Modified Q-Learning Algorithm for Mobile Robot Path Planning Variation using Motivation Model

Hidayat [1], Agus Buono [2], Karlisa Priandana [3]*, Sri Wahjuni [4]
[1, 2, 3, 4] Department of Computer Science, Faculty of Mathematics and Natural Sciences, IPB University, Bogor, Indonesia
[3] Center for Transdisciplinary and Sustainability Sciences, IPB University, Bogor, Indonesia
[1] Department of Computer Engineering, Universitas Komputer Indonesia, Bandung, Indonesia
Email: [1] dedehidayat@apps.ipb.ac.id, [2] agusbuono@apps.ipb.ac.id, [3] karlisa@apps.ipb.ac.id, [4] my_juni04@apps.ipb.ac.id
*Corresponding Author

***Abstract*—Path planning is an essential algorithm in autonomous mobile robots, including agricultural robots, to find the shortest path and to avoid collisions with obstacles. Q-Learning algorithm is one of the reinforcement learning methods used for path planning. However, for multi-robot system, this algorithm tends to produce the same path for each robot. This research modifies the Q-Learning algorithm in order to produce path variations by utilizing the motivation model, *i.e.* achievement motivation, in which different motivation parameters will result in different optimum paths. The Motivated Q-Learning (MQL) algorithm proposed in this study was simulated in an area with three scenarios, *i.e.* without obstacles, uniform obstacles, and random obstacles. The results showed that, in the determined scenario, the MQL can produce 2 to 4 variations of optimum path without any potential of collisions (Jaccard similarity = 0%), in contrast to the Q-Learning algorithm that can only produce one optimum path variation. This result indicates that MQL can solve multi-robots path planning problems, especially when the number of robots is large, by reducing the possibility of collisions as well as decreasing the problem of queues. However, the average computational time of the MQL is slightly longer than that of the Q-Learning.**

Keywords—Mobile Robot; Motivated Q-Learning; Motivation Model; Path Planning; Q-Learning Algorithm.

I. INTRODUCTION

Agricultural technology is rapidly advancing towards the Agriculture 4.0 paradigm. Agriculture 4.0, in [1] chapter 2, refers to the use of artificial intelligence, big data, Internet of Things (IoT), and robotics to increase the efficiency of activities in agricultural production activities. Javaid *et al* in [2] mentioned the importance of implementing robotics in smart farming. However, the change from traditional technology to automated devices provides opportunities and challenges [3], [4], including the use of agricultural robots [5]–[11]. Oliveira *et al* [5] showed the notable advances in mobile robotics and the advantages of investing in technologies. The development of agricultural robotic systems will continue to increase their efficiency and robustness. Another research has also been involved to get solutions for navigation problems on a mobile robot in agriculture [12]–[15].

Autonomous navigation is an important aspect in the field of agricultural robots [12], [16], which covers four key requirements: mapping, localization, motion control, and path planning. Path planning is an essential issue in robotic problems. This task revolves around identifying rotational actions and a series of translations to move from the initial position to the goal while avoiding obstacles [17]. The exploration of robotic path planning is a critical area of investigation in the field of robotics, including in the use of mobile robots in agricultural settings [6], [18].

Many intelligent optimization algorithms have been offered to help robots optimize their paths. These algorithms draw inspiration from natural phenomena or biological groups, such as the ant colony algorithm (ACO) [19]–[28], genetic algorithm (GA) [29]–[32], [33], [34], and particle swarm optimization (PSO) [35]–[39] [40]. Other algorithms such as fuzzy algorithm [41]–[46], A* algorithm [47]–[50], cuckoo algorithm [51]–[53], improved artificial fish swarm algorithm [54], modified probabilistic roadmap algorithm [55], [56], artificial potential field algorithm [57]–[59] [69]–[71], and hybrid algorithms [60]–[64] [57], [72] have also been implemented in robot path planning. In addition, reinforcement learning also has been used to solve path planning problems, as seen in studies [65]–[69].

The Q-Learning algorithm is one of the reinforcement learning algorithms that is currently employed in path planning. It is a classical reinforcement learning algorithm that has been implemented in several studies for producing optimum path [68]–[75]. It is frequently used for path planning on moving robots [69]–[72], [75], [76]. In general, research studies indicated that the benefit of Q-Learning is that it always produces an optimum path. However, the drawback, when the Q-Learning algorithm is applied to several robots in the same area with the same task, the resulting paths tend to be relatively the same. Hence, these paths have the potential for collisions between robots when all robots move simultaneously.

The objective of this study is to modify the Q-Learning algorithm by utilizing a motivation model to generate diverse yet optimum path options for multiple mobile robots in the same area. Previous studies have used motivation models in the algorithms to influence agents in making decisions [77]–[79]. In this study, the achievement motivation model [77] is incorporated in the Q-learning algorithm to produce variations of optimum path. We call this algorithm Motivated Q-Learning (MQL) algorithm. By having more than one optimum path, in the case where communication between robots is not available, the possibility of collisions can be reduced. In addition, in the case where the robots can communicate with each other, the queuing problem can be decreased.

II. Method

A. Reinforcement Learning (RL) and Q-Learning Algorithm

Reinforcement learning (RL) is one method in machine learning. It is a method of taking action based on the reward [80]. The rewards and penalties concepts are used to explore an environment. Five important terms are used in the Q-Learning algorithm, namely agent, state, action, reward, and penalty. In this case, the agent is a mobile robot as an object that moves in the environment. The position of the agent in the environment is represented by state (S). The action (A) represents movement the agent from one state to another state. Rewards are positive values that are given if the agent takes the correct action, while penalties are negative values that are given if the agent takes the incorrect action. Through exploration and exploitation, the agent gains experience. The exploration allows the agent randomly to visit all state-action pairs in the environment without considering the current state. On the contrary, exploitation maximizes the reward from the current state using the agent's acquired knowledge to select actions. One type of RL method is the Q-Learning algorithm [81].

On the Q-Learning algorithm, the Q values are stored in a two-dimensional Q table for each state and potential action. The algorithm chooses the action with the highest reward. The equation (1) is the Q-Learning equation by Watkins [80] to update the Q value.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (1)$$

The position of agent (A) or state at time t is represented by S_t. The agent action in state S_t is represented by A_t. The R_{t+1} is reward value that received after the agent executes action A_{t+1} in state S_t. The $Q(S_t, A_t)$ is generated by action A_t in state S_t. The discount factor (γ) serves as a variable determining the significance of upcoming rewards. Its value ranges between 0 and 1. A value near 0 implies that the agent prioritizes immediate rewards, while a value near 1 signifies the agent's consideration of future rewards. The learning rate (α), ranging from 0 to 1, affects the pace of achieving convergence. When α is close to 0, convergence takes a long time. Conversely, higher values prompt the agent to make drastic adjustments to the Q value, hindering convergence due to fluctuating outcomes. The pseudocode for Q-learning is presented in Algorithm 1.

Algorithm 1 Q-Learning Algorithm

```
1: Initial $Q(s,a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$
2: Looping for each episode:
3: Initial $S$
4:   Loop for each step in the episode:
5:      Select $A$ from $S$ by using policy from $Q$
6:      Take action $A$, observe $R$, $S'$
7:      $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
8:      $S \leftarrow S'$
9:   Until $S$ is target
```

Initially, all values of $Q(s, a)$ in the Q-table is set to zero. The s and a refer to the state and action, respectively, which are elements of the entire state space $(\mathcal{S}^+)$ and all possible actions of that state $\mathcal{A}(s)$. Then, initial state S is determined. The Q value is updated in the looping section.

During the iterative procedure, an action (A) is chosen for execution in the current state (S) based on the policy derived from Q-values. Following this, the agent selects an action (A) and observes both the reward (R) and the subsequent state (S'). The Q-value in the Q-table is then updated using equation (1). Additionally, the current state (S) is set to the value of the next state (S'). This looping process persists until the current state matches the target state.

Fig. 1 illustrates the process of Q-Learning. The state S_t is denoted as the initial state (n), and the feasible actions (A) are obtained from Q using the expression $(\gamma \max_a Q(S_{t+1}, a))$. This selected action transitions the agent to the subsequent state (S_{t+1}), acquiring a reward value (R_{t+1}) in the process. This sequence continues until convergence is achieved.

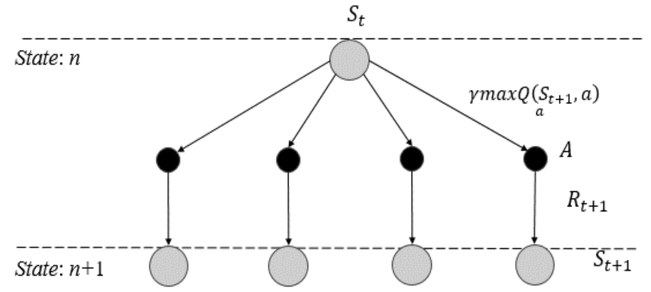


Fig. 1. Illustration of the Q-Learning process

B. Achievement Motivation Model

The motivation model is a model that can be applied to agent intelligence to help identify, prioritize, choose, and adapt to targeted goals. The application of a motivation model in path planning algorithms can help mobile robots move according to the given motivation, resulting in different path variations in the same area and goal. One of the motivation models proposed by Merrick and Shafi is achievement motivation [77]. This motivation can be defined as the need for success or achievement of excellence. According to [77], achievement motivation is based on the estimation of the probability of success and the difficulty of the task, which is modeled by equation (2).

$$T_{ach}(G) = \frac{S_{ach}}{1 + e^{\rho^+_{ach}\left(M^+_{ach} - P_s(G)\right)}} - \frac{S_{ach}}{1 + e^{\rho^-_{ach}\left(M^-_{ach} - P_s(G)\right)}} \quad (2)$$

This model has six parameters $P_s(G)$, M^+_{ach}, M^-_{ach}, ρ^+_{ach}, ρ^-_{ach}, and S_{ach}. $P_s(G)$ is the subjective probability of successfully achieving the goal G. M^+_{ach} is the sigmoid turning point for approach motivation, and M^-_{ach} is the sigmoid turning point for avoidance motivation. ρ^+_{ach} is gradient for approach and ρ^-_{ach} is gradient for avoidance. Finally, S_{ach} is a measure of the relative strength of achievement motivation.

When the approach turning point is to the left of the avoidance turning point (i.e., $M^+_{ach} < M^-_{ach}$), the resulting tendency represents individuals who are motivated to succeed. $M^+_{ach} > M^-_{ach}$ represents individuals motivated by failure. $\rho^+ > 0$ represents the gradient of approach to success, while $\rho^- > 0$ represents the gradient of avoiding failure. The T_{ach} value can be used in the development or modification of artificial intelligence algorithms to influence decision-making processes. Determining the value of the variables can determine the value of the expected motivational tendency. In this case, is the tendency of achievement to avoid collision.

C. The Proposed Method

The proposed modified algorithm is presented in Fig. 2. The reward achievement (r_{ach}) is used to affect the update of the Q-value. R_{new} is the new reward value from the initial reward (R_{old}) added to the reward achievement (r_{ach}). The r_{ach} value is influenced by the probability value (P), α_{ach} value, and the K value, and also the T_{ach} value. Based on equation (3), P is proportional to r_{ach}. This means that the greater the value of P, the greater the value of r_{ach}. However, because T_{ach} is negative (to model obstacles), the larger P, the more negative r_{ach}. The greater the K value, the more negative r_{ach}. K value and α_{ach} value are used to affect the size of the r_{ach} value. In practice, r_{ach} is used to update R_{new} on the state used in the previous path. The more negative r_{ach}, the stronger the state condition which is considered as an obstacle or a state that cannot be passed, so that the next agent is expected to find a new path as a path variation. Therefore, equation (4) shows how the update in Q-value in the marked state and the update in Q-value in the normal state (unmarked) are calculated.

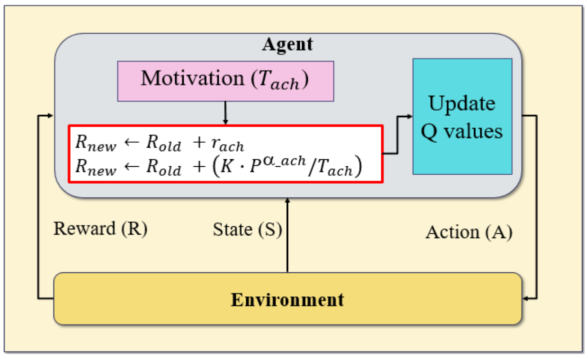

Agent
Motivation (T_{ach})
$R_{new} \leftarrow R_{old} + r_{ach}$
$R_{new} \leftarrow R_{old} + (K \cdot P^{\alpha\_ach}/T_{ach})$
Update Q values
Reward (R)
State (S)
Action (A)
Environment

Fig. 2. The development of the Q-Learning algorithm that utilizes a motivation model in the reward value

$$R_{new} = R_{old} + r_{ach} = R_{old} + \frac{K \cdot P^{\alpha_{ach}}}{T_{ach}} \tag{3}$$

$$Q(S_t, A_t) = \begin{cases} signed, Q(S_t, A_t) + \alpha\left[\left(R_{t+1} + \frac{K \cdot P^{\alpha_{ach}}}{T_{ach}}\right) + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \\ normal, Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \end{cases} \tag{4}$$

Fig. 3 shows the flowchart of the MQL algorithm for finding path variations based on the utilization of the motivation model. In the first route search, the initial reward value (step a) is used to update the Q value. After the process of updating the value (step b) in the Q table is completed, the agent will search for a route from the starting point to the target point (step c) based on the value in the Q table. In step d, if the first path (route 0) is found by the agent then the agent will save the path as route 0 (step e) and continue searching for the second path (route 1) by considering route 0, but if the route 0 is not found then the algorithm will inform that the route was not found (step f). In searching for the second path, the reward value in the state in route 0 will be updated (step g) using equation (4). Each reward on the state (route 0) will be added with the r_{ach} value. Then the algorithm will execute steps h, i, j, and k as well as steps b, c, d and e for route 1. If route 1 is not found then the algorithm will go to step f to inform that route was not found.

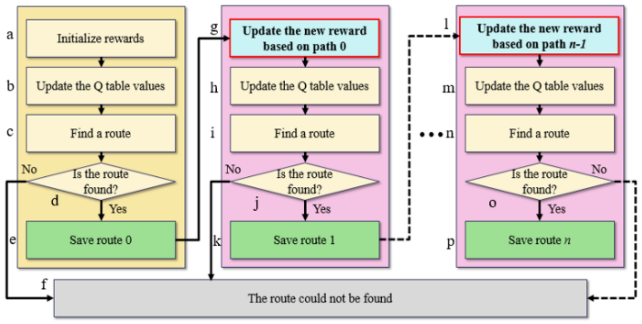

a
Initialize rewards
b
Update the Q table values
c
Find a route
No
Is the route found?
d
Yes
e
Save route 0
g
Update the new reward based on path 0
h
Update the Q table values
i
Find a route
No
Is the route found?
j
Yes
k
Save route 1
l
Update the new reward based on path *n-1*
m
Update the Q table values
n
Find a route
Is the route found?
No
o
Yes
p
Save route *n*
f
The route could not be found

Fig. 3. MQL flowchart with reward value update

Likewise for the search for the next route variation, the reward value in the state (for example, route 1) will be updated by adding the r_{ach} value to the old reward (step l). Then the algorithm will execute steps m, n, o, and p as well as steps b, c, d and e for route 2. If the path search does not find the target point, then the search will be terminated with a path not found notification. The pseudo code of MQL is shown as in Algorithm 2.

Algorithm 2 MQL Algorithm

```
Initial $Q(s,a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$
Initial $M^+_{ach}$, $M^-_{ach}$, $\rho^+_{ach}$, $\rho^-_{ach}$, and $S_{ach}$ for $T_{ach}$
Initial $K$, and $\alpha_{ach}$ for $r_{ach}$
Initial $P$ for $T_{ach}$ and $r_{ach}$
Calculate $T_{ach}$ and $r_{ach}$
Initial $S$
Initial $R$
Initial $Route$
Looping for each route:
    Loop for each step in the episode:
        Select $A$ from $S$ by using $policy$ from $Q$
        Take action $A$, observe $R$, $S'$
        $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
        $S \leftarrow S'$
    Until $S$ is target
    Save $Route$
    Update $R$ by adding $r_{ach}$ to $R$
Until the route is not found
```

The MQL procedure is developed from the Q-Learning procedure. In the MQL procedure, we add variables M^+_{ach}, M^-_{ach}, ρ^+_{ach}, ρ^-_{ach}, and S_{ach}, P, K, α_{ach} to produce the T_{ach} and the r_{ach}. The r_{ach} will be added to reward R in the state used as the previous path.

Simulations were conducted in three areas with different obstacle conditions to determine the performance of the proposed method. Measurements in this research are the number of path variations, computation time, and the number of rewards for each path as well as the value of similarity between paths. In addition, a comparison was made on the Q-Learning algorithm.

III. RESULTS AND DISCUSSION

A. The Change in the Value of r_{ach} Based on P, K, and α_{ach}

The change in the value of P, K, α_{ach} and T_{ach} have a significant impact on the value of r_{ach}. The given values for variable P range from 0.1 to 1 with a step of 0.1, the value of

K ranges from 5 to 50 with a step of 5, and the value of α_{ach} ranges from 1 to 3. Meanwhile, the value of T_{ach} is obtained based on the variables of the motivation model and the value of P. Change r_{ach} value based on P, K, α_{ach} are shown in Table I, Table II, and Table III respectively. The lowest r_{ach} is -319.43, while the largest value is -2.94 (at α_{ach}= 1), -0.29 (at α_{ach}= 2) and -0.03 (at α_{ach}= 3).

In addition, the graph showing the changes in the value of r_{ach}, which is influenced by changes in the values of P, K, α_{ach} and T_{ach}, is displayed in Fig. 4, Fig. 5 and Fig. 6, respectively. The graph shows that as the values of α_{ach}, P and K increase, the r_{ach} value decreases. At K value is 5, the decrease in r_{ach} with respect to P is not significant. The lowest value of r_{ach} occurs when K is increased up to K=15, reaching -95.83. The changes in r_ach that are close to linear occur at α_{ach} values of 1 and 2. Meanwhile, when α_{ach} values are 3, significant changes in r_{ach} occur starting from P =0.5.

These results showed that, in accordance to equation (3), the larger the value of K, the greater the influence of the constant on the probability of obtaining a larger reward. However, if the T_{ach} value is negative, increasing the P value of will weaken the r_{ach} value. The greater the α_{ach} value, the greater the influence of the probability on the r_{ach} value. If the T_{ach} value is negative, increasing the α_{ach} value will also weaken the r_{ach} value. If the value of T_{ach} is negative, the greater the divisor in the equation, the smaller the r_{ach} obtained. In its application, the value of r_{ach} is utilized to update the reward value of the state that has been used in the previous path. The more negative the r_{ach} value, the stronger the state condition which is considered as an obstacle, so that the possibility of a collision is avoided.

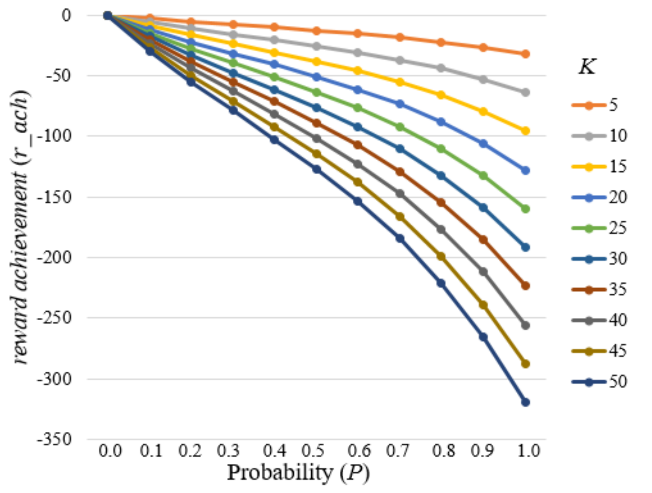


Fig. 4. The graph of r_{ach} value for P and K changes in α_{ach} = 1

B. The MQL Simulation

The MQL algorithm was simulated on a computer device with an Intel Core i5-3570 processor, clock speed of 3.4 GHz, and 4GB of RAM. The software used was Jupyter Notebook with Python 3.9 programming language. The simulation was conducted in an 11x11 area (121 states) with several scenarios, *i.e.* an obstacle-free area (scenario 1) and an area with obstacles i.e. scenario 2, and scenario 3. The values of learning rate (α) and discount factor (γ) that used were 0.9. The iteration used was 5000. The values of achievement motivation model variables that used were M^+_{ach}= 0.7, M^-_{ach}= 0.3, ρ^+_{ach}=ρ^-_{ach}= 2, and S_{ach}= 1. These values will give a tendency to avoid the failed. That means agent or robot will avoid obstacles or states that have been used by another agent. The value of P was alternated from 0.1 to 1. The testing was conducted by providing the values of P, K, and α_{ach} to produce the values of T_{ach} and r_{ach}.

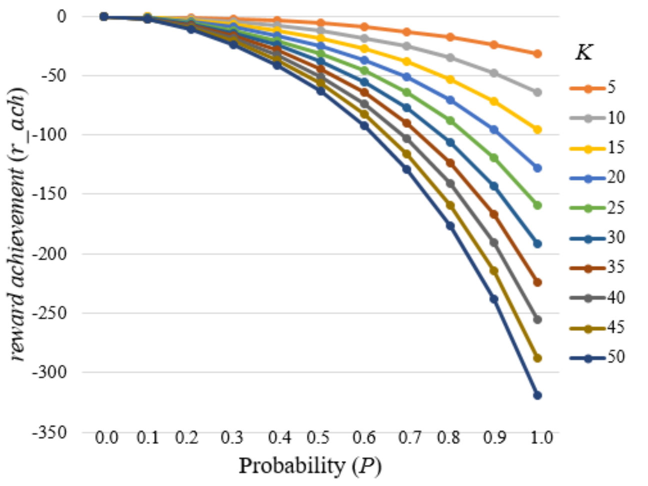


Fig. 5. The graph of r_{ach} value for P and K changes in α_{ach} = 2

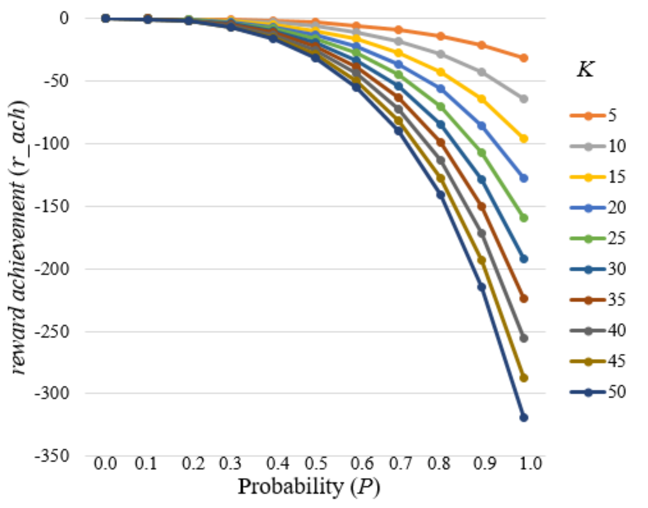


Fig. 6. The graph of r_{ach} value for P and K changes in α_{ach} = 3

The initial reward value for each passable state was -1, and the reward value for the target state was 999. Meanwhile, the reward value for the obstacle state was -100. The rewards used is defined as (5). Meanwhile, the reward value for the obstacle state was -100. In the simulation, the initial state was marked with green color, the target state was marked with orange color, and the obstacle state was marked with black color. Four paths are searched according to the four directions of agent movement *i.e.* forward, backward, left and right. In addition, four routes (simulation results) would be shown in different colors (black = route 0, blue = route 1, brown = route 2, and red = route 3).

$$reward = \begin{cases} 999, target \\ -1, free\ state \\ -100, obstacle \end{cases} \tag{5}$$

TABLE I. THE CHANGE IN THE VALUE OF r_{ach} AT α_{ach} = 1

P	r_{ach}									
	***K*=5**	***K*=10**	***K*=15**	***K*=20**	***K*=25**	***K*=30**	***K*=35**	***K*=40**	***K*=45**	***K*=50**
0.1	-2.94	-5.89	-8.83	-11.78	-14.72	-17.66	-20.61	-23.55	-26.50	-29.44
0.2	-5.52	-11.04	-16.55	-22.07	-27.59	-33.11	-38.63	-44.14	-49.66	-55.18
0.3	-7.90	-15.79	-23.69	-31.58	-39.48	-47.37	-55.27	-63.17	-71.06	-78.96
0.4	-10.23	-20.46	-30.69	-40.92	-51.15	-61.38	-71.61	-81.85	-92.08	-102.31
0.5	-12.67	-25.33	-38.00	-50.66	-63.33	-76.00	-88.66	-101.33	-114.00	-126.66
0.6	-15.35	-30.69	-46.04	-61.38	-76.73	-92.08	-107.42	-122.77	-138.11	-153.46
0.7	-18.42	-36.85	-55.27	-73.69	-92.12	-110.54	-128.96	-147.39	-165.81	-184.24
0.8	-22.07	-44.14	-66.22	-88.29	-110.36	-132.43	-154.50	-176.58	-198.65	-220.72
0.9	-26.50	-52.99	-79.49	-105.98	-132.48	-158.98	-185.47	-211.97	-238.46	-264.96
1	-31.94	-63.89	-95.83	-127.77	-159.72	-191.66	-223.60	-255.55	-287.49	-319.43

TABLE II. THE CHANGE IN THE VALUE OF r_{ach} AT α_{ach} = 2

P	r_{ach}									
	***K*=5**	***K*=10**	***K*=15**	***K*=20**	***K*=25**	***K*=30**	***K*=35**	***K*=40**	***K*=45**	***K*=50**
0.1	-0.29	-0.59	-0.88	-1.18	-1.47	-1.77	-2.06	-2.36	-2.65	-2.94
0.2	-1.10	-2.21	-3.31	-4.41	-5.52	-6.62	-7.73	-8.83	-9.93	-11.04
0.3	-2.37	-4.74	-7.11	-9.47	-11.84	-14.21	-16.58	-18.95	-21.32	-23.69
0.4	-4.09	-8.18	-12.28	-16.37	-20.46	-24.55	-28.65	-32.74	-36.83	-40.92
0.5	-6.33	-12.67	-19.00	-25.33	-31.67	-38.00	-44.33	-50.66	-57.00	-63.33
0.6	-9.21	-18.42	-27.62	-36.83	-46.04	-55.25	-64.45	-73.66	-82.87	-92.08
0.7	-12.90	-25.79	-38.69	-51.59	-64.48	-77.38	-90.28	-103.17	-116.07	-128.96
0.8	-17.66	-35.32	-52.97	-70.63	-88.29	-105.95	-123.60	-141.26	-158.92	-176.58
0.9	-23.85	-47.69	-71.54	-95.39	-119.23	-143.08	-166.92	-190.77	-214.62	-238.46
1	-31.94	-63.89	-95.83	-127.77	-159.72	-191.66	-223.60	-255.55	-287.49	-319.43

TABLE III. THE CHANGE IN THE VALUE OF r_{ach} AT α_{ach} = 3

P	r_{ach}									
	***K*=5**	***K*=10**	***K*=15**	***K*=20**	***K*=25**	***K*=30**	***K*=35**	***K*=40**	***K*=45**	***K*=50**
0.1	-0.03	-0.06	-0.09	-0.12	-0.15	-0.18	-0.21	-0.24	-0.26	-0.29
0.2	-0.22	-0.44	-0.66	-0.88	-1.10	-1.32	-1.55	-1.77	-1.99	-2.21
0.3	-0.71	-1.42	-2.13	-2.84	-3.55	-4.26	-4.97	-5.68	-6.40	-7.11
0.4	-1.64	-3.27	-4.91	-6.55	-8.18	-9.82	-11.46	-13.10	-14.73	-16.37
0.5	-3.17	-6.33	-9.50	-12.67	-15.83	-19.00	-22.17	-25.33	-28.50	-31.67
0.6	-5.52	-11.05	-16.57	-22.10	-27.62	-33.15	-38.67	-44.20	-49.72	-55.25
0.7	-9.03	-18.06	-27.08	-36.11	-45.14	-54.17	-63.19	-72.22	-81.25	-90.28
0.8	-14.13	-28.25	-42.38	-56.50	-70.63	-84.76	-98.88	-113.01	-127.14	-141.26
0.9	-21.46	-42.92	-64.39	-85.85	-107.31	-128.77	-150.23	-171.69	-193.16	-214.62
1	-31.94	-63.89	-95.83	-127.77	-159.72	-191.66	-223.60	-255.55	-287.49	-319.43

Furthermore, the similarity of the states on path variations was measured by the Jaccard similarity [82], [83] using equation (6). The *A* and *B* variables represent the sequence of states on route *A* and route *B*. The total states that are similar between *A* and *B* divided by the number of states in *A and B*.

$$sim(A,B) = \frac{The\ total\ of\ similar\ states\ on\ A\ and\ B}{The\ number\ of\ states\ in\ A\ and\ B} \times 100\% \tag{6}$$

1. *Scenario 1*

Scenario 1 simulates path planning a single agent and single target in an obstacle-free 11×11 area. Q-Learning algorithm simulation result produced four paths (each 17 states). However, all paths tend to be similar. In contrast, MQL simulation can produce several path variations. Table IV shows the detailed results of MQL simulation, which was run with different variations of α_{ach}, P and K. Here, we calculate the maximum number of paths with Jaccard similarity = 0%, which we call "safe" path variations. At α_{ach} = 1, two safe path variations were produced in 1 simulations, three safe path variations were produced in 44 simulations, and four path variations were produced in 55 simulations. At α_{ach} = 2, two safe path variations were produced in 1 simulations, three safe path variations were produced in 61 simulations, and four path variations were produced in 38 simulations. At α_{ach} = 3, two safe path variations were produced in 8 simulations, three safe path variations were produced in 56 simulations, and four path variations were produced in 36 simulations. Fig. 7 shows the recapitulation graph of the number of safe path variation.

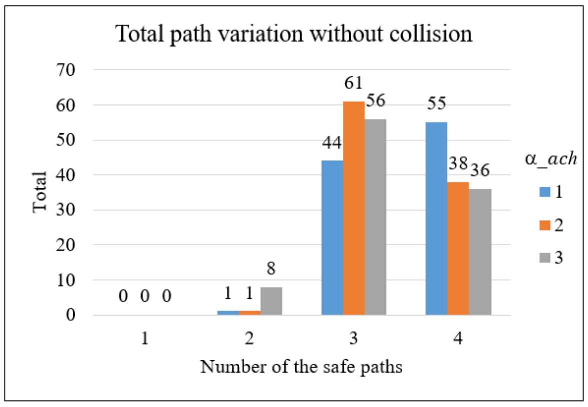


Fig. 7. The number of safe path variation in scenario 1

TABLE IV. THE NUMBER OF SAFE PATH VARIATIONS IN THE MOTIVATED Q-LEARNING ALGORITHM SIMULATION IN SCENARIO 1

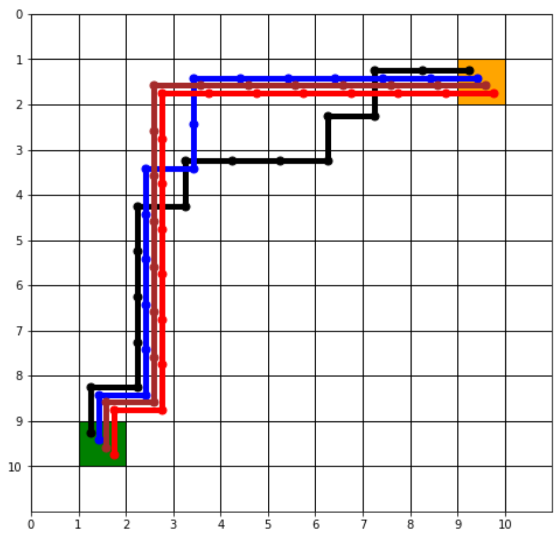
α_{ach}	**P**	***K***									
		5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
1	**0.1**	4	3	4	3	4	3	4	3	4	4
	0.2	3	4	3	4	4	4	3	3	3	4
	0.3	4	3	4	3	3	4	3	4	4	4
	0.4	4	4	3	3	3	4	3	4	4	4
	0.5	3	4	4	4	4	3	4	4	4	4
	0.6	3	3	4	3	3	4	3	4	4	3
	0.7	4	3	4	4	4	4	4	4	4	3
	0.8	3	3	4	4	3	3	3	3	4	3
	0.9	3	4	2	4	4	4	3	3	4	3
	1	4	4	3	4	3	3	3	4	3	3
2	**0.1**	2	3	4	4	4	3	3	4	3	4
	0.2	3	4	4	4	3	3	3	3	3	3
	0.3	3	3	3	4	3	3	4	3	4	4
	0.4	4	3	3	3	4	3	3	3	4	4
	0.5	4	3	3	3	3	3	3	4	4	4
	0.6	3	3	4	3	3	3	3	3	4	3
	0.7	3	3	3	4	4	3	4	4	3	4
	0.8	3	4	3	3	3	3	3	4	3	3
	0.9	4	4	3	3	4	4	4	3	3	4
	1	3	4	3	3	4	3	3	4	3	3
3	**0.1**	2	3	2	2	2	2	2	3	2	2
	0.2	3	4	3	4	3	3	3	3	3	3
	0.3	3	3	4	4	4	4	3	3	3	3
	0.4	4	3	3	3	3	3	4	4	4	4
	0.5	4	4	3	3	3	3	3	3	3	3
	0.6	3	4	3	3	3	4	3	4	3	3
	0.7	4	4	4	3	3	3	3	4	3	3
	0.8	3	4	3	3	4	3	3	3	3	3
	0.9	3	4	4	4	4	4	4	3	4	3
	1	4	3	4	3	4	4	3	4	4	4

Examples of paths produced by Q-Learning simulation and safe path variations generated in the simulation (with P = 0.7, K = 25, and α_{ach} = 2) are shown in Fig. 8(a) and Fig. 8(b), respectively. Even though the four paths MQL simulation have different lengths, all of them do not have a potential collision. Meanwhile, the average path-finding computation time on the Q-Learning algorithm is 1.170 ± 0.04 seconds (95% confidence level), while the average path-finding computation time on the MQL algorithm is 1.356 ± 0.21 seconds (95% confidence level). The computing time on MQL is slightly longer than Q-Learning algorithm. The average reward of Q-Learning simulation results is 983 while in MQL is 981. The difference in the average reward is only 2 points. Table V shows the simulation results data in scenario 1.

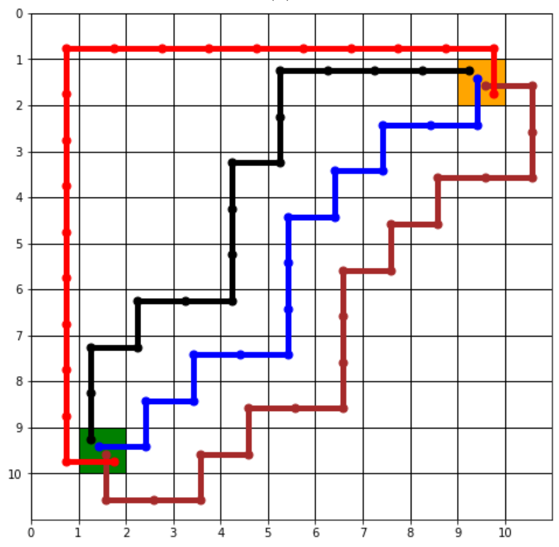
The similarity values for the states traversed by the routes were calculated using Jaccard similarity. The similar states between routes were counted and divided by the total number of states between states (excluding the starting and target states). The similarity value indicates the existence of similar states and the potential for collision between routes. Table VI shows the detailed similarity index values. The average similarity value of the Q-Learning simulation results is 61.17%. It shows potential collision between routes. The highest similarity occurs between routes 2 and 3 *i.e.* 100%. In contrast, the average similarity value of MQL simulations is 0%. It indicates no potential collision between routes.

2. *Scenario 2*

In scenario 2, a simulation is conducted using an area with seven rectangular obstacles. The Q-Learning algorithm simulation resulted four routes and each path consists of 17 states. However, all paths are highly potential colliding. In contrast, the MQL algorithm simulation can produce several safe path variations without potential collisions (see Table VII). The recapitulation graph of the number of the safe path variations is shown in Fig. 9.



(a)



(b)

Fig. 8. An example of the route simulation results in scenario 1

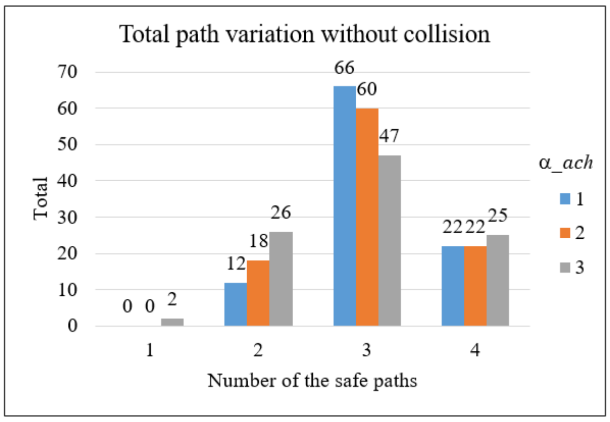


Fig. 9. The number of safe variation path on an 11x11 area with obstacles

TABLE V. DATA FROM THE SIMULATION RESULTS IN SCENARIO 1

Algorithm	Route	State Sequence	State Length	Total Reward	Computation Time (s)
QL	0	(1, 9), (2, 9), (3, 9), (4, 9), (5, 9), (6, 9), (7, 9), (7, 8), (7, 7), (7, 6), (7, 5), (7, 4), (7, 3), (7, 2), (7, 1), (8, 1), (9, 1)	17	983	1.188
	1	(1, 9), (2, 9), (3, 9), (4, 9), (4, 8), (4, 7), (4, 6), (4, 5), (4, 4), (4, 3), (4, 2), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.210
	2	(1, 9), (2, 9), (2, 8), (3, 8), (3, 7), (3, 6), (3, 5), (4, 5), (4, 4), (4, 3), (4, 2), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.130
	3	(1, 9), (2, 9), (2, 8), (3, 8), (3, 7), (3, 6), (3, 5), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.152
MQL	0	(1, 9), (1, 8), (2, 8), (3, 8), (3, 7), (4, 7), (5, 7), (5, 6), (6, 6), (6, 5), (6, 4), (6, 3), (6, 2), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.192
	1	(1, 9), (2, 9), (3, 9), (4, 9), (4, 8), (5, 8), (6, 8), (6, 7), (7, 7), (7, 6), (7, 5), (7, 4), (7, 3), (7, 2), (8, 2), (9, 2), (9, 1)	17	983	1.241
	2	(1, 9), (1, 10), (2, 10), (3, 10), (4, 10), (5, 10), (5, 9), (6, 9), (7, 9), (7, 8), (8, 8), (8, 7), (8, 6), (8, 5), (8, 4), (8, 3), (9, 3), (10, 3), (10, 2), (10, 1), (9, 1)	21	979	1.323
	3	(1, 9), (0, 9), (0, 8), (0, 7), (0, 6), (0, 5), (0, 4), (0, 3), (0, 2), (0, 1), (0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0), (7, 0), (8, 0), (9, 0), (9, 1)	21	979	1.666

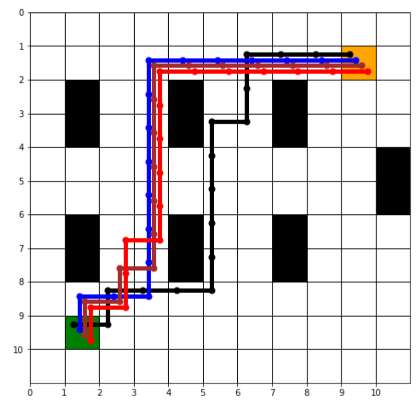
TABLE VI. SIMILARITY STATE OF SCENARIO 1

Algo-rithm	Jaccard similarity ➔ sim (route A, route B) in %						$\overline{x}$
	Sim (0,1)	Sim (0,2)	Sim (0,3)	Sim (1,2)	Sim (1,3)	Sim (2,3)	%
QL	43	36	36	76	76	100	61.17
MQL	0	0	0	0	0	0	0

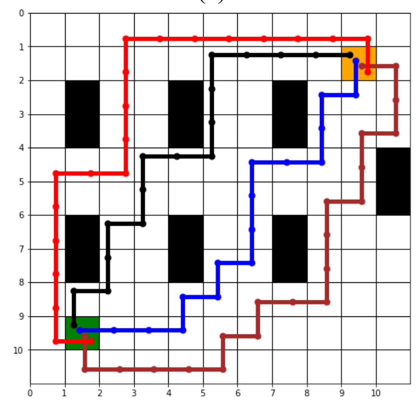
TABLE VII. THE NUMBER OF SAFE PATH IN THE MOTIVATED Q-LEARNING ALGORITHM SIMULATION IN SCENARIO 2

α_{ach}	P	K									
		5	10	15	20	25	30	35	40	45	50
1	0.1	3	3	3	3	4	2	2	2	3	2
	0.2	4	3	4	3	4	3	2	3	4	3
	0.3	3	3	3	2	3	3	3	3	3	2
	0.4	3	3	4	4	2	3	3	3	3	4
	0.5	3	3	4	2	3	3	3	4	4	3
	0.6	3	3	2	3	3	3	3	4	3	3
	0.7	3	4	2	2	3	3	4	3	3	3
	0.8	3	4	4	3	3	3	4	4	3	3
	0.9	3	3	3	4	4	3	3	3	3	3
	1	3	3	3	4	3	4	3	3	3	3
2	0.1	4	4	2	2	2	3	3	3	4	3
	0.2	3	4	3	3	3	4	3	3	4	3
	0.3	4	2	3	3	3	2	2	3	3	3
	0.4	4	3	3	4	3	4	3	4	3	3
	0.5	3	4	3	4	3	3	3	3	3	2
	0.6	3	2	3	4	3	3	2	3	2	3
	0.7	4	3	3	3	3	3	3	3	2	3
	0.8	3	4	3	4	3	3	3	3	2	2
	0.9	4	3	4	2	2	3	3	3	4	3
	1	3	3	3	2	2	4	3	4	3	2
3	0.1	1	2	2	1	2	2	2	3	3	4
	0.2	4	3	4	3	4	2	4	3	2	3
	0.3	4	3	3	4	4	3	4	3	3	3
	0.4	2	3	3	3	2	3	2	4	3	4
	0.5	3	3	3	3	3	4	2	2	3	2
	0.6	3	3	4	4	4	4	3	2	3	2
	0.7	2	2	4	2	3	2	4	3	4	3
	0.8	3	3	3	3	2	4	3	3	3	3
	0.9	2	2	4	2	3	2	4	4	3	3
	1	4	2	3	3	3	4	3	3	3	2

Examples of paths produced by Q-Learning simulation and safe path variations generated in the simulation (with P = 0.7, K = 25, and α_{ach} = 2) are shown in Fig. 10(a) and Fig. 10(b), respectively. Even though the four paths have different lengths, all of them do not have a potential collision. Meanwhile, the average path-finding computation time on the Q-Learning algorithm is 1.117 ± 0.04 seconds (95% confidence level), while the average path-finding computation time on the MQL algorithm is 1.435 ± 0.26 seconds (95% confidence level). The computing time on MQL is slightly longer than Q-Learning. The average reward for each path in the Q-Learning simulation results is 983 while in the MQL is 981. The difference in the average reward is only 2 points. Table VIII shows the scenario 2 simulation results.



(a)



(b)

Fig. 10. An example of the scenario 2 simulation route with seven rectangular obstacles

TABLE VIII. DATA FROM THE SIMULATION RESULTS IN SCENARIO 2

Algorithm	Route	State Sequence	State Length	Total Reward	Computation Time (s)
QL	0	(1, 9), (2, 9), (2, 8), (3, 8), (4, 8), (5, 8), (5, 7), (5, 6), (5, 5), (5, 4), (5, 3), (6, 3), (6, 2), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.173
	1	(1, 9), (1, 8), (2, 8), (3, 8), (3, 7), (3, 6), (3, 5), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.094
	2	(1, 9), (1, 8), (2, 8), (2, 7), (3, 7), (3, 6), (3, 5), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.092
	3	(1, 9), (1, 8), (2, 8), (2, 7), (2, 6), (3, 6), (3, 5), (3, 4), (3, 3), (3, 2), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.108
MQL	0	(1, 9), (1, 8), (2, 8), (2, 7), (2, 6), (3, 6), (3, 5), (3, 4), (4, 4), (5, 4), (5, 3), (5, 2), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.216
	1	(1, 9), (2, 9), (3, 9), (4, 9), (4, 8), (5, 8), (5, 7), (6, 7), (6, 6), (6, 5), (6, 4), (7, 4), (8, 4), (8, 3), (8, 2), (9, 2), (9, 1)	17	983	1.253
	2	(1, 9), (1, 10), (2, 10), (3, 10), (4, 10), (5, 10), (5, 9), (6, 9), (6, 8), (7, 8), (8, 8), (8, 7), (8, 6), (8, 5), (9, 5), (9, 4), (9, 3), (10, 3), (10, 2), (10, 1), (9, 1)	21	979	1.476
	3	(1, 9), (0, 9), (0, 8), (0, 7), (0, 6), (0, 5), (0, 4), (1, 4), (2, 4), (2, 3), (2, 2), (2, 1), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0), (7, 0), (8, 0), (9, 0), (9, 1)	21	979	1.794

The similarity value indicates the existence of similar states and the potential for collision between routes. Table IX shows the detail similarity index values. The average similarity value for the Q-Learning simulation result is 50%. It shows potential for collision between routes. In contrast, the average similarity value for MQL simulations is 0%. It indicates no potential collision among paths.

TABLE IX. SIMILARITY STATE OF SCENARIO 2

Algo-rithm	Jaccard similarity ➔ sim(route A, route B) in %						$\bar{x}$
	Sim (0,1)	Sim (0,2)	Sim (0,3)	Sim (1,2)	Sim (1,3)	Sim (2,3)	%
QL	20	15	15	87	76	87	50
MQL	0	0	0	0	0	0	0

3. *Scenario 3*

Scenario 3 is conducted in an area with randomize obstacles. Q-Learning simulation is shown in Fig. 12a. The MQL results show in Table X. The recapitulation graph of the number of the safe path variations is shown in Fig. 11.

Fig. 12 shows the example routes from QL simulation (Fig. 12(a)) and MQL simulation (Fig. 12a). Example path safe variations in MQL created in the simulation with $P = 0.7$, $K = 30$, and $\alpha_{ach} = 2$). Meanwhile, the average path-finding computation time on the Q-Learning algorithm is 1.472 ± 0.14 seconds (95% confidence level), while the average path-finding computation time on the MQL algorithm is 1.919 ± 0.71 seconds (95% confidence level). Thus, the computing time of MQL algorithm is slightly longer than Q-Learning algorithm. The average reward for each path in the Q-Learning simulation results is 983 while in the MQL is 981. The difference in the average reward is only 2 points. Simulation results (scenario 3) is shown in Table X and Table XI.

Table XII displays the level of similarity between paths in the scenario-3 simulation. The Q-Learning simulation has an average similarity of 68.33%. It suggests a risk of collision between paths. In contrast, the similarity of the MQL is 0%. It indicates that included the motivation model can produce routes without the potential collision.

TABLE X. THE NUMBER OF SAFE PATH IN THE MOTIVATED Q-LEARNING ALGORITHM SIMULATION IN SCENARIO 3

α_{ach}	*P*	*K*									
		5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
1	**0.1**	4	3	4	3	3	3	3	3	3	4
	0.2	3	3	3	4	3	3	3	3	3	2
	0.3	3	4	3	4	4	4	3	4	3	2
	0.4	4	3	4	4	4	3	2	3	3	3
	0.5	4	3	4	3	3	3	4	3	2	3
	0.6	4	3	4	4	3	4	3	3	3	3
	0.7	3	4	3	3	3	3	3	3	2	2
	0.8	4	3	2	4	3	2	3	3	3	2
	0.9	3	4	3	3	3	2	3	3	3	2
	1	3	3	2	3	2	3	3	3	2	3
2	**0.1**	2	3	4	3	3	4	3	4	3	3
	0.2	3	3	4	4	3	3	3	3	3	4
	0.3	3	4	3	4	3	3	2	3	4	3
	0.4	3	4	4	3	3	4	3	4	3	3
	0.5	3	4	4	4	3	4	4	3	3	3
	0.6	4	4	3	3	3	4	3	3	3	4
	0.7	4	3	3	3	3	4	3	3	2	2
	0.8	3	3	3	3	3	3	3	3	2	3
	0.9	3	3	3	3	3	3	3	3	2	3
	1	3	4	3	3	3	2	2	3	3	2
3	**0.1**	2	2	2	1	1	2	2	3	3	2
	0.2	2	2	4	3	3	3	3	3	3	3
	0.3	3	4	3	3	3	3	4	3	4	4
	0.4	3	3	3	4	4	3	4	3	3	3
	0.5	3	4	3	3	3	3	4	3	4	4
	0.6	3	3	3	4	4	3	3	3	4	3
	0.7	4	3	4	3	3	3	4	3	3	3
	0.8	4	3	3	3	3	3	3	2	2	3
	0.9	3	3	2	3	3	3	3	3	2	3
	1	3	4	2	3	3	3	3	2	2	2

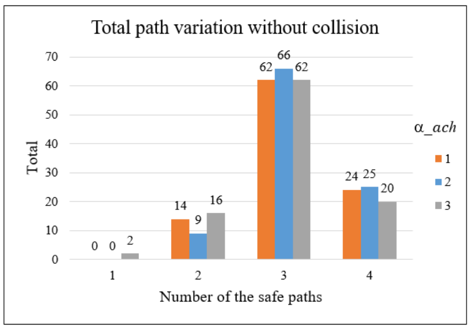


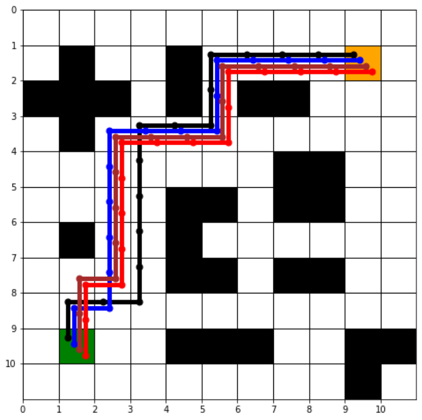
Fig. 11. The number of safe variation path on the area with randomize obstacles

TABLE XI. DATA FROM THE SIMULATION RESULTS IN SCENARIO 3

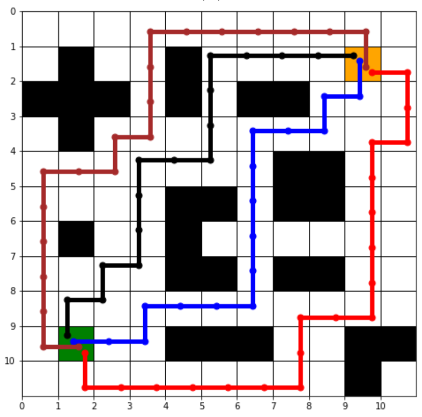
Algorithm	Route	State Sequence	State Length	Total Reward	Computation Time (s)
QL	0	(1, 9), (1, 8), (1, 7), (1, 6), (2, 6), (2, 5), (2, 4), (3, 4), (3, 3), (4, 3), (4, 2), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.435
	1	(1, 9), (1, 8), (1, 7), (1, 6), (1, 5), (1, 4), (2, 4), (3, 4), (3, 3), (4, 3), (4, 2), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.366
	2	(1, 9), (1, 8), (1, 7), (1, 6), (1, 5), (1, 4), (2, 4), (3, 4), (3, 3), (4, 3), (4, 2), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.408
	3	(1, 9), (1, 8), (1, 7), (1, 6), (1, 5), (1, 4), (2, 4), (3, 4), (3, 3), (4, 3), (4, 2), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.679
MQL	0	(1, 9), (1, 8), (2, 8), (3, 8), (3, 7), (3, 6), (3, 5), (3, 4), (3, 3), (4, 3), (4, 2), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)	17	983	1.482
	1	(1, 9), (2, 9), (3, 9), (4, 9), (4, 8), (5, 8), (5, 7), (6, 7), (6, 6), (7, 6), (7, 5), (7, 4), (7, 3), (7, 2), (8, 2), (9, 2), (9, 1)	17	983	1.423
	2	(1, 9), (1, 10), (2, 10), (3, 10), (4, 10), (5, 10), (6, 10), (7, 10), (8, 10), (9, 10), (9, 9), (10, 9), (10, 8), (10, 7), (10, 6), (10, 5), (10, 4), (10, 3), (10, 2), (10, 1), (9, 1)	21	979	1.792
	3	(1, 9), (0, 9), (0, 8), (0, 7), (0, 6), (0, 5), (0, 4), (0, 3), (0, 2), (0, 1), (0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (6, 0), (7, 0), (8, 0), (9, 0), (9, 1)	21	979	2.980

TABLE XII. SIMILARITY STATE OF SCENARIO 3

Algo-rithm	Jaccard similarity ➔ sim(route A, route B) in %						$\overline{x}$
	Sim (0,1)	Sim (0,2)	Sim (0,3)	Sim (1,2)	Sim (1,3)	Sim (2,3)	%
QL	50	43	43	87	87	100	68.33
MQL	0	0	0	0	0	0	0



(a)



(b)

Fig. 12. An example of the scenario-3 simulation route with randomize shaped obstacles

The simulation results show that the MQL algorithm has succeeded in generating 2 to 4 safe path variations in the same area and goal. The Jaccard similarity every between two safe path variations is 0%. It indicates that these safe paths are not potentially collision. However, the computational time of MQL is significantly longer than Q-Learning both in areas without obstacles and in areas with obstacles. The average difference in the total rewards collected by each MQL path (in this case) is two points smaller than the Q-Learning.

In principal, the simulation results show that MQL can be applied to several robots with a same task, operating in the same area. However, the algorithm can only provide a maximum of four path variations, due to the assumption that the robot can only move forward, backward, left and right. In the real implementation, the robot may have more flexibility to move to other directions. Further study is required to analyze whether this additional flexibility will result in more path variations. In addition, the parameters in achievement motivation may need to be re-evaluated for this purpose.

IV. CONCLUSION

We have presented the MQL algorithm that utilizes a motivation achievement to find safe path variations in an unknown environment. The achievement motivation succeeded in influencing the reward value in the state that is used as a path before. This update reward makes the state as an obstacle so that the MQL will avoid that state and find other states for a new route and avoid collision with the last paths. The simulation results show that the MQL algorithm generated 2 to 4 safe path variations (Jaccard similarity = 0%). On the contrary, the Q-Learning algorithm tends to produce the same path for each robot so that is potential collisions. However, the computation time of MQL is slightly longer than Q-Learning. In principal, the simulation results show that MQL can be implemented to multi robots with a same goal in the same area. We hope MQL can solve multi-robot path planning problems by reducing the possibility of collisions as well as decreasing the problem of queues. It can only provide maximum 4 path safe variations because the robot can only move in 4 directions i.e. forward, backward, left and right. Further study is needed to add flexibility robot movement to other directions (i.e. forward left, forward right, backward left and backward right) and to analyze whether this additional flexibility will result in more path variations.

In addition, the achievement motivation parameters need to be re-evaluated for this purpose and simulated in wide area.

ACKNOWLEDGEMENT

This work is partly supported by the Ministry of Education, Culture, Research, and Technology of Indonesia through the Directorate of Research and Innovation, IPB University, research grant *Penelitian Dasar Unggulan Perguruan Tinggi* (PDUPT) No. 001/E5/PG.02.00.PL/2023 jo 15853/IT3.D10/PT.01.02/P/T/2023.

REFERENCES

[1] S. S. Valle and J. Kienzle, "Agriculture 4.0 - Agricultural robotics and automated equipment for sustainable crop production," in *Integrated Crop Management*, vol. 24, no. 24, p. 40, 2020.

[2] M. Javaid, A. Haleem, R. P. Singh, and R. Suman, "Enhancing smart farming through the applications of Agriculture 4.0 technologies," *Int. J. Intell. Networks*, vol. 3, pp. 150–164, 2022, doi: 10.1016/j.ijin.2022.09.004.

[3] M. B. Ahsan, G. Leifeng, F. Mohammad, S. Azam, B. Xu, and S. J. Rayhan, "Barriers, Challenges, and Requirements for ICT Usage among Sub-Assistant Agricultural Officers in Bangladesh : Toward Sustainability in Agriculture," *Sustainability*, vol. 15, no. 782, pp. 1–29, 2023, doi: 10.3390/su15010782.

[4] S. Ruzzante, R. Labarta, and A. Bilton, "Adoption of agricultural technology in the developing world: A meta-analysis of the empirical literature," *World Development*, vol. 146, p. 105599, 2021, doi: 10.1016/j.worlddev.2021.105599.

[5] L. F. P. Oliveira, A. P. Moreira, and M. F. Silva, "Advances in agriculture robotics: A state-of-the-art review and challenges ahead," *Robotics*, vol. 10, no. 2, pp. 1–31, 2021, doi: 10.3390/robotics10020052.

[6] L. C. Santos, F. N. Santos, E. J. Solteiro Pires, A. Valente, P. Costa and S. Magalhães, "Path Planning for ground robots in agriculture: a short review," *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 61-66, 2020, doi: 10.1109/ICARSC49921.2020.9096177.

[7] S. Chakraborty, D. Elangovan, P. L. Govindarajan, M. F. ELnaggar, M. M. Alrashed, and S. Kamel, "A Comprehensive Review of Path Planning for Agricultural Ground Robots," *Sustain.*, vol. 14, no. 15, pp. 1–19, 2022, doi: 10.3390/su14159156.

[8] C. Cheng, J. Fu, H. Su, and L. Ren, "Recent Advancements in Agriculture Robots: Benefits and Challenge," *Machines*, vol. 11, no. 48, pp. 1–24, 2023, doi: https://doi.org/10.3390/machines11010048.

[9] Q. Yang, X. Du, Z. Wang, Z. Meng, Z. Ma, and Q. Zhang, "A review of core agricultural robot technologies for crop productions," *Comput. Electron. Agric.*, vol. 206, p. 107701, 2023, doi: 10.1016/j.compag.2023.107701.

[10] P. Gonzalez-de-Santos, R. Fernández, D. Sepúlveda, E. Navas, L. Emmi, and M. Armada, "Field Robots for Intelligent Farms—Inhering Features from Industry," *Agronomy*, vol. 10, no. 11, p. 1638, 2020, doi: 10.3390/agronomy10111638.

[11] V. Marinoudi, C. G. Sørensen, S. Pearson, and D. Bochtis, "Robotics and labour in agriculture. A context consideration," *Biosyst. Eng.*, vol. 184, pp. 111–121, 2019, doi: 10.1016/j.biosystemseng.2019.06.013.

[12] X. Gao *et al.*, "Review of wheeled mobile robots' navigation problems and application prospects in agriculture," *IEEE Access*, vol. 6, pp. 49248–49268, 2018, doi: 10.1109/ACCESS.2018.2868848.

[13] J. Chen, H. Qiang, J. Wu, G. Xu, and Z. Wang, "Navigation path extraction for greenhouse cucumber-picking robots using the prediction-point Hough transform," *Comput. Electron. Agric.*, vol. 180, p. 105911, 2021, doi: 10.1016/j.compag.2020.105911.

[14] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Eng. Sci. Technol. an Int. J.*, vol. 40, p. 101343, 2023, doi: 10.1016/j.jestch.2023.101343.

[15] I. Nizar and M. Mestari, "Mobile Robot Autonomous Navigation: A Path Planning Approach," *IFAC-PapersOnLine*, vol. 55, no. 12, pp. 610–615, 2022, doi: 10.1016/j.ifacol.2022.07.379.

[16] Y. Bai, B. Zhang, N. Xu, J. Zhou, J. Shi, and Z. Diao, "Vision-based navigation and guidance for agricultural autonomous vehicles and robots: A review," *Comput. Electron. Agric.*, vol. 205, p. 107584, 2023, doi: 10.1016/j.compag.2022.107584.

[17] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Rob. Auton. Syst.*, vol. 86, pp. 13–28, 2016, doi: 10.1016/j.robot.2016.08.001.

[18] M. S. Abed, O. F. Lutfy, and Q. F. Al-Doori, "A Review on Path Planning Algorithms for Mobile Robots," *Eng. Technol. J.*, vol. 39, no. 5, pp. 804–820, 2021, doi: 10.30684/etj.v39i5A.1941.

[19] H. Tian, "Research on Robot Path Planning Based on Improved Ant Colony Algorithm," *Int. J. Comput. Sci. Math.*, vol. 13, no. 1, pp. 80–92, 2021, doi: 10.1088/1742-6596/1992/3/032050.

[20] L. Wu, X. Huang, J. Cui, C. Liu, and W. Xiao, "Modified adaptive ant colony optimization algorithm and its application for solving path planning of mobile robot," *Expert Syst. Appl.*, vol. 215, p. 119410, 2023, doi: 10.1016/j.eswa.2022.119410.

[21] C. Liu *et al.*, "An improved heuristic mechanism ant colony optimization algorithm for solving path planning," *Knowledge-Based Syst.*, vol. 271, p. 110540, 2023, doi: 10.1016/j.knosys.2023.110540.

[22] M. Morin, I. Abi-Zeid, and C.-G. Quimper, "Ant colony optimization for path planning in search and rescue operations," *Eur. J. Oper. Res.*, vol. 305, no. 1, pp. 53–63, 2023, doi: 10.1016/j.ejor.2022.06.019.

[23] C. Miao, G. Chen, C. Yan, and Y. Wu, "Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm," *Comput. Ind. Eng.*, vol. 156, p. 107230, 2021.

[24] D. Di Caprio, A. Ebrahimnejad, H. Alrezaamiri, and F. J. Santos-Arteaga, "A novel ant colony algorithm for solving shortest path problems with fuzzy arc weights," *Alexandria Eng. J.*, vol. 61, no. 5, pp. 3403–3415, 2022, doi: 10.1016/j.aej.2021.08.058.

[25] H.-J. Wang, Y. Fu, Z.-Q. Zhao, and Y.-J. Yue, "An Improved Ant Colony Algorithm of Robot Path Planning for Obstacle Avoidance," *J. Robot.*, vol. 2019, pp. 1–8, 2019, doi: 10.1155/2019/6097591.

[26] X. Pu, C. Xiong, L. Ji, and L. Zhao, "3D path planning for a robot based on improved ant colony algorithm," *Evol. Intell.*, pp. 1-11, 2020, doi: 10.1007/s12065-020-00397-6.

[27] Y. Xue, Y. Chen, Z. Ding, X. Huang, and D. Xi, "Robot path planning based on improved ant colony algorithm," *2021 Power System and Green Energy Conference (PSGEC)*, pp. 129-133, 2021, doi: 10.1109/PSGEC51302.2021.9541872.

[28] T. Wang, L. Zhao, Y. Jia, and J. Wang, "Robot Path Planning Based on Improved Ant Colony Algorithm," *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, pp. 70–76, 2018, doi: 10.1109/WRC-SARA.2018.8584217.

[29] R. Sarkar, D. Barman, and N. Chowdhury, "Domain knowledge based genetic algorithms for mobile robot path planning having single and multiple targets," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 7, pp. 4269–4283, 2022, doi: 10.1016/j.jksuci.2020.10.010.

[30] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Comput. Sci.*, vol. 127, pp. 180–189, 2018, doi: 10.1016/j.procs.2018.01.113.

[31] K. Hao, J. Zhao, Z. Li, Y. Liu, and L. Zhao, "Dynamic path planning of a three-dimensional underwater AUV based on an adaptive genetic algorithm," *Ocean Eng.*, vol. 263, p. 112421, 2022, doi: 10.1016/j.oceaneng.2022.112421.

[32] W. Rahmaniar and A. E. Rakhmania, "Mobile Robot Path Planning in a Trajectory with Multiple Obstacles Using Genetic Algorithms," *J. Robot. Control*, vol. 3, no. 1, pp. 1–7, 2022, doi: 10.18196/jrc.v3i1.11024.

[33] M. Fan, J. He, S. Ding, Y. Ding, M. Li, and L. Jiang, "Research and implementation of multi-robot path planning based on genetic algorithm," *2021 5th International Conference on Automation, Control and Robots (ICACR)*, pp. 140–144, 2021, doi: 10.1109/ICACR53472.2021.9605194.

[34] A. López-González, J. A. Meda Campaña, E. G. Hernández Martínez, and P. P. Contro, "Multi robot distance based formation using Parallel Genetic Algorithm," *Appl. Soft Comput.*, vol. 86, p. 105929, 2020, doi: 10.1016/j.asoc.2019.105929.

[35] A. Al Hilli, M. Al-Ibadi, A. M. Alfadhel, S. H. Abdulshaheed, and A. H. Hadi, "Optimal path finding in stochastic quasi-dynamic environments using particle swarm optimization," *Expert Syst. Appl.*,

vol. 186, p. 115706, 2021, doi: 10.1016/j.eswa.2021.115706.

[36] H. S. Dewang, P. K. Mohanty, and S. Kundu, “A Robust Path Planning for Mobile Robot Using Smart Particle Swarm Optimization,” *Procedia Comput. Sci.*, vol. 133, pp. 290–297, 2018, doi: 10.1016/j.procs.2018.07.036.

[37] P. B. Fernandes, R. C. L. Oliveira, and J. V. Fonseca Neto, “Trajectory planning of autonomous mobile robots applying a particle swarm optimization algorithm with peaks of diversity,” *Appl. Soft Comput.*, vol. 116, p. 108108, 2022, doi: 10.1016/j.asoc.2021.108108.

[38] S. Lin, A. Liu, J. Wang, and X. Kong, “An intelligence-based hybrid PSO-SA for mobile robot path planning in warehouse,” *J. Comput. Sci.*, vol. 67, p. 101938, 2023, doi: 10.1016/j.jocs.2022.101938.

[39] B. Alkhlidi, A. T. Abdulsadda, and A. Al Bakri, “Optimal robotic path planning using intelligent search algorithms,” *J. Robot. Control*, vol. 2, no. 6, pp. 519–526, 2021, doi: 10.18196/jrc.26132.

[40] P. K. Das and P. K. Jena, “Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators,” *Appl. Soft Comput. J.*, vol. 92, p. 106312, 2020, doi: 10.1016/j.asoc.2020.106312.

[41] M. Samadi Gharajeh and H. B. Jond, “An intelligent approach for autonomous mobile robots path planning based on adaptive neuro-fuzzy inference system,” *Ain Shams Eng. J.*, vol. 13, no. 1, p. 101491, 2022, doi: 10.1016/j.asej.2021.05.005.

[42] C. Ntakolia, S. Moustakidis, and A. Siouras, “Autonomous path planning with obstacle avoidance for smart assistive systems,” *Expert Syst. Appl.*, vol. 213, p. 119049, 2023, doi: 10.1016/j.eswa.2022.119049.

[43] R. Zhen, P. Lv, Z. Shi, and G. Chen, “A novel fuzzy multi-factor navigational risk assessment method for ship route optimization in costal offshore wind farm waters,” *Ocean Coast. Manag.*, vol. 232, p. 106428, 2023, doi: 10.1016/j.ocecoaman.2022.106428.

[44] T. Shen and J. Zhai, "Reactive Obstacle Avoidance Strategy Based on Fuzzy Neural Network and Arc Trajectory," *2019 Chinese Automation Congress (CAC)*, pp. 4792-4796, 2019, doi: 10.1109/CAC48633.2019.8996374.

[45] N. Awad, A. Lasheen, M. Elnaggar, and A. Kamel, “Model predictive control with fuzzy logic switching for path tracking of autonomous vehicles,” *ISA Trans.*, vol. 129, pp. 193–205, 2022, doi: 10.1016/j.isatra.2021.12.022.

[46] N. Rinanto, I. Marzuqi, A. Khumaidi, and S. T. Sarena, “Obstacle Avoidance using Fuzzy Logic Controller on Wheeled Soccer Robot,” *J. Ilm. Tek. Elektro Komput. dan Inform.*, vol. 5, no. 1, pp. 26–35, 2019, doi: 10.26555/jiteki.v5i1.13298.

[47] O. O. Martins, A. A. Adekunle, O. M. Olaniyan, and B. O. Bolaji, “An Improved multi-objective a-star algorithm for path planning in a large workspace: Design, Implementation, and Evaluation,” *Sci. African*, vol. 15, p. e01068, 2022, doi: 10.1016/j.sciaf.2021.e01068.

[48] L. Pasandi, M. Hooshmand, and M. Rahbar, “Modified A* Algorithm integrated with ant colony optimization for multi-objective route-finding; case study: Yazd,” *Appl. Soft Comput.*, vol. 113, p. 107877, 2021, doi: 10.1016/j.asoc.2021.107877.

[49] J. P. Vasconez *et al.*, “Comparison of path planning methods for robot robot navigation in simulated agricultural environments,” *The 1st International Workshop on Human-Centric Innovation and Computational Intelligence (IWHICI 2023)*, vol. 220, pp. 898–903, 2023, doi: 10.1016/j.procs.2023.03.122.

[50] X. Zhong, J. Tian, H. Hu, and X. Peng, “Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 99, no. 1, pp. 65–77, 2020, doi: 10.1007/s10846-019-01112-z.

[51] S. Hosseininejad and C. Dadkhah, “Mobile robot path planning in dynamic environment based on cuckoo optimization algorithm,” *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, pp. 1–13, 2019, doi: 10.1177/1729881419839575.

[52] J. Yu, Y. Wang, X. Ruan, G. Zuo, and C. Li, “AGV multi-objective path planning method based on improved cuckoo algorithm,” *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pp. 556–561, 2019, doi: 10.1109/IAEAC47372.2019.8997687.

[53] J. Wang, X. Shang, T. Guo, J. Zhou, S. Jia and C. Wang, "Optimal Path Planning Based on Hybrid Genetic-Cuckoo Search Algorithm," *2019 6th International Conference on Systems and Informatics (ICSAI)*, pp. 165-169, 2019, doi: 10.1109/ICSAI48974.2019.9010519.

[54] L. Zhao, F. Wang, and Y. Bai, “Route planning for autonomous vessels based on improved artificial fish swarm algorithm,” *Ships Offshore Struct.*, vol. 18, no. 6, pp. 897-906, 2022, doi: 10.1080/17445302.2022.2081423.

[55] S. Kumar and A. Sikander, “A modified probabilistic roadmap algorithm for efficient mobile robot path planning,” *Eng. Optim.*, vol. 55, no. 9, pp. 1616-1634, 2022, doi: 10.1080/0305215X.2022.2104840.

[56] J. C. Mohanta and A. Keshari, “A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation,” *Appl. Soft Comput.*, vol. 79, pp. 391–409, 2019, doi: 10.1016/j.asoc.2019.03.055.

[57] M. S. Das, S. Sanyal, and S. Mandal, “Navigation of Multiple Robots in Formative Manner in an Unknown Environment using Artificial Potential Field Based Path Planning Algorithm,” *Ain Shams Eng. J.*, vol. 13, no. 5, p. 101675, 2022, doi: 10.1016/j.asej.2021.101675.

[58] Z. Wu, J. Dai, B. Jiang, and H. R. Karimi, “Robot path planning based on artificial potential field with deterministic annealing,” *ISA Trans.*, vol. 112, p. 106640, 2023, doi: 10.1016/j.ast.2021.106640.

[59] A. Lazarowska, “Discrete Artificial Potential Field Approach to Mobile Robot Path Planning,” *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 334–337, 2019, doi: 10.1016/j.ifacol.2019.08.083.

[60] F. Sui, X. Tang, Z. Dong, X. Gan, P. Luo, and J. Sun, “ACO+PSO+A*: A bi-layer hybrid algorithm for multi-task path planning of an AUV,” *Comput. Ind. Eng.*, vol. 175, p. 108905, 2023, doi: 10.1016/j.cie.2022.108905.

[61] B. Sahu, P. Kumar Das, and R. Kumar, “A Modified Cuckoo Search Algorithm implemented with SCA and PSO for Multi-robot Cooperation and Path Planning,” *Cogn. Syst. Res.*, vol. 79, pp. 24-42, 2023, doi: 10.1016/j.cogsys.2023.01.005.

[62] F. Gul, I. Mir, D. Alarabiat, H. M. Alabool, L. Abualigah, and S. Mir, “Implementation of bio-inspired hybrid algorithm with mutation operator for robotic path planning,” *J. Parallel Distrib. Comput.*, vol. 169, pp. 171–184, 2022, doi: 10.1016/j.jpdc.2022.06.014.

[63] D. Zhang, Y. Yin, R. Luo, and S. Zou, “Hybrid IACO-A*-PSO optimization algorithm for solving multiobjective path planning problem of mobile robot in radioactive environment,” *Prog. Nucl. Energy*, vol. 159, p. 104651, 2023, doi: 10.1016/j.pnucene.2023.104651.

[64] X. Pu, C. Xiong and L. Zhao, "Path Planning for Robot Based on IACO-SFLA Hybrid Algorithm," *2020 Chinese Control And Decision Conference (CCDC)*, pp. 4886-4893, 2020, doi: 10.1109/CCDC49329.2020.9164671.

[65] G. Kulathunga, “A Reinforcement Learning based Path Planning Approach in 3D Environment,” *Procedia Comput. Sci.*, vol. 212, pp. 152–160, 2021, doi: 10.1016/j.procs.2022.10.217.

[66] F. Gismondi, C. Possieri, and A. Tornambe, “A solution to the path planning problem via algebraic geometry and reinforcement learning,” *J. Franklin Inst.*, vol. 359, no. 2, pp. 1732–1754, 2022, doi: 10.1016/j.jfranklin.2021.12.003.

[67] X. Zhang, S. Xia, X. Li, and T. Zhang, “Multi-objective particle swarm optimization with multi-mode collaboration based on reinforcement learning for path planning of unmanned air vehicles,” *Knowledge-Based Syst.*, vol. 250, p. 109075, 2022, doi: 10.1016/j.knosys.2022.109075.

[68] E. S. Low, P. Ong, C. Y. Low, and R. Omar, “Modified Q-learning with distance metric and virtual target on path planning of mobile robot,” *Expert Syst. Appl.*, vol. 199, p. 117191, 2022, doi: 10.1016/j.eswa.2022.117191.

[69] L. D. Hanh and V. D. Cong, “Path following and avoiding obstacle for mobile robot under dynamic environments using reinforcement learning,” *J. Robot. Control*, vol. 4, no. 2, pp. 157–164, 2023, doi: 10.18196/jrc.v4i2.17368.

[70] E. S. Low, P. Ong, and K. C. Cheah, “Solving the optimal path planning of a mobile robot using improved Q-Learning,” *Rob. Auton. Syst.*, vol. 115, pp. 143–161, 2019, doi: 10.1016/j.robot.2019.02.013.

[71] S. Gu, “An algorithm for path planning based on improved Q-Learning,” in *The Genetic and Evolutionary Computing*, pp. 20–29, 2019, doi: 10.1007/978-981-15-3308-2_3.

[72] S. Gu and G. Mao, “An improved Q-Learning algorithm for path

planning in maze environments," *Intelligent Systems and Applications*, vol. 1251, no. 2, pp. 545–557, 2020, doi: 10.1007/978-3-030-55187-2_40.

[73] M. Zhao, H. Lu, S. Yang, and F. Guo, "The experience-memory Q-Learning algorithm for robot path planning in unknown environment," *IEEE Access*, vol. 8, pp. 47824–47844, 2020, doi: 10.1109/ACCESS.2020.2978077.

[74] C. Yan and X. Xiang, "A Path Planning Algorithm for UAV Based on Improved Q-Learning," in *2nd International Conference on Robotics and Automation Sciences (ICRAS)*, pp. 46–50, 2018, doi: 10.1109/ICRAS.2018.8443226.

[75] T. Zhang, X. Huo, S. Chen, B. Yang and G. Zhang, "Hybrid Path Planning of A Quadrotor UAV Based on Q-Learning Algorithm," *2018 37th Chinese Control Conference (CCC)*, pp. 5415-5419, 2018, doi: 10.23919/ChiCC.2018.8482604.

[76] H. Hidayat, A. Buono, K. Priandana, and S. Wahjuni, "Modified Q-Learning algorithm for mobile robot real-time path planning using reduced states," *RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 7, no. 3, pp. 628–636, 2023, doi: 10.29207/resti.v7i3.4949.

[77] K. E. Merrick and K. Shafi, "Achievement, affiliation, and power: Motive profiles for artificial agents," *Adapt. Behav.*, vol. 19, no. 1, pp. 40–62, 2011, doi: 10.1177/1059712310395953.

[78] M. K. D. Hardhienata, V. Ugrinovskii, and K. E. Merrick, "Task allocation under communication constraints using motivated Particle Swarm Optimization," in *IEEE Congress on Evolutionary Computation (CEC)*, pp. 3135–3142, 2014, doi: 10.1109/CEC.2014.6900560.

[79] M. K. D. Hardhienata, K. E. Merrick and V. Ugrinovskii, "Task allocation in multi-agent systems using models of motivation and leadership," *2012 IEEE Congress on Evolutionary Computation*, pp. 1-8, 2012, doi: 10.1109/CEC.2012.6256114.

[80] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An introduction*. MIT Press, 2018.

[81] C. J. C. H. Watkins, "Technical Note Q-Learning," *Mach. Learn.*, vol. 8, pp. 279–292, 1992, doi: 10.1109/ICCC49849.2020.9238991.

[82] P. Jaccard, "The distribution of the flora in the Alpine Zone," *New Phytol.*, vol. 11, no. 2, pp. 37–50, 1912, doi: 10.1111/j.1469-8137.1912.tb05611.x.

[83] H. Seifoddini and M. Djassemi, "The production data-based similarity coefficient versus Jaccard's similarity coefficient," *Comput. Ind. Eng.*, vol. 21, no. 1–4, pp. 263–266, 1991, doi: 10.1016/0360-8352(91)90099-R.