International Journal of Engineering & Technology, 7 (4.27) (2018) 57-62

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper



Mobile Robot Path Planning using Q-Learning with Guided Distance

Ee Soong Low[1], Pauline Ong[2]*, Cheng Yee Low[3]

[1 2 3] *Faculty of Mechanical and Manufacturing Engineering, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia*
**Corresponding author E-mail: ongp@uthm.edu.my*

Abstract

In path planning for mobile robot, classical Q-learning algorithm requires high iteration counts and longer time taken to achieve convergence. This is due to the beginning stage of classical Q-learning for path planning consists of mostly exploration, involving random direction decision making. This paper proposed the addition of distance aspect into direction decision making in Q-learning. This feature is used to reduce the time taken for the Q-learning to fully converge. In the meanwhile, random direction decision making is added and activated when mobile robot gets trapped in local optima. This strategy enables the mobile robot to escape from local optimal trap. The results show that the time taken for the improved Q-learning with distance guiding to converge is longer than the classical Q-learning. However, the total number of steps used is lower than the classical Q-learning.

Keywords: *Guided distance, Mobile robot, Path planning, Q-learning, Reinforcement learning.*

1. Introduction

Mobile robots are used in many fields and applications such as cleaning robots [1, 2], logistics automation robots (automated guided vehicles - AGVs) in factory [3, 4] and self-driving vehicles [5-7]. In order for these robots to operate well, paths are created for these robots to track. A well designed path enables mobile robot to reach the destination point faster and safer. This increases the overall efficiency of mobile robots in transporting goods or items. Therefore, path planning has become an essential study for researchers to discover.

Path planning is a complex task in mobile robot field. It provides a path for mobile robot to move from one point to another point without colliding with any obstacles [8]. Recently, many studies have been done for applying Q-learning in path planning and navigation for mobile robots [9-17]. Q-learning is a simple and basic reinforcement learning used to learn the strategy in obtaining better result through repetitive procedure and iterations. Q-learning works based on penalty and rewards for every action the robots made. All these three parameters (rewards, penalty and actions) will affect the Q-value. As the iterations continue, the Q-value will improve and lean to better results. As an offset of its simplicity, it suffers from several limitations and weaknesses. The time taken to determine the optimal path using Q-learning increases when it is applied in real world path planning [12]. The incompatibility of Q-learning is due to the increase of search space. Moreover, environment with moving obstacles will lead to higher computational complexity and longer reset time.

In this regard, various improved Q-learning models were introduced, in which improving the performance of Q-learning by applying the optimization algorithm as modifier [14, 18-22] was one of the proposed solutions. Das applied the improved particle swarm optimization (IPSO) with perturbed velocity in Q-learning algorithm in order to improve its global search ability and convergence rate [19]. The results showed that the robots were able to perform well even several obstacles were considered in the working environment. Artificial bee colony (ABC) optimization algorithm has been used to do the global search in Q-learning [23]. As a result, the improved Q-learning model was able to solve multi robot navigation problem effectively. On the other hand, adaptive memetic algorithm (AMA) using ABC optimization algorithm was used to improve the Q-learning model, in terms of runtime and accuracy [23]. The same AMA concept was applied to Q-learning, but by using differential evolution (DE) algorithm [14]. The results showed that the combination of DE with Q-learning outperformed particle swarm optimization (PSO) and genetic algorithm (GA) for all three evaluation metrics. Metropolis criterion from simulated annealing (SA) algorithm was used to control the exploration and exploitation of Q-learning algorithm [18]. Then SA-Q-learning algorithm was able to converge faster while preventing degradation in performance caused by excessive exploration.

In addition to optimization algorithm, artificial neural network (ANN) has been used to improve the performance of Q-learning in mobile robot path planning in numerous studies [17, 24]. In [25], Pos-Net neural network was used to determine the next state or direction of mobile robot based on the obstacle detection, current state and current time as input. The mobile robot was able to navigate through environment full with static and dynamic obstacles, while providing good convergence ratio and avoiding local minima. Besides that, back-propagation neural network was used to calculate Q-values for every action in the current state [26]. The learning rate was improved significantly as compared to classical Q-learning, while at the same time, providing ability to avoid obstacle in path planning.

Other than artificial intelligence algorithms, adding distance or global information as consideration factor is crucial step in enhancing performance of Q-learning algorithm. In [27, 28], along with implementation of adaptive Q-learning for [27] and modular Q-learning for [28] in robot soccer system, distance between ball with each robot will be computed in order to select the robot with

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

the minimum distance with the ball. The selected robot will be chosen as attacker while other robots will be defenders or side kickers. This enable role switching among soccer robots in order to achieve better cooperation and coordination in the soccer system. As a result, the tendency of scores of the implemented algorithm is rising as opposed to opponent team which is dropping. While in path planning, Euclidean distance between robots' current position and target position and Euclidean distance among robot with other robots had been used to form the overall fitness function [19]. In this case study, dynamic obstacles are considered, as robots act as dynamic obstacles among each other. The fitness function is optimized by improved particle swarm optimization with differentially perturbed velocity (IPSO-DV) and further synergized with improved Q-learning to form QIPSO-DV path planning algorithm. The proposed algorithm was able to reduce space complexity and time consumed as compared to classical Q-learning (CQL). The statement is proven through performance of QIPSO-DV in achieving smallest average total trajectory path traveled (ATTPT) which contributes to smoother path with lower deviation in direction and fastest ending time relative to other algorithms (IPSO-DV, PSO and CQL).

In this paper, Q-learning model is improved by including the metric of distance into the direction decision making during the exploration and exploitation. This improvement enables Q-learning model to determine the direction that is closest to the target point, contributing to faster convergence rate.

The outline of the paper is as follows. The introduction of Q-learning is first presented. Subsequently, formulation and implementation of metric of distance, are discussed. Simulation and discussion of the proposed Q-learning with guided distance (QL-GD) are presented and lastly, the findings of this study are concluded.

1.1. Classical Q-Learning

Classical Q-learning (CQL) is a type of model-free reinforcement learning developed by Watkins [10]. CQL is performed through an agent executes an action in an environment. The action is evaluated and presented in the form of reward or penalty as the feedback to CQL. The reward or penalty is stored in the form of Q-value in Q-table. These processes are repeated until the goal is achieved. The CQL learns consistently in order to provide optimal results by gaining experience from time to time when the agent moves through the same environment multiple times.

The formula used to update the Q-value in the Q-table is given in (1), while the pseudocode of Q-learning algorithm is summarized in Algorithm 1.

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

Definition 1.1: $Q(s_i,a_i)$ is the Q-value at the state, s_i that performed an action, a_i at the time i. α is the learning rate, γ is the discount factor and r_i is the reward or penalty applied after the action a_i is performed. $\max_a Q(s_{i+1},a_{i+1})$ is the maximum value of Q-values among the all the available actions at time instance $i+1$.

For CQL, the directional decision making criteria is based on the maximum value of Q-values among all the available actions in the next state. If more than one action in the next state have the same highest Q-value, the CQL just randomly pick one among those actions with the highest Q-value. In this strategy, it is difficult for the Q-learning algorithm to converge to the solution or targeted position. This is due to in the beginning stage, all the Q-values are initially zero. Such selection strategy causes a random action or movement of mobile robot in the beginning. To improve the convergence rate, metric of distance is added into the directional decision making criteria during the exploration and exploitation of the CQL algorithm in this study.

Algorithm 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 (1)
7. Move the state to new state, s'
8. end while

2. Methodology

In this section, Q-learning with guided distance (QL-GD) is introduced. QL-GD takes distance between robot and target position as consideration in order to achieve faster convergence rate. On the other hand, an escape mechanism is added as an effort to enable the robot to escape from local minima.

2.1. Formulation of Metric of Distance

Several assumptions and principles are made during the formulation of metric of distance as follows:

1. The information of current coordinate and target position are known.
2. When moving to the next state, the robot has eight available actions to be selected, specifically: (i) North, (ii) South, (iii) East, (iv) West, (v) North-east, (vi) North-west, (vii) South-east and (viii) South-west
3. The movement and position of the mobile robot are grid-based motion. The robot can only move grid by grid in a Cartesian plane. For each action, the robot is able to move one unit in x-axis and one unit in y-axis in either direction.

The total distance is the summation of distance between the current position with next state position, and distance between next state position with target position. The formula of total distance is expressed as:

$$\text{distance}_{total} = \text{distance}_{curr-next} + \text{distance}_{next-t\arg} \quad (2)$$

Definition 2.1: distance_{total} is total distance, $\text{distance}_{curr-next}$ is the distance between the current position with next state position, and $\text{distance}_{next-t\arg}$ is the distance between next state position with target position.

The $\text{distance}_{curr-next}$ and $\text{distance}_{next--t\arg}$ are expressed as:

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

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

Definition 2.2: x and y represent the x-axis and y-axis positions, $next$ represents next position and $t\arg$ represents target position.

The concept of distance is presented graphically in Fig. 1.

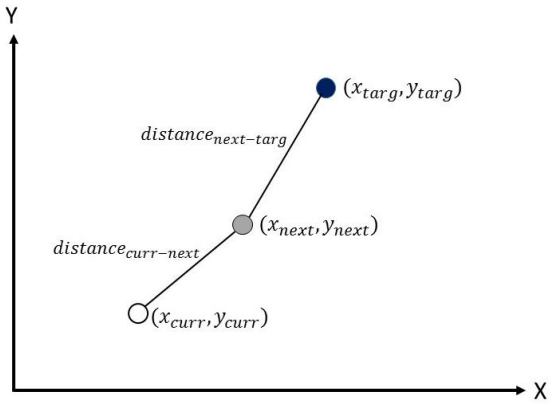

Fig. 1: The concept of distance

2.2. Implementation of the Metric of Distance into Classical Q-Learning

For the proposed QL-GD, the selection of the next state will not be based on the maximum Q-value among all the action-states as in CQL. There are two modes for the QL-GD, which are:
1. Default mode: distance aspect will take into consideration
2. Stuck mode: escape mechanism will be triggered.
For default mode, the selection for the next action will be based on the minimum total distance among all the action-states. Thus, total distance of all possible action-states for the current state will be calculated, and the action-states with the lowest total distance will be selected.
However, with the implementation of distance aspect, the algorithm tends to converge to local minima easily when a dead end is detected. Therefore, a mechanism is added to escape from local minima. In order to detect the occurrence of dead end, all calculated total distance for all states are compared and the lowest total distance is stored. Intuitively, as the iteration continues, the robot is getting nearer to the target position and hence, lower total distance is obtained. Thus, if the total distance of the next state is higher than the lowest total distance stored previously, the robot is considered in a dead end. In this study, if the robot does not obtain a lower total distance than the stored lowest total distance within four iterations, an escape mechanism is triggered. In this case, if the Q-values of all available action-states are zero, the next action is selected randomly. Otherwise, direction selection method of the CQL will be used. Once the improvement in the total distance is observed, the QL-GD resumes.
The pseudocode of the proposed QL-GD is summarized in Algorithm 2.

2.3. Experiment Setup

To evaluate the performance of the proposed QL-GD, the comparison with CQL in mobile robot path planning was conducted. The experiment was carried out in simulated environment with various obstacle patterns in four different maps in MATLAB software by referring [29]. Size of all maps were 20 x 20 unit grid. The starting point of mobile robot is set at (1,20) and target point is set at (20,1).
For both algorithm, the α constant is 0.1 while the γ constant is 0.7. The maximum iteration for each run is 300 iterations. Total of 30 runs were simulated for each map for both algorithms. The performance of algorithms was evaluated based on:
1. The time taken to complete each run
2. Total number of steps used to complete each run
3. The lowest number of steps used in 30 runs
The number of steps is calculated including rotational motion and displacement. Each rotation of 90° is considered one step. For displacement, every transition of state is considered one step.

Algorithm 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 or 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 (1)
22. Move the state to new state, s'
23. end while

3. Results and discussions

3.1. Graphical Illustration of Path Planned

Fig. 2 to Fig. 5 and Fig 6 to Fig. 9 represent the optimal paths obtained by CQL and QL-GD, which have the lowest number of steps taken for map 1 to map 4, respectively.
Fig 6 shows that the QL-GD was able to find the optimum path by passing through various obstacles in the middle of map, while for the CQL, some curvy path are observed in Fig. 2, leading to longer travelled distance. From Fig. 3 to Fig. 5, it can be observed that paths planned by the CQL were constantly changing in direction, whereas Fig. 7 to Fig. 9 show that the paths planned by the QL-GD were less likely to change direction and hence, forming multiple straight lines. The curvy characteristic can be clearly detected by comparing QL-GD and CQL in map 3, as shown in Fig. 8 and Fig. 4, respectively. For the QL-GD, the path is made up of several long straight lines with few short lines in order to steer the robot towards the target position. While for the CQL, the path is made up of many short lines connecting starting point to target. The QL-GD outperformed CQL in this case, since constantly steering direction will lead to higher number of steps and cause real world application of robot to waste more energy and time. Besides that, QL-GD fails to handle the dead end properly, which can be observed in Figure 7 and Figure 9, at the position of (15,6) and (9,10), respectively. A right angle is formed at this point, leading to longer travelled distance since length of hypotenuse is shorter than total length of both sides on a triangle.
Table 1 shows the total time taken for 30 runs, minimum and maximum time taken among 30 runs, average and standard deviation of time taken for 30 runs for both algorithms in all maps. Table 2 presents the total number of steps used by both algorithms and Table 3 shows the lowest number of steps used by both algorithms among 30 runs.

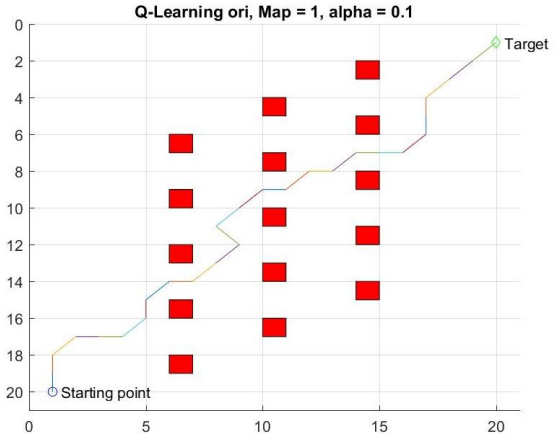

Fig. 2: Path planned by classical q-learning for map 1

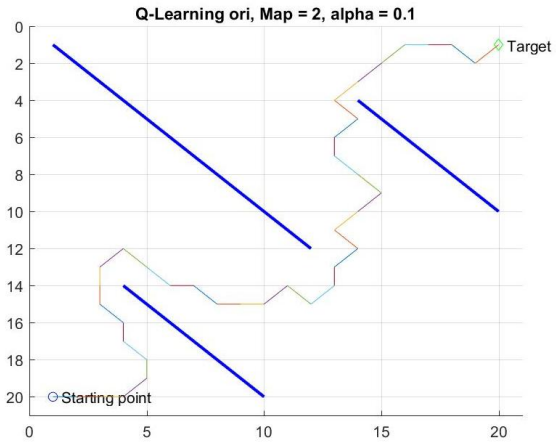

Fig. 3: Path planned by classical q-learning for map 2

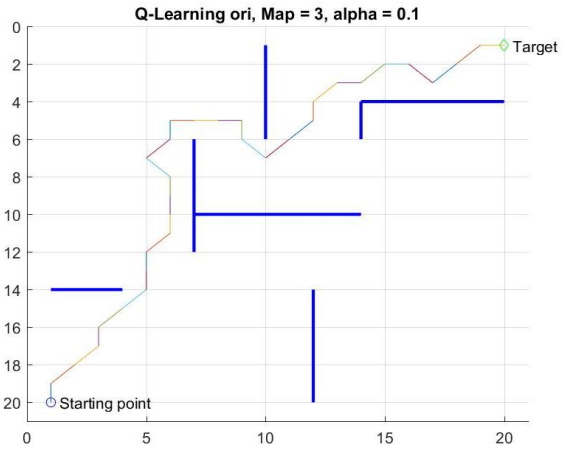

Fig. 4: Path planned by classical q-learning for map 3

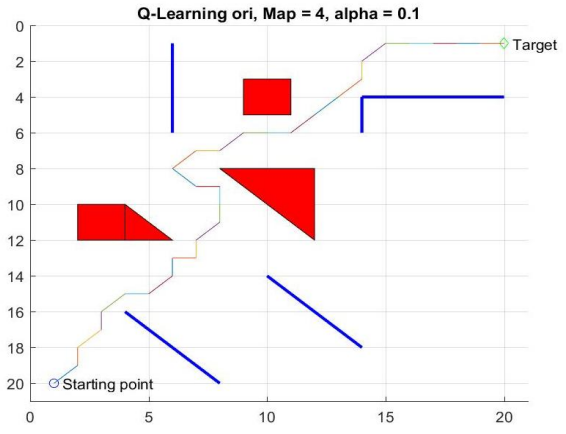

Fig. 5: Path planned by classical q-learning for map 4

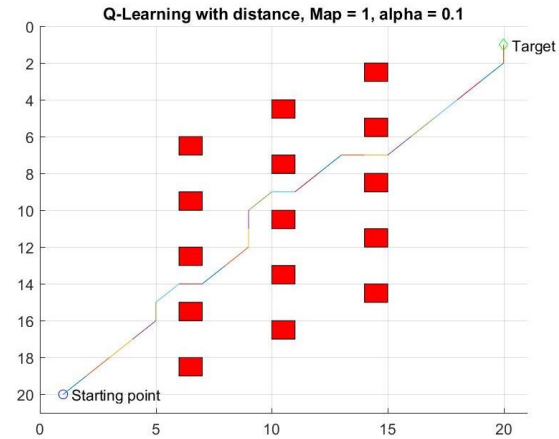

Fig 6: Path planned by q-learning with guided distance for map 1

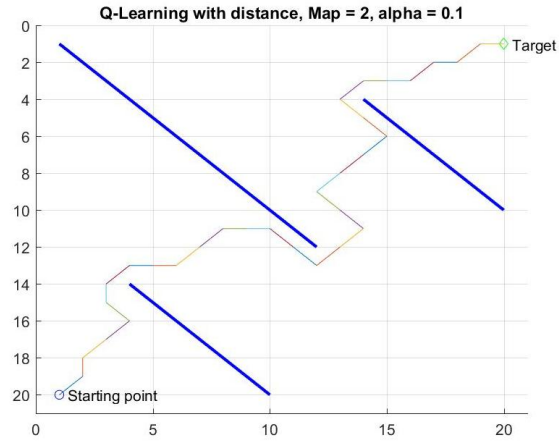

Fig. 7: Path planned by q-learning with guided distance for map 2

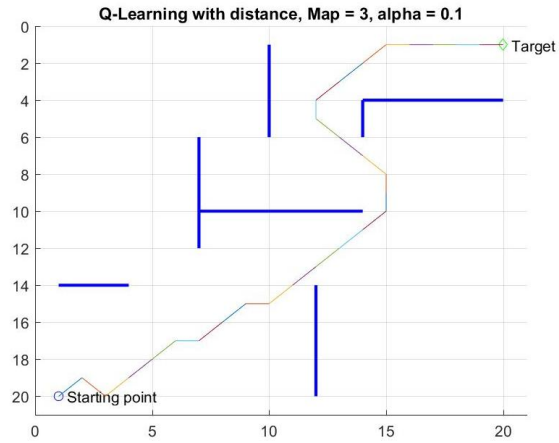

Fig. 8: Path planned by q-learning with guided distance for map 3

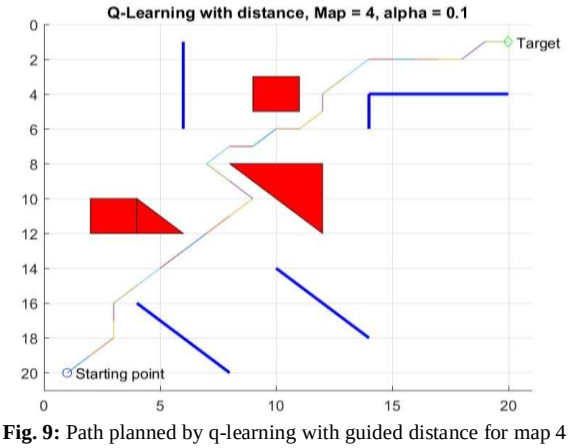

Fig. 9: Path planned by q-learning with guided distance for map 4

As shown in Table 1, for map 1, the QL-GD was able to reach the target position within shorter period, where an improvement of 65% was observed for the average time taken. However, the proposed QL-GD took longer time than the CQL to complete 30 runs for all evaluations (except SD) for maps 2, 3 and 4. It can be observed that the performance of QL-GD is the worst for map 3, which in average, the time taken to complete the path planning is 122% slower than the CQL, followed by map 2 (92%) and map 4 (83%). This is due to the QL-GD needs to calculate the total distance for all possible action-states at each instance of time, in addition to computing the Q-values as in the CQL. For the CQL, only the latter is required and then the maximum Q-value is identified. However, the QL-GD shows lower standard deviation than the CQL, which leads to better consistency in performance. The highest improvement made on standard deviation is on map 1 (75%) while no improvement is observed for map 4. Also, it is pertinent to note that the time taken for CQL in real world application may be longer, since curvy paths are generated. The rotation of robot body and movement of robot from one point to another point in actual world may spend longer time than simulation.

In Table 2, the total number of steps of taken by the QL-GD is lower than the CQL for all maps with significant improvement ranging from 27% (map 3) to as much as 91% (map1). CQL was suffered from high total number of steps due to random motion of mobile robot in the initial stage, due to lack of global knowledge of environment. On the other hand, the QL-GD was able to navigate better in the initial stage due to guidance from distance. Based on improvements from minimum, average and maximum evaluations, it can be observed that generally QL-GD made better improvements on maximum total number of steps instead of improvement on average and minimum total number of steps. This can be clearly observed for map 2, where the improvement made for minimum number of total steps was merely 7%, while 34% and 52% improvements were made for average and maximum number of total steps, respectively. This indicates that the QL-GD is able to guide the robot properly and reduce large amount of unnecessary number of steps when the robot is not heading to the target. While marginal improvement in terms of minimum number of total steps showed that the QL-GD is still able to use lower number of steps when the robot is heading to the target, but with trivial improvement. Besides that, QL-GD also made improvements from standard deviation (SD) evaluation. These enhancements were generally high, from the lowest is 49% for map 4 to the highest is 100% for map 1. The zero standard deviation of the QL-GD in map 1 shows that the QL-GD works effectively in environment without local minima or dead end. Improvement on SD indicates that QL-GD has more stable performance than CQL.

Table 3 shows that the QL-GD generally outperformed the CQL in all evaluations for the lowest number of steps taken for map 1 and map 4, except for minimum evaluation for map 1. The statement is supported by minor improvements made by QL-GD in map 4 for all evaluations (total, minimum, average, maximum and SD) ranging from 9% to 15%. On the other hand, for map 1, the major improvements were made on maximum and SD which are 41% and 100%, respectively, while for total and average were 17% for both. These are due to in these maps, the number of dead end is lower or does not exist. While in map 2, the QL-GD failed to find the shortest path which is hypotenuse around area of (7,12) and (12,9), while CQL was able to pass through those areas is a shorter path manner. This causes the poorer performance of QL-GD for all evaluations. In map 3, slight improvements were made by QL-GD which is 4% on total and average, 1% on maximum and 11% on SD, while for minimum is -2%. This is due to QL-GD picked a longer path (path below), leading to higher lowest number of steps while the CQL was able to pick a shorter path (path above).

4. Conclusion

In this study, the QL-GD was proposed in order to improve the performance of the CQL in time taken and number of steps used to complete and navigate through various obstacles in various simulated environment. The results showed that the time taken for the QL-GD was longer in the simulated environment in most of the maps. On the other hand, the QL-GD was able to outperform the CQL in terms of total number of steps used in all maps. The improvement can be clearly observed in Table 2 where QL-GD had average improvement for total number of steps used in term of percentage from 27% to 91% for all tested maps. Moreover, there were also slight improvement for lowest number of steps used from 4% to 17% for all maps, except map 2.

In future, real world experiments will be conducted in order to obtain empirical results to justify the proposed approach

Table 1: Comparison of Time Taken Used for Both Algorithms

	Time taken (s)											
Map	1			2			3			4		
Algorithm	CQL	QL-GD	Improvement (%)	CQL	QL-GD	Improvement (%)	CQL	QL-GD	Improvement (%)	CQL	QL-GD	Improvement (%)
Total (30)	13.0603	4.6326	65%	25.4992	49.0276	-92%	23.6226	52.4531	-122%	21.5030	39.3253	-83%
Minimum	0.3033	0.1399	54%	0.5438	1.4629	-169%	0.4785	1.5084	-215%	0.4230	0.7323	-73%
Average	0.4353	0.1544	65%	0.8500	1.6343	-92%	0.7874	1.7484	-122%	0.7168	1.3108	-83%
Maximum	0.5973	0.2335	61%	1.2886	1.8484	-43%	1.4279	2.2422	-57%	1.0953	1.7723	-62%
SD	0.0859	0.0212	75%	0.1891	0.0963	49%	0.1760	0.1492	15%	0.1485	0.2294	-54%

Table 2: Comparison of Total Number of Steps Used for Both Algorithms

	Total number of steps used (steps)					
Map	1			2		
Algorithm	CQL	QL-GD	Improvement (%)	CQL	QL-GD	Improvement (%)
Total (30)	4982390	468000	91%	8962914	5898930	34%
Minimum	119628	15600	87%	190317	176948	7%
Average	166080	15600	91%	298764	196631	34%
Maximum	201943	15600	92%	462513	222314	52%
SD	22262	0	100%	67132	11420	83%
Map	3			4		
Algorithm	CQL	QL-GD	Improvement (%)	CQL	QL-GD	Improvement (%)
Total (30)	8476490	6152486	27%	7797882	4649417	40%
Minimum	170282	178668	-5%	152741	84212	45%
Average	282550	205083	27%	259929	154981	40%
Maximum	521176	229747	56%	398791	213822	46%

SD	64282	13571	79%	55934	28788	49%

Table 3: Comparison of Lowest Number of Steps Used for Both Algorithms

	Lowest number of steps used (steps)					
Map	1			2		
Algorithm	CQL	QL-GD	Improvement (%)	CQL	QL-GD	Improvement (%)
Total (30)	1875	1560	17%	1875	2982	-59%
Minimum	45	52	-16%	45	78	-73%
Average	63	52	17%	63	99	-59%
Maximum	88	52	41%	88	125	-42%
SD	10.5430	0.0000	100%	10.5430	12.4197	-18%
Map	3			4		
Algorithm	CQL	QL-GD	Improvement (%)	CQL	QL-GD	Improvement (%)
Total (30)	2219	2139	4%	2200	1946	12%
Minimum	57	58	-2%	56	51	9%
Average	74	71	4%	73	65	12%
Maximum	95	94	1%	99	84	15%
SD	10.5421	9.4290	11%	9.9458	8.4394	15%

Acknowledgement

Financial support from Universiti Tun Hussein Onn Malaysia (UTHM) in the form of IGSP Grant Vot U671, RMC, UTHM and GPPS Vot H034 are gratefully acknowledged.

References

[1] Hofner C, & Schmidt G, "Path planning and guidance techniques for an autonomous mobile cleaning robot." pp.610-617.
[2] Aasen T, "Mobile cleaning robot for floors," Google Patents, 2005.
[3] Sabattini L, Digani V, & Secchi C (2013), "Technological roadmap to boost the introduction of agvs in industrial applications." pp.203-208.
[4] Song-hua Y (2008), "Trajectory tracking and control of logistics AGV [J]," *Modular Machine Tool & Automatic Manufacturing Technique*, vol. 6, pp. 022.
[5] Bojarski M, Del Testa D & Dworakowski D (2010), "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.
[6] S. Thrun, "Toward robotic cars," *Communications of the ACM*, vol. 53, no. 4, pp. 99-106.
[7] Häne C, Sattler T, & Pollefeys M, "Obstacle detection for self-driving cars using only monocular cameras and wheel odometry." pp. 5101-5108.
[8] Liu X & Gong D (1992), "A comparative study of a-star algorithms for search and rescue in perfect maze." pp. 24-27.
[9] Dean T, Basye K, & Shewchuk J (1992), "Reinforcement learning for planning and control," *Machine learning methods for planning*, pp. 67-92.
[10] Watkins CJ & Dayan P (1992), "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279-292.
[11] Xiao J, Michalewicz Z & Zhang L (1997), "Adaptive evolutionary planner/navigator for mobile robots," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 18-28.
[12] Konar A, Chakraborty IG & Singh SJ (2013), "A deterministic improved Q-learning for path planning of a mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1141-1153.
[13] Konar A, Goswami I & Singh SJ (2013), "A deterministic improved q-learning for path planning of a mobile robot," *IEEE Trans. Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1141-1153.
[14] Rakshit P, Konar K, Bhowmik P (2013), "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, no. 4, pp. 814-831.
[15] Kim DH, Kim YJ & Kim KC (2000), "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, no. 1, pp. 75-87.
[16] Chen C, Li HX, & Dong D (2008), "Hybrid control for robot navigation-a hierarchical q-learning algorithm," *IEEE Robotics & Automation Magazine*, vol. 15, no. 2.
[17] Xiao H, Liao L, & Zhou F, "Mobile robot path planning based on q-ann." pp. 2650-2654.
[18] Guo M, Liu Y, and Malec J (2004), "A new Q-Learning algorithm based on the metropolis criterion," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 5, pp. 2140-2143.
[19] Das P, Behera H, & Panigrahi B (2016), "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, no. 1, pp. 651-669.
[20] Juang CF, & Lu CM (2009), "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, no. 3, pp. 597-608.
[21] Muñoz P, Barco R, & de la Bandera I (2013), "Optimization of load balancing using fuzzy q-learning for next generation wireless networks," *Expert systems with applications*, vol. 40, no. 4, pp. 984-994.
[22] Khajenejad M, Afshinmanesh F, Marandi A, "Intelligent particle swarm optimization using Q-learning." pp. 7-12.
[23] Rakshit P, Konar A & S. Das S, "ABC-TDQL: An adaptive memetic algorithm." pp. 35-42.
[24] Li C, Zhang J, & Li Y, "Application of artificial neural network based on q-learning for mobile robot path planning." pp. 978-982.
[25] Duguleana M, & Mogan G (2016), "Neural networks based reinforcement learning for mobile robots obstacle avoidance," *Expert Systems with Applications*, vol. 62, pp. 104-115.
[26] Huang BQ, Cao GY, & Guo M, "Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance." pp. 85-89.
[27] Hwang KS, Tan SW, & Chen CC (2004), "Cooperative strategy based on adaptive q-learning for robot soccer systems," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 4, pp. 569-576.
[28] Park KH, Kim YJ & Kim JH (2001), "Modular Q-learning based multi-agent cooperation for robot soccer," *Robotics and Autonomous Systems*, vol. 35, no. 2, pp. 109-122.
[29] Yijing Z, Zheng Z & Xiaoyi Z, "Q learning algorithm based uav path learning and obstacle avoidence approach." pp. 3397-3402.