A Novel Path Planning Algorithm Based on Q-learning and Adaptive Exploration Strategy

Ting Li, Ying Li
College of Computer Science and Technology, Jilin University, Changchun, 130012, China
E-mail: lt18458398129@163.com

***Abstract*—In an unknown environment, how to plan a path is a fundamental problem for agents. In this paper, we propose an improved reinforcement learning algorithm, called adaptive exploration Q-learning algorithm (AEQ), to solve path planning problem. Firstly, to ensure an agent learns autonomously in the process of trial and error when the agent knows nothing about the environment, AEQ chooses Q-learning algorithms to improve. Secondly, AEQ utilizes an adaptive exploration strategy that aims at speeding up the algorithm's convergence. The adaptive exploration strategy dynamically adjusts the exploration factor according to various situations so that the agent explores the environment sufficiently and makes full use of the environment information. The experimental results show that the agent successfully reaches the goal without collision by AEQ. Besides, compared with the classical Q-learning algorithm and SARSA algorithm, AEQ improves the convergence speed and reduces the convergence time.**

Keywords—Reinforcement learning, Path planning, Obstacle avoidance, Q-learning

I. INTRODUCTION

Path planning is fundamental and significant which is widely used in navigation and express industry. Given an initial position and a target position, path planning seeks out a path that avoids obstacles and reaches destination.

To solve path planning problems, many classical methods are put forward to, such as A* algorithm [1], Probabilistic Roadmap Method [2] and cell decomposition method [3]. Moreover, with the development of artificial intelligence, a lot of intelligent bionic algorithms are applied to path planning problems. For example, genetic algorithm [4] and gray wolf optimization algorithm [3].

Reinforcement learning is one of the artificial intelligence algorithms [5]. And Q-learning algorithm is a reinforcement learning algorithm which enables an agent to learn like human beings. In unknown environment, an agent gradually acquires environmental knowledge by constantly interacting with the environment. In this process, we reward an agent if the agent does what we expect and punish the agent if the agent does what we don't allow. After trial and error, the agent gets enough knowledge of the mapping relationship from states to actions. Then the agent searches for the optimal path by means of maximizing the long-term cumulative reward. But Q-learning algorithm has a limitation, that is, the convergence speed is slow. To speed up the convergence speed, Xingyu Zhao et al combined asynchronous method and Q-learning algorithm so as to efficiently solve path planning problem in discrete space [6]. Ee Soong Low et al used the flower pollination algorithm to improve Q-learning algorithm's computational effectiveness [7].

There is a basic problem called exploration and exploitation dilemma in reinforcement learning problem [8]. Exploration means collecting environment information and exploitation means making the best decision based on the information collected. However, more exploration leads that an agent might concentrate on exploring the new environment and forget to look for the shorter path. More exploitation causes that an agent might find sub-optimal route. Usually, we use epsilon-greedy algorithm to solve this dilemma [9].

In order to solve above problems, we propose a new algorithm called AEQ. AEQ is based on Q-learning algorithm and improves an adaptive exploration strategy. The adaptive exploration strategy divides the whole exploration process into three stages. In the first stage, to familiarize the agent with the environment as soon as possible, the probability of exploration is far greater than the probability of exploitation. In the second stage, the agent knows part of the environment information, so it is in the state of exploring while exploiting. In the third stage, the probability of exploration reduces drastically so as to converge the algorithm quickly. As a result, AEQ seeks out a collision-free path quickly and accelerates convergence velocity.

The rest of the paper is organized as follows. In Section II, the basic Q-learning algorithm is introduced. Section III describes the details of AEQ. Section IV gives the experimental results. Finally, Section V summarizes the paper and discusses future research directions.

II. Q-LEARNING ALGORITHM

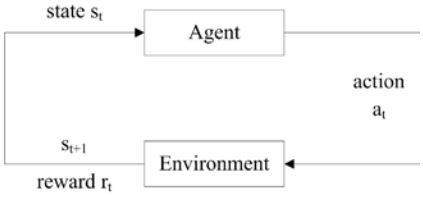

Figure 1. Reinforcement learning framework

Reinforcement learning is an algorithm that teaches an agent how to accomplish a task step by step [10]. As shown in Fig. 1, it describes the reinforcement learning framework. At iteration t, s_t is current state of an agent and a_t is action which the agent chooses and executes among all executable actions. After interacting with the environment,

the agent obtains the next state s_{t+1} and the reward r_t. Then the agent selects the next action until it achieves the destination.

Q-learning algorithm is a classical reinforcement learning algorithm. It makes use of a Markov decision process (MDP) to describe path planning problem [11]. MDP is a five tuple (S,A,P,R,γ), where S and A are a finite set of states and actions, P is a state transition probability function from current state and action to next state, R is a reward function and $\gamma \in [0,1]$ is a discount factor.

After modeling, an agent begins to explore the environment. At each step, the agent selects an action from the set A through policy $\pi = S \to A$. The policy π is a map from states to actions. Mostly, Q-learning algorithm selects the epsilon-greedy algorithm to determine what the next action is. The specific method of the epsilon-greedy algorithm is to select randomly an action or choose the best action. Afterwards another state and a new reward appear.

During the motion process, the agent learns experience constantly [12]. The learning way is expressed as follows:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_t + \gamma \cdot \max_{a_{t+1}} Q(s_{t+1},A) - Q(s_t,a_t)\right] \quad (1)$$

In (1), $Q(s_t,a_t)$ evaluates the value of the state-action pair and it is also a foundation to make the best choice in all actions. r_t is a reward calculated by the reward function R. $\alpha \in [0,1]$ is the learning rate. If α is equal to 1, it means that the impact of future motions is not taken into account. If α is equal to 0, it means that the current Q-value lacks reference and the update of the Q-value basically depends on the present reward and the selection of the subsequent state-action pairs. The discount factor γ measures the importance of the next Q-value towards the current Q-value. If γ is close to 1, the total reward consists of the present reward and the next maximum Q-value. If γ is close to 0, the total reward only includes the present reward.

After the agent learns enough experience, it exploits them. When the agent is in the exploitation stage, the next action is decided using the following formula:

$$a_t^*(s_t) = \max_{a_{t+1}} Q(s_{t+1},A) \quad (2)$$

In the process of iteration, Q-value converges continuously. As a result, a qualified optimal path is generated.

III. Adaptive Exploration Q-learning Algorithm

With regard to the Q-learning algorithm, its disadvantage is slow convergence of Q-table. Furthermore, the convergence of the algorithm depends on the changing process of the exploration factor. For the sake of improving this restriction, we put forward AEQ algorithm which applies adaptive exploration strategy to dynamically update the exploration factor.

A. Adaptive Exploration Strategy

In reinforcement learning problems, a strategy that determines what an agent does next is basic and important. Among strategies, epsilon-greedy algorithm is widely used to solve this problem. The basic idea of epsilon-greedy algorithm is that the probability of selecting an action by maximizing Q-value is ε and the probability of selecting an action randomly is 1-ε. ε is the exploration factor. When ε is close to 0, it means an agent explores the environment. And when ε is close to 1, it means an agent exploits the environment knowledge. Moreover, ε varies linearly from 0 to 1. Thus we propose an adaptive exploration strategy in order to change ε dynamically according to the actual situation.

We divide the process which an agent explores and exploits the environment into three stages.

In phase one, there is an agent knows nothing about the surroundings. Therefore, this stage is a exploration stage and the agent explores the environment as much as possible. For this purpose, ε changes more slowly than the epsilon-greedy algorithm. The calculation formula is written as follows:

$$\varepsilon = step1 * (i/T1)^2 \quad (3)$$

where $step1 \in (0,0.5)$ is the step size of the first stage, i is the current iteration number, and $T1$ is the maximum number of iterations in this stage.

In phase two, the agent explores the environment while it exploits the environment. We define a parameter *sRate* to indicate the probability that the agent arriving at the destination every ten iterations. If *sRate* is lower than the expectation *rate*, ε alters slower than the phase one. If *sRate* is higher than *rate*, the change of ε accelerates. In addition, *rate* is close to 1 which implies when the agent gets enough solutions of path planning problem, it can go to the exploitation stage. And *rate* is close to 0 which implies even if the agent obtains a few solutions, it can quickly move to the exploitation stage. ε is updated as:

$$\varepsilon = \begin{cases} step1 + step2 * sin\left(\dfrac{i-T1}{T2} * \dfrac{\pi}{2}\right) & \text{if } sRate > rate \\ step1 + step2 * sin\left(\dfrac{i-T1}{T2} * \dfrac{\pi}{2}\right) * sRate & \text{otherwise} \end{cases} \quad (4)$$

where $step2 \in (0,1)$ is the step size of the second stage, $T2$ is the maximum number of iterations in this stage and $rate \in [0,1]$ is given by testing.

In phase three, the agent mainly exploits the environment information. More importantly, the agent obtains an optimal solution based on the information it explores. The updating equation of ε is described as:

$$\varepsilon = step2 + step3 * \left((i - T2)/T3\right)^2 \quad (5)$$

where $step3 \in (0.5,1)$ is the step size of the third stage, $step1 + step2 + step3 = 1$ and $T3$ is the maximum number of iterations in this stage.

In this stage, the agent is updated like phase one. Because the agent doesn't know the size of the environment map. If the map is large and the distance between the start point and the end point is very close, the agent is unnecessary to known the whole map. However, we don't want that the agent misses the optimal path. As a result, the agent tries its best to explore the environment in the phase three.

B. The reward function

The reward function quantifies the decision-making process, that is, it estimates the reward after carrying out an action at a given state. The set of states and actions is separated into five situations. And the reward function is expressed as:

$$r_t = \begin{cases} 1 & s_t = s_{end} \\ -1 & d_{obs} = 0 \\ 0.2 & d_t < d_{t-1}, d_{obs} \neq 0 \\ -0.2 & d_t > d_{t-1}, d_{obs} \neq 0 \\ -0.1 & \text{otherwise} \end{cases} \quad (6)$$

where t is current iteration step, r_t is the reward value, s_t is current state and s_{end} is the target position. Additionally, d_{obs} is the distance between the agent and the nearest obstacles and d_t is the distance between the agent and the target position.

C. Adaptive exploration Q-learning algorithm

AEQ algorithm's pseudo code is shown in Algorithm 1. And AEQ algorithm works in accordance with the flow chart presented in Fig. 2.

First, Q-value is initialized. The start point and the end point are also specified.

Second, during every iteration, an agent starts from the start point and moves towards the end point. In each step of the iteration, the agent selects an action by using the epsilon-greedy algorithm. After performing the action, the environment feeds back a reward on the basis of (6).

Third, Q-value is updated according to (1) and ε is changed according to adaptive exploration strategy.

Finally, if the agent collides or arrives at the end point, a new iteration starts until the number of iteration is the maximum.

Algorithm 1 Adaptive exploration Q-learning algorithm

Initialize $Q(S,A)$, s_{start} and s_{end}

Initialize iteration counter $T = 0$ and step counter $t = 0$

Repeat

  Initialize $s_t = s_{start}$

  Repeat

    Select an action a_t randomly with probability ε

    Otherwise, $a_t \leftarrow max_{a_t} Q(s_t, A)$

    Execute a_t

    Obtain the reward and the next state s_{t+1}

    Update Q-value according to (1)

    $s_t \leftarrow s_{t+1}$, $t \leftarrow t+1$

  Until $s_t = s_{end}$

  Update ε according to adaptive exploration strategy

  $T \leftarrow T+1$

Until $T > T_{max}$

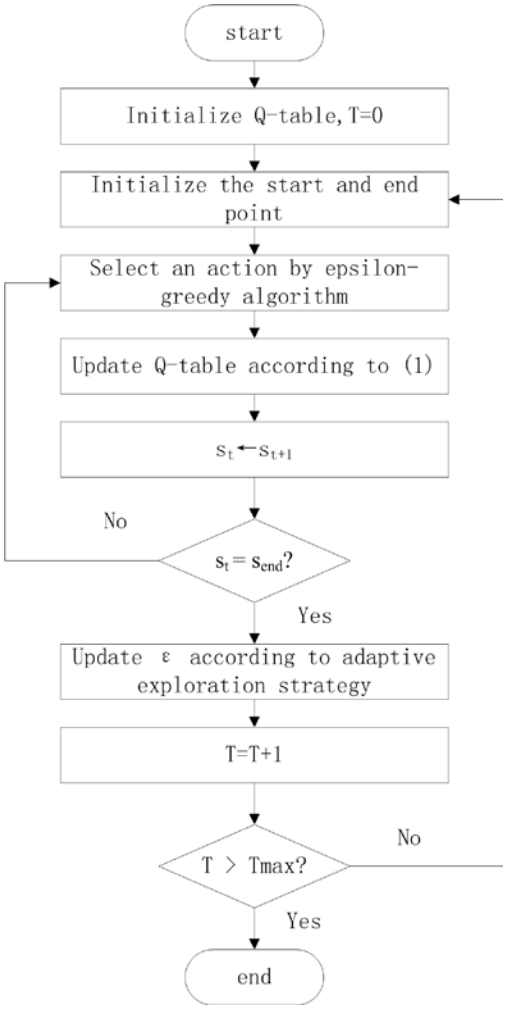

Figure 2. Flow chart of AEQ

IV. EXPERIMENTS

The performance of the algorithm is closely related to the experimental environment. In this case, a series of tests were run on Windows 7 system with Python 3.6.0 installed. The CPU is Intel i5-4200U with 4 cores and the RAM is 4GB.

In the experiments, we used the following constants: $\gamma = 0.9$, $T_{max} = 300$, $T_1 = T_2 = T_3 = 100$, $step1 = 0.4$, $step2 = 0.2$, $step3 = 0.4$, $rate = 0.4$, the action space $A = [left, right, up, down]$ and the learning rate α was decreased linearly with time.

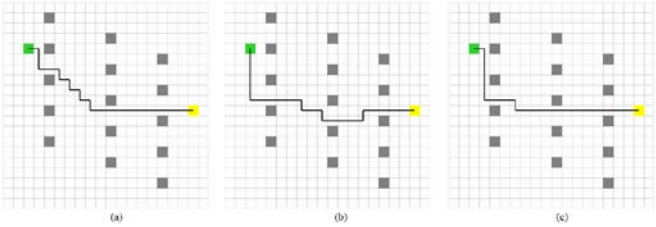

Figure 3. The optimal path obtained by (a)Q-learning, (b)SARSA and (c)AEQ in experimental Scene 1

The map of scene 1 shown in Fig. 3 is used to compare the performance of Q-learning, SARSA and the proposed

AEQ in path planning problem. In this scene, the green square in the upper left corner is the start point and the right-most yellow square is the end point. There are 15 gray square obstacles. From the figure, it's obviously that AEQ and Q-learning plan the shortest route. The total number of steps is 22 units. SARSA spends 24 units to reach the target position. All three algorithms successfully complete the goal.

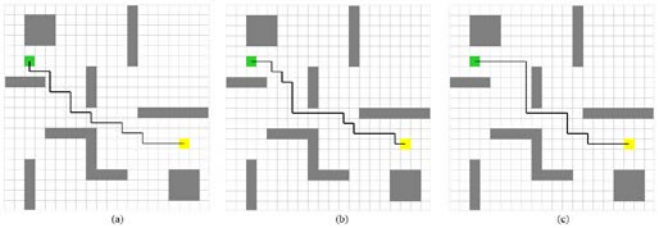

Figure 4. The optimal path obtained by (a)Q-learning, (b)SARSA and (c)AEQ in experimental Scene 2

The map shown in Fig. 4 is scene 2. In this scene, the green square in the upper left corner is the start point and Yellow Square in the lower right corner is the end point. There are 8 big gray obstacles made up of small squares. The steps for AEQ, Q-learning and SARSA to find the goal is 23. But AEQ has fewer inflection points than Q-learning and SARSA. It shows that the optimal route of AEQ is better than Q-learning and SARSA.

TABLE 1 THE COMPARISON OF THREE ALGORITHMS

Experimental Scene	Average Computing Time (s)		
	Q-learning	SARSA	AEQ
Scene 1	13.98	13.86	13.08
Scene 2	14.20	13.72	12.53

Table 1 contrasts the average computing time between Q-learning, SARSA and AEQ. In scene 1, AEQ costs 13.08s to reach the goal on average. And the average time AEQ takes to arrive at the target is 12.53s in scene 2. In terms of average computing time, AEQ takes less time than Q-learning and SARSA. Especially in Scene 2, AEQ reduces 11.76% and 8.67% than Q-learning and SARSA. It proves that the proposed algorithm AEQ accelerates the convergence speed and shortens the convergence time compared with Q-learning and SARSA.

V. CONCLUSION

In this study, AEQ algorithm has been proposed for path planning problem in unknown environment. We used the adaptive exploration strategy to adjust timely the exploration factor in the face of different situations. The experimental results prove that AEQ algorithm speeds up the convergence and improves the convergence efficiency. However, we only consider static obstacles and don't take into account dynamic obstacles that exist in the real environment. In the future, we will study how to use reinforcement learning to complete the path planning in a dynamic environment.

REFERENCES

[1] J.K. Goyal and K.S. Nagla, "A new approach of path planning for mobile robots," International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014, pp. 863-867.

[2] A. Short, Z.X. Pan, N. Larkin and S.V. Duin, "Recent Progress on Sampling Based Dynamic Motion Planning Algorithms," IEEE International Conference on Advanced Intelligent Mechatronics, 2016, pp. 1305-1311.

[3] M.N. Zafar and J.C. Mohanta, "Methodology for path planning and optimization of mobile robots: a review," Procedia Computer Science, vol. 133, 2018, pp. 141-152.

[4] M. Elhoseny, A. Tharwat and A.E. Hassanien, "Bezier curve based path planning in a dynamic field using modified genetic algorithm," Journal of Computational Science, vol. 25, 2018, pp. 339-350.

[5] R.S. Sutton and A.G. Barto, "Reinforcement learning: an introduction," MIT Press, 1998.

[6] X.Y. Zhao, S.F. Ding, Y.X. An and W.K. Jia, "Asynchronous reinforcement learning algorithms for solving discretespace path planning problems," Applied Intelligence, vol. 48(12), 2018, pp. 4889-4904.

[7] E.S. Low, P. Ong and K.C. Cheah, "Solving the optimal path planning of a mobile robot using improved Q-learning," Robotics and Autonomous Systems, vol. 115, 2019, pp. 143-161.

[8] G. Reddy, A. Celani and M. Vergassola, "Infomax strategies for an optimal balance between exploration and exploitation," Journal of Statistical Physics, vol. 163(6), 2016, pp. 1454-1476.

[9] M. Castronovo, F. Maes and R. Fonteneau, "Learning exploration/exploitation strategies for single trajectory reinforcement learning," European Workshop on Reinforcement Learning, vol. 24, 2012, pp. 1-9.

[10] J. Kober, J.A. Bagnell and J. Peters, "Reinforcement learning in robotics: a survey," The International Journal of Robotics Research, vol. 32(11), 2013, pp. 1238-1274.

[11] Z.C. He and W. Jiang, "An evidential markov decision making model," Information Sciences, vol. 467, 2018, pp. 357-372.

[12] M. Duguleana and G. Mogan, "Neural networks based reinforcement learning for mobile robots obstacle avoidance," Expert Systems With Applications, vol. 62, 2016, pp. 104-115.