ISBN 978-981-11-7861-0
Proceedings of 2018 the 8th International Workshop on Computer Science and Engineering (WCSE 2018)
Bangkok, 28-30 June, 2018, pp. 276-281
doi: 10.18178/wcse.2018.06.049

A New and Effective Reinforcement Learning based on Tentative Q learning and Knowledge Transfer

Duan Jun-Hua [+], Zhu Yi-An, Zhong Dong, Zhan Tao, and Luo Shuyan

School of Computer, Northwestern Polytechnical University, Xi'an 710072, PR China

Abstract. Aiming at the problem of slow learning speed of reinforcement learning, TQL-RWKT-RRL algorithm is put forward, which is based on tentative Q learning and knowledge transfer. Tentative Q learning increases times of exploration in each iteration, and improves updating method of Q value function. Knowledge transfer algorithm realizes knowledge transfer under different state space based on the method of rolling windows. The path planning experiences in the simple small environment is transferred to more complex and larger state space, which speeds up robot path planning learning speed in large and more complex environment.

Keywords: reinforcement learning, knowledge transfer, rolling windows, robot path-planning.

1. Introduction

Reinforcement learning(RL) is the main learning method for robot learning [1]. RL uses reward or punishment signal that comes from the experience of robot to improve robot performance. The slow speed of convergence is a main difficult problem in RL algorithm [2].

There are two key factors affecting the speed of RL convergence [3].The first factor is the full and fast exploration of state space. If exploration strategy is fully random, the exploration will be blind and the learning process cannot converge. If action selection always takes greedy strategy, it is easy to fall into local optimal solution or "local trap"(Local trap refers to the phenomenon that robots repeated backtracking through some nodes). Two classical method to balance exploration and exploitation is $\varepsilon - greedy$ and Boltzmann exploration mechanism. $\varepsilon - greedy$ strategy follows the greedy strategy with probability ε, and acts randomly with probability (l-ε). If ε is too large, there will be more actions to take greedy strategy; and if ε is too small, random actions will be increased. Boltzmann strategy chooses different selection pressure according to different stages of learning, so it do not fall into the "local trap", but it causes slow convergence speed. The second factor for slow convergence speed of RL is the lacking of prior knowledge of the environment. In the early stages of learning, robots have to spend more time exploring completely unknown environments.

A lot of research work has been carried out to improve the performance of reinforcement learning. Morozs [4] takes heuristic selection strategy and reward function to speed up the convergence speed. But if the heuristic function is constructed improperly, it will make learning procedure trap to unfavourable direction. Hierarchical reinforcement learning[5] approach accelerates the learning speed through breaking the tasks into subtasks on different levels, then solving sub-tasks on a smaller state space, so subtasks' strategies obtained can be reused by multiple upper tasks. However, when learning in more complex environment or unknown environments, task hierarchy is difficult to determine[6]. The strategy that transfer knowledge between tasks can yield a remarkable speedup for learning process in the robot navigation task[7]. Bianchi[8] uses the previous knowledge as prior knowledge while learning new task. But it requires the learned task and

[+] Corresponding author. Tel.: +13991293414; fax: +029-88431517.
E-mail address: duanjh@nwpu.edu.cn.

new task must have a greater similarity. Some researchers suggest making robot learn behaviour in an ideal simulation environment, and then migrate the experiences to real-world environment. But to create an efficient and accurate model of the robot and the environment is also a challenging problem.

Aiming at the problem of slow learning speed of RL, a new robot reinforcement learning algorithm (TQL- RWKT-RRL) is put forward, which is based on tentative Q learning (TQL) and Rolling Window based Knowledge Transfer (RWTK). TQL algorithm balances the use of exploration and exploitation, which takes fully detection of successor states of each state and accelerates the learning speed. Rolling Window based Knowledge Transfer (RWTK) algorithm applies the path planning experience in the simple small environment to large state space in more complex environment, which speeds up robot learning speed in large and more complex environment.

2. Tentative Q Learning

Q learning due to its needless to establish environmental model and ensure convergence under certain conditions, become one of the most widely used reinforcement learning methods. The Q-learning formula is:

$$Q_t(s_t,a_t)=\begin{cases}(1-\alpha_t)Q_{t-1}(s_t,a_t)+\alpha_t[r_t+\gamma\max\limits_{a\in A}\{Q_{t-1}(s_{t+1},a)\}], & s=s_t,a=a_t\\ Q_{t-1}(s_t,a_t), & otherwise\end{cases} \tag{1}$$

In the formula, s_t is the current state, a_t is the selected action among all possible actions, s_{t+1} is the next state, r_t is the received immediate reward after a_t is executed, α_t is learning rate, and γ is discount factor.

In the traditional Q-learning, at each learning step, it will use a strategy to select an action among all the candidates, and update the selected candidate's Q value. However, the Q values for other investigated actions are not changed. So investigation processes for these actions are not recorded, and it is a waste.

The TQL algorithm will "test" all subsequent states in the current state. The so-called "test" is to refer to examine the reward values of all subsequent states, and to update the corresponding Q values. The subsequent state of the maximum Q value is then selected as the new state. The Q table update formula based on the "test" results is as follows:

$$Q_t(s_t,a_{ti})=\begin{cases}(1-\alpha_t)Q_{t-1}(s_t,a_{ti})+\alpha_t\left[r_{ti}+\gamma Q_{t-1}(s_t,a_{ti})\right] & s=s_t;a=a_{ti}\\ Q_{t-1}(s_t,a_t), & otherwise\end{cases} \tag{2}$$

In the formula, a_{ti} is the ith candidate actions, α_t is learning rate, γ is discount factor, r_{ti} is the reward on the state s_{ti}, $1\le i\le n$,n is the subsequent state number。

Based on formula (2), we choose the action with maximum Q value as the next action, and update the corresponding Q value again. The update formula is set as:

$$Q_t(s_t,a_{tk})=\begin{cases}(1-\alpha_t)Q_{t-1}(s_t,a_{tk})+\alpha_t\left[r_{tk}+\gamma V(s_t)\right] & s=s_t;a=a_{tk}\\ Q_{t-1}(s_t,a_t), & otherwise\end{cases} \tag{3}$$

$$V(s_t)=\max\left\{Q_{t-1}(s_{ti},a_{ti})\right\} \tag{4}$$

In the formula, $k=\arg\max(\{Q(s_{ti},a_{ti})\})$, it selects the tentative action with biggest Q value as candidate action.

The difference between the traditional Q-learning formula (1) and updating Q value in formula (2) is: tentative action only examines and updates its successor action Q value, rather than to perform these actions. After it selects the action with the maximum Q value, it updates the corresponding Q value again by formulas (3) and (4), which strengthen the choice for the second time.

3. Knowledge Transfer base on Rolling Windows

3.1. Transfer strategy

The main idea of RWTK is: when the robot is on the starting point, it will firstly determine the relative direction from current position to the destination position; and then use the rolling window method to determine the path to destination. RWTK transfers the experiences in 4×4 grid environment to complex

environments. When the robot steps out the window, the above process is repeated until it reaches the target point. After initial path is established by RWTK algorithm, the robot applies TQL algorithm to do reinforcement learning and explore the global optimal path.

In RWTK algorithm, obstacles will not be set in original simple environment, because the number, location, and other factors of obstacles cannot be known. In reality, environment is constantly changing, a simple representative and compatible environment is hardly to be found, let alone that transfer learning experiences of path finding in a simple environment to a complex environment. And the negative transfer can also be avoided. Simple environment designed here are 4×4 grid environment without any obstacle. Start and end points are in the corners of the grid respectively, which can be grouped into two categories as shown in Fig. 1. The first class is the start point and the end point on one edge and the second is the two points on the diagonal line in the grid, respectively.

Fig. 1: Two universal settings of starting and ending points.

Assume that r_b is the reward value of obstacles in the environment, r_b is used to adjust the Q value at the location of the obstacle during the transfer process. Assume that $Q_t(s_t, a_t)$ is the Q value of transfer point in simple environment. After transfer, the Q value is $Q_t^{'}(s_t, a_t)$ .Then the updating formula of Q value is shown in formula（5）.

$$Q_t^{'}(s_t, a_t) = \begin{cases} Q_t(s_t, a_t) + r_b, & \text{barrier exist} \\ Q_t(s_t, a_t), & otherwise \end{cases} \tag{5}$$

The implementation of transfer strategy is shown in Fig.2.

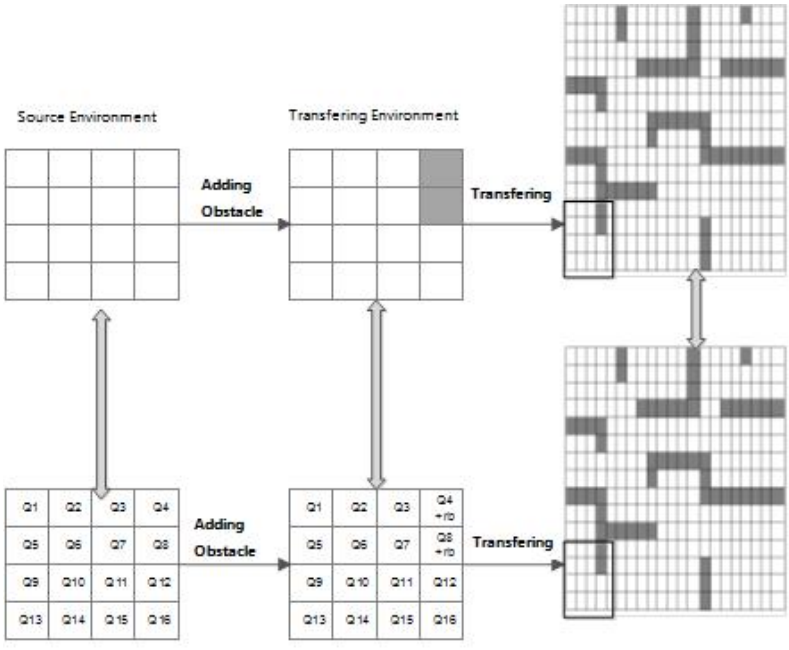

Fig. 2: The implementation of transfer strategy.

3.2. Set the rolling windows

The scale of rolling windows is set as 4×4 grid map. The robot learning experiences in a simple 4×4 grid environment is applied to unknown complex environment by the rolling windows. But in the real environment (as shown in Fig 3), robot has a total of eight target points of eight directions which can be used as the migration endpoints. Then the robot must determine which direction is the fastest direction. The method to choose the transfer endpoint is: to analyze the distance between the target point and each of the

eight guide points, and then select the end point with the minimum distance. As shown in Fig.3, position S is the robot position, and destination position is D. With S as the center point, eight guiding points can be found namely D1, D2 ... D8. Determine the distance between each guiding points and target point, and then select the guiding points with the minimum distance as the end point of migration. In the case shown in Fig.3, d8 is the minimum distance, so choose D8 as end point of migration.

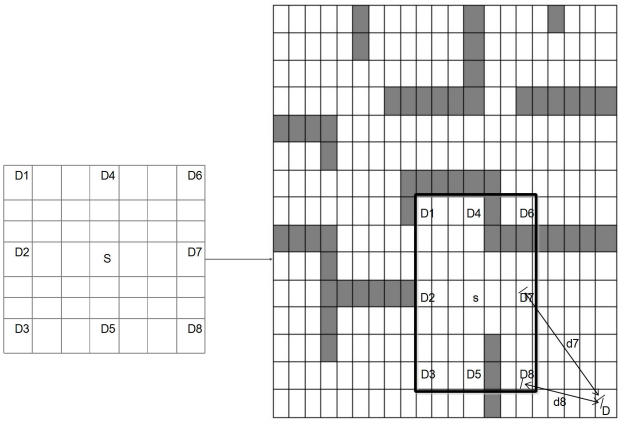

Fig. 3: Choose the guiding point of migration.

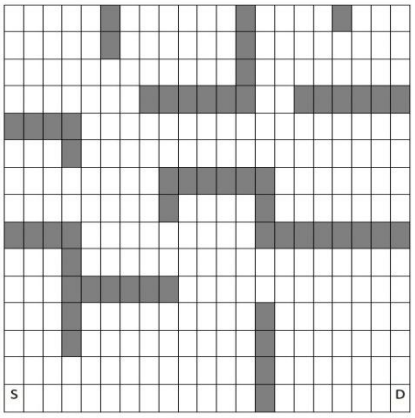

Fig. 4: Simulation experiments map.

4. TQL- RWKT-RRL Algorithm Design

TQL-RWKT-RRL algorithm does the online plan by the method of rolling window information. At first step of learning, knowledge transfer is carried out by RWKT method for the initial environment. And then, robot applies TQL learning algorithms to explore global optimal path.

The steps of TQL-RWKT-RRL algorithm are:

(1) Get the robot start point and target point, and obtain the Q value of small window to be transferred.

(2) Determine the guiding point, establish a rolling window. And transfer the Q value of small window to rolling window according to the formula (5).

(3) Determine whether the target point is in the rolling window. If it is in, conduct an exploration of the map using TQL until the target point is reached, then turn (7); Otherwise, turn (4).

(4) Determine whether the robot is in the rolling window, if is in, turn (6); Otherwise, turn (5);

(5) Determine the guiding point and establish a rolling window. Transfer the Q value of small window to rolling window according to the formula (5) and turn (6);

(6) Conduct an exploration of the map using TQL until the target point is reached using formula (2)-(4); Turn (3);

(7) According to the formula (2) - (4),using the TQL algorithm to do path planning study from the departure point to the target point until the end of study.

(8) The end.

5. Simulation Experiments and Analysis

The simulation experiment is carried out under Visual C++ 2010, and the environment of simulation experiment is as shown in Fig. 4. S represents the starting point of the robot, and D represents the target point. The robot can only move up, down, left, and right, and each time moves one grid. Each experiment starts when the robot is at the starting point and ends when the robot reaches the target point. At the beginning, robot has no experiences in an unfamiliar environment, so the initial values in Q are all 0. The reward of obstacles in environment is -300.The reward of edge of the map is -300.The reward of target point is 100, others are -1. In experiment, α is 0.5 and γ is 0.8.

Under the same circumstances with same starting and end points, robot respectively use Q-learning based Boltzmann selection strategy and ε-greedy selection strategy(the value of ε is set in stages) and TQL algorithm to conduct experiment 400 times. Path length of each experiment is recorded. The experimental

results are shown in Figure 5-8. In the charts, the ordinate represents length of path robot takes to get the target point. The horizontal axis represents the number of iterations.

When the robot using Boltzmann selection strategy, robot could eventually converge to a path, but speed of convergence is slow. When the distance between the state spaces is big or the environment has greater complexity, the speed of convergence is even slower. As shown in Fig. 5.

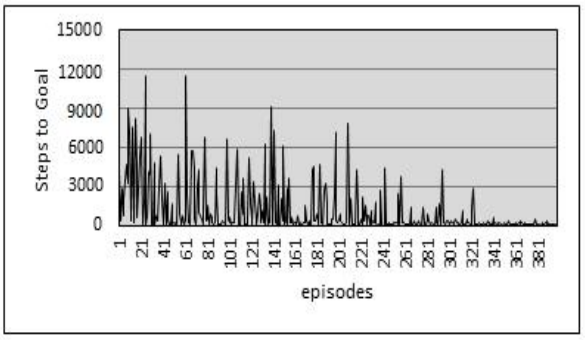

Fig. 5: Boltzmann selection strategy.

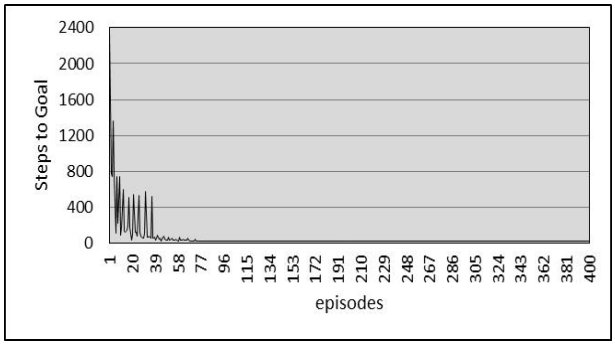

Fig. 6: Segmented ε-greedy strategy.

When robot uses segmented $\varepsilon - greedy$ selection strategy, in early times, robot learns with smaller value of Є and has little probability to select the action with the maximum Q value. In the late times, robot learns with bigger value of Є. The experimental results are shown in Fig. 6.This method is faster than Boltzmann selection strategy, but it sometimes falls into "local trap".

The learning process can quickly converge to an optimal path when the robot using TQL algorithm. Robot has been updated Q value of all directions and selected the direction corresponding to maximum Q value before it actually moved. TQL improves the efficiency of robot learning. The result is shown in Fig.7.

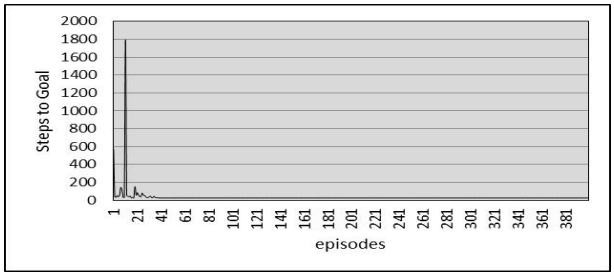

Fig. 7: TQL algorithm.

The solid line in Fig. 8 shows learning curve using knowledge transfer algorithm RWKT. As it is shown in Fig.8,the learning curve is converged after 12 times when using RWKT method, which means the optimal path is found; and which is after 33 times by TQL algorithm without using knowledge transfer. The 20 end points is randomly generated, and then respectively using TQL-RWKT-RRL algorithm and TQL algorithm without using knowledge transfer to find the optimal paths. The results of experiments are shown in Fig.9.As it is shown that in most cases of 20 learning experiments, the iteration numbers in TQL-RWKT-RRL algorithm are less than those in TQL algorithm. It illustrates that transfer learning can improve the learning speed of Q-learning.

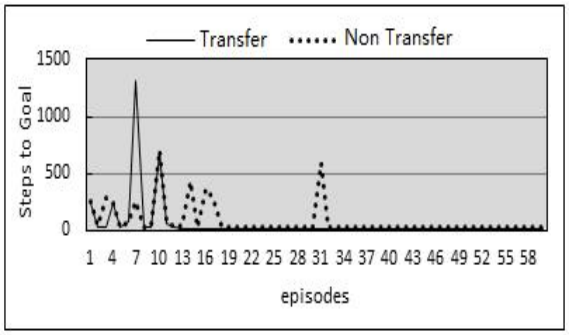

Fig. 8: The comparison between using and not using knowledge transfer algorithm.

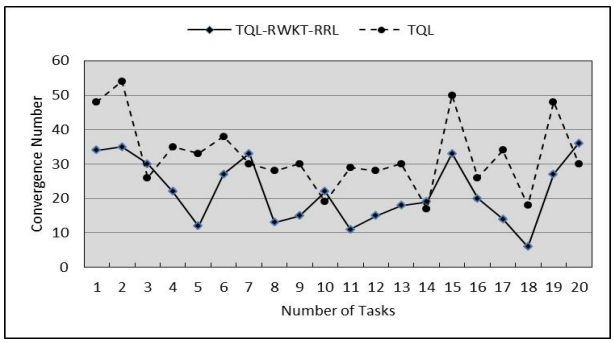

Fig. 9: The comparison between using and not using knowledge transfer algorithm.

TQL algorithm is a deterministic search algorithm, and it selects the next state with maximum Q value each time. But the adequacy of the exploration is guaranteed because the Q value of all the corresponding next states is updated at the same time at each exploration step, so the TQL algorithm does not fall into the "local trap".

6. Conclusion

Aiming at the slow speed of convergence in reinforcement learning, this paper put forwards the TQL-RWKT-RRL algorithm to improving the learning speed in Q-learning. TQL method improves the Q value updating method in each episode, and RWKT method transfers the experience in small environment to complex environment by means of rolling windows strategy. TQL-RWKT-RRL algorithm makes the robot gain useful prior knowledge and the speed of robot learning is accelerated. Finally TQL-RWKT-RRL algorithm is applied to the path-planning problem in large-scale complex environment. Simulation results show that the convergence rate is greatly improved.

7. Acknowledgements

This research project was supported by two Shaanxi Province Founds (Program No. 2017ZDXM-GY-008 and 2016MSZD-G-8-1), and supported by two National Funds (Program No. 2017KF100037 and MJ-2015-D-66).

8. References

[1] Kober Jens, Peters Jan. Reinforcement Learning in Robotics: A Survey [J]. *International Journal of Robotics Research*, 2013, 32 (11): 1238-1274.

[2] Koga Marcelo Li, Silva Valdinei Freire da, Cozman Fabio Gagliardi, et al. Speeding-up reinforcement learning through abstraction and transfer learning[C]: *Proceedings of Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, USA,* 2013:119-126.

[3] Hu J., Zhu Q. B. Path planning of robot for unknown environment based on prior knowledge rolling Q-learning [J]. *Control & Decision*, 2010, 25 (9): 1364-1368.

[4] Morozs Nils, Clarke Tim, Grace David. Distributed Heuristically Accelerated Q-Learning for Robust Cognitive Spectrum Management in LTE Cellular Systems [J]. *IEEE Transactions on Mobile Computing*, 2016, 15 (4): 817-825.

[5] Hengst Bernhard. Hierarchical Reinforcement Learning [J]. *Brain Mapping*, 2015, 41 (3): 367-373.

[6] Liu Quan,Fu Qiming,Yang Xudong, et al. A scalable parallel reinforcement learning method based on intelligent scheduling [J].*Jounal of Computer Research and Development,* 2013, 50 (4): 843-851.

[7] Yu Weiwei, Wang Shen, Madani Kurosh, et al. A framework of autonomous knowledge transfer for robot navigation task[C]: *Proceedings of International Joint Conference on Awareness Science and Technology and Ubi-Media Computing,* 2014:52-58.

[8] Bianchi Reinaldo A.C., Jr Luiz A. Celiberto, Santos Paulo E., et al. Transferring knowledge as heuristics in reinforcement learning: A case-based approach [J]. *Artificial Intelligence,* 2015, 226: 102-121.