Learning Method by Sharing Activity Logs in Multiagent Environment

Keinosuke Matsumoto, Takuya Gohara, and Naoki Mori
Department of Computer Science and Intelligent Systems
Graduate School of Engineering, Osaka Prefecture University
Sakai, Osaka, Japan
email: {matsu, gohara, mori}@cs.osakafu-u.ac.jp

Abstract—**Applications of multiagent systems are expected from the point of view of the parallel and distributed processing. Reinforcement learning is used as an implementation method for learning agents' actions. However, the problem is that, the higher the number of agents to deal with, the slower the speed of learning becomes. To solve this problem, this paper proposes a new reinforcement learning method that can learn quickly by using past actions of its own and of other agents. Agents can learn good actions in the early stage of learning by this method. However, if agents keep learning, learning efficiency will deteriorate. The method controls to reduce effects of other agents' actions in the later stage of learning. In experiments, agents learned good actions in various environments. Thus, the success of the proposed method was verified.**

Keywords- machine learning; Q-learning; sharing of activity history; agents; hunter game

I. INTRODUCTION

In recent years, information has distributed and grown by the rapid development of the Internet and multimedia. Systems also become large and complicated. It is difficult for centralized systems, that make decisions by bringing information in one place, to deal with a lot of information and to process it. From the viewpoint of the parallel and distributed processing, the application of multiagent systems [1] that exchange information between agents [2] is expected.

It is difficult to follow environmental changes that humans could not forecast and do not carry out suitable actions. The most important thing for each agent in a multiagent system is to learn by itself. Each agent needs to learn a suitable judgment standard from one's experience and information collected from other agents. Reinforcement learning [3][4] attracts attention as an implementation method of multiagent systems. It can be very effective means, because it autonomously learns by setting only a reward, if a goal has been given.

In this study, reinforcement learning is applied to a multiagent problem, a hunter game [5]. It is widely used as a cooperative problem solving [6][7] under multiagent environment as a benchmark. If a hunter game becomes complicated and the number of agents increases, the number of states increases exponentially. The problem is that the speed of learning slows down. Ono et al. proposed Modular Q-Learning (MQL) [8] to solve this problem, but it had a disadvantage of using much memory. With respect to memory, another method that reduced memory [9] was proposed. In this method, each agent has only one Q-value table by not distinguishing each agent with the same purpose. On the basis of these methods, this study proposes a method that increases learning efficiency by using each agent's activity log [10][11] of hunter agents.

This method does not need to prepare any special communication algorithms between agents, strategies to exchange information [12][13], special exploration agents [14][15][16] etc., according to various situations. This method saves only activity history and updates the Q-value using its own or other hunters' activity history. In this way, the method shares experiences between agents by simple way of adding other hunters' activity history to Q-value table, and picks up learning speed. It makes collective intelligence efficient.

The rest of the paper is structured as follows. In Section II, the explosion of the number of states in reinforcement learning is explained. In Section III, conventional methods are described. In Section IV, the proposed method is explained. In Section V, the results of application experiments to confirm the validity of the proposed method are given. Finally, in Section VI the conclusion and future work are presented.

II. HUNTER GAME

This section describes a hunter game and the explosion of the number of states.

A. Definition of hunter game

A hunter game is one of the standard problems in multiagent systems. It is a game in which multiple hunters catch a prey (runaway) chasing in on a two-dimensional field. The definition of hunter game in this study is shown below.

-A field is a two-dimensional lattice and torus space as shown in Fig. 1.

-It is possible for multiple agents to take one lattice space.

-Each agent can take five actions of moving right, left, up, down or stop.

 ISBN: 978-1-61208-506-7

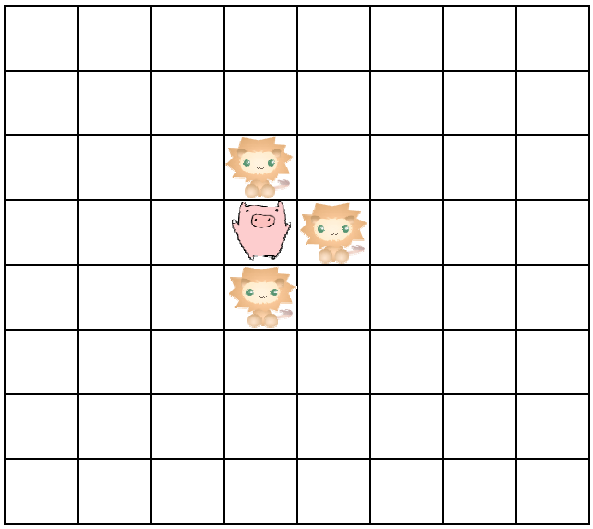

Figure 1. Hunter game.

-A hunter has perfect perception, and it recognizes a prey and other hunters in relative coordinates from itself.

-A unit time during which each agent takes one action is called a time step, and a period of time from an initial state to a goal (i.e., hunters catch a prey) is called an episode.

B. Explosion of the number of states

Q-Learning [17] is one of bootstrap type reinforcement learning. In Markov decision process, like Q-Learning, if a learning rate is appropriately adjusted, convergence to an optimal solution in infinite time has been proven [18].

In Q-Learning of the hunter game, an action is evaluated on pair (s, a) considering all observable states s and each possible action a. The evaluated value is utilized for the same pair of state and action. It requires a lot of information on (s, a) to make Q-Learning effective. For example, if the size of field is m x m and the number of hunters is n, one hunter can see m^{2n} identifiable states (positional combinations of other hunters and the prey). Because each state has five kinds of actions, state and action pair is $5m^{2n}$. In the hunter game with multiple hunters, state explosion cannot be avoided because the exponent includes n.

By the explosion of the number of states, the learning speed will become slow. Therefore, in Q-Learning in multiagent environment, it becomes an important subject to figure out how the number of states can be reduced.

III. CONVENTIONAL METHODS

This section describes related work of this study.

A. Modular Q-Learning

Ono et al. proposed MQL [8] to solve the state explosion in hunter games. Completely Perceptual Q-Learning (CPQL) [19] is perfect perception learning, and it uses relative coordinates of all hunters in order to define states. On the other hand, MQL uses a partial state that consists of a hunter and another one. The number of states of field size m x m and n hunters is m^4. Since the exponent is a constant and it is not influenced by the number of hunters, it can prevent the state explosion.

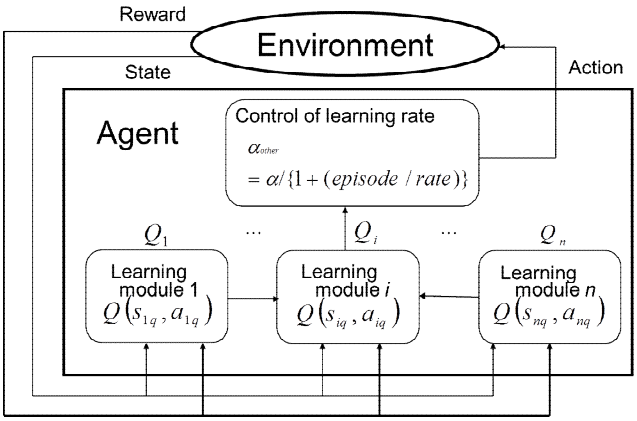

Figure 2. Architecture of the proposed method.

The learning accuracy of MQL deteriorates because of imperfect perception by observing partial states. In addition, if n hunters exist, the number of partial states becomes n-1, and n-1 learning machines are prepared per hunter. A total of $n(n$-1) learning machines are needed. The size of Q-value tables tends to become large and the amount of memory will increase.

B. Centralized Modular Q-Learning

Matsumoto et al. proposed Centralized Modular Q-Learning (CMQL) [9] to solve the memory problem that is one of the problems of MQL. In a hunter game, hunters should just surround a prey. It is not necessary to recognize the kind of hunters that surround the pray. Therefore, CMQL does not distinguish the characteristics of each hunter and n-1 learning machines that the hunter has in MQL can be reduced to one learning machine. In CMQL, a hunter has only one Q-value table of the partial state. Since the number of Q-value tables becomes one per hunter, only n Q-value tables are required in all if n hunters exist.

IV. PROPOSED METHOD

In this section, a method that raises learning efficiency is described on the basis of MQL and CMQL. Fig. 2 shows the basic concept of the proposed method.

A. Learning method by sharing activity logs

In the hunter game, all hunters have the common purpose of catching a prey. In such environment, it is useful to use learned actions of other hunters to catch the prey. Appropriate actions can be learned with fewer number of trial times by learning actions of other hunters. In this study, a method of updating Q-value on the basis of other hunters' activity logs is proposed. The number of times of updating for every episode increases, but the method raises the learning efficiency for every episode. The algorithm of the proposed method is shown below.

 ISBN: 978-1-61208-506-7

The number of hunters is n and a prey is captured at q steps. Each hunter is observing states s_1, s_2, ---, s_n and actions are a_1, a_2, ---, a_n.

(1) In each episode, save hunters' coordinates and actions for every step, and for up to t steps. These are activity logs.
(2) Give awards to all hunters' Q-value Q (s_1, a_1), Q (s_2, a_2), ---, Q(s_n, a_n) if the prey is captured.
(3) $i = q$
(4) $Q(s_{i-1}, a_{i-1}) \leftarrow (1-\alpha)\, Q(s_{i-1}, a_{i-1}) + \alpha\, [r + \gamma \max_a Q(s_i, a_i)]$
(5) Replace i by i-1 and repeat (4) until $i \leq q$-t or $i \leq 1$.

In the above-mentioned algorithm, t is t=1000. Combining this algorithm with CMQL makes a more efficient learning method.

Although learning has become early in the proposed method, final learning results tend to deteriorate compared with the conventional methods without sharing activity logs. The learning accuracy of the proposed method becomes bad by learning actions of other hunters at the final learning stage. For this reason, the learning rate using other hunters' actions is decreased according to the number of episodes. An influence on learning by other hunters' actions is lessened as learning progresses. This will be an approach that utilizes other hunters' activity logs at the early learning stages and uses only each hunter's log at the final learning stage.

B. Control of learning rate

It is difficult to find an optimal action if a hunter learns other hunters' actions in the final stage of learning. Learning rate of learning other hunters' activity logs should be decreased in proportion to the number of episodes. If other hunters' activity logs are used at the last stage of learning, learning accuracy will reduce a little. It does not become bad by learning only for one's own log, and the learning rate at the time of updating for other hunters' activity logs should be gradually made small.

The influence of other hunters' activity logs on learning was reduced with the number of times of learning. This method (hereinafter referred to as Turned Experience CMQL (TECMQL)) is a learning approach that utilizes other hunters' activity logs in the early stage of learning and only its own log in the final stage.

The following formula defines the learning rate at learning other hunters' activity logs.

$$\alpha_{other} = \frac{\alpha}{1 + (episode \ / rate)} \quad (1)$$

where, α_{other} is a learning rate updating Q-value using other hunters' activity logs and *rate* is a constant that determines reduction rate of the learning rate. The learning rate at learning using other hunters' actions should be decreased according to the number of episodes. The value of learning rate is determined to eliminate the effect of other hunters' actions in proportion to the number of episodes.

V. EXPERIMENTS

In this section, the proposed method was applied to hunter games to confirm its validity.

A. The outline of experiments

The experiments compare the learning efficiency of the following three methods.

- Proposed method: CMQL using other hunters' activity logs (referred to as Sharing Experience CMQL (SECMQL)).
- Comparison method: CMQL using only each hunter's log (referred to as Own Experience CMQL (OECMQL)).
- Conventional method: CMQL that does not use activity logs.

These three methods were applied to a hunter game in a maze environment and two-prey hunter game.

B. Experiment 1: Hunter games in maze environment

The performances of the above-mentioned three methods were compared in the hunter game in a maze environment. In this case, hunters learn ways of bypassing walls in the maze and leading a prey to the place where is easy to catch it using the walls. The positions of walls do not change from the beginning of this experiment. Walls are grasped by absolute coordinate system. In this experiment, a partial state of CMQL consists of a relative coordinate from a hunter to any other one hunter, a relative coordinate from the hunter to a prey, and an absolute coordinate of the hunter. By this means, actions can be learned considering the positions of walls in each partial state.

Experimental conditions were as follows:

- Size of field: 8 x 8
- Number of walls in the mazy field: 21
- Number of hunters: n =3
- Action selection strategy: ε-greedy (ε = 0.01)
- Prey's action: It escapes from hunters.
- Capture state: Four lattices in left, right, top and bottom of a prey's position are surrounded by hunters or walls.
- Cost per one time step: 0.05
- Learning rate: α=0.2
- Discount rate: γ=0.8
- Maximum number of learning episodes: 300000 episodes
- Reward of hunter that caught a prey directly: 5
- Reward of hunter that did not caught the prey directly: 4

In this experiment, only three hunters cannot catch a prey without making use of walls. Hunters will learn actions that guide a prey near walls and catch a prey using the walls. At least two or less hunters can catch a prey if they make use of walls. In this case, one hunter could guide a prey for other two hunters to catch it. A reward reduced a little bit was given to the hunter that did not catch a prey directly since it contributed to the catch.

The results are shown in Fig. 3. The horizontal axis indicates the number of episodes and the vertical axis indicates the time steps to catch a prey from an initial state. Every plot shows average time steps to catch a prey of every 100 episodes. The fewer the time steps are, the better action patterns can be learned.

The learning of SECMQL became earlier until near episode no. 5000 than other methods, but the final learning result was bad compared with other methods. On the other

 ISBN: 978-1-61208-506-7

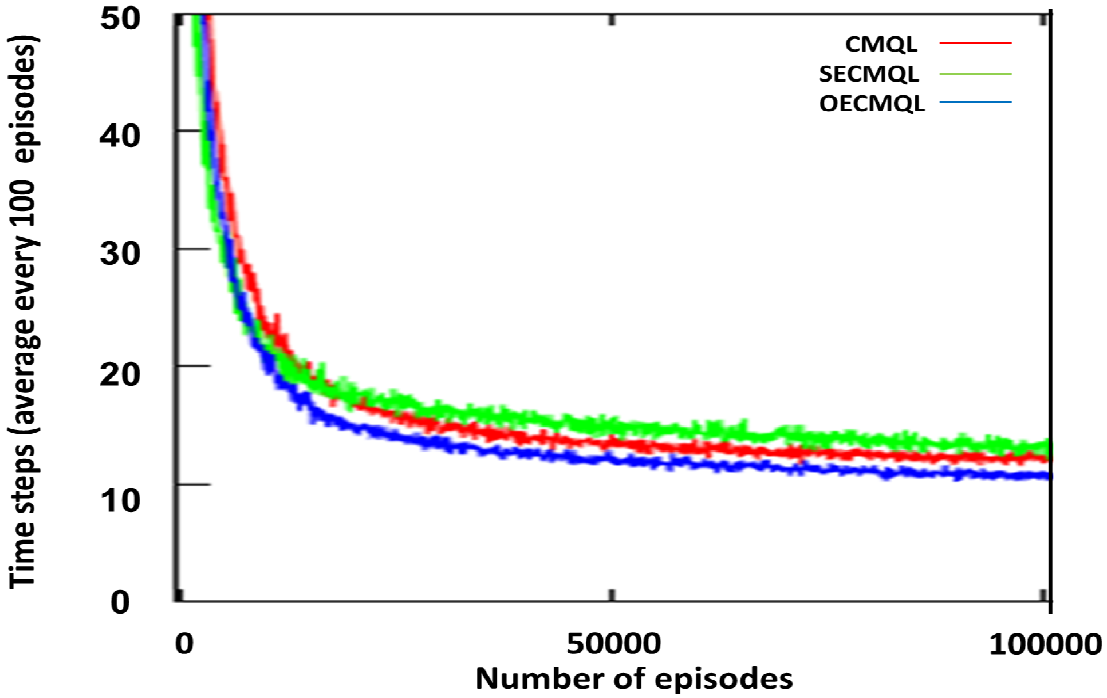

Figure 3. Results of the proposed method for maze task.

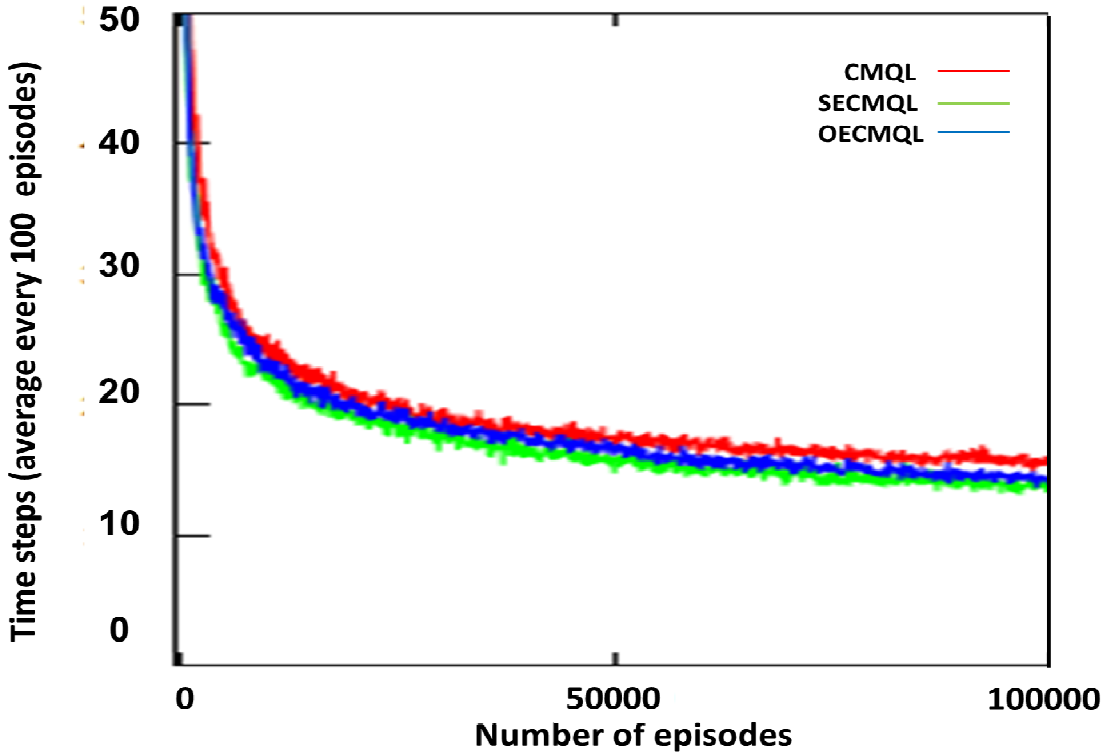

Figure 4. Results of the proposed method for two-prey game.

 ISBN: 978-1-61208-506-7

hand, OECMQL could catch with fewer number of steps compared with CMQL. SECMQL's learning accuracy was deteriorated by learning other hunters' actions in the final stage of learning.

C. Experiment 2: Two-prey hunter games

Performances of the above-mentioned three methods were compared in a hunter game that has two preys. In this game, hunters' purpose is to catch one of two preys. Since the candidate actions of a hunter increase in number, learning becomes difficult compared with the problem of one prey. In this experiment, a partial state of CMQL consists of a relative coordinate from a hunter to any other one hunter and two relative coordinates from the hunter to two preys. Since the positions of both preys can be seen, actions can be learned considering two preys.

Experimental conditions were as follows:

- Size of field: 8 x 8
- Number of hunters: n =3
- Action selection strategy: ε-greedy (ε = 0.01)
- Prey's action: They escape from hunters.
- Capture state: At least two hunters exist in left, right, top and bottom of one prey.
- Cost per one time step: 0.05
- Learning rate: α=0.2
- Discount rate: γ=0.8
- Maximum number of learning episodes: 300000 episodes
- Reward of hunter that caught a prey directly: 5
- Reward of hunter that did not caught the prey directly: 4

In this experiment, preys observe all hunters' positions and they escape from hunters on the basis of hunters' coordinates. A reward reduced a little bit was given to the hunter that did not catch a prey directly since it contributed to catch.

The results are shown in Fig. 4. In this experiment, the learning efficiency of SECMQL is the best in the early stages of learning. Since action patterns that lead to catch in early stages of learning by only one hunter are insufficient, it is useful to use other hunters' activity logs for learning. However, OECMQL found good action strategies over 100000 episode. The way a hunter individually learned in the final stage is better to get good action strategies.

D. Experiment 3: Hunter games in maze environment after control of learning rate

Performance was compared with the cases where they are with or without reducing learning rate of hunter games in a maze environment. TECMQL was added to the three methods of experiments 1 and 2 as a compared method. Experimental conditions were the same as experiment 1, and *rate* of TECMQL was 500.

Results are shown in Fig. 5. In this experiment, OECMQL shows the best learning result. TECMQL also showed equivalent learning result to OECMQL, while TECMQL maintained good efficiency in the early stage of learning.

E. Experiment 4:Two-prey hunter games after control of learning rate

Performance was compared with the cases where they are with or without reducing learning rate of hunter games that have two preys. The compared method was the same as experiment 3. Experimental conditions were the same as experiment 2, and *rate* of TECMQL was 10000.

Results are shown in Fig.6. In this experiment, TECMQL discovered the policy that could catch a prey with fewer steps than other methods. From these results, it seems to be effective to assemble a rough action strategy using actions of other hunters in early stages of learning, and then to learn the action strategy that is suitable for each hunter by individual learning.

In addition to experiment 3, TECMQL found actions that were easy to catch a prey rather than the conventional methods in various environments. However, it is necessary to adjust learning rate according to the environments.

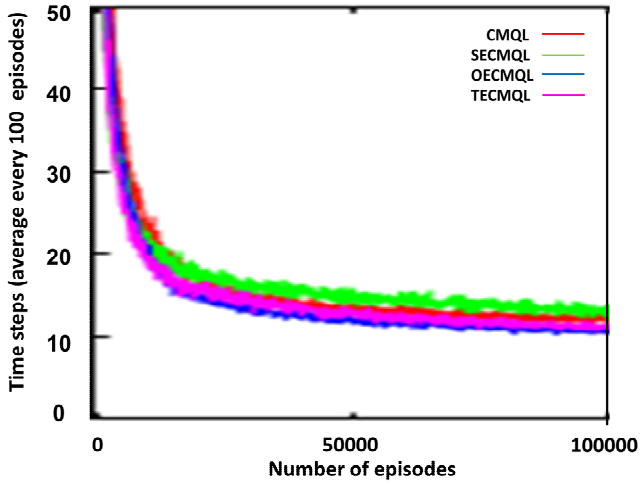

Figure 5. Results of the proposed method for maze task after control of learning rate.

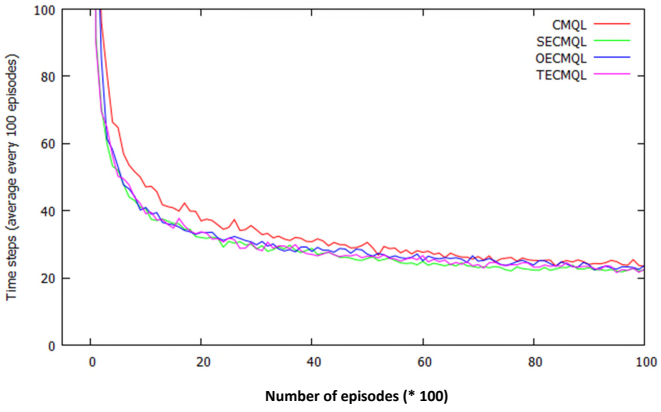

Figure 6. Results of the proposed method for two-prey game after control of learning rate.

VI. CONCLUSION

In this paper, a method, which can learn in fewer trials by sharing activity logs among hunters, was proposed. The method is based on MQL and CMQL that are methods to prevent explosion of the number of states. The performance of the proposed method was compared with CMQL. To solve the problem that the learning performance of the proposed method deteriorates in the later stage of learning when using other hunters' activity logs, it makes learning rate decrease according to the number of episodes. At the present method, the control of learning rate is dependent on the number of episodes, but it is not controlled by the contents of learning. As a future task, an index should be established to control the learning rate according to Q-value during learning.

ACKNOWLEDGMENT

This work was supported in part by JSPS KAKENHI Grant Number JP16K06424.

REFERENCES

[1] G. Weiss, Multiagent Systems: a modern approach to distributed artificial intelligence, MIT Press, 1999.

[2] S. J. Russell and P. Norving, Artificial intelligence: a modern approach, Prentice-Hall, Englewood Cliffs, 1995.

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

[4] H. Van Hasselt, "Reinforcement learning in continuous state and action spaces," in Reinforcement Learning, Springer Berlin Heidelberg, pp. 207-251, 2012.

[5] M. Benda, V. Jagannathan, and R. Dodhiawalla, On optimal cooperation of knowledge sources, Technical Report, BCS-G 2010-28, Boeing AI Center, 1985.

[6] I. Nahum-Shani, M. Qian, D. Almirall, W. E. Pelham, B. Gnagy, G. A. Fabiano, and S. A. Murphy, "Q-learning: A data analysis method for constructing adaptive interventions," Psychological methods, vol. 17, no. 4, p. 478, 2012.

[7] S. Shamshirband, A. Patel, N. B. Anuar, M. L. M. Kiah, and A. Abraham, "Cooperative game theoretic approach using fuzzy Q-learning for detecting and preventing intrusions in wireless sensor networks," Engineering Applications of Artificial Intelligence, vol. 32, pp. 228-241, 2014.

[8] N. Ono and K. Fukumoto, "Multi-agent reinforcement learning: a modular approach," Proc. of AAAI ICMAS-96, pp.252-258, 1996.

[9] K. Matsumoto, T. Ikimi, and N. Mori, "A switching Q-learning approach focusing on partial states," Proc. of the 7th IFAC Conference on Manufacturing Modelling, Management, and Control (MIM 2013) IFAC, pp. 982-986, ISBN: 978-3-902823-35-9, June 2013.

[10] M.Tan, "Multi-agent reinforcement learning : independent vs. cooperative agents," Proc. of the 10th International Conference on Machine Learning, pp.330-337, 1993.

[11] R. M. Kretchmar, "Parallel reinforcement learning," Proc. of the 6th World Conference on Systemics, Cybernetics, and Informatics, vol.6, pp.114-118, 2002.

[12] H. Iima and Y. Kuroe, "Swarm reinforcement learning algorithm based on exchanging information among agents," Transactions of the Society of Instrument and Control Engineers, vol. 42, no. 11, pp. 1244-1251, 2006 (in Japanese).

[13] S. Yamawaki, Y. Kuroe, and H. Iima, "Swarm reinforcement learning method for multi-agent tasks," Transactions of the Society of Instrument and Control Engineers vol. 49, no. 3, pp. 370-377, 2013 (in Japanese).

[14] T. Tateyama, S. Kawata, and Y. Shimomura, "Parallel reinforcement learning systems using exploration agents," Transactions of the Japan Society of Mechanical Engineers Series C vol. 74, no. 739, pp. 692-701, 2008 (in Japanese).

[15] Y. M. De Hauwere, P. Vrancx, and A. Nowe, "Future Sparse Interactions: A MARL approach," Proc. of the 9th European Workshop on Reinforcement Learning, pp. 1-3, 2011.

[16] H. Igarashi, M. Handa, S. Ishihara, and I. Sasano, "Agent control in multiagent systems– Reinforcement learning of weight parameters in particle swarm optimization," The Research Reports of Shibaura Institute of Technology, Natural Sciences and Engineering vol. 56, pp. 1-8, 2012 (in Japanese).

[17] C. J. C. H. Watkins and P. Dayan, "Technical note Q-learning," Machine Learning, vol. 8, no. 3, pp. 279-292, 1992.

[18] S. J. Bradtke and M. O. Duff, "Reinforcement learning method for continuous-time Markov decision problems," Advances in Neural Information Processing Systems, vol. 7, pp. 393-400, 1994.

[19] A. Ito and M. Kanabuchi, "Speeding up multi-agent reinforcement learning by coarse-graining of perception — hunter game as an example—," IEICE Trans. Information and Systems D-I, vol. J84-D-I, no. 3, pp. 285-293, 2001 (in Japanese)

 ISBN: 978-1-61208-506-7