2023 International Conference on Smart Cities, Automation & Intelligent Computing Systems
Bali, Indonesia, 6-8 December 2023

Implementation of Dijkstra Algorithm in Vehicle Routing to Improve Traffic Issues in Urban Areas

Derrick Daniel Utomo
Computer Science Department, School of Computer Science, Bina Nusantara University,
Jakarta 11480, Indonesia,
derrick.utomo@binus.ac.id

Maria Aurelia
Computer Science Department, School of Computer Science, Bina Nusantara University,
Jakarta 11480, Indonesia,
maria.aurelia001@binus.ac.id

Steffi Maria Tanasia
Computer Science Department, School of Computer Science, Bina Nusantara University,
Jakarta 11480, Indonesia,
steffi.tanasia@binus.ac.id

Nurhasanah
Statistics Department, School of Computer Science, Bina Nusantara University,
Jakarta 11480, Indonesia
nurhasanah001@binus.ac.id

Alif Tri Handoyo
Computer Science Department, School of Computer Science, Bina Nusantara University,
Jakarta 11480, Indonesia,
alif.handoyo@binus.ac.id

2023 3rd International Conference on Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS) | 979-8-3503-1829-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICON-SONICS59898.2023.10435225

***Abstract*—Many studies have been conducted on Dijkstra Algorithm, and one of the implementations of those studies includes vehicle path selection or routing. A traditional Dijkstra Algorithm can compute simple path routing problems; however, it is not suitable for complex situations. This paper proposes an improved method of Dijkstra Algorithm that analyzes influencing factors, such as: traffic congestion, travel time reliability, weight of each equivalent path and then produces the best path according to each situation. This new method is a beneficial improvement in efficiency for shortest path search algorithms and reduces time complexity, and so, this paper applies the Dijkstra Algorithm to analyze travel routes between Bung Karno Stadium and Merlynn Park. A node diagram is created based on cost tables and transformed into cost metrics. The goal is to find the shortest route while considering traffic data and analyzing differences with road distances. The cost metrics are inserted into the Dijkstra Algorithm, resulting in minimum costs for each vertex from the source node. Two optimal travel paths are determined based on different datasets, highlighting the impact of traffic data. Further evaluations consider travel times and costs with and without traffic. Incorporating predicted traffic times into the algorithm improves travel cost predictions. This research concludes that the Dijkstra Algorithm is effective in solving the shortest route problem and can be applied to urban traffic networks with the inclusion of traffic predictions, addressing drawbacks and enhancing route planning effectiveness.**

Keywords—dijkstra algorithm, vehicle routing problem, path planning, graph theory, traffic congestion

I. Introduction

In this paper, we will be exploring ways to improve vehicle traffic-related issues within urban areas mainly by using Dijkstra Algorithm for vehicle routing and the effects it brings to enhance safety and efficiency within urban traffic which will have an impact on people's production and life as accident prone environments may be caused by many traffic related issues [1]. In order to circumvent traffic, drivers and passengers alike have been looking for a way in order to find optimal paths within these traffic congested areas. However, some shortest-path selection methods may not meet their requirements [2]. A recent report by a leading Global Positioning System (GPS) company, TomTom, suggested that traffic congestion has resulted in loss of time, energy consumption, commuters in major cities around the world are estimated to be around 75% greater than the actual time needed to travel [3]. The significant factors in this issue are the rapid increase in the number of vehicles and the high percentage of the global population living in urban areas. Improving traffic congestion and implementing optimal route-planning algorithms would greatly benefit over 3 billion people, reducing energy consumption and contributing to mitigating global warming.

One way to reduce traffic in urban areas would be to optimize traffic flow for more efficient routing. An improved vehicle routing system would lead to better utilization of all nearby road networks [3]. There are numerous algorithms that can be used to solve and optimize routing with Dijkstra Algorithm being one with a long history of research in the field of computer science as it has been applied for the case study to improve evacuation routes [4]. Although there is some limitation on Djikstra Algorithm, mainly the issue of the inability to handle graph with negative edge weights and the potential performance issues in very large graphs. Dijkstra Algorithm is a traversal algorithm, and with its development, many improved methods of it have appeared as an improvement to name a few such as calculation methods, storage structure, ect. upon its predecessor [5]. As there are multiple methods of incorporating the Dijkstra Algorithm, in this paper, we will take into consideration some of those proposed methods while comparing each method to find the most efficient method that can be used to solve issues regarding traffic in urban cities for their improvement, benefit, efficiency, and complexity.

Our contributions are the following:

1. We use and combine several algorithms, mainly the Dijkstra Algorithm, with other methods based on the research from various paper sources to obtain the optimal vehicle route to solve and improve the traffic issues.
2. We analyze and observe previous results of attempts from existing research regarding similar issues about vehicle routing.

II. Literature Review

A. Problem Descriptions

Traffic congestion is a significant issue that still needs to be resolved, particularly in urban areas [6]. The primary cause of traffic is due to the road being congested or even beyond road capacity, which means that there is a supply shortage and a consequent increase in accidents and traffic congestion [7].

The inefficient routing of vehicles and the waiting time, which was determined by comparing the travel time on the road segment with the time at the intersection, are two elements that are discovered to contribute to this issue [6]. These problems are frequently referred to as Vehicle Routing Problems (VRP), which has also been one of the most

researched issues in operations research due to the rising cost of emissions within the past few years [8].

B. Path Planning

Extensive research has produced effective global path planning algorithms, including graph search, random sampling, and bionic algorithms. Notably, the Dijkstra Algorithm is a key method in graph search [9] which is known as a method that is able to resolve problems involving optimization, including path planning [10].

The two main types of path planning algorithms are heuristic path planning and optimal path planning. The optimal path planning techniques calculate all costs from one point on a graph to the next in order to find the shortest route to each vertex [11]. The shortest route problem arises while attempting to determine the most direct path from a given starting point to a desired destination. The graph in the image below is a typical representation of this issue [12].

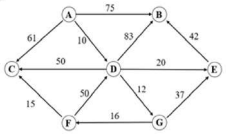

Fig. 1. Shortest path graph example

The optimal path planning algorithm explores all possible solutions to find the globally optimal path. Incremental graph algorithms can be incorporated into optimal algorithms. On the other hand, heuristic path planning algorithms iteratively refine a subset of solutions, aiming for a feasible but potentially suboptimal solution. While not guaranteed to be globally optimal, these algorithms often find solutions of comparable quality. To begin, the heuristic algorithm requires a destination point [11].

C. Routing Algorithms

Numerous vehicle routing algorithms exist, including Bellman-Ford, Floyd Warshall, and A* search algorithm [3]. However, the Dijkstra Algorithm is the most well-known and quickest labeling technique [13] It was developed by Dutch computer scientist Edsger Wybe Dijkstra in 1959 [14], and it is a greedy method-based algorithm that determines the shortest route between the origin, or "node," and the destination, or "vertex" [15]. According to Suardinata et al., Dijkstra algorithm has a minimum length from vertex a to z in a weighted graph which is a positive number and cannot be passed by negative nodes as stated in these expressions [16]:

- The length of a trajectory in a weighted graph is the sum of the weights – *(u, v)* denoted by *l(u, v)*.
- The shortest path with the initial v_0 and end nodes v_k is defined as a path with a minimum length from v_0 to v_k. If there are m different paths, then the shortest is between v_0 and v_k expressed as $l_t(v_0, v_k) = \min_{i=1,\ldots,m} l_i(v_0, v_k)$.

The algorithm is represented briefly as: G = (V, E) where V is a set of vertices and E is a set of edges [17].

Dijkstra started off the research in the grid-map based path planning by examining the shortest path between two nodes on a map [18]. In particular, the Dijkstra Algorithm is a graph search method that may be used to efficiently determine the shortest paths between two points [19]. This approach is applied to solve the Vehicle Routing Problem. Road networks are modeled using graph theory, and computer algorithms efficiently find vehicle routes. Intersections are represented as nodes connected by edges with cost values from departure to destination [4].

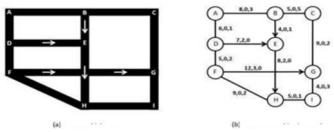

Fig. 2. Graph model of a road network

Dijkstra's method uses a priority queue to trace the highest-priority point. It assigns values to each point and maintains their value until the inspection point is reached, allowing for comparison with newly determined route values [10].

The advantage of using the Dijkstra Algorithm is that it requires that every edge's weight be non-negative since otherwise the path cannot be followed and the results could be incorrect. The algorithm's calculation will utilize the starting point as its center point, then layers of expansion will be applied outward until the target node is found [20].

The Dijkstra algorithm is commonly used in real-time applications, such as geographic mapping, for optimal path planning and considering traffic lights' duration. Additional research incorporates the time dimension into the algorithm [21]. Modified Dijkstra methods were used for finding shortest paths in networks with time-varying links and for vehicle path planning. The calibration technique effectively handles sensor node measurement quality. The Dijkstra algorithm was proposed as a search solution in such circumstances [22].

Dijkstra's method involves assigning certain initial distance values and attempting to incrementally work on them as stated below [23]:

1. Assign a tentative distance value to each node; set it to zero for the first node and to infinity for all other nodes (undefined).
2. Mark the first node as the "start" node and the rest as "unvisited."
3. Using the "start" node, estimate the tentative distances between all of its unvisited neighbors. Compare the newly discovered tentative distance to the "start" node value to assign the smaller one.

   If [dist(u) + cost(u, v) < dist(v)]

   dist'(v) = dist(u) + cost(u, v)
4. After visiting all of the neighbors from the "start" node, mark those neighbors as "visited." A "visited" node will never be checked again, and the smallest number is the most recent tentative distance.
5. Go back to step 3 and change "start" to the "unvisited" node with the smallest tentative distance.

The creation of a graph with nodes and edges using a road network obtained from Google Maps is depicted in Figure 3.a and 3.b below:

Fig. 3. (a) Graph from Google Maps (b) Graph with nodes and edge using road network from google maps

D. Previous Work and Proposed Improvements

Although a lot of improvements to Dijkstra Algorithm and its applications have been made, none have been able to address the dynamically changing variables of traffic density, average road speed, travel time, and traffic flow [3]. To consider intersection delays in urban traffic networks, a reverse labeling Dijkstra algorithm (RLDA) was developed based on the original method. RLDA's runtime decreases with the network's node count. As the online shopping industry grows, efficient transportation routes and faster deliveries become crucial. This has increased interest in the Dijkstra algorithm, leading to innovations. However, the algorithm's efficiency decreases due to the need to record and calculate numerous routes, resulting in redundant computations for routes not used in the final path planning.

In a vector normalization approach provided by Rosita et al., the cost is computed by inverting and dividing the weights (r_{ij}) by the square of all the weights inside the parameter, whereas the benefit parameter uses the same formula but does not use inversion [24].

$$Cost = n_{ij} = 1 - \frac{r_{ij}}{\sqrt[2]{\sum_{i=1}^{m} r_{ij}^2}}$$

$$Benefit = n_{ij} = \frac{r_{ij}}{\sqrt[2]{\sum_{i=1}^{m} r_{ij}^2}}$$

On the other side, Muhamad Akram et al., also suggested an additional Dijkstra Algorithm expansion. The Fuzzy Dijkstra technique, which this extension extends [25]. After allocating zero and designating node 's' as a permanent node, infinity is assigned as the distance to every other node, which will be designated as temporary only. Now, the active node is assigned to 's'. Afterwards, a group "T" of comprising nodes will then be regarded as having temporary labels. The (I, j) linkages from the current node are taken from these labels. Defuzzification of PFNs connected to links comes next, and for each j ∈ T, the j node and likewise dj will then be replaced by min{d_j, $d_i + c_{ij}$.} The node 'j' with the smallest distance from all other nodes in 'T' will then be identified. That node has now been upgraded, given a permanent label, and is currently the active node, afterwhich, all temporary labels will thus subsequently be changed into permanent labels and the procedure will be complete whenever any temporarily labeled node is out of reach.

Fitriansyah et al., applied Dijkstra Algorithm in their research to find the shortest path of tourist destination in Bali. The top 10 tourist destinations on Bali were converted into a node diagram. The source node's distance was set to zero, while all other node distances were initialized as infinity to avoid negative edges. Distances from adjacent nodes were then used to update nodes with an initial distance of infinity. The algorithm repeatedly selects the closest node (minimum distance) as the current node and updates distances until the destination node is visited. This iterative process calculates the minimum distances from the source node to all other nodes in the graph [14].

A new method dubbed the Modified Dijkstra's Shortest Path algorithm by Iqbal et al. employs numerous parameters rather than just one to discover a legitimate shortest path [26]. The time complexity of this method is $O((V + E) * log(V))$. Traffic congestion in urban areas hampers economic prosperity by wasting time and money. People's attempts to avoid congestion by starting early are often futile. To address this, an efficient system is required to assess road conditions and provide optimal routes with less traffic [27]. Since Dijkstra's algorithm does not account for memory usage, it will use a lot of memory to store the points that are skipped over during each iteration of trying to find the shortest path through a very large graph. A mixture of the node combination method and the Dijkstra algorithm is used in Fitro et al.'s search technique [28]. This node combination algorithm allows memory savings by combining the nodes that are the closest to each other into a single node.

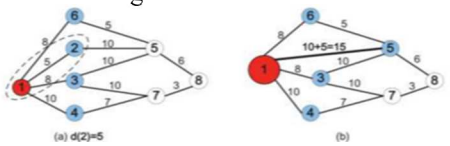

Fig. 4. An example of the combination of the two nodes with the shortest distances (a) Before combination, (b) after combination.

Zhizhu Wu et al. claim that in order to further maximize traffic, planning a traffic distribution prediction may need to be included. This can be done using a four-step process that begins with forecasting the occurrence and draw of traffic using the formula provided below.

$$O_i = BX_i$$
$$D_j = CX_j$$

While X is our attribute variable, which can include its resident population, employed population, etc., the traffic district is thought of as i, j, and. Now, B and C have two distinct meanings. B refers to our unit trip time for travel-related purposes, while C refers to the same thing but for the duration of an attraction. Calculating the minimal shortest path and right of way are both included in the second stage. The Dijkstra Algorithm is used for this. The formula for predicting traffic distribution is show as below.

$$Q_{ij} = [O_i * D_j * f(C_{ij})] / [\Sigma_j D_f * f(C_{ij})]$$

O_i and D_j are the anticipated future of occurrence and traffic attraction, respectively, and $f(C_{ij})$ is the coefficient for traffic impedance. The OD expectation line, a straight line connecting the centers of each municipality, will then be calculated. The third step begins with the modal splitting process, which divides the overall volume of traffic among the different modes of travel. After that, traffic is assigned by assigning the OD matrix of the aforementioned modes to each route in the traffic network [29].

E. Evaluation Method

For evaluation, metrics like mean Travel Time, Waiting Time, Speed, and CO2 Emissions are used to assess routing algorithms' effectiveness in finding the quickest route in smart traffic. The chosen simulation tools include SUMO, NETCONVERT, and DUAROUTER, facilitating the simulation, conversion, and routing processes [30].

III. METHODOLOGY

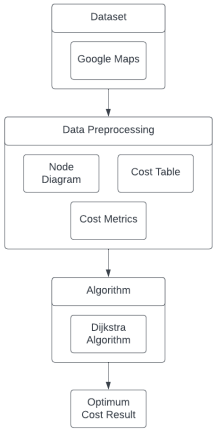

Fig. 5. Workflow of the proposed methodology

A. Dataset

We selected a road network as a sample for our research, extracting coordinate points of routes from Google Maps. Our focus is on a road with heavy traffic congestion to test our algorithm's effectiveness. Specifically, we analyze the road network between Gelora Bung Karno Stadium and Merlynn Park Hotel in Central Jakarta.

Using the provided map, we compiled major intersections and freeway sections between Gelora Bung Karno Stadium and Melynn Park. With Google Maps data, we calculated travel costs (time and distances) between intersections, forming a Node Diagram with these nodes.

TABLE I. DISTANCE AND TIME FROM SOURCE TO DESTINATION

Node	Min	Km	Source Coordinate	Destination Coordinate
X-A	8	3.1	-6.218472081365646, 106.80298042636545	-6.204120498129977, 106.80067496018411
A-B	4	1.7	-6.204120498129977, 106.80067496018411	-6.189686516363067, 106.79705115538236
B-C	1	1.1	-6.189634314902355, 106.79707118991185	-6.179538955562251, 106.7961310552124
B-P	15	6.6	-6.189686516363067, 106.79705115538236	-6.189004162723948, 106.80662300999816

C-D	3	1.7	-6.179538955562251, 106.7961310552124	-6.1676825037133804, 106.78764407606378
D-G	11	4.1	-6.179538955562251, 106.7961310552124	-6.170929398189423, 106.81099348042721
P-A	11	4	-6.189004162723948, 106.80662300999816	-6.204120498129977, 106.80067496018411
D-E	4	1.7	-6.1676825037133804, 106.78764407606378	-6.1660014757913375, 106.80205368117866
X-V	7	2.3	-6.218472081365646, 106.80298042636545	-6.2195157512172 32, 106.81215307016942
V-U	1	0.6	-6.2195157512172 32, 106.81215307016942	-6.216255958254214, 106.81622077227192
U-R	1	0.28	-6.216255958254214, 106.81622077227192	-6.21481755953913, 106.81801914198267
R-Q	2	1.4	-6.21481755953913, 106.81801914198267	-6.20310550035519, 106.82258120963387
R-S	4	1.5	-6.21481755953913, 106.81801914198267	-6.201672523968666, 106.81553258263376
Q-N	3	0.85	-6.20310550035519, 106.82258120963387	-6.195512762941531, 106.82285550668051
N-M	6	1.4	-6.195512762941531, 106.82285550668051	-6.1834072689281205, 106.82288950637397
A-T	6	1.4	-6.204120498129977, 106.80067496018411	-6.202279398124048, 106.80667252710718
M-K	5	1.4	-6.1834072689281205, 106.82288950637397	-6.171384383291367, 106.82260645618788
K-J	2	0.45	-6.171384383291367, 106.82260645618788	-6.168104698004461, 106.82088624723484
T-S	4	1.3	-6.202279398124048, 106.80667252710718	-6.201672523968666, 106.81553258263376
J-I	2	0.4	-6.168104698004461, 106.82088624723484	-6.164779316717398, 106.82002742338545
J-H	6	1.2	-6.168104698004461, 106.82088624723484	-6.17108554502637 75, 106.81112639964383
U-T	8	2	-6.216255958254214, 106.81622077227192	-6.202279398124048, 106.80667252710718
I-Z	1	0.5	-6.164779316717398, 106.82002742338545	-6.165952678120666, 106.8153472290982
S-Q	7	2.4	-6.201672523968666, 106.81553258263376	-6.20310550035519, 106.82258120963387
H-L	4	1.3	-6.171085545026 3775, 106.81112639964383	-6.182167400780821, 106.81375407642118
L-M	8	2.2	-6.182167400780821, 106.81375407642118	-6.1834072689281205, 106.82288950637397
L-O	6	1.7	-6.182167400780821, 106.81375407642118	-6.1928615257817174, 106.81468434034025
L-P	9	3.2	-6.182167400780821, 106.81375407642118	-6.189004162723948, 106.80662300999816
P-O	8	2.8	-6.189004162723948, 106.80662300999816	-6.1928615257817174, 106.81468434034025
L-K	8	2.2	-6.182167400780821, 106.81375407642118	-6.171384383291367, 106.82260645618788
O-N	7	1.4	-6.1928615257817174, 106.81468434034025	-6.195512762941531, 106.82285550668051
S-O	11	2.1	-6.201672523968666, 106.81553258263376	-6.1928615257817174, 106.81468434034025
H-G	4	0.55	-6.171085545026 3775, 106.81112639964383	-6.170929398189423, 106.81099348042721
E-F	2	0.9	-6.1660014757913375, 106.80205368117866	-6.1656186490006295, 106.81030184098444
F-Z	5	1.6	-6.1656186490006295, 106.81030184098444	-6.165952678120666, 106.8153472290982
E-G	10	2.6	-6.1660014757913375, 106.80205368117866	-6.170929398189423, 106.81099348042721
H-F	7	2.2	-6.171085545026 3775, 106.81112639964383	-6.1656186490006295, 106.81030184098444

B. Data Preprocessing

Using the cost table, two node diagrams are created. The first diagram focuses on distance, allowing the Dijkstra Algorithm to calculate the shortest distances between each vertex from the source node (Gelora Bung Karno) and the destination node (Merlynn Park). The second diagram incorporates traffic data, obtained from Google Maps during weekdays from 1-3 PM, into the time cost column. The Dijkstra Algorithm then determines the shortest route considering both traffic conditions and time distance.

Based on both node diagrams, we can create cost metrics on each node diagram type. The cost metric will be inserted into the Dijkstra Algorithm which will then calculate the shortest distances and time between each vertex which should find us the shortest route possible between the source node and every other node.

C. Dijsktra Algorithm

The Dijkstra Algorithm is a greedy algorithm used to solve shortest route problems which can also be incorporated into vehicle routing problems in a graph with non-negative side weights. It will calculate the distances from the starting vertex to the shortest distance that can be taken to reach the given destination.

We can apply the Dijkstra Algorithm by using the relaxation formula when updating a node's cost:

$$If\ [dist(u) + cost\ (u, v) < dist(v)]$$

Where 'u' is the initial vertex and 'v' is the vertex that will get updated. For example, on solving this graph above from Figure. 8 using the Dijkstra Algorithm. Where vertex 'A' is our starting point, and vertex 'F' is our destination node.

$cost(A, B) = 2,$ the distance of 'B' is now 2

$cost(A, C) = 4,$ the distance of 'C' is now 4

D, E, F will be considered as infinity for now.

Now we'll check if C needs an update by using the relaxation formula above.

$$dist(B) + cost(B\ to\ C) < dist(C)$$

$2 + 1 < 4,$ and so, we'll update 'C' to be $2 + 1 = 3$

Then, repeat the same exact steps until we get to the final node.

$$cost(dist(B), D) = 2 + 7 = 8 = dist(D)$$

$$cost(dist(C), E) = 3 + 3 = 6 = dist(E)$$

$dist(E) = 6 < 9 + 2,$ and so, we won't be updating 'E'

$$cost(dist(D), F) = 8 + 1 = 9$$

$$cost(dist(E), F) = 6 + 5 = 11$$

$$cost(dist(D), F) < cost(dist(E), F)$$

$$9 < 11$$

From that, it's found that 9 is the minimum cost from node A to F by using the Dijkstra Algorithm.

This algorithm should provide us with the best path choice(s) to reach any given destination from any given graph from the sample taken. The main idea is to apply this algorithm by using Python as the programming language to the cost diagram taken for our analysis.

D. Evaluation Method

This paper will compare the results between the dataset taken with traffic in consideration and the one without any traffic. This will be done by first showcasing the shortest paths taken from the two different datasets and then converting the without traffic results taken from purely the distances between nodes into minutes by using an average car speed approximate travel time per minute. From that we will be comparing the evaluation results with a bar chart.

The dataset with time and traffic included should show a slower travel time overall compared to the distance only dataset due to the heavy traffic in the dataset as mentioned before.

IV. Results and Discussion

A. Algorithm Results

In this chapter, we will be applying the samples to the Dijkstra Algorithm by creating a node diagram and cost metric based on the cost table. Using the road network between Bung Karno Stadium and Merlynn Park, we calculate distances from the source node. This allows us to find the shortest route and analyze the differences between data with and without traffic.

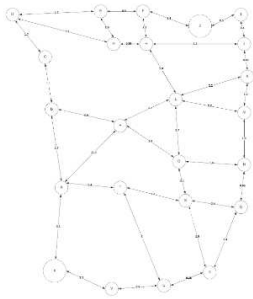

Fig. 6. Node diagram given the data taken from the distance column of the cost table

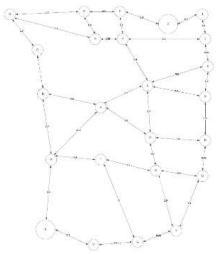

Fig. 7. Node diagram given data taken from the time spent between vertices with traffic included in time traveled

Figure 7 shows a node diagram given the data taken from the distance column of the cost table while Figure 8 shows a node diagram given data taken from the time spent between vertices with traffic included in that time traveled.

From these two diagrams, we can then formulate a cost metric taken from the cost of travel from between each vertex with x representing the source node which is Gelora Bung Karno and z being Merlynn Park.

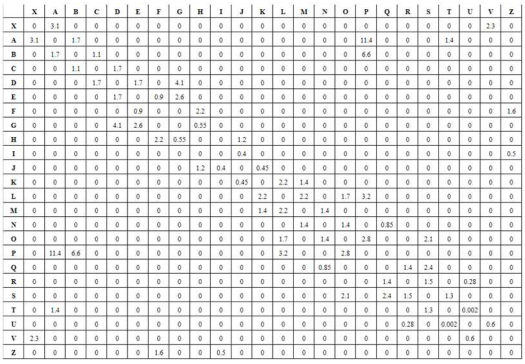

Fig. 8. Distance Cost Metric

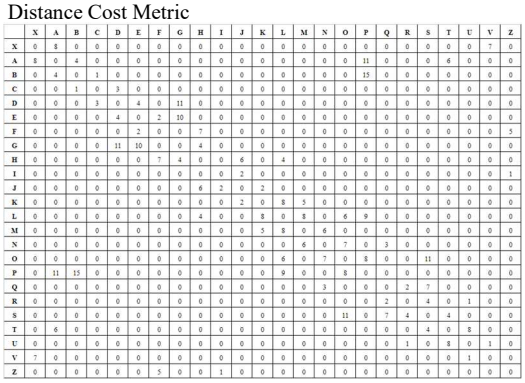

Fig. 9. Time Cost Metric

Figure 9 shows us the distance-based metric while Figure 10 gives us the time cost metric. With both cost metrics done, we can insert them into the Dijkstra Algorithm.

TABLE II. DISTANCE-BASED METRIC

Vertex	Distance from X Source (km)	Route
X	0	X
A	3.1	X A
B	4.8	X A B
C	5.9	X A B C
D	7.6	X A B C D
E	9.3	X A B C D E
F	10.2	X A B C D E F
G	10.43	X V U R Q N M K J H G
H	9.88	X V U R Q N M K J H
I	9.08	X V U R Q N M K J I
J	8.68	X V U R Q N M K J
K	8.23	X V U R Q N M K
L	8.002	X V U T S O L
M	6.83	X V U R Q N M
N	5.43	X V U R Q N
O	6.302	X V U T S O
P	9.102	X V U T S O P
Q	4.58	X V U R Q
R	3.18	X V U R
S	4.202	X V U T S
T	2.902	X V U T
U	2.9	X V U
V	2.3	X V
Z	9.58	X V U R Q N M K J I Z

TABLE III. TIME COST METRIC

Vertex	Distance from X Source (km)	Route
X	0	X
A	8	X A
B	12	X A B
C	13	X A B C
D	16	X A B C D
E	20	X A B C D E
F	22	X A B C D E F
G	27	X A B C D G
H	29	X A B C D E F H
I	28	X A B C D E F Z I
J	27	X V U R Q N M K J
K	25	X V U R Q N M K
L	27	X V U R Q N O L
M	20	X V U R Q N M
N	14	X V U R Q N
O	21	X V U R Q N O
P	19	X A P
Q	11	X V U R Q
R	9	X V U R
S	13	X V U R S
T	14	X A T
U	8	X V U
V	7	X V
Z	27	X A B C D E F Z

Table 2 displays the minimum cost of each vertex from the source node using distance as the dataset in Python 3's Dijkstra Algorithm. Table 3 shows the minimum cost of each vertex using time adjusted with traffic as the dataset.

B. Evaluation

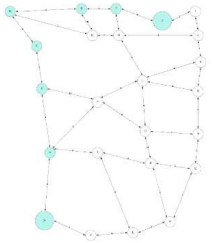

Fig. 10. Road between Gelora Bung Karno Stadium and Merlynn Park Hotel in Central Jakarta from google maps

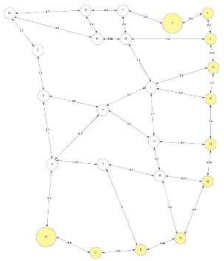

Fig. 11. Road between Gelora Bung Karno Stadium and Merlynn Park Hotel in Central Jakarta from google maps

Figure 11 and 12 depict the Dijkstra Algorithm selecting two distinct optimal travel paths with the lowest cost from X (Gelora Bung Karno) to Z (Merlynn Park), highlighting the impact of different datasets on the algorithm's outcome.

Taken from the results of the distance dataset after running it into The Dijkstra Algorithm, we now know the minimum cost of travel from the source node of Gelora Bung Karno to Merlynn Park which is 27 minutes included or 9.58 kilometers without traffic and the route is taken. We can then calculate the time needed to travel by using the minimum average and the maximum average car travel speeds. This is done by dividing each distance by the assumed average lowest speed of 0.63 km/minute and the assumed average highest speed of 0.8 km/minute.

The result of that can then be put into the Figure 13 in which the blue bars show the time from the source node to each node without traffic on the assumed lowest average speed of 0.63 km/minute while the red bars show the time from the source node to each node without traffic on the assumed highest average speed of 0.8km/minute. On the other hand, the yellow bars show the travel cost with traffic involved in the calculation prior.

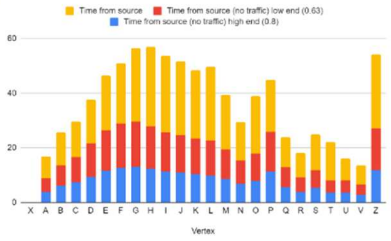

Fig. 12. Cost Evaluation Chart

Based on the chart shown above, we can see that travel costs with traffic involved are much higher than the one without traffic. From this, we know that incorporating predicted traffic times into the Dijkstra Algorithm should result in a more accurate travel cost prediction to be implemented on modern-day traffic-imbued Vehicle Routing Problem.

V. Conclusion

This research has shown that The Dijkstra Algorithm is a viable method for solving the shortest route problem to find the minimum cost of travel. It has also proven that there are various ways possible for The Dijkstra Algorithm to be incorporated into use for urban traffic congested road networks by making use of traffic predictions.

Additionally, it has been shown that the Dijkstra Algorithm's proposed method is effective in overcoming some of the traditional algorithm's drawbacks, particularly when handling complicated situations including variables like traffic congestion and trip time reliability. The improved algorithm is able to provide the optimum path appropriate for the current situation by taking these influencing elements into account and weighing the importance of each equivalent path. As a result, our research advances effective shortest-path search algorithms, reducing time complexity and improving overall route planning effectiveness in transportation networks.

References

[1] D.-D. Zhu and J.-Q. Sun, "A New Algorithm Based on Dijkstra for Vehicle Path Planning Considering Intersection Attribute," IEEE Access, vol. 9, pp. 19761–19775, 2021, doi: 10.1109/ACCESS.2021.3053169.

[2] L. Zhi, X. Zhou, and J. Zhao, "Vehicle Routing for Dynamic Road Network Based on Travel Time Reliability," IEEE Access, vol. 8, pp. 190596–190604, 2020, doi: 10.1109/ACCESS.2020.3030654.

[3] P. Udhan, A. Ganeshkar, P. Murugesan, A. R. Permani, S. Sanjeeva, and P. Deshpande, "Vehicle Route Planning using Dynamically Weighted Dijkstra's Algorithm with Traffic Prediction," May 2022.

[4] S. Kirono, M. Irfan Arifianto, R. Eka Putra, and A. Musoleh, "Graph-Based Modeling and Dijkstra Algorithm for Searching Vehicle Routes on Highways," IAEME Publication, vol. 9, no. 8, pp. 1273–1280, Aug. 2018.

[5] H. Bing and L. Lai, "Improvement and Application of Dijkstra Algorithms," Academic Journal of Computing & Information Science, vol. 5, no. 5, 2022, doi: 10.25236/AJCIS.2022.050513.

[6] L. Zhi, X. Zhou, and J. Zhao, "Vehicle Routing for Dynamic Road Network Based on Travel Time Reliability," IEEE Access, vol. 8, pp. 190596–190604, 2020, doi: 10.1109/ACCESS.2020.3030654.

[7] P. Sembiring, A. S. Harahap, and K. S. Zalukhu, "Implementation of Dijkstra's algorithm to find an effective route to avoid traffic jam on a busy hour," J Phys Conf Ser, vol. 1116, p. 022042, Dec. 2018, doi: 10.1088/1742-6596/1116/2/022042.

[8] L. Costa, C. Contardo, and G. Desaulniers, "Exact Branch-Price-and-Cut Algorithms for Vehicle Routing," Transportation Science, vol. 53, no. 4, pp. 946–985, Jul. 2019, doi: 10.1287/trsc.2018.0878.

[9] L. Liu et al., "Path Planning for Smart Car Based on Dijkstra Algorithm and Dynamic Window Approach," Wirel Commun Mob Comput, vol. 2021, pp. 1–12, Feb. 2021, doi: 10.1155/2021/8881684.

[10] A. A. Zubir, Andriansyah, and S. Derisma, "Determining the Distribution Route Using the Clustered Generalised Vehicle Routing Problem Model and Dijkstra's Algorithm," Journal of Science, Technology, and Visual Culture, vol. 2, no. 2, pp. 232–240, 2022.

[11] S. Kim, H. Jin, M. Seo, and D. Har, "Optimal Path Planning of Automated Guided Vehicle using Dijkstra Algorithm under Dynamic Conditions," in 2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA), IEEE, Nov. 2019, pp. 231–236. doi: 10.1109/RITAPP.2019.8932804.

[12] R. D. Gunawan, R. Napianto, R. I. Borman, and I. Hanifah, "Implementation of Dijkstra's Algorithm in Determining the Shortest Path (Case Study: Specialist Doctor Search in Bandar Lampung)," IJISCS (International Journal of Information System and Computer Science), vol. 3, no. 3, p. 98, Dec. 2019, doi: 10.56327/ijiscs.v3i3.768.

[13] A. Sedeño-noda and M. Colebrook, "A biobjective Dijkstra algorithm," Eur J Oper Res, vol. 276, no. 1, pp. 106–118, Jul. 2019, doi: 10.1016/j.ejor.2019.01.007.

[14] A. Fitriansyah, N. W. Parwati, D. R. Wardhani, and N. Kustian, "Dijkstra's Algorithm to Find Shortest Path of Tourist Destination in Bali," J Phys Conf Ser, vol. 1338, no. 1, p. 012044, Oct. 2019, doi: 10.1088/1742-6596/1338/1/012044.

[15] Y. Sun, M. Fang, and Y. Su, "AGV Path Planning based on Improved Dijkstra Algorithm," J Phys Conf Ser, vol. 1746, no. 1, p. 012052, Jan. 2021, doi: 10.1088/1742-6596/1746/1/012052.

[16] Suardinata, R. Rusmi, and M. A. Lubis, "Determining Travel Time and Fastest Route Using Dijkstra Algorithm and Google Map," SISTEMASI: Jurnal Sistem Informasi, vol. 11, no. 2, pp. 496–505, May 2022.

[17] O. Khaing, Dr. H. H. Wai, and Dr. E. E. Myat, "Using Dijkstra's Algorithm for Public Transportation System in Yangon Based on GIS," International Journal of Science and Engineering Applications, vol. 7, no. 11, pp. 442–447, Nov. 2018, doi: 10.7753/IJSEA0711.1008.

[18] Y. Singh, S. Sharma, R. Sutton, D. Hatton, and A. Khan, "Feasibility study of a constrained Dijkstra approach for optimal path planning of an unmanned surface vehicle in a dynamic maritime environment," in 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE, Apr. 2018, pp. 117–122. doi: 10.1109/ICARSC.2018.8374170.

[19] M. Luo, X. Hou, and J. Yang, "Surface Optimal Path Planning Using an Extended Dijkstra Algorithm," IEEE Access, vol. 8, pp. 147827–147838, 2020, doi: 10.1109/ACCESS.2020.3015976.

[20] B. Li et al., "Path planning of mobile robots based on an improved A*algorithm," in Proceedings of the 2020 4th High Performance Computing and Cluster Technologies Conference & 2020 3rd International Conference on Big Data and Artificial Intelligence, New York, NY, USA: ACM, Jul. 2020, pp. 49–53. doi: 10.1145/3409501.3409524.

[21] Yu, Jiang, and Hua, "Anti-Congestion Route Planning Scheme Based on Dijkstra Algorithm for Automatic Valet Parking System," Applied Sciences, vol. 9, no. 23, p. 5016, Nov. 2019, doi: 10.3390/app9235016.

[22] G. Pradeep Reddy, Y. V. P. Kumar, M. Kalyan Chakravarthi, and A. Flah, "Refined Network Topology for Improved Reliability and Enhanced Dijkstra Algorithm for Optimal Path Selection during Link Failures in Cluster Microgrids," Sustainability, vol. 14, no. 16, p. 10367, Aug. 2022, doi: 10.3390/su141610367.

[23] A. Candra, M. A. Budiman, and K. Hartanto, "Dijkstra's and A-Star in Finding the Shortest Path: a Tutorial," in 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA), IEEE, Jul. 2020, pp. 28–32. doi: 10.1109/DATABIA50434.2020.9190342.

[24] Y. D. Rosita, E. E. Rosyida, and M. A. Rudiyanto, "Implementation of Dijkstra Algorithm and Multi-Criteria Decision-Making for Optimal Route Distribution," Procedia Comput Sci, vol. 161, pp. 378–385, 2019, doi: 10.1016/j.procs.2019.11.136.

[25] M. Akram, A. Habib, and J. C. R. Alcantud, "An optimization study based on Dijkstra algorithm for a network with trapezoidal picture fuzzy numbers," Neural Comput Appl, vol. 33, no. 4, pp. 1329–1342, Feb. 2021, doi: 10.1007/s00521-020-05034-y.

[26] 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.

[27] M. Farhan, "Traffic Routing Algorithm for Road Network," Scientific Bulletin, vol. 24, no. 2, pp. 131–138, Dec. 2019, doi: 10.2478/bsaft-2019-0015.

[28] A. Fitro, O. S. Bachri, A. I. S. Purnomo, and I. Frendianata, "Shortest path finding in geographical information systems using node combination and dijkstra algorithm," International Journal of Mechanical Engineering and Technology (IJMET), vol. 9, no. 2, pp. 755–760, Feb. 2018.

[29] Z. Wu, M. Huang, A. Zhao, and Z. lan, "Urban Traffic Planning and Traffic Flow Prediction based on ulchis gravity model and Dijkstra algorithm," J Phys Conf Ser, vol. 1972, no. 1, p. 012080, Jul. 2021, doi: 10.1088/1742-6596/1972/1/012080.

[30] E. D. S. Derar and M. E. M. Mahmoud, "Performance Evaluation of Dijkstra and A* Traffic Routing Algorithms in Smart Cities," in 2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), IEEE, Sep. 2019, pp. 1–6. doi: 10.1109/ICCCEEE46830.2019.9070323.