Journal of Theoretical and Applied Information Technology
15th October 2021. Vol.99. No 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMAL PATH PLANNING FOR WAITER ROBOT USING MODIFIED ANT COLONY OPTIMIZATION

[1]NUKMAN HABIB, [2]DJOKO PURWANTO, [3]ADI SOEPRIJANTO

[1,2,3]*Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, East Java, Indonesia*
Email: [1]habib13@mhs.ee.its.ac.id, [2]djoko@ee.its.ac.id, [3]adisup@ee.its.ac.id

ABSTRACT

The waiter robot is utilized to give a quick service for customer's satisfaction in the restaurant. In performing its task, the waiter robot facing problems such as many obstacles consist of tables, chairs, or customers along the path in the workspace. Therefore, it has capable of path planning before moving to the goal. A traditional algorithm such as A-star and Dijkstra algorithm can solve path planning for the environment like this easily, they produce the shortest path but not safely motion. Modified Ant Colony Optimization (M-ACO) algorithm is presented in this paper used to get optimal path planning for waiter robot in the restaurant. We evaluate the M-ACO algorithm for the waiter robot by optimizing some parameters to observe the performance of this approach in terms safely motion and path length. At the same time, we also examine A-star and Dijkstra algorithms in the same working environment, so we can compare the performance of each approach. The experiment results for case 1 show that M-ACO with global transfer factor (P0) value of 0.9 gives path length 79.5264 cm without crashing any obstacles while A-star and Dijkstra give path length 73.907 cm with crashing any obstacles, and another experiment result for case 2 show that M-ACO with P0 value 0.9 give path length 99.745 cm without crashing any obstacles while Astar and Dijkstra give path length 84.5617 cm with crashing any obstacles.

Keywords: *Waiter Robot, Path Planning, Modified Ant Colony Optimization*

1. INTRODUCTION

The service robot has been widely used to facilitate man activity in human life. The utilization of service robots is classified by the International Federation of Robotics (IFR) Statistical Department into two classes of application i.e. Personal/ Domestic use and Professional use [1]. The example of a service robot for Personal/ Domestic use is vacuuming, floor cleaning, window cleaning, pool cleaning, education, and research, etc. While for Professional use such as agriculture, mining systems, professional cleaning, security application, hotel, and restaurant, etc [2,3].

The waiter robot is one of the service robots for professional use, its task to assist bring food and beverage ordered by a customer such as a human waiter in the restaurant. Waiter robot has been utilized excessively for the restaurants in many Asian countries, such as Japan, China, Korea, Singapore, and Thailand.

The waiter robot was designed to deliver some ordered food or beverages from a source point to a certain customer. To navigate freely, a waiter robot must be integrated with an autonomous mobile robot, which is the ability of a mobile robot to find its path to the target without human intervention [3]. To perform this task, initially, the mobile robot has to plan the path in the restaurant before moving, such task is named "intelligent path planning" term. When a mobile robot does path planning, it needs to discover a suitable path without collides obstacles from starting point to a goal point.

In the last few decades, many researchers have been interested in mobile robot path planning (MRPP). They try to solve some problems of MRPP from a source point to the destination by many path planning approaches, such categorized as classical and heuristic approaches.

The classical approach is also called the traditional technique in MRPP, such as graph search, A star, Dijkstra algorithm [4]. Whereas, the heuristic approaches in MRPP are also called modern techniques, however, the heuristic/ modern approach could be proven are more efficient in MRPP if it is compared with some classical/ traditional approaches. Some of the modern techniques that are inspired by biological and have

been applied in the MRPP problems are Ant Colony Optimization (ACO), Genetic Algorithm (GA), and Particle Swarm
Optimization (PSO) [5-13].

Some classical approaches have been demonstrating for various tasks and giving good performance when they are implemented, but sometimes it falls into local optimum and needs more computation time because it needs more nodes to traverse from starting point to the endpoint, such as the Dijkstra algorithm. However, some heuristic approaches such as GA and PSO are simple and have more efficiency than Dijkstra, but it has low convergence because they get initial values randomly. The ACO algorithm has global search capabilities and more stable than the other previous methods mentioned above so it can find a better path.

The ACO algorithm is one of the optimization approaches inspired by the real ant behavior in searching food. ACO has been implemented in the MRPP problem because of the similarity with the foraging behavior of ant colonies [5,6]. There are a lot of contributions of ACO methods for an autonomous mobile robot in the RPP research such as planning trajectory, path, route, and navigation [5-11]. The ACO algorithm is suitable for solving combinatorial optimization problems like MRPP. For example, Wen et al. [5] proposed the hybrid ACO algorithm for RPP in static environments to find an optimum path. Zeng Bi et al [6] presented an improved ACS for mobile robot navigation include fuzzy modeling of the environment at the beginning. the best route according to the fuzzy cost function. Chia et al [7] presented mobile robot path planning using the ACO algorithm. Maurya and Shukla [8] compared generalized and modified ACO for MRPP in a static environment. Also, Zhangqi et.al apply GA for optimization and configuration parameters of basic ACO algorithms to MRPP [9]. Those traditional ACO algorithms have some weakness such as slow convergence and prematurity, so it is necessary to be modified and improved by adopts a new pheromone trail updating rule and dynamic adjustment of the evaporation rate [14, 15] to accelerate the convergence speed and prevent not fall into the local minimum.

This paper presents a proposed approach called a modified Ant Colony Optimization (MACO) algorithm which has been developed by the author to get optimal path planning in the tested environment. This algorithm chooses the generated random nodes in the obstacles-filled environment based the transition probability and then finishing it by point to point (PTP) motion planning to make an optimal path [14]. The proposed approach will be implemented for waiter robot path planning in the restaurant comprises many obstacles.

The remainder paper is organized as follows. Section 2 give detail explanation of the proposed approach for waiter robot path planning. The review of M-ACO algorithm will be presented in section 3. Evaluation of path planning approaches by simulation with two scenarios is described in section 4. And in the last section, we summarize all of the research results.

2. PROPOSED ALGORITHM FOR WAITER ROBOT

This paper proposed the M-ACO algorithm for the waiter robot path planning. The framework of waiter robot path planning using the M-ACO algorithm [14] is displayed in Fig. 1. The mathematical notations are defined in Table 1.

The M-ACO algorithm requires some nodes as medium for waiter robot to plan a path. Random nodes generator is used to create some nodes based on the environment map of waiter robot workspace.

2.1. Working Environment Modeling

The working environment is modeled as a restaurant in 2D space. There are some static obstacles and a dynamic one in the restaurant. The example of static obstacles such as some tables and some chairs, also a dynamic one such as the people walking around in the restaurant. The restaurant such modeled as the environment is ilustrated in Fig. 2.

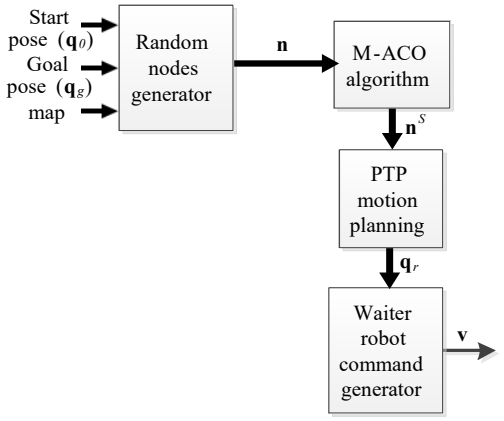


Figure 1: The Frame Work of Proposed Algorithm for Waiter Robot

Table 1: Notation List of M-ACO Algorithm for Waiter Robot

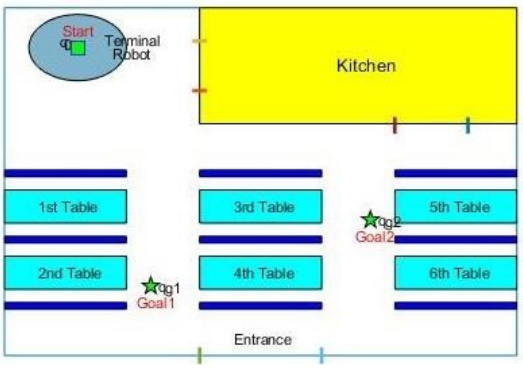
Symbol	Description
$\mathbf{q}_0$	Start robot pose
$\mathbf{q}_g$	Goal robot pose
$\mathbf{n}$	Random nodes generated set
$\mathbf{n}^S$	Selected next nodes set
$\mathbf{q}_r$	Robot pose for the selected nodes
$\mathbf{v}$	Linear and angular velocity of mobile robot

In this model, it is assumed the restaurant has six tables that each table is surrounded by two long chairs on the two-faced side of each other, there is one gate as a doorway (called "Entrance") and a kitchen. Initially, the waiter robot is placed at"Terminal Robot" and then he starts serving from a terminal robot is located near the kitchen as seen in Fig. 2. Customers can enter the restaurant through the entrance, then they could choose what table they want to be sitting.

After customers inside the restaurant, they order some food while they are sitting on the chair. There are nine chairs surrounded six tables have been located exactly. The table's location is used for determining the target point of the waiter robot.

This paper using two scenarios for different target points to evaluate the performance of the proposed approaches. For example, customers come in then sit on the 2nd table so we locate the goal point (green star) close from the 2nd table (Goal 1) as well as they sit on the 5th table so we locate the goal point (green star) close from on the 5th table (Goal 2) as ilustrated in Fig. 2.

The main objective of this paper is to build a safely shortest route from the terminal robot (start point) to the target point (Goal 1 or Goal 2) which



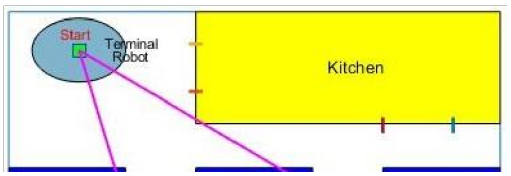


Figure 3:

avoids every obstacle inside the restaurant. Shortest route is defined as euclidean distance between two points connected. Firstly, drawing a straight-line connecting start point to each goal point as shown in Fig. 3. However, since this straight-line crash and across the 1st table (goal1) and the 3th table (goal2), so it's necessary to make an alternative route to avoid these obstacles using path planning approaches.

2.2. Random Nodes Generator

The start pose of waiter robot is presented as $\mathbf{q}_0 = [x_0\ y_0\ \square_0]^T$, while goal pose as $\mathbf{q}_g = [x_g\ y_g\ \square_g]^T$, and map are input to the random nodes generator. It generates some random nodes **n** around straight lines but out of the obstacles between start and goal point as described in Fig. 4.

There are the number of random nodes R are determined before they are processed by the MACO algorithm. The determination number of generated random nodes **n** based on the complexity of the traversed path will be explained in the next section. For instance, there are 10 random nodes generated for the first case (start to goal 1) are displayed in Fig. 4 (left) and 15 random nodes generated for the second case (start to goal 2) are displayed in Fig. 4 (right).

The generated random nodes are defined as $\mathbf{n} = [\mathbf{n}_1, \mathbf{n}_2, \ldots \mathbf{n}_R]^T$. These random nodes **n** are input to the M-ACO algorithm.

2.3. Modified Ant Colony Optimization Algorithm

The general description of the M-ACO algorithm processes is presented in [14] used the following formulas.

Journal of Theoretical and Applied Information Technology
15th October 2021. Vol.99. No 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

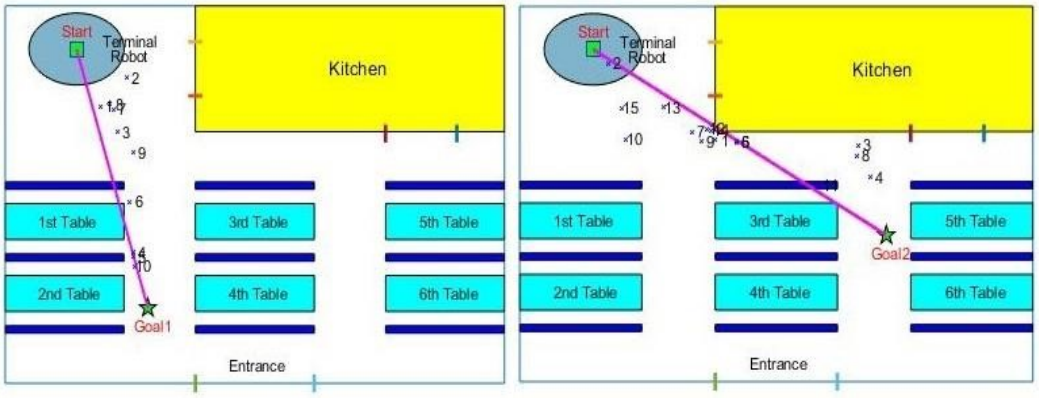


Figure 4: Generated random nodes between: start and goal 1 (left) and start and goal 2 (right)

First, we define the nodes between start and goal point. Then establish a link (*i, j*) from i^{th} node to the destination j^{th} node, this link has cost d_{ij} in

Eq. (1) which is accordance with a pheromone concentration denoted as τ_{ij} that is defined in Eq. (2).

$$d_{ij} = \sqrt[2]{(x_i - x_j)^2 + (y_i - y_j)^2}$$

$$\tau_{ij} = \frac{1}{d_{ij}} \quad (1)$$

(2)

As stated in [14], all ants build a route from the initial node (the nest) to the next node (the food source) for one cycle. They use the probabilistic formula $p_{ij}^k(t)$ as presented in Eq. (3) to select the next node.

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha}{\sum_{k \in allowed_k} d_k [\tau_{ik}(t)]^\alpha} & \\ 0 & \text{If } j \notin allowed_k \end{cases} \quad (3)$$

In Eq. (3), $p_{ij}^k(t)$ otherwise

denoted the probability of transition from node *i* to node *j* in the link (*i,j*) for the k^{th} ant at time t; $\tau_{ij}(t)$ is pheromone concentration of link (*i,j*); α is a positive constant used as gain for the pheromone concentration

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n^k} \Delta\tau_{ij}^k$$

influence; and $allowed_k$ states the nodes that have not been tracked by the k^{th} ant.

The pheromone concentration $\tau_{ij}(t)$ always changed caused of evaporation factor ρ according to the following Eq. (4) [14]:

(4) So, the pheromone concentration must be updated by using Eq. (5) [14]:

$$\tau_{ij}(t) = (1-\rho).\tau_{ij}(t) \quad \tau_{ij}(t) \quad (5)$$

The pseudo code of M-ACO algorithm is shown in Algorithm 1.

Algorithm 1 M-ACO algorithm

Input: random nodes set **n**
number of random nodes R
number of ants m
init pheromone τ
gain for pheromone α
evaporation rate ρ
global transfer factor P_0
For i=1 to m do take all possible node **For** j=1 to R do
calculate cost d_{i-j}
calculate probability transition p_{i-j}
While $p_{i-j} > P_0$ selected next nodes set **n***s*
STOP
End **End**

$$\tau_{i-j} := (1-\rho)\tau_{i-j}$$

End

2.4. Global Transfer Factor

As presented in the previous section, ant move from one node to another node based on ant transition probability in ACO algorithm [5-7]. This research proposes an addition parameter called global transfer factor P_0 to modify the ACO algorithm [14], this parameter is used to control ant transition probability so it can accelerate algorithm to obtain convergence more quickly.

2.5. Selection of Random Nodes

The M-ACO algorithm is used to select the random nodes set **n** by ant transition probability in Eq. (3) to find the best route. This transition probability $p_{ij}^{k}(t)$ is compared with P_0 value until the best path obtained. The result of the nodes selection by M-ACO algorithm are defined as $\mathbf{n}^S = [\mathbf{n}_1,\mathbf{n}_2,...\mathbf{n}_i,\mathbf{n}_{i+1}]^T$. then they are connected each other to establish links between nodes $\mathbf{n}_i$ and $\mathbf{n}_{i+1}$.

2.6. PTP Motion Planning

The selected random nodes $\mathbf{n}^S$ are used to do point to point (PTP) motion planning using a linear function, such that the waiter robot to go from the initial pose to the final pose without collision any tables and chairs. Then the waiter robot moves from current node to the next node yield the robot pose $\mathbf{q}_r$.

2.7. Waiter Robot Command Generator

The waiter robot in this paper is controlled by a mobile robot (MR) which is exemplified as a rectangle with two rear wheels.

Mobile robot pose is represented in Eq. (6) as follows.

$$\mathbf{q}_r = [x_r\ y_r\ \theta_r]^T \quad (6)$$

where MR position is represented by x_r and y_r, and MR orientation by θ_r, then by invers kinematic we get MR velocity **v** that is defined in Eq. (7).

$$\mathbf{v} = [v_r\ \omega_r]^T \quad (7)$$

where v_r is the linear velocity and ω_r is the angular velocity.

3. SIMULATION RESULT AND DISCUSSION

This section presents the simulation result of proposed path planning approaches to a waiter robot. In this simulation, a waiter robot moves from start pose to the goal poses (goal 1 and goal 2).

Length of mobile robot body is assumed 2.5 cm, while width is 1.8 cm, and radius of MR wheels are 0.5 cm. Mobile robot will move with constant speed 20 cm/s

First, we evaluate the M-ACO algorithm by making one of the parameters variably but other parameters remain constant for the first waiter robot path planning case as shown in Fig. 5 and Fig. 6.

According to Fig. 5, best path obtained at 78.35 cm when $P_0 > 0.2$ for $\alpha = 1$ and $\beta = 0.9$. Refer to Fig. 6, best path obtained at 80.28 cm when $\beta > 0$ for $\alpha = 1$ and $P_0 = 0.9$. Two figures show that the greater value of P_0 and β so developed stability of the best path length.

Selection of P_0 and β value is based on path length obtained from the simulation.

After determining the proper parameter values of M-ACO algorithm, then we will evaluate the number of nodes that produce the optimal path from start to goal 1. Implementation of each path planning algorithm are M-ACO, Astar, and Dijkstra for waiter robot to determine the appropriate number of nodes is shown in Fig. 7.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

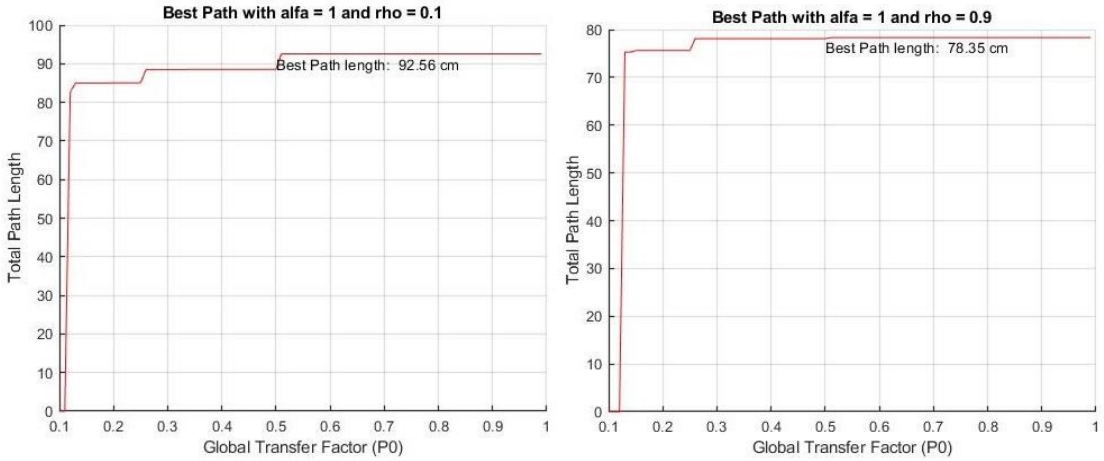


Figure 5: Total Path Length vs Global Transfer Factor, α = 1, □=0.1 (left) and α = 1 and □=0. 9 (right)

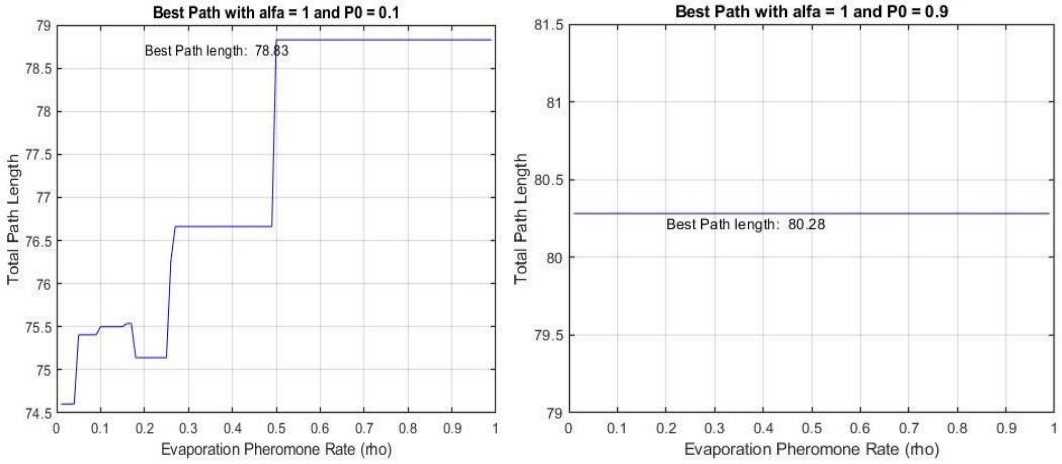


Figure 6: Total Path Length vs Evaporation Pheromone Rate, α = 1, P₀=0.1 (left) and α = 1, P₀=0.9 (right)

Refer to Fig. 7, it can be seen that the best path of 77.91 cm using M-ACO algorithm for 10 random nodes and 76.41 cm for 15 random nodes, meanwhile best path by A-star and Dijkstra algorithm is 73.79 cm for 25 random nodes and 73.82 cm for 20 random nodes. The waiter robot path planning simulation of each approach for performance comparisons is shown in Fig. 8 and Fig. 9.

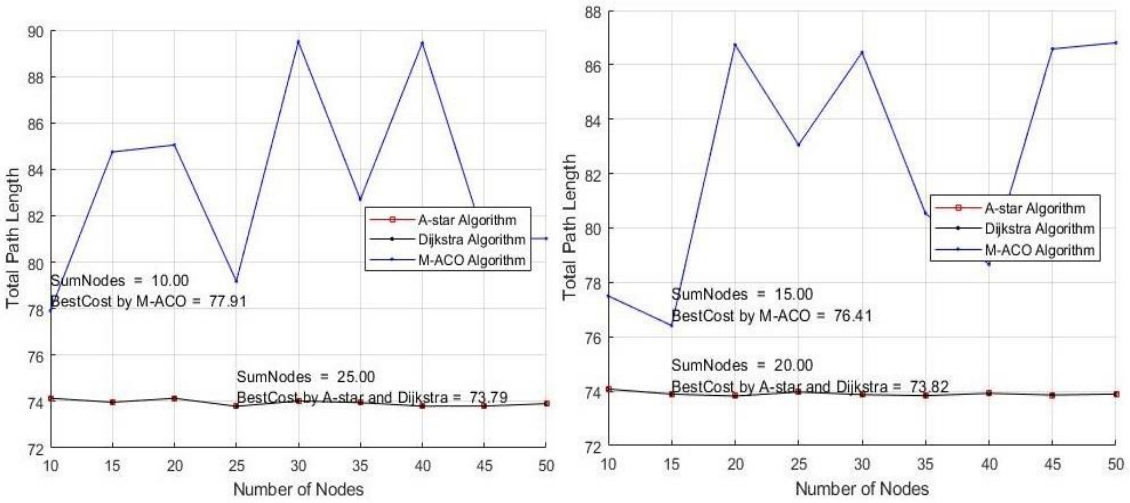


Figure 7: Total Path Length vs Number of nodes: M-ACO (left:10, right:15), A-star and Dijkstra (left:25, right:20)

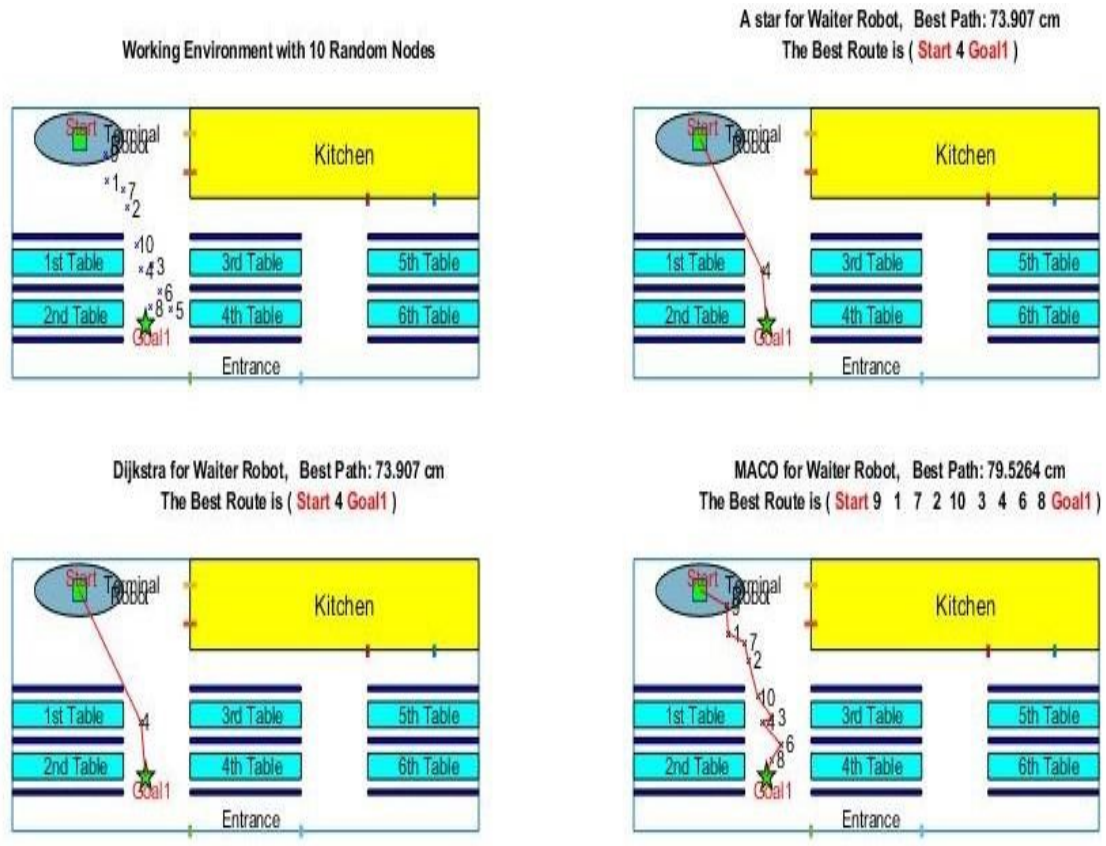


Figure 8: The Best Waiter Robot Route with 10 random nodes

Figure 9: The Best Waiter Robot Route with 15 random nodes

For the case in which the waiter robot moving from start to goal 2, we run simulations with some parameter values of the proposed algorithm as presented above, also include determining the number of nodes needed for the best path. The simulation results are shown in Fig. 10, Fig. 11, Fig. 12, Fig. 13, and Fig. 14.

Fig. 10 depicts about influence P_0 value on the best path, the result show path length of 96.03 cm is obtained for $P_0 > 0.5$ when $\alpha = 1$ and $\square = 0.9$. While Fig. 11 shows the best path of 105.59 cm for $\square > 0$ when $\alpha = 1$ and P_0=0.9.

According to Fig. 12, it can be seen that the best path of 90.07 cm using the M-ACO algorithm for 10 random nodes and 95.70 cm for 15 random nodes, meanwhile best path by A-star and Dijkstra algorithm is 81.29 cm for 25 random nodes and 81.25 cm for 20 random nodes. Path planning simulation of each approach for performance comparisons is shown in Fig. 13 and Fig. 14. Optimal path planning is shown Fig. 14, the best route from start to goal 2 without crashing any obstacles is obtained for 15 random nodes

Journal of Theoretical and Applied Information Technology
15th October 2021. Vol.99. No 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

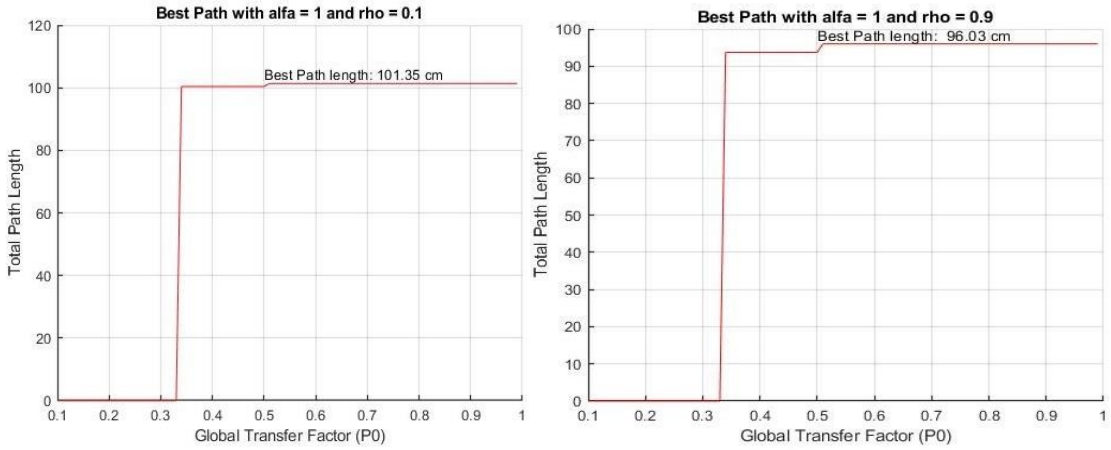


Figure 10: Total Path Length vs Global Transfer Factor, α = 1, □ = 0.1 (left) and α = 1 and □ = 0. 9 (right)

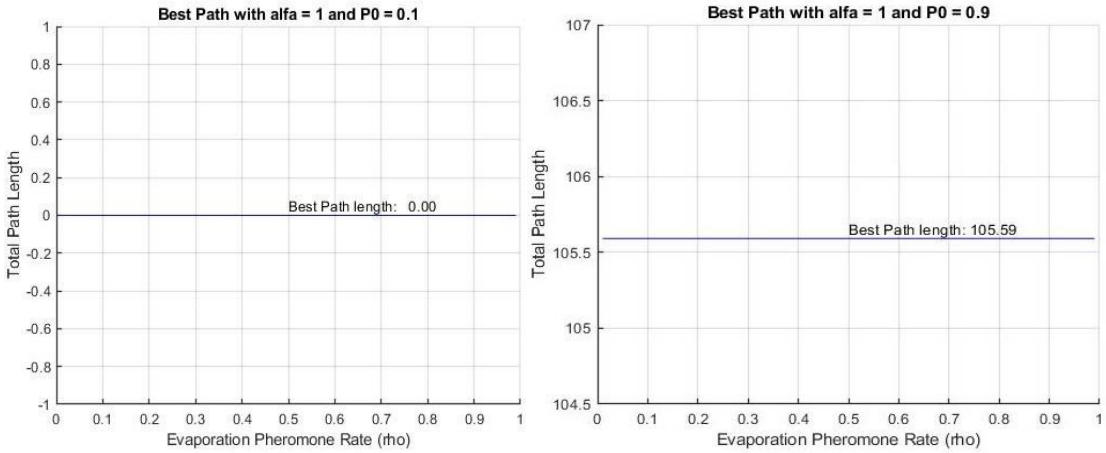


Figure 11: Total Path Length vs Evaporation Pheromone Rate, α = 1, P 0= 0.1 (left) and α = 1, P0= 0.9 (right)

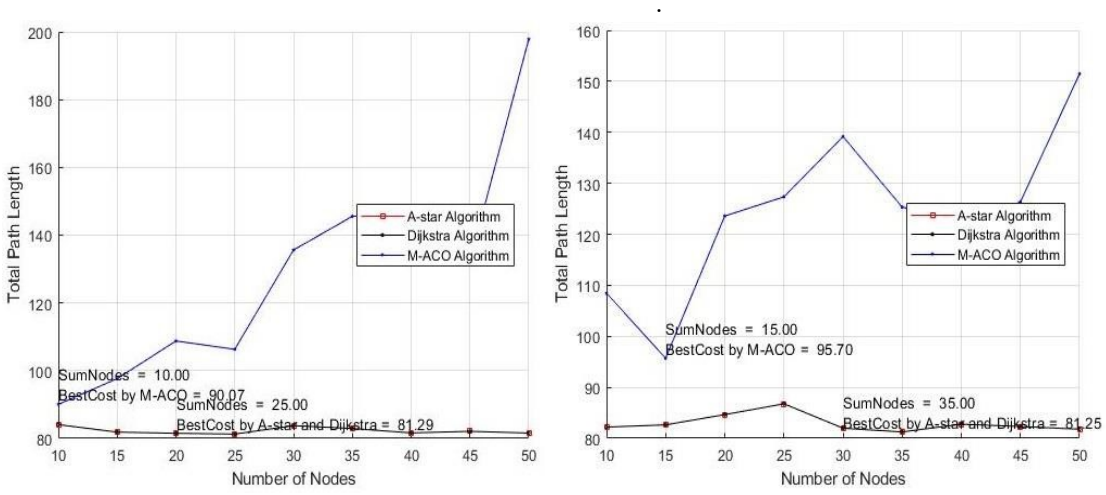


Figure 12: Total Path Length vs Number of nodes: M-ACO (left:10, right:15), A-star and Dijkstra (left:25, right:35)

The performance comparisons from the evaluations that have been done are shown in Table 2. Table 2 displays a summary of all the simulation results above. A-star and Dijkstra algorithm give shorter path length than M-ACO algorithm, but the waiter robot nudge obstacle.

This shows that the M-ACO algorithm was more reliable than A-star and Dijkstra algorithm when they are evaluated to the waiter robot in the same environment.

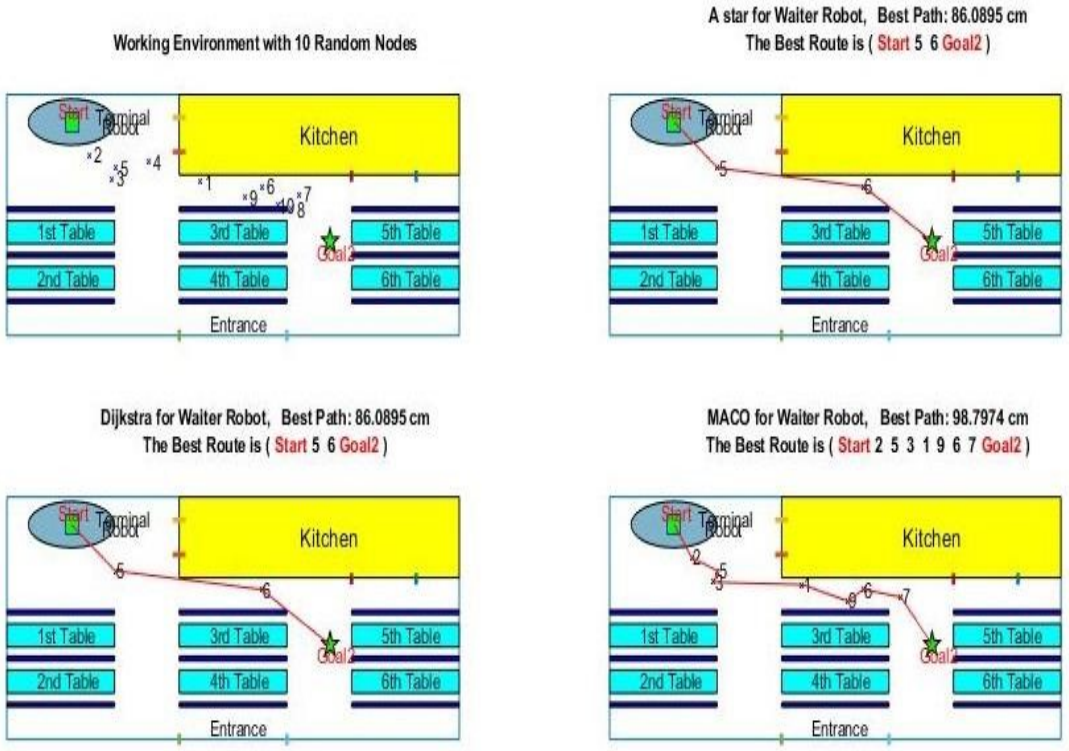


Figure 13: The Best Waiter Robot Route 10 random nodes

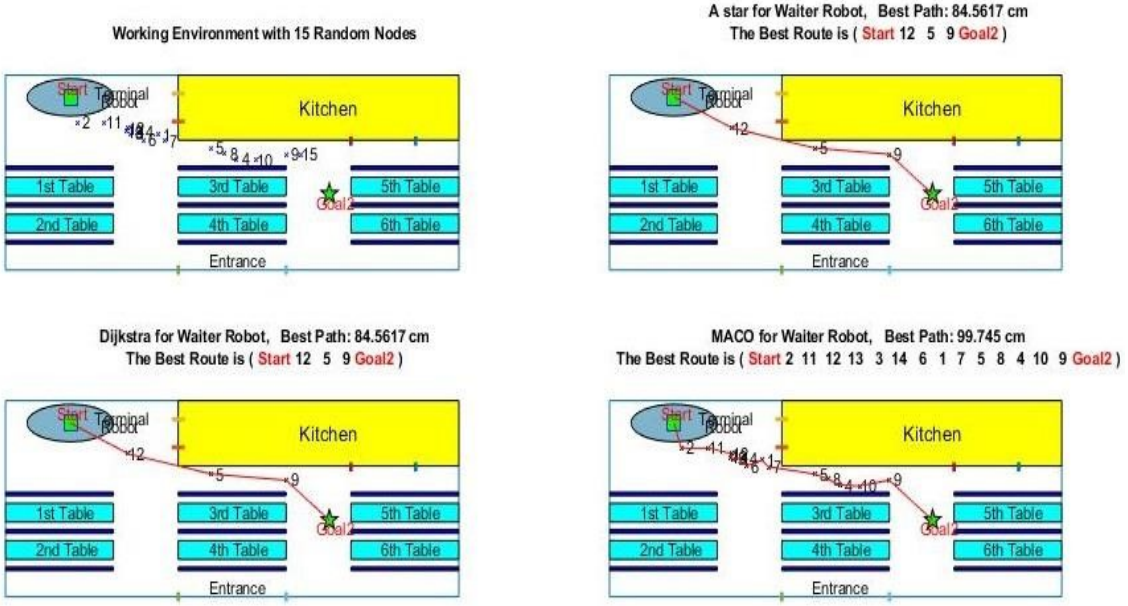


Figure 14: The Best Waiter Robot Route 15 random nodes

4. CONCLUSION

Briefly, we proposed an implementation of the M-ACO algorithm to get optimal path planning for waiter robot. Then this proposed approach is compared with A-star and Dijkstra algorithm for the same scenarios. The M-ACO, A-star, and Dijkstra algorithm have similarities in the path planning processes, so this is the reason why these approaches are used and compared.

After doing many simulations with Matlab, some performance comparisons of three algorithms with related inferences has been completed so we could obtain the best path and safe route for the waiter robot. The waiter robot motion planning by the M-ACO algorithm gives an optimal result i.e safely route (not crash with obstacles), while by the A-star and Dijkstra algorithm crash. However, path length obtained by the M-ACO algorithm is longer than path length by the A-star and Dijkstra algorithm.

REFERENCES

[1] M. Haegele, "World Robotics 2015 Service Robots", *International Federation of Robotics (IFR) Statistical Department*, Germany, 2015.

[2] H. N. Mohd Shah, L. Z. Chen, Z. Kamis, A. Ahmad, M. R. Baharon, "Mobile Robot Navigation System Vision Based Through Indoor Corridors", *International Journal of Mechanical & Mechatronics Engineering*, Vol. 19 (5, 2019), pp. 158-167.

[3] C. Chen, Q. Gao, Z. Song, O. Liping, X. Wu, "Catering Service Robot", *Proceeding of 8th World Congress on Intelligent Control and Automation* (WCICA), IEEE Xplore, 2010, pp. 599-604.

[4] Z. Zhang, Z. Zhao, "A Multiple Mobile Robots Path Planning Algorithm Based on A star and Dijkstra Algorithm", *International Journal of Smart Home*, Vol.8, No.3, 2014, pp.75-86.

[5] S. I. Abdulkadir, S. A. Fadzli, A. A. Jamal, "Indoor Global Path Planning Based On Critical Cells Using Dijkstra Algorithm", *Journal of Theoretical and Applied Information Technology*, Vol. 79, No. 1, 2015, pp. 115-121.

[6] Z. Q. Wen, Z. X. Cai, "Global Path Planning Approach Based on Ant Colony Optimization Algorithm", *J. Cent. South Univ. Technology*, 13, 2006, pp. 707–712.

[7] J. Zhang, H. Li, L. Ren, Z. Shi, "Scheduling Optimization in Construction Project Based on Ant Colony Genetic Algorithm", *Journal of Theoretical and Applied Information Technology*, Vol. 48, No. 3, 2013, pp. 15401545.

[8] Z. Bi, Y. Yimin, X. Yisan, "Mobile Robot Navigation in Unknown Dynamic Environment Based on Ant Colony Algorithm", *Global Congress on Intelligent Systems*, 2009.

[9] S. H. Chia, K. L. Su, J. H. Guo, C. Y. Chung, "Ant Colony System Based Mobile Robot Path Planning", *Fourth International Conference on Genetic and Evolutionary Computing*, 2010, pp. 210 – 213.

[10] R. Maurya, A. Shukla, "Generalized and Modified Ant Algorithm for Solving Robot Path Planning Problem", *3rd International Conference on Computer Science and Information Technology*, Chengdu, 2010, pp. 643-646.

[11] H. Yang, Q. Jie, Y. Miao, "A New Robot Navigation Algorithm Based on a DoubleLayer Ant Algorithm and Trajectory Optimization", *IEEE Transactions on Industrial Electronics*, Vol. 66, No. 11, November 2019

[12] W. Zhangqi, Z. Xiaoguang, H. Qingyao, "Mobile Robot Path Planning based on Parameter Optimization Ant Colony Algorithm", *Advanced in Control Engineering and Information Science, Procedia Engineering*, Vol. 15, 2011, pp. 2738 – 2741.

[13] X. Xu, Y. Li, Y. Yang, H. Xu, "A Method of Trajectory Planning for Ground Mobile Robot Based on Ant Colony Algorithm", *Proceedings of the 2016 IEEE International Conference on Robotics and Biomimetics*, Qingdao, China, December 3-7, 2016.

[14] I. Al-Taharwa, A. Sheta, M. Al-Weshah, "A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment", *Journal of Computer Science 4*, Vol. 4, 2008, pp. 341344.

[15] H. S. Dewang, P. K. Mohanty, S. Kundub, "A Robust Path Planning for Mobile Robot Using Smart Particle Swarm Optimization", *International Conference on Robotics and Smart Manufacturing* (RoSMa2018), Procedia Computer Science, 2018, pp. 290297.

[16] N. Habib, A. Soeprijanto, Dj. Purwanto, M. H. Purnomo, "Mobile Robot Motion Planning to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Avoid Obstacle Using Modified Ant Colony Optimization", *Applied Mechanics and Materials*, Vol. 776, April 2015, pp. 396402.

[17] K. Akka, F. Khabeer, "Mobile Robot Path Planning Using An Improved Ant Colony Optimization", *International Journal of Advanced Robotic Systems*, 2018.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2. *The Performance comparison for variation of M-ACO parameter, Dijkstra, and A-star in simulation*

Case	Number of Nodes	Algorithm	P_0	□	Route via Nodes	Path Length	Safely Motion (Crash or Not)
1	10	M-ACO	0.9	0.9	Start-9-1-7-2-10-3-4-6-8-Goal1	79.5264	Not
		Dijkstra			Start-4-Goal1	73.9070	Crash
		A-star			Start-4-Goal1	73.9070	Crash
	15	M-ACO	0.9	0.9	Start-10-8-11-4-14-13-3-15-1-712-5-2-Goal1	85.9857	Not
		Dijkstra			Start-13-Goal1	73.9756	Crash
		A-star			Start-13-Goal1	73.9756	Crash
2	10	M-ACO	0.9	0.9	Start-2-5-3-1-9-6-7-Goal2	98.7974	Not
		Dijkstra			Start-5-6-Goal2	86.0895	Crash
		A-star			Start-5-6-Goal2	86.0895	Crash
	15	M-ACO	0.9	0.9	Start-2-11-12-13-3-14-6-1-7-5-84-10-9-Goal2	99.7450	Not
		Dijkstra			Start-12-5-9-Goal2	84.5617	Crash
		A-star			Start-12-5-9-Goal2	84.5617	Crash