Downloaded from http://iranpaper.ir
http://tarjomano.com ترجمه تخصصی مقالات علمی Tarjomano.com

SICE-ICASE International Joint Conference 2006
Oct. 18-21, 2006 in Bexco, Busan, Korea

Reinforcement Learning through Interaction among Multiple Agents

Hitoshi Iima[1] and Yasuaki Kuroe[2]

[1]Department of Information Science, Kyoto Institute of Technology, Kyoto, Japan
(Tel : +81-75-724-7467; E-mail: iima@kit.ac.jp)
[2]Center for Information Science, Kyoto Institute of Technology, Kyoto, Japan
(Tel : +81-75-724-7445; E-mail: kuroe@kit.ac.jp)

Abstract: In ordinary reinforcement learning algorithms, a single agent learns to achieve a goal through many episodes. If a learning problem is complicated, it may take a much computation time to obtain the optimal policy. Meanwhile, for optimization problems, multi-agent search methods such as particle swarm optimization have been recognized that they are able to find rapidly a global optimal solution for multi-modal functions with wide solution space. This paper proposes a reinforcement learning algorithm by using multiple agents. In this algorithm, the multiple agents learn through not only their respective experiences but also interaction among them. For the interaction methods this paper proposes three strategies: the best action-value strategy, the average action-value strategy and the particle swarm strategy.

Keywords: reinforcement learning, multi-agent, particle swarm optimization

1. INTRODUCTION

In ordinary reinforcement learning algorithms [1], a single agent learns to achieve a goal through many episodes. If a learning problem is complicated, it may take a much computation time to obtain the optimal policy. Meanwhile, for optimization problems, multi-agent search methods such as genetic algorithms and particle swarm optimization (PSO) [2] have been recognized that they are able to find rapidly a global optimal solution for multi-modal functions with wide solution space. Thus, in the reinforcement learning, an optimal policy could be obtained rapidly by using multi-agent algorithms in which multiple agents learn through referring learning results of other agents.

In this paper, a reinforcement learning algorithm by using the multiple agents is proposed for finding rapidly the optimal policy. In this algorithm, the multiple agents learn through interaction among them, while each of them learns individually by using an usual reinforcement learning algorithm. Any existing reinforcement learning algorithm can be used for an usual reinforcement learning algorithm, and we use Q-learning [4] which is one of typical reinforcement learning algorithms.

In the proposed learning method it is important how to design update strategies of Q-values through the interaction. We propose three strategies for this update procedure. In the first strategy, we utilize actively the best Q-values among those of all the agents during Q-learning. In the second strategy, the Q-values of each agent are updated by using both its current Q-values and the best Q-values. In the third strategy, they are updated by the update equations of PSO in which the best solutions found by the swarm so far are utilized for optimization. As mentioned above, PSO has been recognized that it is able to find a global optimal solution for multi-modal functions with wide solution space. Thus, the update equations of PSO are considered to be effective for complicated reinforcement learning problems.

In these strategies, it is necessary to evaluate the Q-values of each agent by an adequate method, because they are updated based on the results. This paper presents an evaluation method, in which the evaluated value is estimated based on rewards obtained during an episode. The proposed algorithm is applied to a shortest path problem, and its effectiveness is examined by comparing to ordinary Q-learning with a single agent.

2. LEARNING ALGORITHM

In ordinary reinforcement learning algorithms with a single agent, the agent often takes an useless action with a small reward, which results in a long learning time. On the other hand, in the proposed algorithm, multiple agents are prepared and some agents could take useful actions with a larger reward. In addition, since the Q-values of all the agents are updated according to Q-values of such agents who take the useful actions, it is expected that agents can learn in a shorter learning time. The proposed algorithm realizing this is described in the rest of this section.

2.1 Flow of Algorithm

First, each agent updates its own Q-values individually by using Q-learning for some episodes. Then, the Q-values of all the agents are evaluated by an adequate method, and the Q-values evaluated superior to those of the other agents are selected. We call them the best Q-values. Then, each agent receives the best Q-values from another agent, and updates its own Q-values according to an adequate strategy by using the best Q-values. These procedures are repeated until a termination condition is satisfied.

Although the Q-values are not evaluated in ordinary Q-learning, the proposed algorithm requires to evaluate them in order to select the best Q-values which bring a large reward. Thus, an appropriate criterion to evaluate them has to be introduced, which will be discussed in Subsection 2.3.

In the proposed algorithm, there are two kinds of procedures of updating the Q-values of each agent. One is

89-950038-5-5 98560/06/$10 © 2006 ICASE

Downloaded from http://iranpaper.ir
http://tarjomano.com ترجمه تخصصی مقالات علمی Tarjomano.com

the procedure of Q-learning, which is performed in the inner loop. The other is the procedure based on interaction among the agents, which is performed in the outer loop. The proposed algorithm should be designed with balancing frequencies of applying these procedures. Thus, we introduce a parameter which indicates the number of episodes Y for which Q-learning is performed in the inner loop.

The flow of the proposed algorithm is as follows.

Flow of algorithm

$Q_i(s,a)$: state-action value of agent i for action a in state s.
Y : number of episodes for which Q-learning is performed in the inner loop.
I : number of agents
T : total number of episodes
Step 1 Set the initial values of $Q_i(s,a)$ $(\forall i,s,a)$, and set $t \leftarrow 0$.
Step 2 Update $Q_i(s,a)$ by using Q-learning for Y episodes for each agent i. Calculate the evaluated value for Q-values by the method described in Subsection 2.3 for every agent and every episode. This update procedure of Q-learning will be described in Subsection 2.2.
Step 3 Update $Q_i(s,a)$ by the interaction among the agents based on the evaluation result in Step 2. This update procedure to do it will be described in Subsection 2.4.
Step 4 Set $t \leftarrow t+IY$. If $t \geq T$, terminate this algorithm. Otherwise, return to Step 2.

2.2 Update of Q-values by Q-learning

In Step 2 of **Flow of algorithm**, each agent updates its own Q-values individually by using Q-learning for Y episodes. Q-learning for one episode is performed in the following standard way. Note that the symbol i of agent is omitted in this procedure for simplicity.

Q-learning for one episode

s^* : current state.
a' : action taken according to $Q(s^*,a)$ in state s^*.
r : reward.
$s_{\rm n}$: next state.
ε : probability of random action.
$A(s^*)$: set of allowable actions in state s^*.
α : step-size parameter $(0<\alpha<1)$.
γ : discount-rate parameter $(0<\gamma<1)$.
K : upper bound of number of actions.
Step 1 Set the initial state of s^*, and set $k \leftarrow 1$.
Step 2 Take an action a' according to $Q(s^*,a)$ by using the ε-greedy method. As a result, a reward r and the next state $s_{\rm n}$ are obtained. In the ε-greedy method, an action is taken randomly with probability ε, and the action whose value is maximum $(\max_a Q(s^*,a))$ is taken with probability $1-\varepsilon$.
Step 3 Update $Q(s^*,a')$ by

$$Q(s^*,a') \leftarrow Q(s^*,a') + \alpha\,[\,r + \gamma \max_{a'' \in A(s_{\rm n})} Q(s_{\rm n},a'') - Q(s^*,a')]. \quad (1)$$

Step 4 Set $s^* \leftarrow s_{\rm n}$.
Step 5 If the agent achieves its goal or $k=K$, terminate this procedure. Otherwise, set $k \leftarrow k+1$ and return to Step 2.

2.3 Evaluation of Q-values

In Step 2 of **Flow of algorithm**, the evaluated value for the Q-values of each agent is calculated in order to evaluate them updated by using Q-learning for one episode. This subsection presents a method of evaluating them. Since the objective of reinforcement learning is to maximize the return, it seems to be most suitable to evaluate them by directly calculating the return. However, it is not practical, because this calculation requires many simulations and is not done in ordinary reinforcement learning algorithms.

We propose a method of evaluating Q-values by using an evaluation which approximates the return. Basically the evaluated value is defined as the sum of rewards obtained during one episode. However, to simply sum up rewards causes the following problem. Since Q-values are updated whenever an agent takes an action, Q-values around the beginning of the episode are different from those at the end of the episode. Even if a new episode begins with the Q-values at the end, the same actions are not necessarily taken and the same rewards are not necessarily obtained. The rewards obtained by using the Q-values around the beginning are considered to have a little relation with those at the end in this sense. Thus, such rewards are summed up after they are discounted. Therefore, the evaluated value E for the Q-values which each agent has updated by using Q-learning for one episode is defined as

$$E = \sum_{k=1}^{L} d^{L-k} r_k \quad (2)$$

where L is the number of actions in the episode, r_k is the reward for the k-th action, and $d(<1)$ is the discount parameter.

2.4 Update of Q-values through Interaction among Agents

In the proposed algorithm, the Q-values of all the agents are updated through interaction among the agents according to the evaluation result for the Q-values. For this purpose, three kinds of strategies are proposed in this subsection. We call the reinforcement learning algorithms using these strategies BEST-Q, AVE-Q and PSO-Q, respectively.

(a) BEST-Q

If the best Q-values are copied to those for each of the other agents, each agent can improve its Q-values in future episodes. In BEST-Q the following algorithm is performed based on this idea.

BEST-Q

$Q^{\rm best}(s,a)$: best state-action value for action a in state s.
E : evaluated value for Q-values updated.
$E^{\rm best}$: evaluated value for the best Q-values.

Downloaded from http://iranpaper.ir
http://tarjomano.com ترجمه تخصصی مقالات علمی Tarjomano.com

Step 1 Set the initial values of $Q_i(s,a)$ $(\forall i,s,a)$, and set $t \leftarrow 0$.

Step 2 Perform the following procedures.

Step 2-1 Set $i^* \leftarrow 1$ and $E^{\text{best}} \leftarrow -\infty$.

Step 2-2 Set $y \leftarrow 1$.

Step 2-3 Update $Q_{i^*}(s,a)$ by performing **Q-learning for one episode** for agent i^*.

Step 2-4 Calculate the evaluated value E for $Q_{i^*}(s,a)$ by Eq. (2).

Step 2-5 If $E > E^{\text{best}}$, set $Q^{\text{best}}(s,a) \leftarrow Q_{i^*}(s,a)$ $(\forall s,a)$ and $E^{\text{best}} \leftarrow E$.

Step 2-6 If $y < Y$, set $y \leftarrow y+1$ and return to Step 2-3.

Step 2-7 If $i^* = I$, go to Step 3. Otherwise, set $i^* \leftarrow i^*+1$ and return to Step 2-2.

Step 3 Set $Q_i(s,a) \leftarrow Q^{\text{best}}(s,a)$ $(\forall i,s,a)$.

Step 4 Set $t \leftarrow t+IY$. If $t \geq T$, terminate this algorithm. Otherwise, return to Step 2.

(b) AVE-Q

In BEST-Q, the Q-values which are not best are discarded, and replaced with the best Q-values. However, they are not necessarily worthless because they are updated by agents through learning. Moreover, the diversity of Q-values may be lost, because the Q-values of all the agents for each state-action become the same Q-value whenever they are updated. Consequently, optimal Q-values may not be found. Thus, in AVE-Q each agent updates its Q-values by averaging its current Q-value and the best Q-value for each state-action. AVE-Q is the same as **BEST-Q** except Step 3, and Step 3 in AVE-Q is as follows.

AVE-Q (Same as BEST-Q except Step 3)

Step 3 Set $Q_i(s,a) \leftarrow \dfrac{Q^{\text{best}}(s,a)+Q_i(s,a)}{2}$ $(\forall i,s,a)$.

Although AVE-Q is similar to an algorithm proposed by M. Tan [3], it is used as reinforcement learning for a multi-agent environment. M. Tan applies it to a pursuit problem in which two hunter agents seek to capture prey agents, and reports that its performance is better than one obtained by ordinary Q-learning.

(c) PSO-Q

Multi-agent search methods are often used for finding a global optimal solution in optimization. Thus, an optimal policy could be also obtained rapidly by applying an update procedure used in the multi-agent search methods. In this paper, PSO [2] is used as one of them. It originates in social behavior, and each agent updates its candidate solution by utilizing its own personal best and the global best.

Here, PSO is outlined. Consider a problem of determining values of N decision variables $x = (x_1, x_2, \cdots, x_N)$ which minimize an objective function $f(x)$. In PSO, a swarm is prepared and its all particles are used for solving this problem. Let J be the number of the particles (the swarm size), and j-th particle is denoted by P^j $(j = 1,2,\cdots,J)$. Let $x^j(t) = (x_1^j(t), x_2^j(t), \cdots, x_N^j(t))$ be the decision variable vector (the candidate solution) of particle P^j at iteration t. The candidate solution of P^j at the next iteration $t+1$ is determined as follows.

$$v_n^j(t+1) = wv_n^j(t) + c_1r_1(p_n^j(t) - x_n^j(t)) + c_2r_2(g_n(t) - x_n^j(t)) \quad (3)$$

$$x_n^j(t+1) = x_n^j(t) + v_n^j(t+1) \quad (4)$$

$$(n = 1,2,\cdots,N; j = 1,2,\cdots,J; t = 1,2,\cdots,T)$$

where

$v^j(t) = (v_1^j(t), v_2^j(t), \cdots, v_N^j(t))$: velocity vector of P^j at t,

w : weight parameter called *inertia weight*,

c_1, c_2 : weight parameters called *acceleration coefficients*,

r_1, r_2 : uniform random numbers in the range from 0 to 1,

$p^j(t) = (p_1^j(t), p_2^j(t), \cdots, p_N^j(t))$: best solution found by P^j so far, which is called *personal best*,

$g(t) = (g_1(t), g_2(t), \cdots, g_N(t))$: best solution found by all particles so far, which is called *global best*,

T : maximum iteration number.

The reinforcement learning problem coped with in this paper is considered to be a kind of optimization problem by regarding a solution candidate and an objective value as Q-values and an evaluated value E, respectively. Thus, optimal Q-values could be found in a short time by using the procedure of updating the candidate solution in PSO.

In PSO-Q, the personal best of agent i and the global best are memorized. The personal best is defined as the best Q-values found by the agent i so far, and the global best is defined as the best Q-values found by all agents so far. Each agent updates its Q-values by using these two kinds of best Q-values. The flow of PSO-Q is as follows.

PSO-Q

$V_i(s,a)$: difference of $Q_i(s,a)$ before and after its update.

$P_i(s,a)$: best state-action value for action a in state s found by agent i so far.

$G(s,a)$: best state-action value for action a in state s found by all agents so far.

E_i : evaluated value for $P_i(s,a)$.

E^{G} : evaluated value for $G(s,a)$.

W, C_1, C_2 : weight parameters.

R_1, R_2 : uniform random numbers in the range from 0 to 1.

Step 1 Set the initial values of $Q_i(s,a)$ and $V_i(s,a)$ $(\forall i,s,a)$, and set $t \leftarrow 0$, $E_i \leftarrow -\infty$ $(\forall i)$ and $E^{\mathrm{G}} \leftarrow -\infty$.

Step 2 Perform the following procedures.

Step 2-1 Set $i^* \leftarrow 1$.

Step 2-2 Set $y \leftarrow 1$.

Step 2-3 Update $Q_{i^*}(s,a)$ by performing **Q-learning for one episode** for agent i^*.

Step 2-4 Calculate the evaluated value E for $Q_{i^*}(s,a)$ by Eq. (2).

Step 2-5 If $E > E_{i^*}$, set $P_{i^*}(s,a) \leftarrow Q_{i^*}(s,a)$ $(\forall s,a)$ and $E_{i^*} \leftarrow E$.

Step 2-6 If $E > E^{\mathrm{G}}$, set $G(s,a) \leftarrow Q_{i^*}(s,a)$ $(\forall s,a)$ and $E^{\mathrm{G}} \leftarrow E$.

Downloaded from http://iranpaper.ir
http://tarjomano.com ترجمه تخصصی مقالات علمی Tarjomano.com

Step 2-7 If $y < Y$, set $y \leftarrow y + 1$ and return to Step 2-3.

Step 2-8 If $i^* = I$, go to Step 3. Otherwise, set $i^* \leftarrow i^* + 1$ and return to Step 2-2.

Step 3 Update $V_i(s,a)$ and $Q_i(s,a)$ $(\forall i, s, a)$ by

$$V_i(s,a) \leftarrow WV_i(s,a) + C_1R_1(P_i(s,a) - Q_i(s,a)) + C_2R_2(G(s,a) - Q_i(s,a)), \quad (5)$$

$$Q_i(s,a) \leftarrow Q_i(s,a) + V_i(s,a). \quad (6)$$

Step 4 Set $t \leftarrow t + IY$. If $t \geq T$, terminate this algorithm. Otherwise, return to Step 2.

3. NUMERICAL EXPERIMENTS

The effectiveness of the proposed learning algorithms is examined by applying them to a shortest path problem. If the start cell in this problem is close to the goal cell and the position of the goal cell is fixed, then the optimal policy can be obtained simply by ordinary Q-learning. As a problem which can not be solved simply by ordinary Q-learning, we set up a shortest path problem with a goal position being changing within a range.

3.1 Shortest Path Problem

The shortest path problem is that an agent finds the shortest path from the start cell to the goal cell in an n by n grid world. In this grid world, we let the coordinates at the bottom left be (1,1), and those at the top right be (n,n). While the start cell is (1,1) and fixed, the goal cell is changing within a range and its coordinates are determined at random.

An agent recognizes its own coordinates (x,y), and has four possible actions to take from moving up, down, left or right. That is to say, it has actions to move into $(x, y+1)$, $(x, y-1)$, $(x-1, y)$ or $(x+1, y)$. Some cells have walls at the boundaries with their adjacent cells, and the movement of the agent is blocked by a wall and the edge of the grid world.

Fig. 1 shows an example of the grid world for n=10. In our experiments, the size n is forty, and the x and y coordinates of the goal cell are determined randomly in the range from 35 to 40, respectively. Hence, they are determined randomly from 36 cells. Four cases 1, 2, 3 and 4 in which the number and positions of the walls are different each other are generated according to the above condition. These cases are numbered in the ascending order of the number of walls, and there is no wall in Case 1.

3.2 Experimental Set up and Results

The proposed algorithms, BEST-Q, AVE-Q and PSO-Q, and ordinary Q-learning with a single agent are applied to the shortest path problem, and their experimental results are compared. The following values are used for the parameters in ordinary Q-learning:

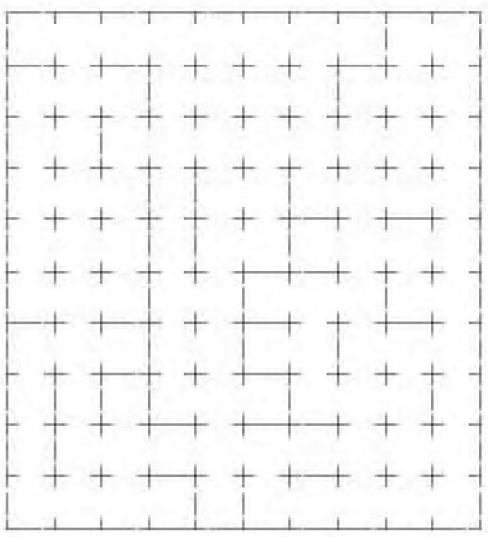

Fig. 1 Example of grid world for n=10

step-size parameter : α=0.1,
discount-rate parameter : γ=0.999,
probability of random action : ε=0.2,
upper bound of number of actions : K=10000,
total number of episodes : 20000.

Moreover, all the initial values of $Q_i(s,a)$ are zero. The agent learns under the condition that the coordinates of the goal cell are changed every episode. When the agent reaches the goal cell, it gains the reward $+100$. Otherwise, it gains -1. By using this rule of reward, the evaluated value E becomes large when the number of actions to the goal cell is small. Consequently Q-values can be evaluated adequately.

As for the proposed algorithms, the following values are used for their parameters:

total number of episodes : T=20000,
number of agents : I=4,
weights in PSO-Q : W=0, C_1=C_2=2.2,
number of episodes for which Q-learning is performed in the inner loop :

$$Y = \begin{cases} 1 & \text{(for BEST-Q and AVE-Q)} \\ 5 & \text{(for PSO-Q)}, \end{cases}$$

discount parameter in Eq. (2) : d=0.999.

The number of episodes in the proposed algorithms is the same as that in ordinary Q-learning in order to compare them fairly. The values of parameters used in the procedure of Q-learning of the proposed algorithms are also the same as those in ordinary Q-learning. The values of all the parameters used in our experiments are determined through the preliminary experiments in such a way that each algorithm works best.

Figs. 2-5 show the variation of number of actions through the learning phase obtained by the four algorithms: BEST-Q, AVE-Q, PSO-Q and (ordinary) Q-learning. Note that the y-axis in these figures represents the minimum number of actions to reach the goal found

Downloaded from http://iranpaper.ir
http://tarjomano.com
ترجمه تخصصی مقالات علمی
Tarjomano.com

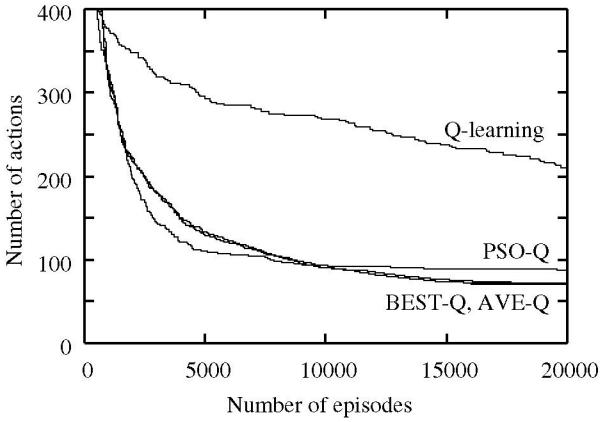

Fig. 2 Variation of number of actions through the learning phase (Case 1)

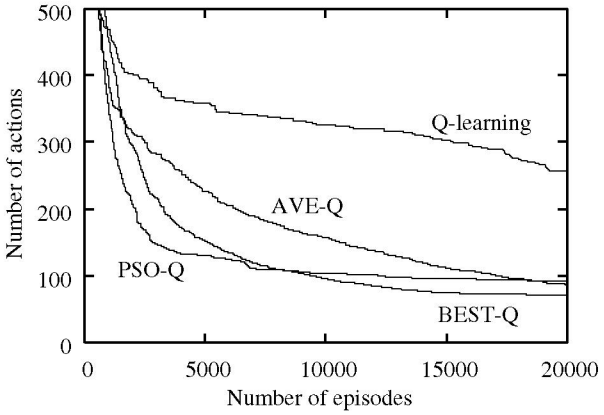

Fig. 3 Variation of number of actions through the learning phase (Case 2)

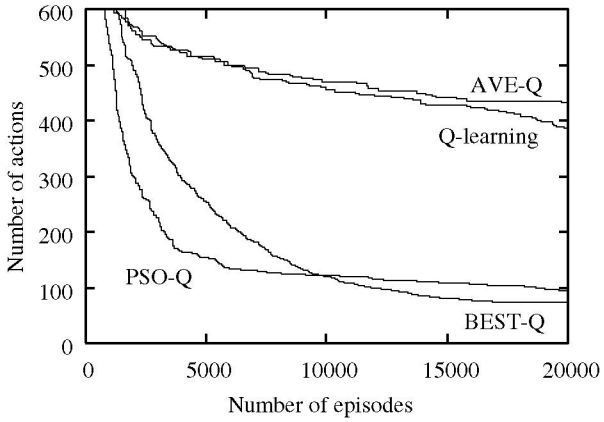

Fig. 4 Variation of number of actions through the learning phase (Case 3)

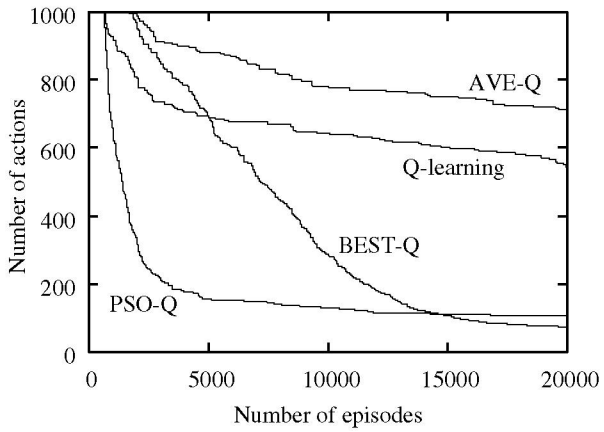

Fig. 5 Variation of number of actions through the learning phase (Case 4)

so far. Each algorithm is performed thirty times with various random seeds, and their results are averaged.

3.3 Discussion

For Case1 it is observed from Fig. 2 that the number of actions at the 20000th episode in BEST-Q and AVE-Q is smaller than that in PSO-Q and Q-learning. Therefore, a better policy can be obtained by utilizing the best Q-values. On the other hand, the number of actions before the 9000th episode in PSO-Q is smaller than that in BEST-Q and AVE-Q. Since the Q-values in PSO-Q are updated by using the global best $G(s, a)$, PSO-Q searches intensively in the neighborhood of $G(s, a)$. Consequently better Q-values are found in a shorter time. However, they become hardly better in the latter half of episodes because PSO-Q does not search out of the neighborhood of $G(s, a)$. As for Q-learning, the number of actions at every episode is larger than that obtained by the other algorithms.

For Case 2, AVE-Q becomes worse than BEST-Q, as shown in Fig. 3. In addition, the number of actions in AVE-Q barely becomes almost the same as PSO-Q when the number of episodes reaches 20000. The trend of performance of the other algorithms in Case 2 is similar to that in Case 1.

It is found from Fig. 4 that the trend of performance of PSO-Q and Q-learning in Case 3 is similar to those in Cases 1 and 2. Although BEST-Q can find the best policy around the 20000th episode in Case 3, the difference between BEST-Q and PSO-Q before the 10000th episode is larger as compared with Cases 1 and 2. That is to say, the decrease in the number of actions for BEST-Q is slower than that for PSO-Q, and the convergence speed of BEST-Q is slower. As for AVE-Q, its performance deteriorates seriously, and AVE-Q gives worse results than Q-learning.

At the 20000th episode for Case 4 in which the number of walls is most among the four cases, the number of actions in BEST-Q is the smallest among the four algorithms, as shown in Fig. 5. BEST-Q is the best algorithm at the last episode for every case. Its convergence speed, however, is slower, and it is worse than PSO-Q before the 15000th episode. Since there exist many walls in this case, a large number of actions is usually required until an agent reaches the goal cell. Therefore, even if the best Q-values which are promising are accidentally obtained at a certain episode, better Q-values are not necessarily

Downloaded from http://iranpaper.ir
http://tarjomano.com ترجمه تخصصی مقالات علمی Tarjomano.com

Table 1 Comparison of number of actions among four algorithms after learning (Case 2)

(a) T=5000

BEST-Q	AVE-Q	PSO-Q	Q-learning
2211	2924	1160	2995

(b) T=20000

BEST-Q	AVE-Q	PSO-Q	Q-learning
282	454	497	999

found at the next episode. This is because the convergence speed of BEST-Q becomes slower as the number of walls increases.

As for the other algorithms in Case 4, PSO-Q can find rapidly good Q-values as shown in Fig. 5, and the update equations of PSO are effective regardless of the number of walls. Although Q-learning before the 5000th episode is better than BEST-Q, the number of episodes hardly decreases in Case 4 as well as in Cases 1, 2 and 3. AVE-Q is worse than Q-learning at every episode in Case 4 as well as in Case 3. Consider a reason why the performance of AVE-Q deteriorates for cases with many walls. As mentioned above, a large number of actions is usually required in these cases until an agent reaches the goal cell. Therefore, the Q-values of each agent are bad around the beginning of episodes. Even if the best Q-values which are promising are accidentally obtained, the Q-values of each agent are not improved because each Q-value is updated by averaging it and the best one. This trend appears remarkably as the number of walls increases. This is because the performance of AVE-Q deteriorates. On the other hand, the performance of BEST-Q is good, because bad Q-values are discarded and the best Q-values which are promising are copied to those of each agent. The performance of PSO-Q is also good, because the best Q-values, which are promising, obtained at a certain episode are stored as the personal best and the global best, and each agent can update its own Q-values by using them after the episode.

Finally, simulations are performed by using the Q-values obtained through the learning phase with 5000 or 20000 episodes in each algorithm, and the number of actions to reach the goal cell is compared. Since there exist 36 coordinates of the goal cell in this problem, simulations are performed ten times for each of them and their results are averaged. The ε-greedy method is used for taking an action in the simulations, and the probability ε of random action is 0.2, which is the same value as the learning phase. Table 1 shows the mean number of actions to the goal cell obtained by the simulations. These results are for Case 2, and similar results are observed for the other cases. It is confirmed from this table that PSO-Q and BEST-Q are the best for T=5000 and T=20000, respectively.

It is concluded from the above experimental results that PSO-Q is superior in the sense that it can obtain a somewhat good policy in the shortest computation time. Furthermore, BEST-Q is superior in the sense that it can obtain a better policy if a longer computation time is allowable.

4. CONCLUSION

A reinforcement learning algorithm in which multiple agents learn in parallel has been proposed. In this algorithm, each agent learns individually by using Q-learning, and also learns through interaction among agents. Three strategies have been proposed for realizing this interaction. In the first strategy, the best Q-values are straightforwardly copied to those of each agent. In the second strategy, each agent updates its Q-values by averaging its Q-value and the best one for each state-action. In the third strategy, each agent updates its Q-values by the update equations of PSO using the personal best and the global best. We have implemented three algorithms, BEST-Q, AVE-Q and PSO-Q, with these strategies.

These algorithms have been applied to a shortest path problem, and it has been confirmed from the experimental results that PSO-Q is superior in the sense that it can obtain a somewhat good policy in the shortest computation time. Furthermore, BEST-Q is superior in the sense that it can obtain a better policy if a longer computation time is allowable. The proposed algorithms could be effective for not only this problem but also other problems by designing an interaction strategy adequately.

REFERENCES

[1] R.S. Sutton and A.G. Barto: *Reinforcement Learning*, MIT Press, 1998.

[2] J. Kennedy and R.C. Eberhart: *Swarm Intelligence*, Morgan Kaufmann Publishers, 2001.

[3] M. Tan: "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents", *Proceedings of the 10th International Conference on Machine Learning*, pp.330–337, 1993.

[4] C.J.C.H. Watkins and P. Dayan: "Q-Learning", *Machine Learning*, Vol. 8, No. 3, pp.279–292, 1992.