Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 6, No. 1, November 2024, Page 265-273
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v6i1.6336

The Determination of Availability Path Planning in Natural Tourist Attractions using Dijkstra's Algorithm and Ant Colony Optimization

Heri Gustami[1], Muhammad Rizal[2,*], Riyadhul Fajri[1]

[1] Faculty of Computer Science, Medical Informatics, Almuslim University, Bireuen Aceh, Indonesia
[2] Faculty of Economics, Development Economics, Almuslim University, Bireuen Aceh, Indonesia.
Email: [1]herigustami@umuslim.ac.id, [2,*]muhammadrizal@umuslim.ac.id, [3]fajri071113@gmail.com
Correspondence Author Email: muhammadrizal@umuslim.ac.id
Submitted: 23/11/2024; Accepted: 30/11/2024; Published: 30/11/2024

Abstract−The research on determining Availability Path Planning for natural tourist attractions using Dijkstra and Ant Colony algorithms aims to identify the most efficient routes in terms of time, distance, and environmental conditions. This is achieved by combining the Dijkstra and Ant Colony Optimization algorithms, each with its respective advantages and disadvantages, to create a system for finding the shortest, fastest, and safest paths to natural tourist destinations in Bireuen Regency. The methodology employed in this study is the waterfall model, which encompasses stages from analysis to application development, integrating both Dijkstra and Ant Colony Optimization algorithms. The research findings reveal that the shortest route to the Ceuraceu Waterfall destination can be accessed via the Samagadeng Village road in Pandrah District, while the longest route can be accessed through the city center of Bireuen Regency. Furthermore, for the Krueng Simpo River Bathing Tourism, the shortest route is via the Gayo Bireuen road, while the longest route is through the KKA Aceh Utara road. Lastly, the shortest route to the Kuala Jangka Beach tourist attraction is via the Jangka road from the Matangglumpang Dua area, while the longest route can be accessed through the Kuala Bireuen road.

Keywords: Path Planning; Shortest Path; Natural Tourist Attractions; Efficiency; Information Technology

1. INTRODUCTION

Law of the Republic of Indonesia Number 10 of 2009 on Tourism [1]. Tourism refers to all activities related to tourism that are multidimensional and multidisciplinary, arising as a manifestation of the needs of individuals and countries, as well as the interaction between tourists and local communities, as well as among tourists themselves [2]. The search for the fastest route or Shortest Path Planning has become a significant challenge in the development of intelligent industrial machines in recent years. However, the search for the fastest route is closely related to the environmental factors of the paths taken by an object, such as smart vehicles, robots, and others [3]. Path planning is a common problem in science and artificial intelligence, with numerous applications in fields such as robotics and computer games [4]. In the concept of route search, the method used must be able to significantly adapt to the situation and environmental conditions, which will affect whether or not a device reaches the destination or destination node [5]. When determining a journey to a destination, it is expected to meet the calculated time, meaning that the use of intelligent properties depends on the computation of time to reach a target [6]. In its application, the fastest path planning is closely related to the control of the intelligent device [7]. There are supporting factors in the field that contribute to the success of the designed control, such as sensors that produce outputs that can be classified into a series of instructions based on the terrain traversed by the machine [8].

Dijkstra's algorithm, as a method used to find the fastest route from a designed graph schema [9], remains one of the most effective methods in its applications, such as path detection in the development of machines for obstacle avoidance, determining the best route in mapping, and many others [10]. However, a challenge remains: Dijkstra's algorithm requires distance data for each city before starting the algorithm process. Therefore, a method is needed to generate the travelable route distances, which can determine the availability of a route to find the fastest path [11]. The Ant Colony Algorithm (Ant Colony Optimization) is used to determine the availability of paths within a graph [12]. The working method of this algorithm involves checking each node in the graph randomly. Once all nodes have been traversed using this method, the process of releasing nodes from the stack one by one will be performed [13]. An interesting feature of this algorithm is that it does not require the distance of each node, as in Ant Colony, the distance between nodes is calculated after the ant has completed its journey [14]. This feature is expected to assist Dijkstra's algorithm in making decisions to find the fastest route to reach the endpoint of the graph [15].

To generate the shortest route or shortest path, time complexity is also crucial in route planning or path planning [5]. Chen's research addresses a fundamental issue related to unmanned vehicles, which is how to calculate the shortest time from one point to another and how to generate a path that can optimize the time to various specific locations [16].

The research titled "Research on Path Planning Method of Forging Handling Robot Based on Combined Strategy" states that in the artificial potential field method, the local minimum problem refers to the process where the robot avoids obstacles using the classical artificial potential field method. The robot may become stagnant or hover due to the influence of the position and shape of different obstacles [17]. Further, the study

Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 6, No. 1, November 2024, Page 265-273
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v6i1.6336

titled "Robot Path Planning with Avoiding Obstacles in Known Environment Using Free Segments and Turning Point Algorithm" designs an algorithm to solve path planning problems for robots with static obstacles [18]. In this study, the developed algorithm is characterized by reactive behavior to obtain obstacle-free paths and smooth trajectories. Research conducted by Devanta found that Dijkstra's Algorithm performs a blind search, which results in significant time consumption [19]. Research by Szczepanski applies the method where the path is pre-determined [20], and research by Xu, related to path planning algorithms, develops a method using the Fusion PRM and P-Bi-RRT algorithms, by combining both algorithms [21].

The research developed by the author proposes a method that generates options within a graph, specifically available paths or availability paths, and identifies the fastest route using Dijkstra's Algorithm. In this study, a combination of these two methods will yield the fastest and shortest path to the destination node after the first fastest route encounters an obstacle [22]. In this research, pre-path planning will be implemented as a random node detector after the node arrangement within the graph is lost. The Ant Colony Algorithm, when used for searching the fastest route to a destination, still faces challenges in planning the fastest path within a graph, as it requires more time to complete the process, thus limiting its effectiveness. Therefore, a combination of algorithms is needed to speed up the process and make it more efficient [23]. The combination of the Ant Colony Algorithm and Dijkstra's Algorithm will function as both a searcher and a sorter of available paths or routes within the graph to a destination, allowing the intelligent system to determine the path to be taken more quickly with the availability of paths in advance [24].

2. RESEARCH METHODOLOGY

2.1 Research Stages

This study uses the Ant Colony Optimization algorithm, which is employed to obtain correlations between existing nodes. This method generates all available routes or availability path planning [25]. In this research, it is expected that this method will assist the Dijkstra algorithm in determining the shortest path planning from all available routes, which will be processed from the results of weighting. The stages of the research are depicted in the figure below:

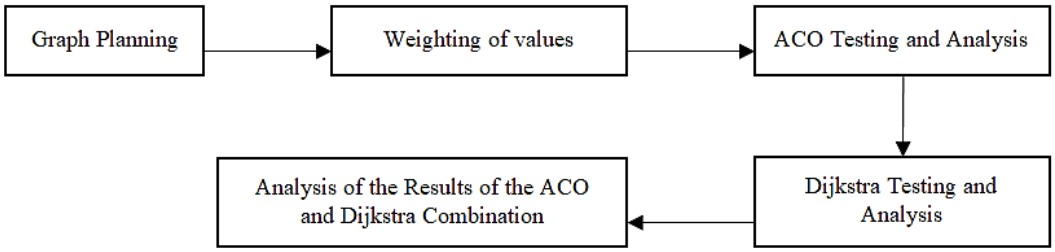

Figure 1. Research Flow

In Figure 1 above, it is explained that to obtain the shortest path, available routes (Availability Path Planning) are required, which are determined using the Ant Colony Optimization (ACO) method. The following are the stages:

a. Weighting of edge values
b. Testing and Analysis using ACO
c. Testing and Analysis of the Dijkstra Algorithm based on edge value weighting
d. Testing and Analysis of the combination of the Dijkstra Algorithm and ACO

To determine whether this research can produce effective results in the field of shortest path searching, an analysis of the methods is conducted, which will lead to conclusions that must first generate the available routes. This research can be divided into several stages, starting with the assignment of values or weighting to the graph. Essentially, this weighting is used in the application of the Dijkstra Algorithm in this study; however, in the implementation of the ACO Algorithm, the designed weighting procedure is intended as a distance counter for all available routes (Available Path) [26]. The weighting of values between edges significantly impacts the Time Complexity and Space Complexity of the designed graph.

2.2 Research Location

There are several natural tourist locations that will serve as testing sites for the fastest route, namely:

a. Ceuraceu Waterfall, Bireuen Regency, Aceh Province, Indonesia
b. Krueng Simpo River Bathing Tourist Site, Bireuen Regency, Aceh Province, Indonesia
c. Kuala Jangka Beach Tourist Site, Bireuen Regency, Aceh Province, Indonesia

Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 6, No. 1, November 2024, Page 265-273
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v6i1.6336

2.3 Research methods

2.3.1 Model Graph Plannning

The process of designing the graph involves 6 nodes or vertices that are interconnected. Graph-based path planning algorithms are widely applied because they prioritize Space Complexity and Time Complexity [19].

Table 1. Graph Representation in Matrix Form

	A	B	C	D	E	F
A	0	1	1	0	0	0
B	1	0	1	1	1	0
C	1	1	0	1	1	0
D	0	1	1	0	1	1
E	0	1	1	1	0	1
F	0	0	0	1	1	0

In the planning and design of the graph in this study, it is not based on Dynamic Path Planning because the root node and destination node have been pre-designed in such a way that, to reach the destination node, the author has already prepared the available routes [5].

2.3.2 Edge value weighting

The weighting of values on the edges in the graph in this study is as follows:

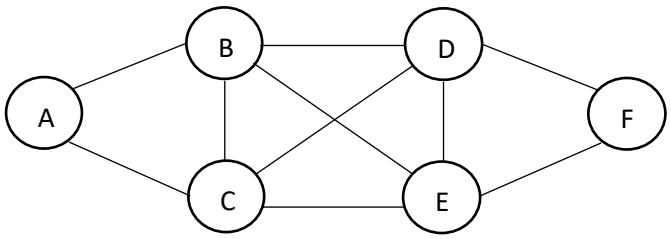

Figure 2. Edge Value Weighting on the Graph

In Figure 2 above, the edge value weighting can be seen, which generates values that will determine the fastest route, or shortest path, in this study. These values will influence the time complexity in a graph.

2.3.3 Implementation Ant Colony Optimization (ACO)

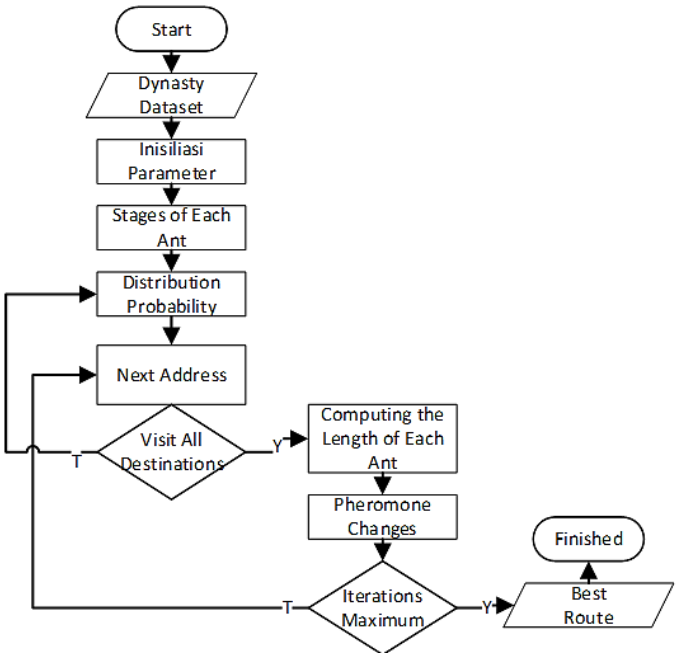

Figure 3. Ant Colony Optimization Flowchart

The determination of Availability Path Planning to identify the Shortest Path involves the process of distributing several homogeneous commodities from various sources to a set of destinations. Each request is specified with the level of the commodity required [27]. The objective of this problem is to allocate the available

Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 6, No. 1, November 2024, Page 265-273
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v6i1.6336

supply at each source to meet the demand at each destination optimally [23]. The most common objective function is to minimize the total distance traveled during the allocation process. The flowchart depicting the operation of the Ant Colony Optimization (ACO) Algorithm applied to determining availability path planning is as follows:

2.3.4 Analysis of the Dijkstra and ACO Combination

The combination of the Dijkstra Algorithm and the ACO (Ant Colony Optimization) method lies in the number of node relaxations that occur each time the fastest route or shortest path is lost from the available routes to the destination node. It can be concluded that the Ant Colony Optimization method will always check the availability of routes to the destination node by treating the root in the Dijkstra Algorithm as the root node [23]. The advantage of the Dijkstra Algorithm lies in the results of the relaxation computation in determining the shortest path in a graph [18].

The use of the backtracking feature of the ACO (Ant Colony Optimization) Algorithm, combined with the node relaxation from the Dijkstra Algorithm, is expected to produce a decision-making process that can change, leading to the second fastest route based on the results of route testing or paths already obtained from storing the first fastest route into the available variables [26]. In principle, the ACO method tests each node to check for route changes. The following presents a flowchart for the combination of these two methods [19]. The flowchart designed in Figure 4 below can be summarized as follows: if a change occurs in the first route after the computation process of the Dijkstra Algorithm, a re-testing will be conducted on the new graph, where the last node or root node is positioned at the final computation step of the Dijkstra Algorithm. The testing will determine the second fastest route based on the computation results of the ACO (Ant Colony Optimization) Algorithm [28]. The entire process of the flowchart will be repeated if the second fastest route undergoes changes due to factors within the graph in order to find an alternative fastest route that matches the destination node. In this computation, each change will have a corresponding available path with its respective weight.

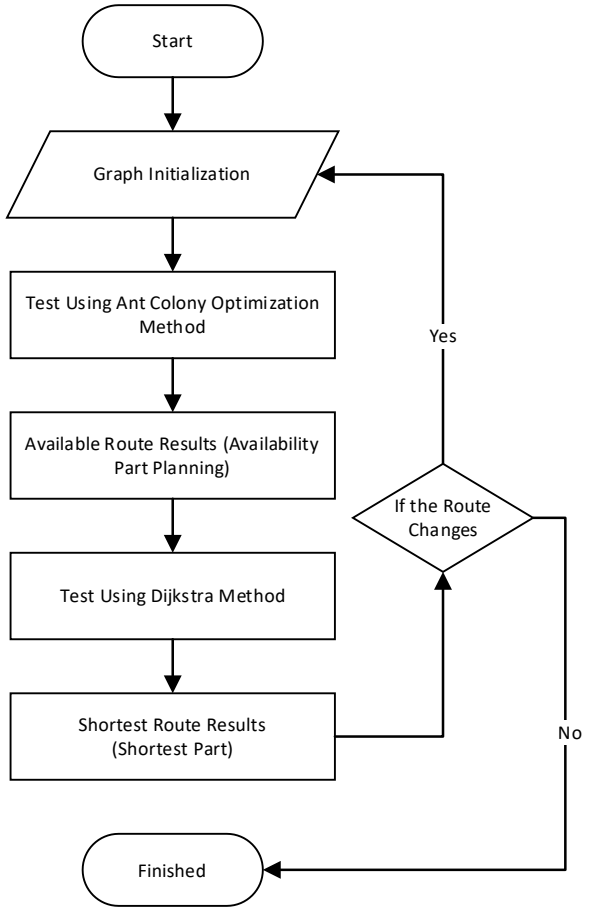

Figure 4. Flowchart of method combination

2.3.5 Research Support Tools

This study uses Python programming language version 3.8.3 and a computer with an Intel Core i5 processor and 8 GB of memory. This setup is used to facilitate the computation of the combination of algorithms employed to determine the available routes or Availability Path Planning and the fastest route or shortest path.

Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 6, No. 1, November 2024, Page 265-273
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v6i1.6336

3. RESULT AND DISCUSSION

3.1 Research Result

The route availability calculation scheme will generate values that serve as a reference for determining the fastest route to the natural tourist locations, which will be used as testing sites for the fastest route. These locations include Ceuraceu Waterfall, Bireuen Regency, Krueng Simpo River Bathing, Bireuen Regency, and Kuala Jangka Beach, Bireuen Regency, Aceh Province, Indonesia.

3.2 ACO Test *(Ant Colony Optimization)*

This research is conducted through two testing processes: the first test aims to determine the availability of paths using a graph with Ant Colony Optimization, where the testing will be performed at each node. The expected outcome of this first test is to produce a reference algorithm that can serve as a foundation for the Dijkstra algorithm to select the shortest route to reach the predetermined destination. The figure below displays the graph program code designed for Ant Colony Optimization. The Ant Colony Optimization test, based on the graph, results in 20 possible paths to reach node "F." However, this test does not yet determine which path will be selected as the fastest route by the Dijkstra algorithm. The table below presents the available routes leading to node "F."

Table 2. Graph Route Testing with Ant Colony Optimization

No	Rute
1	'A', 'B', 'E', 'F'
2	'A', 'B', 'E', 'D', 'F'
3	'A', 'B', 'E', 'C', 'D', 'F'
4	'A', 'B', 'D', 'F'
5	'A', 'B', 'D', 'E', 'F'
6	'A', 'B', 'D', 'C', 'E', 'F'
7	'A', 'B', 'C', 'E', 'F'
8	'A', 'B', 'C', 'E', 'D', 'F'
9	'A', 'B', 'C', 'D', 'F'
10	'A', 'B', 'C', 'D', 'E', 'F'
11	'A', 'C', 'E', 'F'
12	'A', 'C', 'E', 'D', 'F'
13	'A', 'C', 'E', 'B', 'D', 'F'
14	'A', 'C', 'D', 'F'
15	'A', 'C', 'D', 'E', 'F'
16	'A', 'C', 'D', 'B', 'E', 'F'
17	'A', 'C', 'B', 'E', 'F'
18	'A', 'C', 'B', 'E', 'D', 'F'
19	'A', 'C', 'B', 'D', 'F'
20	'A', 'C', 'B', 'D', 'E', 'F'

In calculating the Time Complexity of the Ant Colony Optimization syntax above, it heavily depends on the computational power of the computer used to execute the program. In the first test, the best-case result of the running program is $O(n^3)$ due to the dominant Quadratic Time Complexity in this method. This type of time complexity occurs every time an iteration is performed over the data set. Quadratic Time Complexity arises when the designed function is n^2, where n represents the total number of inputs and functions. This time complexity occurs because a linear function is executed within another linear function (n*n). In this study, the time complexity of the algorithm used is represented in the form of a graph, which shows the more dominant direction of the Quadratic Time Complexity derived from the results of the first test method. The calculations from the operations performed on the elements are as follows:

$$T_{(n)} = 1 + 1 + (n\ x\ 1\ x\ 1\ x\ 1) + (n\ x\ 1\ x\ n\ x\ 1\ xn\ x\ 1\ ) + 1 + 1$$

$$T_{(n)} = 2 + \ n + n^3 + 2$$

$$T_{(n)} = n + n^3 = O(n^3)$$

It can be concluded that the Time Complexity is $O(n^3)$ and the Space Complexity is $O(n^3)$ because the program contains.

Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 6, No. 1, November 2024, Page 265-273
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v6i1.6336

3.3 Dijkstra Algorithm Testing

In this study, two tests were conducted on the Dijkstra Algorithm. The first test was to find the shortest route using the implemented method. The second test involved measuring time complexity (Time Complexity) and space complexity (Space Complexity), which indicates how much memory space is required when this algorithm operates. In this test, the node relaxation of Dijkstra became the main focus in determining the shortest route to each node within the graph. Below is a re-representation of the graph in Figure 6, which is included within the zonation. This step is performed to facilitate the tracing of node relaxation within the graph.

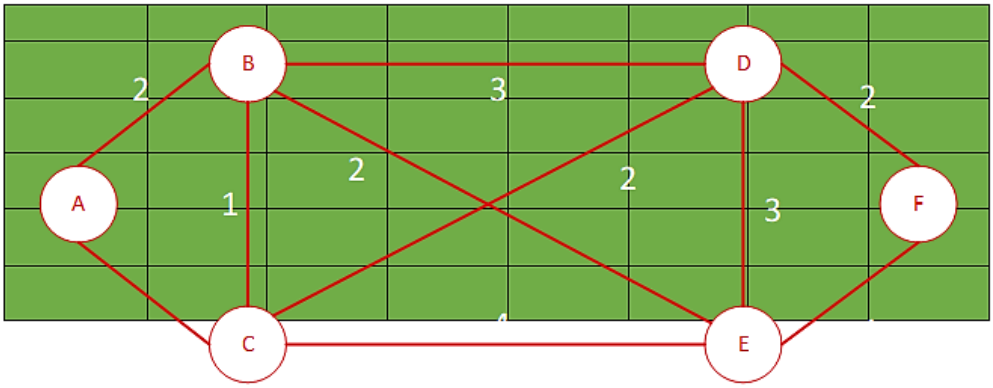

Figure 5. Node Zonation Scheme in the Graph.

In this graph, the root node is located at point A. The relaxation in the conventional Dijkstra algorithm involves calculating each node to obtain the lowest weight. The designed zonation is expected to illustrate the situation of node relaxation taking place between nodes.

3.3.1 Shortest Route Testing

In the shortest route testing, it serves as a reference for the fastest route from the root node to the destination node. Below is the program code for finding the fastest route.

Table 3. Results of Dijkstra's Algorithm Testing

Location	Route	Distance (Km)
House	Matangglumpangdua	1
	Bireuen	11
	Peudada	16
	Pandrah	32
	Jinieb	38
	Air Terjun Ceuraceuk	46
	Blang Keutumba	18
	Alue Rambong	26
	Krueng Simpo	28
	Jangka	10
The City of Matanglumpangdua	Bireuen	10
	Peudada	24
	Pandrah	39
	Jinieb	36
	Air Terjun Ceuraceuk	47
	Blang Keutumba	16
	Alue Rambong	24
	Krueng Simpo	26
	Jangka	9

3.3.2 Testing and Analysis of Time Complexity and Space Complexity

The testing and analysis of Time Complexity and Space Complexity in this study heavily depend on the graph being computed. The more nodes in a graph, the greater the impact on the complexity. The time complexity of the Dijkstra Algorithm is O (E Log V), where "V" represents the number of nodes and "E" represents the number of edges. The testing and analysis in this study, as shown in Figure 7, were conducted using the Python programming language. In this study, the author has calculated the results for the Best Case, Average Case, and Worst Case of the method used, as shown in the table below.

Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 6, No. 1, November 2024, Page 265-273
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v6i1.6336

Table 4. Best, Average, and Worst Case Time Complexity

Best Case	O(e+v log v)
Average Case	O(e+v log v)
Worst Case	O(e+v log v)

In this study, the values for the Best, Average, and Worst Case of the Dijkstra Algorithm are O(e + v log v). Starting from the initial node or starting point, the search for the fastest path in this study can be described as follows.

1. The first step has a value of = O(e) time.
2. The second step has a value of = O(1) time.
3. The third step has a value of = O(e) time.
4. The fourth step has a value of = O(e + v) log v) time.

Therefore, for Step 5 and beyond, the value will remain O(e + v) log v) time, where "e" is the number of edges in the graph, and "v" is the number of vertices or nodes in the graph. The number of edges and vertices or nodes is directly proportional in the computation of this method, meaning that regardless of the number of nodes and vertices, the time complexity in this method remains the same.

3.4 Pre Path Planning

In this study, a method is applied to identify nodes that do not collide with obstacles. This process will assist in the execution of the Ant Colony Optimization (ACO) algorithm in finding available routes within the graph. x_{last} is the last node visited from the execution of Dijkstra's Algorithm, and $xnode$ is the array of nodes distributed within the graph to provide the coordinates of obstacles in order to avoid obstacles located within the graph.

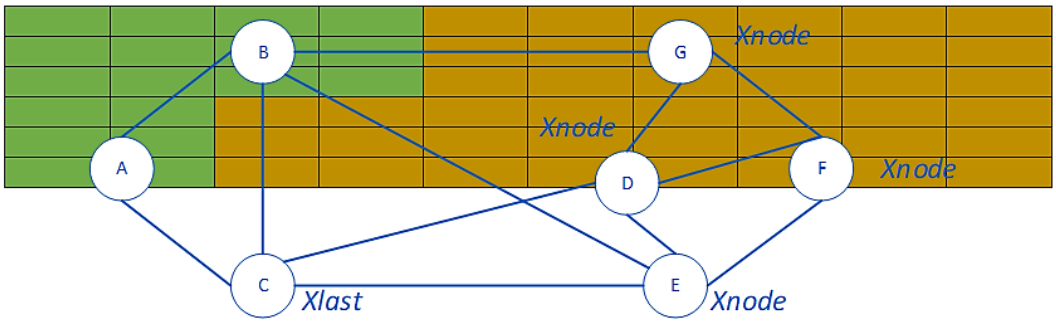

Figure 6. Node Array Transition within the Graph

As seen in Figure 9 above, the transition of nodes within the graph shows that the X_last position is at node "C," indicating that the new root node is located at node "C."

Table 5. Route Search Results from Pre-Path Planning

No	Rute
1	C ➡ D ➡ G ➡ F
2	C ➡ D ➡ E ➡ F
3	C ➡ E ➡ F

Based on the results of the Pre-Path Planning test conducted on route search within the graph, the outcomes are improved, as it produces fewer and shorter available routes. This condition facilitates the path to avoid collisions with obstacles within the graph. However, the time complexity and space complexity in Pre-Path Planning computation is O((e + v) log v) time.

3.5 Testing and Analysis of the Combination of ACO and Dijkstra's Algorithm

In this study, a combination of ACO and Dijkstra's Algorithm is implemented to detect the second fastest route. In accordance with the initial analysis of the combination in Chapter 3, the use of backtracking is found in the node relaxation process within Dijkstra's Algorithm. In this test, the computation of node relaxation will incorporate additional variables from the ACO (Ant Colony Optimization) method. The table below represents the combination of these two methods, starting with the pseudocode of the employed method.

Table 6. Evaluation of the Variables Used According to the Pseudocode

Ant Colony Optimization	Algoritma Dijkstra
Starting from the root node	Set root node as the starting node or starting point
Insert root node into stack.	Assigning a value of “0” to the root node

Journal of Computer System and Informatics (JoSYC)
ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 6, No. 1, November 2024, Page 265-273
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v6i1.6336

Visit the next node that is directly related to the root node.	Mark each node as a "not yet passed" node.
After getting a new node, the root node position is at the new node.	Perform the relaxation process to be able to calculate the distance between nodes that are connected to the root node.
Reinsert the new root node into the queue or stack	Perform distance storage between nodes, and select the next node that has the smallest distance load among all connected nodes.
Remove the node from the queue and mark it as visited.	Mark the selected node as the root node.
Repeat process number 6 until the queue is empty again.	Perform the second and subsequent relaxations on the remaining nodes in the Graph. Nodes that have been passed are not checked again, so the last stored distance value is the minimum value of each movement towards the node. In the matrix table, nodes that have the same side or edge load will be lowered.

Table 6 above is presented to facilitate the observation of the processes within both methods that can be combined. The first equation derived from these two methods is to designate the initial node as the root node of a graph, where at this position, the ACO (Ant Colony Optimization) method will insert the root node into the stack or queue. The use of the stack in this method is useful during reverse tracking when encountering deadlocks in the graph. By utilizing this ACO feature, Dijkstra's Algorithm can use the node at the top of the stack as the root node. This utilization will lighten the computational load of Dijkstra's Algorithm in tracking all the shortest paths within the graph. In this study, the author modified Dijkstra's Algorithm in the node relaxation process. Below is an image of a program snippet that has been modified to implement the reverse feature of ACO (Ant Colony Optimization).

Based on the results obtained with this combination, the ACO (Ant Colony Optimization) process will repeat its task when Dijkstra's Algorithm loses the next node during the node relaxation process.

4. CONCLUSION

Based on the research conducted on the Determination of Availability Path Planning for Natural Tourist Objects Using Dijkstra's Algorithm and Ant Colony, it was found that Dijkstra's Algorithm performs a blind search, which results in significant time consumption. Furthermore, Pre-path Planning assists the ACO (Ant Colony Optimization) algorithm in analyzing the available paths. However, when obstacles are dynamic, Pre-path Planning must re-identify the graph. Pre-path Planning will result in fewer available route options, and the travel distance to the target destination is minimized, compared to having to return to the initial node to identify all nodes within the graph. The Time Complexity and Space Complexity of the combination with the implementation of Pre-path Planning is O (e + v) log v), as the calculation of available paths in a graph still depends on the number of edges and vertices within the graph. In the testing of the combination of Dijkstra's Algorithm and ACO (Ant Colony Optimization), the reverse feature of ACO will activate once a node is missing during the traversal of Dijkstra's Algorithm. However, a drawback is that the ACO algorithm will revisit all the nodes, resulting in a longer process to reach the target destination node. The conclusion of this study is that the shortest route to the Ceuraceu Waterfall destination can be accessed via the Samagadeng Village road in Pandrah District, while the longest route can be accessed through the city center of Bireuen Regency. Furthermore, for the Krueng Simpo River Bathing Tourism, the shortest route is via the Gayo Bireuen road, while the longest route is through the KKA Aceh Utara road. Lastly, the shortest route to the Kuala Jangka Beach tourist attraction can be accessed via the Jangka road from the Matangglumpang Dua area, while the longest route can be accessed through the Kuala Bireuen road.

ACKNOWLEDGMENTS

This research activity is conducted as part of the implementation of the 2024 Internal Research Grant at Universitas Almuslim. We would like to express our gratitude to Universitas Almuslim and the LPPM for their support of this research, as well as to all parties who have contributed to the completion of this study.

REFERENCES

[1] Kementerian Sekretariat Negara RI, "Undang-undang Republik Indonesia Nomor 10. Tahun 2009. Tentang kepariwisataan," Jakarta, 2009.
[2] Riani NK., "Pariwisata adalah pisau bermata 2," *J Inov Penelit*, vol. 2, no. 5, pp. 69–74, 2021.
[3] S. C. Hidayatullah AS, Jati AN, "Realization of depth first search algorithm on line maze solver robot," *Int. Conf. Control. Electron. Renew. Energy Commun.*, pp. 247–51, 2022.
[4] L. Wenzheng, L. Junjun, and Y. Shunli, "An improved Dijkstra's algorithm for shortest path planning on

ISSN 2714-8912 (media online), ISSN 2714-7150 (media cetak)
Volume 6, No. 1, November 2024, Page 265-273
https://ejurnal.seminar-id.com/index.php/josyc
DOI 10.47065/josyc.v6i1.6336

2D grid maps," *Int. Conf. Electron. Inf. Emerg. Commun.*, pp. 438–41, 2021.
[5] Y. Sun, R. Gu, X. Chen, L. Xin, and L. Bai, "Efficient time-optimal path planning of AUV under the ocean currents based on graph and clustering strategy," *Ocean Eng*, 2022.
[6] C.-C. Sun, G. Jan, S.-W. Leu, K.-C. Yang, and Y.-C. Chen, "Near-Shortest Path Planning on a Quadratic Surface With O (n\log n) Time," *Sens J.*, vol. 11, no. 15, pp. 6079–80, 2022.
[7] Harahap MK and Khairina N, "Pencarian Jalur Terpendek dengan Algoritma Dijkstra," *Sink J.*, vol. 2, no. 2, pp. 18–23, 2020.
[8] A. kelik Nugroho and I. Permadi, "Implementasi Jalur Terpendek Menggunakan Ant Colony Optimization," *Din. Rekayasa*, vol. 1, no. 16, 2020, doi: 10.20884/1.dr.2020.16.1.294.
[9] N. A. M. Sabri, A. S. H. Basari, and K. A. F. A. Samah, "Dijkstra-Ant Colony Optimization Algorithm For Shortest And Safest Evacuation In High Rise Building," *J. Teknol.*, vol. 3, 2022, doi: 10.11113/jt.v79.5912.
[10] Salim Y and Kurniati N, "Penerapan Metode Dijkstra Untuk Menentukan Lokasi Dan Jarak Tempuh Terpendek Kampus UMI Makassar," *Bul Sist Inf dan Teknol Islam*, vol. 2, no. 3, pp. 197–207, 2021.
[11] Inayah AM, Resti NC, and Ilmiyah NF, "Analisa Perbandingan Algoritma Floyd-Warshall Dan Algoritma Dijkstra Untuk Penentuan Rute Terdekat," *J Ilm Mat Realis*, vol. 2, no. 4, pp. 146–55, 2023.
[12] García M, López N, and Rodríguez I, "A full process algebraic representation of Ant Colony Optimization," *Inf Sci*, 2024.
[13] Karjono K, Moedjiono M, and Kurniawan D, "Ant colony optimization," *J Ticom*, vol. 3, no. 4, 2023.
[14] Nugroho AK, Permadi I, Kurniawan YI, Hanifa A, and Nofiyati N, "Decision tree using ant colony for classification of health data," *AIP Conf. Proc.*, 2023.
[15] Fallo DY, "Pencarian Jalur Terpendek Menggunakan Algoritma Ant Colony Optimization," *J Pendidik Teknol Inf*, vol. 1, no. 1, pp. 28–32, 2020.
[16] S. Z. Liu S, Jiang H, Chen S, Ye J, He R, "Integrating Dijkstra's algorithm into deep inverse reinforcement learning for food delivery route planning," *Transp Res Part E Logist Transp Rev*, 2020.
[17] Y. X. Qing, Zheng Z, "Path-planning of automated guided vehicle based on improved Dijkstra algorithm," *Chinese Control Decis. Conf.*, pp. 7138–43, 2020.
[18] W. Y. Wang T, Wang L, Li D, Cai J, "Monte Carlo-based improved ant colony optimization for path planning of welding robot," *J King Saud Univ - Comput Inf Sci*, vol. 7, no. 35, 2023.
[19] Cui J, Wu L, Huang X, Xu D, Liu C, and Xiao W, "Multi-strategy adaptable ant colony optimization algorithm and its application in robot path planning," *Knowledge-Based Syst*, 2024.
[20] Pasandi L, Hooshmand M, and Rahbar M, "Modified Algorithm integrated with ant colony optimization for multi-objective route-finding; case study: Yazd," *Appl Soft Comput*, 2021.
[21] Xu D and Tian Y, "A comprehensive survey of clustering algorithms," *Ann data Sci*, vol. 2, pp. 165–93, 2020.
[22] Cevizci B, "Calculating the Shortest Path Using Dijkstra's Algorithm," *J. Inq Based Act*, vol. 2, no. 8, pp. 70–85, 2022.
[23] Tyas YS and Prijodiprodjo W, "Aplikasi Pencarian Rute Terbaik dengan Metode Ant Colony Optimazation (ACO).," *Indones. J Comput Cybern Syst*, vol. 1, pp. 55–64, 2023.
[24] Maryani R, "Sistem Pendukung Keputusan Cerdas Menggunakan Metode Ant Colont Optimization (ACO) untuk Pencarian Jalur Optimum Rantai Pasok Bioenergi Berbasis Kelapa Sawit," *J Inf. Ekon Bisnis*, 2022.
[25] Karimi M, Kolahdouz-Rahimi S, and Troya J, "Ant-colony optimization for automating test model generation in model transformation testing," *J Syst Softw*, 2024.
[26] Peranginangin J, Purba R, and Halim A, "Analisis Perbandingan Algoritma ACO-TS dan ACO-SMARTER Dalam Menyelesaikan Traveling Salesman Problem," *J media Inf. budidarma*, vol. 4, no. 5, pp. 1698–705, 2021.
[27] L. F. Sang H, You Y, Sun X, Zhou Y, "The hybrid path planning algorithm based on improved and artificial potential field for unmanned surface vehicle formations," *Ocean Eng*, 2021.
[28] Kyriakakis NA, Marinaki M, and Marinakis Y, "A hybrid ant colony optimization-variable neighborhood descent approach for the cumulative capacitated vehicle routing problem," *Comput Oper Res*, 2021.