Comparison of Graph Theory Algorithms for Shortest Path Problem

Yosefina Finsensia Riti[1], Jonathan Steven Iskandar[2], Hendra[3]

Informatics Study Program, Faculty of Engineering[1], [2], [3]

Darma Cendika Catholic University, Surabaya, Indonesia

yosefina.riti@ukdc.ac.id[1], jonathan.iskandar@student.ukdc.ac.id[2], hendra@student.ukdc.ac.id[3]

***Abstract*—** **Implementation of graph theory in the case of finding the shortest distance can be solved using the shortest path search algorithm, which is classified into a single source and multi-source, where the single source algorithm is the shortest path search algorithm from one source node to various other destination nodes. These algorithms are among Dijkstra, and Ballman-Ford algorithms. Meanwhile, multi-source algorithms look for the shortest path by calculating all pairs of vertices in a graph, including the Floyd-Warshall, Ant Colony, and Johnson algorithms. This study aims to compare the shortest path search algorithms with the parameters to compare the results of the shortest distance, algorithm complexity, and execution time. Based on testing of the five algorithms and adjusted to the parameters that have been compiled, it is found that each algorithm has its own type. Dijkstra's algorithm can be used to calculate the shortest route for single-source and single-destination types. It's the same as the Bellman-Ford algorithm, except that the Bellman-Ford algorithm can be used at the same time on graphs that have negative weight values. Meanwhile, the Floyd-Warshall algorithm is suitable for use on the all-pairs type. For Johnson's algorithm it can be used to determine the shortest path from all pairs of paths where the destination node is on a graph, and Ant Colony for calculating from a node to every one pair of destination nodes.**

***Keywords—*** ***Dijkstra Algorithm, Bellman-Ford Algorithm, Floyd-Warshall Algorithm, Johnson Algorithm, Ant Colony Algorithm***

***Abstrak*—** **Penerapan teori graf dalam hal pencarian jarak terpendek dapat diselesaikan dengan menggunakan algoritma pencarian jalur terpendek yang diklasifikasikan menjadi single source dan multi source, dimana algoritma single source adalah algoritma pencarian jalur terpendek dari satu node sumber ke node sumber. berbagai node tujuan lainnya. Algoritma tersebut antara lain algoritma Dijkstra, dan Ballman-Ford. Sementara itu, algoritma multi-sumber mencari jalur terpendek dengan menghitung semua pasangan simpul dalam graf, termasuk algoritma Floyd-Warshall, Ant Colony, dan Johnson. Penelitian ini bertujuan untuk membandingkan algoritma pencarian jalur terpendek dengan parameter untuk membandingkan hasil jarak terpendek, kompleksitas algoritma dan waktu eksekusi. Berdasarkan pengujian terhadap kelima algoritma tersebut dan disesuaikan dengan parameter yang telah disusun, diketahui bahwa setiap algoritma memiliki jenisnya masing-masing. Algoritma Dijkstra dapat digunakan untuk menghitung rute terpendek untuk tipe single-source dan single-destination. Ini sama dengan algoritma Bellman-Ford, hanya saja algoritma Bellman-Ford dapat digunakan secara bersamaan pada graf yang memiliki nilai bobot negatif. Sedangkan algoritma Floyd-Warshall cocok digunakan pada tipe all-pairs. Untuk Algoritma Johnson dapat digunakan untuk menentukan jalur terpendek dari semua pasangan jalur dimana node tujuan berada pada graf, dan Ant Colony untuk menghitung dari sebuah node ke setiap satu pasang node tujuan.**

***Kata Kunci—*** ***Algoritma Dijkstra, Algoritma Bellman-Ford, Algoritma Floyd-Warshall, Algoritma Johnson, Algoritma Ant Colony***

#  Introduction

 Graph theory is one of the topics in the computer field that is studied and is a branch of discrete mathematics [1] that studies related graphs, namely the relationship between one object and another object, where the connection consists of a set of vertices/points (vertex/nodes) and sides/lines (edges) that connect one point to another point. Graph theory is used in studies related to relations or relationships between objects that are implemented to solve various problem models, one of which is the optimal path or route search problem [2][3]so that it can minimize costs [4] and time efficiency [5], optimization scheduling [6][7][8], communication network modeling [9], and other issues [9]. Searching for the shortest path is the process of finding a path between two vertices, namely from the source node to the destination node in a graph by minimizing the number of weights so that the path with the smallest weight is obtained.

 Implementation of graph theory in the case of finding the shortest distance can be solved using the shortest path search algorithm, which is classified into a single source and multi-source [10] where the single source algorithm is the shortest path search algorithm from one source node to various other destination nodes. These algorithms are among the Greedy [11], Dijkstra, and Ballman-Ford algorithms. Meanwhile, multi-source algorithms look for the shortest path by calculating all pairs of vertices in a graph, including the Floyd Warshall, Ant Colony, and Johnson algorithms.

 Some research using the Greedy algorithm includes determining the best tour packages for tour,communicatists [12] and determining the fastest routes for public transportation [13].The application of Dijkstra's algorithm includes determining the shortest path [14][15][16][17][18][19][20][21], that there is the application of the Bellman-Ford algorithm in finding the best path including the network [22], a comparison of the Bellman-Ford algorithm with other algorithms such as Prim's algorithm [23], Dijkstra [24][25][26][27]. Regarding the application of the Floyd-Warshall algorithm in finding the shortest route, they include [16][28]. The application of other shortest route search algorithms such as Ant Colony includes [1][20][21][29][30]. Meanwhile, the application of Johnson's algorithm in finding the shortest path [6]. There are also other studies that make comparisons between the Dijkstra, Bellman-Ford, and Floyd-Warshall algorithms [10].

 The shortest path search algorithms that have been mentioned have their own advantages and disadvantages which can be measured through several parameters including, the results of the shortest path or route given [24][25], negative sides [26], negative cycles [26][10], the memory allocation used [10], the complexity of the algorithm [24], as well as the execution time required for an algorithm [26][10].

 This study aims to compare the shortest path search algorithms with the parameters to compare the results of the shortest distance, algorithm complexity, and execution time

# Methodology

 The data used in this research is about the distance data between tourist objects in Southwest Sumba and West Sumba. The data is taken from google maps and consists the distance between TMC Airport to Kita Beach, and also to Kabisu Lobo Oro Site, Lendongara Hill, Waikelo Beach, Kawona Beach, Karakar Indah Beach, Waikuri Lagoon, Mandorak Beach, Tanjung Karoso Beach, Tanjung Karoso Resort, Rotenggaro Beach, Bawana Beach, Watu Malandong Beach, Sumba Culture House, Praijing Village, Lailiang Beach, Rua Beach, and Mata Yangu Waterfall. The route data was taken with the consideration of good road access, paved and two-way without any damage.

The following Fig. 1 is for distance data between one tourist object and another in Southwest Sumba.



1. Distance Data between Tourist Objects

 The algorithms used in this study include Dijkstra, Bellman-Ford, Floyd-Warshall, Johnson, and Ant Colony algorithms. Dijkstra's algorithm is an algorithm that is used to find the shortest distance or route from the initial node to the final node in a weighted graph. Dijkstra's algorithm is that it is the most important and useful algorithm for optimal solutions in a large class of shortest path problems [14]. Bellman-Ford algorithm can also be used to solve problems related to the shortest path problem where this algorithm can work even though there are negative edge weight values. This algorithm itself is a refinement of Dijkstra's algorithm. Bellman-Ford algorithm offers a dynamic solution for all nodes in a graph to determine the minimum route with edge weights that can be negative, but still does not contain negative cycles [23]. Floyd-Warshall algorithm is a dynamic algorithm and is often used to determine the shortest route between all points on a directed graph without negative cycles. This algorithm is presented to find solutions to the shortest-path problem between certain fixed nodes and other related nodes [16]. Johnson's algorithm is also one of the commonly used algorithms to solve the shortest route problem. This algorithm is a combination of the Bellman-Ford and Dijkstra algorithms. Johnson's algorithm, among other things, can be used on negative weighted graphs. [6] In addition, Johnson's algorithm is also an appropriate solution method for solving scheduling problems such as the research by M. Okwu and I. Emovon [7] and by M. Redi and M. Ikram [8]. Ant Colony Optimization (ACO) algorithm is an optimization algorithm that uses probabilistic techniques and is used to solve computational problems and find optimal paths. The Ant Colony Optimization strategy will choose the shortest path based on the path most frequently traveled by ants, using mechanisms that mimic behavior or social strategies that exist in nature [31]. In testing the five algorithms, it is also assessed based on the type of shortest-path, regarding which algorithm is suitable for use in finding solutions based on a particular type of shortest-path. There are several types such as single-pair, then single-source, single-destination, and all-pairs shortest path problem.

 The parameter used as an indicator of the assessment of an algorithm being tested. The first is related to the calculation of the shortest distance, then also related to complexity, execution time, and the final result of the distance that being traveled. Calculation of the shortest distance is to find the path between the initial node and the final node so that the number of edges with minimum weight is found. For example, that a node can be connected directly to another node through one edge or a series of edges. So this research has observed visually, that there may be shorter 'distances' between some vertices and others in the graph [17]. Complexity is a parameter that provides information regarding how complicated aspects of the algorithm used are. Complexity here actually has a broader definition, in the sense that this parameter can cover several aspects such as the complexity of memory usage, space, running time [18][19][20], and so on. Execution time is used to find out how efficient and effective an algorithm is in terms of time to process data. In research by Abusalim, et al. [27]Studying that the number of nodes that make up the graph has an influence on the fast or slow running time when executing certain algorithms. Then, for the final result is to talk about how the output is generated in the program.

 The research was carried out through several stages, such as receiving input in the form of data used, then proceeding with the determination of the starting point and destination point used during the process. Implementation was carried out on five algorithms and through these implementations, followed by an analysis based on the parameters that have been compiled to obtain results. The following Fig. 2 is to describe the flow of the research through a flowchart.



1. Distance Data between Tourist Objects

 Based on the flowchart, the first step is to input the data. In this case, the dataset that will be used in the research. From this dataset, the starting point and destination point are determined as a starting point for starting the test. After determining the point of reference, the implementation is carried out with five algorithms with the help of the Python program. From the tests carried out, then the results and comparisons are analyzed referring to the parameters that have been compiled, related to the calculation of the shortest distance, complexity, execution time, and the final result of the distance traveled.

# Result and Analysis

The results and analysis of the shortest-path search calculation use 19 data from the Sumba Region nodes with five algorithms, including the Dijkstra, Bellman-Ford, Floyd-Warshall, Johnson, and Ant Colony algorithms. From these algorithms, modeling is implemented using Python programming language with Spyder application (Python 3.7). For the explanation as follows.

## Dijkstra’s Algorithm

### Calculation of the shortest distance

 The steps for calculating Dijkstra's algorithm use an approach in determining the shortest path starting from a TMC Airport node to several other destination nodes in a graph, including the following :

(a) The first step is to mark all the nodes to be visited,

(b) Mark the selected initial node with current distance 0, and other nodes with infinity “∞”,

(c) Then update the initial node as the current node,

(d) For the current node, analyze all unvisited neighbor nodes and measure their distance by adding the current distance from the current node to the edge weight connecting the neighbor node and the current node,

(e) Compare the distance measured recently with the currently assigned distance to the neighboring nodes and take that as the new current distance from the neighboring node,

(f) After that, consider all unvisited neighbors of the current node, mark the current node as visited,

(g) If the marked destination node is visited then stops, then the algorithm has ended. If not, select the unvisited node marked with the closest distance, fix it as the current new node, and repeat the process again from step (d).

### Complexity

 The complexity when running Dijkstra's algorithm program in Python is 70 lines.

### Execution Time

 Dijkstra's Algorithm is carried out 5 times to run the Python language program, so that it can better determine the amount of time the program is running, and can take the average time. The following Table I is for the test results related to the execution time of the program.

TABLE I. Dijkstra’s Algorithm Execution Time

|  |  |
| --- | --- |
| **Execution** | **Time** |
| First Execution | 0.0013222999999982221 second |
| Second Execution | 0.0003200000000020964 second |
| Third Execution | 0.00033470000000335176 second |
| Fourth Execution | 0.0004951999999960321 second |
| Fifth Execution | 0.0007295999999996639 second |
| Average Time | 0.000640359999999873252 second |

### Final Result

 The results of the shortest distance that being traveled are calculated from TMC Airport to all destinations in the Sumba Region, with the results of the Table II data as follows.

TABLE II. The result of Dijkstra’s Algorithm

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Distance** | **Path Traversed** | **Value (km)** |
| 1 | TMC Airport | TMC Airport – TMC Airport | 0 |
| 2 | Kita Beach | TMC Airport – Pantai Kita | 19 |
| 3 | Kabisu Lobo Oro Site | TMC Airport – Kabisu Lobo Oro Site | 9 |
| 4 | Lendongara Hill | TMC Airport – Lendongara Hill | 10 |
| 5 | Waikelo Beach | TMC Airport – Waikelo Beach | 8.3 |
| 6 | Kawona Beach | TMC Airport – Kawona Beach | 8.7 |
| 7 | Karakat Indah Beach | TMC Airport – Karakat Indah Beach | 22 |
| 8 | Waikuri Lagoon | TMC Airport – Waikuri Lagoon | 42 |
| 9 | Mandorak Beach | TMC Airport – Mandorak Beach | 40 |
| 10 | Tanjung Karoso Beach | TMC Airport – Tanjung Karoso Resort – Tanjung Karoso Beach | 48.7 (from; 51) |
| 11 | Tanjung Karoso Resort | TMC Airport – Tanjung Karoso Resort | 43 |
| 12 | Rotenggaro Beach | TMC Airport – Rotenggaro Beach | 47 |
| 13 | Bawana Beach | TMC Airport – Bawana Beach | 58 |
| 14 | Watu Malandong Beach | TMC Airport - Watu Malandong Beach | 63 |
| 15 | Sumba Culture House | TMC Airport – Sumba Culture House | 6.5 |
| 16 | Praijing Village | TMC Airport – Praijing Village | 43 |
| 17 | Lailiang Beach | TMC Airport – Praijing Village – Lailiang Beach | 62 (from; 66) |
| 18 | Rua Beach | TMC Airport – Rua Beach | 59 |
| 19 | Mata Yangu Waterfall | TMC Airport – Mata Yangu Waterfall | 65 |

## Bellman-Ford Algorithm

### Calculation of the shortest distance

 Calculations with the Bellman-Ford algorithm can be used to find the shortest distance between source nodes to each node. The Bellman-Ford algorithm can be used to find the shortest route solution of the all-pairs type, but in terms of implementation, the program is executed in the form of a single-source shortest path problem, namely from a certain node to all other nodes [24]. Calculation results are displayed with good and accurate results.

### Complexity

 For the complexity of the Bellman-Ford algorithm, it is considered quite complex with 409 lines of source code.

### Execution Time

 Based on 5 experiments on the Python program, the following data obtained for testing results on program execution as follows.

TABLE III. Floyd-Warshall Algorithm Execution Time

|  |  |
| --- | --- |
| **Execution** | **Time** |
| First Execution | 0.0000004 second |
| Second Execution | 0.0000009 second |
| Third Execution | 0.0000005 second |
| Fourth Execution | 0.0000006 second |
| Fifth Execution | 0.0000004 second |
| Average Time | 0.00000056 second |

### Final Result

 For the final results to be represented through rows and columns, the following is a Table IV for variable initialization for each tourist attraction based on the dataset used.

TABLE IV. Variable Initialization

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Initialization** | **Sumba Regional Destination** | **Variable Initialization** | **Sumba Regional Destination** |
| 1 | TMC Airport | 11 | Tanjung Karoso Resort |
| 2 | Kita Beach | 12 | Rotenggaro Beach |
| 3 | Kabisu Lobo Oro Site | 13 | Bawana Beach |
| 4 | Lendongara Hill | 14 | Watu Malandong Beach |
| 5 | Waikelo Beach | 15 | Sumba Culture House |
| 6 | Kawona Beach | 16 | Praijing Village |
| 7 | Karakat Indah Beach | 17 | Lailiang Beach |
| 8 | Waikuri Lagoon | 18 | Rua Beach |
| 9 | Mandorak Beach | 19 | Mata Yangu Waterfall |
| 10 | Tanjung Karoso Beach |  |  |

 Based on the table above, the final results in the Fig. 3 below are adjusted for the variable initialization in rows and columns with their respective distance values.



1. The result of Bellman-Ford algorithm

 So, by implementing the Bellman-Ford algorithm with a complexity of 409 lines of source code and several tests, it was found that the Bellman-Ford algorithm was considered effective in being able to find the minimum distance value of each node in the dataset. However, the efficiency of the program line is considered sufficient, because when compared to the other algorithms, the application of the source code to the Bellman-Ford algorithm is executed in large numbers.

## Floyd-Warshall Algorithm

### Calculation of the shortest distance

 Calculations on the Floyd-Warshall algorithm are indeed more suitable for use for the all-pairs type [16], calculations are carried out using a Python program with 19 nodes which are then written in matrix form. From the implementation, can obtain that the calculations can run well and can find the most optimal solution, in this case the shortest route between all pairs of vertices.

### Complexity

 For the complexity of the program lines used, there are as many as 59 lines of source code.

### Execution Time

 Based on 5 experiments on the Python program, the following data obtained for testing results on Table V program execution as follows.

TABLE V. Floyd-Warshall Algorithm Execution Time

|  |  |
| --- | --- |
| **Execution** | **Time** |
| First Execution | 0.0000009 second |
| Second Execution | 0.0000005 second |
| Third Execution | 0.0000004 second |
| Fourth Execution | 0.0000006 second |
| Fifth Execution | 0.0000005 second |
| Average Time | 0.00000058 second |

### Final Result

 The final result of applying the algorithm based on each row and column is initialized as follows.

TABLE VI. Variable Initialization

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Initialization** | **Sumba Regional Destination** | **Variable Initialization** | **Sumba Regional Destination** |
| 1 | TMC Airport | 11 | Tanjung Karoso Resort |
| 2 | Kita Beach | 12 | Rotenggaro Beach |
| 3 | Kabisu Lobo Oro Site | 13 | Bawana Beach |
| 4 | Lendongara Hill | 14 | Watu Malandong Beach |
| 5 | Waikelo Beach | 15 | Sumba Culture House |
| 6 | Kawona Beach | 16 | Praijing Village |
| 7 | Karakat Indah Beach | 17 | Lailiang Beach |
| 8 | Waikuri Lagoon | 18 | Rua Beach |
| 9 | Mandorak Beach | 19 | Mata Yangu Waterfall |
| 10 | Tanjung Karoso Beach |  |  |

 The table above represents the variable initialization for each tourist attraction, then the results will be made in the form of rows and columns represented by the initialization as in Table VI. The following is a Fig. 4 for the results after applying the Floyd-Warshall algorithm to the dataset used.



1. The result of Bellman-Ford algorithm

 So, by applying the Floyd-Warshall algorithm with a source code line complexity of 59 lines and several tests, it can be obtained that the Floyd-Warshall algorithm is considered effective and efficient in finding the minimum distance value for each node in the dataset. This algorithm is also very effective for use in the problem of determining the shortest route of the all-pairs type.

## Johnson’s Algorithm

### Calculation of the shortest distance

 The stages of the Johnson algorithm calculation steps use an approach in determining the shortest path of all pairs of paths where the destination vertices in a graph are among others [32], as follows.

(a) In the first step, add a vertex S to all path points on graph G by giving it a set of 0.

(b) Run the Bellman-Ford algorithm on G' with node S as the source by finding the minimum weight, then calculating the heuristic or h[v-1] for every number of nodes in graph G.

(c) When calculating h[…], then recalculate the edges of the graph using the formula: w(u, v) = w(u, v) + h[u] – h[v].

(d) If the iteration is complete, then delete the added S node and all weights are now positive on the graph. If the calculation iteration has not been completed, repeat step (c).

(e) Run Dijkstra's shortest path algorithm for each node as the source and calculate the shortest route (u,v).

(f) If the iteration is complete, the shortest route u and v will be found. If the calculation iteration has not been completed, repeat step (e).

### Complexity

 The complexity of source code when running Johnson's Algorithm program in Python is 86 lines.

### Execution Time

 The time to run the Johnson’s algorithm in Python program implemented 5 times, so that it can better determine the amount of time the program is running and can take the average time. The following Table VII is the result of the execution of the program.

TABLE VII. Johnson’s Algorithm Execution Time

|  |  |
| --- | --- |
| **Execution** | **Time** |
| First Execution | 0.013958700000000768 second |
| Second Execution | 0.01639690000000016 second |
| Third Execution | 0.013379900000000333 second |
| Fourth Execution | 0.019444300000003523 second |
| Fifth Execution | 0.01641610000000071 second |
| Average Time | 0.0159191800000010988 second |

### Final Result

 The results are managed by Johnson's algorithm which then finds the results of the shortest distance weights with those that have been modified from vertex-1 to vertex-19 to several other vertices. The results of the distance when running the Johnson’s algorithm program with the Python programming language, found 72 shorter path data from each node that was originally searched for all destination nodes.

 So, in the data Johnson's algorithm looks for the distance from vertex-1 and vertex-19 to all destination data, by obtaining 86 lines of program complexity and 0.0159191800000010988 program time in seconds. The results obtained found 72 shorter path data from each initial node that was searched for all destination nodes.

## Ant Colony Algorithm

### Calculation of the shortest distance

 The steps for calculating the Ant Colony algorithm use an approach in determining the shortest path starting from a TMC Airport node to every one pair of destination nodes on a graph, including the following:

(a) The first step, set the value of Ant Colony optimization on the initial node. With conditions, nodes that have been visited are included in the tabu list, so they will not be visited again.

(b) After performing probabilistic node calculations, then select the next node to be assigned the Ant Colony value.

(c) Move to the next node and the visited node becomes tabu list.

(d) View all visited nodes, if not repeat step (b).

(e) Then it will record the length of the side or the distance of the route taken, and delete the entire tabu list.

(f) Determine the shortest route from the current nodes and update the pheromone, if the Ant Colony travels one full route back to the beginning.

(g) If the limit for the number of Ant Colony is the maximum iteration that has been reached and will be completed, thus determining the shortest route. If the maximum iteration has not been completed, repeat step (a).

### Complexity

 The complexity of source code when running the Python language of Ant Colony algorithm program is 84 lines.

### Execution Time

 When running the Python language program, the Ant Colony algorithm implemented 5 times, so that it can better determine the amount of time the program is running and can take the average time. In testing the execution of the Python language program, it can be seen in the Table VIII below the results of testing the program.

TABLE VIII. Ant Colony Algorithm Execution Time

|  |  |
| --- | --- |
| **Execution** | **Time** |
| First Execution | 1.2428267000000233 second |
| Second Execution | 0.8022379999999885 second |
| Third Execution | 0.8272481999999854 second |
| Fourth Execution | 1.2401753999999983 second |
| Fifth Execution | 0.8539233000000195 second |
| Average Time | 0.993282320000003 second |

### Final Result

 In this algorithm, tests were carried out on 19 ants from 19 data on the destinations in the Sumba area used. Then on the 19 data used variable initialization is applied, as in the following Table IX.

TABLE IX. Variable Initialization

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Initialization** | **Sumba Regional Destination** | **Variable Initialization** | **Sumba Regional Destination** |
| 1 | TMC Airport | 11 | Tanjung Karoso Resort |
| 2 | Kita Beach | 12 | Rotenggaro Beach |
| 3 | Kabisu Lobo Oro Site | 13 | Bawana Beach |
| 4 | Lendongara Hill | 14 | Watu Malandong Beach |
| 5 | Waikelo Beach | 15 | Sumba Culture House |
| 6 | Kawona Beach | 16 | Praijing Village |
| 7 | Karakat Indah Beach | 17 | Lailiang Beach |
| 8 | Waikuri Lagoon | 18 | Rua Beach |
| 9 | Mandorak Beach | 19 | Mata Yangu Waterfall |
| 10 | Tanjung Karoso Beach |  |  |

 Then, testing was carried out and the shortest distance was found from the iteration of the ant in the form of a circuit. The following Table X represents the test results data with the Ant Colony algorithm.

TABLE X. Variable Initialization

|  |  |  |
| --- | --- | --- |
| **Ant Iteration** | **Path Traversed** | **Value (km)** |
| 1st | 1 - 15 - 3 - 4 - 2 - 16 - 17 - 18 - 19 - 13 - 14 - 12 - 10 - 8 - 9 - 11 - 7 - 6 - 5 - 1 | 385.3 |
| 2nd | 1 - 15 - 3 - 2 - 4 - 5 - 6 - 7 - 8 - 9 - 11 - 10 - 12 - 13 - 14 - 18 - 17 - 16 - 19 - 1 | 342.0 |
| 3rd | 1 - 15 - 5 - 6 - 7 - 9 - 8 - 11 - 10 - 12 - 14 - 13 - 17 - 16 - 19 - 18 - 2 - 4 - 3 - 1 | 376.0 |
| 4th | 1 - 15 - 5 - 6 - 7 - 9 - 8 - 11 - 10 - 12 - 13 - 14 - 19 - 16 - 18 - 17 - 3 - 4 - 2 - 1 | 382.0 |
| 5th | 1 - 15 - 6 - 5 - 3 - 4 - 2 - 18 - 17 - 16 - 19 - 13 - 14 - 10 - 11 - 12 - 8 - 9 - 7 - 1 | 402.0 |
| 6th | 1 - 6 - 5 - 3 - 4 - 2 - 16 - 17 - 18 - 19 - 7 - 15 - 9 - 8 - 11 - 10 - 12 - 13 - 14 - 1 | 428.0 |
| 7th | 1 - 6 - 5 - 15 - 3 - 4 - 2 - 16 - 19 - 18 - 17 - 13 - 14 - 12 - 10 - 11 - 9 - 8 - 7 - 1 | 374.0 |
| 8th | 1 - 5 - 6 - 7 - 19 - 16 - 17 - 18 - 14 - 13 - 12 - 9 - 8 - 10 - 11 - 15 - 4 - 2 - 3 - 1 | 380.0 |
| 9th | 1 - 15 - 6 - 5 - 3 - 4 - 2 - 11 - 10 - 9 - 8 - 7 - 12 - 13 - 14 - 18 - 17 - 16 - 19 - 1 | 395.0 |
| 10th | 1 - 3 - 4 - 2 - 5 - 6 - 7 - 8 - 9 - 11 - 10 - 12 - 13 - 14 - 18 - 17 - 16 - 19 - 15 - 1 | 337.5 |
| 11th | 1 - 15 - 3 - 4 - 2 - 6 - 5 - 9 - 10 - 11 - 12 - 13 - 14 - 17 - 18 - 16 - 19 - 7 - 8 - 1 | 452.0 |
| 12th | 1 - 5 - 6 - 15 - 3 - 4 - 2 - 16 - 19 - 7 - 8 - 9 - 11 - 10 - 12 - 14 - 13 - 17 - 18 - 1 | 447.0 |
| 13th | 1 - 15 - 3 - 4 - 2 - 5 - 6 - 7 - 9 - 8 - 10 - 11 - 12 - 13 - 14 - 16 - 17 - 18 - 19 - 1 | 380.0 |
| 14th | 1 - 15 - 3 - 5 - 6 - 7 - 9 - 8 - 11 - 10 - 12 - 14 - 13 - 18 - 17 - 16 - 19 - 2 - 4 - 1 | 343.0 |
| 15th | 1 - 15 - 6 - 5 - 7 - 8 - 9 - 11 - 10 - 12 - 14 - 13 - 19 - 16 - 17 - 18 - 3 - 4 - 2 - 1 | 389.0 |
| 16th | 1 - 5 - 6 - 7 - 8 - 9 - 11 - 10 - 12 - 14 - 13 - 19 - 18 - 17 - 16 - 2 - 15 - 3 - 4 - 1 | 385.0 |
| 17th | 1 - 15 - 7 - 6 - 5 - 3 - 4 - 2 - 18 - 17 - 16 - 19 - 13 - 14 - 12 - 10 - 11 - 9 - 8 - 1 | 400.0 |
| 18th | 1 - 15 - 5 - 6 - 7 - 9 - 8 - 11 - 10 - 12 - 13 - 14 - 18 - 17 - 19 - 16 - 3 - 4 - 2 - 1 | 353.0 |
| 19th | 1 - 5 - 6 - 15 - 7 - 8 - 9 - 11 - 10 - 12 - 13 - 14 - 18 - 16 - 17 - 19 - 3 - 4 - 2 - 1 | 377.0 |

 So, in the Ant Colony data algorithm, find the distance from TMC Airport by visiting all 18 destinations and then returning to TMC Airport, obtaining 84 lines of program complexity and 0.993282320000003 program time in seconds. The results obtained are 337.5 Km, with the following routes: TMC Airport - Kabisu Lobo Oro Site - Lendongara Hill - Kita Beach - Waikelo Beach - Kawona Beach - Karakat Indah Beach - Waikuri Lagoon - Mandorak Beach - Tanjung Karoso Resort - Tanjung Beach Karoso - Rotenggaro Beach - Bawana Beach - Watu Malandong Beach - Rua Beach - Lailiang Beach - Praijing Village - Mata Yangu Waterfall - Sumba Cultural House - TMC Airport.

##### Conclusion

Based on testing of the five algorithms and adjusted to the parameters that have been compiled, it is found that each algorithm has its own type. Dijkstra's algorithm can be used to calculate the shortest route for single-source and single-destination types. It's the same as the Bellman-Ford algorithm, except that the Bellman-Ford algorithm can be used at the same time on graphs that have negative weight values. Meanwhile, the Floyd-Warshall algorithm is suitable for use on the all-pairs type. For Johnson's algorithm it can be used to determine the shortest path from all pairs of paths where the destination node is on a graph, and Ant Colony for calculating from a node to every one pair of destination nodes. Through the implementation of the Python program, it was found that there was a change in the dataset used, namely the Southwest Sumba data, to become data with a more optimum distance value, in this case the minimum distance value from a starting point to a destination point.

Through the Python program that has been researched, the complexity of each of the various algorithms is obtained. Dijkstra's algorithm has 70 lines of source code, Bellman-Ford has 409 source codes, Floyd-Warshall has 59 source codes, Johnson has 86 source codes, and Ant Colony has 84 source codes. Thus, the algorithm with the lowest complexity is owned by Floyd-Warshall algorithm.

Then, Dijkstra's algorithm with execution time running in the Python program obtained an average time of 0.00064035999999873252 seconds, Bellman-Ford's algorithm for 0.00000056 seconds, Floyd-Warshall's algorithm for 0.00000058 seconds, Johnson's algorithm for 0.0159191800000010988 seconds, and Ant Colony's algorithm for 0.993282320000003 seconds. Thus, the Bellman-Ford and Floyd-Warshall algorithms are considered to have the fastest execution time.

Based on all the parameters that have been examined, it can be concluded that the Ant Colony algorithm has high effectiveness for solving the most efficient route finding for all nodes. And the Floyd-Warshall algorithm is considered to be the most effective algorithm in finding all-pairs type routes, and is able to find the most optimal distance value between one node and another node. Also, Bellman-Ford has the fastest execution time when a Python program is run.

##### Acknowledgment

Collate Thanks to LPPM Darma Catholic University of Cendika for its contribution to support this research in the form of internal grant funding.

References

[1] B. S. N Assistant professor GFGC, “A Study on Graph Coloring,” *Int J Sci Eng Res*, vol. 8, no. 5, 2017, [Online]. Available: http://www.ijser.org

[2] Tirastittam Pimploi and Waiyawuththanapoom Phutthiwat, “Public Transport Planning System by Dijkstra Algorithm Case Study Bangkok Metropolitan Area,” *International Journal of Computer and Information Engineering*, vol. 08, 2014.

[3] M. Iqbal, K. Zhang, S. Iqbal, and I. Tariq, “A Fast and Reliable Dijkstra Algorithm for Online Shortest Path,” *International Journal of Computer Science and Engineering*, vol. 5, no. 12, pp. 24–27, Dec. 2018, doi: 10.14445/23488387/IJCSE-V5I12P106.

[4] Z. Jiang, V. Sahasrabudhe, A. Mohamed, H. Grebel, and R. Rojas-Cessa, “Greedy algorithm for minimizing the cost of routing power on a digital microgrid,” *Energies (Basel)*, vol. 12, no. 16, Aug. 2019, doi: 10.3390/en12163076.

[5] A. Kejriwal and A. Temrikar, “Graph Theory and Dijkstra’s Algorithm:A solution for Mumbai’s BEST buses,” *The International Journal of Engineering and Science (IJES) ||*, pp. 23–42, 2019, doi: 10.9790/1813-0810014047www.theijes.com.

[6] Umamaheswari K., Pavithra A., Srinivashini S., and Subipriya S., “Tackling a Shortest Path Having a Negative Cycle by using Johnson\_s Calculation,” *TEST Engineering & Management*, vol. 81, 2019.

[7] M. Okwu and I. Emovon, “Application of Johnson’s algorithm in processing jobs through two-machine system,” *Journal of Mechanical and Energy Engineering*, vol. 4, no. 1, pp. 33–38, Aug. 2020, doi: 10.30464/jmee.2020.4.1.33.

[8] M. Redi and M. Ikram, “Dimension Reduction and Relaxation of Johnson’s Method for Two Machines Flow Shop Scheduling Problem,” *Sultan Qaboos University Journal for Science [SQUJS]*, vol. 25, no. 1, p. 26, Jun. 2020, doi: 10.24200/squjs.vol25iss1pp26-47.

[9] C. Dalfó and M. À. Fiol, “Graphs, friends and acquaintances,” *Electronic Journal of Graph Theory and Applications*, vol. 6, no. 2, pp. 282–305, 2018, doi: 10.5614/ejgta.2018.6.2.8.

[10] X. Z. Wang, “The Comparison of Three Algorithms in Shortest Path Issue,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Oct. 2018. doi: 10.1088/1742-6596/1087/2/022011.

[11] A. A. B. A. Homaid, A. R. A. Alsewari, K. Z. Zamli, and Y. A. Alsariera, “Adapting the elitism on greedy algorithm for variable strength combinatorial test cases generation,” *IET Software*, vol. 13, no. 4, pp. 286–294, Aug. 2019, doi: 10.1049/iet-sen.2018.5005.

[12] A. M. Benjamin, S. A. Rahman, E. M. Nazri, and E. A. Bakar, “Developing A Comprehensive Tour Package Using An Improved Greedy Algorithm With Tourist Preferences,” 2019. [Online]. Available: https://www.researchgate.net/publication/335541687

[13] M. Elizabeth, B. Gani, M. A. Safitri, C. Lusiana, and M. Dewi, “Real Time Public Transportation Navigator System in Jakarta by Using Greedy Best First Search Algorithm,” 2018, [Online]. Available: http://www.emarketer.com/Article/Asia-Pacific-

[14] S. Orhani, “Finding the Shortest Route for Kosovo Cities Through Dijkstra’s Algorithm,” *Middle European Scientific Bulletin*, vol. 25, 2022, [Online]. Available: https://www.researchgate.net/publication/361443814

[15] R. Chen, “Dijkstra’s Shortest Path Algorithm and Its Application on Bus Routing,” 2022.

[16] V. Sakharov, S. Chernyi, S. Saburov, and A. Chertkov, “Automatization Search for the Shortest Routes in the Transport Network Using the Floyd-warshell Algorithm,” in *Transportation Research Procedia*, Elsevier B.V., 2021, pp. 1–11. doi: 10.1016/j.trpro.2021.02.041.

[17] N. Syuhada, M. Pazil, N. Mahmud, S. H. Jamaluddin, N. Farasyaqirra, and B. Mustafa, “Shortest Path from Bandar Tun Razak to Berjaya Times Square using Dijkstra Algorithm,” 2020. [Online]. Available: https://jcrinn.com

[18] G. Deepa, P. Kumar, A. Manimaran, K. Rajakumar, and V. Krishnamoorthy, “Dijkstra Algorithm Application: Shortest Distance between Buildings,” *International Journal of Engineering & Technology*, vol. 7, no. 4.10, p. 974, Oct. 2018, doi: 10.14419/ijet.v7i4.10.26638.

[19] Sari I. P., Fahroza M. F., Mufit M. I., and Qathrunad I. F., “Implementation of Dijkstra’s Algorithm to Determine the Shortest Route in a City,” *Journal of Computer Science, Information Technologi and Telecommunication Engineering*, vol. 02, pp. 134–138, Mar. 2021, doi: 10.30596/jcositte.v2i1.6503.

[20] O. Khaing, H. Htight Wai, and E. Ei Myat, “Using Dijkstra’s Algorithm for Public Transportation System in Yangon Based on GIS,” 2018. [Online]. Available: www.ijsea.com442

[21] T. Sari, A. S. Zain, and A. N. Handayani, “Application Of DIJKSTRA Algorithm For Network Troubleshooting In SMK Telkom Malang,” 2019.

[22] O. K. Sulaiman, A. M. Siregar, K. Nasution, and T. Haramaini, “Bellman Ford algorithm - In Routing Information Protocol (RIP),” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Apr. 2018. doi: 10.1088/1742-6596/1007/1/012009.

[23] P. Gupta and V. Pathak, “A Minimum Spanning Tree-based Routing Technique of FAT Tree for Efficient Data Center Networking,” *Mathematical Statistician and Engineering Applications*, vol. 71, no. 1, Jan. 2022, doi: 10.17762/msea.v71i1.44.

[24] F. Mukhlif and A. Saif, “Comparative Study On Bellman-Ford And Dijkstra Algorithms,” 2020. [Online]. Available: https://www.researchgate.net/publication/340790429

[25] Botsis D. and Panagiotopoulos E., “Determination of the shortest path in the university campus of Serres  using the Dijkstra and Bellman‐Ford algorithms,” 2020.

[26] A. Manan, S. Imran, and A. Lakyari, “Single source shortest path algorithm Dijkstra and Bellman-Ford Algorithms: A Comparative study,” *INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND EMERGING TECHNOLOGIES (IJCET)*, vol. 3, no. 2, pp. 25–28, 2019, [Online]. Available: https://ijcet.salu.edu.pk

[27] S. W. G. Abusalim, R. Ibrahim, M. Zainuri Saringat, S. Jamel, and J. Abdul Wahab, “Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization,” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing Ltd, Sep. 2020. doi: 10.1088/1757-899X/917/1/012077.

[28] O. Yu Lavlinskaya, T. V. Kurchenkova, and O. V. Kuripta, “Shortest path algorithm for graphs in instances of semantic optimization,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, May 2020. doi: 10.1088/1742-6596/1479/1/012036.

[29] Rachmad A., Syarief M., Rochman E. M. S., and R. G. Husni, “Ant colony optimization model for determining the shortest route in Madura-Indonesia tourism places,” *Journal of Mathematical and Computational Science*, 2021, doi: 10.28919/jmcs/7078.

[30] D. R. Anamisa, A. Rachmad, and E. M. S. Rochman, “Ant Colony System Based Ant Adaptive for Search of the Fastest Route of Tourism Object Jember, East Java,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, 2020. doi: 10.1088/1742-6596/1477/5/052053.

[31] T. Herianto, “Implementation of the Ant Colony System Algorithm in the Lecture Scheduling Process,” *Instal : Jurnal Komputer*, vol. 12, 2020.

[32] I. G. Ivanov, G. V. Hristov, and V. D. Stoykova, “Algorithms for optimizing packet propagation latency in software-defined networks,” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing Ltd, Feb. 2021. doi: 10.1088/1757-899X/1031/1/012072.