Optimization of ISP Service Maintenance Router Using Dijkstra and Floyd-Warshall Algorithm

Wiwien Hadikurniawati\(^1\)\(^*,\) Edy Winarno\(^1\), Aldy Hernawan\(^1\) and Dahlan Abdullah\(^2\)

\(^1\)Faculty of Information Technology, Universitas Stikubank, Semarang, Indonesia
\(^2\)Department of Informatics, Universitas Malikussaleh, Banda Aceh, Indonesia

\(*\)wiwien@edu.unisbank.ac.id

Abstract. In this paper, Dijkstra algorithm and Floyd-Warshall algorithm are proposed to reduce the number of repetitive operations and find the shortest path and the fastest time from the starting point to all other nodes to arrive at the destination. The Dijkstra algorithm is used to sort location data by distance and sort location data based on travel time using the Floyd-Warshall algorithm. This application is expected to provide benefits as an example to facilitate ISP personnel in obtaining the shortest path information or route to the point of service interruption. This will optimize disruption handling time and vehicle fuel. Operational costs are more efficient. For ISP consumers, handling disruptions will be more quickly resolved. Test result indicate that the application can show steps to find the route to the ISP service disruption and advice which node is passed for the shortest distance and the fastest time. A numerical example was used to illustrate the efficiency of the proposed technique. The propose technique can be applied to real application transportation system and many other network optimization problem that can be formulated as shortest path problem.

1. Introduction

Internet service providers (ISP) are companies that provide Internet connection services and other related services. They have networks both domestically and internationally so that customers or users of connections provided by ISPs can connect to global Internet networks. ISP customers every year continue to increase. The impact is a lot of disruption to services. The more number of customers, will be directly proportional to the number of possible disruptions. Limited maintenance personnel will certainly make a lot of disruption handling to experience obstacles such as long handling time. Waste of travel time for maintenance personnel to customers who experience connection disruptions is one of the factors that make handling disruptions take a long time.

Disruption handling is sorted on the shortest route, it will save travel time and operational costs. Shortest path problem are one of the optimization problems which attracts researchers to be used in various applications \[1\],\[2\]. The shortest path is the minimum impedance path between two points in network. The impedance can be the actual distance, time etc \[3\]. In graph theory, shortest path problem is a problem to find a path between two vertices in a graph so that the sum of the weight of the constituent channels can reach minimum values. Short path problem can be implemented for a directed graph, without direction, or a combination of both of them. The most common path algorithms are Dijkstra algorithm, A star algorithm, Bellman-Ford algorithm, Floyd-Warshall algorithm \[4\].
Now the method for finding the shortest route in a system is needed. This makes the user more efficient, faster and more precise to get to the destination. The Dijkstra algorithm chooses the side with the smallest weight that connects a selected node with another node that has not been selected [5]. The Dijkstra algorithm requires parameters of the place of origin and destination so as to produce the shortest distance from the place of origin to the destination and its route. Dijkstra algorithm is one of celebrated algorithm in computer science. It is the most used applications such as navigations based on Google Maps.

By determining the variables, among others, type of direction, level of congestion, road width, road condition, road volume, distance, number of deviation, an effective alternative route will be generated. The number of vertices and edges are very large in actual traffic networks. The amount of deviation affects time, because every deviation is likely to have a traffic light which is certainly time consuming. The more variables that affect the better the shortest route choice. The traffic is modeled as weighted graph. Then the problem can be solved by applying Dijkstra algorithm to find the shortest path between two nodes.

This application uses Dijkstra Algorithm and Floyd-Warshall algorithm. This application is expected to provide benefits as an example to facilitate ISP personnel in obtaining the shortest path information or route to the point of service interruption. This will optimize disruption handling time and vehicle fuel. Operational costs are more efficient. For ISP, disrupted handling disruptions will be more quickly resolved.

This paper is organized as follows: In Section 2, we give an overview of background and related works. In Section 3, we present our algorithm and the basic theory used in our proposed technique used to find the optimal route. In Section 4, we discuss the testing and evaluation results. In Section 5, we talk about the conclusion.

2. Related Work
The shortest path is an important research topic in graph theory. The shortest path algorithm in the graph theory includes both the one between the node pair and the one among all the nodes. The average person encounters the shortest path algorithms almost in a daily basis. For example when seeking direction on the internet or on GPS devices. The shortest path algorithms are applied to many problems such as transportation and communication [6].

Research conducted by [3] shows that Dijkstra algorithm can improve storage efficiency and reduce useless operations, short operating time and increase the space and complexity of the search time. [7] in his research has proposed a parallel Dijkstra algorithm that is useful in some cases applications such as computer systems with large node densities. This algorithm is one step towards improving the system in several fields such as computer networks or the internet, GPS systems to support the shortest path or lowest cost. The Dijkstra algorithm is applied in research conducted by [8].

There is a study that addresses the issue of effective travel planning and resolve physical travel problems as shortest path problem. The Floyd-Warshall method is used to solve this case and reduce computation in finding the shortest route [9]. [10] have conducted research aimed at finding out the shortest pathway to evacuate disaster victims by using a combination of Fuzzy and Floyd Algorithm Warshall. The output of fuzzy logic is the value of the weight of each route to be input for processing with Floyd Warshall algorithm. The results obtained from a combination of both methods are optimal paths.
3. Algorithm

3.1. Dijkstra Algorithm

Dijkstra algorithm gets the shortest path one by one according to the increasing order of path length. Dijkstra algorithm needs to be improved in time and space in order to improve the efficiency of path searching.

Dijkstra algorithm works for the non-negative edge cost case. It requires that the weight of any edge can not be negative so that there are no negative loops, making the problem solvable. The basic idea of Dijkstra's algorithm is the search for the smallest value closest to the destination used on a weighted graph, so that it can help provide a choice of paths. In the Dijkstra Algorithm, the node is used because the Dijkstra algorithm uses a directed graph to determine the shortest route [11]. This algorithm aims to find the shortest path based on the smallest weight from one point to another. In Figure 1, an example graph is given with its weight in determining the path using the Dijkstra Algorithm.

First, determine the point that will be the initial point, then give weight to the first node to the nearest node one by one. The Dijkstra algorithm then develops the search from one point to another, then to the next point at the next stage, there are some logical sequences of the Dijkstra Algorithm.

1. Assign to every node a distance value then set the value 0 at the initial node and to infinity for all other node.
2. Mark all nodes as unvisited and set initial node as current.
3. For current node, consider all its unvisited neighbors and then calculate their tentative distance. If this distance is less than the previously recorded, overwrite the distance.
4. Considering all neighbors of the current node, mark it as node visited. A visited node will not be checked against its distance recorded as final and minimal.
5. If all nodes have been visited, stop. Otherwise, set the unvisited node with the smallest distance from the initial node, considering all nodes in the graph) as the next “current node” and continue from step 3.

3.2. Floyd-Warshall Algorithm

Floyd-Warshall algorithm is more efficient than a simplistic approach single-pair shortest path algorithm all relevant node pair [12]. Floyd-Warshall algorithm which is also known as plaid point method, is a dynamic programming approach used to find the shortest path between vertices of a given weighted graph. Floyd-Warshall for the purpose of reducing the computation complexities. It can simplify the method of finding the shortest path [13]. The Floyd-Warshall algorithm will compare all possible path trajectories for each side of all nodes. In one algorithm execution, we will get the distance as the weighted sum of the shortest paths between each pair of vertices [12].

3.3. Proposed Optimization Technique

The Dijkstra algorithm finds the shortest path from one node to another that may include all other nodes in a weighted graph. The steps include setting the current node, labelling its weight, finding the...
current neighbour node and the distance weight, labelling the smallest node as a permanent node and as the new node currently.

Dijkstra algorithm is an algorithm that uses a greedy principles that needs to collaborate Floyd-Warshall algorithm to find the best solution of ISP maintenance personnel path to the disruption point. It can reduce decision making without any solution. The proposed technique shown in Figure 2.

![Dijkstra Algorithm](image)

**Figure 2.** Proposed Technique

The system will do graph model. To sort location data by distance, it uses the Dijkstra algorithm and to sort location data based on travel time, it uses Floyd-Warshall algorithm. The next process is to match the distance and location of the route on the smallest number. This is the final result.

4. Result and Discussion

The case study area used in the research of route optimization of ISP service disruption handling uses the Dijkstra algorithm. The area is from sukun raya road - jati raya - kruing raya - meranti raya - karang rejo raya - anton sujarwo road. The map shown in Figure 3.

![Study Case Area](image)

**Figure 3.** Study Case Area

There are 233 crossing points that are likely to be passed by ISP maintenance technicians to the location of the point of disturbance in the area. Each intersection point is numbered from 1 to 233, with a description of the road.
System testing has been carried out by taking part of the ISP service disruption area. Figure 4 shows the route to be tested using the proposed model.

![Figure 4. Maps Sample Testing](image_url)

The starting points to search the shortest path can be seen in Table 1. The starting point of the personnel maintenance route starts from Automatic Telephone Exchange (STO) Banyumanik.

| Points  | Track Information                      |
|---------|----------------------------------------|
| STO     | Starting point                         |
| T32     | Sukun Raya - Sukun 2                   |
| T33     | Sukun Raya - Jendral Pol Anton Sujarwo |
| T34     | Jendral Pol Anton Sujarwo - Gang Mangga|
| T225    | Sukun 2 - Gang Mangga                  |
| T226    | Gang Mangga - Mangga Dalam             |
| T227    | Jambing Timur Dalam – Mangga Dalam     |
| T228    | Mangga Dalam – Jati Barat              |
| T229    | Gang Jati Barat - Mangga Dalam         |

Then the possible paths that can be traversed are:
1) First path from STO > T33 > T32 > T225 > T226 > 228 > 229
2) Second path from STO > T33 > 32 > 225 > T226 > T227 > T228 > T229
3) Third path from STO > T33 > T34 > T225 > T226 > T228 > T229
4) Fourth path from STO > T33 > T34 > T225 > T226 > T227 > T228 > T229

Of the four sample journeys and graphs can be obtained as shown in Figure 5.
Figure 5. Sample Test Graph

From the sample graph, the system performs the Dijkstra algorithm calculation as following the steps:
1. Input the initial node, destination node and graph model. In this case, the initial node is STO, the destination node is T229. The graphs are STO, T32, T33, T34, T225, T226, T227, T228, T229.
2. Set the initial node with the status found and visited. The specified node is the STO node. While the status of other nodes is not yet found and visited, the specified node is the STO node. While the status of other nodes is not yet found and visited, Table 2 shows the initial search conditions.

| Node | Status  | Distance | Route |
|------|---------|----------|-------|
| STO  | Visited | STO      | STO   |
| T32  | Not found yet | -     | -     |
| T33  | Not found yet | -     | -     |
| T34  | Not found yet | -     | -     |
| T35  | Not found yet | -     | -     |
| T225 | Not found yet | -     | -     |
| T226 | Not found yet | -     | -     |
| T227 | Not found yet | -     | -     |
| T228 | Not found yet | -     | -     |
| T229 | Not found yet | -     | -     |

3. Search for each node that can be reached directly from the node being visited. Because at that time the STO node is being visited, a node is directly connected to the STO node. In this case, the node found is node 33, with accumulated weights (the total weight that can be reached from the initial node to the node in the route being traced) of 46 meters.
Figure 6. Graph condition where T33 node is found

4. If the node obtained in step 3 has never been found, then change the status to be found. But if the node obtained in step 3 has already been found, then update the weight, and take the smaller one. At this time only the T33 node has never been found, so it changes its status to be found.

5. The next step is to look for nodes that have the smallest weight of all nodes that are found and visit them. For now the nodes found with the smallest weight are T33 which weighs 46. The T33 node changes to be visited.

6. Then back in step 3, the search is each node that can be reached directly from the node being visited. The node being visited is T33, the node that can be reached directly from T33 is the T32 node and T34 node. The sum of weights T32 = 156 meters and T34 = 196 meters.

7. Node T34 and T32 have never been found, so change the status to be found.

8. Search for nodes that have the smallest weight of all nodes that are found, then visit them. At this time, the search results in a T32 node with the smallest weight of 110. Furthermore, the algorithm is to visit the T32 node.

9. Similar to the third step, searching for nodes that are directly related to the current visited node (node T32) is the node T225. Then change the status of the T225 node to be found.

10. Look for nodes that have the smallest weight of all nodes that are found, then visit them. This search results in node T34 with the smallest weight of 196. The next step is to visit node T34.

11. Re-search the node that connects directly to the T34 node. T225 node is found, which has a sum weight of 426. Since we found the T225 weight is smaller than the previous weight, change the T225 weight to 426 from the T34 path.

Search for nodes that have the smallest weight of all nodes with status found then visit them.

Search results in node T226 with the smallest weight of 426. The next step is to visit node T226.

13. Similar to the third step, searching for nodes that are directly related to the current visited node (node T226) is the node T227. Then change the status of the T226 node to be found.

14. Search for nodes that have the smallest weight of all nodes with status found then visit them.

Search results in node T226 with the smallest weight of 506. The next step is to visit node T226.

15. Similar to the third step, searching for nodes that are directly related to the current visited node (node T226) is the node T227 with the sum of weights 642. Then change the status of the T227 dan T228 nodes to be found.
16. Searching for nodes that have the smallest weight of all nodes with status found then visit them. Search results in node T228 with the smallest weight of 642. The next step is to visit node T228.

17. Then back in step 3, the search for each node that can be reached directly from the node being visited. The node being visited is T33, the node that can be reached directly from T33 is the T32 node and T34 node. The sum of weights T32 = 156 meters and T34 = 196 meters.

18. Similar to the third step, searching for nodes that are directly related to the current visited node (node T228) is the node T229 with the sum of weights 692.

19. Searching for nodes that have the smallest weight of all nodes with status found then visit them. Search results in node T227 with the smallest weight of 644. The next step is to visit node T227.

20. Next from node T227 it will look for nodes that are directly related to the node (node 228) with the sum of weights 644 + 30 = 674. Because the weight is greater than the previous weight, the weight is not changed, and the system returns to search for nodes that have the smallest weight from nodes with status found, then visit them. Only the T229 node has a status found, so the system visits node T229 with a total weight of 692.

The steps to find the routes above are arranged in a table and can be seen in Table 3.

| Node  | Status | Distance | Routes                      |
|-------|--------|----------|-----------------------------|
| STO   | Visited| 0        | STO                         |
| T32   | Visited| 156      | STO>T32                     |
| T33   | Visited| 46       | STO>T32>T33                |
| T34   | Visited| 196      | STO>T32>T34                |
| T225  | Visited| 426      | STO>T32>T34>T225           |
| T226  | Visited| 506      | STO>T32>T34>T225>T226      |
| T227  | Visited| 644      | STO>T32>T34>T225>T226>T227 |
| T228  | Visited| 692      | STO>T32>T34>T225>T226>T227 |
| T229  | Visited| 692      | STO>T32>T34>T225>T226>T227 |

The T229 node is the destination node, the search process stops, and the shortest route is reached with the following route: STO> T32> T34> T225> T226> T228> T229 with a distance of 692 meters.

ISP service disruption location will be calculated using Floyd-Warshall algorithm to get the order of location data with the shortest distance. The system also calculate the location data using Dijkstra algorithm to obtain the location data sequence with the closest distance to the furthest distance. The final sorting begins by determining the shortest distance and then continuing with the shortest time. Table 4 shows the order of the nearest distance and the shortest time. The nearest distance and shortest time is third path.

| Starting Direction (Path) | Distance | Time       |
|---------------------------|----------|------------|
| First Path                | 698 meters| 100.512 second |
| Second Path               | 712 meters| 102.528 second |
| Third Path                | 692 meters| 99.648 second  |
| Fourth Path               | 792 meters| 114.048 second |

The result of the application calculation can be seen in Figure 7 and Figure 8 is the resulting route.
Figure 7. The Result of Application Calculation

Figure 8. The Resulting Route

The distance obtained from the application is the same as manual calculation, the shortest route from the starting point of STO location to the endpoint location is 692 meters.

5. Conclusion
This application is built to help users (ISP personnel maintenance) in determining the location of service disruptions. It is useful to save time required to deliver personnel maintenance to the nearest location. This application provides service disruption information based on distance and the nearest travel time. The initial setting begins by determining the shortest distance and then continuing with the shortest time. The shortest distance and shortest time is the third path.

A numerical example was used to illustrate the efficiency of the proposed technique. The proposed technique can be applied to real application transportation system and many other network optimization problems that can be formulated as the shortest path problem.

References
[1]. L.M.S. Bento, et al., “Dijkstra graphs”, Discrete Applied Mathematics Elsevier, 2017, pp. 1-11.
[2]. Y. Gao, “Shortest Path Problem with Uncertain arc Lengths”, Computer and Mathematics with Applications Elsevier, pp. 2591-2600, 2011.
[3]. Y. Chao, “A Developed Dijkstra Algorithm and Simulation of Urban Path Search”, The 5th International Conference on Computer Science & Education, pp. 1164-1167, 2010.
[4]. Dinitz and R. Itzhak, “Hybrid Bellman-Ford-Dijkstra Algorithm”, Journal of Discrete Algorithms Elsevier, pp. 35-44, 2017.
[5]. F.E. Andiany and W. Hadikurniawati, “Implementasi Algoritma Dijkstra Untuk Mencari Rute Terpendek Antar Kantor dan Estimasi Penggunaan Bahan Bakar Kendaraan (Studi Kasus PT. Telkom Indonesia Regional IV Jateng - DIY)”, Prosiding Seminar Nasional Multi Disiplin Ilmu Dan Call for Papers Unisbank, pp. 238-243, 2018.
[6]. L. Boguchwal, “Shortest Path Algorithms for Functional Environments”, *Discrete Optimization Elsevier*, pp. 217-251, 2015.

[7]. P. Bogdan, “Dijkstra Algorithm in Paallel-Case Study”, *IEEE*, pp. 50-53, 2015.

[8]. Y. Chao and W. Hongxia, “Developed Dijkstra Shortest Path Search Algorithm and Simulation”, *International Conference on Computer Design and Application*, pp. 116-119, 2010.

[9]. Y. Chen, S. Shen, T. Chen, R. Yang, “Path Optimization Study for Vehicles Evacuation Based on Dijkstra Algorithm”, *Procedia Engineering Elsevier*, 71, pp. 159-165, 2014.

[10]. Y. Deng, Y. Chen, Y. Zhang, S. Mahadevan, “Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment”, *Applied Soft Computing Elsevier*, 12, pp. 1231-1237, 2012.

[11]. R. Xu, L. Liu, D. Miao, “An Optimal Travel Route Plan for Yangzhou Based on the Improved Floyd Algorithm”, *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 168-176, 2017.

[12]. A. Aziz, M.M. Farid, E. Suryani, “Floyd Warshall Algorithm with FIS Sugeno FIS Search Evacuation Route Optimization”, *International Seminar on Applications for Technology of Information and Communication (Semantic)*, pp. 147-151, 2017.

[13]. J.C.D. Cruz et al. “Items-mapping and Route Optimization in a Grocery Store using Dijkstra’s, Bellman-Ford and Floyd-Warshall Algorithms”, *IEEE*, pp. 244, 2017.