Accredited Ranking SINTA 2
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online on: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 7 No. 3 (2023) 628 - 636 ISSN Media Electronic: 2580-0760

Modified Q-Learning Algorithm for Mobile Robot Real-Time Path Planning using Reduced States

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
[1]Department of Computer Engineering, Faculty of Engineering and Computer Science, Universitas Komputer Indonesia
[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

Abstract

Path planning is an essential algorithm in any autonomous mobile robot, including agricultural robots. One of the reinforcement learning methods that can be used for mobile robot path planning is the Q-Learning algorithm. However, the conventional Q-learning method explores all possible robot states in order to find the most optimum path. Thus, this method requires extensive computational cost especially when there are considerable grids to be computed. This study modified the original Q-Learning algorithm by removing the impassable area, so that these areas are not considered as grids to be computed. This modified Q-Learning method was simulated as path finding algorithm for autonomous mobile robot operated at the Agribusiness and Technology Park (ATP), IPB University. Two simulations were conducted to compare the original Q-Learning method and the modified Q-Learning method. The simulation results showed that the state reductions in the modified Q-Learning method can lower the computation cost to 50.71% *from the computation cost of the original Q-Learning method, that is, an average computation time of* 25.74*s as compared to* 50.75*s, respectively. Both methods produce similar number of states as the robot's optimal path, i.e. 56 states, based on the reward obtained by the robot while selecting the path. However, the modified Q-Learning algorithm is capable of finding the path to the destination point with a minimum learning rate parameter value of 0.2 when the discount factor value is 0.9.*

Keywords: agricultural robot; mobile robot; path planning; Q-Learning algorithm.

1. Introduction

The rapid development of agricultural technology is moving towards the Agriculture 4.0 paradigm, where digitization, automation, and artificial intelligence play a crucial role in crop production [1]–[4]. The advancement in digital technology, automation, and artificial intelligence is transforming agricultural management processes, leading to increased yield and sustainable agricultural production. However, this shift from manual technology to mechanical and automated devices presents both challenges and opportunities, such as the use of agricultural robots [5]–[8]. Agricultural robots are utilized to achieve sustainable production results, reduce high labor costs, and minimize the risk of accidents for farm workers.

The use of agricultural robots has been implemented in various tasks, including planting, harvesting, monitoring, spraying, and pruning. In planting, the robot aims to plant crops precisely for optimal growth. In harvesting, robots are used to carefully and accurately harvest crops to ensure high-quality results. Monitoring plants with robots can help in managing the growth process. In spraying, robots can perform faster and more efficiently than humans. Pruning can also be performed with precision using robots. The performance of agricultural robots depends on both the type of crops and the task being performed. For instance, some studies have focused on general tasks for robots in agriculture (such as [9], [10]), while others have focused on specific tasks such as harvesting [11], [12]. Additionally, research has also been conducted to find solutions for mobile robot navigation problems in agriculture [13], [14].

In the field of agricultural robots, autonomous navigation is a crucial aspect, as discussed in [13], [15]. The process of autonomous navigation involves four key requirements: localization, mapping, motion control, and path planning. Path planning involves finding a sequence of translational and rotational movements from the starting point to the destination while avoiding obstacles in the working space [16].

Accepted: 18-02-2023 | Received in revised: 01-05-2023 | Published: 02-06-2023

Robotic path planning research is a significant area of study within robotics, particularly in the context of mobile robots operating in agricultural environments [6], [17]. In terms of topology, route planning refers to calculating the optimal sequence of visiting nodes, whereas path planning involves determining a collision-free path between a specified start and target point, taking into account topology, geometry, or trajectories.

In recent years, there has been a lot of research in the field of robot path planning. Many intelligent optimization algorithms have been proposed to provide the robot with the ability to optimize its path through multiple iterations. For example, algorithms such as the ant colony algorithm, genetic algorithm, particle swarm optimization (PSO), and others inspired by natural phenomena or biological groups have been used to improve the optimization of the robot's path. In studies such as [18] by Wu et al. and [19] by Wang, the improved ant colony algorithm was used to avoid obstacles and find the optimal path. Sarkar et al. in [20] and Hao et al. in [21] proposed improved genetic algorithms to solve the path planning problem by generating an effective path. Dewang et al. in [22], Fernandes et al. in [23], and Hilli et al. in [24] used PSO algorithms to avoid obstacles and reach the target more quickly than conventional PSO algorithms. In addition, other algorithms such as fuzzy algorithms [25]–[27], A* algorithms [28], improved artificial fish swarm algorithms [29], modified probabilistic roadmap algorithms [30], artificial potential field algorithms [31], and hybrid algorithms [32]–[34] have also been applied in robot path planning. Additionally, in recent years, reinforcement learning has been used to solve path planning problems, as seen in studies [35]–[38].

One of the currently used reinforcement learning algorithms in path planning is the Q-Learning algorithm. It is a classical reinforcement learning algorithm that has been applied in several studies for path planning [38]–[44]. These studies have shown that the Q-Learning algorithm is able to produce optimal path planning for mobile robots. The Q-Learning algorithm is often used for path planning on moving robots [39]–[41]. However, its traditional approach of exploring all states can lead to long computation times, especially when dealing with a large number of grids in unknown environment [39]–[42]. For example, the study results cited in [39] claim that their improved Q-learning method is able to reduce computing time required to achieve convergence, but the resulting route is not as smooth as with conventional Q-learning. This study aims to modify the original Q-Learning algorithm by eliminating inaccessible areas, thereby reducing the number of states to be calculated. Different from the previous studies, it is expected that the modification proposed in this paper maintains the ability of the Q-learning algorithm in finding the optimum path while reducing its computation time. By eliminating states, it is expected that smaller parameter values can be used to find the path from the starting point to the target point. Furthermore, this study also analyzes the effects of the learning rate and discount factor parameters to the success of the Q-learning algorithm in finding the optimum path. In this paper, the modified Q-Learning algorithm is tested as a pathfinding algorithm for autonomous robots at the Agribusiness and Technology Park (ATP) IPB University, and the results are compared with the conventional Q-Learning methods.

2. Research Methods

2.1 Q-Learning Algorithm

The Q-Learning algorithm is a type of reinforcement learning (RL) method, first introduced by Watkins [45]. It uses the concepts of rewards and penalties to explore an unstructured environment. In this method, important terms include state, action, agent, reward, and penalty. The agent, such as a mobile robot, moves through the environment. The state (*S*) represents the position of the agent in the environment, and the action (*A*) represents the movement made by the agent from one state to another. Rewards are positive values that increase the Q value for each correct action taken by the agent in a particular state, while penalties are negative values that decrease the Q value for each incorrect action.

The agent gains experience through exploration and exploitation. Exploration involves randomly selecting actions in the early stages of the learning process, allowing the agent to visit all state-action pairs in the environment without considering the current state. Exploitation, on the other hand, involves the use of the agent's acquired knowledge to select actions that maximize the reward from the current state. In Q-Learning, it is not necessary to have a complete model of the environment, as the transition probability matrix for all states and actions can be learned over time.

The Q-Learning algorithm uses a two-dimensional Q table to store the Q values for each state and potential action. The algorithm selects the action with the highest benefit by searching the Q table. This method has been proven to be effective in medium-sized applications, such as robot path planning and chess games. The equation (1) below is the Q-Learning equation by Watkins [46] that updates 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 S_t represents a state or the position of agent (A) at time t. The A_t represents the action taken by the agent in state S_t. The reward R_{t+1} is the value received after the agent performs action A_{t+1} in state S_t. The $Q(S_t, A_t)$ is the value of Q generated by action A_t in state S_t. The gamma (γ) or discount factor is a parameter that determines the importance of future rewards. The value of γ ranges from 0 to 1. If γ is close to zero, the agent will only consider immediate rewards. If γ is close to

DOI: https://doi.org/10.29207/resti.v7i3.4949

one, the agent will consider future rewards more heavily and be willing to delay immediate rewards. The alpha (α) or learning rate is a parameter that sets the speed at which convergence is achieved. The value of α can range from 0 to 1. If α is close to one, the agent will make aggressive adjustments to the Q value, leading to fluctuating results that may not converge. On the other hand, if α is close to zero, adjustments will be slower and it will take more time to converge. The Q-Learning algorithm is described in procedure 1 [46].

Procedure 1. Description of The Q-Learning algorithm

Q-Learning Algorithm

Initiate $Q(s,a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$
Looping for each episode:
Initiate S
  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

Initially, the Q-table is initialized with all values of $Q(s,a)$ set to zero. The variables 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 $\mathcal{A}(s)$. The initial state S is then determined. In the looping section, the Q value is updated.

The process of the loop starts by selecting an action (A) to be performed on the current state (S). The action selection is based on the policy derived from Q. The next step is to take action A and observe the resulting reward (R) and the next state (S'). The value of Q in the Q-table is then updated using Equation (1). Finally, the current state (S) is updated to the value of the next state (S'). The looping process continues until the current state is the target state.

The Q-Learning process is depicted in Figure 1. At the starting point, the agent is in state S_t, which is labeled as the initial state (n). The agent will then choose one of the available actions, A, for state S_t based on the policy derived from Q $\gamma max_a Q(S_{t+1}, a)$) where a is an element of all possible actions of that state. This action will transition the agent to the next state, S_{t+1}, and yield a reward value of R_{t+1}. This process will continue until a converged Q value is achieved.

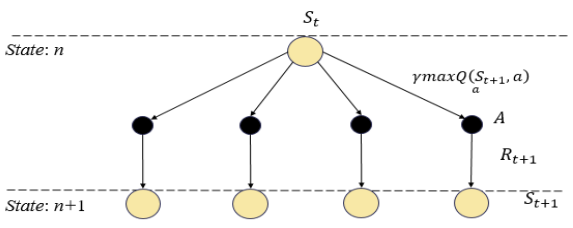

Figure 1. Illustration of the Q-Learning process

2.2 Modification of Q-Learning Algorithm

In this study, a modification to the original Q-Learning algorithm is done by reducing the number of possible states to be computed. In the original Q-learning algorithm, all the robot's working area are divided into some grids and considered in the Q-learning computation. Grids that can be traversed by the robot have higher rewards, whereas grids that are impassable are given punishment (minus rewards, for example). Considering that the impassable grids are static, *i.e.* due to the existence of buildings or other static obstacles, it is therefore unnecessary to even consider these grids as the possible states. Thus, removing these grids reduces the computation cost of the algorithm while maintaining the ability of the algorithm to find the optimum path.

As a case study, we conducted a simulation for an agricultural mobile robot operated at the Agribusiness and Technological Park (ATP), IPB University, Bogor, Indonesia. Here, the mobile robot was given a task to find the most optimum path to bring the agricultural yields from one of the greenhouses to the warehouse in ATP IPB. The considered area is from the front of the greenhouse and stretched towards the warehouse. The area was mapped and divided into grids with a $1x1m^2$ scale, resulting in a total of 936 states, as shown in Figure 2. Here, gray grids represent buildings and static obstacles that the mobile robot cannot pass through and white grids represent areas that the mobile robot can traverse. As can be seen from the figure, there are 552 gray states and 384 white states. The starting point was at state L915, located at one of the greenhouse doors, and the goal point was at state L015, located at one of the warehouse areas. The original Q-Learning algorithm considers all the grids into its computation, and we refer this simulation as Scenario-1.

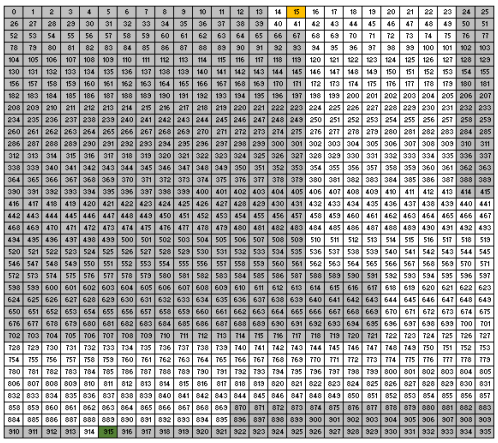

Figure 2. Original Q-Learning Method (Scenario-1): all grids in ATP IPB are computed as the possible robot states – white grids represent the traversable grids and gray grids represent the impassable grids

In the next simulation, we propose a modified Q-Learning algorithm by introducing the reduced states, that is, by removing the grids that cannot be traversed

DOI: https://doi.org/10.29207/resti.v7i3.4949

by the mobile robot, *i.e.*, the gray grids. The "new" considered grids are given new numbering/indexing as shown in Figure 3 and we refer this simulation as Scenario-2. Using the new grid indexing, the starting point in Scenario-2 is at L383 which is similar to L915 in Scenario-1, and the goal point is at L001 which is similar to L015 in Scenario-1.

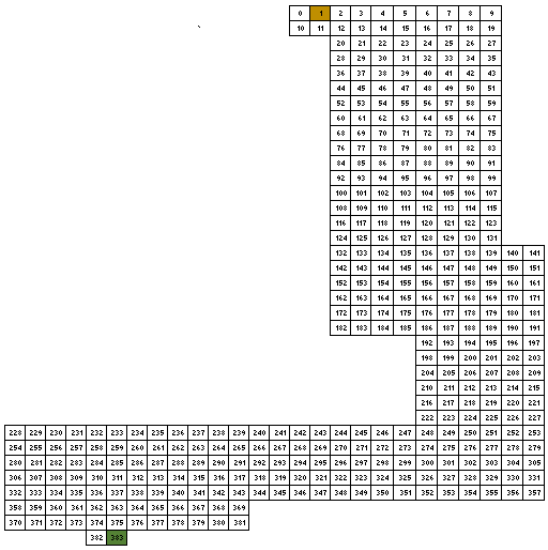

Figure 3. Modified Q-Learning Method (Scenario 2): only the traversable grids in ATP IPB are considered as the possible robot states

3. Results and Discussions

The Q-Learning algorithm simulations for both Scenario-1 (original Q-Learning algorithm) and Scenario-2 (modified Q-Learning algorithm) were conducted on a computer with an Intel Core i5-3570 processor, a clock speed of 3.4GHz and a 4GB of RAM. The software used was a Jupyter Notebook with the Python 3.9 programming language. The learning rate (α) used was 0.9, and the discount factor (γ) was varied from 0.1 to 0.9. Additionally, a fixed value of discount factor (γ) (0.9) and a learning rate (α) that varied between 0.1 to 0.9 were also used. The two variations of these values were applied to both Scenario-1 and Scenario-2, with a total of 250000 iterations. Each test was conducted ten times.

3.1 Simulation using a fixed learning rate value

The results of the simulation test with a fixed learning rate (α) of 0.9 can be seen in Table 1 (for Scenario-1) and Table 2 (for Scenario-2). The computation time was measured to determine the amount of time taken from the start of program execution until the optimal path was obtained. For discount factor (γ) values of 0.0, 0.1, 0.2, 0.3, and 0.4, the Q-Learning algorithm failed to find a path from the starting point to the target point because these settings did not converge during the iteration process. The optimal path is first discovered when the value of the γ parameter is 0.5 or greater. This indicates that the minimum value of the discount factor for the algorithm to discover a path under the condition that the learning rate parameter is 0.9 is 0.5.

Table 1. The computation time with a fixed learning rate (0.9) and 250000 iterations on Scenario-1

No.	Computation time (second)				
	γ = 0.5	γ = 0.6	γ = 0.7	γ = 0.8	γ = 0.9
1	50.45	50.42	50.41	50.59	50.67
2	50.37	50.39	50.54	50.82	50.31
3	50.43	50.57	50.44	50.70	50.71
4	51.06	50.53	50.79	50.64	50.41
5	51.37	50.51	50.32	50.63	50.79
6	50.56	50.73	51.55	51.11	51.47
7	50.43	50.55	50.57	50.67	50.80
8	50.47	51.54	50.82	51.30	50.43
9	50.42	50.58	50.48	51.30	50.48
10	50.54	50.30	51.44	**50.19**	51.26
$\bar{x}$	**50.61**	**50.61**	**50.74**	**50.79**	**50.73**

Table 2. The computation time with a fixed learning rate (0.9) and 250000 iterations on Scenario-2

No.	Computation time (second)				
	γ = 0.5	γ = 0.6	γ = 0.7	γ = 0.8	γ = 0.9
1	25.48	25.76	25.73	25.70	25.69
2	25.72	25.67	25.73	25.72	25.73
3	25.61	25.83	25.60	25.60	25.71
4	25.67	25.58	25.64	25.71	25.70
5	25.61	25.79	25.61	25.57	25.66
6	25.73	25.71	25.74	25.66	25.74
7	26.26	25.70	25.65	25.69	25.75
8	25.58	25.65	25.74	25.75	25.77
9	25.80	25.66	25.70	25.62	25.75
10	25.79	26.19	25.82	25.63	25.73
$\bar{x}$	**25.72**	**25.76**	**25.70**	**25.67**	**25.72**

In Table 1, the shortest computation time to find the optimal path in Scenario-1 was 50.19 seconds, which was achieved when the discount factor was 0.8. The longest computation time was 51.55 seconds, which was achieved when the discount factor was 0.7. The average computation time in Scenario-1 was 50.70 seconds. In Scenario-2, the shortest computation time to find the optimal path was 25.48 seconds, which was achieved when the discount factor was 0.5. The longest computation time was 26.26 seconds, also when the discount factor was 0.5. The average computation time in Scenario-2 was 25.71 seconds.

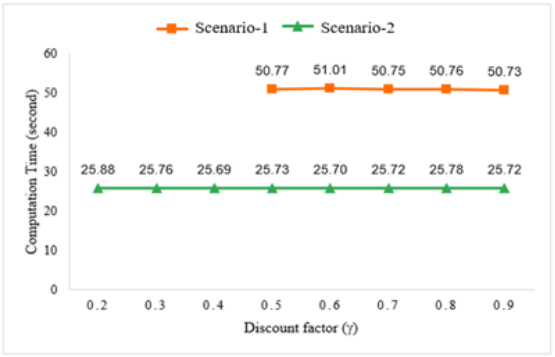

Figure 4. Differences in computational time trends to find the optimal path using a variety of discount factor (γ) values with a fixed learning rate (α) of 0.9 and 250000 iterations in the Scenario-2 and Scenario-1 areas

DOI: https://doi.org/10.29207/resti.v7i3.4949

The differences in computation time for finding the optimal path using different values of the discount factor (γ) with a fixed learning rate (α) of 0.9 and 250000 iterations in both scenarios can be seen in Figure 4. The computation time in Scenario-2 was faster than in Scenario-1, which aligns with previous studies, such as [39]–[41], which found that the more states there are, the longer it takes to reach convergence.

3.2 Simulation using a fixed discount factor

The simulation results using a fixed discount factor value of 0.9 are presented in Table 3 (for Scenario-1) and Table 4 (for Scenario-2).

Table 3. The computation time with a fixed discount factor (0.9) and 250000 iterations on Scenario-1

No.	Computation time (second)				
	α = 0.5	α = 0.6	α = 0.7	α = 0.8	α = 0.9
1	50.47	50.26	50.70	51.36	50.67
2	50.66	50.42	50.65	51.56	50.31
3	51.37	51.31	50.73	50.57	50.71
4	50.82	50.82	51.37	50.83	50.41
5	50.50	51.66	50.64	50.40	50.79
6	50.58	51.61	50.33	50.45	51.47
7	50.58	50.56	50.58	50.87	50.80
8	50.94	51.21	50.51	50.22	50.43
9	50.94	51.71	50.94	50.55	50.48
10	50.80	50.52	51.07	50.74	51.26
$\bar{x}$	**50.77**	**51.01**	**50.75**	**50.76**	**50.73**

Table 4. The computation time with a fixed discount factor (0.9) and 250000 iterations on Scenario-2

No	**Computation time (second)**							
	α = 0.2	α = 0.3	α = 0.4	α = 0.5	α = 0.6	α = 0.7	α = 0.8	α = 0.9
1	25.89	25.61	25.83	25.68	25.84	25.65	26.22	25.69
2	25.66	25.76	25.71	25.74	25.70	25.66	25.67	25.73
3	26.06	26.19	25.63	26.24	25.84	25.71	25.67	25.71
4	25.85	25.69	25.64	25.67	25.58	25.79	25.72	25.70
5	25.99	25.49	25.60	25.55	25.64	25.77	25.73	25.66
6	25.83	25.70	25.68	25.72	25.64	25.71	25.74	25.74
7	26.00	25.59	25.72	25.72	25.63	25.71	25.66	25.75
8	25.94	25.71	25.70	25.74	25.72	25.76	25.70	25.77
9	25.88	26.17	25.85	25.65	25.69	25.75	25.74	25.75
10	25.73	25.70	25.57	25.60	25.65	25.72	25.99	25.73
$\bar{x}$	**25.88**	**25.76**	**25.69**	**25.73**	**25.70**	**25.72**	**25.78**	**25.72**

In Scenario-1, when the learning rate (α) values are 0.0, 0.1, 0.2, 0.3, and 0.4, the Q-Learning algorithm cannot find the optimal path from the starting point to the target point. The optimal path starts to emerge when the learning rate value is 0.5. The shortest computation time to obtain the optimal path in Scenario-1 is 50.22 seconds, which is achieved when the learning rate value is 0.8. The longest computation time, 51.71 seconds, is obtained when the learning rate value is 0.6. The average computation time in Scenario-1 is 50.80 seconds. In Scenario-2, the optimal path is first discovered when the value of the α parameter is 0.2 or greater. This indicates that the minimum value of the learning rate for the algorithm to discover a path, provided that the discount factor parameter is 0.9, is 0.2. The shortest computation time to obtain the optimal path in Scenario-2 is 25.49 seconds, which is achieved when the learning rate value is 0.3. The longest computation time, 26.24 seconds, is obtained when the learning rate value is 0.5. The average computation time in Scenario-2 is 25.75 seconds.

The difference in computation time to find the optimal path using different learning rate (α) values with a fixed discount factor (γ) of 0.9 and 250000 iterations in scenario-2 and Scenario-1 areas is shown in Figure 5. The computation time in Scenario-2 is faster than the computation time in Scenario-1, which is consistent with other studies [39]–[41].

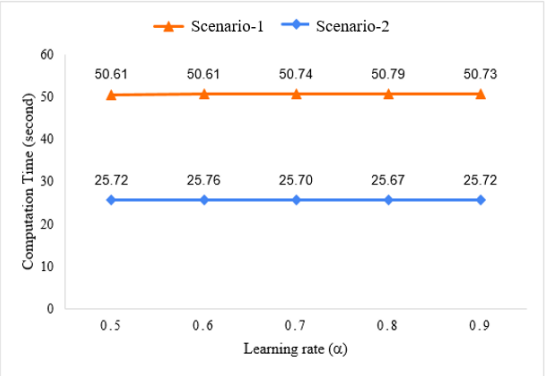

Figure 5. Differences in computational time trends to find the optimal path using a variety of learning rate (α) values with a fixed discount factor (γ) values of 0.9 and 250000 iterations in the Scenario-1 and Scenario-2 areas

4.3 The path planning results and the computation time for the original Q-Learning algorithm and the modified Q-Learning algorithm

The results of optimum path for the original Q-Learning algorithm (Scenario-1) are shown in Table 5. Meanwhile, the obtained optimum path for the modified Q-Learning algorithm (Scenario-2) are shown in Table 6. These results showed that both the original and modified Q-Learning algorithms produce the same number of states as the optimum path: 56 states. Variations are observed in the chosen 56 states as shaded gray in Table 5 and Table 6, with a maximum of five state variations.

Table 5 and Table 6 also showed the comparison of the computation time between the two algorithms. The original Q-Learning algorithm (Scenario-1) required 50.75s in average to find the most optimum path, whereas the modified Q-Learning algorithm (Scenario-2) required only 25.74s. These results indicate that the modified Q-Learning algorithm reduces the computation time by 50.71% while maintaining the algorithm's ability to find the most optimum path.

DOI: https://doi.org/10.29207/resti.v7i3.4949

Table 5. Original Q-Learning algorithm: the obtained optimum path and its computation cost

α	γ	Total states	Sequence of states from L915 to L015	Computation time (s)
0.5	**0.9**	56	'L915', 'L889', 'L890', 'L891', 'L892', 'L866', 'L867', 'L868', 'L842', 'L843', 'L817', 'L791', 'L792', 'L793', 'L794', 'L768', 'L769', 'L743', 'L744', 'L745', 'L746', 'L747', 'L748', 'L722', 'L696', 'L670', 'L644', 'L618', 'L592', 'L566', 'L540', 'L514', 'L513', 'L487', 'L486', 'L460', 'L434', 'L408', 'L382', 'L381', 'L355', 'L329', 'L328', 'L302', 'L276', 'L250', 'L224', 'L198', 'L172', 'L146', 'L120', 'L094', 'L068', 'L042', 'L016', 'L015'	**50.61**
0.6	0.9	56	'L915', 'L889', 'L890', 'L891', 'L892', 'L893', 'L867', 'L868', 'L869', 'L843', 'L844', 'L845', 'L819', 'L793', 'L794', 'L795', 'L796', 'L797', 'L798', 'L772', 'L746', 'L747', 'L748', 'L722', 'L696', 'L670', 'L644', 'L618', 'L592', 'L566', 'L565', 'L539', 'L513', 'L512', 'L486', 'L460', 'L434', 'L408', 'L407', 'L381', 'L355', 'L354', 'L328', 'L302', 'L276', 'L250', 'L224', 'L198', 'L172', 'L146', 'L120', 'L094', 'L068', 'L042', 'L016', 'L015'	**51.61**
0.7	0.9	56	'L915', 'L889', 'L890', 'L864', 'L865', 'L839', 'L813', 'L814', 'L788', 'L762', 'L763', 'L737', 'L738', 'L739', 'L740', 'L741', 'L742', 'L743', 'L744', 'L745', 'L746', 'L747', 'L748', 'L722', 'L696', 'L670', 'L644', 'L618', 'L592', 'L566', 'L540', 'L514', 'L488', 'L462', 'L461', 'L435', 'L409', 'L383', 'L357', 'L331', 'L305', 'L279', 'L253', 'L252', 'L226', 'L200', 'L174', 'L148', 'L147', 'L121', 'L095', 'L069', 'L043', 'L017', 'L016', 'L015'	**50.74**
0.8	0.9	56	'L915', 'L889', 'L890', 'L891', 'L892', 'L893', 'L867', 'L841', 'L815', 'L816', 'L817', 'L818', 'L819', 'L793', 'L767', 'L768', 'L742', 'L743', 'L744', 'L745', 'L746', 'L747', 'L748', 'L722', 'L696', 'L670', 'L644', 'L618', 'L592', 'L566', 'L540', 'L514', 'L488', 'L462', 'L436', 'L410', 'L384', 'L358', 'L332', 'L306', 'L280', 'L254', 'L228', 'L202', 'L176', 'L175', 'L149', 'L123', 'L097', 'L071', 'L070', 'L044', 'L043', 'L017', 'L016', 'L015'	**50.79**
0.9	0.9	56	'L915', 'L889', 'L890', 'L891', 'L892', 'L893', 'L894', 'L868', 'L869', 'L843', 'L844', 'L845', 'L846', 'L847', 'L848', 'L849', 'L823', 'L797', 'L771', 'L772', 'L773', 'L774', 'L748', 'L722', 'L696', 'L670', 'L644', 'L618', 'L592', 'L566', 'L540', 'L514', 'L488', 'L487', 'L461', 'L435', 'L409', 'L383', 'L382', 'L356', 'L330', 'L304', 'L303', 'L277', 'L276', 'L250', 'L224', 'L198', 'L172', 'L146', 'L120', 'L094', 'L068', 'L042', 'L016', 'L015'	**50.73**
0.9	0.8	56	'L915', 'L889', 'L863', 'L864', 'L865', 'L839', 'L813', 'L814', 'L815', 'L816', 'L817', 'L818', 'L792', 'L766', 'L767', 'L741', 'L742', 'L743', 'L744', 'L745', 'L746', 'L747', 'L748', 'L722', 'L696', 'L670', 'L644', 'L618', 'L592', 'L566', 'L565', 'L539', 'L538', 'L512', 'L511', 'L485', 'L459', 'L433', 'L407', 'L381', 'L355', 'L329', 'L303', 'L277', 'L251', 'L225', 'L199', 'L173', 'L147', 'L121', 'L095', 'L069', 'L068', 'L042', 'L041', 'L015'	**50.76**
0.9	0.7	56	'L915', 'L889', 'L890', 'L864', 'L865', 'L866', 'L867', 'L868', 'L842', 'L843', 'L817', 'L791', 'L792', 'L793', 'L794', 'L795', 'L769', 'L770', 'L771', 'L745', 'L746', 'L747', 'L748', 'L722', 'L696', 'L670', 'L644', 'L618', 'L592', 'L566', 'L540', 'L539', 'L513', 'L512', 'L486', 'L460', 'L434', 'L408', 'L407', 'L381', 'L355', 'L329', 'L328', 'L302', 'L276', 'L250', 'L224', 'L198', 'L172', 'L146', 'L120', 'L094', 'L068', 'L042', 'L041', 'L015'	**50.75**
0.9	0.6	56	'L915', 'L889', 'L890', 'L891', 'L892', 'L866', 'L867', 'L841', 'L842', 'L843', 'L817', 'L818', 'L819', 'L820', 'L794', 'L795', 'L796', 'L797', 'L771', 'L772', 'L773', 'L747', 'L748', 'L722', 'L696', 'L670', 'L644', 'L618', 'L592', 'L566', 'L565', 'L539', 'L513', 'L512', 'L511', 'L485', 'L459', 'L433', 'L407', 'L381', 'L355', 'L329', 'L303', 'L277', 'L251', 'L225', 'L199', 'L173', 'L172', 'L146', 'L120', 'L094', 'L068', 'L042', 'L041', 'L015'	**51.01**
0.9	**0.5**	56	'L915', 'L889', 'L863', 'L837', 'L838', 'L839', 'L840', 'L841', 'L842', 'L816', 'L817', 'L791', 'L792', 'L766', 'L767', 'L768', 'L769', 'L770', 'L744', 'L745', 'L746', 'L747', 'L748', 'L722', 'L696', 'L670', 'L644', 'L618', 'L592', 'L566', 'L565', 'L539', 'L538', 'L512', 'L486', 'L460', 'L434', 'L408', 'L382', 'L356', 'L330', 'L304', 'L278', 'L252', 'L226', 'L225', 'L199', 'L173', 'L147', 'L146', 'L120', 'L094', 'L068', 'L042', 'L041', 'L015'	**50.77**
			$\bar{x}$	**50.75**

Table 6. Modified Q-Learning algorithm: the obtained optimum path and its computation cost

α	γ	Total states	Sequence of states from L383 to L001	Computation time (s)
0.9	**0.5**	56	'L383', 'L375', 'L363', 'L337', 'L311', 'L285', 'L259', 'L233', 'L234', 'L235', 'L236', 'L237', 'L238', 'L239', 'L240', 'L241', 'L242', 'L243', 'L244', 'L245', 'L246', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L176', 'L166', 'L156', 'L146', 'L136', 'L128', 'L120', 'L112', 'L104', 'L096', 'L088', 'L080', 'L072', 'L064', 'L056', 'L048', 'L040', 'L032', 'L024', 'L016', 'L006', 'L005', 'L004', 'L003', 'L002', 'L001'	**25.72**
0.9	0.6	56	'L383', 'L375', 'L363', 'L337', 'L311', 'L285', 'L259', 'L233', 'L234', 'L235', 'L236', 'L237', 'L238', 'L239', 'L240', 'L241', 'L242', 'L243', 'L244', 'L245', 'L246', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L176', 'L166', 'L156', 'L146', 'L136', 'L128', 'L120', 'L112', 'L104', 'L096', 'L088', 'L080', 'L072', 'L064', 'L056', 'L048', 'L040', 'L032', 'L024', 'L016', 'L006', 'L005', 'L004', 'L003', 'L002', 'L001'	**25.76**
0.9	0.7	56	'L383', 'L375', 'L363', 'L337', 'L311', 'L285', 'L259', 'L233', 'L234', 'L235', 'L236', 'L237', 'L238', 'L239', 'L240', 'L241', 'L242', 'L243', 'L244', 'L245', 'L246', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L176', 'L166', 'L156', 'L146', 'L136', 'L128', 'L120', 'L112', 'L104', 'L096', 'L088', 'L080', 'L072', 'L064', 'L056', 'L048', 'L040', 'L032', 'L024', 'L016', 'L006', 'L005', 'L004', 'L003', 'L002', 'L001'	**25.70**

DOI: https://doi.org/10.29207/resti.v7i3.4949

α	γ	Total states	Sequence of states from L383 to L001	Computation time (s)
0.9	0.8	56	'L383', 'L375', 'L363', 'L364', 'L365', 'L366', 'L367', 'L368', 'L342', 'L343', 'L317', 'L318', 'L319', 'L320', 'L321', 'L322', 'L323', 'L297', 'L298', 'L299', 'L273', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L176', 'L166', 'L156', 'L146', 'L145', 'L135', 'L127', 'L126', 'L118', 'L110', 'L109', 'L108', 'L100', 'L092', 'L084', 'L076', 'L068', 'L060', 'L052', 'L044', 'L036', 'L028', 'L020', 'L012', 'L011', 'L001'	**25.67**
0.9	0.9	56	'L383', 'L375', 'L376', 'L377', 'L365', 'L366', 'L367', 'L368', 'L342', 'L343', 'L317', 'L291', 'L265', 'L239', 'L240', 'L241', 'L242', 'L243', 'L244', 'L245', 'L246', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L176', 'L175', 'L165', 'L155', 'L145', 'L135', 'L134', 'L126', 'L118', 'L110', 'L102', 'L094', 'L093', 'L092', 'L084', 'L076', 'L068', 'L060', 'L052', 'L044', 'L036', 'L028', 'L020', 'L012', 'L002', 'L001'	**25.72**
0.2	**0.9**	56	'L383', 'L375', 'L376', 'L377', 'L378', 'L379', 'L380', 'L368', 'L342', 'L316', 'L290', 'L264', 'L238', 'L239', 'L240', 'L241', 'L242', 'L243', 'L244', 'L245', 'L246', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L185', 'L184', 'L183', 'L182', 'L172', 'L162', 'L152', 'L142', 'L132', 'L124', 'L116', 'L108', 'L100', 'L092', 'L084', 'L076', 'L068', 'L060', 'L052', 'L044', 'L036', 'L028', 'L020', 'L012', 'L011', 'L001'	**25.88**
0.3	0.9	56	'L383', 'L375', 'L363', 'L337', 'L311', 'L285', 'L259', 'L260', 'L234', 'L235', 'L236', 'L237', 'L238', 'L239', 'L240', 'L241', 'L242', 'L243', 'L244', 'L245', 'L246', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L185', 'L184', 'L174', 'L173', 'L172', 'L162', 'L152', 'L142', 'L132', 'L124', 'L116', 'L108', 'L100', 'L092', 'L084', 'L076', 'L068', 'L060', 'L052', 'L044', 'L036', 'L028', 'L020', 'L012', 'L011', 'L001'	**25.76**
0.4	0.9	56	'L383', 'L375', 'L376', 'L377', 'L378', 'L379', 'L367', 'L368', 'L342', 'L343', 'L344', 'L345', 'L346', 'L347', 'L321', 'L295', 'L269', 'L243', 'L244', 'L245', 'L246', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L185', 'L184', 'L183', 'L182', 'L172', 'L162', 'L152', 'L142', 'L132', 'L124', 'L116', 'L108', 'L100', 'L092', 'L084', 'L076', 'L068', 'L060', 'L052', 'L044', 'L036', 'L028', 'L020', 'L012', 'L002', 'L001'	**25.69**
0.5	0.9	56	'L383', 'L375', 'L376', 'L377', 'L378', 'L379', 'L380', 'L368', 'L342', 'L343', 'L344', 'L345', 'L346', 'L347', 'L348', 'L322', 'L323', 'L297', 'L298', 'L272', 'L246', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L176', 'L166', 'L156', 'L146', 'L136', 'L135', 'L127', 'L126', 'L125', 'L117', 'L116', 'L108', 'L100', 'L092', 'L084', 'L076', 'L068', 'L060', 'L052', 'L044', 'L036', 'L028', 'L020', 'L012', 'L011', 'L001'	**25.73**
0.6	0.9	56	'L383', 'L375', 'L363', 'L364', 'L365', 'L366', 'L367', 'L368', 'L342', 'L343', 'L317', 'L318', 'L292', 'L266', 'L267', 'L268', 'L269', 'L270', 'L271', 'L272', 'L273', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L185', 'L184', 'L174', 'L173', 'L172', 'L162', 'L152', 'L142', 'L132', 'L124', 'L116', 'L108', 'L100', 'L092', 'L084', 'L076', 'L068', 'L060', 'L052', 'L044', 'L036', 'L028', 'L020', 'L012', 'L002', 'L001'	**25.70**
0.7	0.9	56	'L383', 'L375', 'L363', 'L364', 'L365', 'L366', 'L367', 'L368', 'L342', 'L343', 'L317', 'L318', 'L292', 'L266', 'L267', 'L268', 'L269', 'L270', 'L271', 'L272', 'L273', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L185', 'L184', 'L174', 'L173', 'L172', 'L162', 'L152', 'L142', 'L132', 'L124', 'L116', 'L108', 'L100', 'L092', 'L084', 'L076', 'L068', 'L060', 'L052', 'L044', 'L036', 'L028', 'L020', 'L012', 'L002', 'L001'	**25.72**
0.8	0.9	56	'L383', 'L375', 'L376', 'L377', 'L378', 'L379', 'L380', 'L368', 'L342', 'L343', 'L344', 'L345', 'L346', 'L347', 'L348', 'L322', 'L323', 'L297', 'L298', 'L272', 'L246', 'L247', 'L248', 'L222', 'L216', 'L210', 'L204', 'L198', 'L192', 'L186', 'L176', 'L166', 'L156', 'L146', 'L136', 'L135', 'L127', 'L126', 'L125', 'L117', 'L116', 'L108', 'L100', 'L092', 'L084', 'L076', 'L068', 'L060', 'L052', 'L044', 'L036', 'L028', 'L020', 'L012', 'L011', 'L001'	**25.78**
			$\bar{x}$	**25.74**

4. Conclusion

This study proposes a modified version of Q-Learning algorithm by removing the impassable grids from the Q-Learning computation. Justification of the modified Q-Learning algorithm was done by some simulations, where both the original and the modified Q-Learning algorithm were used to find the most optimum path for an autonomous agricultural robot in Agribusiness and Technology Park (ATP), IPB University, Bogor, Indonesia. In this case, the agricultural robot was given a task to find the most optimum path in order to bring some agricultural yields from one of the greenhouses to the warehouse in ATP IPB. Simulation results showed that both the original and modified Q-Learning algorithms produced the same number of states as the optimum path for the robot, *i.e.*, 56 states. However, the modified Q-Learning algorithm is capable of finding the path to the destination point with a minimum learning rate parameter value of 0.2 when the discount factor value is 0.9. This demonstrates that providing a small learning rate parameter value in the modified Q-Learning algorithm can still result in the discovery of the optimal path. Furthermore, the results showed that the original Q-Learning algorithm required an average of 50.75s to find the optimal path, whereas the modified Q-Learning algorithm required an average of 25.74s. In other words, the proposed modified Q-Learning algorithm reduced the computation cost to 50.71% from the original Q-Learning algorithm, while maintaining the ability of the algorithm in finding the most optimum path. Further study will focus on the improvement of modified Q-Learning algorithm so that the algorithm

DOI: https://doi.org/10.29207/resti.v7i3.4949

can produce some different routes as the robot's optimum paths. This is important to avoid collisions between robots, especially when there are multiple agricultural robots that need to perform the same task at a given time.

References

[1] S. S. Valle and J. Kienzle, *Agriculture 4.0 - Agricultural robotics and automated equipment for sustainable crop production*, vol. 24. Rome: FAO, 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, no. July, 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.

[4] S. Ruzzante, R. Labarta, and A. Bilton, "Adoption of agricultural technology in the developing world: A meta-analysis of the empirical literature," *World Dev.*, 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. Magalhaes, "Path planning for ground robots in agriculture: A short review," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2020*, 2020, pp. 61–66, 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.

[9] L. F. P. Oliveira, M. F. Silva, and A. P. Moreira, "Agricultural robotics: A state of the art survey," *Robot. Hum. Life- Proc. 23rd Int. Conf. Climbing Walk. Robot. Support Technol. Mob. Mach. CLAWAR 2020*, no. August, pp. 279–286, 2020, doi: 10.13180/clawar.2020.24-26.08.44.

[10] S. Fountas, N. Mylonas, I. Malounas, E. Rodias, C. H. Santos, and E. Pekkeriet, "Agricultural robotics for field operations," *Sensors (Switzerland)*, vol. 20, no. 9, pp. 1–27, 2020, doi: 10.3390/s20092672.

[11] X. Yu *et al.*, "A lab-customized autonomous humanoid apple harvesting robot," *Comput. Electr. Eng.*, vol. 96, p. 107459, Dec. 2021, doi: 10.1016/j.compeleceng.2021.107459.

[12] K. Zhang, K. Lammers, P. Chu, Z. Li, and R. Lu, "System design and control of an apple harvesting robot," *Mechatronics*, vol. 79, Nov. 2021, doi: 10.1016/j.mechatronics.2021.102644.

[13] 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.

[14] 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, Jan. 2021, doi: 10.1016/j.compag.2020.105911.

[15] 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, Feb. 2023, doi: 10.1016/j.compag.2022.107584.

[16] 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.

[17] 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. 05, pp. 804–820, 2021.

[18] 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, Apr. 2023, doi: 10.1016/j.eswa.2022.119410.

[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] 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.

[21] 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, Nov. 2022, doi: 10.1016/j.oceaneng.2022.112421.

[22] 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.

[23] 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, Feb. 2022, doi: 10.1016/j.asoc.2021.108108.

[24] 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, Dec. 2021, doi: 10.1016/j.eswa.2021.115706.

[25] 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.

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

[27] 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, Feb. 2023, doi: 10.1016/j.ocecoaman.2022.106428.

[28] 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.

[29] L. Zhao, F. Wang, and Y. Bai, "Route planning for autonomous vessels based on improved artificial fish swarm algorithm," *Ships Offshore Struct.*, pp. 1–10, Jun. 2022, doi: 10.1080/17445302.2022.2081423.

[30] S. Kumar and A. Sikander, "A modified probabilistic roadmap algorithm for efficient mobile robot path planning," *Eng. Optim.*, pp. 1–19, Aug. 2022, doi: 10.1080/0305215X.2022.2104840.

[31] 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, 2022, doi: 10.1016/j.asej.2021.101675.

[32] 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.

DOI: https://doi.org/10.29207/resti.v7i3.4949

108905, Jan. 2023, doi: 10.1016/j.cie.2022.108905.
[33] 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.*, Jan. 2023, doi: 10.1016/j.cogsys.2023.01.005.
[34] 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, Nov. 2022, doi: 10.1016/j.jpdc.2022.06.014.
[35] 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.
[36] 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, Jan. 2022, doi: 10.1016/j.jfranklin.2021.12.003.
[37] 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, Aug. 2022, doi: 10.1016/j.knosys.2022.109075.
[38] 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, Aug. 2022, doi: 10.1016/j.eswa.2022.117191.
[39] 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.
[40] S. Gu, "An algorithm for path planning based on improved Q-Learning," in *The Genetic and Evolutionary Computing*, 2019, pp. 20–29, doi: https://doi.org/10.1007/978-981-15-3308-2_3.
[41] S. Gu and G. Mao, "An improved Q-Learning algorithm for path planning in maze environments," in *Intelligent Systems and Applications*, 2020, vol. 1251, no. 2, pp. 545–557, doi: https://doi.org/10.1007/978-3-030-55187-2_40.
[42] 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.
[43] 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)*, 2018, pp. 46–50, doi: 10.1109/ICRAS.2018.8443226.
[44] T. Zhang, X. Huo, S. Chen, B. Yang, and G. Zhang, "Hybrid path planning of a quadrotor UAV based on Q-Learning algorithm," in *Chinese Control Conference (CCC)*, 2018, vol. 2018-July, pp. 5415–5419, doi: 10.23919/ChiCC.2018.8482604.
[45] C. J. C. H. Watkins, "Technical Note Q-Learning," *Mach. Learn.*, vol. 8, pp. 279–292, 1992, doi: 10.1109/ICCC49849.2020.9238991.
[46] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An introduction*, Second. Cambridge, Massachusetts: MIT Press, 2018.

DOI: https://doi.org/10.29207/resti.v7i3.4949