Engineering and Technology Journal
Journal homepage: https://etj.uotechnology.edu.iq

A Modification of Shortest Path Algorithm According to Adjustable Weights Based on Dijkstra Algorithm

Luay S. Jabbar [a*], **Eyad I. Abbas** [b], **Sundus D. Hasan** [b]

[a] Ministry of Education, Engineer of Baghdad Education, Baghdad, Iraq.
[b] Electrical Engineering Dept., University of Technology-Iraq, Alsina'a street, 10066 Baghdad, Iraq.
*Corresponding author Email: eee.19.37@grad.uotechnology.edu.iq

HIGHLIGHTS

- Dijkstra algorithm was used in this study.
- The shortest path was found using the Dijkstra algorithm for static weights for each of the single-direction and two-direction maps
- The dynamic shortest path was found using the Dijkstra algorithm according to changing the following weights for each of single direction and two-direction maps

ARTICLE INFO

Handling editor: Ivan A. Hashim

Keywords:
Shortest path; Adaptive path weighted; Modification of Dijkstra algorithm; practical shortest path; optimal path.

ABSTRACT

It is possible to represent the road map on the paper and study it using Dijkstra`s algorithm to find the shortest path on the real earth. Dijkstra`s Algorithms are used for calculating the shortest path from source to sink to enable query operations that follow. Dynamic shortest-route techniques are needed to accommodate the modifications within the underlying community topology. Every solution includes figuring out the nodes whose shortest routes could be impacted through the updates and producing a list of affected vertices and their updated shortest pathways. In this work, the advent of the retroactive priority queue information structure makes the Dijkstra algorithm dynamic. In this research, the stepping is changed forward in the shortest direction for 2 networks and two directions. That changed by locating the shortest static path, then circulating to the first node after the beginning node. At this node, the weights inside the segments directed from this node will exchange and cancel the vintage shortest direction and locate the new direction. Then flow to the next node in the new find shortest path, and repeat the operation until the end. The idea is that the best path can be constantly changed based on the latest data. These continuous changes are addressed in this paper, where the proposed system can find the best methods and update them automatically according to the variables.

1. Introduction

A graphical application of graph theory is to follow the shortest route between two nodes, starting at one node and linking to another node through many intermediate nodes. This search includes a distance or other cost optimization, so the one optimum route has the lowest total cost. Therefore, it is appropriate to construct shortest path algorithms to solve transportation issues since the transport cost is the compensable cost of running a route between two sites. Conveniently, the automated control has the data which reveals on what route the traffic congestion occurs and how terrible the traffic is, making it possible to use these facts to calculate a route that yields the most optimal traffic patterns.

In 1980 GALLO wrote a paper on this subject, and the result found the new shortest routes in two instances from the oldest shortest routes: I alter the node from which the shortest pathways to be discovered are to be determined; ii) modify the cost of one arc; (either increased or decreased). Key Words: Shortest Path, Networks, [1], and 2007 four searchers (Bin Xiao, Jiannong Cao, Zili Shao, and Edwin H.-M. Sha) made a paper. The result presented a novel updating technique that is computer friendly and keeps unmodified nodes from the old SPT to a new SPT (DSPT) dynamic shortest path tree. The suggested technique eliminates duplication using a dynamic updating approach when an edge is retrieved from a built-in edges list Q. By probability analysis based on arbitrary tree topology, the average number of important edges is found. The DSPT method neglects the most redundant edges, which do not contribute to building a new SPT. The update is better generated from meaningful edges, and they were a study of complexity and tested findings that suggest DSPT is quicker than any other known approaches. In a graph with negative weight edges, the SPT updating issue may also be solved [2]. In 2011 YagvalkyaSharma, Subhash Chandra Saini,& Manisha Bhandhari made a. The result was found to be the shortest path for static and dynamic

http://doi.org/10.30684/etj.2022.136107.1296
Received 22 September 2022; Accepted 12 October 2022; Available online 27 November 2022
2412-0758/University of Technology-Iraq, Baghdad, Iraq

routing network ks between the source and the destination node. To locate the shortest path, first, we used the algorithm Dijkstra (AA) and, subsequently, the Genetic Algorithm (GA). The task was simulated by utilizing the GM tool. To discover the shortest way, both algorithms offer the same result. The findings appear to be two algorithms with the same potential. However, genetic algorithms have less time than the algorithm of Dijkstra [3].In 2014 Okengwu U.A .was made a paper, and the result was shown how the modified Dijsktra algorithm is applied in practice to determine the shortest route for numerous sources at hospitals within the state of Rivers [4].In 2015 Hanaa M. Abu-Ryash and Dr.Abdelfatah A. Tamimi wrote a paper. The result discussed the shortest route algorithms, such as Dijkstra, Bellman-Ford, Floyd-Warshall, and Johnson's algorithm, using the Bellman method, Ford's, which calculates the total of all other decisions. The study demonstrates that the efficiency differs across methods, suggesting which variation of the shortest path problem should be chosen for solving. There are three distinct kinds of single-pair, one-source, and one-destination problems. the Dijkstra algorithm finds a solution. In all paired shortest route issues, the Johnsons algorithm identifies a solution. The Floyd Warshall algorithm provides the shortest route among all vertical pairings by a graph analytical method. This is a dynamic planning example. The only downside to the single source publication is that the edge weight is negative. The Bellman-Ford algorithm provides a solution [5]. In the same year, a paper was written by KhyrinaAirinFariza Abu Samah, Burairah Hussin,& Abd Samad Hasan Basari. The result presented Dijkstra's approach to changing the algorithm to identify the safest and quickest way. The first correction included confining the exposure of the knot of the bottom plan and the alternate bone-blocked fire bumps. The illustration shows the results produced using the algorithm change. Therefore, the effectiveness of the change in relating the optimum route for evacuation processes is demonstrated [6]. In 2016 two groups of searchers has done a study on this topic. The first group (Elizabeth Nurmiyati Tamatjita& Aditya Wikan Mahastama) was studied to find a dynamic shortest path with dynamic weights on graphs utilizing Dijkstra's method for one-way and two ways and conclude that three findings from the study performed. To begin, Dijkstra's method can solve a shortest-path issue where the weights are dynamic and utilize a digraph with one-way edges (digraph without parallel edges). Also, Dijkstra's algorithm cannot solve the shortest route issue with dynamic weighting., making use of a single-character digraph (digraph without shunt edges). Additionally, Dijkstra's method cannot be used to find the shortest path issue with dynamic weights because a large "loop" may develop between two nodes after re-calculation. A two-way digraph (digraph with opposing parallel edges) is used. As a result, a normal road map with two-way traffic cannot be solved using Dijkstra's algorithm. To avoid this, program Along parallel edges. No matter how far apart, two nodes cannot form a single route. The third point is that, due to recalculating the shortest path, there are no performance problems. after each modification. For instance, it takes almost no computing time when positioned as a problem-solving alternative for avoiding traffic congestion on the city streets and the fire department's level of road class on the way to the fire. It is unclear if the effects on a larger network with more than 30 nodes would be apparent [7].The second group of searchers was (M. Malathi, K. Ramar 1 2 and 3C. Paramasivam, and M. Malathi). They made a paper, and the result was the hydraulic algorithm (WSA) with particle swarm optimization (PSO) implanted in this research to achieve the best way between start and destination in the Matlab coding environment. The working environment image will originally be provided as an input for the watershed algorithm to provide an initial solution, and the output will then be delivered from the watershed algorithm to optimize the path to be taken by a mobile robot to complete the task specified. Also, the Dijkstra method is presented here for the same working environment to get the shortest path. The output performance through PSO was afterward compared with Dijkstra. This approach is directed at robot planning, such as delivering materials in a factory environment and selecting places in an airplane environment. This suggested technique [8]. In 2017 Hong-mei Zhang and Ming-long Li wrote a paper. The result was the rolling window principle, Dijkstra's algorithm, and A* algorithm navigation and avoidance of obstacles occurring in real-time for mobile robots in not static environments. The process begins by computing an initial path from the beginning state to the target state. The algorithm dynamically changes this path along the traversal if a collision is expected. Many simulations were performed to aid in the study of the method. The suggested approach finds a better path in the graph during re-planning and decreases the time to re-plan the redesign by 53.3% less -99.7%, demonstrating that the technique is feasible and successful. As such, the problem-specific solution found by the problem-solving method is only suited to focusing on that particular state to find a suitable trail route. If the map data pre-processing aim is changed because of the robot's movement, the algorithm's efficiency will be substantially lowered. In our future research effort, we will use a changing target state to calculate the trail planning issue, and we can increase the algorithm's efficiency further. Moreover, we will study the procedures regulating transient system performance [9]. In 2018 Sunita did a study on this subject. The result was a study of dynamic Dijkstra's algorithm to discover the retroactive data structure paradigm used to find everything else equal, the quickest route through the Retroactive Priority Queue. Thus, the costs of all priority queue operations are linked to the size of the data structure used for search trees, as they employ height-balanced search trees to construct the retroactive priority queue. Another way to think about this is that when you do a dynamic graph modification, you're doing it after you've arrived at a point just before the moment when the edge with the change will be utilized to construct the shortest path. a method in which a change in a specific location is noted and the modification is applied exclusively to that location. It may easily adjust the solution to loosen the premise on which we've based our work. Considerable transformations, known as replacing weights, can be used to address the issue of generic graphs. The Dijkstra method can also find the shortest route of all all-pairs and solve the other special path-finding issue, known as Dijkshoogt's longest common subsequence. It may be that the solution's performance is not as good as all of the currently known best resource limits. Furthering the underlying retroactive priority queue's efficiency will improve the approach's performance [10]. In the same year Jitendra Bahadur Singh, & R.C.Tripathi, were made a paper, and the findings were presented and contrasted between the Dijkstra algorithm and the Bellman-Ford method for small and big nodes, not only nodes. The research helps to discover and recommend the method utilized in the shortest path issues for a certain variation. The runtime of Dijkstra's algorithms and in microseconds compare observations with average running time and conclude that the method of Bellman-Ford is more efficient for nodes than that of Dijkstra. Still, the algorithm of Dijkstra is

more efficient for big not [11]. In 2019 Alyasin researched this subject, and the result was for the minimum time in graphic search, the Dijkstra algorithm provided the answer for two nodes. The experimental findings were derived from the Dijkstra method gradient training to get a defined route. These findings showed great precision in selecting the shortest robot route. In the shortest feasible time and with minimal weight, the mobile robot blocked the road with the objective of at least being cheaper [12].

In this effort, the finding of the shortest path was found in the static case using static weights and dynamic case using updating the weights after moving from the start to the first node located in the static shortest path case, then taking the shortest path and moving to the next node and so on arriving to the endpoint.

The work was done for two maps in the methodology paragraph, each map has sixteen nodes and thirty-seven branches connected between two nodes, and the two maps have the same shape and the names of nodes and weights for the same branches and the angles of the same branches. Still, the first map has all branches with two opposite sides(normal segment), and the second map with single side (directed segment). The methodology section analyzed paths using the Dijkstra algorithm to find the shortest static path. Three points per map were selected as examples of starting points, and the results were arranged in the six tables in the methodology paragraph. The static shortest paths were obtained using constant weights. The dynamic shortest paths were obtained by locating the shortest path, moving from the start point to the first node in the shortest static path, and checking the new weights in the branches opposite the current node. If the same, the robot will complete the next node in the located road. If there is any update weight, we will take the sums of the weights for each path from the current node to the target, and take the shortest among them using the Dijkstra algorithm, then move to the next node in the shortest path and so on until arrived the target.

The importance of the system lies in the applications that can be adopted. For example, the proposed system can be used in large warehouses to store parcels and other goods, where the use of this system is of great benefit to organizing the work and movement of robots.

2. Theoretical Foundation

It is possible to represent the road map on the paper and study its lengths of streets and weights or costs, obstacles, or crowding that cause a delay in time and use Dijkstra`s algorithm to enable us to find the shortest path and avoid obstacles on the real earth to minimize the delay in time which symbols by weight unit.

The graph is a graphical depiction of network-enabled and the relationships between the components. A map illustrates how a series of linked nodes illustrates how reality is structured. It is a study of mathematics that is done using graphs. In this short path graph, with vertices (nodes) labeled with V and edges (links) labeled with E, the relationship is a union of two sets: the nodes of set V and the edges of set E. The size of graph G is calculated by adding the number of members in set E, while the quantity of the order of graph G is defined by the elements of set V, a graph with weighted edges statistical graphing. In this manner, the weight is put to prevent any ambiguity about the name and is inscribed next to the edge confusion. Undirected graph, or just graph or digraph. Digraphs have arrow-like points on their edges, indicating the direction)that shown in Figure 1. Dijkstra's method is used to find the shortest route using a graph in which the distance from the starting point to the endpoint is weighting the path length to identify a path with the shortest length. It should be a positive number, in other words, not a negative. v2, v3, ..., vn} where the weights are W (G) = {w1, w2, w3, ..., wn} and where W (G)={w1, w2, w3, ..., wn}from v1 through vn, Dijkstra's algorithm starts. Dijkstra's calculation will discover a successor hub that costs less than or equal to the up-to-target in each cycle. An additional node with a greater weight following-in-line vertices is set aside and excluded from the next step iteration. Those functioned as potential road conditions where the path selection was straightforward occasionally. Thus, these products are readily accessible since they are almost similar; and occasionally, just a few options exist, such as the ability to pivot. If it is required, backtrack to find the shortest route. When creating two-way road maps, the borders of maps are drawn to mimic actual conditions, such as city streets, in which there is commonly a two-way activity framework, and cars can do a U-turn to select distance better; a much better; a higher; a stronger; an improved course: hub, references [1,2,3,5,6,7,10,13,14].

2.1 Dijkstra's Algorithm

This algorithm is used to identify the shorter route without including connections with negative weights. While this method may be used to determine the quickest route between two cities, it can also be used for many other things, including creating a path on a map between two points that consider factors like street length, traffic, or how many lanes the streets have.

In the beginning position, we will name the initial node the initial node; in our final position, we will call the target node the target node, and the issue of finding the shortest routes from a source is what we will deal with.

1) Source vertex s
2) The graph must be connected
3) Weighted graph G = (E,V)
4) V to all vertices v
5) V Solution to the single-source shortest path problem in graph theory
6) Both directed and undirected graphs
7) vertex v to all other vertices in the graph,

For an example of a Single-Source Shortest Path problem, Given a weighted directed graph shown in the figure below for finding the shortest path between two given nodes 1 and 13. To find the shortest path from vertex 1 to vertex 13, it must find

the length of all paths that lead to the target, and the length of each path is equal to the sum of the weights of each of the edges in that path. Figure 1 appears to be an incident of many edges or parallel paths with the same pair of nodes.

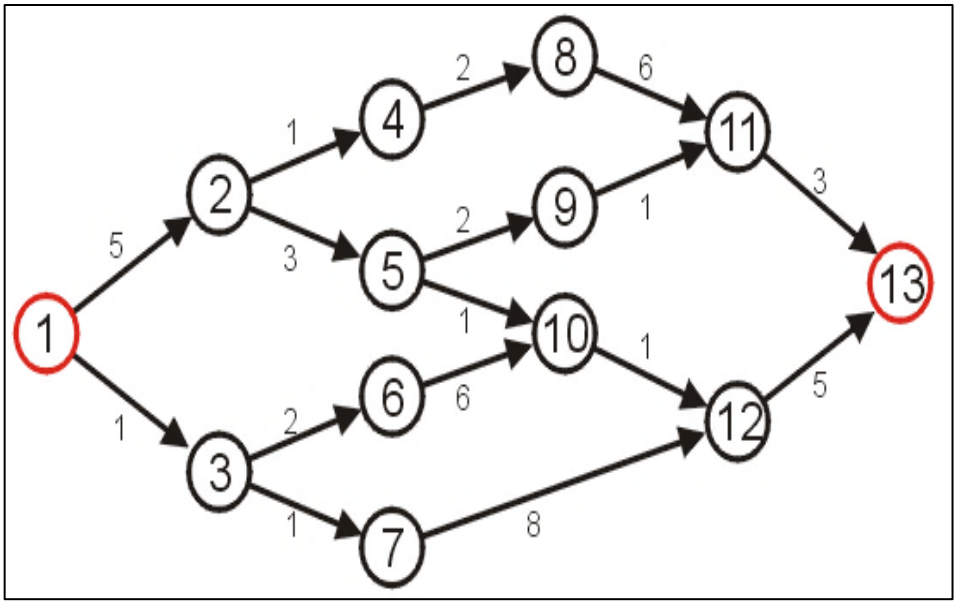

Figure 1: Minimum cost using Dijkstra algorithm

There are many paths for arriving from start to end

1-2-4-8-11-13 with length 17
1-2-5-9-11-13 with length 14
1-2-5-10-12-13 with length 15
1-3-6-10-12-13 with length 15
1-3-7-12-13 with length 15

After some consideration, we may determine that the shortest path is as follows, with a length of 14

1-2-5-9-11-13

Other paths exist, but they are longer

2.1.1 Adaptive shortest path

The dynamic obstacles that may occur in the way while moving make the shortest road longer than some roads in some time. The solution to this problem is using a dynamic layout by stopping at each follow node in the shortest path, checking if there is a new weight for the next branch of the shortest road, comparing the new weights for the following roads, and taking the shortest path. If there is no new weight, we will complete the shortest path to the next node until the target arrives. Dynamic changes utilizing a two-way digraph and single-way digraph are performed after every two sessions. Dijkstra's method is tested using a two-way digraph study done in the past have not focused on a theory that has only just recently come to light

2.2 Workflow Diagram

Figure 2 appear the procedure to find the shortest static path, and Figure 3 appear the procedure to find the shortest dynamic path.

SLP=Shortest Located Path. In other words, this path is assumed to be the shortest path in the sum of weights

$\sum$Wpn =sum of weights from start to end node for the path with number N, wpnt=weight of the last edge for the path N

$\sum$Wp1=w1+w2+....wp1t

$\sum$Wp2=w1+w2+....wp2t

$\sum$Wpn=w1+w2+....wpnt

N=N+1 mean to take the other parallel path

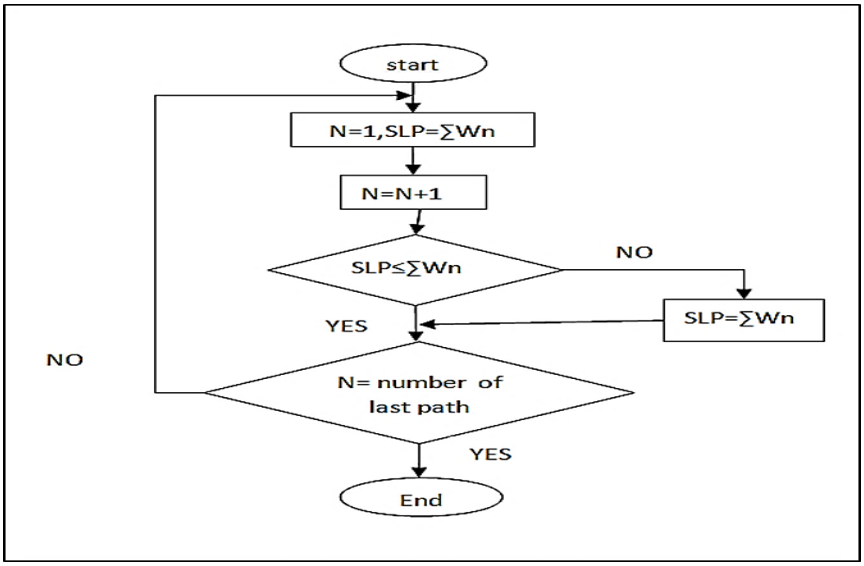

Figure 2: The flowchart of the static algorithm

For dynamic case
Wneo=The sum of old weights from this node to the end
Wnen=the sum of the new weights from this node to the end
Ws=the sum of the weights of the path that locates the shortest of other paths at this node

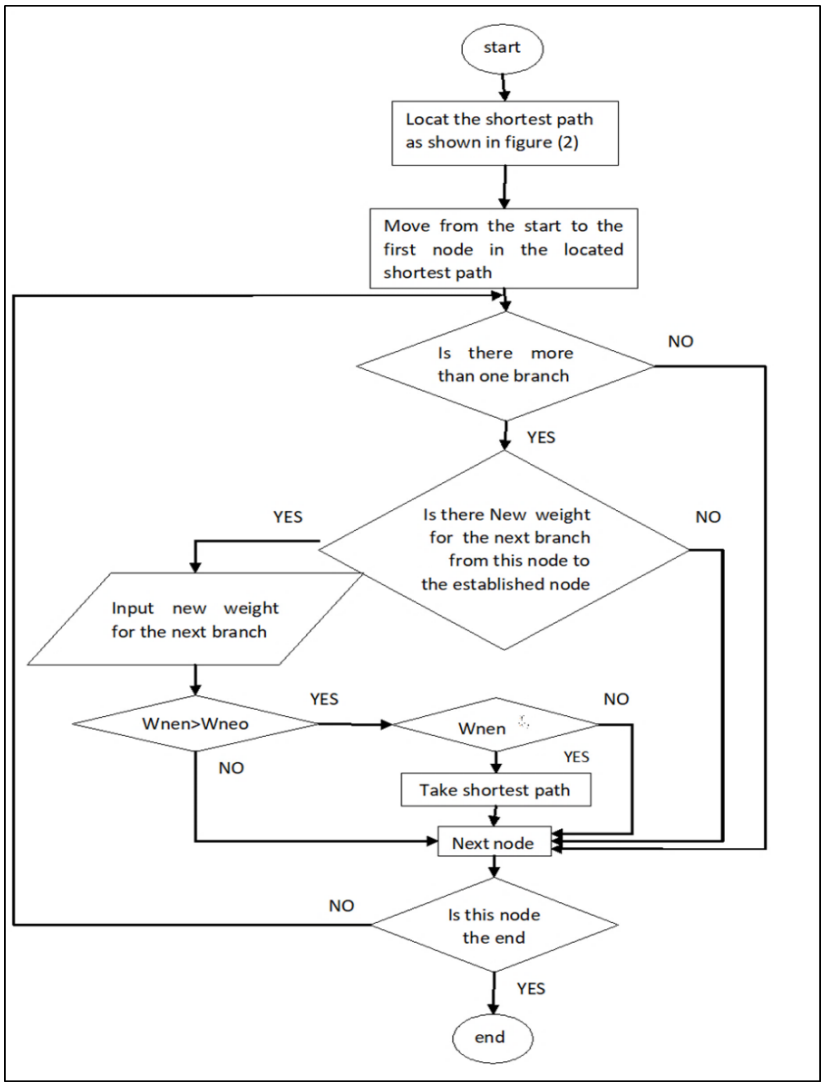

Figure 3: The flowchart of the dynamic algorithm

3. Methodology

In this paper, the two implemented networks were as shown in Figures 4, 5 below, with three suggested nodes for a start (A, B, C) and three suggested nodes for the target(end),(N, O, P) for each map. Also, it may take any node to start or end. The two maps were similar in shape, nodes, and weights, but the first was for represented the edges with normal segments (two-way or two sides), as shown below, and the second represented the edges with arrows (one side), as shown in Figures 4, 5 below. It was done by taking six different paths for each graph for different starts and targets, three for a static case for Figure 4, and three for Figure 5 for the same starts and ends. In the dynamic case, the same starts and target nodes were taken, but the difference was by changing the weights at each node in the shortest path.

It can consider the points (A, B, and C) as start points for each map and analyze the shortest paths in the static case for each node as shown in Tables 1, 2, 3, 4, 5, 6.

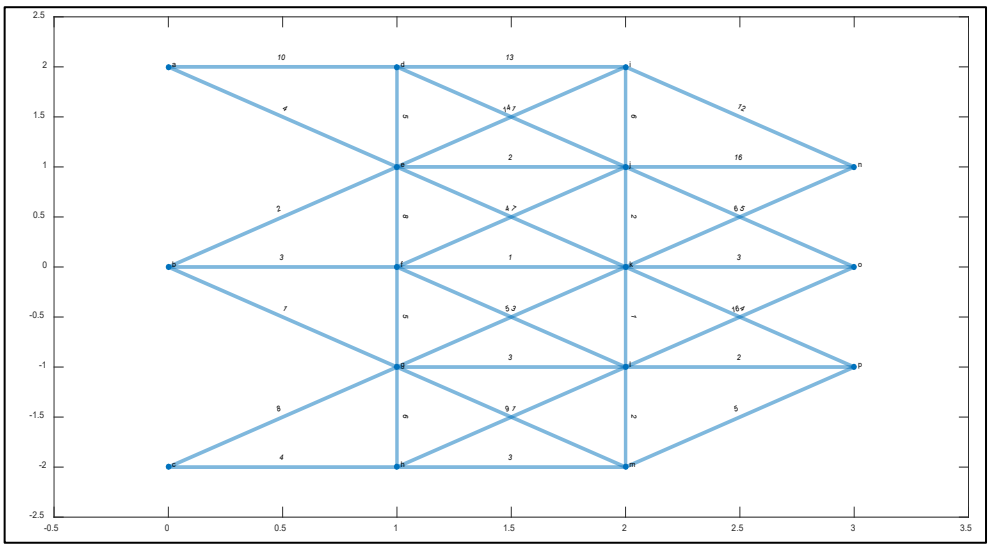

Figure 4: Suggested graph for an undirected edge

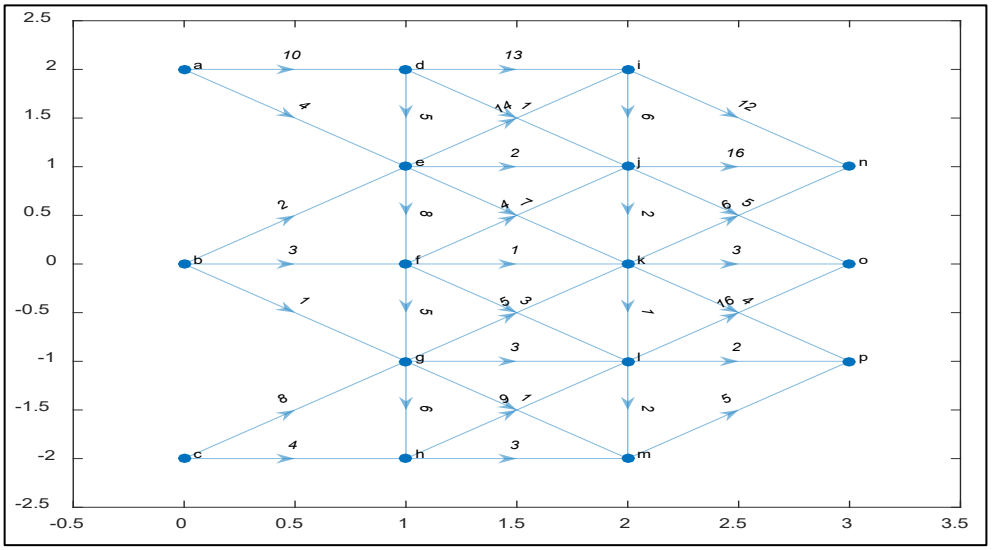

Figure 5: Suggested graph for a directed edge

3.1 Two Way Case

In this case, each edge takes two direct sides. Therefore, Table 1consists of a number of rows and columns equal to the number of nodes plus one.

1. First of all, write the name of nodes in the first row of the table from the second position to the end, then choose any point as a starting point. For example, A in Table 1 is shown below. The second step is writing the start node A in the first position of the second row, then finding the weight from point A to each node according to three steps, and writing each weight to the node under the name node directly in the second row.

    a) The weight from start point A to the same point A is zero
    b) If there is no direct segment between the start point A and the other node, the weight will be ∞, for example, points B, C, F, G, and H.

c) If there is a weight in the segment between the start point A and the other node, the weight will be the same as the point which started (A), for example, points D with weight 10 and E with weight 4.

2. Choose the point which has minimum weight, and write this point in the first position of the third row, which was the start (E), tXC3hen find the weight from point E to each node according to four steps, and write each weight to the node under the name node in the third row.

a) The weight from point E to start point A is free because it was chosen in the previous step as a start point.
b) If there is no direct segment between point E and another node, the weight will be ∞ , for example, point C, G, H, and L.
c) For the same point, the weight is the same in the above position, making this position with color.
d) If there is a weight in the segment between point E and the other node, we will take the sum of weight from the first start point (A) to each of the following nodes and compare it with the weight in the direction above in the above row. The resulting weight was calculated for three cases.

1. If the same or larger, the resulting weight will remain.
2. If smaller, take a new weight. For example, for point D, if the new weight is 9 smaller than 10, the resulting weight is 9E, F the new weight is 12 smaller ∞, and the resulting weight is 12E.

3. Cancel the start points, choose the node with minimum weight, node J with weight 6 (4+2), then repeat the procedure in step 2 until the last node arrives.
4. The colored positions with nodes in the last columns of the table were the results of the shortest paths from start point A to this point. For example, node N with weight 14 came from node K, node K came from node J. Node J came from node E, and node E came from node A. In other words, the shortest path from the start of node A to node N is A-E-J-K-N.

Table 1: With beginning A

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**
A	0	∞	∞	10A	4A	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
E		6E	∞	9E	4A	12E	∞	∞	18E	6E	11E	∞	∞	∞	∞	∞
J		6E	∞	7J		10J	7B	∞	12J	6E	8J	∞	∞	22J	11J	∞
B		6E	∞	7J		9B	7B	∞	12J		8J	∞	∞	22J	11J	∞
G			15G	7J		9B	7B	13G	12J		8J	10G	8G	22J	11J	∞
D			15G	7J		9B		13G	12J		8J	10G	8G	22J	11J	∞
K			15G			9B		11M	12J		8J	9K	8G	14K	11K	12K
M			15G			9B		11M	12J			9K	8G	14K	11K	12K
F			15G			9B		11M	12J			9K		14K	11K	12K
L			15G					11M	12J			9K		14K	11K	11L
H			15G					11M	12J					14K	11K	11L
O			15G						12J					14K	11K	11L
P			15G						12J					14K		11L
I			15G						12J					14K		
N			15G											14K		
C			15G													

For Tables 2, 3, the same procedure was done, but the difference in the start point B, C instead of A

Table 2: With beginning B

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**
B	∞	0	∞	∞	2B	3B	1B	∞	∞	∞	∞	∞	∞	∞	∞	∞
G	∞		9G	∞	2B	3B	1B	7G	∞	∞	6G	4G	2G	∞	∞	∞
E	6E		9G	7E	2B	3B		7G	16E	4E	6G	4G	2G	∞	∞	∞
M	6E		9G	7E		3B		5M	16E	4E	6G	4G	2G	∞	∞	7M
F	6E		9G	7E		3B		5M	16E	4E	4F	4G		∞	∞	7M
J	6E		9G	5J				5M	10J	4E	4F	4G		20J	9J	7M
K	6E		9G	5J				5M	10J		4F	4G		10K	7K	7M
L	6E		9G	5J				5M	10J			4G		10K	7K	6L
D	6E		9G	5J				5M	10J					10K	7K	6L
H	6E		9G					5M	10J					10K	7K	6L
A	6E		9G						10J					10K	7K	6L
P			9G						10J					10K	7K	6L
O			9G						10J					10K	7K	
C			9G						10J					10K		
I									10J					10K		
N														10K		

Table 3: With beginning C

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
C	∞	∞	0	∞	∞	∞	8C	4C	∞	∞	∞	∞	∞	∞	∞	∞
H	∞	∞		∞	∞	∞	8C	4C	∞	∞	∞	13H	7H	∞	∞	∞
M	∞	∞		∞	∞	∞	8C		∞	∞	∞	9M	7H	∞	∞	12M
G	∞	9G		∞	∞	13G	8C		∞	∞	13G	9M		∞	∞	12M
B	∞	9G		∞	11B	12B			∞	∞	13G	9M		∞	∞	12M
L	∞			∞	11B	12B			∞	∞	10L	9M		∞	25L	11L
K	∞			∞	11B	11K			∞	12K	10L			16K	13K	11L
E	15E			16E	11B	11K			25E	12K				16K	13K	11L
F	15E			16E		11K			25E	12K				16K	13K	11L
P	15E			16E					25E	12K				16K	13K	11L
J	15E			13J					18J	12K				16K	13K	
D	15E			13J					18J					16K	13K	
O	15E								18J					16K	13K	
A	15E								18J					16K		
N									18J					16K		
I									18J							

3.2 Single Way Case

In this case, each edge takes one direct side. Therefore, Table 4 consists of the number of rows equal to the number of nodes plus one.

1. First of all, we write the name of nodes in the first row of the table from the second position to the end of the row, then choose any point as a starting point. For example, A in this table is shown below. The second step is writing the start node A in the first position of the second row, then finding the weight from point A to each node according to three steps, and writing each weight to the node under the name node directly in the second row.
   a) The weight from start point A to the same point A is zero
   b) If there is no direct segment between the start point A and the other node, the weight will be ∞, for example, points B, C, F, G, and H.
   c) If there is a weight in the segment between the start point A and the other node, the weight will be the same as the point which started (A), for example, points D with weight 10, and E with weight 4.
2. Choose the point with minimum weight, and write this point in the first position of the third row, which was the start (E). Then, find the weight from point E to each node according to four steps, and write each weight to the node under the name node in the third row.

   a) The weight from point E to start point A is free because it was chosen in the previous step as a start point.
   b) If there is no direct segment between point E and the other node, the weight will be ∞ , for example, points B, C, G, H, and L.
   c) For the same point, the weight is the same in the above position, making this position with color.
   d) If there is a weight in the segment between point E and the other node, we will take the sum of weight from the first start point (A) to each of the following nodes and compare it with the weight in the direction above position in the above row. The resulting weight was calculated for three cases.
   1. If the same or larger, the resulting weight will remain. For example, the weight between E and D is ∞ in node D.
   2. If smaller, take the new weight, for example, points F, I, J, and K; the new weights are smaller ∞, resulting in 12,18,6,11 respectively.

3. Cancel the start points, choose the node with minimum weight, node J with weight 6 (4+2) , then repeat the procedure in step 2 until the last node arrives.
4. The colored positions with nodes in the table's last columns were the results of the shortest paths from start point A to this point. So, for example, node N with weight 14 came from node K, node K came from node J, node J came from node E, and node E came from node A. In other words, the shortest path from the start of node A to node N is A-E-J-K-N.

Table 4: With beginning A

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	0	∞	∞	10A	4A	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
E		∞	∞	10A	4A	12E	∞	∞	18E	6E	11E	∞	∞	∞	∞	∞
J		∞	∞	10A		12E	∞	∞	18E	6E	8J	∞	∞	22J	11J	∞
K		∞	∞	10A		12E	∞	∞	18E		8J	9K	∞	14K	11J	12K
L		∞	∞	10A		12E	∞	∞	18E			9K	11L	14K	11J	11L
D		∞	∞	10A		12E	∞	∞	18E				11L	14K	11J	11L
M		∞	∞			12E	∞	∞	18E				11L	14K	11J	11L
O		∞	∞			12E	∞	∞	18E					14K	11J	11L
P		∞	∞			12E	∞	∞	18E					14K		11L
F		∞	∞			12E	17F	∞	18E					14K		
N		∞	∞				17F	∞	18E					14K		
G		∞	∞				17F	23G	18E							
I		∞	∞					23G	18E							
H		∞	∞					23G								

For Tables 5, 6, the same procedure was done, but the difference in the start point B, C instead of A

Table 5: With beginning B

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
B	∞	0	∞	∞	2B	3B	1B	∞	∞	∞	∞	∞	∞	∞	∞	∞
G	∞		∞	∞	2B	3B	1B	7G	∞	∞	6G	4G	2G	∞	∞	∞
M	∞		∞	∞	2B	3B		7G	∞	∞	6G	4G	2G	∞	∞	7M
E	∞		∞	∞	2B	3B		7G	16E	4E	6G	4G		∞	∞	7M
F	∞		∞	∞		3B		7G	16E	4E	4F	4G		∞	∞	7M
J	∞		∞	∞				7G	16E	4E	4F	4G		20J	9J	7M
K	∞		∞	∞				7G	16E		4F	4G		10K	7K	7M
L	∞		∞	∞				7G	16E			4G		10K	7K	6L
P	∞		∞	∞				7G	16E					10K	7K	6L
H	∞		∞	∞				7G	16E					10K	7K	
O	∞		∞	∞					16E					10K	7K	
N	∞		∞	∞					16E					10K		
I	∞		∞	∞					16E							

Table 6: With beginning C

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
C	∞	∞	0	∞	∞	∞	8C	4C	∞	∞	∞	∞	∞	∞	∞	∞
H	∞	∞		∞	∞	∞	8C	4C	∞	∞	∞	13H	7H	∞	∞	∞
M	∞	∞		∞	∞	∞	8C		∞	∞	∞	13H	7H	∞	∞	12M
G	∞	∞		∞	∞	∞	8C		∞	∞	13G	11G		∞	∞	12M
L	∞	∞		∞	∞	∞			∞	∞	13G	11G		∞	27L	12M
P	∞	∞		∞	∞	∞			∞	∞	13G			∞	27L	12M
K	∞	∞		∞	∞	∞			∞	∞	13G			19K	16K	
O	∞	∞		∞	∞	∞			∞	∞				19K	16K	
N	∞	∞		∞	∞	∞			∞	∞				19K		

4. Results and Discussion

This work is distinguished from the works close to it referred to in the references by several advantages, including Reliance on the dynamic instead of the static. In this work, sixteen points are distributed as follows (three input nodes, three output nodes, and ten intermediate nodes). In addition, two different map models were adopted. The first allows the robot to pass in both directions (two-direction segment), while the second model allows the robot to pass in one direction only (one-direction segment).

The system was implemented using the Matlab program and the two models mentioned above. The shortest path was determined in all the experiments and indicated in red in all the figures.

4.1 Static Case

The weights adopted in this case are fixed and do not change during the implementation of the proposed algorithm.

4.1.1 Two direction segment

The weight adopted for each segment is the same in both directions. The meaning is that each segment is treated as two segments in opposite directions and of the same weight in Figure 6. The robot starts from node A. The robot starts with two possible paths, one to the D node and the other to node E. Thus, the choices continue after moving from one node to another, and the calculations are done according to the algorithm. The shortest path indicated in red was found.

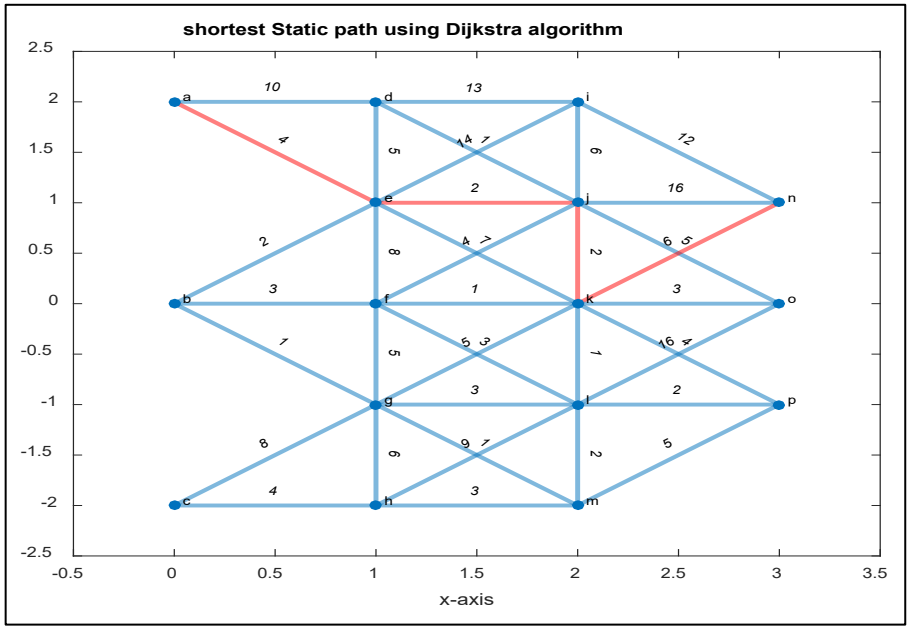

Figure 6: Static shortest path with start node A and end node N

In Figure 7. The robot starts from node B. The robot starts with three possible paths, one to the E node and the other to the nodes F and G. Thus, the choices continue after moving from one node to another, and the calculations are done according to the algorithm, and the shortest path indicated in red was found.

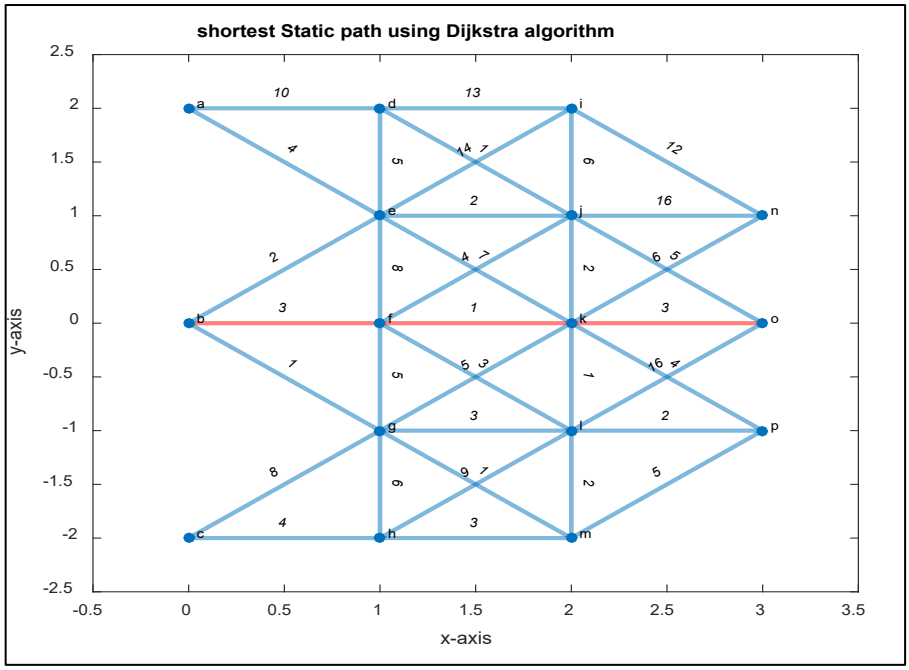

Figure 7: Static shortest path with start node B and end node O

In Figure 8. The robot starts from node C. The robot starts with two possible paths, one to the G node and the other to node H. Thus, the choices continue after moving from one node to another, and the calculations are done according to the algorithm, and the shortest path indicated in red was found.

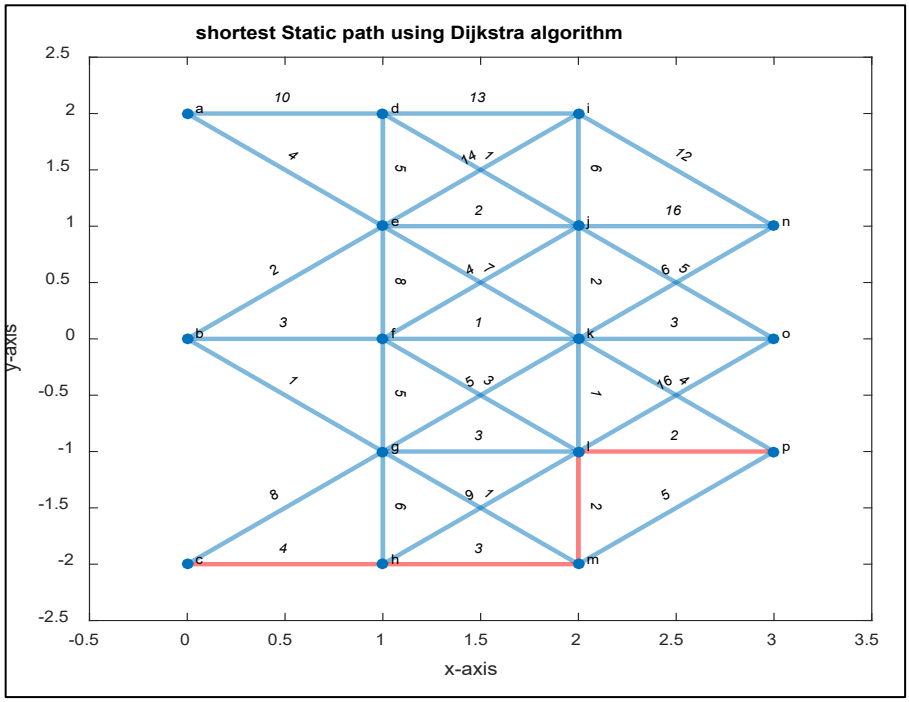

Figure 8: Static shortest path with start node C and end node P

4.1.2 One direction segment

At this stage, the results for the second case were obtained. This situation allows the robot to move in only one direction from each map segment.

Figures 9 and 10. Indicates that the results, in this case, are identical to the previous case (two-way). In contrast, the difference is shown in Figure 11 only. This is because the section between node i and node m allows only one direction. Thus, the algorithm found an alternative path for the previous case.

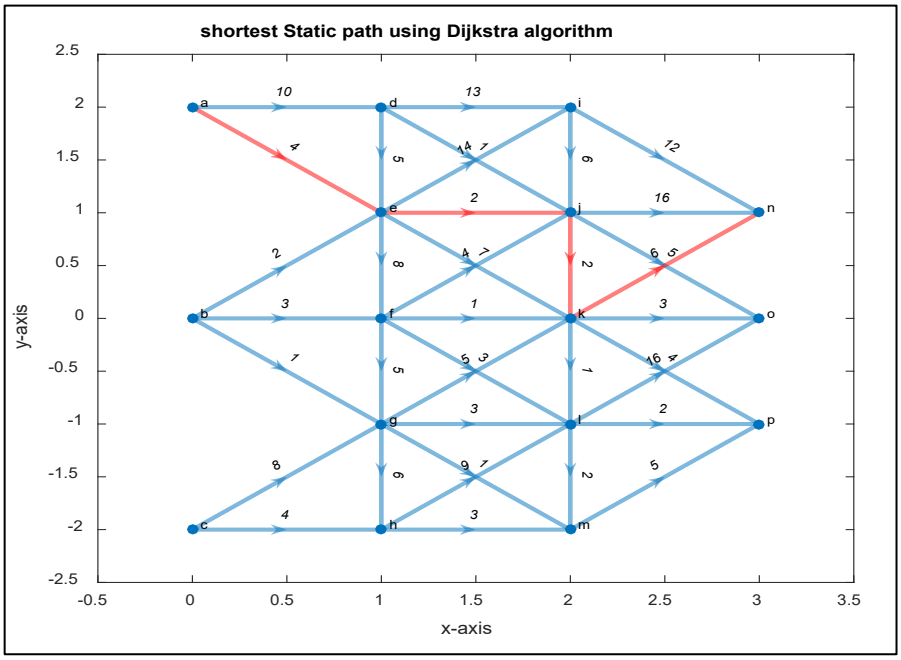

Figure 9: Static shortest path with start node A and end node N

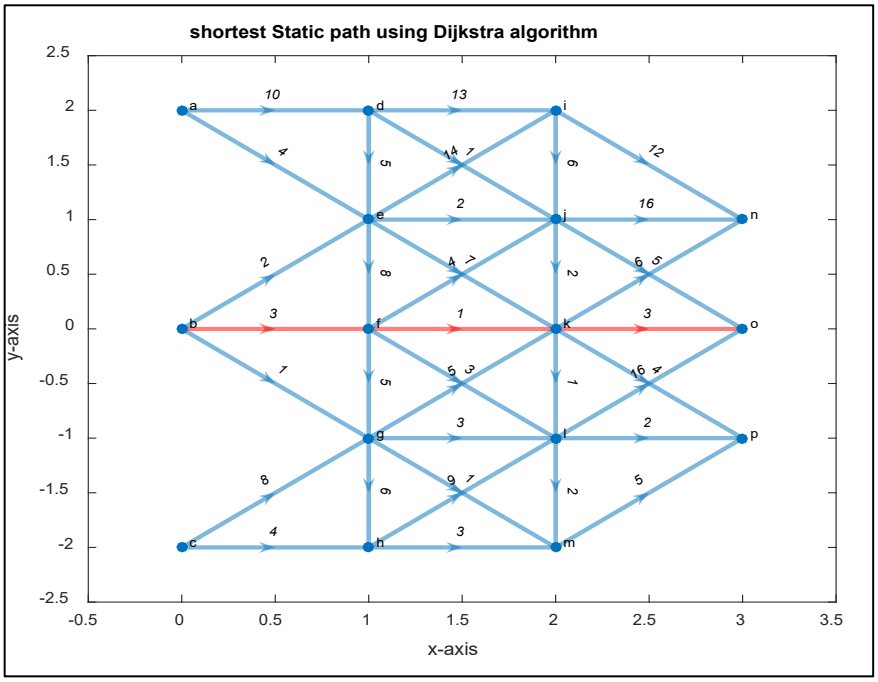

Figure 10: Static shortest path with start node B and end node O

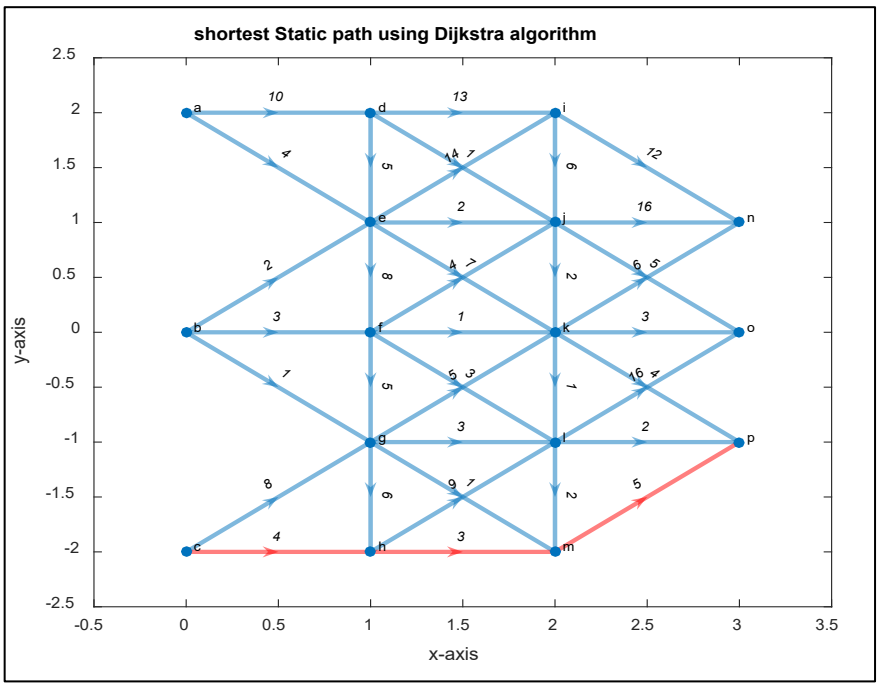

Figure 11: Static shortest path with start node C and end node P

4.2 Dynamic Case

The main principle adopted in this case is that the algorithm reapplies all the static state calculations when the robot reaches any node. Re-calculation is based on the following assumptions:

- First: the initial node is the node on which the robot stands.
- Second: The new weights of the nearby paths on the robot are entered periodically at each turn.
- Third: The goal node is fixed and does not change.
- Fourth: In the event that the new weights of the segments close to the robot are not entered, the previous weights will be approved

4.2.1 Two direction segment

Figures 12, 13, and 14. This method was adopted, and results were obtained. Note in the command window on Matlab. The new weights are entered as a vector when the robot reaches any node. The figures show that the start node and the target node are obvious in the shortest path.

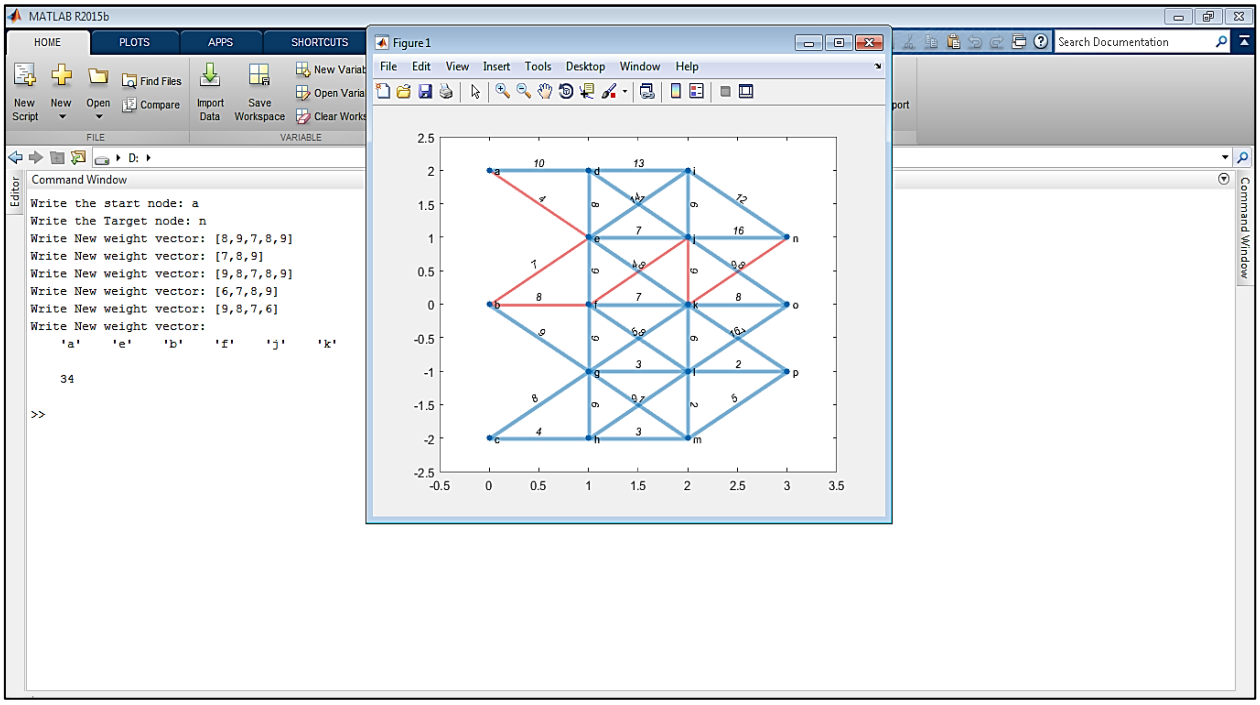

Figure 12: Dynamic shortest path with start node A and end node N

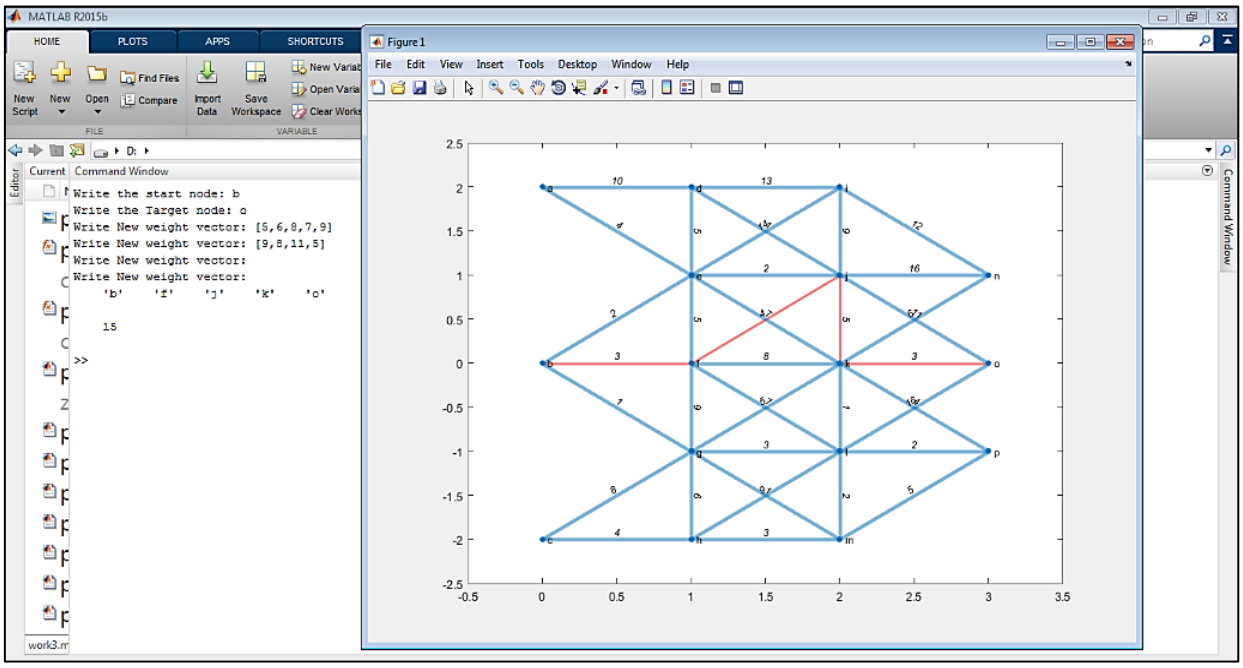

Figure 13: Dynamic shortest path with start node B and end node O

4.3 One Direction Segment

Figures 15, 16, and 17. This method was adopted, and results were obtained. Note in the command window on Matlab. The new weights are entered as a vector when the robot reaches any node. The start node and the target node are obvious in the shortest path in mentioned Figures. The results were as shown in Table 7 that appear the difference between the static and dynamic paths

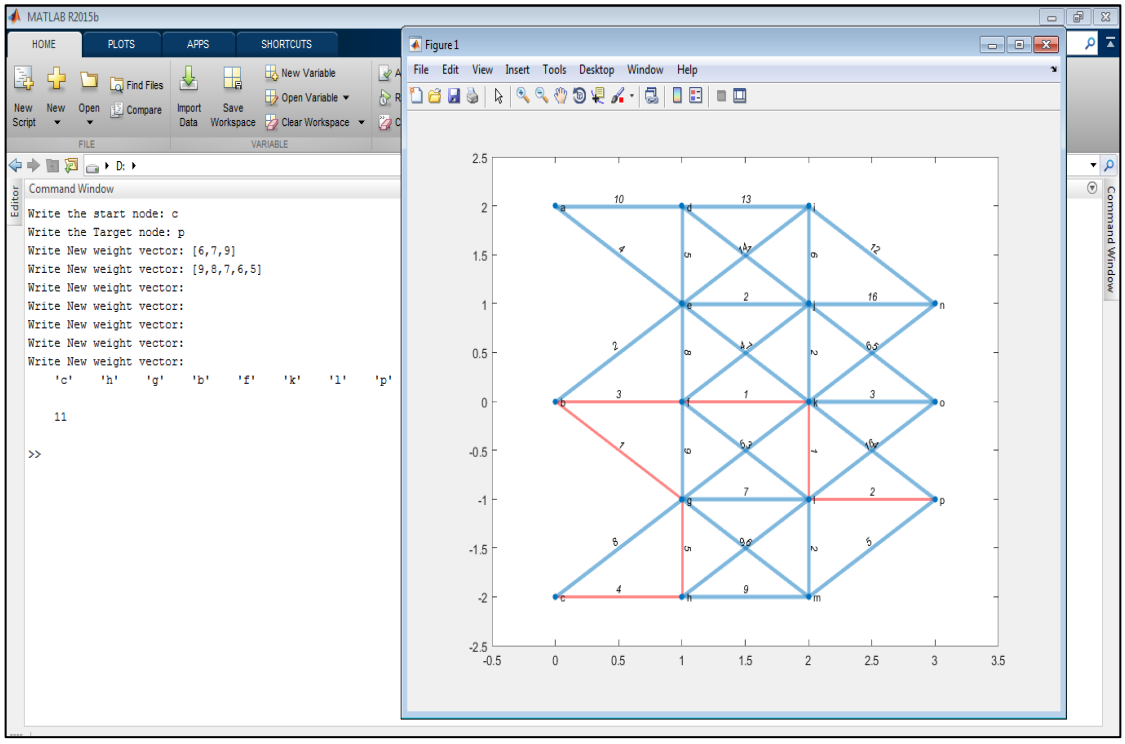

Figure 14: Dynamic shortest path with start node C and end node P

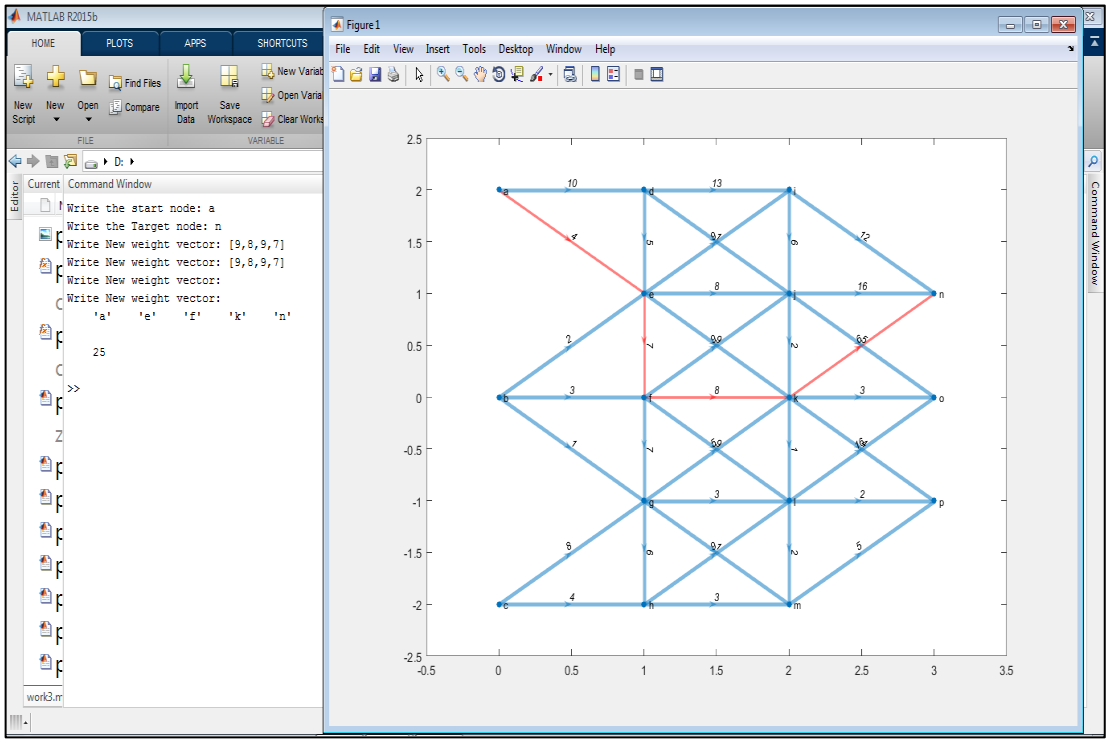

Figure 15: Dynamic shortest path with start node A and end node N

Table 7: Results

Start node	Target node	Static path two way	Static path single way	Dynamic path two way	Dynamic path single way
a	n	a-e-j-k-n	a-e-j-k-n	a-e-b-f-j-k-n	a-e-f-k-n
b	o	b-f-k-o	b-f-k-o	b-f-j-k-o	b-f-j-o
c	p	c-h-m-l-p	c-h-m -p	c-h-g-b-f-k-l-p	c-h-l-m-p

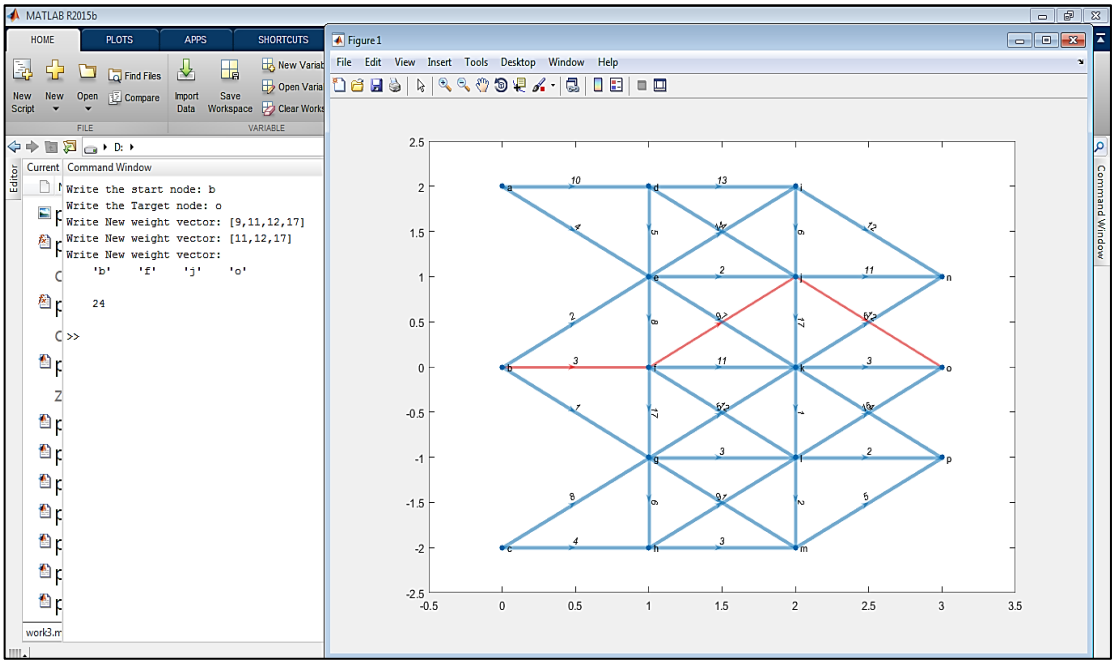

Figure 16: Dynamic shortest path with start node B and end node O

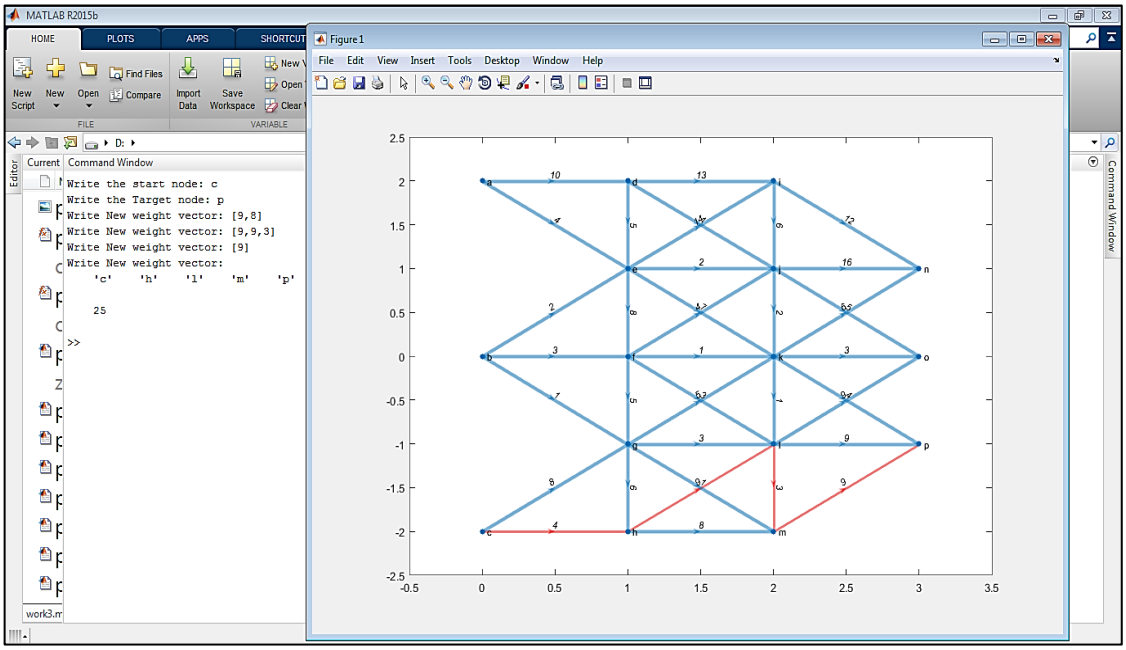

Figure 17: Dynamic shortest path with start node C and end node P

5. Conclusion

The results in the Figures have proven that the dynamic method is more practical and nearest to reality because it takes the dynamic obstacles and shortest path at each node, and the two side way method has more choices than single way (directed-way) method. It is impossible to know the dynamic shortest path using tables. On a single side was impossible to make a path between some points, for example, the starts nodes A, B, and C. In Figure 5 with segments that have a single side (directional Figure), not all nodes can be start points, such as target nodes N, O, and P, but in the figure of two side segments can make any node such as a start point or target point.

Author contribution

All authors contributed equally to this work.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability statement

The data that support the findings of this study are available on request from the corresponding author.

Conflicts of interest

The authors declare that there is no conflict of interest.

References

[1] G. Gallo, Reoptimization procedures in shortest path problem. Springer, 3 (1980) 3–13. https://doi.org/10.1007/BF02092136

[2] B. Xiao, J. Cao, Z. Shao, E.H.-M. Sha, An Efficient Algorithm for Dynamic Shortest Path Tree Update in Network Routing. in J. Commun. Networks, 9 (2007) 499–510. https://doi.org/10.1109/JCN.2007.6182886

[3] Y. Sharma, S.C. Saini, M. Bhandhari, Comparison of Dijkstra's Shortest Path Algorithm with Genetic Algorithm for Static and Dynamic Routing Network. Int. J. Electr. Comput. Sci. Eng.,1 (2012) 416-425.

[4] U.A. Okengwu, E.O. Nwachukwu, Modified Dijkstra Algorithm for Determining Multiple Source Shortest Path of Hospital Location in Rivers State (Nigeria), Glob. J. Eng. Sci. Res. Manag., 1 (2014) 46-50.

[5] H. Abu-Ryash, Dr.A. Tamimi , Comparison Studies for Different Shortest Path Algorithms, Int. J .Comput. Technol., 14 (2015) 5979-5986.

[6] K. A. Fariza. Abu Samah, B. Hussin, A.H. Basari, Modification of Dijkstra's Algorithm for Safest and Shortest Path during Emergency Evacuation . Faculty of Information Technology and Communication Universiti ,Teknikal Malaysia Melaka (UTeM) 76100 Durian Tunggal, Appl. Math. Sci., 9 (2015) 1531 – 1541. http://dx.doi.org/10.12988/ams.2015.49681

[7] E.N. Tamatjita, A.W. Mahastama, Shortest Path with Dynamic Weight Implementation Using Dijkstra`s Algorithm. J. Binus . Sci., 7 (2016) 161-171. https://doi.org/10.21512/comtech.v7i3.2534

[8] M. Malathi, Optimal Path Planning for Mobile Robots Using Particle Swarm Optimization and Dijkstra Algorithm with Performance Comparison, Middle East J. Sci. Res., 24 (2016) 312-320. https://doi.org/10.5829/idosi.mejsr.2016.24.S1.65

[9] H. Zhang, M. Li, Rapid path planning algorithm for mobile robot in dynamic environment, Adv. Mech. Eng.,9 (2017) 1-12. https://doi.dox.org/10.1177/1687814017747400

[10] S., D. Garg, Dynamizing Dijkstra, A Solution to Dynamic Shortest Path Problem through Retroactive Priority Queue.J. King Saud Univ. - Comput. Inf. Sci., 33(2018)1-21. https://doi.org/10.1016/j.jksuci.2018.03.003

[11] J.B. Singh, R.C. Tripathi, Investigation of Bellman–Ford Algorithm, Dijkstra's Algorithm for suitability of SPP. Int. J. Eng. Res., 6 (2018) 755-758.

[12] A. Alyasin, E.I. Abbas, S.D. Hasan, An Efficient Optimal Path Finding for Mobile Robot Based on Dijkstra Method, In: 4th Scientific International Conference – Najaf – IRAQ. 2019,11-14. https://doi.org/10.1109/SICN47020.2019.9019345

[13] S.W.G. AbuSalim, R. Ibrahim, M. Saringat, S. Jamel, J.A. Wahab, Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization, In: International Conference on Technology, Engineering and Sciences (ICTES), IOP Conf. Series: Mater. Sci. Eng., 917,2020,012077. https://doi.org/10.1088/1757-899X/917/1/012077

[14] Rawaa J. Abdul Kadhim, Optimum Path Finding for Mobile Robot Based on Artificial Intelligent Systems. M.Sc. Thesis, University of Technology-Iraq, Iraq, 2020.