Penerbit UTHM

IJIE

Journal homepage: http://penerbit.uthm.edu.my/ojs/index.php/ijie
ISSN : 2229-838X e-ISSN : 2600-7916

The International Journal of Integrated Engineering

Mobile Robot Path Planning using Q-Learning with Guided Distance and Moving Target Concept

Ee Soong Low[1], Pauline Ong[1*], Cheng Yee Low[1]

[1]Faculty of Mechanical and Manufacturing Engineering,
Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor, MALAYSIA

*Corresponding Author

DOI: https://doi.org/10.30880/ijie.2021.13.02.020
Received 1 January 2020; Accepted 3 December 2020; Available online 28 February 2021

Abstract: Classical Q-learning algorithm is a reinforcement of learning algorithm that has been applied in path planning of mobile robots. However, classical Q-learning suffers from slow convergence rate and high computational time. This is due to the random decision making for direction during the early stage of path planning. Such weakness curtails the ability of mobile robot to make instantaneous decision in real world application. In this study, the distance aspect and moving target concept were added to Q-learning in order to enhance the direction decision making ability and bypassing dead end. With the addition of these features, Q-learning is able to converge faster and generate shorter path. Consequently, the proposed improved Q-learning is able to achieve average improvement of 29.34-94.85%, 18.29-29.69% and 75.76-99.50% in time used, shortest distance and total distance used, respectively.

Keywords: Guided distance, moving target, mobile robot, path planning, Q-learning, reinforcement

1. Introduction

Many applications and fields in this modern world use mobile robots, such as automated guided vehicles [1, 2], self-driving vehicles [3-5] and cleaning robots [6, 7]. A path is essential for the normal and smooth operation of the robots. If an optimized path is given, the efficiency of mobile robots increases gradually. This enables the robots to complete the given task in a faster and safer manner. Hence, researchers start discovering ways to further improve path planning process for mobile robots.

Path planning for mobile robots has never been easy. A path enables a robot to move from one point to another desired point without hitting any obstacles [8]. Q-learning is a reinforcement learning that has been widely applied in mobile robot path planning [9-18]. Q-learning has the ability to learn a suitable strategy through repetitive iteration. The strategy is continuously improved through actions made by robots at specific states by giving rewards/penalty to Q-values based on the taken actions. As the iteration increases, when a robot reaches the same state, Q-learning is able to provide a better action/solution based on the historical Q-values. Ironically, several weaknesses arise from these advantages. First, Q-learning consumes high computation time to achieve the optimal path. The computational cost is more critical when dealing with real-world situation due to large amount of states needed [12]. On the other hand, navigation in the environment with dynamic obstacles requires even more computation power.

In this regard, several modifications have been introduced to Q-learning in order to enhance its performance, where one of them is by hybridizing the optimization algorithm with Q-learning [14, 19-23]. Integration of improved particle swarm optimization with perturbed velocity into Q-learning by Das has improved the convergence rate and global search ability [20]. As a result, robots are able to operate smoothly in an environment with multiple obstacles. In addition to particle swarm optimization (PSO), artificial bee colony (ABC) optimization algorithm has been utilized to improve the global search ability of Q-learning in solving multirobot navigation problem [24]. Other than direct implementation of optimization algorithm, adaptive memetic algorithm (AMA) is used as a medium between Q-learning and optimization algorithm. AMA is hybridized with ABC optimization in [24] and differential evolution (DE) algorithm in [14], and

*Corresponding author: ongp@uthm.edu.my

subsequently, the hybridized AMA is integrated with Q-learning. Besides improvement in runtime and accuracy, the proposed algorithm outperforms PSO and genetic algorithm (GA) in all simulations. Besides, the abilities of Q-learning in exploration and exploitation are refined through the introduction of metropolis criterion from simulated annealing (SA) algorithm [19]. This refinement prevents Q-learning from excessive exploitation while enhancing the convergence performance.

Artificial neural network (ANN) is another aspect that has been used to improve Q-learning in path planning [17, 25]. Duguleana and Mogan [26] use Pos-Net neural network to determine the next action by using current time, current state and obstacle detection as references. Experimental results show that better convergence and local minimum avoidance are achieved, even in an environment full of dynamic and static obstacles. On the other hand, back-propagation neural network is implemented by Huang et. al for calculation of Q-values [27]. This has significantly accelerated the learning rate of Q-learning while preserving the obstacle avoidance ability.

It has been reported in literature that the abilities of Q-learning can be improved by including additional information, such as global knowledge or distance. The studies in [28, 29] reported that including the information of distance between robots and ball enables the robots to identify which robot is the closest to the ball for modular Q-learning and adaptive Q-learning in soccer robots. The chosen robot will act as attacker while others will be side kickers and defenders. This will provide the robots a more refined coordination, and enhance cooperation among soccer robots through role switching. The results show that soccer robot team with these algorithms scores better than others. On the other hand, Euclidean distance is used in the formation of fitness function in path planning of mobile robot [20]. In [20], the Euclidean distance defines the distance between robots and distance between current position and desired position. For multirobot case in [20], Q-learning with improved particle swarm optimization and differentially perturbed velocity (QIPSO-DV) is used to optimize the fitness function of multirobot. It has been demonstrated that the time taken and space complexity are greatly reduced by using classical Q-learning (CQL) as reference. Besides, QIPSO-DV is able to provide better performance in evaluation function of average total trajectory path travelled (ATTPT), in comparison with IPSO-DV, PSO and CQL. The better performance in ATTPT is, the lower travelling time and smaller direction fluctuation are.

In order to improve Q-learning, metric of distance and moving target concept will be added into Q-learning in this work. The improved Q-learning (IQL) model will be able to determine the closest direction towards the target through metric of distance and avoid local minimum through moving target concept, and thus, improving the convergence rate.

This paper is organized as follows. First, Q-learning is introduced in Section 2, followed by the implementation of metric of distance and moving target concept in Section 3. Next, in Section 4, results and discussion of the proposed IQL are presented. Lastly, the conclusion of this work is made in Section 5.

2. Classical Q-Learning

CQL, developed by Watkins in 1992 [10], falls under the category of model-free reinforcement learning. The model of CQL is formed through continuous learning in an environment by performing actions, evaluating the actions and calculating the rewards. The Q-table is used to store the reward values (Q-values). The historical Q-values will be recalled when the agent or robot reaches the same state. The process of updating the Q-values is repeated until the agent reaches the goal. As the iterations increase, the CQL is able to provide better result when the robot experiences the same state for multiple times.

The Q-value is updated using:

$$Q(s_i,a_i) = Q(s_i,a_i) + \alpha\left[r_i + \gamma \max_a Q(s_{i+1},a_{i+1}) - Q(s_i,a_i)\right] \quad (1)$$

where $Q(s_i,a_i)$ refers to Q-value with state, s_i and action, a_i at interval i. α represents the learning rate, γ represents the discount factor and r_i represents the reward given at interval i. $\max_a Q(s_{i+1},a_{i+1})$ represents the highest Q-values among all action-state sets at interval $i+1$. The pseudo code of CQL is presented in Algorithm 1 (Table 1).

When the CQL moves on to the next state, the decision will be based on the maximum Q-values among the next states. However, when multiple states have the same maximum Q-values, random selection will be done. This situation becomes a drawback in the initial stage of Q-learning where all Q-values are initially zero. The random selection causes low convergence rate in the initial stage. Therefore, the metric of distance is integrated into CQL in order to enhance the convergence rate in this study.

Table 1 - Classical Q-Learning algorithm

1.	Initiate all Q-values, $Q(s,a)$ in Q-table which is zero
2.	Select a starting state, $Q(s_1,a_1)$
3.	while (iteration < max iteration or goal is not achieved)

4.	Select an action, a within the available actions in the current state according to the highest Q-value in the next state (randomly choose one if more than 1 state is having Q-values which is highest)
5.	Perform the selected action, a and reward or penalty, r will be given
6.	Update the Q-value using Equation (1)
7.	Move the state to new state, s'
8.	end while

3. Methodology

In this section, the metric of distance and moving target concept are introduced. The metric of distance measures the distance between target and robot position as reference for Q-learning to enable better convergence. The escape mechanism is introduced at the same time to allow robots to break free from local minimum. In addition, moving target concept further enhances the path planning by bypassing the local minimum area.

3.1 Formulation of Metric of Distance

Several rules and assumptions were made for the designed metric of distance, which are:

- The location of robot and target are known
- There are eight possible actions for the robot to select for the next state: (i) North, (ii) North-east, (iii) North-west, (iv) West, (v) East, (vi) South, (vii) South-east and (viii) South-west
- The environment is modelled into grid in Cartesian plan. Robot motion is limited by grid to grid motion. The maximum unit for x-axis and y-axis motion of robot is 1 unit

The metric of distance is defined as the total distance from robot to the next state, and the next state to the target, which is expressed as:

$$\text{distance}_{total} = \text{distance}_{curr-next} + \text{distance}_{next-\text{targ}} \tag{2}$$

The distance_{total} represents total distance, $\text{distance}_{curr-next}$ represents the distance between the robot's current position with next state position, and $\text{distance}_{next-\text{targ}}$ represents the distance between next state position with target's position.

Particularly, the $\text{distance}_{curr-next}$ is expressed as:

$$\text{distance}_{curr-next} = \sqrt{\left(x_{next} - x_{curr}\right)^2 + \left(y_{next} - y_{curr}\right)^2} \tag{3}$$

while $\text{distance}_{next--\text{targ}}$ is expressed as:

$$\text{distance}_{next-targ} = \sqrt{\left(x_{targ} - x_{next}\right)^2 + \left(y_{targ} - y_{next}\right)^2} \tag{4}$$

where x is the x-axis position, y is the y-axis position, $next$ is the next state position and targ is the target's position. The idea metric of distance is illustrated in Figure 1.

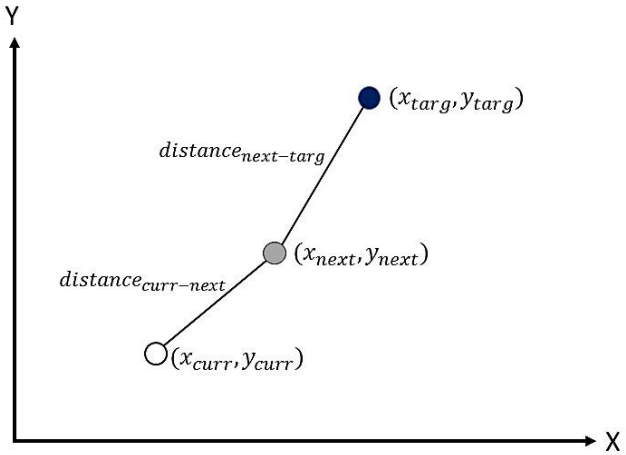


Fig. 1 - Metric of distance

3.2 Applying of Metric of Distance Into CQL

For the IQL, choosing the next state was no longer based on the maximum Q-value from the available possible states. Instead, the next state was selected based on the available mode. In this case, two modes were introduced, which are:

- Default mode: The metric of distance was enacted in the selection of the next state.
- Stuck mode: The escape mechanism was enacted.

The default mode was implemented most of the time during the operation of IQL, except when the robot was stuck in a local minimum. For the default mode, the next state was selected based on the lowest total distance among the next possible states. Therefore, the total distance of next possible states was calculated beforehand when selecting the next state.

However, the IQL lost its ability to escape from the dead end or local minimum when the distance of metric was applied. This is due to the location of dead end which tends to be the shortest distance towards the target. As a result, the stuck mode was introduced to prevent the IQL from getting trapped in the local minimum. The stuck mode will be activated when a dead end is found. The detection of dead end is based on comparing the total distance of the selected next state and the lowest stored total distance. When the total distance of next state is equal or lower than the shortest distance, it means that the robot encounters a dead end.

The dead end situation is illustrated in Figure 2, where the robot is located at the position of the lowest total distance. The next available states are not possible to be lower than the lowest total distance. Therefore, for IQL, when the total distance of next available state is equal or higher than the lowest total distance for 4 consecutive iterations, the stuck mode will be activated to replace the default mode until the total distance of the next state is shorter than the lowest total distance.

In stuck mode, if all the Q-values of next states are zero, the IQL will select the next state randomly from all possible next states. Otherwise, the next possible state with the highest Q-values is selected. The operation of stuck mode when the next state is selected randomly is presented in Figure 3.

In order to terminate the stuck mode, the total distance of next state must be shorter than the lowest total distance. When a robot has escaped from the local minima, the total distance of the next state will be lower than the lowest total distance. Figure 4 demonstrates stuck mode is deactivated as the total distance of next state is lower than the recorded lowest total distance.

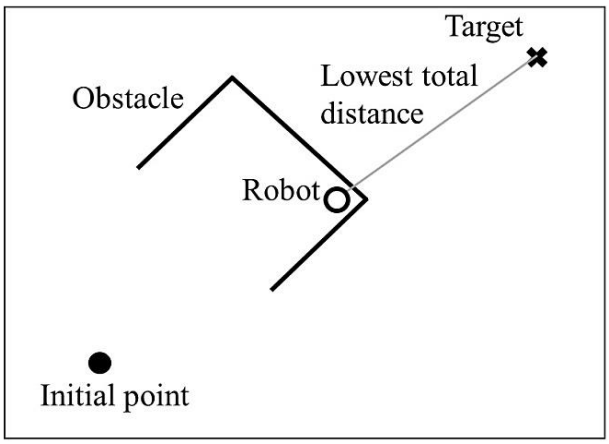


Fig. 2 - Dead end encountered by robot

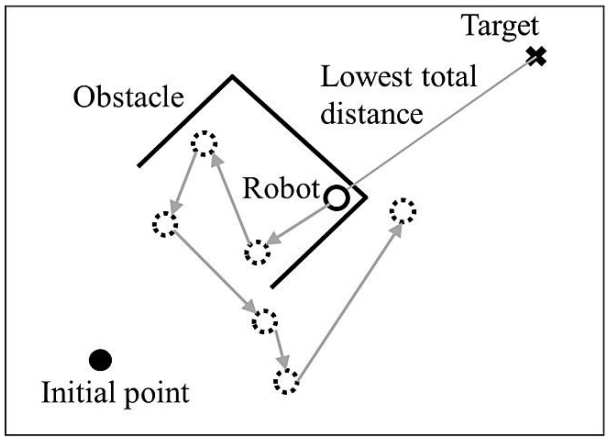


Fig. 3 - Random motion when stuck mode activated

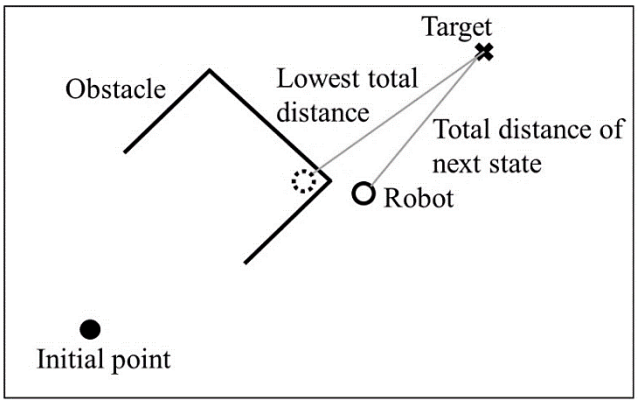


Fig. 4 - Deactivation of stuck mode

3.3 Applying of Moving Target Into CQL

Although the IQL is able to escape from dead end by activating the stuck mode, the generated path will follow the trace of the agent during the learning stage. Therefore, the final path will still be tangling around the dead end prior to moving towards the target position. The concept of moving target was introduced in order to solve this limitation. In this regard, a virtual target was created to act as a real target in the modelled environment. Initially, the virtual target was at the same position as the target. For each iteration, the virtual target moved to the last step before the robot reached the virtual target. As the iteration increased, the virtual target would get closer to the initial position. When the virtual target reached the entry of a dead end, the robot bypassed it, as the total distance towards the virtual target became shorter than the total distance towards the dead end. Thus, the generated path was able to bypass the local minimum. These processes are presented in Figure 5 and Figure 6.

The transition of position of virtual target is formulated as follows:

$$x_{n,i+1} = x_{n-1,i}^{\,x_{step,iteration}} \tag{5}$$

$$y_{n,i+1} = y_{n-1,i}^{\,y_{step,iteration}} \tag{6}$$

where x is x-coordinate, y is y-coordinate and n is the last step for iteration i.

The pseudocode of IQL is presented in Algorithm 2 (Table 2).

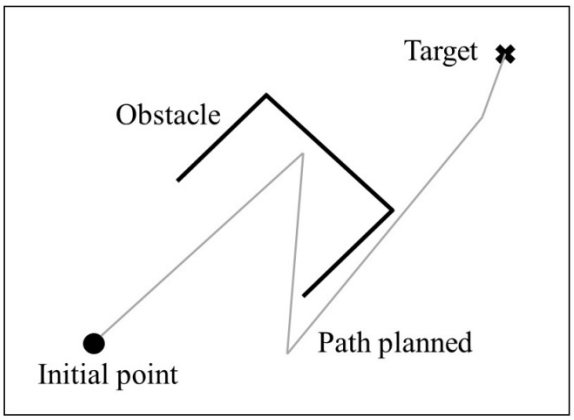


Fig. 5 - Path planned without moving target

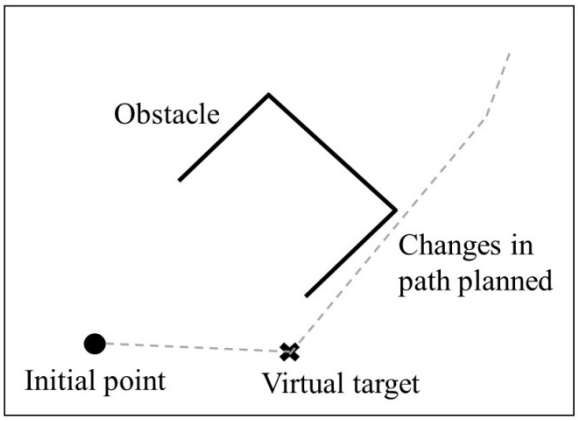


Fig. 6 - Changes in path planned with moving target

Table 2 - Q-learning with guided distance

1.	Initiate all Q-values, $Q(s,a)$ in Q-table which is zero
2.	Select a starting state, $Q(s_1,a_1)$
3.	while (iteration < Max iteration)

	while (goal is not achieved)
4.	Calculate next all action-states total distance
5.	if (total distance < previous lowest distance)
6.	Replace previous lowest distance with total distance
7.	Reset stuck status
8.	else
9.	stuck status = stuck status + 1
10.	end if
11.	if (stuck status >= 4)
12.	if (one of the Q-value of possible next states > 0)
13.	Select an action, a within the available actions in the current state according to the highest Q-value in the next state (randomly choose one if more than 1 state is having Q-values which is highest)
14.	else
15.	Select an action, a randomly within the available actions in the current state
16.	end if
17.	else
18.	Select an action, a within the available actions in the current state according to the lowest total distance
19.	end if
20.	Perform the selected action, a and reward or penalty, r will be given
21.	Update the Q-value using Equation (1)
22.	Move the state to new state, s'
	end while
23.	Move virtual target to robot last step's position using Equations (5) and (6)
	end while
24.	end while

3.4 Experimental Setup

The proposed IQL was compared with CQL in order to observe the performance of IQL. Simulation was done for both algorithms in four distinct maps by using MATLAB software. Some of the maps were referred from [30]. Table 3 summaries the setup of the simulation, including the characteristics of maps and the used parameters in both algorithms.

Table 3 - Setup of simulation and parameters for both algorithm

Parameter	Value
Map size	20x20 unit
Initial location	(1,20)
Target location	(20,1)
α	0.1
γ	0.7
Iteration	300
Number of run	30

Both algorithms are evaluated based on the following criteria:

1. The time used to finish each run
2. The total distance used to finish each run
3. The shortest distance used in 30 runs

4. Results And Discussions

This subsection discusses the optimal paths found by IQL and CQL in four different maps. The optimal path was determined by selecting the result with the shortest distance in 30 runs.

4.1 Map 1

Map 1 is made up of fifteen equal sized square shaped obstacles, as shown in Figure 7. Multiple narrow paths existed in between the obstacles due to tight arrangement of obstacles. Passing through these narrow paths has thus become a challenge for the robot with condition of avoiding collision with obstacles.

By observing paths planned by both algorithms, both paths have similarity in terms of general shape. However, the path planned by CQL had sharp turnings around the starting point and target point. Thus, the robot suffered drastic changes in acceleration at both points for CQL that caused jerking.

Table 4 presents the comparison between CQL and IQL in terms of time used, shortest distance used and total distance used for map 1 in 30 runs. Referring to Table 4, it can be seen that the IQL was able to achieve average improvement of 94.85% in map 1 in terms of the time used to find the optimal path. The large improvement in time taken showed that metric of distance was able to guide the robot towards the target effectively.

Other than time used, the average shortest distance attained by IQL has been improved by 29.69% compared to CQL. Moreover, it can be seen that the shortest path found by the IQL was the same for all simulations (standard deviation of 0). This is due to no dead end that existed in map 1, therefore stuck mode has not been activated in all 30 runs. The absence of random selection of next state behaviour in stuck mode eliminated the fluctuation in the path taken.

The total distance taken by the IQL to reach the target position has been improved by 99.50% compared to CQL. The significant reduction in total distance confirmed the effectiveness of metric of distance in reducing randomized motion and guiding robot towards target once again. Apart from that, zero standard deviation of IQL occurred in evaluation of total distance used due to there is no local minima in map 1 environment.

In comparison with the attained improvements for time taken and total distance, the achieved improvement for the shortest distance was the lowest for IQL. This might due to the paths found by CQL and IQL were similar in this map.

4.2 Map 2

Figure 8 shows the optimal path obtained by CQL and IQL for map 2. Map 2 consists of three collateral walls located in opposite positions. Consequently, such arrangement forced the robot to travel in multiple S-shaped paths.

From Figure 8, it can be seen that the path formed by CQL was curvy, and had several U-turns at (5,17) and (14,9). This was due to the existence of dead end at the first wall and the third wall. On the other hand, the IQL was able to bypass both dead ends smoothly, and travelled in the path mostly made up of straight line. The path generated by the IQL truly presented the usefulness of the moving target concept in bypassing the dead end. The robot was able to travel from edge to edge of obstacles regardless of existence of dead end.

Table 5 presents the comparison between CQL and IQL in terms of time used, the shortest distance used and the total distance used for map 2 in 30 runs. In terms of the time used, the CQL consumed more time to reach the target position, which was 0.7958s in average. The less satisfactory performance of CQL may be due to large amount of free spaces that existed in map 2 compared to other maps full of obstacles. When the free space increased, the CQL had more available next states to consider. The worst was the computational complexity that increased exponentially as the free space increased. On the other hand, the IQL used 0.2808s to complete this map in average, with improvement of 64.72% compared to CQL.

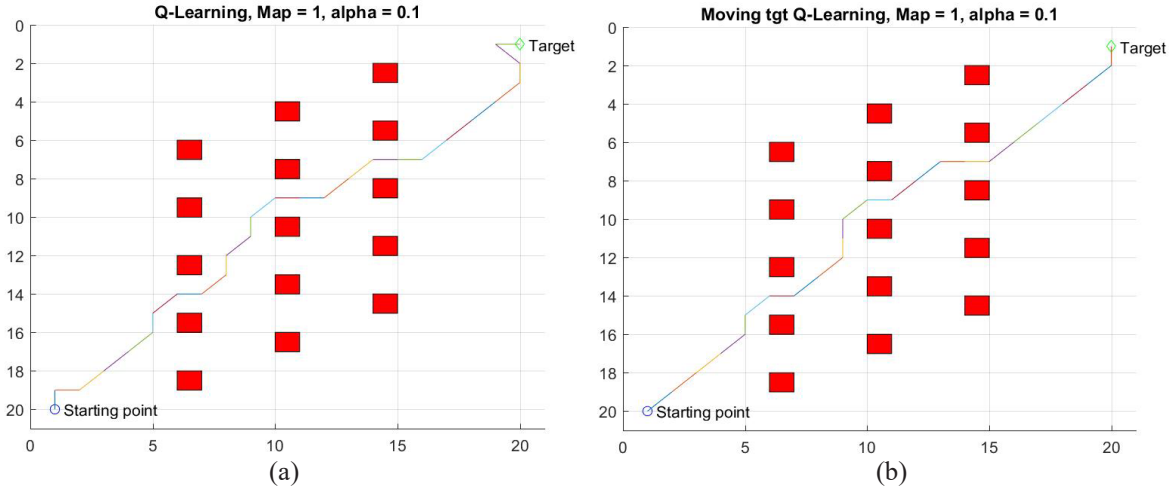


Fig. 7 - Optimal path planned by (a) CQL and (b) IQL for map 1

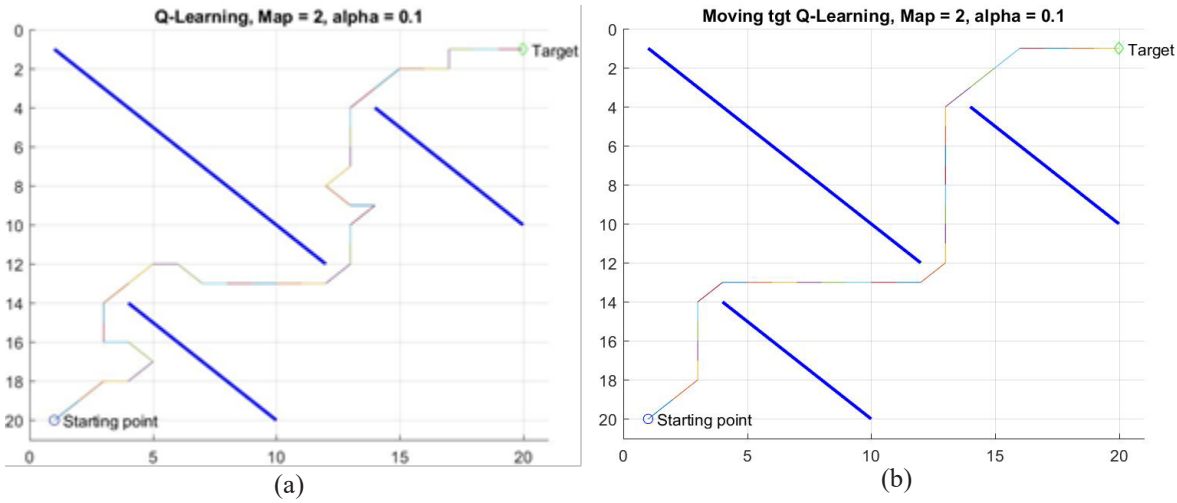
Table 4 - Comparison of both algorithms for map 1

Evaluation	Time used			Shortest distance used			Total distance used		
Algorithm	CQL	IQL	Improve-ment (%)	CQL	IQL	Improve-ment (%)	CQL	IQL	Improve-ment (%)
Total (30)	12.7778	0.6585	94.85	1246.4449	876.3961	29.69	2131311	10681	99.50
Minimum	0.3044	0.0140	95.41	31.7990	29.2132	8.13	50708	356	99.30
Average	0.4259	0.0219	94.85	41.5482	29.2132	29.69	71044	356	99.50
Maximum	0.6171	0.0599	90.30	49.3553	29.2132	40.81	91550	356	99.61
S.D.	0.0785	0.0093	88.21	3.9520	0.0000	100.00	10540	0	100.00

While that, for the shortest distance and the total distance used, the IQL had made an average improvement of 26.86% and 88.38%, respectively. The improvement was in terms of the shortest distance that remained low compared to the time used and the total distance. However, the improvement in total distance used remained high, with the improvements of 84.81-90.41% attained.

4.3 Map 3

In map 3, multiple walls were used to form wall traps that divided the map into several parts although large gaps were provided for the robot to access from one part to another. The optimal paths generated by both algorithms for map 3 are shown in Figure 9. The path planned by CQL was curvy, and had a few sharp 90° edges such as edge located at (15,10) and another edge located at (11,12), while the path planned by the IQL was mostly made up of straight lines such as straight line located at (5,16) and only one sharp 90° edge at (15,10). A curvy path indicates that the robot has to travel further in order to reach the destination. This can be observed in Table 4 that CQL used 43.4567 units in average in terms of the shortest distance used compared to IQL which used 35.5081 units in average.



(a) (b)

Fig. 8 - Optimal path planned by (a) CQL and (b) IQL for map 2

Table 5 - Comparison of both algorithms for map 2

Evaluation	Time used			Shortest distance used			Total distance used		
Algorithm	CQL	IQL	Improve-ment (%)	CQL	IQL	Improve-ment (%)	CQL	IQL	Improve-ment (%)
Total (30)	23.8748	8.4230	64.72	1485.3637	1086.3646	26.86	3560361	413873	88.38
Minimum	0.5844	0.2017	65.49	40.3848	33.8995	16.06	88409	8481	90.41
Average	0.7958	0.2808	64.72	49.5121	36.2122	26.86	118679	13796	88.38
Maximum	1.0547	0.3930	62.73	59.3553	39.5563	33.36	157402	19586	87.56
S.D.	0.1217	0.0530	56.48	4.5299	1.1121	75.45	18228	2768	84.81

Table 6 presents the comparison between CQL and IQL in terms of time used, shortest distance used and total distance used for map 3 in 30 runs. The average improvement made by IQL in terms of time used for map 3 was considered low, which is 29.34%. The drop in performance of IQL was due to the dead end located at (20,5). This dead end was different from map 2 as the dead ends in map 2 did not have a border that prevented the robot from escaping from the dead end. The border located at (14,6) in map 3 heightened the difficulty for the robot to escape from the dead end. The robot had to travel in the opposite direction (downward) in order to bypass the border. As a result, the average improvement made by IQL compared to CQL in map 3 was the lowest among all maps in all evaluations. In spite of that, no improvement (-40.40%) was made in standard deviation by IQL in map 3. The poor performance of IQL in map 3 for standard deviation may be due to more than one available path towards the target. The IQL may select the path above the 'T' shaped wall or below the 'T' shaped wall. The existence of two available paths produced two different paths with different time used for the robot to complete. The divergence in path occurred when the virtual target was moving. When the path diverged, different dead ends were met by robot at different locations. Although IQL triggered the stuck mode wherever it met a dead end, the next state was selected randomly for a new dead end as stated in Section 3.2. This is because the surrounding Q-values were zero for new dead end. This randomized motion might take longer time to escape from the dead end.

Aside from average improvement of time used, the average improvement in terms of the shortest distance was merely 18.29%, and the average total distance used was 75.76%. Through observation for trend of improvement in terms of the shortest distance in map 1 and map 2, it was not surprising that the improvement remained low. While for the improvements made by the IQL for the total distance used in map 3, even though the improvement was not as significant as in other maps, it was still acceptable, with improvement of 75.76% attained for all 30 runs.

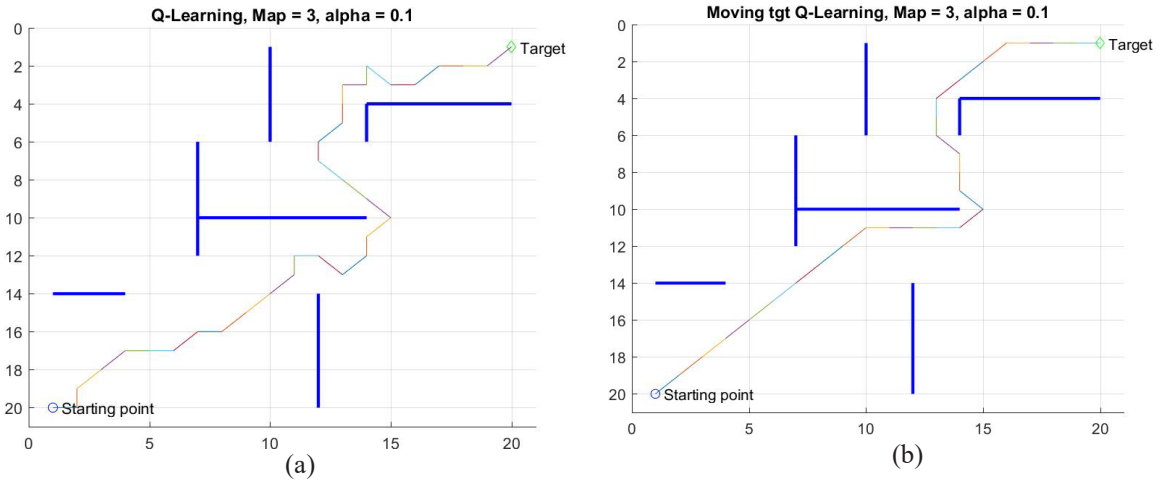


Fig. 9 - Optimal path planned by (a) CQL and (b) IQL for map 3

Table 6 - Comparison of both algorithms for map 3

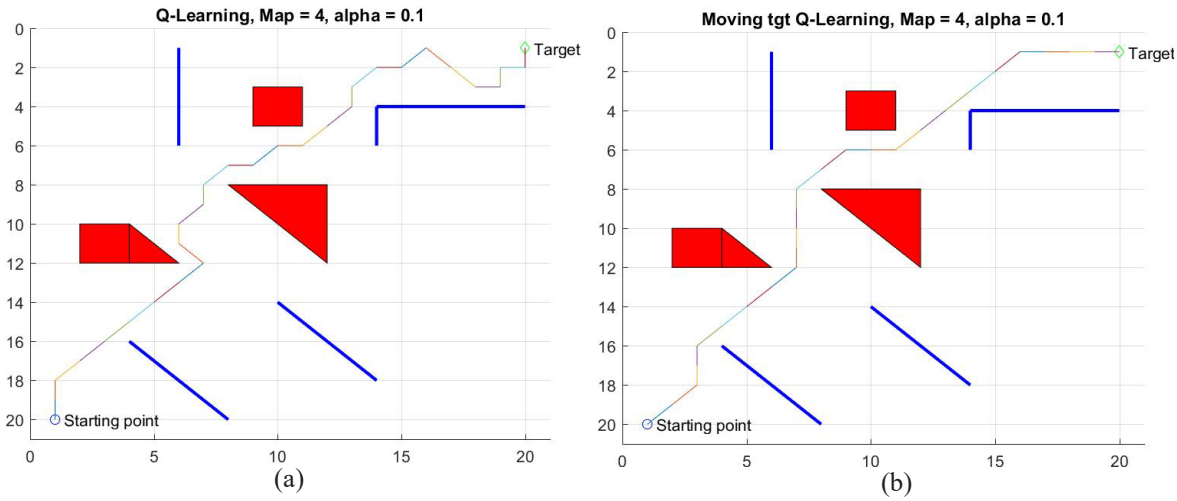
Evaluation	Time used			Shortest distance used			Total distance used		
Algorithm	CQL	IQL	Improve-ment (%)	CQL	IQL	Improve-ment (%)	CQL	IQL	Improve-ment (%)
Total (30)	22.5841	15.9585	29.34	1303.6997	1065.2418	18.29	333733 6	80913 4	75.76
Minimum	0.4518	0.2255	50.08	38.6274	33.2132	14.02	67110	11309	83.15
Average	0.7528	0.5320	29.34	43.4567	35.5081	18.29	111245	26971	75.76
Maximum	1.0424	0.9925	4.78	50.2843	39.2132	22.02	156567	50742	67.59
S.D.	0.1417	0.1990	-40.40	3.1936	1.5855	50.35	21030	10330	50.88

4.4 Map 4

Map 4 has a few similar wall traps as in map 3 with addition of several different obstacles. As the maps were alike, same problems were encountered by the CQL. The paths generated by the CQL were curvy, and had a few sharp 90° edges such as in edge located at (12,12), while the paths planned by IQL were mostly made up of straight line. In fact, there was no sharp 90° edge for the IQL in map 4. However, as opposed to map 3, the IQL was able to achieve better average improvement in map 4. Even though the existence of the same dead end located at (14,6) remained, the IQL took the upper path when avoiding obstacles in map 4. By maneuvering the robot to the upper path, the dead end can be avoided and bypassed easily, thus, saving time and travelled distance in order to escape from the dead end.

Table 7 presents the comparison between CQL and IQL in terms of time used, shortest distance used and total distance used for map 4 in 30 runs. Although the improvement in time used was attained with the average improvements of 75.81%, the same problem was faced by IQL for map 4 with -9.76% of improvement in standard deviation. For map 4, the robot can move towards the target point by taking the path above the triangle obstacle located at (11,9) or taking the path below the triangle obstacle. Diverging in path taken produced a huge difference in time taken. The differences can be observed in Table 7 when IQL was able to complete the run with minimum time of 0.0284 (without diverging path), but with maximum of 0.4578 (with diverging path).

Aside from that, the improvement in terms of the shortest distance by IQL was 25.64% in map 4. The low improvement was within expectation. In spite of that, zero standard deviation in finding the shortest distance used was also experienced by IQL in map 4, similar to map 1. Even though there was a dead end in map 4 located at (15,5), the final path (upper path) selected in each run by IQL did not go through the dead end. Thus, the shortest path for each run did not fluctuate. As for the total distance used, 92.17% improvement was made by IQL in map 4. The improvement was more than 90%



(a) (b)

Fig. 10 - Optimal path planned by (a) CQL and (b) IQL for map 4

Table 7 - Comparison of both algorithms for map 4

Evaluation	Time used			Shortest distance used			Total distance used		
Algorithm	**CQL**	**IQL**	**Improve-ment (%)**	**CQL**	**IQL**	**Improve-ment (%)**	**CQL**	**IQL**	**Improve-ment (%)**
Total (30)	19.3734	4.6872	75.81	1225.8175	911.5433	25.64	2811919	220049	92.17
Minimum	0.4427	0.0284	93.57	34.6274	30.3848	12.25	64626	1064	98.35
Average	0.6458	0.1562	75.81	40.8606	30.3848	25.64	93731	7335	92.17
Maximum	0.9453	0.4578	51.57	48.5269	30.3848	37.39	120041	22478	81.27
S.D.	0.1064	0.1168	-9.76	3.6700	0.0000	100.00	14402	5692	60.48

5. Conclusion

To alleviate the slow convergence of CQL, distance of metric and moving target concept are proposed in the IQL. Both algorithms are simulated and compared in four different environments full of obstacles with various shapes. The results indicate that IQL outperforms the CQL. The average improvements of time used and shortest distance used ranging from 29.34% to 94.85% and from 18.29% to 29.69% are made by IQL, respectively. Besides that, the outstanding average improvements are achieved by IQL for total distance used ranging from 75.76% to 99.50%.

Other than that, future research will attempt to verify the simulation results in real-world mobile robot path planning.

Acknowledgement

The authors would like to express the deepest appreciation to Universiti Tun Hussein Onn Malaysia, for funding this project through the GPPS (Vot H034). Additional support from Ministry of Higher Education Malaysia in the form of Fundamental Research Grant Scheme (FRGS/1/2018/ICT02/UTHM/02/2 Vot K070) is also gratefully acknowledged.

References

[1] L. Sabattini, V. Digani, C. Secchi, G. Cotena, D. Ronzoni, M. Foppoli and F. Oleari, "Technological roadmap to boost the introduction of agvs in industrial applications," In: Intelligent Computer Communication and Processing (ICCP), 2013 IEEE International Conference on: IEEE, pp 203-8, 2013

[2] Y. Song-hua, "Trajectory Tracking and Control of Logistics AGV [J]," Modular Machine Tool & Automatic Manufacturing Technique, vol. 6, pp. 022, 2008

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L.D. Jackel, M. Monfort, U. Muller and J. Zhang, "End to end learning for self-driving cars," arXiv preprint arXiv:1604.07316, 2016

[4] S. Thrun, "Toward robotic cars," Communications of the ACM, vol. 53, pp. 99-106, 2010

[5] C. Häne, T. Sattler and M. Pollefeys, "Obstacle detection for self-driving cars using only monocular cameras and wheel odometry," In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on: IEEE, pp 5101-8, 2015

[6] C. Hofner and G. Schmidt, "Path planning and guidance techniques for an autonomous mobile cleaning robot," In: Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on: IEEE, pp. 610-7, 1994

[7] T. Aasen, Mobile cleaning robot for floors. Google Patents, 2005

[8] X. Liu and D. Gong, "A comparative study of A-star algorithms for search and rescue in perfect maze," In: Electric Information and Control Engineering (ICEICE), 2011 International Conference on: IEEE, pp 24-7, 2011

[9] T. Dean, K. Basye and J. Shewchuk, "Reinforcement learning for planning and control," Machine learning methods for planning, pp. 67-92, 1992

[10] C.J. Watkins and P. Dayan, "Q-learning," Machine learning, vol. 8, pp. 279-92, 1992

[11] J. Xiao, Z. Michalewicz, L. Zhang and K. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," IEEE transactions on evolutionary computation, vol. 1, pp. 18-28, 1997

[12] A. Konar, I.G. Chakraborty, S.J. Singh, L.C. Jain and A.K. Nagar, "A deterministic improved Q-learning for path planning of a mobile robot," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 43, pp. 1141-53, 2013
[13] A. Konar, I. Goswami, S.J. Singh, L.C. Jain and A.K. Nagar, "A Deterministic Improved Q-Learning for Path Planning of a Mobile Robot," IEEE Trans. Systems, Man, and Cybernetics: Systems, vol. 43, pp. 1141-53, 2013
[14] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L.C. Jain and A.K. Nagar, "Realization of an adaptive memetic algorithm using differential evolution and q-learning: a case study in multirobot path planning," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 43, pp. 814-31, 2013
[15] D-H. Kim, Y-J. Kim, K-C. Kim, J-H. Kim and P. Vadakkepat, "Vector field based path planning and Petri-net based role selection mechanism with Q-learning for the soccer robot system," Intelligent Automation & Soft Computing, vol. 6, pp. 75-87, 2000
[16] C. Chen, H-X. Li and D. Dong, "Hybrid control for robot navigation-a hierarchical Q-learning algorithm," IEEE Robotics & Automation Magazine, vol. 15, 2008
[17] H. Xiao, L. Liao and F. Zhou, "Mobile robot path planning based on q-ann," In: Automation and Logistics, 2007 IEEE International Conference on: IEEE, pp. 2650-4, 2007
[18] E.S. Low, P. Ong and K.C. Cheah, "Solving the optimal path planning of a mobile robot using improved Q-learning," Robotics and Autonomous Systems, vol. 115, pp. 143-61, 2019
[19] M. Guo, Y. Liu and J. Malec, "A new Q-learning algorithm based on the metropolis criterion," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 34, pp. 2140-3, 2004
[20] P. Das, H. Behera and B. Panigrahi, "Intelligent-based multi-robot path planning inspired by improved classical Q-learning and improved particle swarm optimization with perturbed velocity," Engineering Science and Technology, an International Journal, vol. 19, pp. 651-69, 2016
[21] C-F. Juang and C-M. Lu, "Ant colony optimization incorporated with fuzzy Q-learning for reinforcement fuzzy control," IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol. 39, pp. 597-608, 2009
[22] P. Muñoz, R. Barco and I. de la Bandera, "Optimization of load balancing using fuzzy Q-learning for next generation wireless networks," Expert systems with applications, vol. 40, pp. 984-94, 2013
[23] M. Khajenejad, F. Afshinmanesh, A. Marandi and B.N. Araabi, "Intelligent particle swarm optimization using Q-learning," In: Proc. IEEE Swarm Intell. Symp: Citeseer, pp 7-12, 2006
[24] P. Rakshit, A. Konar, S. Das and A.K. Nagar, "ABC-TDQL: An adaptive memetic algorithm," In: Hybrid Intelligent Models and Applications (HIMA), 2013 IEEE Workshop on: IEEE, pp. 35-42, 2013
[25] C. Li, J. Zhang and Y. Li, "Application of artificial neural network based on q-learning for mobile robot path planning," In: Information Acquisition, 2006 IEEE International Conference on: IEEE, pp. 978-82, 2006
[26] M. Duguleana and G. Mogan, "Neural networks based reinforcement learning for mobile robots obstacle avoidance," Expert Systems with Applications, vol. 62, pp. 104-15, 2016
[27] B-Q. Huang, G-Y. Cao and M. Guo, "Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance," In: Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on: IEEE, pp. 85-9, 2005
[28] K-H. Park, Y-J. Kim and J-H. Kim, "Modular Q-learning based multi-agent cooperation for robot soccer," Robotics and Autonomous Systems, vol. 35, pp. 109-22, 2001
[29] K-S. Hwang, S-W. Tan and C-C. Chen, "Cooperative strategy based on adaptive Q-learning for robot soccer systems," IEEE Transactions on Fuzzy Systems, vol. 12, pp. 569-76, 2004
[30] Z. Yijing, Z. Zheng, Z. Xiaoyi and L. Yang, "Q learning algorithm based UAV path learning and obstacle avoidence approach," In: Control Conference (CCC), 2017 36th Chinese: IEEE, pp. 3397-402, 2017