Advances in Engineering Research, volume 207
Proceedings of the 2nd International Seminar of Science and Applied Technology (ISSAT 2021)

Determination of the Fastest Path on Logistics Distribution by Using Dijkstra Algorithm

Anie Lusiani[1,*] Euis Sartika[2] Agus Binarto[3] Endang Habinuddin[3] Irfani Azis[4]

[1]*Department of Mechanical Engineering, Politeknik Negeri Bandung, Bandung, 40559, Indonesia*
[2]*Department of Commercial Administration, Politeknik Negeri Bandung, Bandung, 40559, Indonesia*
[3]*Department of Electrical Engineering, Politeknik Negeri Bandung, Bandung, 40559, Indonesia*
[4]*Department of Mathematics, Universitas Pamulang, Tangerang Selatan, 15417, Indonesia*
[*]*Corresponding author. Email: anie.lusiani@polban.ac.id.*

ABSTRACT
Dijkstra algorithm is one of the algorithms that is used to determine the path with the minimum total weight in the computer network, communication network, and transportation network problems. Some problems that have been studied using Dijkstra algorithm, namely multi-hop calibration of networked embedded system, achieving superior timing-driven routing trees and adaptive protection in microgrids. This article will determine the fastest path in the distribution of logistics by Bulog in West Java region by using Dijkstra algorithm manually and also by using Matlab. Data and information including the path connecting a warehouse to another Bulog warehouse will be used to build a connected weighted graph model. This data was obtained directly from Bulog office in West Java and through Google Maps application during the Covid-19 pandemic. Furthermore, path optimization is carried out by using Dijkstra algorithm, so that the fastest path tree is obtained. The fastest path tree is a path which edge set is a subset of the connected edge set of the connected weighted graph and has minimum total weight. Based on this optimization, the fastest path from Cibitung warehouse Bekasi to Bojong warehouse Cianjur is 154 minutes.

Keywords: *connected weighted graph, Dijkstra algorithm, fastest path, logistics distribution.*

1. INTRODUCTION

The development of graph theory in terms of the common application is on the determination of the shortest path. However, the business or density of a track will lead to the need for the determination of the fastest path. Dijkstra algorithm is an algorithm used in determining the minimum weighted path in the problem of computer networks, communication networks, or transportation networks. This weight can be either the cost, distance, or time required through the track. Thus, the Dijkstra algorithm can be used in the development of graph theory both on the determination of the shortest path and the fastest path in the logistics distribution transportation network. This algorithm was discovered by Edsger W. Dijkstra in 1956. The shortest-path tree is generated from the Dijkstra Algorithm by finding the shortest path from a vertex to all other vertices in the graph. In other words, this algorithm will find the shortest path between any vertex to another vertex [1].

The graph used in this article is simple and finite. Let G be a connected graph that each edge has weights in the form of travel costs, mileage, or travel time. A graph whose edges have weight is called a weighted graph. The weight of an edge e in G is written as $w(e)$. The weight $w(H)$, where H is a subgraph of G, is defined as the sum of the edge weights:

$$w(H) = \sum_{e \in E(H)} w(e), \qquad (1)$$

In this formula, $E(H)$ is the edge set of H.[2]

2. BACKGROUND

A good distribution network is an important competitive advantage for a company that manages logistics distribution such as Bulog General Company, especially in terms of punctuality. This is influenced by the network of vehicle routes passed in the distribution of logistics, as the study conducted in Hongkong in 2019. The vehicle routing problem algorithm is one of the important parts to establish an integrated modular integrated construction (MiC) logistics planning project [3]. The problem that occurs today is the hamper of logistics distribution due to road congestion in some areas, resulting in delays in fulfilling logistical needs. Therefore, efforts are needed to overcome the delay in distributing logistics. One of them is to determine the fastest path on logistics distribution by using Dijkstra algorithm.

A comparative study of multicast using Dijkstra and other algorithms was conducted by Sahana et al. in 2010 [4]. Some problems that have been studied in using Dijkstra algorithm, namely multi-hop calibration of the networked embedded system [5], achieving superior timing-driven routing trees [6], and adaptive protection in microgrids [7]. Discussion of the benefits of combining graph theories and board games for the Dijkstra and Prim minimum spanning tree theories have been conducted in introducing competitive board games to motivate students to learn the concept of minimum spanning tree algorithm [8]. The results of the priority search tree are purely functional, simple, and efficient have also been determined by Lammich et al. theoretically, as well as their application in Prim and Dijkstra algorithms [9]. The application of graphs as models to a problem is not only in determining the shortest path and the fastest path, but it is also used as a model for traffic congestion problems such as determining waiting times in overcoming congestion at the intersection of Toll-Pasteur Bandung road [10]. This article will determine the fastest path in the distribution of logistics by Bulog in West Java region by using Dijkstra Algorithm.

3. RESEARCH METHODS

The model to be reviewed is a model in the analysis of Bulog logistics distribution's case, namely the distribution of logistics from one Bulog warehouse to another, conducted by the Regional Division (Divre) Bulog Perum West Java. In this article, the analysis is devoted to the travelling of distribution of logistics from warehouse Bulog Cibitung, Jalan Imam Bonjol, Sukadanau, West Cikarang District, Bekasi to warehouse Bulog Bojong, Jalan Raya Bandung No.253, Bojong, Karangtengah, Cianjur Regency, West Java.

The initial model built into this algorithm was a connected weighted graph with a weight of travel time. In determining the fastest path, it used weights in the form of travel time. The flow of research can be described in general. First, data travel time distribution of logistics is processed from the starting point to the end of the destination. Second, this data is written in the form of a connected graph. Third, the Dijkstra algorithm is run on this graph so that the fastest path is obtained.

The shortest-path tree is generated from the Dijkstra Algorithm by finding the shortest path from a vertex to all other vertices in the graph. In other words, this algorithm will find the shortest path between any vertex to another vertex. This algorithm can also be limited to find the shortest path from an initial vertex to a destination vertex. Because the weight used is travel time instead of mileage, the shortest path becomes the fastest path.

Table 1. The Weights in Travel Time

Edge	Travel Time (minute)	Edge	Travel Time (minute)
$e_1 = (1, 2)$	41	$e_9 = (3, 7)$	165
$e_2 = (1, 3)$	107	$e_{10} = (4, 5)$	146
$e_3 = (2, 3)$	100	$e_{11} = (4, 6)$	45
$e_4 = (2, 4)$	13	$e_{12} = (5, 6)$	147
$e_5 = (2, 5)$	146	$e_{13} = (5, 7)$	105
$e_6 = (3, 4)$	168	$e_{14} = (5, 8)$	70
$e_7 = (3, 5)$	17	$e_{15} = (6, 7)$	33
$e_8 = (3, 6)$	139	$e_{16} = (7, 8)$	22

Table 1 shows the weight of each edge, i.e. travel time in minutes from one region to another. For example, the edge (1, 2) is the edge that connects areas 1 and area 2. Travel time from area 1 to area 2 is 41 minutes.

The vertices in the graph show the areas that the logistics distribution track passes through, while the edges indicate the path that can be passed by the logistics distribution track by using a truck. Vertex 1 shows Cibitung warehouse, Bekasi as the starting point of logistics distribution. Vertex 8 shows Bojong warehouse in Cianjur as the final destination of distribution. Vertices 2, 3, 4, 5, 6, and 7 respectively show the areas of Cikampek, Cariu, Sadang (Purwakarta), Cikabuyutan, Cikalong, and Cipatat. So, total areas are eight regions. Thus, the number of vertices are also eight. Since not every region is connected to another region, the weighted connected graph obtained is not completed as shown in the following figure. However, this connected weighted graph G is a graph with 16 edges.

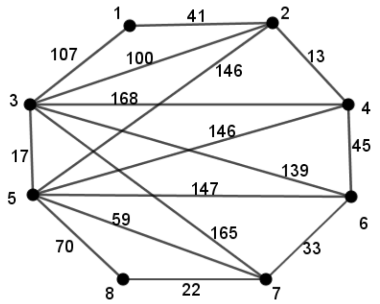

Figure 2 The Connected Weighted Graph G

The connected weighted graph G as the initial model is constructed from the travel time weight data for each edge shown in Table 1. The steps on algorithm Dijkstra work as follows. The first step is to establish a starting point with the status of having been found and has been visited. Because the desired path is starting from vertex 1, then vertex 1 status has been found and has been visited. The second step is to find other vertices that are adjacent to vertex 1. If these vertices have statuses that are not found yet, then their status will be changed into ‘found’. However, if it has been found, then they only update the weight. The third step is to choose a vertex with a minimum weight to visit.

These second and third steps result the vertex that is always being ‘found’ if the vertex is connected to a vertex that has been visited (in fact, a vertex is found more than once if the vertex is connected to some vertices that have been visited). However, not all vertex needs to be visited. In the optimization process, if a vertex does not have a minimum weight, then the vertex does not need to be visited. Next, we loop in the second and third steps, so that vertex which is the final destination has a status that has been visited. In this case, looping will stop if vertex 8 as the final destination vertex has been visited. The following figures and table show the second and third steps.

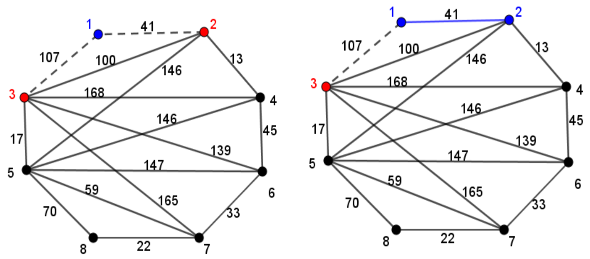

Figure 3 Graph *G* in The Second and The Third Steps on Algorithm Dijkstra

Vertex 1 has been determined as the starting point, so in Figure 3, vertex 1 is coloured blue to indicate that it has been found and visited. The first row in Table 2 shows the route from vertex 1 to vertex 1 yields zero weight. Then, in the second step, vertex 2 and vertex 3 are found adjacent to vertex 1. Vertices 2 and 3 are coloured red. Since edge 1 and 2 are the edge with minimum weight, the next vertex to visit is vertex 2. So, vertex 2 is coloured blue. This is shown by the second row in Table 2.

Table 2. The Vertices Conditions in The Third Step of Algorithm Dijkstra

Vertex	Status	Weight	Route
1	visited	0	1
2	visited	41	1 ⟶ 2
3	found	107	1 ⟶ 3
4	not yet found	∞	-
5	not yet found	∞	-
6	not yet found	∞	-
7	not yet found	∞	-
8	not yet found	∞	-

The next step is to loop sequentially from the second step to the third step until the destinated vertex has been visited.

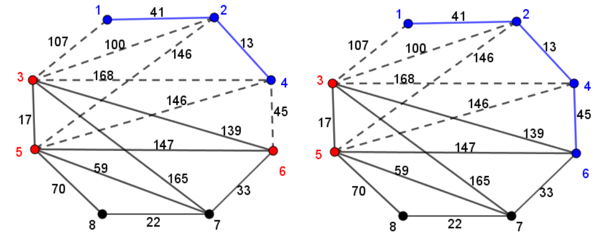

Figure 4 Graph *G* in The First Looping Step on Algorithm Dijkstra

The fourth and fifth rows in Table 3 show vertex 4 has been found and visited, while vertex 5 has been found. The route from vertex 1 to vertex 4 is 1 ⟶ 2 ⟶ 4 for a total weight of 54 minutes. This result is based on Figure 4 on the left. Vertex 4 is coloured blue and found vertices 3, 5, and 6 so that these three vertices are coloured red.

Table 3. The Vertices Conditions in The First Looping Step of Algorithm Dijkstra

Vertex	Status	Weight	Route (Path)
1	visited	0	1
2	visited	41	1 ⟶ 2
3	found	107	1 ⟶ 3
4	visited	54	1 ⟶ 2 ⟶ 4
5	found	124	1 ⟶ 3 ⟶ 5
6	visited	99	1 ⟶ 2 ⟶ 4 ⟶ 6
7	not yet found	∞	-
8	not yet found	∞	-

Since the edge (4, 6) is the edge with minimum weight, the next vertex to visit is vertex 6. So, vertex 6 is colored blue. See Figure 4 on the right. As shown in row 6 in Table 3, the route from vertex 1 to vertex 6 is 1 ⟶ 2 ⟶ 4 ⟶ 6 for a total weight of 99 minutes.

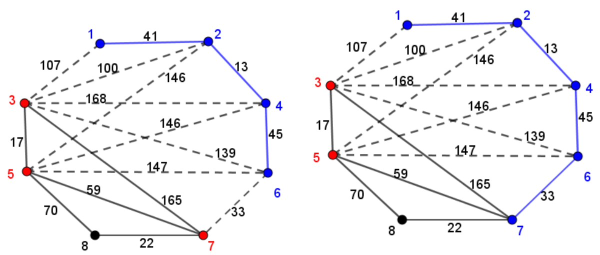

Figure 5 Graph *G* in The Second Looping Step on Algorithm Dijkstra

Next, vertex 6 is adjacent to vertices 3, 5, and 7. Therefore, these three vertices have a status that has been found and colored red as seen on Figure 5 on the left.

Table 4. The Vertices Conditions in The Second Looping Step of Algorithm Dijkstra

Vertex	Status	Weight	Route (Path)
1	visited	0	1
2	visited	41	1 ⟶ 2
3	found	107	1 ⟶ 3
4	visited	54	1 ⟶ 2 ⟶ 4
5	found	124	1 ⟶ 3 ⟶ 5
6	visited	99	1 ⟶ 2 ⟶ 4 ⟶ 6
7	visited	132	1 ⟶ 2 ⟶ 4 ⟶ 6 ⟶ 7
8	not yet found	∞	-

Since the edge (6, 7) is the edge with minimum weight, the next vertex to visit is vertex 7. So, vertex 7 is colored blue. See Figure 5 on the right. As shown in row 7 in Table 4, the route from vertex 1 to vertex 7 is 1 ⟶ 2 ⟶ 4 ⟶ 6 ⟶ 7 for a total weight of 132 minutes.

4. RESULT

The last looping is done to obtain the route from vertex 1 to vertex 8. The route at this stage is 1 ⟶ 2 ⟶ 4 ⟶ 6 ⟶ 7 ⟶ 8 with a weight of 154 minutes as shown in the following table. The final destination vertex is vertex 8, so this route is the shortest path from an initial vertex to a destination vertex. Because the weight used is travel time instead of mileage, the shortest path becomes the fastest path. Thus, route 1 ⟶ 2 ⟶ 4 ⟶ 6 ⟶ 7 ⟶ 8 is the fastest path, which is 154 minutes. The explanation is shown with the following figure and table in the last step.

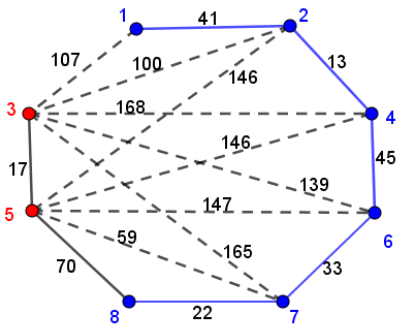

Figure 6 Graph *G* in The Last Step on Algorithm Dijkstra

Vertex 7 is adjacent with vertices 3, 5, and 8. Therefore, these three vertices have a status that has been found and coloured red. Next, since the edge (7, 8) is the edge with minimum weight, the next vertex to visit is vertex 8. So, vertex 8 is colored blue as seen on Figure 6. As shown in row 8 in Table 5, the fastest path from vertex 1 to vertex 8 is 1 ⟶ 2 ⟶ 4 ⟶ 6 ⟶ 7 ⟶ 8 for a total weight of 154 minutes.

Table 5. The Vertices Conditions in The Last Step of Algorithm Dijkstra

Vertex	Status	Weight	Route (Path)
1	visited	0	1
2	visited	41	1 ⟶ 2
3	found	107	1 ⟶ 3
4	visited	54	1 ⟶ 2 ⟶ 4
5	found	124	1 ⟶ 3 ⟶ 5
6	visited	99	1 ⟶ 2 ⟶ 4 ⟶ 6
7	visited	132	1 ⟶ 2 ⟶ 4 ⟶ 6 ⟶ 7
8	visited	154	1 ⟶ 2 ⟶ 4 ⟶ 6 ⟶ 7 ⟶ 8

The data processing are shown in graph *G*. As the addition to the manual analysis, it is also used Matlab softwareas shown in the following figure.

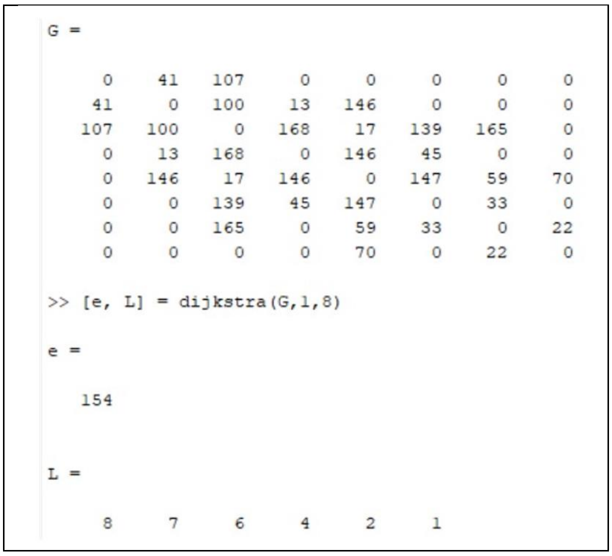

Figure 7 Adjacency Matrix *G* of Graph *G* on Algorithm Dijkstra

Matrix G is the adjacency matrix of graph G which is the input data in Dijkstra's Algorithm. The number 41 in the first row of the second column indicates the travel time required from vertex 1 to vertex 2, while the number 41 in the second row of the first column indicates the travel time required from vertex 2 to vertex 1, and so on. The symbol *e* represents the total weight of the fastest path L shown in Figure 7. The results of Matlab show the same results as the manual way as the fastest path from vertex 1 to vertex 8 is 1 ⟶ 2 ⟶ 4 ⟶ 6 ⟶ 7 ⟶ 8 for a total weight of 154 minutes.

5. CONCLUSION

After optimizing the path using Dijkstra Algorithm, the fastest path is obtained. The fastest path is a path which edge set is a subset of a set of connected weighted graph edges and has minimum total weight. Based on this optimization, the fastest route or the fastest path from Cibitung warehouse in Bekasi to Bojong warehouse in

Cianjur is Cibitung⟶Cikampek⟶Sadang (Purwakarta) ⟶Cikalong⟶Cipatat⟶Cianjur for a total weight of 154 minutes.

Optimization of the logistics distribution path is also done with another method named the Prim Algorithm by using the same travel time data. The results are shown at [11], which is obtained the same optimal time of 154 minutes. Thus, Dijkstra Algorithm has been confirmed as a method that can be used for path optimization problems.

ACKNOWLEDGMENT

This research has been supported by “DIPA POLBAN in the Penelitian Peningkatan Daya Saing Kelompok Bidang Keahlian research scheme No. B/78.8/PL1.R7/PG.00.03/2021, Politeknik Negeri Bandung, Indonesia”.

REFERENCES

[1] K. Mehlhorn and P. Sanders, *Algorithms and data structures: The basic toolbox*. Springer Science & Business Media, 2008.

[2] G. Chartrand, L. Lesniak, and P. Zhang, *Graphs & Digraphs. Textbooks in Mathematics*. CRC Press, Boca Raton, FL, 2016.

[3] S. Niu, Y. Yang, and W. Pan, “Logistics planning and visualization of modular integrated construction projects based on BIM-GIS integration and vehicle routing algorithm,” *Modul. Offsite Constr. Summit Proc.*, pp. 579–586, 2019.

[4] S. K. Sahana, D. A. Jain, and A. Mustafi, “A Comparative Study on Multicast Routing using Dijkstra’s, Prims and Ant Colony Systems,” *Int. J. Comput. Eng. Technol.*, vol. 1, no. 2, pp. 16–25, 2010.

[5] V. Freschi and E. Lattanzi, “A Prim-Dijkstra Algorithm for Multi-Hop Calibration of Networked Embedded Systems,” *IEEE Internet Things J.*, 2021.

[6] C. J. Alpert *et al.*, “Prim-Dijkstra revisited: Achieving superior timing-driven routing trees,” in *Proceedings of the 2018 International Symposium on Physical Design*, 2018, pp. 10–17.

[7] O. V. G. Swathika and S. Hemamalini, “Prims-aided dijkstra algorithm for adaptive protection in microgrids,” *IEEE J. Emerg. Sel. Top. Power Electron.*, vol. 4, no. 4, pp. 1279–1286, 2016.

[8] W.-C. Chang, T.-H. Wang, and Y.-D. Chiu, “Board game supporting Learning Prim’s Algorithm and dijkstra’s Algorithm,” *Int. J. Multimed. Data Eng. Manag.*, vol. 1, no. 4, pp. 16–30, 2010.

[9] P. Lammich and T. Nipkow, “Proof pearl: Purely functional, simple and efficient Priority Search Trees and applications to Prim and Dijkstra,” 2019.

[10] A. Lusiani, E. Sartika, A. Binarto, and E. Habinuddin, “Compatible Graphs on Traffic Lights Waiting Time Optimization,” in *Proceedings of the International Seminar of Science and Applied Technology (ISSAT 2020)*, 2020, pp. 467–471, doi: 10.2991/aer.k.201221.077.

[11] A. Lusiani, E. Sartika, E. Habinuddin, A. Binarto, and I. Azis, “Algoritma Prim dalam Penentuan Lintasan Terpendek dan Lintasan Tercepat pada Pendistribusian Logistik Bulog Jawa Barat,” in *Prosiding Industrial Research Workshop and National Seminar*, 2021, vol. 12, pp. 673–677.