Expert Systems With Applications 199 (2022) 117191

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Modified Q-learning with distance metric and virtual target on path planning of mobile robot

Ee Soong Low [a], Pauline Ong [a,1,*], Cheng Yee Low [a], Rosli Omar [b]

[a] Faculty of Mechanical and Manufacturing Engineering, Universiti Tun Hussein Onn Malaysia (UTHM), 86400 Parit Raja, Batu Pahat, Johor, Malaysia
[b] Faculty of Electric and Electronic Engineering, Universiti Tun Hussein Onn Malaysia (UTHM), 86400 Parit Raja, Batu Pahat, Johor, Malaysia

ARTICLE INFO

Keywords:
Moving target
Obstacle avoidance
Path planning
Q-learning, reinforcement learning
Mobile robot

ABSTRACT

Path planning is an essential element in mobile robot navigation. One of the popular path planners is Q-learning – a type of reinforcement learning that learns with little or no prior knowledge of the environment. Despite the successful implementation of Q-learning reported in numerous studies, its slow convergence associated with the curse of dimensionality may limit the performance in practice. To solve this problem, an Improved Q-learning (IQL) with three modifications is introduced in this study. First, a distance metric is added to Q-learning to guide the agent moves towards the target. Second, the Q function of Q-learning is modified to overcome dead-ends more effectively. Lastly, the virtual target concept is introduced in Q-learning to bypass dead-ends. Experimental results across twenty types of navigation maps show that the proposed strategies accelerate the learning speed of IQL in comparison with the Q-learning. Besides, performance comparison with seven well-known path planners indicates its efficiency in terms of the path smoothness, time taken, shortest distance and total distance used.

1. Introduction

Autonomous mobile robot plays an important role in the era of automation. Its application can be seen in multitudinous domains, in the household as cleaning robots (Muthugala et al., 2020), on-road as self-driving vehicles (Bresson, Alsayed, Yu, & Glaser, 2017), in the industry as automated guided vehicles (Lee, Wong, Ignatius, Rahman, & Tseng, 2020) and in the medical field as medical assistant robots (Su et al., 2021). Path planning is an essential and fundamental part of an autonomous mobile robot to efficiently move from one position to another. A well-planned path must satisfy two requirements: firstly, the feasibility of the generated path is guaranteed, meaning that the path must be collision-free; and secondly, the efficiency of autonomous navigation is optimized, indicating that the mobile robot can reach the desired position promptly with the shortest travelled distance without making impossible sharp turns (Chiang, Hsu, Fiser, Tapia, & Faust, 2019).

To date, various methods, for instance, cell decomposition (CD) (Brooks & Lozano-Perez, 1985), visibility graph (VG) (Lozano-Pérez & Wesley, 1979), rapidly-exploring random tree (RRT) (LaValle, 1998), probabilistic roadmap (PRM) (Kavraki, Svestka, Latombe, & Overmars, 1996), artificial potential field (Khatib, 1985), A* algorithm (Hart, Nilsson, & Raphael, 1968) and voronoi diagram (VD) (Aurenhammer, 1991) have been introduced to solve the path planning problem. These methods, however, experience difficulties when the environment information is not sufficient, particularly for the VG and VD (Zhang, Mao, Liu, & Liu, 2015). Furthermore, the path generated by the VG is near to the vertices of obstacles, which may limit its use in real-practice (Mac, Copot, Tran, & De Keyser, 2016). The path generated by VD, too, is not optimal in terms of path length (Mac et al., 2016). The artificial potential field is limited in the way that it may easily get trapped in the local minima, preventing the agent to reach its goal destination (Montiel, Orozco-Rosas, & Sepúlveda, 2015). Despite simplified roadmaps are produced by CD and VG, these methods are computationally complicated when dealing with the complex environment due to the graph search algorithms, such as A* and dijkstra algorithms, are integrated into their calculation to produce a complete and optimal path (Qureshi & Ayaz, 2016). The PRM and RRT, instead, randomly generate and allocate samples to form a non-collision path, yet some drawbacks arise, particular in passing through some hard to solve regions (R. Kang, Zhang, Tang, & Zhao, 2016). This is because insufficient samples are allocated in these regions. As a result, there is no guarantee that full

* Corresponding author.
E-mail addresses: ongp@uthm.edu.my (P. Ong), cylow@uthm.edu.my (C.Y. Low), roslio@uthm.edu.my (R. Omar).
[1] ORCID: 0000-0003-3136-7254.

https://doi.org/10.1016/j.eswa.2022.117191
Received 19 June 2020; Received in revised form 15 March 2022; Accepted 2 April 2022
Available online 7 April 2022

connectivity in the C-space can be successfully captured by the PRM (Kavraki et al., 1996).

In recent years, artificial intelligence (AI) has seen to emerge as an indispensable element in an autonomous mobile robot. The research of AI in path planning has escalated, in which the solutions using the AI methods to counter the limitations of classical path planning approaches have been developed (Mac et al., 2016). Among these, Q-learning (QL), proposed by Watkins (Watkins, 1989), has a wide application prospect to solve the complex optimization problems in path planning due to its advantages of being model-free and having the ability to learn and update the knowledge through continuous interaction with the environment (Klidbary, Shouraki, & Kourabbaslou, 2017). The key concept of QL is that the algorithm will be rewarded or punished for each action it takes at a state (Watkins, 1989). Hence, the QL seeks to maximize the rewards received from a series of actions. A Q-table, or simply a lookup table, is used to record the expected returns, known as Q-values, for each state-action pair. Through repetitive learning in an environment, the Q-values are continuously updated until an optimal action-state decision policy is achieved.

Despite QL has proven its effectiveness and broad utility to many domains, for instance, control process (Lei Yang, Nagy, Goffin, & Schlueter, 2015), energy management (Lingxiao Yang, Sun, Ma, & Wei, 2019), task scheduling (Wei et al., 2019) and forecasting (C. Feng, Sun, & Zhang, 2019), its slow learning process due to the curse of dimensionality is concerned (Klidbary et al., 2017). During the learning of QL, all action-state pairs are visited at least once. An increasing dimensionality of state spaces associated with a larger environment, or increasing action spaces, inevitably causes the memory size of the Q-table to increase exponentially (Vanhulsel, Janssens, Wets, & Vanhoof, 2009). This is particularly critical for real physical systems like path planning, where the environment is usually large in size.

Various enhancements have been made to solve the Achilles heel of QL, where hybridization of AI algorithms, in particular, the bio-inspired metaheuristic optimization algorithm and artificial neural networks (ANNs) with QL, was widely studied. Das *et al.* (Das, Behera, & Panigrahi, 2016) successfully integrated particle swarm optimization (PSO) with perturbed velocity (QIPSO) into QL for better global search ability and rate of convergence in multiple obstacles environment. In (Rakshit et al., 2013), the global search ability of QL was improved using the artificial bee colony optimization algorithm in a multi-robot environment. To balance the exploitation and exploration abilities of QL, the metropolis criterion of simulated annealing was introduced (Guo, Liu, & Malec, 2004).

In terms of the ANNs model, by utilizing the current state, time and obstacles as inputs, the pos-net neural network was used to decide the next action in QL (Duguleana & Mogan, 2016). This has greatly improved the performance of QL for dead-end avoidance and convergence in both static and dynamic environments. Jiang *et al.* (L. Jiang, Huang, & Ding, 2019) applied ANNs to replace Q-table in deep QL with an experience replay mechanism. This method resulted in faster convergence time, lesser steps taken, lesser training rounds and better adaptiveness in exploring the unknown environment. It has been reported by Song *et al.* (Song, Li, Li, & Zhang, 2012) that using the dynamic wave expansion neural network can enhance the convergence speed and stability of QL, on top of the learning complexity.

Besides optimization algorithms and ANNs, the integration of QL with other methods was also widely studied. In (Zhao, Ding, An, & Jia, 2018), the realization of parallel agents to explore the environment was conducted using the asynchronous method. This has greatly reduced the learning time of the algorithm as compared to the QL. The environment of QL was represented topologically using an instantaneous topological map in (Hafez & Loo, 2015). Through the topological representation, the Q-table was updated not only for the current state but the neighboring states as well, leading to a higher learning rate and better adaptation to environmental changes. The adaptive kernel model has been integrated into QL to store behavior policies of QL by Hu *et al.* (Hu, Li, He, & Han, 2019). This has improved the adaptability of QL in different environments through learning various behaviors.

Efforts on reducing or modifying state spaces of QL were also conducted. In (H. Chen, Ji, & Niu, 2020), an obstacle area expansion strategy was implemented in Q(λ)-learning algorithm to decrease learning state spaces and improve convergence. This strategy expanded the concave obstacle areas to lower the state space dimension. Hence, the learning efficiency has increased as the number of state space was reduced. Besides, the model shaping method was introduced into the QL to predict the unvisited state spaces by using the neighboring state space information (Hwang, Jiang, & Chen, 2016). This has greatly increased the learning efficiency as the agent did not need to visit every state space. Approximating unknown state spaces using the Kernel smoothing technique was also studied by Cruz and Yu (Cruz & Yu, 2017). To avoid the explosion of dimensionality, fuzzification of state spaces in QL has been conducted by Jiang and Xin (J. Jiang & Xin, 2019). It has been observed the convergence rate and collision rate were greatly improved by integrating the fuzzification.

The modification of the Q function in QL was also investigated in numerous studies. In (Das et al., 2016), based on the introduced lock status and the distance to the target position, different Q functions were applied during the path planning. As a result, the computational time and space complexity of the proposed QL were greatly reduced. In the proposed double QL algorithm (Carlucho, De Paula, & Acosta, 2019), two Q functions were modified, attempting to solve the over-estimation problem of QL. Wen *et al.* and Yan *et al.* decomposed the Q function into state value function and advantage function to stabilize the output (Wen et al., 2020; Yan, Xiang, & Wang, 2019).

The Euclidean distance is crucial information to improve the ability of QL. Numerous studies applied the Euclidean distance in the calculation of fitness function and reward function of QL. Exemplarily, the Euclidean distance between a mobile robot and the target position was used in the fitness function evaluation for a multi mobile robots environment (Das et al., 2016). The fitness function was then optimized using the Improved QL with Improved Particle Swarm Optimization and Differential Perturbed Velocity algorithm, leading to a significant reduction in terms of space complexity and computation time in comparison with the QL. The log function has been added to the Euclidean distance to form the reward value field function (Sah, Mohanty, Kumar, & Chhotray, 2019). In an unknown environment, the proposed method could avoid collision with the obstacles with an accuracy of 90%. By utilizing the Euclidean distance as the reward function of QL (C. Chen, Chen, Ma, Zeng, & Wang, 2019), the generated paths were shorter than that of A* and RRT in environments with a higher number of obstacles and with a lower number of sharp-turns. The Euclidean distance was also used in the calculation of the reward function of Deep QL in the case study of automation of multiple ships (Shen et al., 2019). A negative reward was given when the Euclidean distance between the obstacle and ship was lower than the threshold. The results showed that the proposed method performed satisfactorily in either open sea, congested or restricted environments even different ships models were considered.

Previous studies have shed light on how the integration of the Euclidean distance and modification of Q function could enhance the performance of QL (Carlucho et al., 2019; Das et al., 2016; Shen et al., 2019). These findings prompt the formulation of a modified QL in this study, involving the implementation of Euclidean distance through a distance metric and a modified Q function. Additionally, a moving target concept is introduced to resolve the dead-end scenario. The contributions of this study are:

- An improved Q-learning (IQL) is developed to improve the convergence of QL. The modification is based on the concepts of a distance metric, modified Q function and moving target.
- The performance comparison of the IQL with the baseline method, *i.e.*, QL has been investigated. The proposed IQL outperforms the baseline method in terms of the path smoothness, computation time,

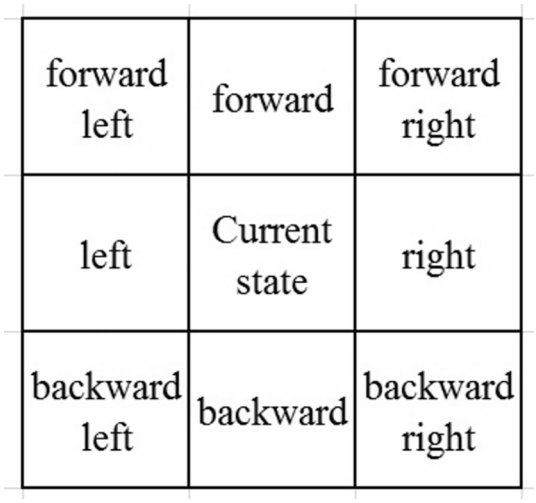


Fig. 1. The actions in the action set consist of forward, backward, left, right, forward left, forward right, backward left and backward right.

shortest distance travelled and total distance travelled, which are some of the major concerns in the path planning of a mobile robot.

This paper is structured as follows. Section 2 reviews the basic concept of QL, while Section 3 discusses the important features of IQL, specifically, the distance metric and virtual target concept, are presented. The simulation results of IQL and performance comparison with others in twenty navigation maps are discussed in Section 4. Finally, Section 5 summarizes the conclusions of this study.

2. Q-learning

QL, the widely used model-free reinforcement learning, was put forward by Watkins in 1989 (Watkins, 1989). The QL is defined as a Markov decision process (Bellman, 1957) in an environment that is fully defined by the variables (Sato, Abe, & Takeda, 1988), with:

S is the finite number of possible states.

A is the finite number of actions.

P is the probability matrix of the state transition. When action a is performed at state s, the probability of reaching state s' is $P(s,s',a)$.

R is the reward function.

γ is the discount factor, where $0 \leqslant \gamma \leqslant 1$.

$\pi : \rightarrow S \times A$ is the state transition function.

An agent of QL gains experiences through a set of episodes. For each episode, an agent will perform the following procedures in every iteration:

a. Determine the current state s
b. Choose and perform an action a
c. Determine and evaluate the new state s'
d. Obtain a reward R
e. Update the Q-value

The combination of Q-values of all states and selected actions will produce a Q-table. With sufficient episodes, the actions performed by an agent can maximize the total reward based on the Q-table.

Definition 1. ((*State set*)) *Since the environment is represented in the grid form, the states are referred to as the coordinates* (x,y) *of the grid. A detailed explanation of the environment and grid is discussed in* Section 3.1. *The state set S is defined as*:

$$S = \{s|(x,y)\ \} \tag{1}$$

Definition 2. ((*Action set*)) *The action set is the collection of all admissible actions of an agent to move from the current state to the next state. Exemplarily, the action set of* $A = \{$*forward, backward, left, right, forward left, forward right, backward left, backward right*$\}$ *as shown in* Fig. 1 *is used in this study. At the current state of* $s(x,y)$, *when an action a is taken, the new state will become* $s'(x',y')$.

Definition 3. ((*Reward function*)) *The reward function R encourages the mobile robot to move towards the target and avoid collision with the obstacles at the same time. Therefore, the reward function R is assigned as*:

$$R = \begin{cases} 1, & \text{when reached the target} \\ -1, & \text{when an obstacle is met} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Definition 4. ((*Q-value updating*)) *A Q-value* $Q(s,a)$ *will be assigned when the mobile robot is located at state s and an action a is taken. The Q-function or the Q-value is updated using*:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[R + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \tag{3}$$

where $Q(s,a)$ is the Q-value when the action a is executed at state a, $\max_{a'} Q(s',a')$ is the highest Q-value among all actions at next state s', R is the obtained reward, α is the learning rate and γ is the discount factor.

The selection of action is based on the Q-value. Among the Q-values of all actions at the current state, the state-action pairs with the highest Q-value will be selected and the selected action a is performed. The action selection function is formulated as:

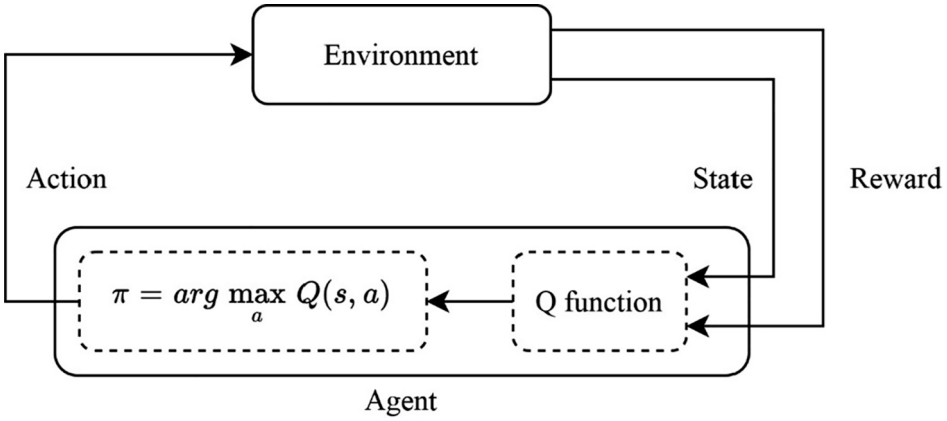


Fig. 2. The interaction between the agent and the environment.

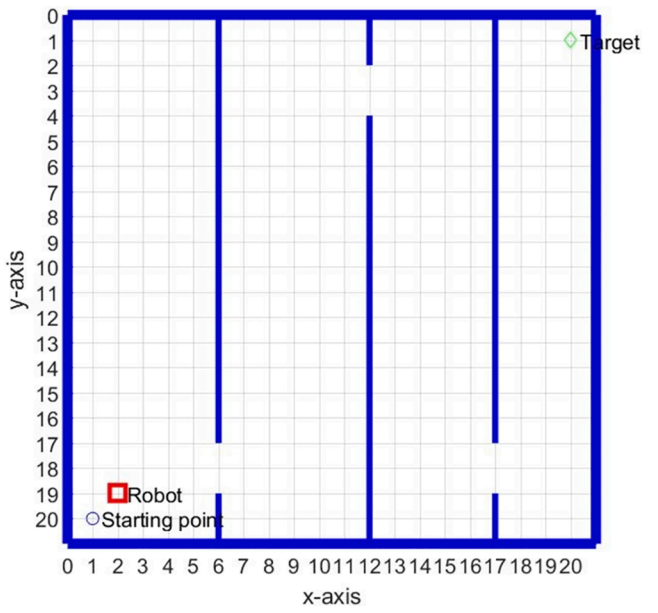


Fig. 3. The grid-based C-space representation of a map with mobile robot, target and starting positions indicated through points. The *x*-axis and *y*-axis are the horizontal and vertical axes on a Cartesian plane, respectively.

$$\pi = \underset{a}{\operatorname{argmax}} Q(s,a) \tag{4}$$

where $\max_a Q(s,a)$ is the highest Q-value among all actions at current state s.

Fig. 2 presents the relationship between an agent and the environment. In an environment, the agent will determine its current state. Through performing an action, the agent reaches a new state in the environment. Subsequently, the agent will gain a reward or penalty based on the status of the new state. The entire process is repeated by taking the new state as the current state.

During the early learning stage of QL, all Q-values are zeros except the target position, which is assigned to a large positive value. This has resulted in a random selection for the next state since all the action-state pairs have the same values of zeros. As a result, the agent has to endure the blind search phase at the beginning, up until some Q-values are refined. To address the slow convergence caused by no prior information of the environment, a distance metric is introduced in this work to serve as some guides for the agent during the exploration process. Although the distance metric improves the convergence of QL, it may lead the agent towards a dead-end. To overcome this issue, a virtual moving target concept is implemented to bypass the dead-ends. Additionally, scrutinizing the Q function in Equation (3) reveals that the resulted Q-value after an action is taken will remain zero if the following conditions are met:

a. The Q-value of the current state $Q(s,a)$ is zero
b. The Q-value of the next state $Q(s^{'},a^{'})$ is zero
c. The reward R is zero

During the first few episodes, most of the states satisfy the aforesaid conditions except the states around the target and the obstacles. As most of the Q-values remain zero, no experience will be gained by the agent. This further slows down the convergence of QL and increases time consumption to escape from the dead-end. Therefore, a modified Q function is introduced for some specific conditions.

3. Improved Q-learning

In this study, an IQL is proposed wherein, the modifications from the aspects of (i) distance metric, (ii) modified Q function and (iii) virtual target, are attached. Besides, two different path planning mechanisms of normal path planning and special path planning with escape mode are introduced to IQL. The path planning mechanisms are described in the following subsections.

second dimension																				
1	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	2
2	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
3	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
4	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
5	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
6	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
7	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
8	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
9	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
10	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
11	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
12	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
13	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
14	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
15	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
16	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
17	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
19	0	1	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
20	0	0	0	0	0	-1	0	0	0	0	0	-1	0	0	0	0	-1	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	first dimension																			

Fig. 4. The matrix MAP of Fig. 3 that contains the information of C-space with the value of −1 as obstacles, 1 as mobile robot, 2 as target. First dimension represents *x*-coordinate and second dimension represents *y*-coordinate.

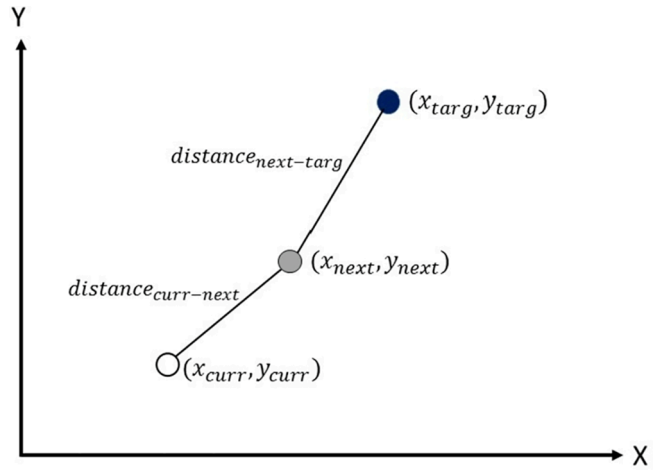


Fig. 5. The graphical illustration of the distance metric.

3.1. Configuration space

Configuration space, also known as C-space, is the map representation technique used to simplify the real-world environment. Multiple essential elements such as agent, obstacles and target are represented by points and colored areas in the C-space. The navigation maps in this case study are represented in grid-based C-space. For the environment with the dimension of $u \times w$ grids, each grid is considered as a state for QL and IQL. The grids are in the Cartesian plane with the x-axis and y-axis are the horizontal and vertical axes, respectively. Thus, a coordinate can be defined in the format of (x,y). Fig. 3 shows an example of the grid-based C-space representation. The information of C-space is stored in a matrix named MAP, with the same dimension as C-space grid size, where x-coordinate is the first dimension and y-coordinate is the second dimension. Each state or grid of MAP is assigned a value. If the grid is occupied by an obstacle in the C-space, the grid will take the value of -1. For space, agent and target, the values in MAP are assigned to 0, 1 and 2, respectively. In short, by referring to the MAP, an agent can define the occurrence of a collision or a safe path. The matrix MAP of the example in Fig. 3 is illustrated in Fig. 4.

3.2. Normal path planning mechanism

For IQL, the path planning is set by default to normal path planning mechanism, except for the specific situation that will be discussed hereinafter. Under the normal path planning mechanism, the selection of the action-state pair is determined by the distance metric, subjected to the following assumptions:

a. The global path planning is considered wherein, the locations of target and mobile robot are known.
b. The admissible actions are forward, backward, left, right, forward left, forward right, backward left and backward right.
c. The cell decomposition or grid-based is used to model the environment.
d. The agent, or the mobile robot, is allowed to move 1 unit per iteration or time frame.

Definition 5. ((*Distance metric*)) *The distance metric is implemented to assist the IQL in selecting the next state with the shortest distance to the target position. The quantitative value of the distance metric is evaluated based on the distance from the current position to the next state and the target position, which is defined as*:

$$\text{distance}_{total} = \text{distance}_{curr-next} + \text{distance}_{next-\text{targ}} \tag{5}$$

where distance_{total} is the total distance, $\text{distance}_{curr\text{-}next}$ is the distance from the current position to next position and $\text{distance}_{next\text{-}targ}$ is the distance from the next position to the target position. The $\text{distance}_{curr\text{-}next}$ and $\text{distance}_{next\text{-}targ}$ are calculated as:

$$\text{distance}_{curr-next} = \sqrt{(x_{next} - x_{curr})^2 + (y_{next} - y_{curr})^2} \tag{6}$$

$$\text{distance}_{next-targ} = \sqrt{(x_{targ} - x_{next})^2 + (y_{targ} - y_{next})^2} \tag{7}$$

where x represents the x-coordinate, y represents the y-coordinate, *next*

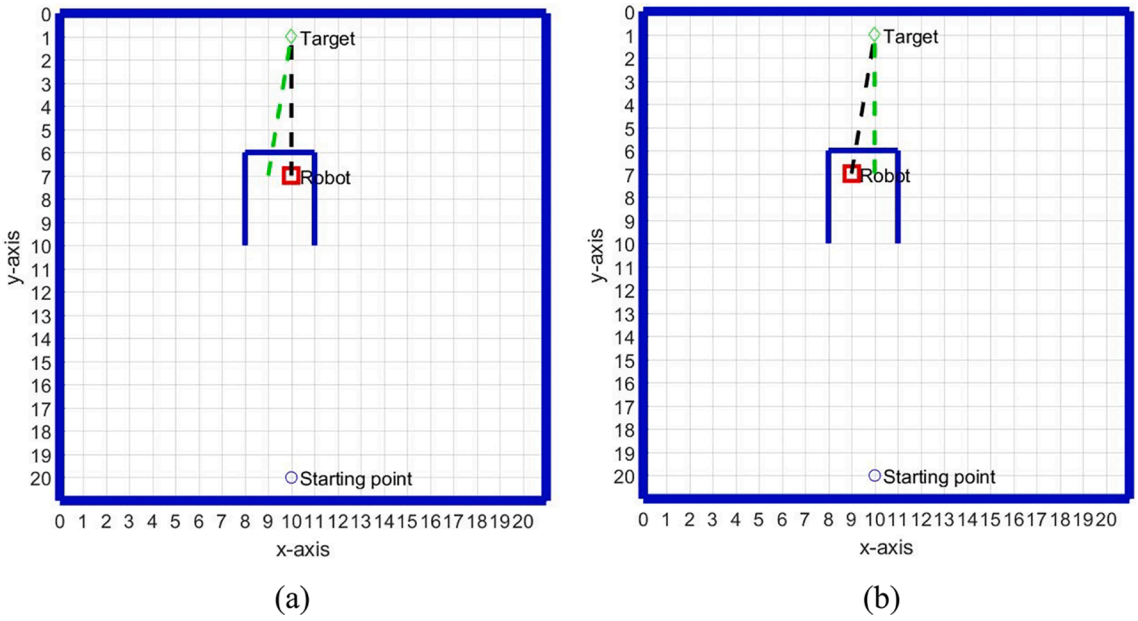


Fig. 6. An example to illustrate the possible limitation of using distance metric where the mobile robot might stuck in a dead-end: (a) The mobile robot will move to the left since this is the action-state pair with the shortest total distance; (b) The mobile robot will move to the right since this is the action-state pair with the shortest total distance. Hence, it returns to the original state as in (a).

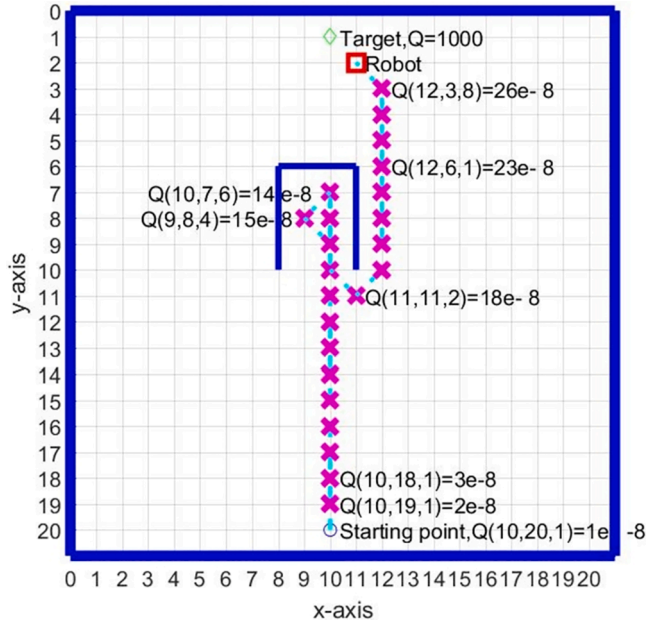


Fig. 7. An example to illustrate the gradual increment of the Q-value. At the starting point, the Q-value is 1e-8 and gradually increases from 1e-8 to 13e-8 when the mobile robot moves forward in the subsequent iterations (cyan color dashed line and magenta color cross shape). The Q-values located around the target such as Q(12,3,8) are relatively higher than the Q-values located around the starting point such as Q(10,19,1).

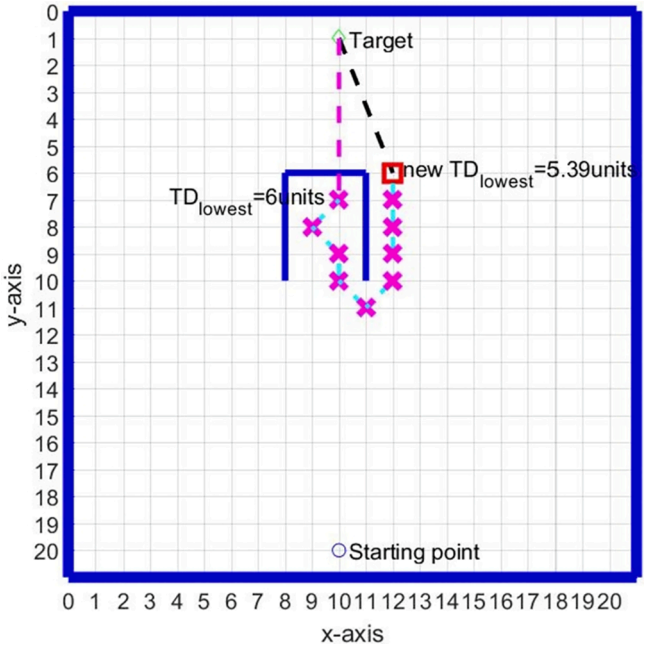


Fig. 8. As the mobile robot escaped from the dead-end, a state with the total distance (5.39 units, indicated by black color dashed line) lower than the TD_{lowest} (6 units, indicated by magenta color dashed line) is found at (12,6). This leads to the termination of escape mode.

represents the next position, *curr* represents the current position and *targ* represents the target position. The graphical representation of the distance metric is shown in Fig. 5.

During the process of action selection, the total distances of all available action-state pairs are calculated, as in Equation (5). The action-state pair with the lowest total distance is chosen. Through the selection of the lowest total distance, the limitation of random selection during the beginning stage of QL is addressed. With the guidance of the total distance mechanism, an appropriate selection of the next state can be performed, leading to a shorter completion time. However, by using this selection mechanism, the mobile robot might get trapped in a dead-end.

To further illustrate, an example of a dead-end is shown in Fig. 6(a) and Fig. 6(b) to demonstrate the potential weakness of the distance metric. In Fig. 6(a), the mobile robot gets trapped in a dead-end. The next possible state with the lowest total distance is indicated by the green line. Therefore, the mobile robot will move to the left at the next state, and the new state is as illustrated in Fig. 6(b). However, scrutinizing Fig. 6(b) shows that the next state with the lowest total distance (indicated by the green line) is the same as the previous state in Fig. 6(a). By moving to the right, hence, the mobile robot will end up in an infinite loop. This prevents the mobile robot from reaching the target. Being the case, an escape mode is introduced to enable the IQL to escape from the entrapment of dead-end.

3.3. Special path planning mechanism with escape mode

As observed in Fig. 6(a) and Fig. 6(b), when the mobile robot meets a dead-end, it is impossible to obtain an action-state pair with a lower total distance in the subsequent iteration. Here, a variable to continuously store the obtained lowest total distance, TD_{lowest}, is introduced. When the value of the TD_{lowest} remains constant for four continuous iterations, the mobile robot is considered stuck in a dead-end. Hence, the special path planning with escape mode takes over the normal path planning mechanism wherein, the selection of the action-pair based on the distance metric is disabled.

During the execution of escape mode, the Q function in Equation (3) is utilized to select the next available action with the highest Q-value. This has overcome the dead-end issue, but the high computational cost remains a concern. As mentioned in Section 2, in the early stage of exploration, most of the Q-values are zeros. This results in a random selection of action since no prior knowledge about the environment is available. Such random motion produces inconsistent performance and a low convergence rate due to the blind search phase at the beginning. The situation is worsening when the leading effect from distance metric in the normal path planning mechanism directs the mobile robot to the same dead-end in each episode. Hence, a modified Q function is proposed in this work.

Definition 6. (*(Modified Q function)*) *The purposes of implementing the modified Q function are two-fold*: (*i*) *to accelerate the escape from the dead-end, and* (*ii*) *to avoid the random motion during the escape mode while ensuring the least adverse impact to the final optimal path. It should be noted that the modified Q function is only applied to the first episode of IQL. In the first episode, in every state that the mobile robot is in, the Q-value at i-th iteration is updated with the modified Q function, given by*:

$$Q(s_i, a_i) = i \times 1e-8 \tag{8}$$

The Q-value is assigned to a value of 1e-8 at the first iteration and the value will increase by 1e-8 in every subsequent iteration.

The graphical illustration of the implementation of the modified Q function is shown in Fig. 7. The summation of negligible small value, *i.e.* 1e-8 in this case, to the Q-value will prevent the random motion of the mobile robot during the execution of the escape mode. The gradual increment of the Q-value in each iteration guides the mobile robot to get out from the dead-end. Upon completion of the first episode, the Q-values of the selected action-states distributed near the target position are higher than those scattered around the starting position, as shown in Fig. 7.

Once the mobile robot has escaped from the dead-end, inevitably, there is a modification to the value of TD_{lowest}. Consequently, the escape mode will be disabled and the normal path planning mechanism is switched back into operation. Fig. 8 demonstrates the situation when the

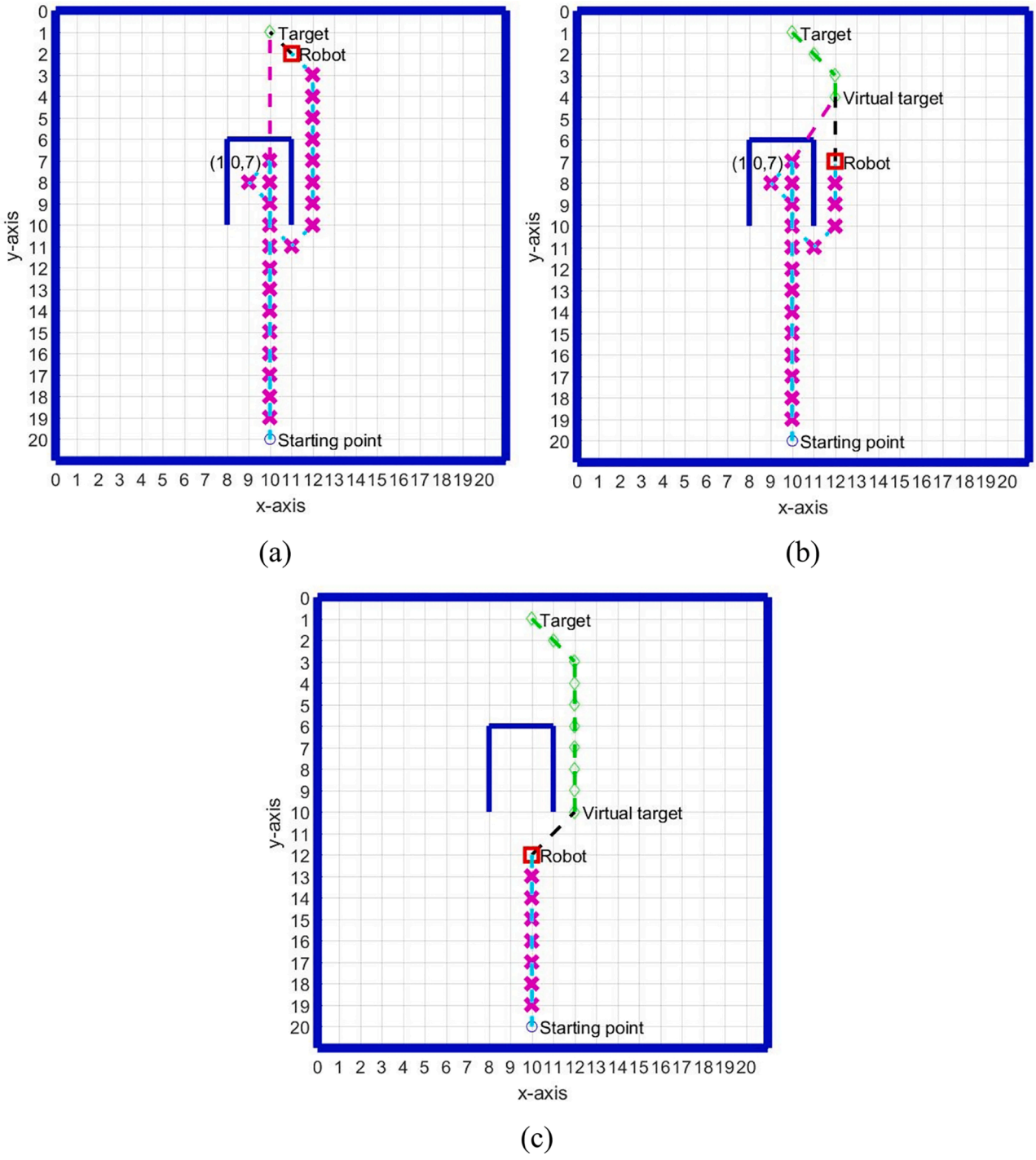


Fig. 9. An illustration of how the mobile robot could bypass the dead-end using the virtual moving target concept in: (a) episode 1; (b) episode 4; and (c) episode 10.

escape mode is disabled.

3.4. Virtual target concept

Despite the successful escape mechanism is witnessed in Section 3.3, the generated path tends to follow along the escape path without bypassing the dead-end. Here, the virtual target concept is presented, attempting to smoothen the final path generated around the dead-ends.

Definition 7. ((*Virtual target*)) *The virtual target mimics the real target in the simulation. Initially, the virtual target is located at the target position. Once the mobile robot reaches the target position, an episode is completed. For the next episode, the virtual target will move to the previous position that the mobile robot is in, just before it reached the target position. Now, the target position will be replaced by the position of the virtual target. In other words, the implementation of either normal path planning or special path planning with the escape mode will be now based on the new position of the virtual target. The shift of the virtual target to the new position is repeated, where the position of the virtual target can be defined as*:

$$x_{n,epi+1} = x_{n-1,epi} \tag{9}$$

$$y_{n,epi+1} = y_{n-1,epi} \tag{10}$$

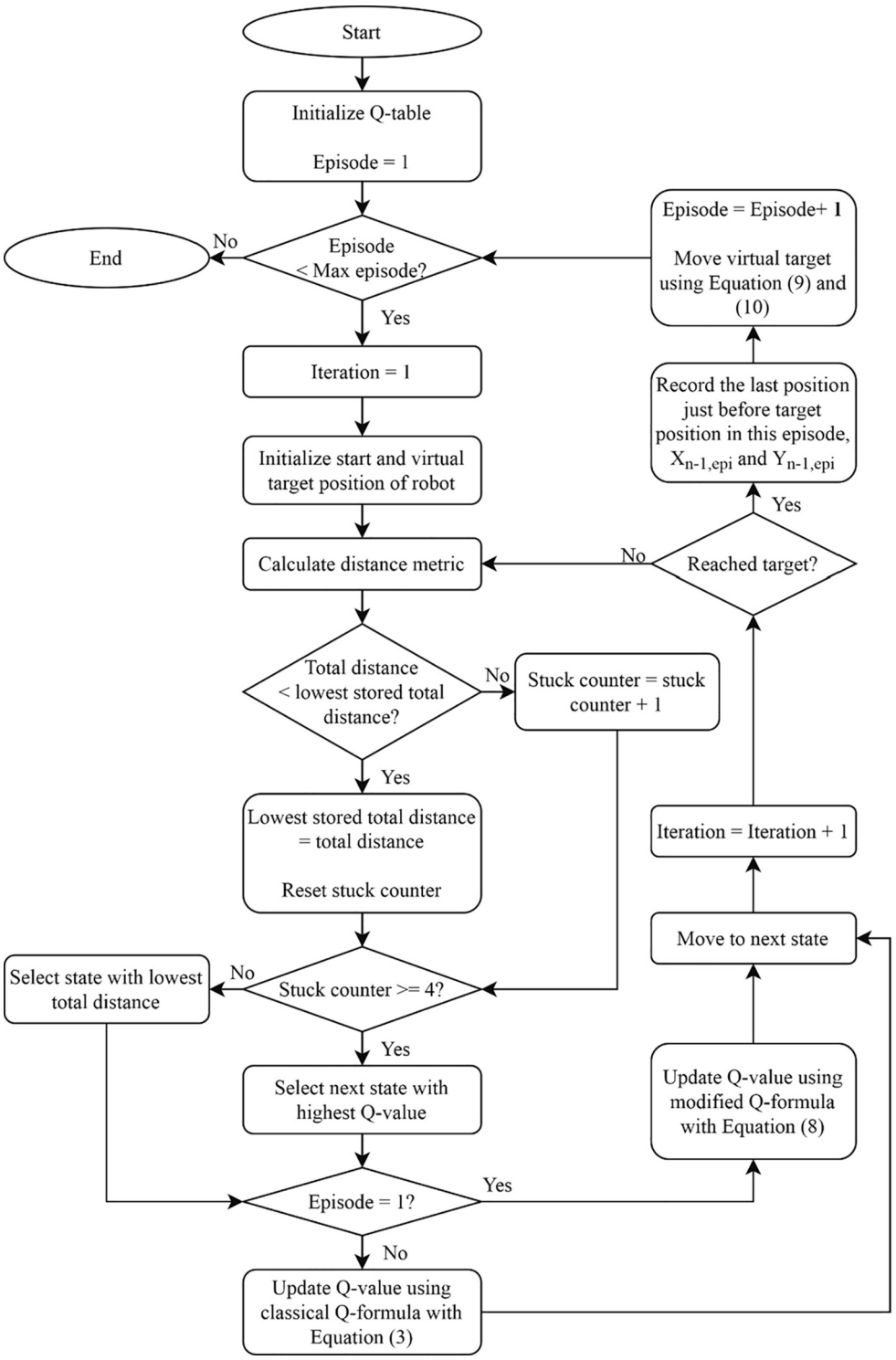


Fig. 10. The flowchart of IQL.

with x is the x-position, y is the y-position and n is last position in episode *epi*.

As the episode goes on, the virtual target is getting closer and closer to the starting point. Hence, the mobile robot could bypass the dead-end since the target position (*i.e.* the virtual target position) is no longer behind the dead-end. Fig. 9(a) illustrates the generated path in the first episode by applying the mechanism of distance metric. As presented in this figure, the mobile robot encounters a dead-end, activates the special path planning mechanism with escape mode at (10,7) and moves towards the target position after escaping from the dead-end state. In episode 4 shown in Fig. 9(b), the virtual target is now 4 steps closer to the starting point. However, the mobile robot still encounters the dead-end. As the episodes continue, the mobile robot will bypass the dead-end and alter the produced final path in episode 10, as shown in Fig. 9(c).

Fig. 10 presents the flowchart of the proposed IQL.

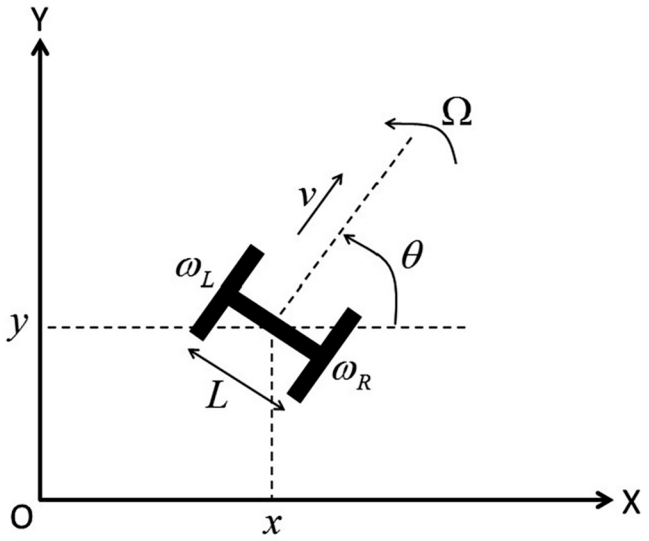


Fig. 11. Schematic diagram of a DDWMR on Cartesian plane (Y. Feng et al., 2015).

3.5. The feasibility of applying the IQL for differential drive wheeled mobile robot

To further illustrate the feasibility of applying the IQL algorithm in mobile robot path planning, the differential drive wheeled mobile robot (DDWMR) is used as an example. The schematic diagram of a DDWMR in the Cartesian plane is shown in Fig. 11 (Y. Feng et al., 2015). Here, x and y represent the x-coordinate and y-coordinate of the robot, ω_L and ω_R are the angular velocities of the left wheel and right wheel, L is the distance between the left and right wheels, v is the translational velocity of the robot, θ is the orientation of the robot referring to the x-axis and Ω is the angular velocity of the robot. The pose of DDWMR is described as:

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{11}$$

The kinematics model of the DDWMR is formulated as (Martins, Adekunle, Adejuyigbe, Adeyemi, & Arowolo, 2020):

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \Omega \end{bmatrix} \tag{12}$$

The velocities of the left and right wheels are calculated as:

$$\begin{aligned} v_L &= r\omega_L \\ v_R &= r\omega_R \end{aligned} \tag{13}$$

where r is the wheel radius.

The translational velocity of the mobile robot can be calculated using:

$$v = (v_L + v_R)/2 \tag{14}$$

By substituting Equation (13) into Equation (14), the following equation is formed:

$$\begin{aligned} v &= (r\omega_L + r\omega_R)/2 \\ v &= r(\omega_L + \omega_R)/2 \end{aligned} \tag{15}$$

The angular velocity of the mobile robot can be obtained using:

$$\Omega = (v_R - v_L)/L \tag{16}$$

By substituting Equation (13) into Equation (16), the following equation is formed:

$$\begin{aligned} \Omega &= (r\omega_R - r\omega_L)/L \\ \Omega &= r(\omega_R - \omega_L)/L \end{aligned} \tag{17}$$

The kinematic motion of the DDWMR in discrete time is formulated as (Aouf, Boussaid, & Sakly, 2019):

$$\begin{aligned} x_{i+1} &= x_i + T\dot{x}_i \\ &= x_i + Tv_i\cos\theta_i \\ &= x_i + Tr\left(\omega_{L,i} + \omega_{R,i}\right)/2{*}\cos\theta_i \end{aligned} \tag{18}$$

$$\begin{aligned} y_{i+1} &= y_i + T\dot{y}_i \\ &= y_i + Tv_i\sin\theta_i \\ &= y_i + Tr\left(\omega_{L,i} + \omega_{R,i}\right)/2{*}\sin\theta_i \end{aligned} \tag{19}$$

$$\begin{aligned} \theta_{i+1} &= \theta_i + T\dot{\theta}_i \\ &= \theta_i + T\Omega_i \\ &= \theta_i + Tr\left(\omega_{R,i} - \omega_{L,i}\right)/L \end{aligned} \tag{20}$$

where T is the sampling time.

The state set S for the DDWMR is:

$$S = \{s|(x, y, \theta)\ \} \tag{21}$$

The values of x, y and θ can be calculated using Equation (18), Equation (19) and Equation (20), by using the values of ω_L and ω_R from the action set.

The action set A is defined as:

$$A = \{a|(\omega_L, \omega_R)\ \} \tag{22}$$

The other features of IQL, including the Q-value updating function, distance metric, modified Q function will change accordingly to the state set S and action set A. While for other features such as the reward function and the virtual target concept, their working mechanisms are not affected by the kinematic model of the mobile robot. The reward function and Q-value updating function are calculated as in Equation (2) and Equation (3). For the distance metric, the total distances are calculated using Equation (5). The action which leads to the lowest total distance in the next state is chosen. The modified Q function is updated using Equation (8) in the first episode. The shift of the virtual target follows Equation (9) and Equation (10).

Table 1
Parameter settings for all navigation maps and path planning algorithms.

Parameter	Path Planning Algorithm	Value
Number of runs	A*	1
	Other algorithms	30
Maximum episode	QL and IQL	300
	Other algorithms	1
Map size	Map 8	31x31 unit
	Map 18	34x34 unit
	Other maps	20x20 unit
Obstacle type	All maps	Static
Starting point	Map 8	(1,30)
	Map 9	(2,20)
	Map 16	(2,17)
	Map 18	(1,34)
	Other maps	(1,20)
Target point	Map 8	(31,1)
	Map 9	(20,5)
	Map 16	(20,2)
	Map 18	(34,1)
	Other maps	(20,1)
QL and IQL	α	0.1
	γ	0.7
PRM	Number of nodes	1000
	Number of neighboring nodes	10
CD	Resolution	(Map size + 1)/30

4. Results and discussion

In this section, the proposed IQL is compared with seven path

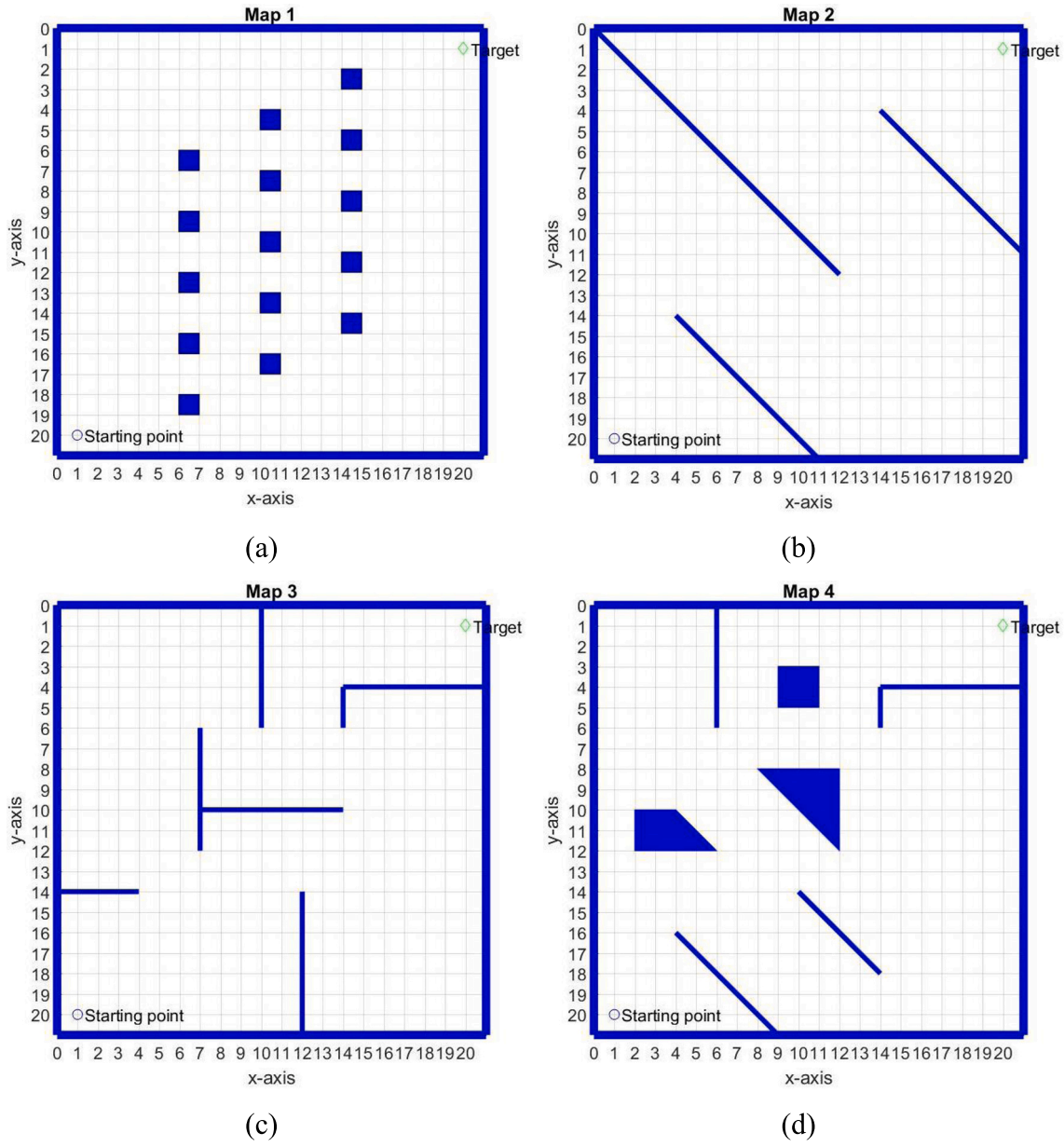


Fig. 12. The map structure of 20 navigation maps: (a) Map 1; (b) Map 2; (c) Map 3; (d) Map 4; (e) Map 5; (f) Map 6; (g) Map 7; (h) Map 8; (i) Map 9; (j) Map 10; (k) Map 11; (l) Map 12; (m) Map 13; (n) Map 14; (o) Map 15; (p) Map 16; (q) Map 17; (r) Map 18; (s) Map 19; and (t) Map 20.

planning algorithms, namely, the baseline method - QL, A* search algorithm, RRT, VG, PRM, VD and CD. Twenty environments are selected for the path planning simulations and performed in an Intel® Core™ i7-5500U CPU @ 2.40 GHz, 8 GB RAM with Windows 8.1 Pro 64-bit operating system. Table 1 summarizes the parameter settings for all the navigation maps and path planning algorithms. It is pertinent to note that for all simulations, the VG, VD and CD will be integrated with the Dijkstra algorithm. For the A*, a grid-based motion is applied to search for the optimal path. Since A* is a deterministic algorithm, its simulation is conducted once while the simulations of other algorithms are repeated 30 times for each case study. The parameters of PRM are assigned as in the reported work in (Contreras-Cruz, Ayala-Ramirez, & Hernandez-Belmonte, 2015). The working mechanism of each algorithm is as follows:

- The A* search algorithm determines the optimal path from the initial position to the target position by choosing the path with the minimum cost (N. K. Kang, Son, & Lee, 2018). An open list and a closed list are used in the A*. The node experienced by the A* and its adjacent nodes will be added to the open list and the node with the minimum cost will be selected through sorting. The selected node is then removed from the open list and added to the closed list.
- The RRT searches the optimal path by randomly constructing a space-filling tree (Wang, Deng, & Wu, 2018). The tree incrementally expands by placing some random samples in the environment.
- The VG forms the nodes on the vertices of obstacles (Donatelli, Giannelli, Mugnaini, & Sestini, 2017). An edge is then formed between the nodes if the line segment connecting them does not collide with the obstacles.
- Initially, the PRM randomly creates a certain number of nodes in the environment (Tsardoulias, Iliakopoulou, Kargakos, & Petrou, 2016).

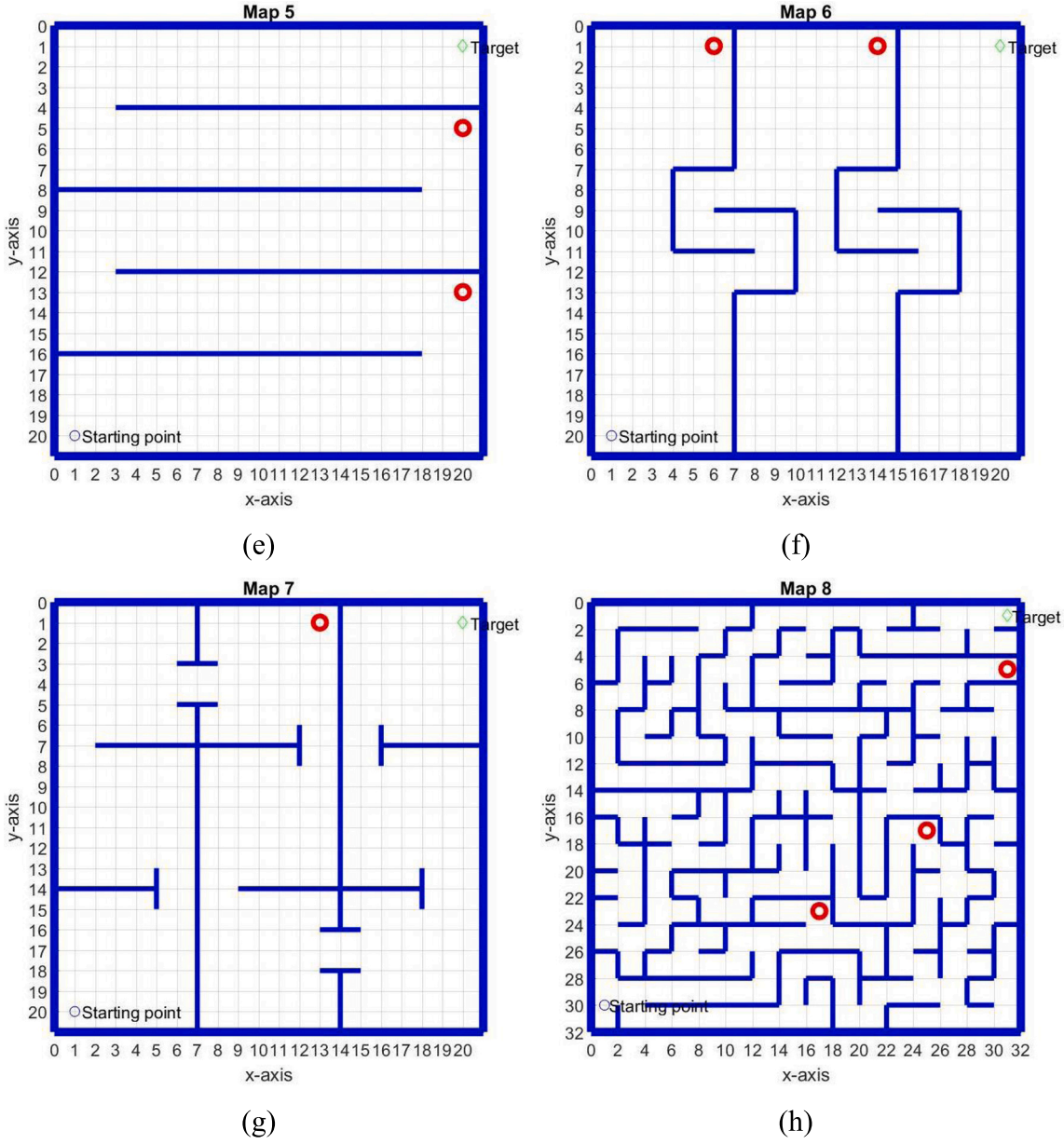


Fig. 12. (*continued*).

Then, the nodes are connected to some neighboring nodes, in which the connecting line segments shall be collision-free. The configuration and connection are repeated until a dense roadmap is obtained.

- The VD divides a configuration space into several regions based on the distribution of the obstacles (Candeloro, Lekkas, & Sørensen, 2017). The nodes are generated using the free space between the vertices of the obstacles. Subsequently, the collision-free edges are formed by connecting these nodes.
- The CD decomposes a configuration space into polygonal areas using the same shape (Mac, Copot, Tran, & De Keyser, 2017). The commonly used shapes are trapezoidal, triangular and rectangular. The vertices or midpoints of these polygons will form the nodes and edges to construct the path.

For the path planning problem, several evaluation functions can be used to indicate a good path, such as the shortest travelled distance, total travelled distance, computational time, path completeness (Omar, 2012) and path smoothness. In this study, all path planning algorithms are compared in terms of the path smoothness, computational time, the shortest distance among 30 runs and the total distance. The path smoothness is evaluated using (Mac et al., 2017):

$$path\_smoothness = 1/\sum_{i=1}^{n} |a\tan2[(y_{i+1}-y_i)/(x_{i+1}-x_i)] - a\tan2[(y_i-y_{i-1})/(x_i-x_{i-1})]| \tag{23}$$

where n is the total number of turnings, x_i and y_i are the x-coordinate and y-coordinate at the turning point i, respectively.

The distance is calculated using:

$$distance = \sum_{i}^{n-1} \sqrt{(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2} \tag{24}$$

where (x_i, y_i) is the current position of the mobile robot, (x_{i+1}, y_{i+1}) is the next position of the mobile robot and n is the total number of positions.

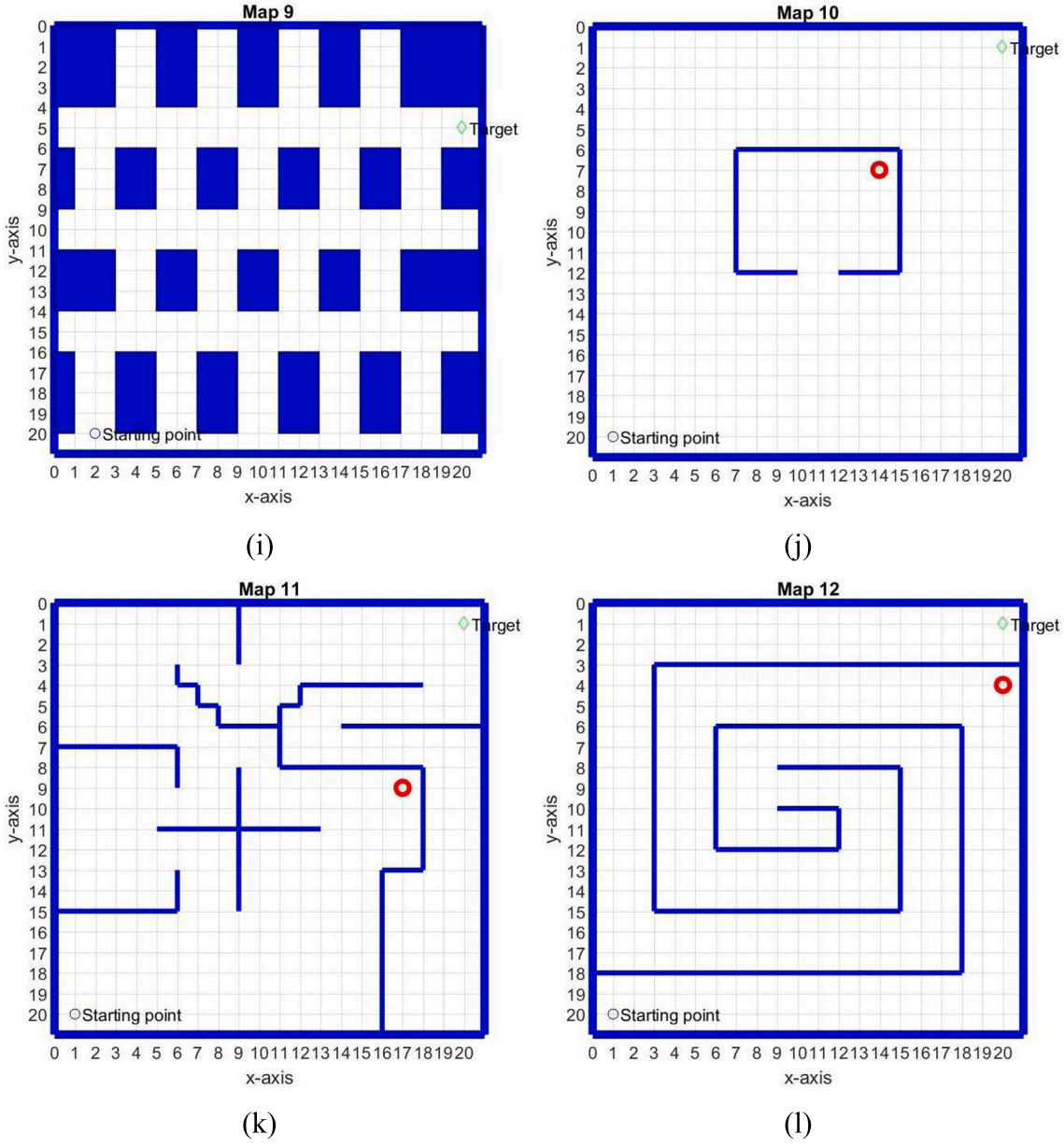


Fig. 12. (*continued*).

4.1. Navigation maps

Twenty maps with different layouts are considered in this study, as presented in Fig. 12. In these maps, the red circles are used to indicate the location of deep dead-ends. A dead-end is considered as a deep dead-end if the mobile robot needs to travel a long distance to escape from the dead-end. The explanation of each map is as follows:

- Map 1 in Fig. 12(a) consists of 15 obstacles of equal size (Yijing, Zheng, Xiaoyi, & Yang, 2017). The arrangement of obstacles creates tiny passages between them. Moreover, the obstacles are not aligned horizontally to increase the difficulty of navigation.
- Map 2 in Fig. 12(b) is made up of three collateral walls arranged in opposite direction (Yijing et al., 2017). This arrangement of walls requires the mobile robot to travel in S-shaped curves or zigzag path and creates dead-ends near the vertices of each wall.
- Map 3 in Fig. 12(c) consists of several walls. These walls divide the available space into multiple parts while also have openings to allow the mobile robot to travel from part to part (Yijing et al., 2017). The walls are positioned in such a way that the mobile robot can reach the target by taking the path above the 'T' shaped wall or the path below the 'T' shaped wall.
- Map 4 in Fig. 12(d) consists of a combination of walls, triangle and square shaped obstacles (Yijing et al., 2017). The layout of Map 4 is similar to Map 3 with some collateral walls exist. The difference is that the 'T' shaped wall has been replaced by the obstacles in the triangle and square shapes.
- Map 5 in Fig. 12(e) consists of four horizontal walls, with two walls are attached to the left side and right side, respectively (Contreras-Cruz et al., 2015). Despite the similar setting as in Map 2, the walls of Map 5 are positioned horizontally instead of diagonally.
- Map 6 in Fig. 12(f) consists of some connected walls, forming two mini mazes in the middle (Contreras-Cruz et al., 2015). To reach the targeted position, the mobile robot has to pass through the mazes.
- Map 7 in Fig. 12(g) divides the navigation environment into nine equally space using multiple walls (Contreras-Cruz et al., 2015). The

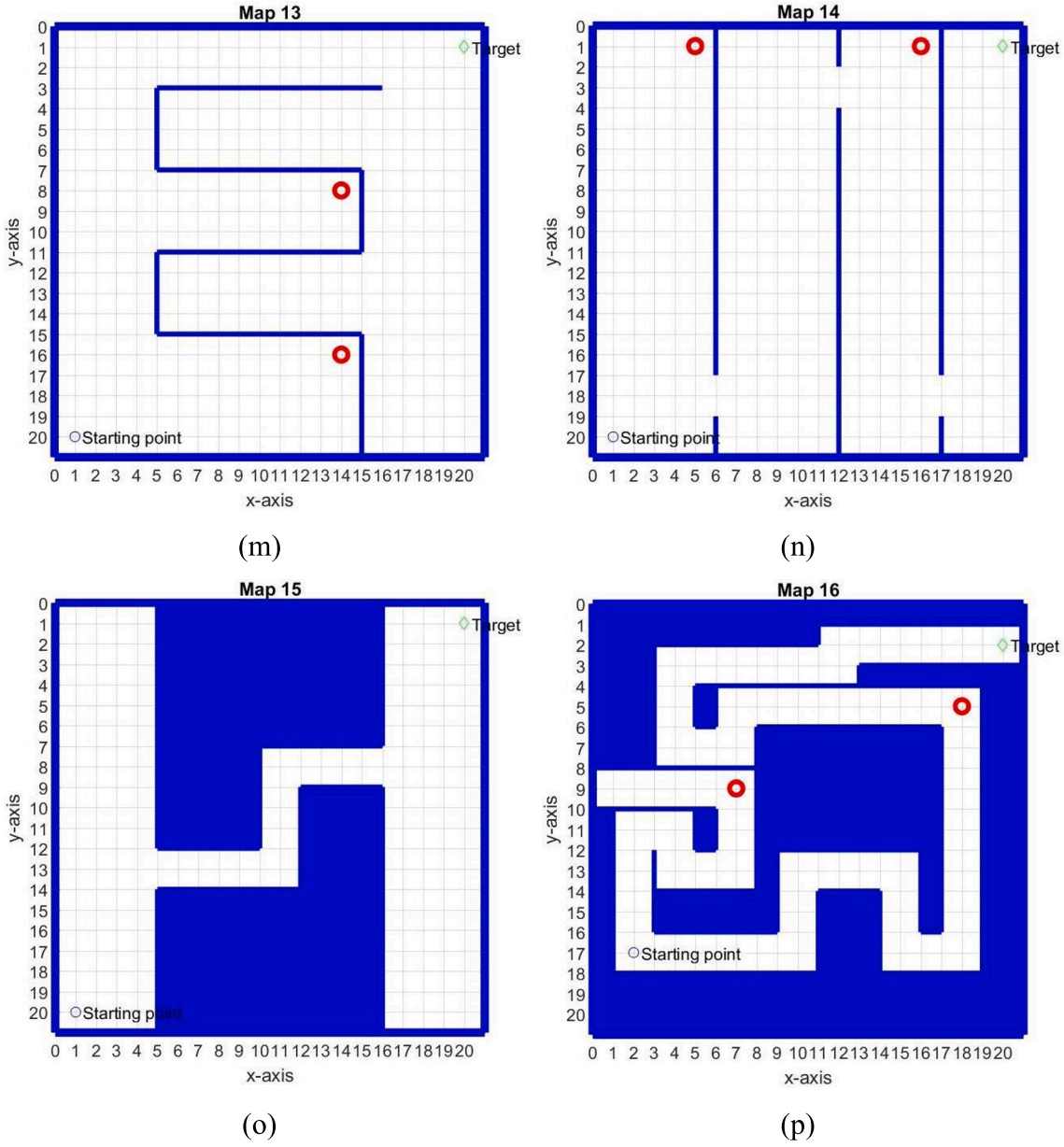


Fig. 12. (*continued*).

small openings at the wall allow the mobile robot to move from one part to another. Some walls are added to the sides at some openings to increase the difficulty of navigation.

- Map 8 in Fig. 12(h) is the most complicated map among all (Sturtevant, 2012), consisting of a huge maze map with multiple dead-ends and different possible selection of directions at some junctions.
- Map 9 in Fig. 12(i) is made up of blocks of rectangular obstacles, aligned on four rows (Contreras-Cruz et al., 2015). Several dead-ends are found at the first row of obstacles, increasing the difficulty of navigation.
- Map 10 in Fig. 12(j) consists of a dead-end in the middle (Contreras-Cruz et al., 2015). This is to test the escape ability of the path planning algorithms. A small opening is located at the bottom to allow the mobile robot from escaping from the dead-end.
- Map 11 in Fig. 12(k) has multiple walls connecting to each other (Contreras-Cruz et al., 2015). From the starting position, the mobile robot can choose either to use the path on the left or right side of the '+' shaped wall located in the middle of the map. Two dead-ends exist at the left and right sides of this map.
- Map 12 in Fig. 12(l) is made up of connected walls which form a spiral maze (Contreras-Cruz et al., 2015). The mobile robot has to travel in a counter-clockwise direction to reach the middle of the map, and subsequently in a clockwise direction to reach the target position.
- Map 13 in Fig. 12(m) has multiple 'S' shaped walls in the middle of the map (Contreras-Cruz et al., 2015). The indentations located at the left and right sides are the dead-ends that the mobile robot has to deal with.
- Map 14 in Fig. 12(n) consists of three vertical walls parallel to each other, with a small opening to allow the mobile robot to pass through (Contreras-Cruz et al., 2015). Map 14 is similar to Map 5, except that the walls of Map 14 are arranged in the vertical direction.
- Map 15 in Fig. 12(o) has two large blocks of obstacles located at the top and the bottom of the map (Contreras-Cruz et al., 2015). The

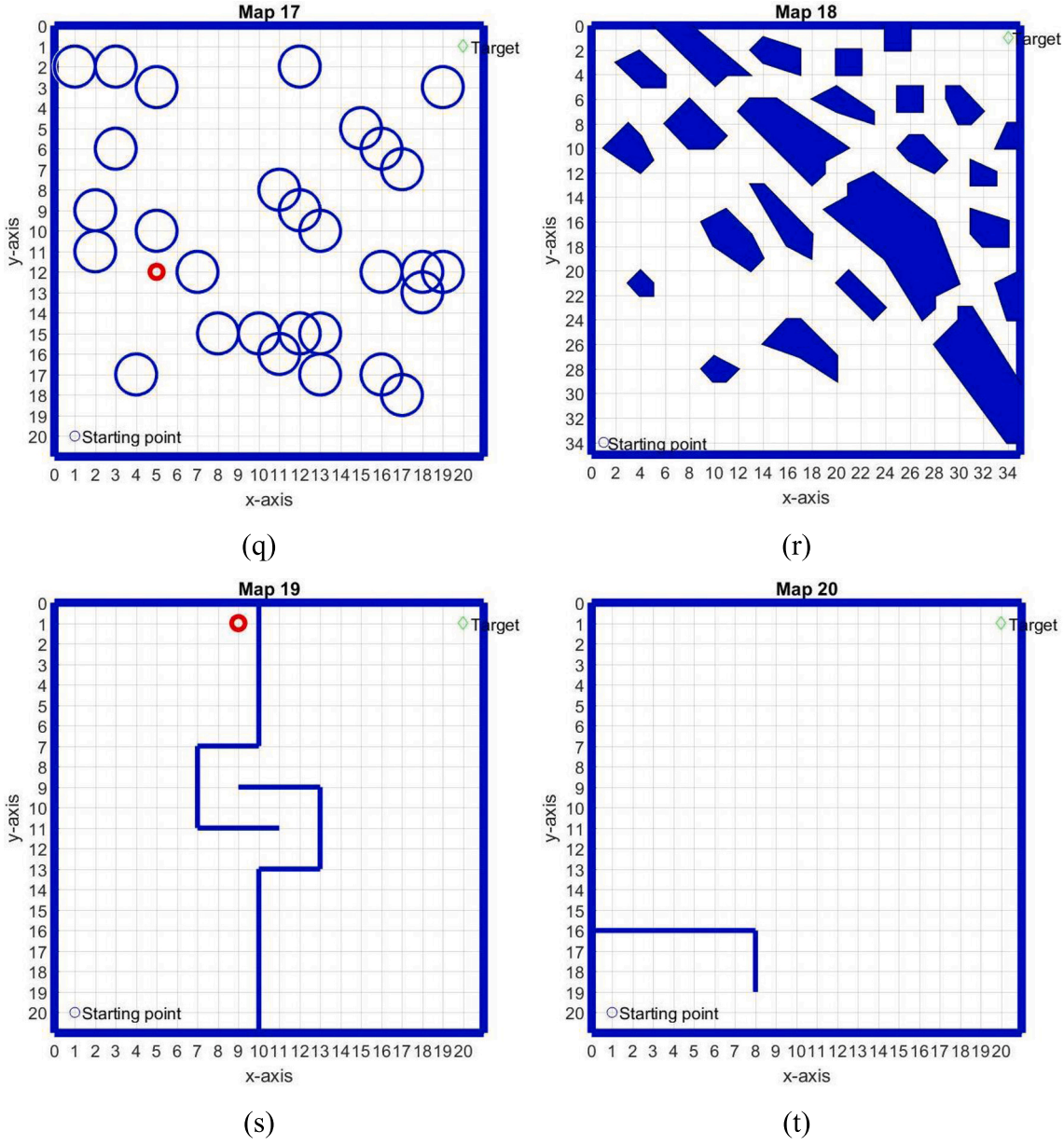


Fig. 12. (*continued*).

layout of the obstacles forms a narrow staircase-like path for the mobile robot to pass through.

- Map 16 in Fig. 12(p) is a maze map, which is relatively simpler than Map 8 (Contreras-Cruz et al., 2015). A dead-end is located near to the top of the starting point. There is only one feasible path toward the target position.
- Map 17 in Fig. 12(q) is a self-designed map, with some circular obstacles are randomly generated. Some obstacles are stacked together, with only a tiny gap is available for the mobile robot to move through.
- Map 18 in Fig. 12(r) consists of multiple irregular shards (Contreras-Cruz et al., 2015). Smaller shards are at the top right and bottom left corners of the map, with a higher density of shards at the top right corner. Larger shards are in between these two corners.
- Map 19 in Fig. 12(s) is a simplified version of Map 6, with only one mini maze in the middle of the map (Contreras-Cruz et al., 2015).
- Map 20 in Fig. 12(t) is a simple map with a hook-shaped dead-end near the starting point (Contreras-Cruz et al., 2015). There is a small opening at the bottom right side of the dead-end to allow the mobile robot to escape from the dead-end. Moreover, the existence of numerous spaces may increase computation time.

4.2. Performance comparison and discussion

For all the considered navigation maps, all algorithms are repeated for 30 runs, except for A*. The performance of each algorithm is evaluated based on the path smoothness, time taken to complete the path, the shortest travelled distance and the total travelled distance. The performance comparisons between the proposed IQL, QL, A*, RRT, VG, PRM, VD and CD are made.

4.2.1. Path smoothness

Fig. 13 presents the shortest paths generated by all algorithms for Map 2. The obtained shortest paths of other maps are omitted here for brevity and can be referred to Appendix A. Table 2 presents the path smoothness of the shortest planned paths by all algorithms among 30 runs. As shown in this table, the IQL outperformed or comparable with the baseline method QL in all the maps, with improvements of 22.22% to

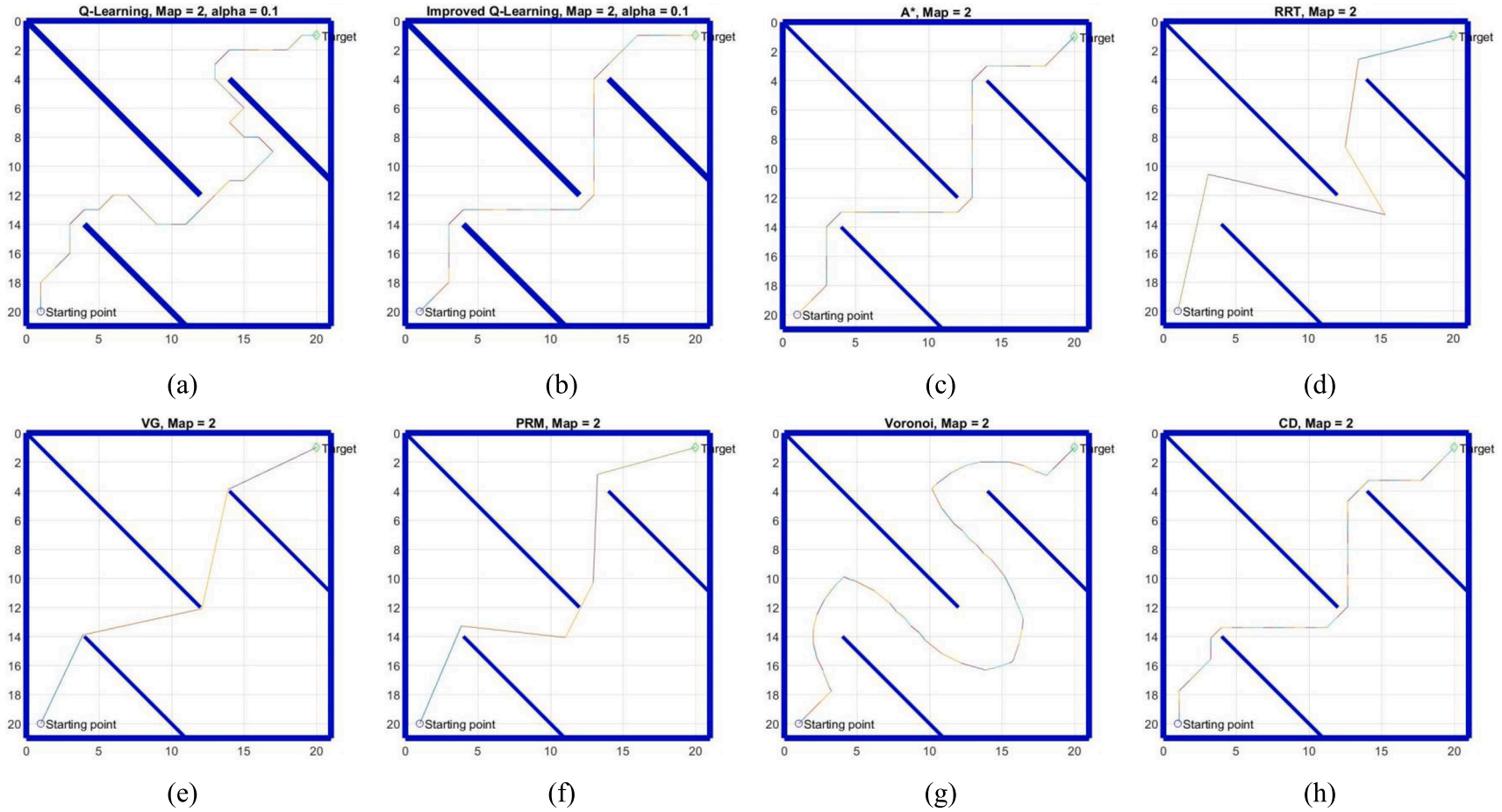


Fig. 13. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 2.

Table 2
Comparison of path smoothness for all algorithms with IQL for all maps.

Map	Smoothness (rad^{-1})															
	QL		IQL		A*		RRT		VG		PRM		VD		CD	
	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)
1	0.0707	63.64	0.1157	0.00	0.1157	0.00	0.7625	−84.82	**2.5529**	−95.47	1.4543	−92.04	0.0596	94.31	0.1819	−36.36
2	0.0354	414.29	0.1819	0.00	0.1592	14.29	0.0950	91.42	**0.3378**	−46.15	0.2391	−23.91	0.0664	173.77	0.1415	28.57
3	0.0509	177.78	0.1415	0.00	0.1819	−22.22	0.1998	−29.20	**0.4650**	−69.58	0.3859	−63.34	0.0465	204.34	0.1415	0.00
4	0.0531	242.86	0.1819	0.00	0.1592	14.29	0.2115	−14.00	**1.1230**	−83.80	0.5602	−67.53	0.0766	137.48	0.1415	28.57
5	0.0070	302.22	0.0283	0.00	0.0749	−62.22	0.0167	69.54	**0.0934**	−69.71	0.0286	−1.17	0.0268	5.57	0.0283	0.00
6	0.0174	37.74	0.0240	0.00	0.0344	−30.19	0.0163	47.22	**0.1022**	−76.50	0.0790	−69.57	0.0222	8.07	0.0296	−18.87
7	0.0196	124.14	0.0439	0.00	0.0289	51.72	0.0178	146.94	**0.0948**	−53.67	0.0671	−34.58	0.0179	144.74	0.0606	−27.59
8	0.0146	29.85	0.0190	0.00	0.0170	11.94	0.0044	328.01	**0.0469**	−59.51	0.0295	−35.52	0.0068	180.74	0.0205	−7.46
9	0.1592	33.33	0.2122	0.00	0.1592	33.33	0.0677	213.58	**0.2636**	−19.49	0.1644	29.12	0.0527	302.80	0.1415	50.00
10	0.0411	933.33	0.4244	0.00	0.6366	−33.33	0.2761	53.71	**1.5519**	−72.65	1.4634	−71.00	0.2144	97.99	0.2546	66.67
11	0.0260	32.43	0.0344	0.00	0.0439	−21.62	0.0447	−22.94	**0.1926**	−82.14	0.1158	−70.28	0.0322	6.87	0.0979	−64.86
12	0.0063	185.92	0.0179	0.00	0.0344	−47.89	0.0114	56.75	**0.0419**	−57.16	0.0229	−21.66	0.0245	−26.76	0.0260	−30.99
13	0.0364	1066.67	0.4244	0.00	0.3183	33.33	0.6783	−37.43	**0.8187**	−48.16	0.5926	−28.38	0.1445	193.71	0.1819	133.33
14	0.0260	188.24	0.0749	0.00	0.0707	5.88	0.0288	160.02	**0.1775**	−57.81	0.1499	−50.04	0.0934	−19.77	0.1157	−35.29
15	0.1157	22.22	0.1415	0.00	0.1415	0.00	0.0443	219.30	**0.3782**	−62.59	0.2475	−42.84	0.1273	11.11	0.1415	0.00
16	0.0398	0.00	0.0398	0.00	0.0398	0.00	0.0171	133.23	**0.0622**	−36.03	0.0445	−10.53	0.0330	20.75	0.0411	−3.13
17	0.0182	94.44	0.0354	0.00	0.0909	−61.11	0.0427	−17.25	**0.2323**	−84.77	0.2215	−84.03	0.0248	42.61	0.0509	−30.56
18	0.0326	254.55	0.1157	0.00	0.0849	36.36	0.0548	111.29	**0.6079**	−80.96	0.2066	−43.96	0.0475	143.66	0.1061	9.09
19	0.0354	56.52	0.0554	0.00	0.1157	−52.17	0.0547	1.27	**0.2800**	−80.23	0.2350	−76.45	0.0438	26.44	0.0670	−17.39
20	0.0490	1200.00	0.6366	0.00	0.6366	0.00	1.0475	−39.23	**1.1586**	−45.05	0.9699	−34.36	0.1754	262.89	0.4244	50.00

Note: The bolded font is the best result obtained for each map. The abbreviations of AVG and IMP represent average and improvement, respectively.

Table 3
Comparison of time taken for all algorithms with IQL for all maps.

Map	Time taken (s)															
	QL		IQL		A*		RRT		VG		PRM		VD		CD	
	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)
1	0.3654	96.97	0.0111	0.00	0.1620	93.17	**0.0011**	−944.03	0.1299	91.49	4.5013	99.75	0.4643	97.62	0.7661	98.56
2	0.9295	91.55	0.0785	0.00	0.2450	67.94	**0.0080**	−875.74	0.0827	4.99	5.1222	98.47	0.1852	57.60	0.7683	89.78
3	0.8493	82.29	0.1504	0.00	0.2283	34.12	0.0801	−87.77	**0.0100**	−1398.41	5.1878	97.10	0.1055	−42.61	0.7541	80.06
4	0.6906	93.57	0.0444	0.00	0.1363	67.40	**0.0264**	−68.14	0.0676	34.26	5.6949	99.22	0.1951	77.23	0.7315	93.93
5	7.7272	72.51	2.1242	0.00	0.2644	−703.49	2.2521	5.68	**0.0042**	−50273.44	6.8130	68.82	0.1408	−1408.21	0.7329	−189.82
6	5.9316	68.75	1.8536	0.00	0.1989	−832.13	29.7785	93.78	**0.0168**	−10904.78	8.1577	77.28	0.1722	−976.42	0.7425	−149.65
7	7.9359	76.49	1.8660	0.00	0.4400	−324.08	13.7219	86.40	**0.0566**	−3194.29	8.1601	77.13	0.2384	−682.67	0.7352	−153.82
8	15.4530	65.45	5.3389	0.00	**0.1459**	−3559.43	1127.4683	99.53	1.3201	−304.42	24.4529	78.17	3.4026	−56.91	0.6824	−682.34
9	0.2263	92.97	**0.0159**	0.00	0.0558	71.49	0.0754	78.92	0.1287	87.64	6.7291	99.76	0.3931	95.95	0.6110	97.40
10	0.4929	88.57	0.0563	0.00	0.2282	75.31	**0.0002**	−23469.69	0.0046	−1125.13	4.5078	98.75	0.0849	33.61	0.7480	92.47
11	1.7599	65.50	0.6071	0.00	0.2159	−181.25	1.4701	58.70	**0.0533**	−1038.02	7.4117	91.81	0.1843	−229.48	0.7138	14.95
12	15.9508	72.91	4.3212	0.00	0.2849	−1416.61	56.2149	92.31	**0.0103**	−41752.28	8.4477	48.85	0.2295	−1782.80	0.7366	−486.64
13	0.8893	79.52	0.1821	0.00	0.1878	3.03	**0.0054**	−3245.50	0.0056	−3126.97	5.7337	96.82	0.1295	−40.59	0.6714	72.88
14	3.7169	69.64	1.1285	0.00	0.3329	−238.99	1.1287	0.02	**0.0086**	−13091.70	5.7043	80.22	0.1082	−943.38	0.7334	−53.87
15	1.5083	98.94	0.0160	0.00	0.0997	83.92	0.6435	97.51	**0.0050**	−220.55	6.3806	99.75	0.1193	86.56	0.6305	97.46
16	1.8431	46.23	0.9911	0.00	0.0600	−1553.12	5.4670	81.87	**0.0279**	−3448.44	11.0190	91.01	0.3404	−191.14	0.5679	−74.53
17	1.6563	80.04	0.3305	0.00	0.1275	−159.17	95.0360	99.65	**0.0876**	−277.20	7.7470	95.73	0.2032	−62.65	0.6503	49.17
18	4.4119	94.66	**0.2357**	0.00	0.6384	63.07	1.8290	87.11	1.6161	85.41	8.6120	97.26	1.3457	82.48	0.7272	67.59
19	2.6801	78.03	0.5888	0.00	0.2783	−111.54	5.4840	89.26	**0.0062**	−9371.44	6.3147	90.68	0.1059	−455.71	0.7337	19.75
20	0.4489	90.77	0.0414	0.00	0.2229	81.42	0.1025	59.58	**0.0020**	−1975.29	5.8009	99.29	0.0506	18.23	0.7513	94.49

Note: The bolded font is the best result obtained for each map. The abbreviations of AVG and IMP represent average and improvement, respectively.

1200.00% are achieved. The findings indicate that the integration of the distance metric and virtual moving target into the IQL has significantly improved the path smoothness. Straighter paths are produced by the IQL since the distance metric will guide the mobile robot moves towards the target. Also, the virtual moving target will form smoother paths by bypassing the dead-ends. On the other hand, the action-state decision policy of the QL does not consider the total distance, and hence, the paths with lower smoothness are generated.

Comparing the path smoothness of IQL with A*, it can be seen that the IQL has higher path smoothness than A* in 8 maps, with improvements of 5.88% to 51.72% are attained. Besides, both IQL and A* have the same path smoothness for Map 1, 15, 16 and 20. The CD also shows a similar trend to that of A*. The IQL outperformed the CD in terms of the path smoothness in 7 maps, achieving improvements of 9.09% to 133.33%. Both have the same path smoothness for Map 3, 5 and 15. The possible reason for the similar path smoothness among IQL, A* and CD is that the action-state decision policies of these algorithms are based on the distance between the next states and the target position. Upon completing the path, the used total distance is similar because these algorithms opted for the lowest total distance. Hence, this might explain why the resulted paths have identical smoothness.

Comparing IQL and RRT, it can be seen that the path smoothness obtained using the IQL is higher than that of RRT in 13 maps, with improvement ranging from 1.27% to 328.01%. The RRT expands the branches in random directions. The joints of these branches might form sharp turnings, as observed in Fig. 13(d). The smoothness of the generated path is thus lower than the IQL in most of the maps.

Also, the path smoothness obtained by the IQL is generally higher than the VD, with improvements ranging from 5.57% to 302.80% are obtained in 19 maps. The generated path by VD usually crosses the middle of openings or passages and has relatively large turning around each vertex of the walls, leading to lower path smoothness. This can be seen in Fig. 13(g) that the path generated by VD is made up of several large 'U' shaped turnings around the vertices of walls.

It can be seen in Table 2 that the IQL has lower path smoothness than VG in all maps. The navigation paths produced by VG, in fact, are the smoothest among all algorithms for all maps. This is due to VG forms the nodes using the vertices of obstacles. Not only producing the shortest path, but the resulting navigation path also is with the least turning points. The lower the number of the turning point, the higher the path smoothness is. As shown in Fig. 13(e), the path generated by VG consists of only 3 turning points.

The PRM also has higher path smoothness than IQL in all maps, except for Map 9. The possible reason is that the high number of nodes (1000 nodes in this case) used by the PRM enables a higher possibility to select a smoother path than the IQL.

4.2.2. Computational time

Table 3 shows the comparison of computation time taken by all algorithms to complete the navigation path. Comparing to QL and PRM, the IQL is significantly a faster method, with at least 46.23% improvement in terms of the computational time could be observed. The influence of the modified strategies on the path planning of IQL is again evident in the comparison with the baseline method QL. The modified strategies introduced to IQL accelerate the search for an optimal path, particularly, enable the IQL to get out from the dead-end more rapidly. The costly time consumption of PRM is due to the utilization of a large number of nodes. All nodes will be connected and subsequently, the Dijkstra algorithm is used to evaluate the shortest path from one node to another. Inevitably, a large amount of nodes exponentially increases the computation time.

Further comparison of IQL with A* shows that the computational time of path planning using IQL is lower in 10 navigation maps (Map 1, 2, 3, 4, 9, 10, 13, 15, 18 and 20), where 34.12% to 93.17% improvements are reported. In these maps, the normal path planning is mostly implemented due to the simpler environment without dead-ends (except Map 10 and Map 13) that does not frequently trigger the escape mode. For example, map 15 is an environment without any deep dead-end (see Fig. 12(o)). Hence, the inactivation of escape mode for a long duration significantly improves the computational time of IQL by 83.92% in comparison with A*. For the environments with deep dead-ends (Map 5, 6, 7, 8, 11, 12, 14, 16, 17 and 19), A* solves the problem instances faster than the proposed IQL. Consider Map 8 (see Fig. 12(h)), which is a symbolic of complex navigation environment with three deep dead-ends, the computational time used by A* is 3559.43% faster than IQL. The result shows that considerable time is consumed by the escape mode to overcome the deep dead-end entrapment. In addition to frequent activation of the escape mode strategy, the imposed condition to terminate the escape mode contributes to the increasing computational time as well. As discussed in Section 3.3, the escape mode is terminated when the IQL reaches a state that changes the value of the TD_{lowest}. This makes intuitive sense that a huge amount of time is consumed to find the corresponding state if its position is far from the place where the escape mode is triggered. For example, there is a deep dead-end at (14,1) in Map 6 (see Fig. 12(f)). The IQL has to overcome an S-shaped curve at (15,10) to find a state that contributes to the lower total distance as defined in Equation (5). Such exploration along all the way requires far more computation than A*, in particular, a difference of 832.13% in terms of the computational time is reported. On the other hand, the computation time used by the IQL in Map 10 and Map 13 is lower than A* even though these maps have deep dead-ends. These deep dead-ends are easily bypassed by IQL through the virtual target concept.

Table 3 shows that the computational time of IQL is 14.95–98.56% faster than CD in 13 navigation maps. The performance comparison between IQL and CD shows an interesting trend: the navigation maps that IQL outperforms the CD are the same as those in A*, with the addition of Map 11, Map 17 and Map 19. This is due to both A* and CD are grid-based path planning algorithm; however, the time taken by CD to complete the run is longer because it has a higher resolution of space decomposition than A*. Similarly, CD succeeded in producing a valid path in a considerably faster way than IQL for those environments with deep dead-ends. The reason is that the nodes located around the deep dead-ends will not be selected (through sorting with the total distance used) as the potential nodes to form the optimal path due to they contribute to a longer total distance, hence this indirectly speeds up the computation of CD.

The IQL performance is superior to that of the RRT, where the reduction of computation time from 0.02 to 99.65% can be observed in 14 maps. Contradicting to A*, the RRT shows a very long planning time for those complex navigation maps with narrow passages. This is as expected as the complex maps inhibit the expansion of RRT branches. The RRT needs to randomly pick a node in the space and form a branch using the selected node. If the expanded branch collides with the obstacles, the branch will be eliminated. A complex map such as Map 8 (see Fig. 12(h)) with many narrow passages causes an enormous amount of branches to be eliminated and hence, is time-consuming. For a relatively simple map, the path planning time of RRT is shorter than IQL due to the former is not restricted to grid-to-grid movement. As such, the RRT can move rather freely to a far distance within one step, while the movement of IQL is constrained to only one unit per step. An environment with wide passages even benefits the RRT further since it encourages the expansion of RRT branches. This is evidenced in Map 2 (see Fig. 13(d))

Table 4
Comparison of shortest distance for all algorithms with IQL for all maps.

Map	Shortest distance (unit)															
	QL		IQL		A*		RRT		VG		PRM		VD		CD	
	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)
1	39.12	25.33	29.21	0.00	29.21	0.00	36.12	19.12	**26.97**	−8.30	28.07	−4.07	35.27	17.17	27.90	−4.71
2	48.80	28.39	34.94	0.00	33.90	−3.08	57.05	38.75	**30.30**	−15.33	36.30	3.74	51.34	31.93	32.99	−5.92
3	45.09	23.27	34.59	0.00	33.21	−4.16	42.82	19.20	**29.38**	−17.75	32.68	−5.85	44.22	21.77	31.47	−9.93
4	41.52	26.82	30.38	0.00	30.38	0.00	47.35	35.83	**27.94**	−8.76	30.00	−1.29	38.24	20.55	30.02	−1.22
5	126.06	22.90	97.19	0.00	94.87	−2.45	110.44	11.99	**82.69**	−17.54	91.66	−6.04	103.92	6.47	90.35	−7.57
6	58.39	12.33	51.19	0.00	49.70	−3.00	63.77	19.73	**39.09**	−30.94	50.55	−1.27	56.64	9.63	43.16	−18.61
7	87.74	14.91	74.66	0.00	72.08	−3.57	93.11	19.82	**61.14**	−22.12	69.20	−7.89	79.86	6.52	66.86	−11.67
8	115.12	7.07	106.98	0.00	106.98	0.00	135.08	20.80	**87.51**	−22.25	97.28	−10.01	122.16	12.42	100.62	−6.33
9	33.11	9.19	30.07	0.00	30.07	0.00	39.67	24.20	**25.86**	−16.29	28.29	−6.28	32.75	8.19	28.22	−6.56
10	41.07	22.89	31.67	0.00	30.97	−2.27	33.61	5.77	**28.30**	−11.94	29.59	−7.02	33.87	6.48	30.44	−4.04
11	60.05	21.21	47.32	0.00	42.04	−12.55	59.91	21.02	**34.31**	−37.91	42.67	−10.89	55.69	15.03	36.66	−29.07
12	166.33	18.72	135.19	0.00	129.80	−4.16	151.62	10.83	**115.08**	−17.48	124.66	−8.45	141.78	4.64	121.69	−11.10
13	46.12	23.96	35.07	0.00	35.07	0.00	37.40	6.24	**32.76**	−7.06	35.03	−0.13	37.32	6.03	34.69	−1.11
14	77.51	22.01	60.44	0.00	58.63	−3.10	73.70	17.99	**51.09**	−18.31	58.12	−4.00	65.78	8.11	56.86	−6.29
15	36.34	11.56	32.14	0.00	32.14	0.00	40.54	20.72	**28.22**	−13.88	33.90	5.18	35.55	9.59	30.44	−5.58
16	67.91	1.99	66.56	0.00	66.56	0.00	73.34	9.25	**57.07**	−16.62	61.47	−8.27	70.05	4.99	61.53	−8.17
17	61.59	19.82	49.39	0.00	49.21	−0.36	62.71	21.24	**44.60**	−10.74	48.65	−1.53	55.37	10.80	46.55	−6.10
18	69.99	21.02	55.28	0.00	51.60	−7.13	72.69	23.96	**47.54**	−16.28	55.29	0.02	59.01	6.33	50.62	−9.20
19	41.56	15.73	35.03	0.00	34.87	−0.45	41.83	16.27	**30.33**	−15.49	55.85	37.28	36.28	3.46	32.84	−6.65
20	40.23	20.92	31.81	0.00	30.97	−2.71	34.62	8.11	**28.82**	−10.38	31.99	0.57	33.78	5.84	30.44	−4.49

Note: The bolded font is the best result obtained for each map. The abbreviations of AVG and IMP represent average and improvement, respectively.

that RRT could pass through the wide passages between the walls to reach the target position with an average time taken of 0.0080 s.

The IQL has a shorter computation time than VD and VG in only 8 and 5 maps, respectively. The efficiency of VG and VD attributed to these algorithms form the roadmap based on the shape and number of obstacles. The VG and VD only decompose the space around the obstacles, but IQL evaluates every grid in the whole environment. The representation of obstacles through several nodes by VG and rough decomposition of the obstacles cluster into a large group by VD significantly simplify the path planning process. Moreover, the VG and VD do not suffer from the dead-end problem because the roadmaps are solved using the Dijkstra algorithm with a mechanism that sorts nodes according to the total distance. As a result, both VG and VD show very good performance when the total number of vertices or the number of obstacles is less. Map 5 with just 4 walls is a good example where VG and VD are excelling. In this map, only several nodes and lines are formed around the vertices of walls (see Appendix A-4). However, both VG and VD suffer from long computation time when dealing with an obstacle-rich environment, such as map 18 in Fig. 12(r). In this case, the computation times of VG and VD are 85.41% and 82.48% longer than IQL, respectively.

4.2.3. The shortest travelled distance

Table 4 presents the shortest distance that the mobile robot takes to reach the target position in different environments. Comparing to QL, RRT and VD, it can be seen that the IQL uses shorter distance for all maps, with the improvements of 7.07–28.39%, 5.77–38.75% and 3.46–31.93% are observed for QL, RRT and VD, respectively. These improvements suggest that the introduction of the distance metric can direct the mobile robot towards the target position using a shorter path. Besides, the moving target concept is effective for bypassing the dead-end, thus creating a shorter path than others. On the other hand, the paths of RRT contain a large turning radius at the corners of obstacles, due to the expansion of RRT branches is picked in random length and direction. Taking Map 2 in Fig. 13 as an example, the IQL creates a path that crosses the vertices of walls and moves directly towards the target, whereas the RRT creates a path with a larger turning radius and sharp turning angle at the vertices of the walls. Hence, a longer shortest distance is attained by the RRT in comparison with the IQL.

The optimal path generated by the IQL has a shorter distance than that of PRM in 5 maps, with the improvements within 0.02–37.29%. The path planning mechanism of the PRM that randomly positions the nodes in the environment encourages the PRM to find a shorter path than IQL. Note that both RRT and PRM possess random feature (expansion of branches and random placement of nodes), but the amount of nodes placed by the PRM is massive as compared to the branches of the RRT. This allows PRM for a higher possibility to access the area around the obstacles, thus potentially creating a shorter optimal path but at the cost of longer computational time.

The sorting mechanism of the A* algorithm allows it to select nodes with a shorter distance. As a result, the IQL is only comparable to the A* algorithm in 7 maps. Moreover, the IQL could not apply the distance metric when the escape mode is activated, but the distance-based sorting mechanism of the A* algorithm is used at all times. The inactivation of the distance metric in the IQL leads to path fluctuation in some runs. Meanwhile, the A* algorithm is a deterministic algorithm, meaning that an identical path is generated for all simulations. Hence, it is not surprising that the average shortest distance of IQL is longer than the A* in some maps.

It is apparent from Table 4 that the CD outperforms the IQL in terms of the shortest distance for all the considered environments. The space decomposition of CD has a higher resolution than the IQL, which employs a grid decomposition for the environment. A higher resolution in space configuration enables the CD to pick a path that is closer to the obstacle vertices and move further towards the target. An example is shown in Map 2 (Fig. 13). The CD can create a path closer to the vertices of walls, while the generated path by the IQL has a relatively larger distance from those vertices and hence, resulting in a longer path distance.

The superiority of accessing the areas around the vertices of obstacles becomes increasingly obvious when the VG can get an optimal path with the shortest distance in all maps as compared to others. The vertex to vertex moving motion can be seen in Map 2 (see Fig. 13(e)). Such a moving pattern results in a shorter path, in comparison with the moving in the passages. Hence, the obtained shortest distance is lower than the others, including the IQL.

4.2.4. The total travelled distance

Table 5 shows the comparison of the total travelled distance that the mobile robot takes to reach the target position in 30 runs for all maps.

Table 5
Comparison of total distance used for all algorithms with IQL for all maps.

Map	Total distance used (unit)														
	QL		IQL		A*		RRT		VG		PRM		VD		CD	
	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)	AVG	IMP (%)
1	67,228	99.47	353	0.00	630	43.99	**281**	−25.58	14,514	97.57	335,402,312	100.00	288	−22.74	2733	87.08
2	120,638	96.05	4771	0.00	1402	−240.19	989	−382.28	11,497	58.50	335,397,647	100.00	**260**	−1737.53	2733	−74.59
3	114,906	93.15	7872	0.00	1754	−348.89	2422	−225.06	1151	−583.69	335,399,249	100.00	**266**	−2855.15	2733	−188.06
4	91,669	97.73	2080	0.00	847	−145.61	1876	−10.88	8181	74.57	335,400,557	100.00	**322**	−546.30	2733	23.88
5	992,357	89.18	107,368	0.00	1143	−9296.35	15,070	−612.46	603	−17692.08	335,400,011	99.97	**388**	−27573.52	2733	−3828.86
6	779,655	87.90	94,311	0.00	1110	−8397.72	50,767	−85.77	1803	−5130.43	335,402,436	99.97	**366**	−25646.53	2733	−3351.05
7	1,020,472	90.84	93,429	0.00	1342	−6860.26	39,445	−136.86	6825	−1268.94	335,401,500	99.97	**379**	−24541.23	2733	−3318.78
8	1,939,165	89.41	205,414	0.00	2489	−8152.52	801,502	74.37	302,232	32.03	335,488,391	99.94	**1319**	−15478.84	4164	−4832.75
9	28,825	98.44	**450**	0.00	480	6.31	6454	93.04	13,848	96.75	335,391,057	100.00	549	18.11	2733	83.55
10	70,649	96.12	2739	0.00	1133	−141.76	**61**	−4370.65	395	−592.59	335,400,290	100.00	222	−1132.48	2733	−0.22
11	252,312	87.98	30,340	0.00	1162	−2511.23	9111	−233.00	5100	−494.87	335,401,712	99.99	**374**	−8011.05	2733	−1010.22
12	2,238,195	90.44	214,014	0.00	960	−22203.16	54,292	−294.19	1172	−18157.46	335,400,877	99.94	**464**	−46017.26	2733	−7731.27
13	126,697	93.42	8340	0.00	1105	−654.85	599	−1293.28	635	−1214.00	335,401,066	100.00	**334**	−2399.93	2733	−205.19
14	535,527	89.60	55,688	0.00	1764	−3057.64	8064	−590.54	1096	−4979.44	335,398,705	99.98	**360**	−15354.15	2733	−1937.77
15	220,084	99.79	469	0.00	456	−3.05	8393	94.41	434	−8.22	335,401,400	100.00	**361**	−30.16	2733	82.82
16	252,435	87.87	30,612	0.00	**477**	−6314.23	22,803	−34.25	2868	−967.45	335,397,252	99.99	514	−5861.12	2733	−1020.16
17	247,833	93.34	16,513	0.00	**503**	−3180.40	65,793	74.90	9492	−73.97	335,399,543	100.00	511	−3130.40	2733	−504.24
18	671,342	98.23	11,880	0.00	2507	−373.87	16,799	29.28	330,418	96.40	335,524,977	100.00	**1136**	−946.16	4555	−160.83
19	409,856	92.62	30,227	0.00	996	−2935.93	18,039	−67.56	481	−6186.52	335,401,116	99.99	**238**	−12626.87	2733	−1006.08
20	68,025	97.09	1977	0.00	642	−207.72	5039	60.77	**118**	−1571.07	335,396,776	100.00	127	−1460.78	2733	27.67

Note: The bolded font is the best result obtained for each map. The abbreviations of AVG and IMP represent average and improvement, respectively.

Table 6
The number of methods where the IQL has better performance in terms of four evaluation criteria for twenty navigation maps.

Map	The obtained score			
	Path Smoothness	Computational Time	The Shortest Distance	The Total Travelled Distance
1	**6**	**6**	3	**5**
2	3	**6**	**4**	3
3	**5**	**4**	3	2
4	**4**	**6**	3	**4**
5	**4**	3	3	2
6	**5**	3	3	2
7	**4**	3	3	2
8	**4**	3	**5**	**4**
9	2	**7**	3	**7**
10	**4**	**5**	3	2
11	**6**	**4**	3	2
12	**6**	3	3	2
13	**4**	**4**	3	2
14	**5**	3	3	2
15	**4**	**6**	**5**	**4**
16	**6**	3	**4**	2
17	**6**	**4**	3	3
18	3	**7**	**4**	**4**
19	**5**	**4**	**4**	2
20	**5**	**6**	**4**	**4**

Note: The bolded font indicates that the IQL outperforms at least half of the competitive algorithms in the respective evaluation criterion.

Comparing to QL, it can be concluded that the paths planned by the IQL are shorter than the ones produced by the QL, regardless of the complexity of the maps. The significant improvements from 87.87 to 99.79% can be observed, indicating that the IQL can direct the mobile robot effectively towards the target position through the guidance of the distance metric as compared to the QL. Moreover, the modified Q function reduces unnecessary random motions during the exploitation stage. The comparative performance of IQL and PRM, again, shows that the IQL has a lower total distance than the PRM, with improvements ranging from 99.94 to 100%. The poor performance of PRM is due to many nodes are distributed in the entire environment. Each line formed between the nodes is evaluated, leading to a significantly longer total distance.

It is apparent from Table 5 that both A* and CD outperform the IQL in terms of the total distance. The IQL only improves the total distance in 2 maps (Map 1 and 9) and 5 maps (Map 1, 4, 9, 15 and 20) for A* and CD, respectively. The obtained results are intuitive. The repetitive motions of the IQL during the escape mode greatly increase the total distance travelled by the mobile robot. Besides, through the sorting mechanism in the A* and the Dijkstra algorithm used by the CD, no repetitive motions are made, contributing to a shorter total travelled distance. Nevertheless, as observed in Map 1 and Map 9, the IQL has better performance when no dead-end exists and hence, the escape mode will not be triggered.

The comparison with RRT shows that IQL surpasses the RRT in maps without deep dead-end and wide passage. Map 18 (see Fig. 12(r)) is one of the maps with those characteristics. The absence of wide passage inhibits the expansion of branches in the RRT. With the increasing number of obstacles in a narrow environment, the expanded branches of RRT will easily collide with the obstacles. The removal of those branches contributes to a higher total distance. Conversely, in the map with deep dead-end such as Map 10 (Fig. 12(j)), the total distance of the RRT turns out to be lower than the IQL. Moreover, the wide passage available in the environment enables the RRT to connect the starting point with the target point easily just by using several branches.

Since VG forms the roadmap by placing the nodes on the vertices of obstacles, the total number of vertices greatly affects the total distance reported by the VG. The nodes will be connected to form a line. With the

increasing nodes, more evaluations on the distance between the nodes are required. Hence, for the maps with a higher number of vertices, such as Map 1 and Map 8 (refer to Fig. 12(a) and Fig. 12(h)), the paths produced by the VG require a higher total distance to reach the target position in comparison with the IQL. However, the opposite finding also holds for maps with a low total number of vertices, such as Map 5 and Map 14 (refer to Fig. 12(e) and Fig. 12(n)). The paths planned by the VG require a much lower total distance than the IQL.

It is worth noting that the option of the roadmap approach, again, results in a shorter total distance than that of IQL, as observed in the obtained results using VD. Although the roadmap approach is utilized in both VG and VD, the VD is different in the way that the paths are generated between the spaces of obstacles, instead of the vertices of obstacles. Hence, fewer lines are created and this significantly reduces the total distance used to evaluate each line. Consequently, the VD uses a lower total distance in completing the path compared to VG and IQL in most of the maps.

4.3. The advantages of the proposed IQL

To further summarize the advantages of the proposed IQL with respect to other state-of-art methods, specifically, QL, A*, RRT, VG, PRM, VD and CD, Table 6 presents the number of methods in which the IQL has better performance in terms of the four evaluation aspects for each map. Exemplarily, referring to Table 5, for Map 1, the total travelled distance achieved by the IQL is shorter than that of the QL, A*, VG, PRM and CD. Hence, the IQL outperforms five state-of-art methods in this case. A score of 5 is then recorded in Table 6 for the evaluation criterion of the total travelled distance for Map 1. The IQL is considered to be a better path planning method if it outperforms or comparable with others. Also, since a total number of seven competitive algorithms are considered in this study, the IQL is said to have better performance for this kind of map, if the obtained score is at least 4, *i.e.,* more than half of the competitive algorithms in the respective evaluation criterion.

As seen in Table 6, the IQL scores 4 and above in at least three evaluation aspects for Map 1, 4, 8, 15, 18, 19 and 20. Two common characteristics can be observed in these maps: the absence of deep dead-end (Map 1, 4 and 18) and the presence of narrow passages (Map 1, 8, 15, 19 and 20). It thus can be suggested that the IQL is more preferable to be applied in the navigation map with these type of characteristics. The reasons are intuitive. Firstly, the IQL consumes a lot of time to overcome the dead-end entrapment when the escape mode is put into place. Secondly, the modified strategies introduced to IQL could accelerate the search for an optimal path in the environment with many narrow passages.

5. Conclusion

To overcome the low learning rate of QL, the features of a distance metric, modified Q function and virtual moving target were introduced into the proposed IQL. The proposed IQL and other algorithms (QL, A*, RRT, VG, PRM, VD and CD) were simulated in 20 different environments and subsequently, the path planning performances were compared in terms of the path smoothness, computational time, the shortest distance and the total travelled distance. Since the IQL is a modified QL, the QL can be served as the baseline comparison algorithm. The comparison validated that the IQL can effectively produce a smoother and shorter path, and faster computation than the QL in all the considered maps, regardless of the complexity, number and type of obstacles.

Overall, the IQL outperformed other algorithms in terms of the computational time for the navigation maps without deep dead-ends. In terms of the path smoothness and the shortest distance, the IQL gave a better performance than the RRT and VG in most of the navigation maps. Besides, the efficiency of the IQL in terms of the shorter total distance can be witnessed in the maps with a high number of vertices, narrow passages and without deep dead-ends.

One of the limitations of the proposed IQL is that the repetitive motions performed by the agent during the escape mode are time-consuming. This increases the computational time and total distance, and thus, may affect the navigation performance in real-time. In future, the escape mode can be improved to reduce time consumption. In this study, only static obstacles were considered. Future work can be extended for environments involving dynamic obstacles and multi mobile robots.

CRediT authorship contribution statement

Low Ee Soong: Conceptualization, Investigation, Methodology, Software, Writing – original draft. **Ong Pauline:** Supervision, Writing – review & editing. **Low Cheng Yee:** Supervision. **Rosli Omar:** Resources.

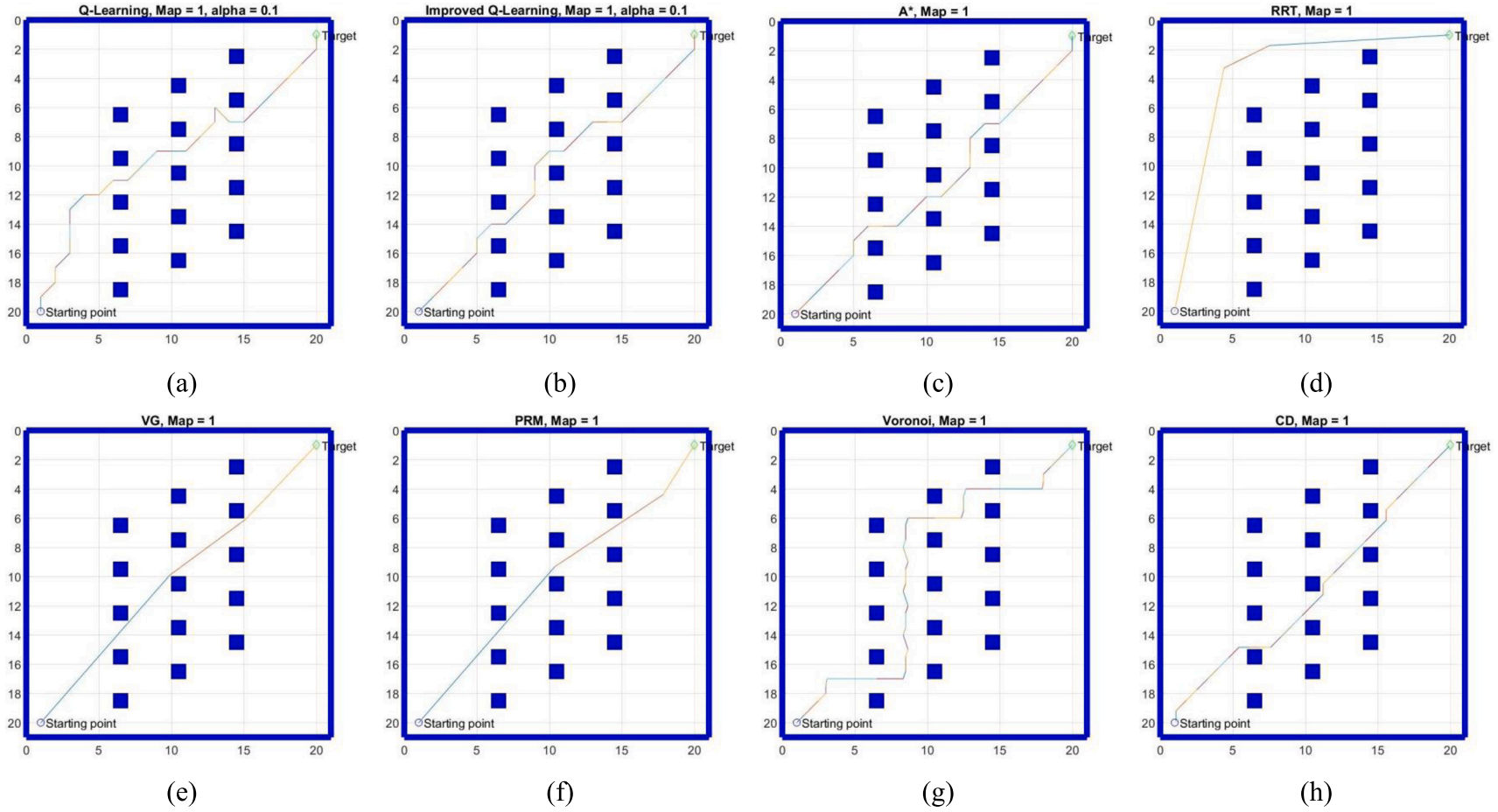
Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

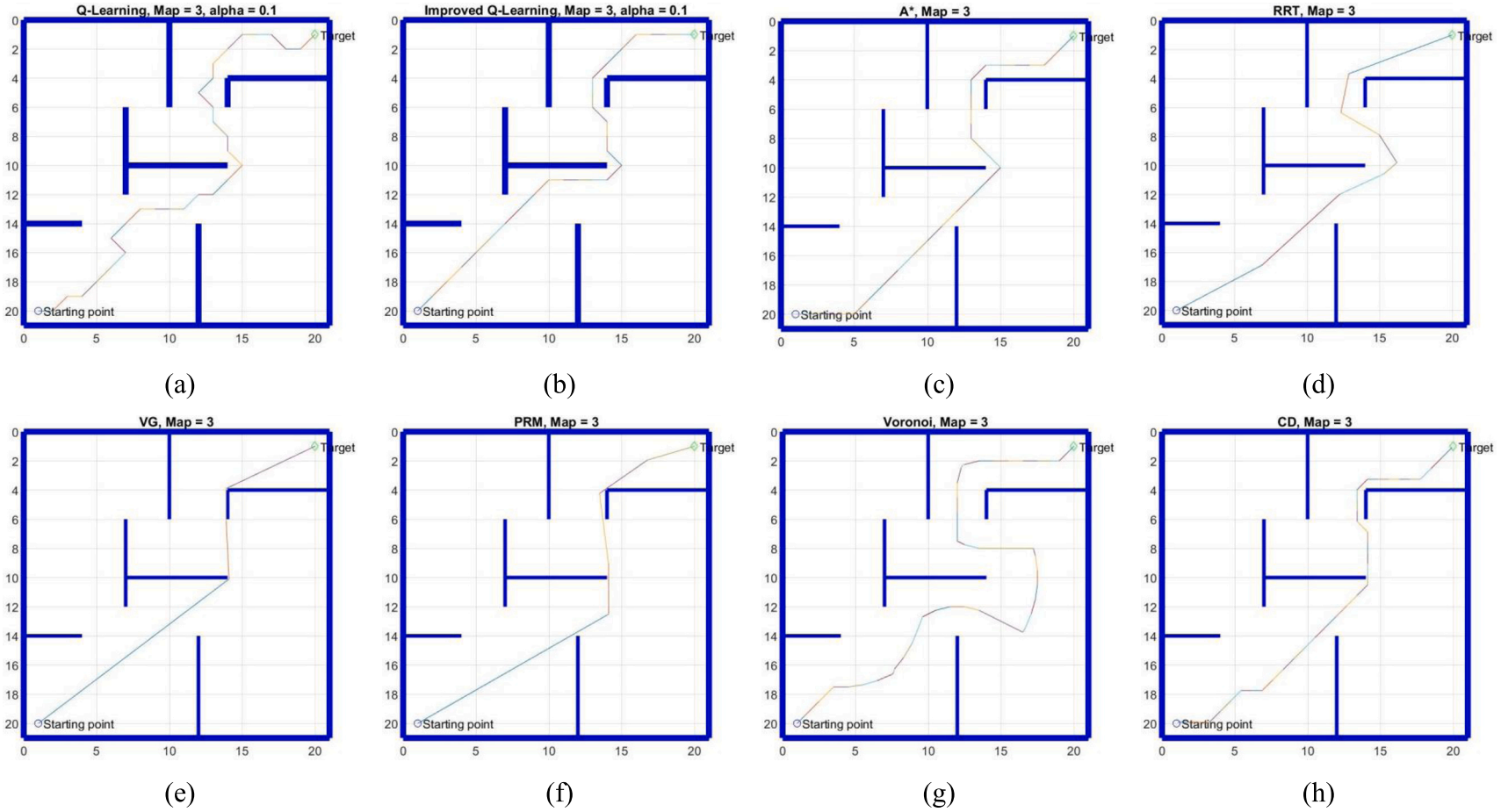
Acknowledgments

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

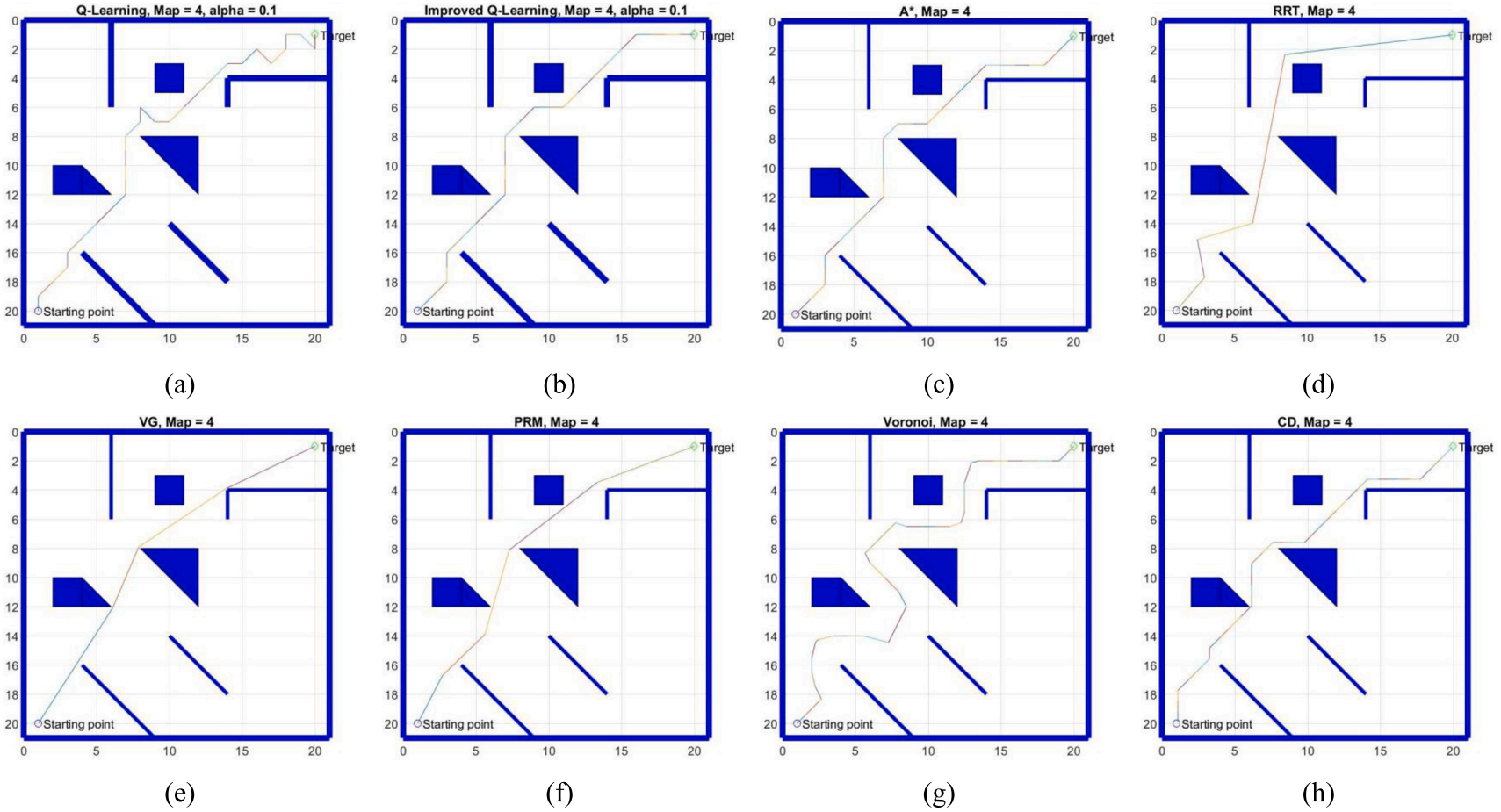
Appendix A



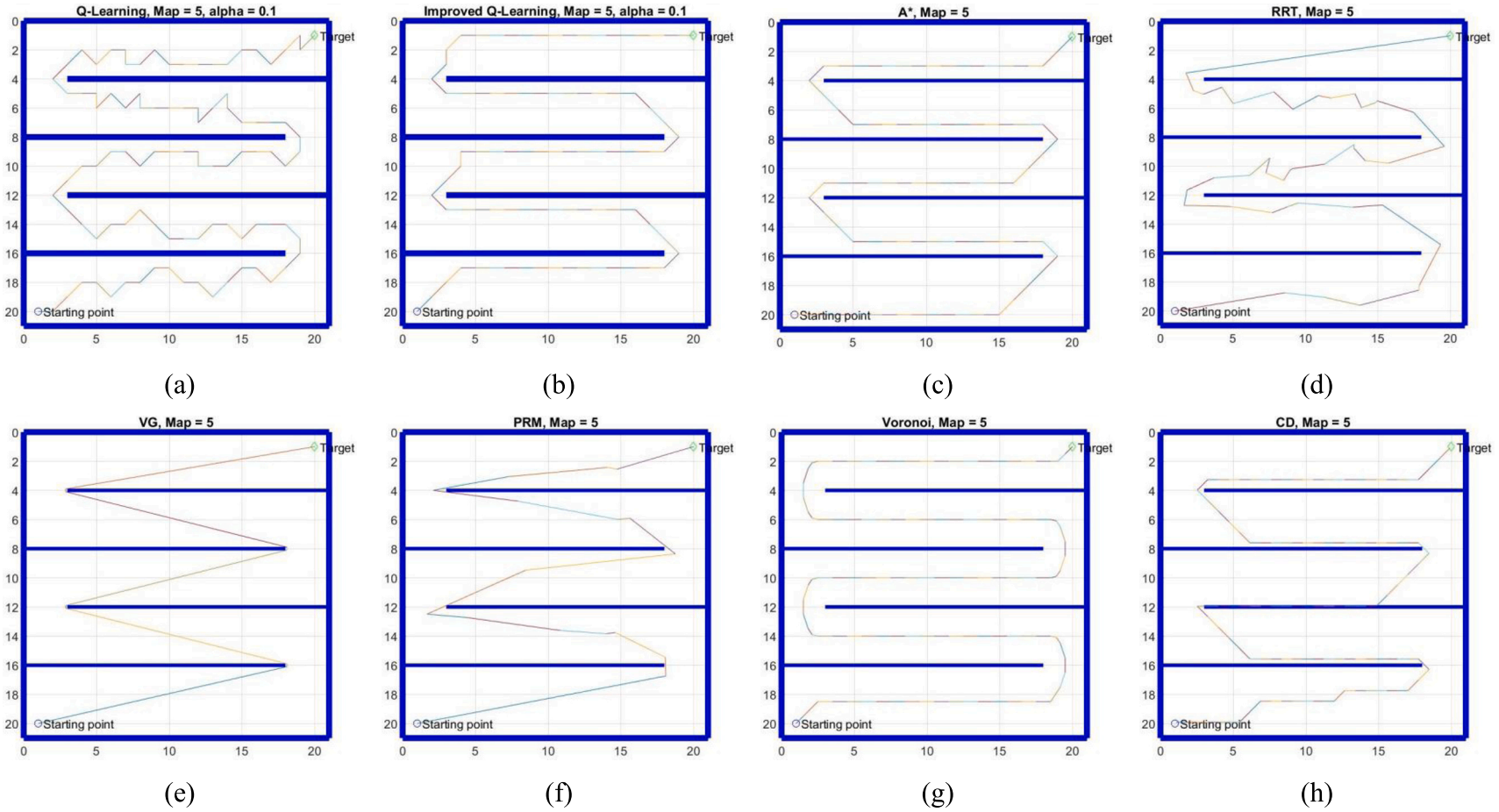
A-1. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 1



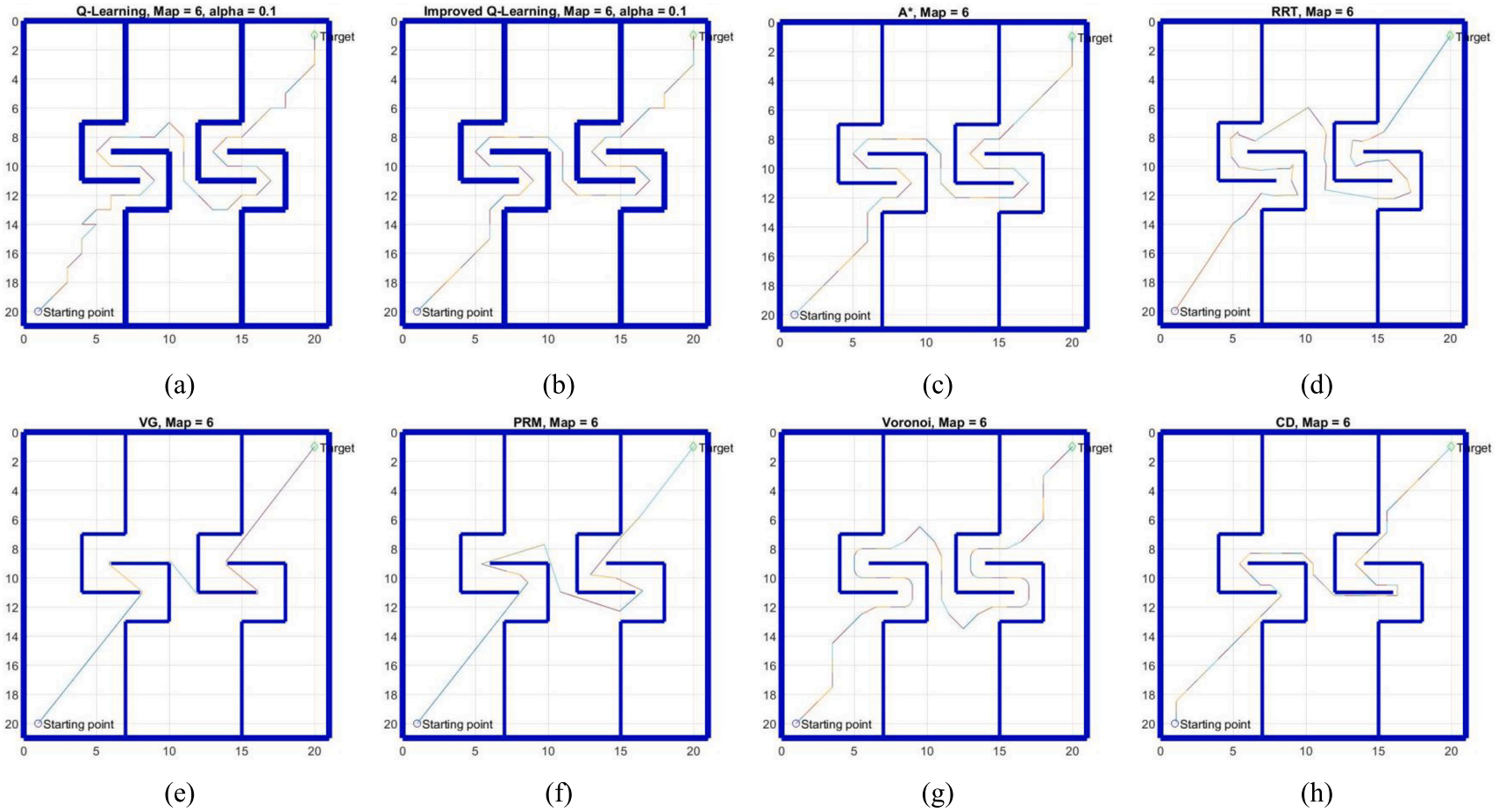
A-2. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 3



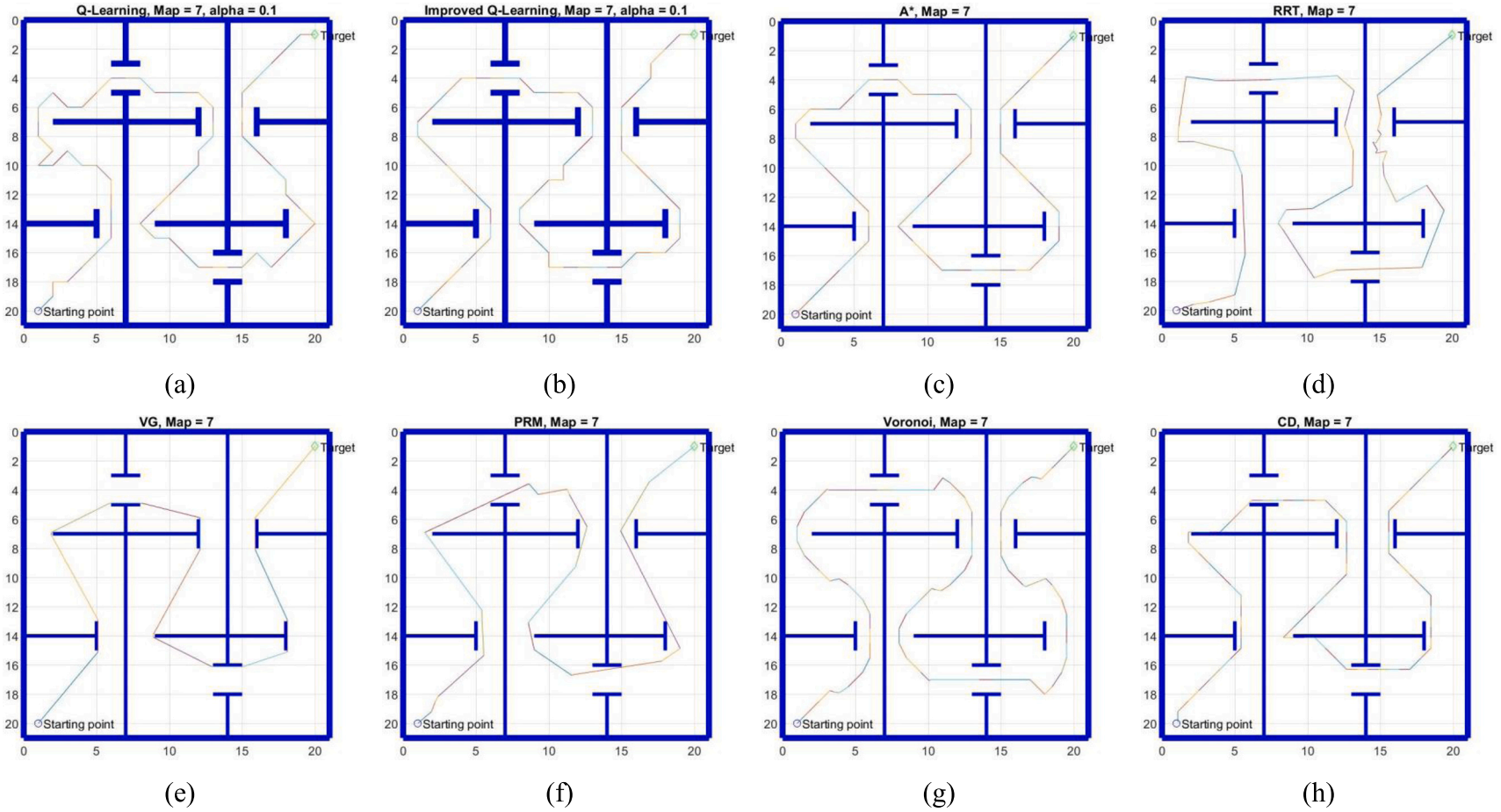
A-3. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 4



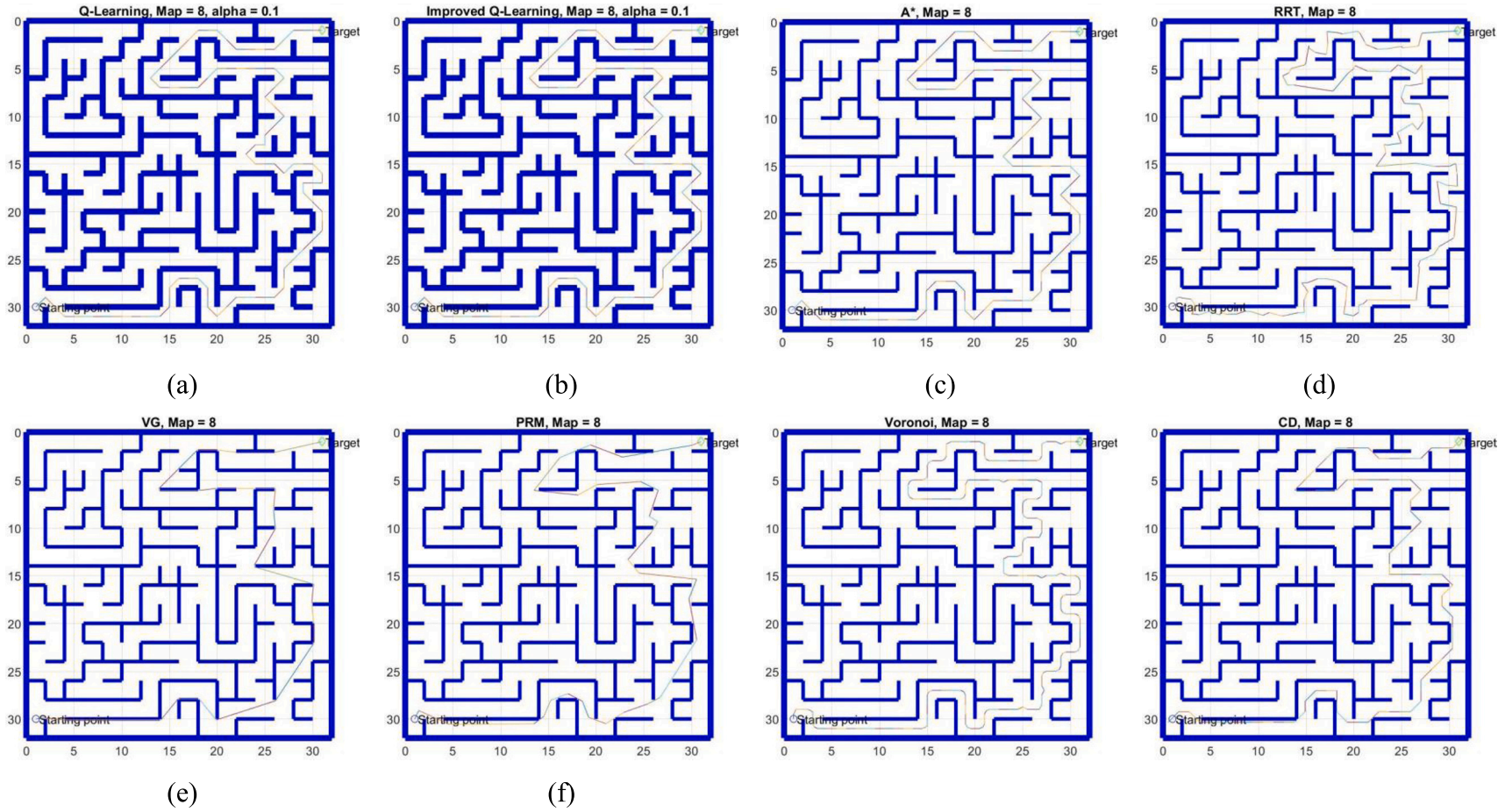
A-4. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 5



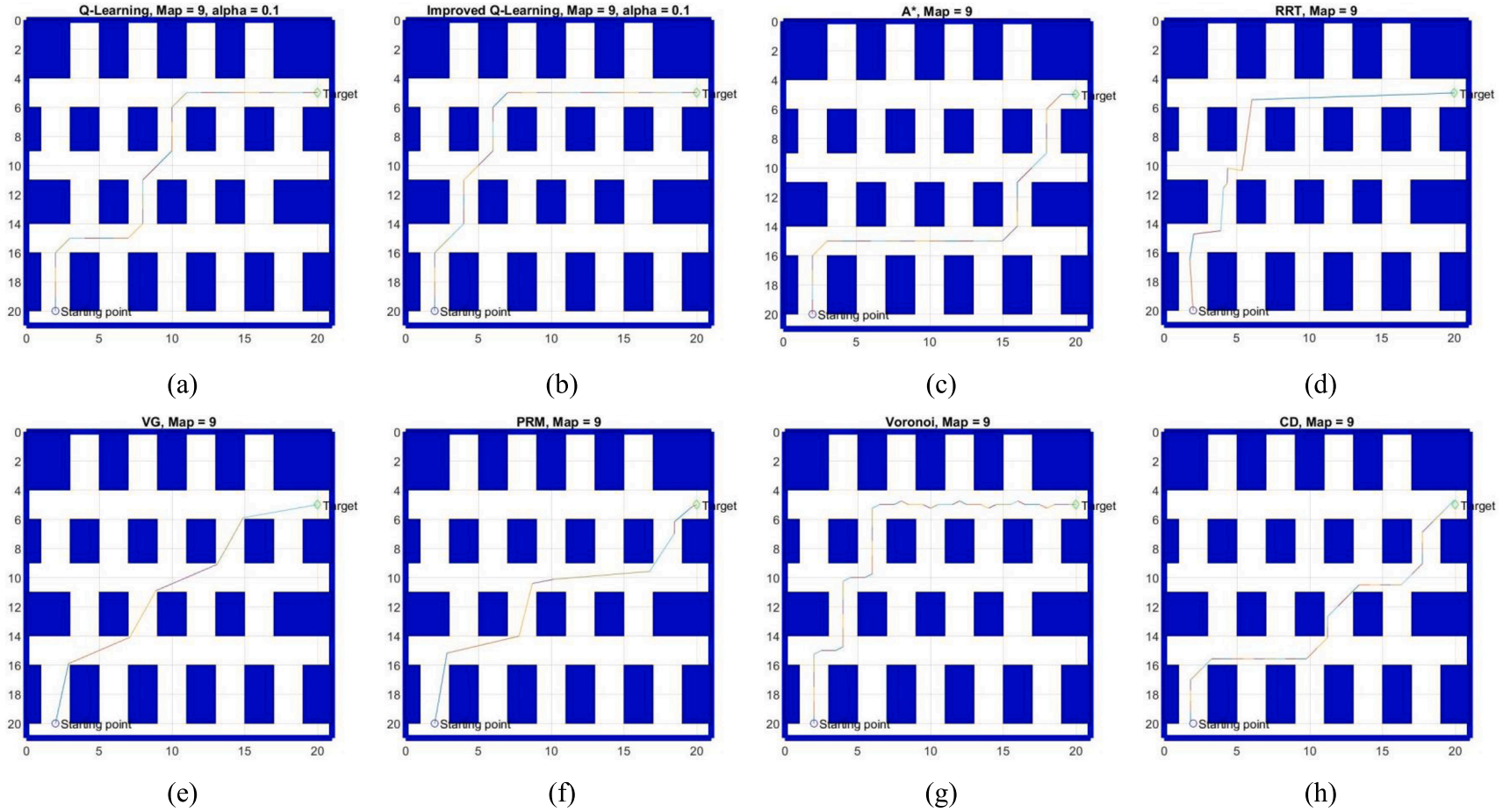
A-5. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 6



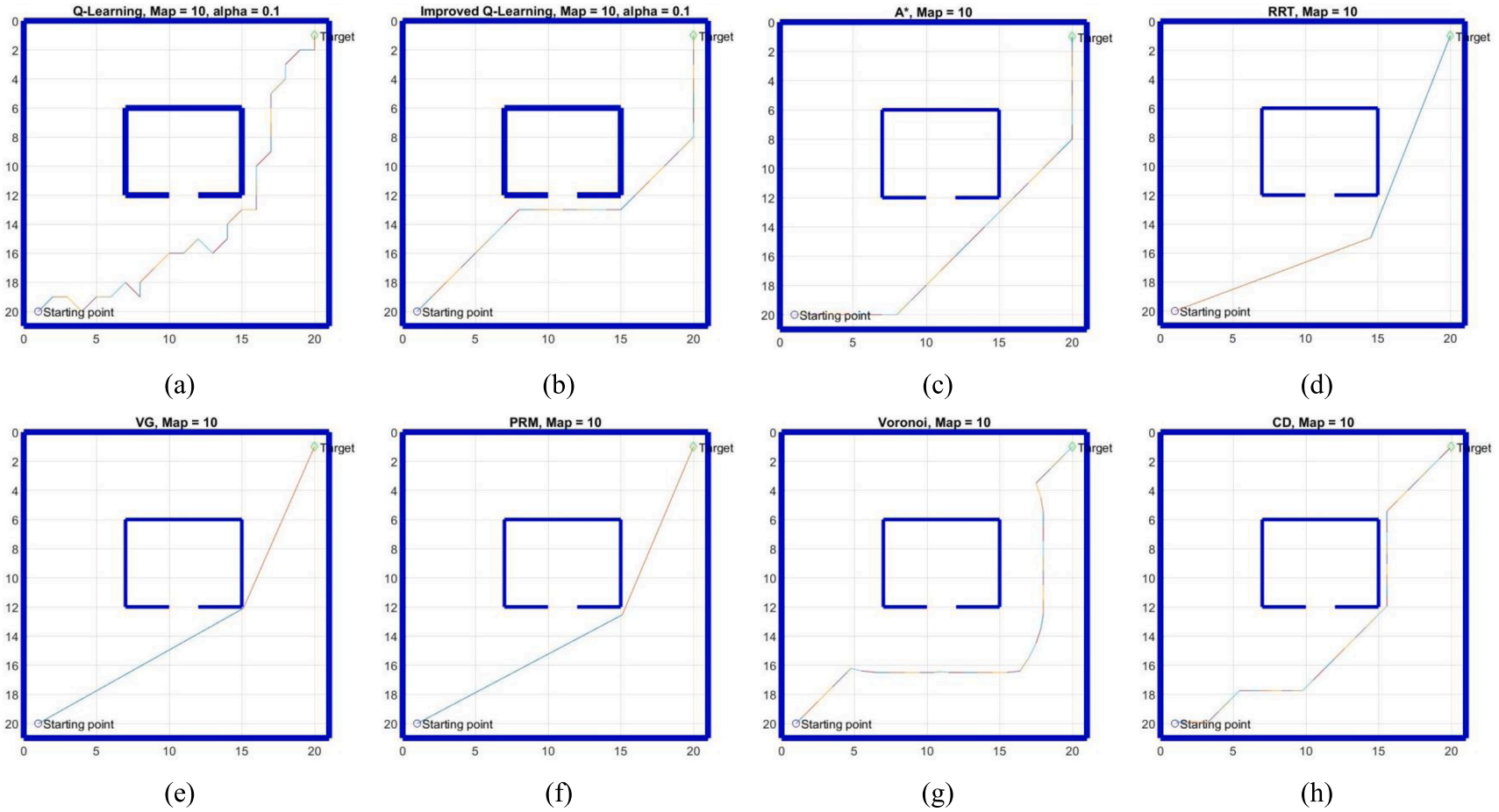
A-6. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 7



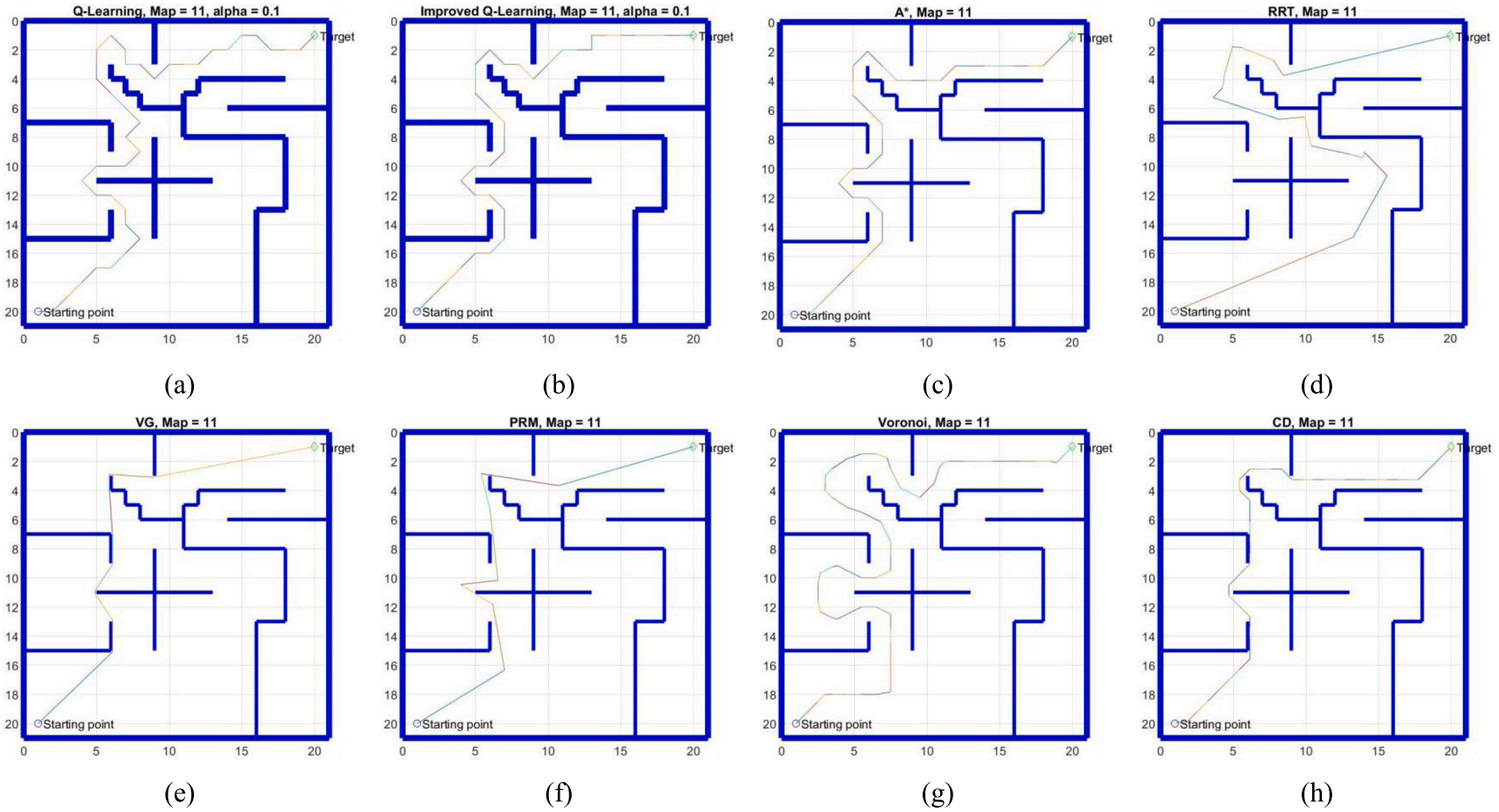
A-7. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 8



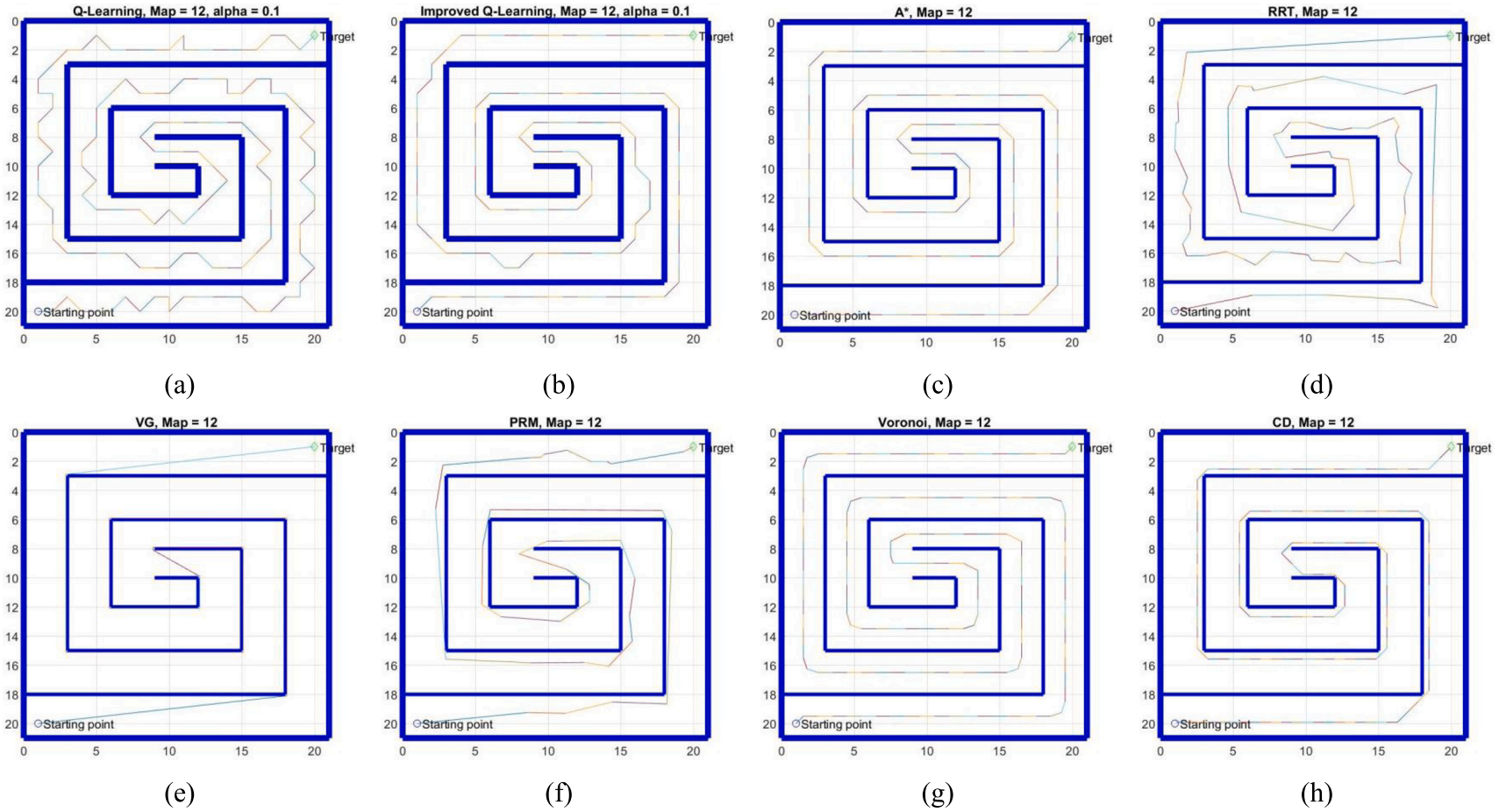
A-8. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 9



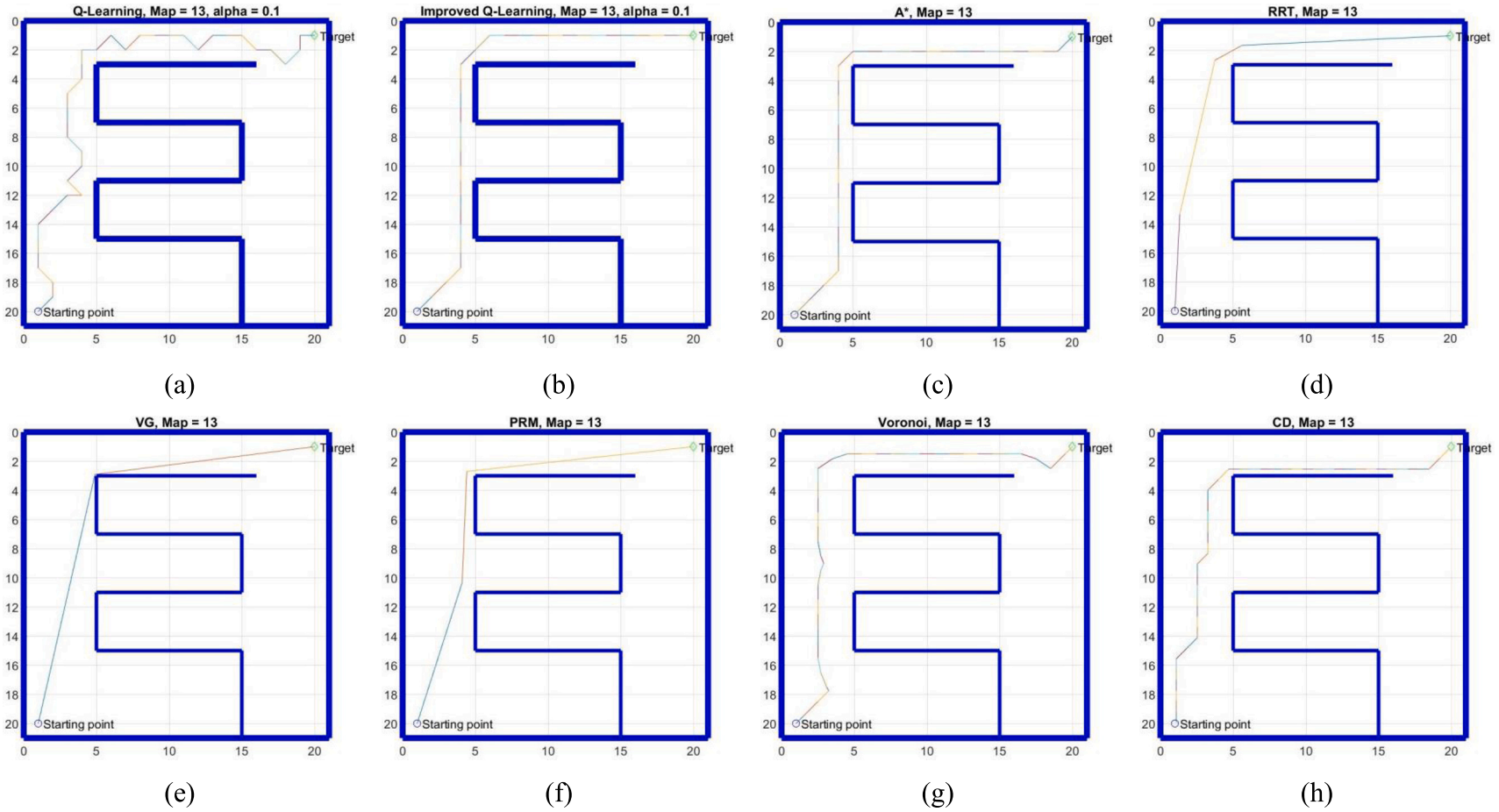
A-9. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 10



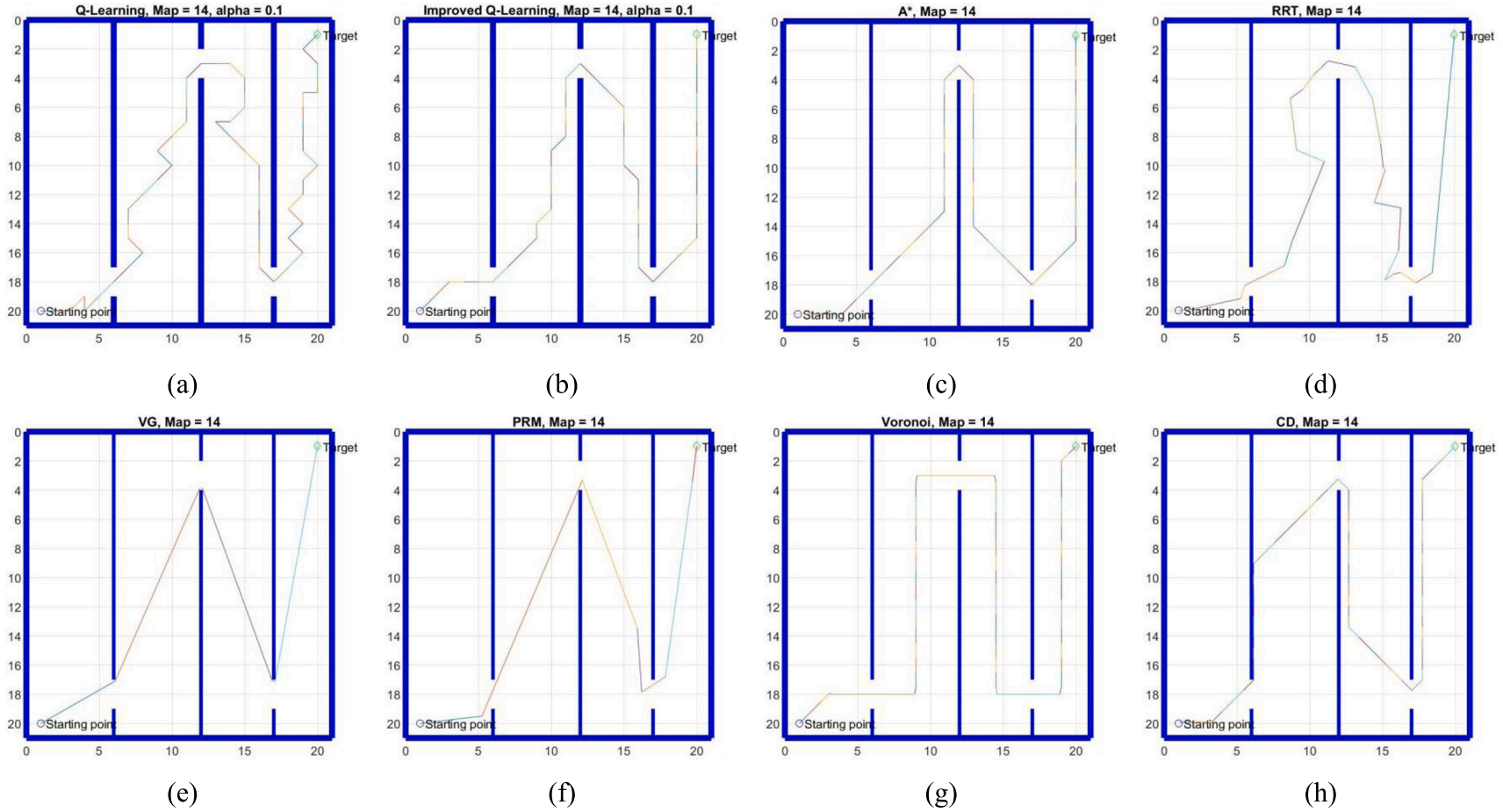
A-10. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 11



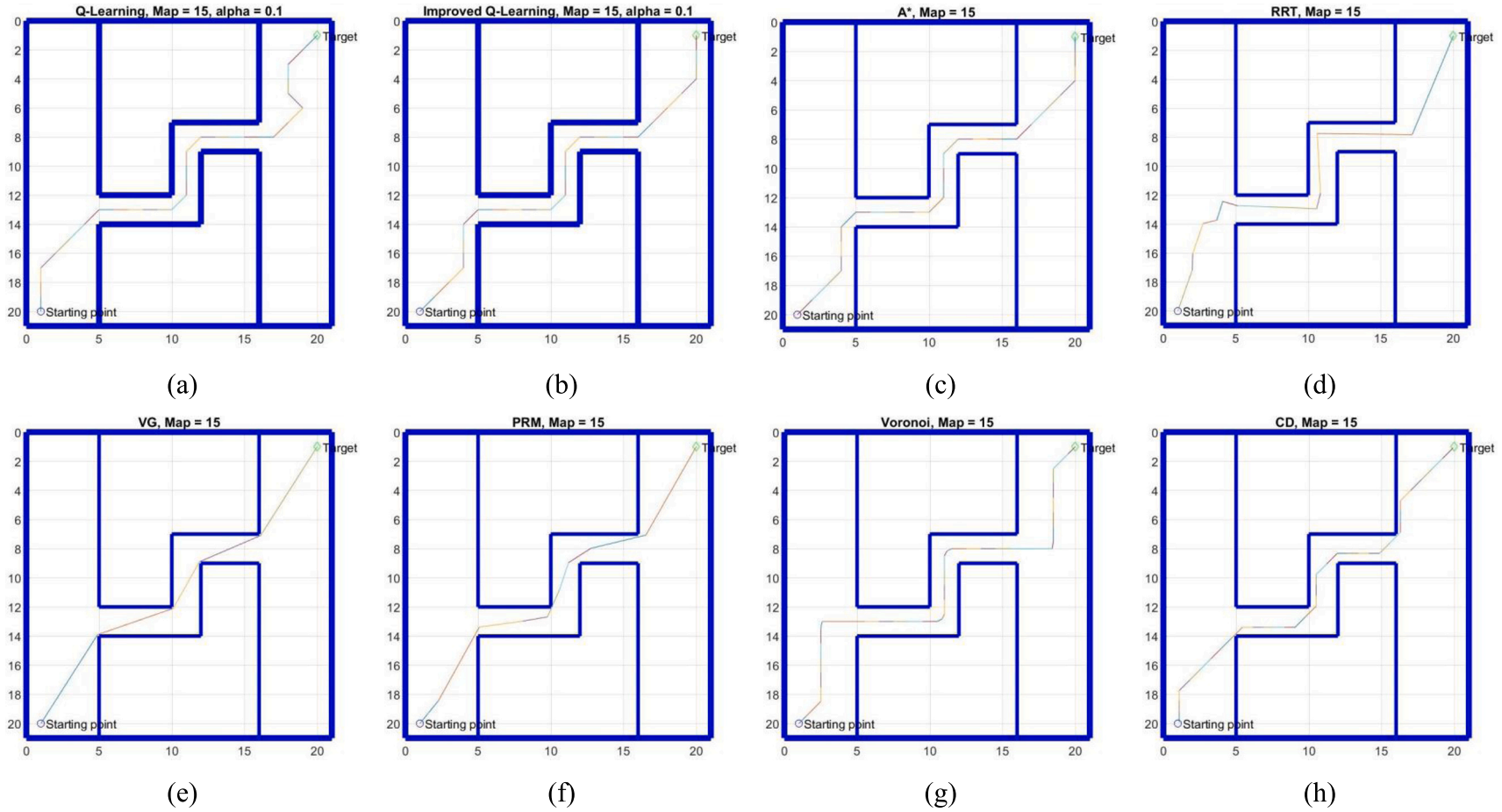
A-11. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 12



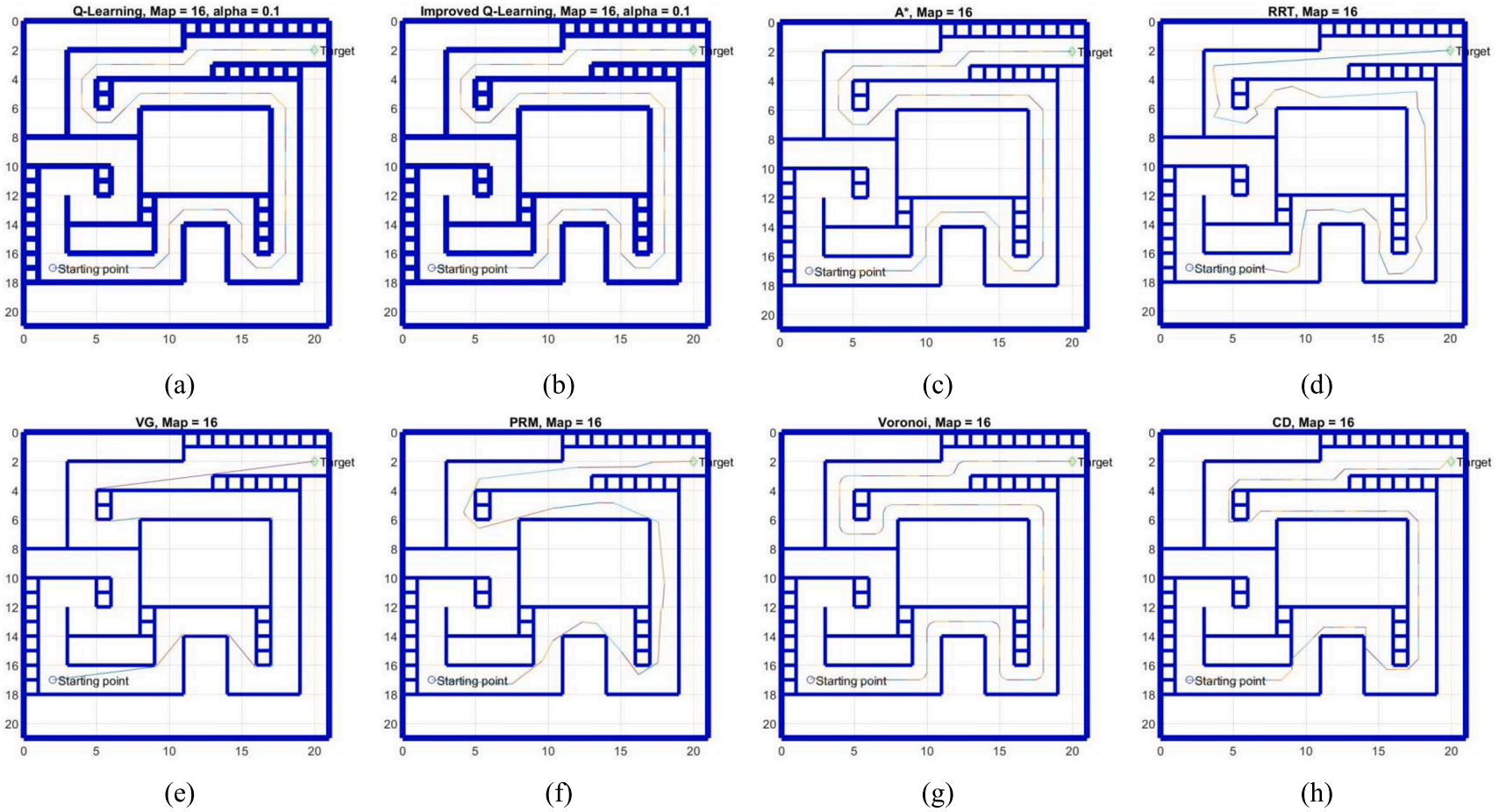
A-12. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 13



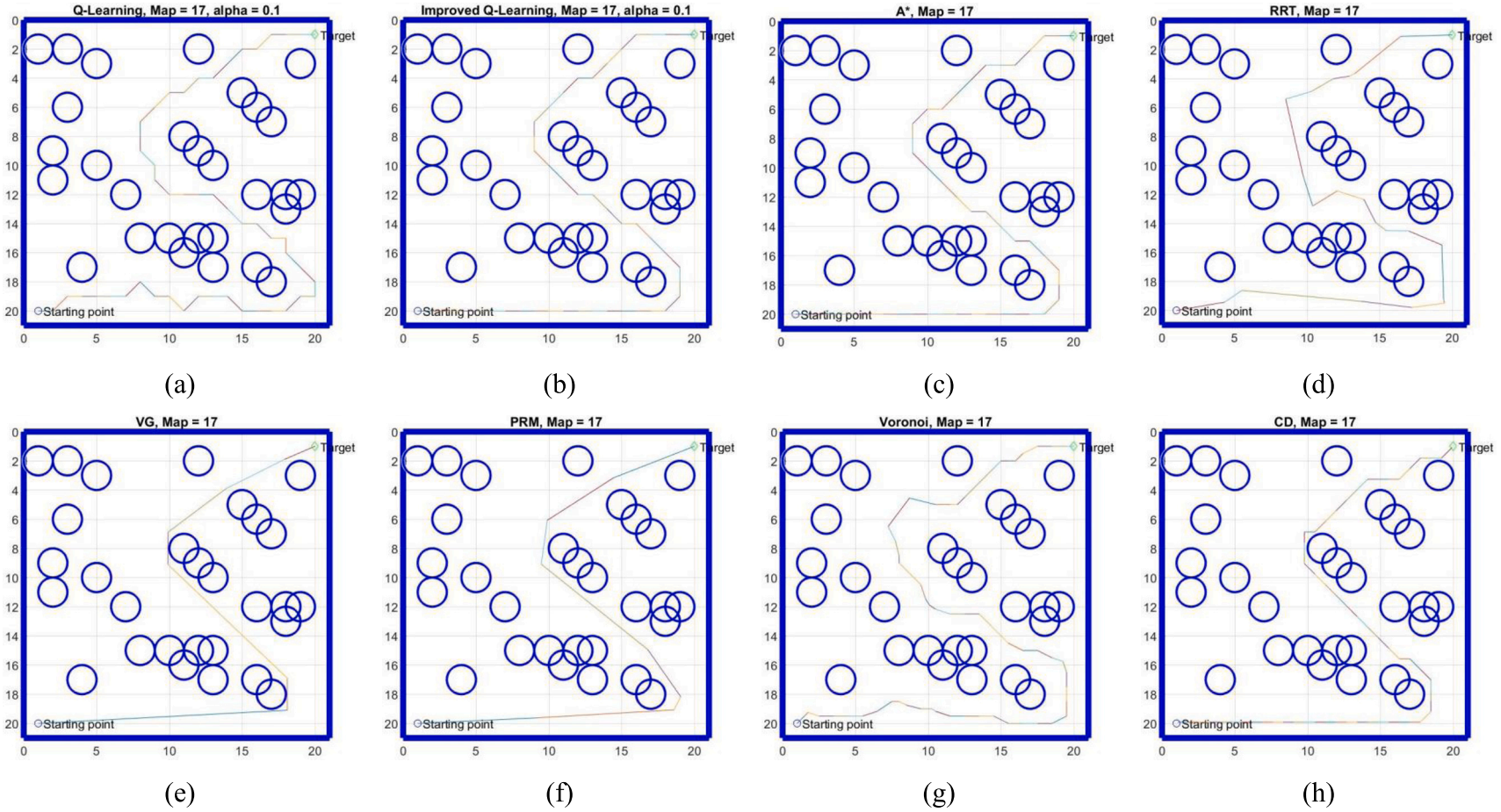
A-13. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 14



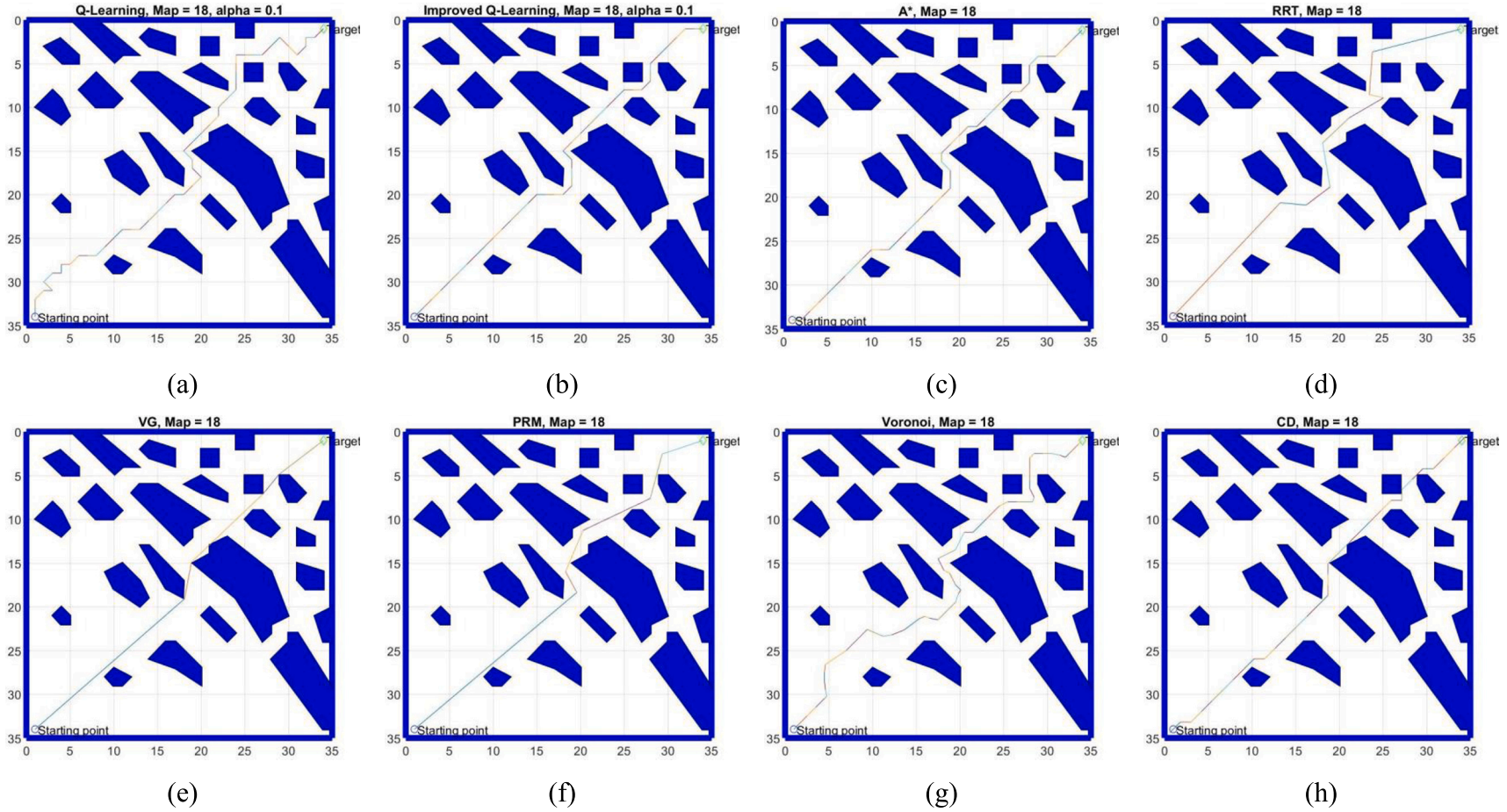
A-14. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 15



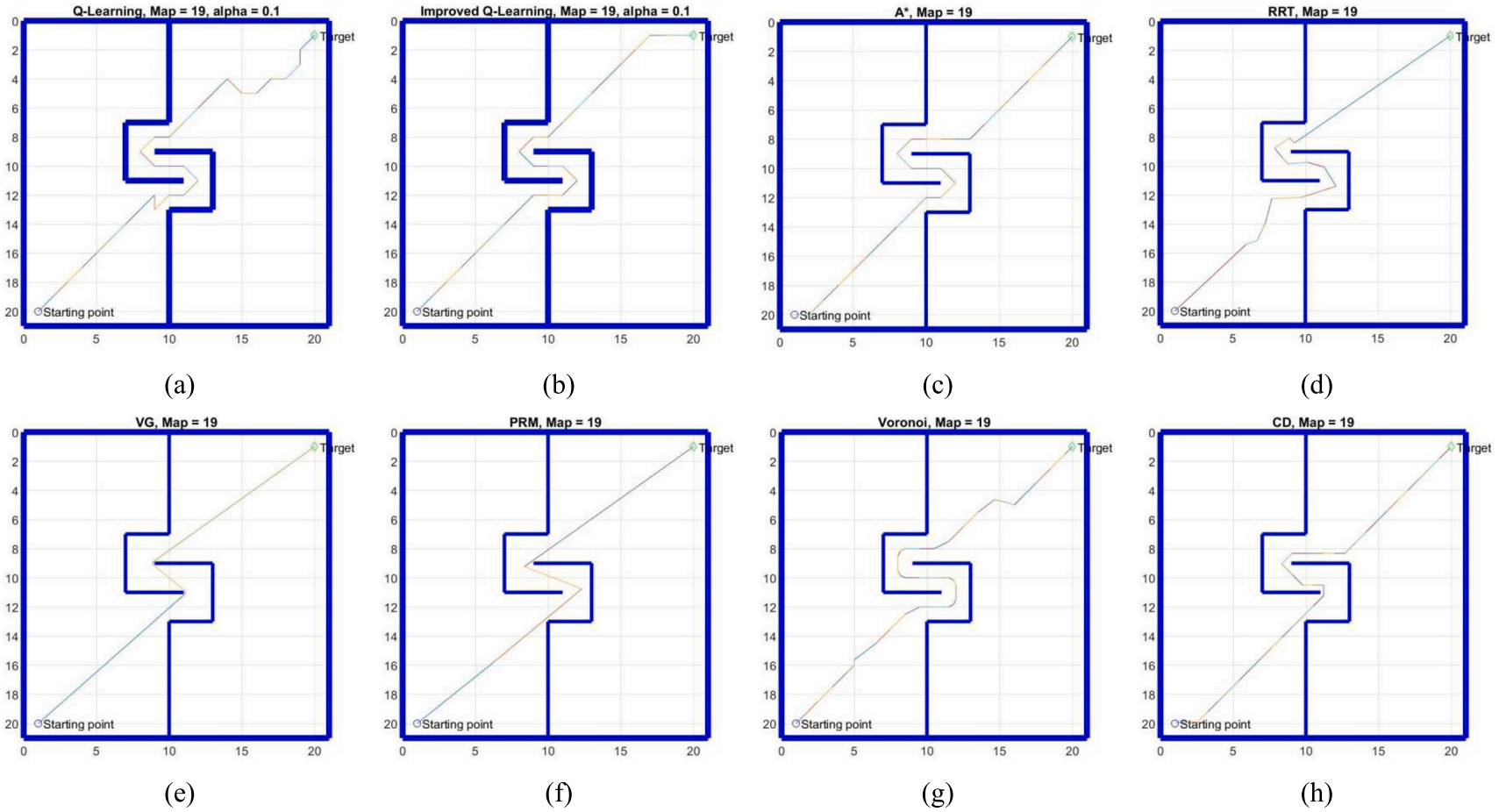
A-15. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 16



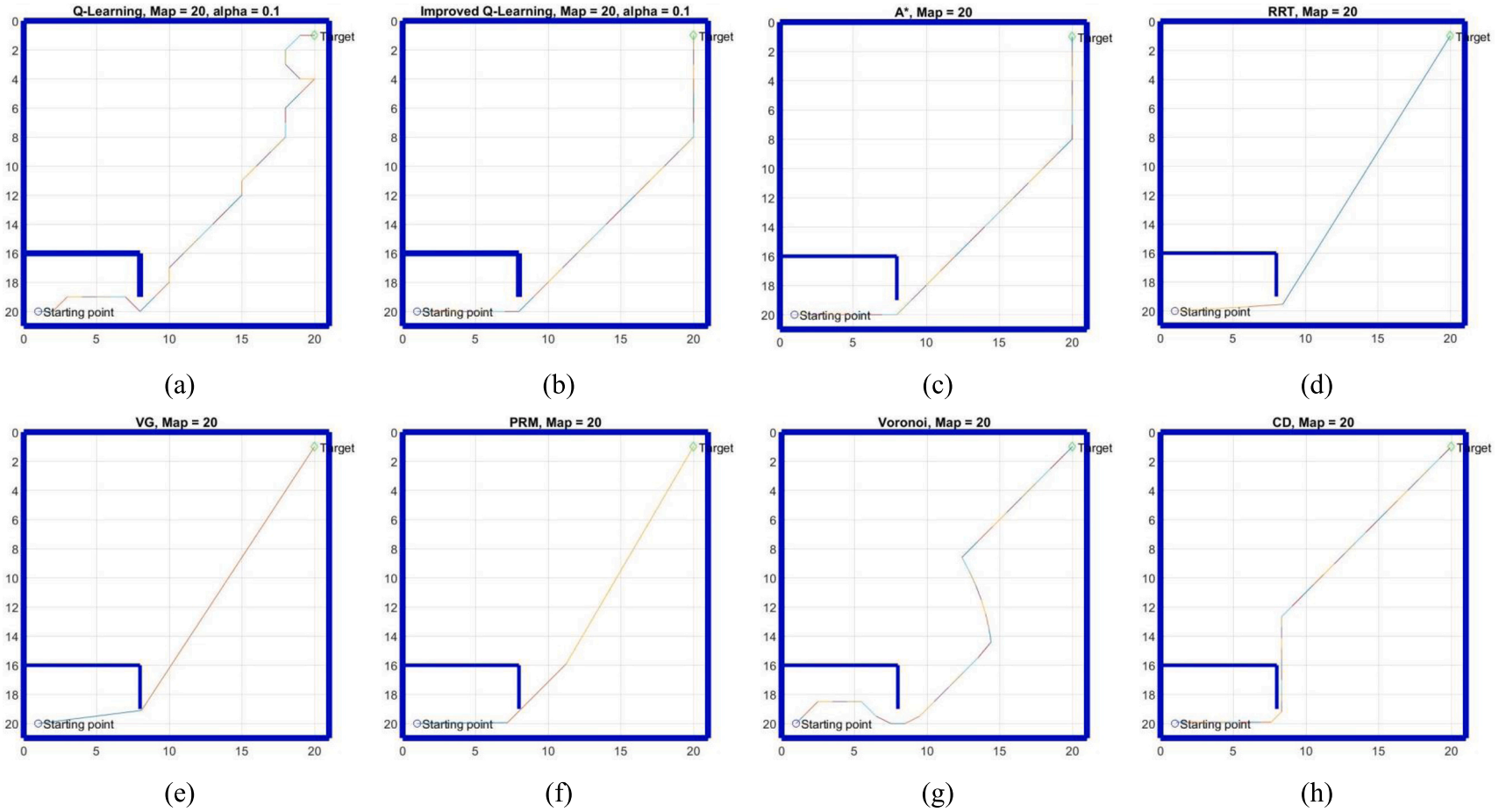
A-16. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 17



A-17. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 18



A-18. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 19



A-19. The generated shortest path using (a) QL; (b) IQL; (c) A*; (d) RRT; (e) VG; (f) PRM; (g) VD; and (h) CD for Map 20

References

Aouf, A., Boussaid, L., & Sakly, A. (2019). Same fuzzy logic controller for two-wheeled mobile robot navigation in strange environments. *Journal of Robotics, 2019*.

Aurenhammer, F. (1991). Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR), 23*(3), 345–405.

Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, 679–684.

Bresson, G., Alsayed, Z., Yu, L., & Glaser, S. (2017). Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles, 2*(3), 194–220.

Brooks, R. A., & Lozano-Perez, T. (1985). A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics(2)*, 224–233.

Candeloro, M., Lekkas, A. M., & Sørensen, A. J. (2017). A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels. *Control Engineering Practice, 61*, 41–54.

Carlucho, I., De Paula, M., & Acosta, G. G. (2019). Double Q-PID algorithm for mobile robot control. *Expert Systems with Applications, 137*, 292–307.

Chen, C., Chen, X.-Q., Ma, F., Zeng, X.-J., & Wang, J. (2019). A knowledge-free path planning approach for smart ships based on reinforcement learning. *Ocean Engineering, 189*, Article 106299.

Chen, H., Ji, Y., & Niu, L. (2020). Reinforcement learning path planning algorithm based on obstacle area expansion strategy. *Intelligent Service Robotics*, 1–9.

Chiang, H. L., Hsu, J., Fiser, M., Tapia, L., & Faust, A. (2019). RL-RRT: Kinodynamic motion planning via learning reachability estimators from RL policies. *IEEE Robotics and Automation Letters, 4*(4), 4298–4305. https://doi.org/10.1109/LRA.2019.2931199

Contreras-Cruz, M. A., Ayala-Ramirez, V., & Hernandez-Belmonte, U. H. (2015). Mobile robot path planning using artificial bee colony and evolutionary programming. *Applied Soft Computing, 30*, 319–328.

Cruz, D. L., & Yu, W. (2017). Path planning of multi-agent systems in unknown environment with neural kernel smoothing and reinforcement learning. *Neurocomputing, 233*, 34–42.

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, 19*(1), 651–669.

Donatelli, M., Giannelli, C., Mugnaini, D., & Sestini, A. (2017). Curvature continuous path planning and path finding based on PH splines with tension. *Computer-Aided Design, 88*, 14–30.

Duguleana, M., & Mogan, G. (2016). Neural networks based reinforcement learning for mobile robots obstacle avoidance. *Expert Systems with Applications, 62*, 104–115.

Feng, C., Sun, M., & Zhang, J. (2019). Reinforced deterministic and probabilistic load forecasting via Q-learning dynamic model selection. *IEEE Transactions on Smart Grid*.

Feng, Y., Yu, W., Chen, Y., Tan, X., Wang, R., & Madani, K. (2015). Option-based motion planning and ANFIS-based tracking control for wheeled robot in cluttered environment. *Paper presented at the 2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*.

Guo, M., Liu, Y., & Malec, J. (2004). A new Q-learning algorithm based on the metropolis criterion. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 34*(5), 2140–2143.

Hafez, M. B., & Loo, C. K. (2015). Topological Q-learning with internally guided exploration for mobile robot navigation. *Neural Computing and Applications, 26*(8), 1939–1954.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics, 4*(2), 100–107.

Hu, Y., Li, D., He, Y., & Han, J. (2019). Incremental learning framework for autonomous robots based on Q-learning and the adaptive kernel linear model. *IEEE Transactions on Cognitive and Developmental Systems*.

Hwang, K.-S., Jiang, W.-C., & Chen, Y.-J. (2016). Pheromone-based planning strategies in Dyna-Q learning. *IEEE Transactions on Industrial Informatics, 13*(2), 424–435.

Jiang, J., & Xin, J. (2019). Path planning of a mobile robot in a free-space environment using Q-learning. *Progress in Artificial Intelligence, 8*(1), 133–142.

Jiang, L., Huang, H., & Ding, Z. (2019). Path planning for intelligent robots based on deep Q-learning with experience replay and heuristic knowledge. *IEEE/CAA Journal of Automatica Sinica*.

Kang, N. K., Son, H. J., & Lee, S.-H. (2018). Modified A-star algorithm for modular plant land transportation. *Journal of Mechanical Science and Technology, 32*(12), 5563–5571.

Kang, R., Zhang, T., Tang, H., & Zhao, W. (2016). Adaptive Region Boosting method with biased entropy for path planning in changing environment. *CAAI Transactions on Intelligence Technology, 1*(2), 179–188.

Kavraki, L. E., Svestka, P., Latombe, J.-C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on robotics and automation, 12*(4), 566–580.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. *Paper presented at the Proceedings. 1985 IEEE International Conference on Robotics and Automation*.

Klidbary, S. H., Shouraki, S. B., & Kourabbaslou, S. S. (2017). Path planning of modular robots on various terrains using Q-learning versus optimization algorithms. *Intelligent Service Robotics, 10*(2), 121–136.

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.

Lee, C. W., Wong, W. P., Ignatius, J., Rahman, A., & Tseng, M.-L. (2020). Winner determination problem in multiple automated guided vehicle considering cost and flexibility. *Computers & Industrial Engineering, 142*, Article 106337. https://doi.org/10.1016/j.cie.2020.106337

Lozano-Pérez, T., & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM, 22*(10), 560–570.

Mac, T. T., Copot, C., Tran, D. T., & De Keyser, R. (2016). Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems, 86*, 13–28.

Mac, T. T., Copot, C., Tran, D. T., & De Keyser, R. (2017). A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization. *Applied Soft Computing, 59*, 68–76.

Martins, O., Adekunle, A., Adejuyigbe, S., Adeyemi, O., & Arowolo, M. (2020). Wheeled Mobile Robot Path Planning and Path Tracking Controller Algorithms: A Review. *Journal of Engineering Science & Technology Review, 13*(3).

Montiel, O., Orozco-Rosas, U., & Sepúlveda, R. (2015). Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. *Expert Systems with Applications, 42*(12), 5177–5191.

Muthugala, M. V. J., Vengadesh, A., Wu, X., Elara, M. R., Iwase, M., Sun, L., & Hao, J. (2020). Expressing attention requirement of a floor cleaning robot through interactive lights. *Automation in Construction, 110*, Article 103015.

Omar, R.b. (2012). *Path planning for unmanned aerial vehicles using visibility line-based methods*. University of Leicester.

Qureshi, A. H., & Ayaz, Y. (2016). Potential functions based sampling heuristic for optimal path planning. *Autonomous Robots, 40*(6), 1079–1093.

Rakshit, P., Konar, A., Bhowmik, P., Goswami, I., Das, S., Jain, L. C., & Nagar, A. K. (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, 43*(4), 814–831.

Sah, A. K., Mohanty, P. K., Kumar, V., & Chhotray, A. (2019). Log-Based Reward Field Function for Deep-Q-Learning for Online Mobile Robot Navigation. In *Computational Intelligence in Data Mining* (pp. 237-248): Springer.

Sato, M., Abe, K., & Takeda, H. (1988). Learning control of finite markov chains with an explicit trade-off between estimation and control. *IEEE Transactions on Systems, Man, and Cybernetics, 18*(5), 677–684.

Shen, H., Hashimoto, H., Matsuda, A., Taniguchi, Y., Terada, D., & Guo, C. (2019). Automatic collision avoidance of multiple ships based on deep Q-learning. *Applied Ocean Research, 86*, 268–288.

Song, Y., Li, Y.-B., Li, C.-H., & Zhang, G.-F. (2012). An efficient initialization approach of Q-learning for mobile robots. *International Journal of Control, Automation and Systems, 10*(1), 166–172.

Sturtevant, N. R. (2012). Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games, 4*(2), 144–148.

Su, B., Shi, Y., Li, X., Gong, Y., Li, H., Ren, Z., … Yao, W. (2021). Autonomous Robot for Removing Superficial Traumatic Blood. *IEEE Journal of Translational Engineering in Health and Medicine*.

Tsardoulias, E., Iliakopoulou, A., Kargakos, A., & Petrou, L. (2016). A review of global path planning methods for occupancy grid maps regardless of obstacle density. *Journal of Intelligent & Robotic Systems, 84*(1–4), 829–858.

Vanhulsel, M., Janssens, D., Wets, G., & Vanhoof, K. (2009). Simulation of sequential data: An enhanced reinforcement learning approach. *Expert systems with applications, 36*(4), 8032–8039. https://doi.org/10.1016/j.eswa.2008.10.056

Wang, W., Deng, H., & Wu, X. (2018). Path planning of loaded pin-jointed bar mechanisms using Rapidly-exploring Random Tree method. *Computers & Structures, 209*, 65–73.

Watkins, C. J. C. H. (1989). Learning from delayed rewards.

Wei, Z., Liu, F., Zhang, Y., Xu, J., Ji, J., & Lyu, Z. (2019). A Q-learning algorithm for task scheduling based on improved SVM in wireless sensor networks. *Computer Networks, 161*, 138–149.

Wen, S., Zhao, Y., Yuan, X., Wang, Z., Zhang, D., & Manfredi, L. (2020). Path planning for active SLAM based on deep reinforcement learning under unknown environments. *Intelligent Service Robotics*, 1–10.

Yan, C., Xiang, X., & Wang, C. (2019). Towards Real-Time Path Planning through Deep Reinforcement Learning for a UAV in Dynamic Environments. *Journal of Intelligent & Robotic Systems*, 1–13.

Yang, L., Nagy, Z., Goffin, P., & Schlueter, A. (2015). Reinforcement learning for optimal control of low exergy buildings. *Applied Energy, 156*, 577–586.

Yang, L., Sun, Q., Ma, D., & Wei, Q. (2019). Nash Q-learning based equilibrium transfer for integrated energy management game with We-Energy. *Neurocomputing*.

Yijing, Z., Zheng, Z., Xiaoyi, Z., & Yang, L. (2017). *Q learning algorithm based UAV path learning and obstacle avoidence approach.* Paper presented at the Control Conference (CCC), 2017 36th Chinese.

Zhang, B., Mao, Z., Liu, W., & Liu, J. (2015). Geometric reinforcement learning for path planning of UAVs. *Journal of Intelligent & Robotic Systems, 77*(2), 391–409.

Zhao, X., Ding, S., An, Y., & Jia, W. (2018). Asynchronous reinforcement learning algorithms for solving discrete space path planning problems. *Applied Intelligence, 48*(12), 4889–4904.