e-ISJN: A4372-3114
p-ISJN: A4372-3115

ISSN: 2321-7782 (Online)
ISSN: 2347-1778 (Print)
Impact Factor: 7.529

Volume 8, Issue 9, September 2020

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Using GIS to Determine the Shortest Distance in the Searching Engines

Al Bager A. Al Bager[1]
Faculty of Computer Science and Information Technology,
Neelain University,
Khartoum, Sudan

Al Samani A. Ahmed[2]
Faculty of Computer Science and Information Technology,
Neelain University,
Khartoum, Sudan

Abstract: Finding the shortest paths to reach a place is one of the main transportation problems in modern cities. Therefore, the current study aims to implement two algorithms: Dijkstra and BF to search the best route for reaching different destinations. The case study considered in this study involves finding the shortest path between two terminals in Bandung city. The calculations were performed by using graph and Haversine Formula, which further helped in calculating Dijkstra and Bellman-Ford Algorithm. The comparison of both the algorithms showed that Dijkstra's algorithm was able to produce a shorter path, as compared to the Bellman-Ford Algorithm.

Keywords: Bellman-Ford Algorithm, Dijkstra Algorithm, Geographical Information System, Search Engine, Shortest Path.

I. INTRODUCTION

Geographical Information System (GIS) has the ability to handle georeferenced data which is mainly concerned with the geographic coordinates (latitude and longitude). The association between geographic data and socio-economic data (population density) is strengthened by GIS as it generates thematic maps (Jiang et al., 2010). Rodríguez-Torres and Rodríguez-Puente (2010) described a technique to generate these types of thematic maps, and further identified their importance in facilitating the process of decision making with an increased social impact.

Usually, large road networks with thousands of streets are utilized by GIS for responding to specific requests. Therefore, it is important to focus on the mechanism of information processing (Rodríguez-Puente & Lazo-Cortés, 2013). The search of the shortest route of travel from one point to another is one of the most import reasons for implementing GIS. The researchers extensively use Dijkstra algorithm (also known as single source shortest path algorithm), which is one of the popular techniques used for solving problems. Some of the important reasons of implementing GIS is that it searches for the shortest evacuation route (Miler et al., 2013), finds route leading to nearest parking area, and assist construction workers in organizing exit routes (Soltani et al., 2002). The organization of exit routes includes vehicle carrier materials for accelerating the distribution of building materials and minimizing the risk of accidents (Miler et al., 2013). However, use of memory is not handled by Dijkstra algorithm. Therefore, it is expected that large memory space will be used by this algorithm for storing points that pass over each iteration as the shortest route is searched from a huge graph. This highlights the need of presenting techniques of search engine, combined with node combination algorithm for producing the shortest route search system. Shortest route search engine is efficient in terms of utilization of memory (Fitro et al., 2018). Memory saving is made possible by using the shortest route search engines as it combines the nodes to a single node with the smallest distance.

There is a wide usage of methods such as Bellman-ford and Floyd Marshall and Dijkstra method used to find the shortest path to a specific destination (Pramudita et al., 2019). Finding the shortest path is considered as a routing process to search minimum distance using network. Dijkstra algorithm assists in calculating path from the source of unvisited vertices using graph with sets of vertices and edges (Xie et al., 2012; Magzhan & Jani, 2013). This algorithm is also capable of finding the shortest path with minimum cost (Singal & Chhillar, 2014). The use of Dijkstra algorithm is advantageous as it only focuses on point reflationary and minimum value forming a single point (Dermawan, 2019). A study conducted by Sangaiah et al. (2014) reported the use of Dijkstra algorithm for solving the problem related to shortest distance.

Similarly, the use of Bellman-Ford (BF) algorithm plays an important role in appropriating negative graph weights (Retanto, 2009). However, a long processing time is required by BF algorithm to produce accurate output (Rofiq & Uzzy, 2014). Further, another algorithm named Haversine theorem, finds latitude and longitude of geographic locations. This theorem also finds the navigation system and calculates the distance between two locations based on latitude and longitude in a spherical (Type, 2016). Spherical trigonometry provides the basis for Haversine Theory as explained by one of the previous studies (Ganesh & Kumar, 2015). The distance between different nodes is calculated based on the Haversine Theory (Basyir et al., 2017). Accurate formula is required for precise method used to calculate different distances.

Shortest path algorithm is widely used in different applications (Panda & Mishra, 2018). GIS as a computer-based information system includes the attributes used to highlight spatial information. This algorithm is also used to process, manipulate, analyze, and present the geographical information. Therefore, it is considered as an efficient process that can be used to solve problems encountered in the link with geographic information (Pramudita et al., 2019). The issue related to the shortest path is the main concern in transportation management, and GIS plays an important role in this regard (Patel & Bagar, 2014). The shortest alternative path applies the rapid developments associated with the use of GIS (Wahyuningsih& Syahreza, 2018). In the similar context, the current study aims to combine the use of both the algorithms: Dijkstra and BF algorithms to find out the best route for each destination. The study also discusses the process of finding the shortest method.

II. Methods

The direct comparison approach is applied in this study, which includes 5 steps: researching, verifying, comparing, adjusting analysis, and reconciling (18). The direct comparison approach step is explained in table 1.

Table 1: Methodological Steps

Input	Output
Step 1: Literature and secondary data study	Step 1: Mapping using google map and presenting algorithm component
Step 2: Mapping using google map and verifying data by observing	Step 2: Mapping using google map considering the observation result
Step 3: Picking up measurement and algorithm component	Step 3: Comparing different variables
Step 4: Comparing variable. Google map, and observation result step by step	Step 4: Calculations are performed by each algorithm method
Step 5: Results of calculation provided by each algorithm	Step 5: Generating conclusions

The methodological steps followed in this study are illustrated in figure 1. The most important step is the inclusion of verified data and information, which is conducted before comparing the algorithm analysis, variable selection, and reconciliation.

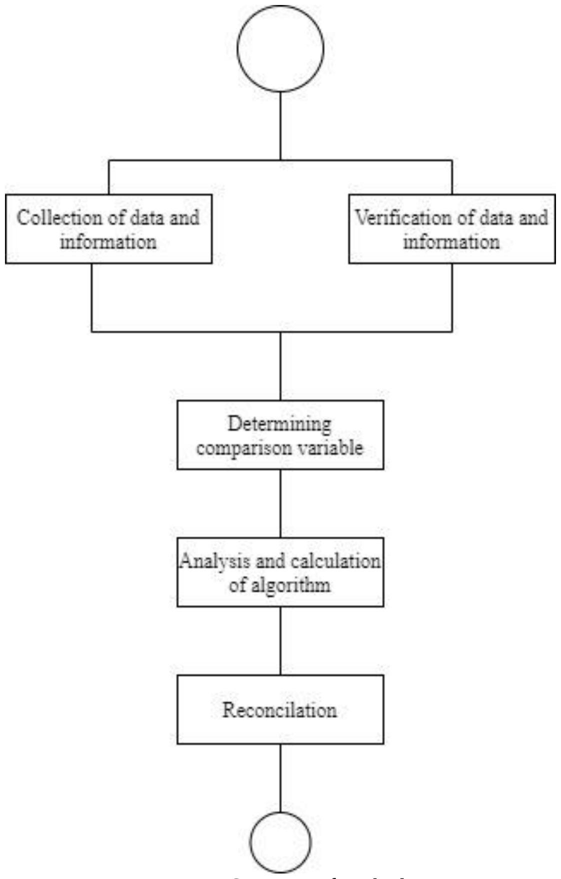
Collection of data and information
Verification of data and information
Determining comparison variable
Analysis and calculation of algorithm
Reconcilation

Figure 1: Summary of methods

III. Results and Discussion

In the current study, Dijkstra and Bellman-Ford algorithms have been compared to implement the shortest path through data collection, verification, and calculation. Google map was used to capture the secondary data. Paths with roads and nodes were collected with corresponding distances between Ir. H. Djuanda and Lewipanjang streets. The nodes for creating graphs are shown in figure 2, with their corresponding longitudes and latitudes. Google map was used to gather the values of the latitudes and longitudes as shown in table 2. In the next step, calculations were performed using Haversine theorem for finding the distances between different nodes.

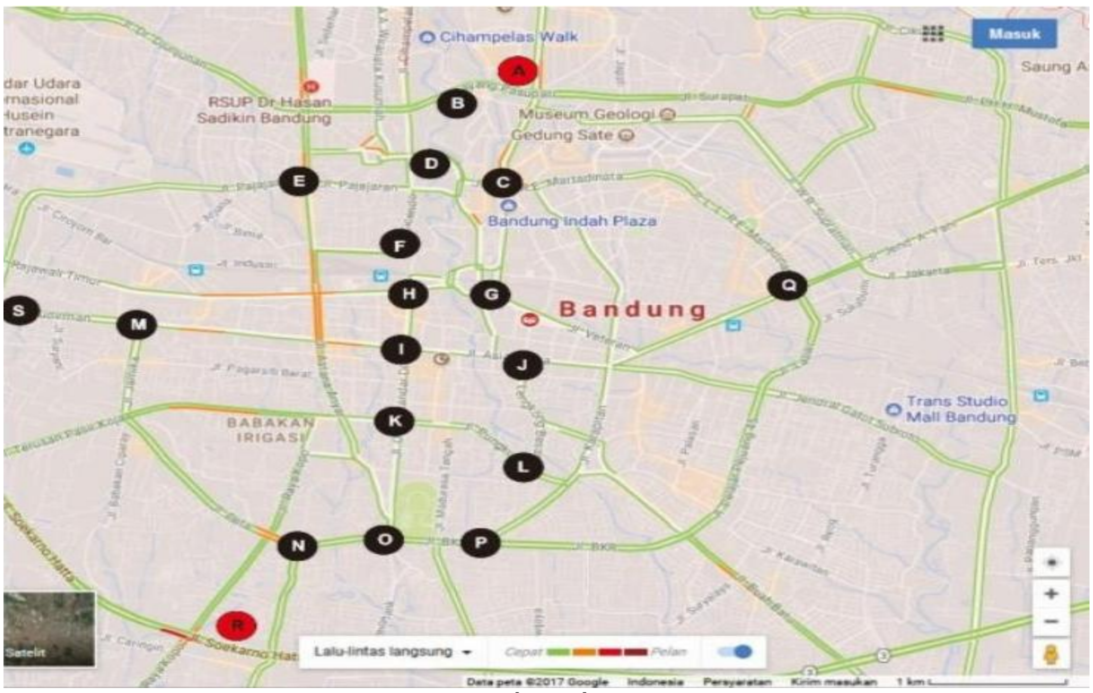

Figure 2: The simulation map

Table 2: Calculating the coordinate values

Node	Longitude	Latitude
A	107.612479	-6.898649
B	107.609317	-6.900398
C	107.610454	6.906970
D	107.607740	-6.905521
E	107.597537	-6.906448
F	107.603964	-6.912594
G	107.610026	-6.916503
H	107.604490	-6.915843
I	107.604093	6.920806
J	107.612022	-6.921786
K	107.603664	-6.927207
L	107.612387	-6.931244
M	107.586627	-6.918740
N	107.597077	-6.937783
O	107.602999	-6.937219
P	107.609136	6.937592
Q	107.630089	-6.915726
R	107.593182	-6.945430
S	107.574396	6.917409

IV. SEARCHING FOR SHORTEST PATH

The shortest path was calculated using Dijkstra and Bellman-Ford algorithms.

Dijkstra Algorithm

The weighted graph problem single source shortest path (SSSP) was solved using Dijkstra algorithm to produce a SP tree (Singh & Tripathi, 2018). The labels given to all the vertex had value of ∞, except for A=0. In figure 3, the first and last nodes are represented as red nodes, while, the nodes passing through are represented as black nodes. The distance between A adjacent to B, C, and Q was calculated in first iteration. Figure 3 (b) represented values of 1.1, 0.9, and 3 for B, C, and Q, respectively. The tabular presentation of values from B, C, and Q sides are shown in table 3. In order to find the shortest path, iteration was carried on till the shortest path, along with all the nodes were checked. The shortest path with value of 6.35km was found for A-C-G-H-I-K-O-N-R from Dijkstra algorithm.

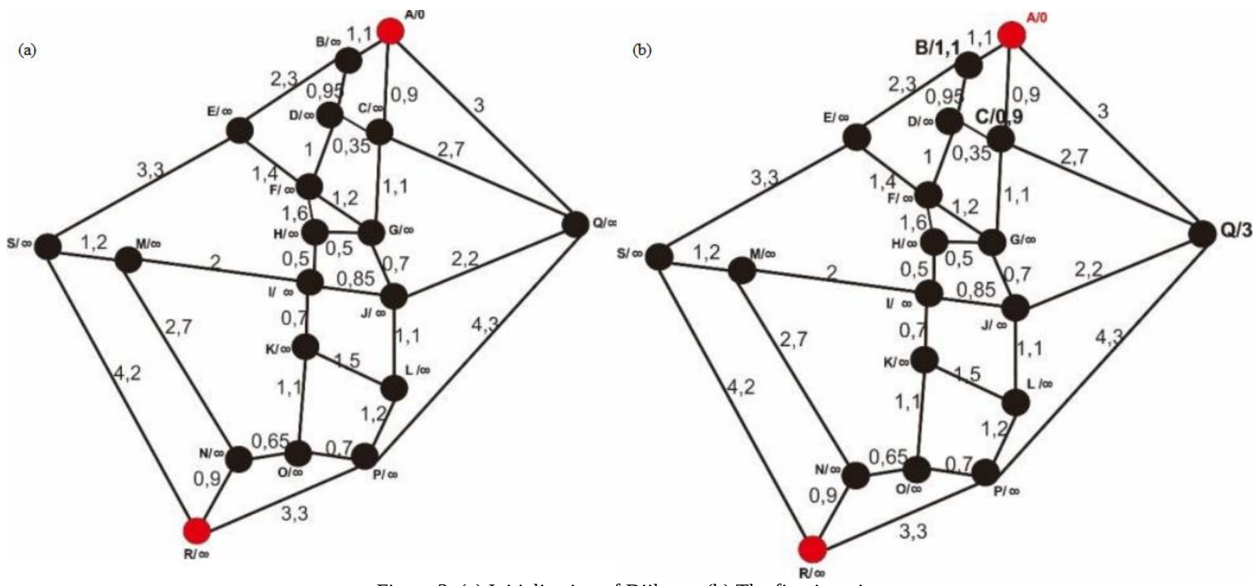

Figure 3: (a) Initialization of Dijkstra; (b) The first iteration

Table 3: First iteration of Dijkstra algorithm

u	S	L(u)	V	W(u,v)	L(v)
A	{A}	0	B	1,1	1,1
			C	0,9	0,9 (min)
			Q	3	3

Bellman Ford Algorithm

Unlike Dijkstra algorithm, Bellman Ford algorithm is used for graphs having negative values, (Kiruthika & Umarani, 2012). There is zero value of first node in the initialization step of Bellman Ford algorithm, whereas ∞ value was used for the other nodes (Table 4). Following Bellfam-Ford equation was used for calculating the shortest path;

$$M\,[i, v] = \min(M[i-1, v], (M[i-1, n] + cvn)) \qquad (1)$$

Where;

i = iteration,

v = vertex/node,

n = node neighbor,

C = cost

At the first iteration, cost, weight, and distance were added to begin the final calculations (Table 5). There was a connection between node A and B, C, and Q nodes.

$$M[1, B] = \min(M\,[0, B], (M\,[0, A] + CAB))$$

$$= \min(\infty, (0 + 1,1\ km))$$

$$= \min(\infty, 1{,}1km)$$

$$= 1{,}1km$$

$$M\,[1, C] = \min(M\,[0, C], (M\,[0, A] + CAC))$$

$$= \min(\infty, (0 + 0{,}9km))$$

$$= \min(\infty, 0{,}9km)$$

$$= 0{,}9km$$

$$M\,[1, Q] = \min(M[0, Q], (M\,[0, A] + CAQ))$$

$$= \min(\infty, (0 + 3km))$$

$$= \min(\infty, 3km)$$

$$= 3km$$

Table 6 shows the results for iteration along with the values of each node.

Table 4: Initialization of Bellfam-Ford algorithm

		Simpul (km)																		
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**
Iterasi	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

Table 5: First iteration of Bellfam-Ford algorithm

	Simpul (km)

		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
Iterasi	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	1	0	1,1	0,9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	3	∞	∞

Table 6**:** Result of Bellfam-Ford algorithm

		Simpul (km)																		
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
Iterasi	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	1	0	1,1	0,9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	3	∞	∞
	2	0	1,1	0,9	2,05	3,4	∞	2	∞	∞	∞	∞	∞	∞	∞	∞	7,3	3	∞	∞
	3	0	1,1	0,9	2,05	3,4	3,05	2	2,5	6,05	5,2	∞	6,3	∞	∞	8	7,3	3	6,7	10,6
	4	0	1,1	0,9	2,05	3,4	3,05	2	2,5	6,05	5,2	6,75	6,3	8,05	8,65	8	7,3	3	6,7	10,6

Comparing the Algorithms

The results showed that both the algorithms were capable of finding the shortest path. However, table 7 shows that the finding of the shortest path was possible through Dijkstra algorithm, but with many nodes.

Table 7: Comparing both Dijkstra and Bellman-Ford algorithms

Algorithm	**Iteration**	**Shortest Path**	**Total Distance**
Dijkstra	18	A-C-G-H-I-K-ON-R	6.35 km
Bellman-Ford	5	A-Q-P-R	10.6 km

V. Conclusion

GIS-based application is widely used for calculating the shortest path in order to find the best route for the user. Although both Dijkstra and Bellman-Ford algorithms produce good result in terms of finding the shortest path, in the current study, Dijkstra algorithm showed better results. However, Bellman-Ford algorithm showed more effective results as it minimized the number of nodes in reaching the destination point. The findings of the current study suggest that future studies need to include re-routing mechanism to enhance the dynamic route calculation, considering the change in weights.

The approach proposed in this study is particularly applicable to GIS as the users are likely to perform a shortest path search in these systems. This may also allow the users to expand vertices by avoiding the influence of the time for shortest path search used in this operation. Moreover, the scalability regarding size of graph is ensured by shortest path search on reduced graph. The present study just showed modifications made on Dijkstra and Bellman-Ford algorithms concerned with using new function through reduced vertex. However, future studies need to include and modify other algorithms for making shortest path search through reduced vertex, along with cost of path.

References

1. Basyir, M., Nasir, M., Suryati, S., & Mellyssa, W. (2017). Determination of Nearest Emergency Service Office using Haversine Formula Based on Android Platform. EMITTER International Journal of Engineering Technology, 5(2), 270-278.
2. Dermawan, T. S. (2019). Comparison of Dijkstra dan Floyd-Warshall Algorithm to Determine the Best Route of Train. IJID (International Journal on Informatics for Development), 7(2), 54-58.
3. Fitro, A., P Sulistio Ilham, A., B Saeful, O., & Frendianata, I. (2018). Shortest path finding in geographical information systems using node combination and dijkstra algorithm. Shortest Path Finding in Geographical Information Systems Using Node Combination and Dijkstra Algorithm, 9(2), 755-760.
4. Ganesh, L., & Kumar, D. V. (2015). Indoor wireless localization using haversine formula. International Advanced Research Journal in Science, Engineering and Technology, 2(7), 59-63.
5. Jiang, L., Qi, Q., & Zhang, A. (2010, June). The thematic mapping system on internet. In 2010 18th International Conference on Geoinformatics (pp. 1-4). IEEE. doi:10.1109/GEOINFORMATICS.2010.5567802
6. Kiruthika, R., & Umarani, R. (2012). Shortest path algorithms: A comparative analysis. International Journal of Managment, IT and Engineering, 2(4), 55-62.
7. Magzhan, K., & Jani, H. M. (2013). A review and evaluations of shortest path algorithms. International journal of scientific & technology research, 2(6), 99-104.
8. Miler, M., Medak, D., & Odobašić, D. (2014). The shortest path algorithm performance comparison in graph and relational database on a transportation network. Promet-Traffic&Transportation, 26(1), 75-82.

9. Panda, M., & Mishra, A. (2018). A Survey of Shortest-Path Algorithms. International Journal of Applied Engineering Research, 13(9), 6817-6820.
10. Patel, V., & Bagar, 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, 5, 1-4.
11. Pramudita, R., Heryanto, H., Handayanto, R. T., Setiyadi, D., Arifin, R. W., & Safitri, N. (2019, October). Shortest Path Calculation Algorithms for Geographic Information Systems. In 2019 Fourth International Conference on Informatics and Computing (ICIC) (pp. 1-5). IEEE. doi: 10.1109/ICIC47613.2019.8985871
12. Retanto, Y. (2009). Algoritma Dijkstra dan Bellman-Ford dalam Pencarian Jalur Terpendek. Institut Teknologi Bandung: Bandung.
13. Rodríguez-Puente, R., & Lazo-Cortés, M. S. (2013). Algorithm for shortest path search in Geographic Information Systems by using reduced graphs. SpringerPlus, 2(1), 291.
14. Rofiq, M., & Uzzy, R. F. (2014). Penentuan Jalur Terpendek Menuju Cafe Di Kota Malang Menggunakan Metode Bellman-Ford dengan Location Based Service Berbasis Android. Jurnal Ilmiah Teknologi Informasi Asia, 8(2), 49-64.
15. Sangaiah, A. K., Han, M., & Zhang, S. (2014). An investigation of Dijkstra and Floyd algorithms in national city traffic advisory procedures. International Journal of Computer Science and Mobile Computing, 3(2), 124-138.
16. Singal, P., & Chhillar, R. S. (2014). Dijkstra Shortest Path Algorithm using Global Positioning System. International Journal of Computer Applications, 101(6), 12-18.
17. Singh, J. B., & Tripathi, R. C. (2018). Investigation of Bellman-Ford Algorithm, Dijkstra's Algorithm for suitability of SPP. Int. J. Eng. Dev. Res. 1 Dean Res, 6(1), 2321-9939.
18. Soltani, A. R., Tawfik, H., Goulermas, J. Y., & Fernando, T. (2002). Path planning in construction sites: performance evaluation of the Dijkstra, A∗, and GA search algorithms. Advanced engineering informatics, 16(4), 291-303. doi: 10.1016/S1474-0346(03)00018-1
19. Torres, AR, & Puente, RR (2010). Thematic map service. Mapping, (139), 36-39.
20. Type, M. (2016). Ltd. Calculate Distance, Bearing and More between Latitude/Longitude Points. Retrieved from: http://www.movable-type.co.uk/scripts/latlong.html.
21. Wahyuningsih, D., & Syahreza, E. (2018). Shortest Path Search Futsal Field Location with Dijkstra Algorithm. IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 12(2), 161-170. doi: 10.22146/ijccs.34513
22. Xie, D., Zhu, H., Yan, L., Yuan, S., & Zhang, J. (2012, June). An improved Dijkstra algorithm in GIS application. In World Automation Congress 2012 (pp. 167-169). IEEE.