorithms: Johnson's, Dijkstra's, Floyd-Warshall and Bellman-Ford for solving the shortest path problem. Their search shows that Dijkstra algorithm is more effective in memory for sparse graphs because it does not involve the representation of the distance matrix as a dense matrix. The complexity of Bellman-Ford algorithm with respect to time is slower than the algorithm of Dijkstra. Dijkstra's algorithm can be used only with non-negative edge weights graphs. The algorithm of Johnson is quicker than Floyd – Warshall when the graph is sparse, but on the densegraph, the Floyd–Warshall algorithm is faster.

Lacorte and Chavez [6] provided an analysis of apply both algorithms A* and Dijkstra in the projection of route optimization model for development of smart school transport system. Each algorithm was tested using a tool called EESCOOL. The results proved that algorithm A* performs better and produce very minimum expected time of arrival (ETA) while routine wise traffic on a small graph. Unlike the Dijkstra’s algorithm that performs better in heavy traffic and on a small or large graph with shorter projected time of appearance (ETA).

Permana *et al.* [7] compared three algorithms namely A*, Breadth First Search (BFS)and Dijkstra in order to find which algorithm is appropriate to apply in Maze Runner game. Maze Runner is a game that needs a path-finding algorithm with the shortest path to reach the target. Comparison approach of these algorithms was once performed by way of changing the game algorithm and by way of

evaluating technique time, route size, and block numbers played in the current computing system. Their result showed that A* is the best pathfinding algorithm, particularly in the game/grids of Maze.

Wang [8] compared between three algorithms that are Dijkstra, Bellman-Ford and Floyd–Warshall algorithms based on time and space complexity. Their analysis showed that the Dijkstra algorithm is only useful in the shortest route issue of a single source and it always optimized in a real implementation, such as heap optimization. The implementation of Bellman-Ford algorithm is simple but the algorithm itself is inefficient for finding shortest path. The Floyd-Warshall algorithm is the slowest and inefficient space for redundancy while working with an oversized different kind of points and edges.

Abbas *et al.* [9] presented an algorithm named Caption algorithm to discover a solution to the shortest path trouble compared to the Dijkstra algorithm with reduced time complexity. Their proposed algorithm can be an alternativeto Dijkstra's algorithm because their algorithm has the ability to repeat the search process by increasing the reduction coefficient. The Capton algorithm's effectiveness and performance are better than the Dijkstra's algorithm on real-world examples such as discovering the shortest route between towns. It can be used in constructing new shortest routes by visualizing and manipulating missing paths in design-related tools and models.

Chan et al. [16] compared and conducted an experiment for six shortest path algorithms which are: Dijkstra's, Symmetrical Dijkstra's, A*, Bellman-Ford, Floyd-Warshall and Genetic Algorithm in order to determine the one of most competent algorithms to resolve the shortest path problem. Experiments have shown that the Bellman algorithm is superior among other algorithms as it produces the optimal solution in a short time. On the other hand, the genetic algorithm has also been shown to produce the optimal solution, but at a higher running time. Also, the number of generations of a genetic algorithm affects its performance. The more generations, the higher the operating times and the better solution.

Sapundzhi and Popstoilov [17] evaluated the Dijkstra's algorithm, Floyd-Warshall algorithm, Bellman-Ford algorithm, and Dantzig's algorithm in resolving the shortest path problem, that can be applied to very different biological systems and problems for biological system modelling. A brief overview of each algorithm was presented, as well as the C# application for each of them to show how each algorithm works. The result shows that the complexity of time regarding shortest path algorithms depends on the quantity of vertices, number of edges and edge length. For time complexity the Bellman-Ford algorithm was higher than Dijkstra's algorithm. For higher number of nodes, the Dijkstra's algorithm is better and efficient.

Kwiecie and Pasieka [18] suggested using an algorithm called cockroach swarm optimization (CSO) to define paths and modified them to work with the extended time model. The results concluded that the performance of their approach is satisfactory in terms of convergence of best solutions. The results of this algorithm were compared with the results of both Dijkstra algorithm and PSO (Particle Swarm Optimization) algorithm. By comparison, it was concluded that the CSO algorithm takes more computational time and is very slow to solve large problems. Also, the results for this algorithm were not the same as the results obtained with the Dijkstra algorithm, but the CSO algorithm is better than the PSO in terms of the best time to travel.

Ramadhan et al. [19] compared between two algorithms: Prim and Floyd-Warsall and also to demonstrate that the Prim algorithm can be used to solve the shortest path problem and is not used to solve the Minimum Spanning Tree problem. The results demonstrated that by generating the Prim algorithm a less active number of vertices, it makes them faster in determining the optimal path and decision-making. While the Floyd-Warshall algorithm is better able to determine the shortest path, but not the optimal one.

The summary of the comparison of the shortest path algorithms are listed in Table 1.

Table 1: The comparison of shortest path algorithms

Author and Year	Algorithm	Method
Oyola *et al.* [2]	Dijkstra-based algorithm	The suggested strategy makes the algorithm appropriate for use in dynamic settings where distinct types of sensors can change the accessibility status of inner areas.
Dinitz and Itzhak [3]	Bellman_Ford and Dijkstra algorithms	The new proposed Bellman_Ford algorithm can produce the shortest paths tree.
Singh and Tripathi [4]	Bellman_Ford and Dijkstra algorithms	The Bellman_Ford algorithm deals more effective with a small quantityof nodes, and the Dijkstra algorithm is greater efficient for a large wide of nodes.
Ryash and Tamimi [5]	Dijkstra, Bellman_Ford, Floyd_ Warshall, and Johnson algorithms	Dijkstra-based algorithm'scomplexity is faster than the Bellman_Ford algorithm with respect to time. The algorithm of Johnson is faster than Floyd_Warshall when the graph is sparse, but on the dense graph, the Floyd–Warshall algorithm is faster.
Lacorte and Chavez [6]	A* and Dijkstra algorithms	The algorithm of A* performs better than Dijkstra's algorithmas expected arrival time (ETA) is shorter on a small graph but Dijkstra's algorithm produce shorter ETA in both small and big graphs.
Permana *et al.* [7]	A*, Breadth FirstSearch and Dijkstra algorithms	A* can be considered as the good path finding algorithm, especially in the game/grids of Maze.
WANG [8]	Dijkstra, Bellman_Ford and Floyd_Warshall algorithms	Bellman-Ford algorithm is simple. Floyd-Warshall algorithm is very slow among the three algorithms. Dijkstra algorithm can only be useful in the shortest route issueof a single source.
Abbas et al [9]	Caption algorithm	The new proposed algorithm effectiveness and performance is better than the Dijkstra's algorithm on real-world examples such as discovering the shortest route between towns.

3. The Z Specification and Implementation

In this Section, we describe the Z specification and the implementation of the algorithms for Dijkstra and Bellman-Ford in solving issues of the shortest path.

3.1 Dijkstra Algorithm

Dijkstra algorithm is designed by Dutch computer scientist Edsger Dijkstra in 1956 and then published in 1959 [10, 11] based on the graph that is used to resolve the difficulties in single-source shortest path and producing a shortest path tree.Dijkstra's algorithm to determine the shortest path has important applications that we use on a daily basis. The data that is transmitted over the Internet passes different paths around the world so it is important that it is routed through short and less crowded paths and this is possible through the Dijkstra algorithm. Also, the OSPF (Open Shortest Path First) routing protocol that is used in Internet Protocol (IP) based on Dijkstra's algorithm [20]. Also, by using this algorithm, Google Maps can now specify the shortest path that we can take to go from a specific point to a specific restaurant or airport.It is an instance of an effective greedy algorithm. Greedy algorithm is

considered the perfect general solution to certain optimization problems, but can find less than optimal alternatives to other issues in some cases.One of the most important advantages of Dijkstra's algorithm is that it does not visit the remaining unwanted nodes when the intended destination node is reached [16].On the other side, the drawback ofDijkstra's algorithm is consuming a lot of CPU memory when executed on a very large number of nodes on a computer program [21]. Also, it cannot handle negative edges, its application is done only on positive weight graphs. In the Dijkstra algorithm, the path is not known. The nodes are divided into two groups: temporary (t) and permanent(p). First of all, initialize the distance of source node to the zero value [distance(a) = 0], and assign the distance for other nodes to value infinity [distance(x) = ∞]. Step 2, search for node x with the smallest value of d(x). If no temporary nodes exist or the value of d(x) equal to infinity, the node x was labeled as permanent that means that the d(x) and parent of d(x) will not change anymore. Step 3, apply the following comparison for each temporary node labelled vertex y adjacent to x:

If $d(x) + w(x, y) < d(y)$ then

$$D(y) = d(x) + w(x, y) \quad (1)$$

Based on equation (1), if the distance of labelled node x plus link weight (x, y) is less than the labelled node y distance, then the distance of labelled node y will be updated. Algorithm 1 presents the Dijkstra algorithm.

Algorithm 1 Dijkstra Algorithm

```
1: function DIJKSTRA (Graph, source)
2:     create vertex set D
3:     for each vertex v in Graph:
4:         distance[v] ← INFINITY
5:         previous[v] ← UNDEFINED
6:         add v to D
7:     distance[source] ← 0
8:
9:     while D is not empty do:
10:            u ← in D with min distance[u]
11:            remove u from D
12:            for each neighbour v of u:
13:        alt ← distance[u] + length (u, v)
14:        if alt < distance[v]
15:            distance[v] ← alt
16:            previous[v] ← u
17:    end while
18:    return distance [], previous []
19: end function
```

The process of this algorithm is more clearly described in Figure 1 and Table 2. Figure 1 shows the example of the nodes being travest based on the Dijkstra algorithm. The results are shown in Table 2.

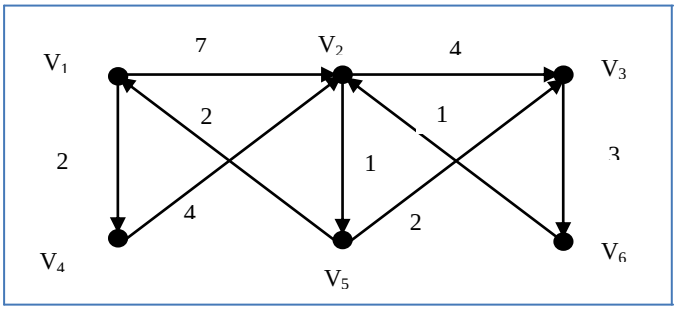

Figure 1: Dijkstra algorithm example

Table 2: Dijkstra algorithm result

S	V_1	V_2	V_3	V_4	V_5	V_6
{}	0	∞	∞	∞	∞	∞
{V_1}	0	7	∞	2	∞	∞
{V_1, V_4}	0	6	∞	2	∞	∞
{V_1, V_4, V_2}	0	6	10	2	7	∞
{V_1, V_4, V_2, V_5}	0	6	9	2	7	∞
{V_1, V_4, V_2, V_5, V_3}	0	6	9	2	7	12
{V_1, V_4, V_2, V_5, V_3, V_6}	0	6	9	2	7	12

3.2Bellman-Ford Algorithm

Richard Bellman released the Bellman-Ford algorithm in 1958 [12]. Bellman-Ford algorithm applied to search the shortest path when some of the edges of the directed graph G may have negative weight from the source node to all other nodes [22]. Graphs with negative edge weights are difficult to resolve by using the algorithm of Dijkstra [5]. The algorithm of Dijkstra is faster than the Bellman-Ford algorithm, but more versatile is the Bellman-Ford algorithm. This algorithm, like Dijkstra's Algorithm, use of the idea of area relaxation but doesn't use with greedy technique [13]. The advantages of this algorithm are it a dynamic algorithm [23], it can calculate the negative directed edges (in addition to the positive), can minimize the cost when network was built, because it can finds the shortest path from one node to another, so we don't have to build a lot of router path. Also, this algorithm is simple and it does not need complicated data structures for applications and can find the minimum path weight with high efficiency and accuracy [24]. The disadvantages of the Bellman algorithm when used in the Routing Information Protocol (RIP) are that it does not take into account weight and also a slow response to changes in network topology resulting from slow updates passed from the RIP device to the next device. These flaws lead to an attempt to use idle tracks that waste time and network resources. [20]. Bellman-Ford algorithm returns a Boolean value to indicate whether a negative cycle can be reached by the origin. If no such cycle occurs, the algorithm returns the shortest path, if a negative cycle exists, the algorithm reveals that there is no shortest path [5]. The Bellman-Ford algorithm is executed as shown in Algorithm 2. Step 1, set the distance of source vertex s to zero value (distance[s] = 0) and assign other vertices distance with INFINITY. Step 2 relaxes each edge for (n − 1) times when n is the numbers of nodes. Relaxing an edge means checking to see if it is possible to shorten the route to the node to which the edge points and, if so, replace the route to the node with the route found [14, 15]. Step 3: check if the graph has any negative cycle with execute the Nth loop. Algorithm 2 presents the Bellman-Ford algorithm.

Algorithm 2 Bellman-Ford Algorithm

```
1: function bellmanFord (G, S)
2:     for each vertex V in G
3:         distance[v] ← INFINITY
4:         previous[v] ← NULL
5:     distance[s] ← 0
6:     for each vertex V in G
7:         for each edge (u, v) in G
8:             alt ← distance[u] + length (u, v)
9:             if alt < distance[v]
10:                distance[v] ← alt
11:                previous[v] ← u
12:
13:  for each edge (u, v) in G
14:      if distance[u] + length (u, v) < distance(v)
15:          Error: Negative Cycle Exists
16: return distance [], previous []
```

The process of this algorithm is more clearly in Figure 2 and Table 3. Figure 2 shows the example of travest between nodes based on Bellman-Ford algorithm.

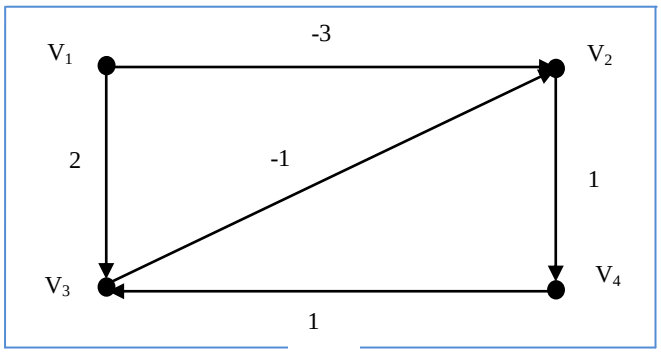

Figure 2: Bellman-Ford algorithm example

Table 3: Bellman-Ford algorithm result

i	V1	V2	V3	V4
0	0	∞	∞	∞
1	0	-3	2	∞
2	0	-3	2	-2
3	0	-3	-1	-2

3.3The Representation of the Algorithms in Z

Z notation is a collection of conventions for presenting mathematical text, chosen to make it easy to explain computer systems using simple mathematics. We implement the two algorithms as shown in figure 3 and 4. Using a formal notation enhances the knowledge of a system's operation, particularly early in a model. It helps organize a designer's ideas, making clearer, easier designs and the probability of mistakes is decreased. Some researchers are using formal specification instead of Z specification such as [25], [26], [27] and [28].For these two algorithms, Figure 3 shows the Z notation for the Dijkstra algorithm and Figure 4 shows the Z notation for the Bellman-Ford Algorithm. Once the Z notations are developed, the algorithms are then implemented using Java programming language to find the shortest path between two nodes.

Dijkstra

G? W? S? Q? V?, out !: z

S? = ⊘
Q = G?. V?
While Q ≠ ⊘
u? = EXTRACT -MIN(Q)
S? = S U {u}
for each vertex v Є G. Adj [u]
RELAX (u, V, W)

Figure 3: Z representation of the Dijkstra algorithm

Bellman-Ford

G? W? S? E? , V?, out !: z

for i = 1 to |G. V| - 1
for each edge (u, v) Є G.E
RELAX (u, v, w)
for each edge (u, v) Є G.E
if v. d > u. d + w (u. v)
return FALSE
return TRUE

Figure 4: Z representation of the Bellman-Ford algorithm

Based from Figures 3 and 4, we implemented the simple system to find the shortest path. Figure 5 shows the UML use case diagram for the Shortest Path Optimization.

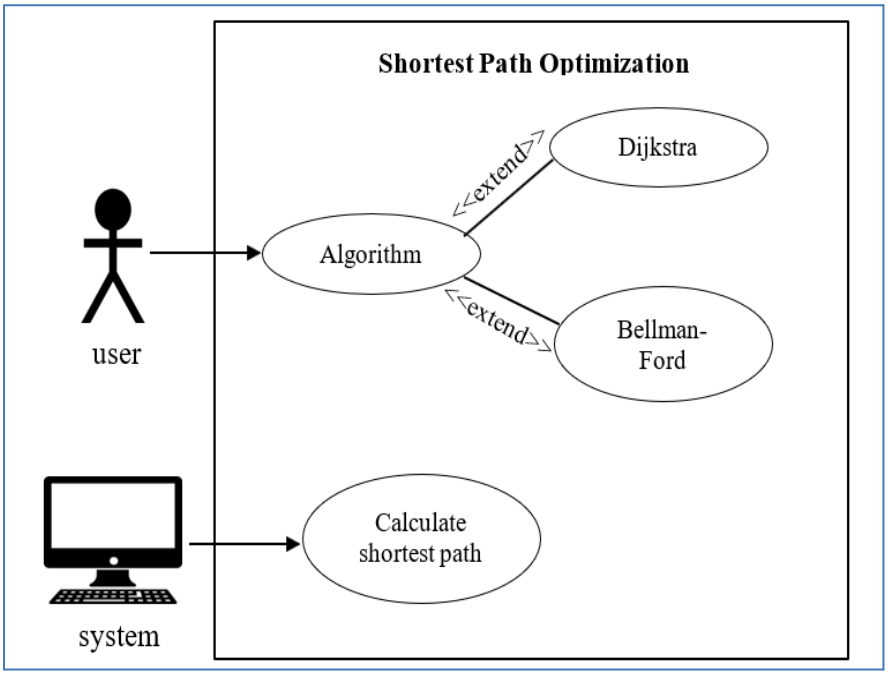

Figure 5: UML Use Case Diagram for the Shortest path Optimization

Based from Figure 5, the user selects the algorithm that he wants to use to solve the shortest path problem and then the system calculate the shortest path amongtwo nodes using Dijkstra and Bellman-Ford algorithms.

4. Simulation Results

This Section discusses the results obtained after running the simulation of the two algorithms. For algorithms of Dijkstra and Bellman-Ford, they are used for finding the solution for single destination and single source with single pair of the shortest path problem. For Bellman-Ford algorithm, it works well with negative edges and is capable of detecting a negative edge cycle in a graph. However, for Dijkstra algorithm, it works well with positive weights in a directed or undirected graph. The performance of the Bellman-Ford algorithm is most effective when the graph has a small number of nodes, while Dijkstra's algorithm can work better when the graph has a large number of nodes. The results for the running time are shown in Table 4.

Table 4: Running time of the two algorithms

No of nodes	Dijkstra (in ms)	Bellman-Ford (in ms)
5	1351	513
10	3724	756
50	2072	9577

Based from Table 4, for Bellman-Ford algorithm, when the number of nodes is small, the running time is better than Dijkstra algorithm. However, Dijkstra algorithm has better running time when the number of nodes is larger. Our results are same to other research papers. For example, in [4], the authors showed that Dijkstra is most effective for a massive variety of nodes. In [17], for higher number of nodes, the Dijkstra's algorithm is better and efficient. Also, in [6], Dijkstra's algorithm emerged to produce shorter time in both small and big graphs.

Dijkstra's algorithm takes a time on a graph with edges E and vertices V can be represented using the Big-O notation as a function of |E| and |V|. Dijkstra's algorithm takes time 0(|E|+|V|Log|V|). And the time taken by the algorithm of Bellman-Ford is O (| V | ·| E |) as shown in Table 5. Based on the time complexity, we can conclude that Bellman-Ford algorithm takes more time than Dijkstra algorithm. The summary of the differences between the two algorithms is shown in Table 5.

Table 5: Comparison of the two algorithms

The Algorithm	Complexity for Time	Complexity for Space	Number of nodes
Dijkstra	0(|E|+|V|Log|V|)	O (V^2)	Large
Bellman-Ford	O (| V | ·| E |)	O (V^2)	Small

5. Conclusion

This paper presented the comparative analysis in term of shortest path optimization between two algorithms. The two algorithms are compared which are Dijkstra and Bellman-Ford algorithms to conclude which of them is more efficient for finding the shortest path between two vertices. Our results show that the Dijkstra algorithm is much faster than the algorithm of the Bellman ford and commonly used in real-time applications.

Acknowledgments

This project is funded by the Ministry of Education Malaysia under the Malaysian Technical University Network (MTUN) grant scheme Vote K234 and SENA Traffic Systems Sdn. Bhd.

6. References

[1] Madkour,A., Aref,W. G., Rehman,F. U., Rahman,M. A. and Basalamah, S. M. (2017). A survey of shortest-path algorithms. CoRR, vol. abs/1705.02044, 2017. [Online]. Available: http://arxiv.org/abs/1705. 02044.

[2] Oyola, A., Romero, D. G., & Vintimilla, B. X. (2017). A Dijkstra-Based Algorithm for Selecting the Shortest-Safe Evacuation Routes in Dynamic Environments (SSER). Lecture Notes in Computer Science, 131–135. doi:10.1007/978-3-319-60042-0_15.

[3] Dinitz, Y., & Itzhak, R. (2017). Hybrid Bellman–Ford–Dijkstra algorithm. Journal of Discrete Algorithms, 42, 35–44. doi:10.1016/j.jda.2017.01.001.

[4] Singh, J.B., Tripathi, R.C. (2018). Investigation of Bellman–Ford Algorithm, Dijkstra's Algorithm for suitability of SPP. IJEDR | Volume 6, Issue 1 | ISSN: 2321-9939

[5] AbuRyash, H.& Tamimi, A. (2015). Comparison Studies for Different Shortest path Algorithms. International Journal of Computers and Applications, Vol.14. 10.24297/ijct.v14i8.1857.

[6] Lacorte, A. M., & Chavez, E. P. (2018). Analysis on the Use of A* and Dijkstra's Algorithms for Intelligent School Transport Route Optimization System. Proceedings of the 4th International Conference on Human-Computer Interaction and User Experience in Indonesia, CHIuXiD '18 - CHIuXiD '18. doi:10.1145/3205946.3205948

[7] Permana, S. H., Bintoro, K. Y., Arifitama, B., & Syahputra, A. (2018). Comparative Analysis of Pathfinding Algorithms A*, Dijkstra, and BFS on Maze Runner Game. IJISTECH (International Journal Of Information System & Technology Vol. 1, No. 2, (2018), pp. 1-8).

[8] WANG, X.Z. (2018). The Comparison of Three Algorithms in Shortest Path Issue. IOP Conf. Series: Journal of Physics: Conf. Series 1087 (2018) 022011 doi :10.1088/1742-6596/1087/2/022011

[9] Abbas, Q., Hussain, Q., Zia, T. & Mansoor, A. (2018). Reduced Solution Set Shortest Path Problem: Capton Algoritm With Special Reference To Dijkstra's Algorithm. Malaysian Journal of Computer Science, [S.l.], v. 31, n. 3, p. 175-187, july 2018. ISSN 0127-9084.

[10] Hofner, P. andMoller, B. (2012). Dijkstra, Floyd and Warshall meet Kleene". Formal Aspects of Computing, 459–476.

[11] Huang, B, Wu, Q, and Zhan, F.B. (2007). A shortest path algorithm with novel heuristics for dynamic transportation networks. International Journal of Geographical Information Science. Vol. 21, No. 6, 625–644.

[12] Hutson, K.R., Schlosser, T.L., and Shier, D.R. (2007). On the Distributed Bellman-Ford Algorithm and the Looping Problem. Informs Journal on Computing. Vol 19, No 4, 542-551.

[13] Singh, J.B., Tripathi, R.C. "Investigation of Bellman–Ford Algorithm, Dijkstra's Algorithm for suitability of SPP" 2018 IJEDR | Volume 6, Issue 1 | ISSN: 2321-9939

[14] Hemalatha, S, and Valsalal, P. (2012).Identification of Optimal Path in Power System Network Using Bellman Ford Algorithms. Journal of Modeling and Simulation in Engineering, Hindawi Publishing Corporation, Article ID 913485.

[15] Patel, V.& Baggar, C. (2014). A survey paper of Bellman-ford algorithm and Dijkstra algorithm for finding shortest path in GIS application", International Journal of P2P Network Trends and Technology (IJPTT) – Volume 4 Issue 1 January to February 2014.

[16] Chan, S., Adnan, N., Sukri, S.S., & Zainon, W.M. (2016). An experiment on the performance of shortest path algorithm.Knowledge Management International Conference (KMICe) 2016, 29 – 30 August 2016, Chiang Mai, Thailand

[17] Sapundzhi, F. I., Popstoilov, M. S. (2018). Optimization algorithms for finding the shortest paths. Bulgarian Chemical Communications, Volume 50, Special Issue B, (pp. 115 – 120)

[18] Kwiecień, J., and Pasieka, M. Cockroach Swarm Optimization Algorithm for Travel Planning. Entropy 19.5 (2017): 213.

[19] Ramadhan, Z., Siahaan, A.P., Mesran, M. (2018) "Prim and Floyd-Warshall Comparative Algorithms in Shortest Path Problem", ICASI 2018, April 23-24, Medan, Indonesia.

[20] K, H. (2017) "A Graph Theory Algorithm To Find Shortest Path In Routing Protocol", International Journal of Scientific & Engineering Research Volume 8, Issue 5, May-2017, ISSN 2229-5518

[21] Aghaei, M.R.S., Zukarnain, Z.A., Mamat, A., Zainuddin, H., (2009) "AHybrid Algorithm for Finding Shortest Path in Network Routing",Journal of Theoretical and Applied Information Technology.

[22] Glabowski, M., Musznicki, B., Nowak, P., and Zwierzykowski, P., (2013) "Efficiency Evaluation of Shortest Path Algorithms", AICT 2013: The Ninth Advanced International Conference on Telecommunications.

[23] Rai, S., JB, A., Sharma, A. (2017) "Single Source Shortest Path Algorithms - Comparison Through Implementation", International Journal For Technological Research In Engineering, Volume 5, Issue 1, September-2017.

[24] Sanan, S., jain, L., Kappor, B. (2013) "Shortest Path Algorithm", International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 2, Issue 7, July 2013.

[25] Wong, S.Y., Mit, E. & Sidi, J. (2016). "Integration of use case formal template using mapping rules", Proceedings of 3rd International Conference on Information Retrieval and Knowledfe Management (CAMP2016).

[26] Aman, H., Ibrahim, R. (2014). "Formalization of Transformation Rules from XML Schema to UML Class Diagram",International Journal of Software Engineering and Its Application, *8*(12), pp.75-90, 2014.

[27] Mondal, B., Das, B. & Banerjee, P. (2014). "Formal Specification of UML Use Case Diagram – A CASL based Approach", International Journal of Computer Science and Information Technologies, Vol. 5 (3), 2014, pp. 2713-2717.

[28] Ibrahim, N., Ibrahim, R., Saringat, M.Z., Mansor, D and Herawan, T. (2011). "Consistency rules between UML use case and activity diagrams using logical approach." International Journal of Software Engineering and its Applications, 5 (3), pp. 119-134, 2011.