ICIC Express Letters
Part B: Applications
Volume **11**, Number **3**, March **2020**

ICIC International ©2020 ISSN 2185-2766
pp. **237–244**

VISION-BASED BEHAVIOR ACQUISITION BY DEEP REINFORCEMENT LEARNING IN MULTI-ROBOT ENVIRONMENT

Ryoma Watanuki[1], Tadashi Horiuchi[1] and Toshiyuki Aodai[2]

[1]Advanced Engineering Faculty
National Institute of Technology, Matsue College
14-4, Nishi-ikuma, Matsue, Shimane 690-8518, Japan
{ s1821; horiuchi }@matsue-ct.jp

[2]Monozukuri Engineering Department
Tokyo Metropolitan College of Industrial Technology
8-17-1, Minami-Senju, Arakawa, Tokyo 116-8523, Japan
aodai@metro-cit.ac.jp

Received September 2019; accepted December 2019

Abstract. *Deep Q-Network (DQN) is one of the most famous methods of deep reinforcement learning. DQN approximates the action-value function using Convolutional Neural Network (CNN). We have already realized that a single mobile robot acquired behavior to avoid walls and obstacles by using DQN. In this research, we apply DQN method to multi-robot environment and we aim to realize that multiple mobile robots acquire behavior to avoid walls and other robots. This task is more difficult in multi-robot environment than in a single robot environment, due to the influence by the actions of other robots. Hence, we introduce curriculum learning framework to stabilize learning. As first step, we applied DQN and curriculum learning framework to the simulation environment that three robots exist. We realized that each robot acquired behavior to avoid the wall and other robots based on images of the camera mounted on itself. In addition, we confirmed that curriculum learning framework accelerates learning.*
Keywords: Deep reinforcement learning, Deep Q-Network (DQN), Multi-robot system, Mobile robots, Behavior acquisition, Curriculum learning

1. **Introduction.** Deep reinforcement learning offers a new promise for solving complex control tasks using visual information. Developments of Deep Q-Network (DQN) [1] which is superior to human experts in several video games and AlphaGo [2] which won a human champion had strong impact on all over the world. Deep reinforcement learning has remarkable features to directly learn the action policy from the high-dimensional image data by using Convolutional Neural Network (CNN) [3].

We have already realized that the mobile robot learns to acquire good behaviors such as avoiding the wall and moving along the center line by using high-dimensional visual information as input data both in simulation environment and in real robot environment [4, 5]. Maruyama et al. [6] realized behavior acquisition by deep reinforcement learning using semantic segmentation for camera images. They expected to reduce the gap of the input data between simulator and real world by using semantic segmentation for the camera images of real world. However, success rate of robot learning is not high in the evaluation experiment using real robot.

The objective of the above researches is to realize behavior acquisition for a single robot. However, in the future, it can be assumed that cooperative actions with other robots are necessary at homes, warehouses, factories, welfare facilities, and so on. Preferred Networks, Inc. conducted technical demonstration using deep reinforcement learning in

DOI: 10.24507/icicelb.11.03.237

multi-robot environment [7]. It realized that robots acquired behavior to avoid obstacles and other robots based on omnidirectional distance.

We have already manufactured three real mobile robots and started the research to realize behavior acquisition of multi-robot system by deep reinforcement learning in real robot environment. As the first step of the research, we apply DQN which is one of the most famous methods of deep reinforcement learning, in simulation environment where three mobile robots exist. Moreover, we realize acceleration of learning in multi-robot system by introducing curriculum learning framework, where the level of task difficulty is gradually raised.

2. Method.

2.1. Deep Q-Network (DQN).

DQN uses the Q-learning method [8] which is one of the most widely-used reinforcement learning and also uses CNN in order to approximate the action-value function. The action-value function is shown by the following equation:

$$Q^{\pi}(s,a) = \boldsymbol{E}_{\boldsymbol{\pi}}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k} \,\middle|\, s_t = s, a_t = a\right] \tag{1}$$

where π denotes the policy $\pi = P(a|s)$, a is an action among candidate actions $\mathcal{A}$, s is a specific state in $\mathcal{S}$, r_t is reward at time step t and γ is discount rate.

DQN uses the method called experience replay. In order to perform experience replay, the agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ at each time step t are stored in the data set $D = \{e_1, \ldots, e_t\}$. The data set D is also called replay memory. During learning, we apply Q-learning update rule on samples of experience $e_j = (s_j, a_j, r_j, s_{j+1})$, $(1 \leq j \leq t)$ drawn uniformly at random from the data set D.

The Q-learning update at iteration i uses TD error δ_i and loss L_i calculated by the following Huber loss function:

$$\delta_i(\theta_i) = r_j + \gamma \max_{a' \in \mathcal{A}} Q\left(s_{j+1}, a'; \theta_i^{-}\right) - Q(s_j, a_j; \theta_i) \tag{2}$$

$$L_i(\delta_i) = \begin{cases} \dfrac{1}{2}\delta_i^2 & \text{if } |\delta_i| < 1 \\ |\delta_i| - \dfrac{1}{2} & \text{otherwise} \end{cases} \tag{3}$$

in which r is the reward, γ is the discount factor, a' is an action among candidate actions $\mathcal{A}$ at state s_{j+1}, θ_i are the network parameters of the learning network Q at iteration i, and θ_i^{-} are the network parameters of the target network $\hat{Q}$ which is used to compute the target $y_j = r_j + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s_{j+1}, a'; \theta_i^{-})$ at iteration i.

After the learning of action-value function based on this approach during C steps, the target network parameters θ^{-} are updated. The loss function is minimized by using RMSPropGraves [9] which is a kind of stochastic gradient descent methods. Figure 1

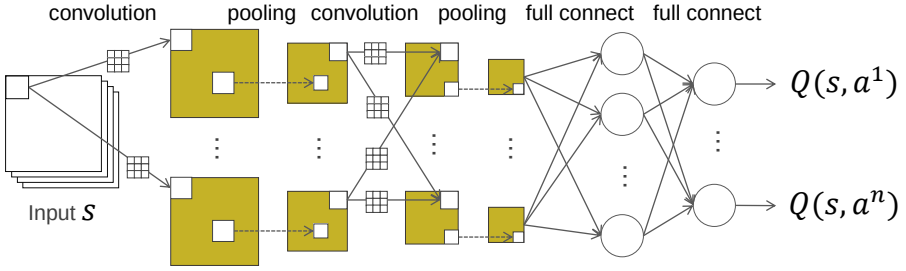


FIGURE 1. Structure of DQN ($\mathcal{A} = \{a^1, a^2, \ldots, a^n\}$)

shows the structure of DQN. DQN computes each Q-value $Q(s, a)$ from the input (specific state) s for all candidate actions $\mathcal{A} = \{a^1, a^2, \ldots, a^n\}$.

2.2. **DQN method for multi-robot system.** In this research, we apply DQN to multi-robot system where three mobile robots exist in the same environment. Figure 2 illustrates our framework to apply DQN to the multi-robot system.

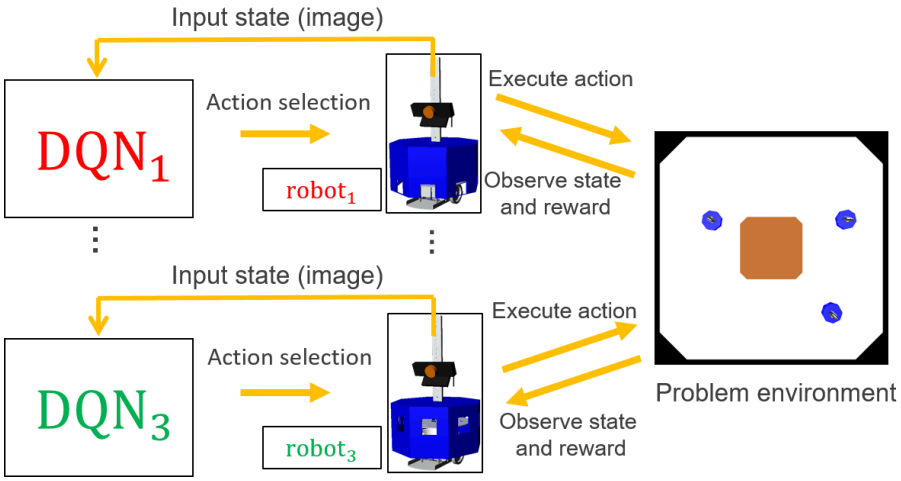


FIGURE 2. Application method of DQN to multi-robot system

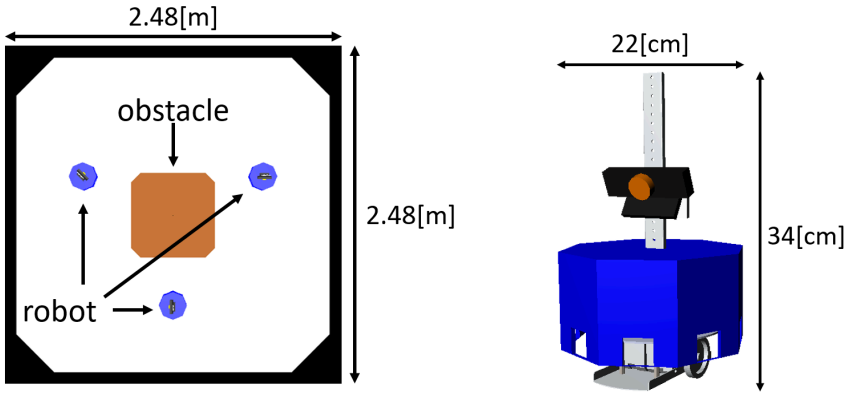
In our method, each robot has its own DQN. Each robot selects its own action independently without inter-robot communication which gives information about the observed state and executed action of each robot. The inputs of each DQN are camera images of each robot and the output of each DQN is action-value function (Q-value) for candidate actions of each robot. All robots observe their states and execute their actions synchronously at the same timing.

2.3. **Curriculum learning.** In multi-robot system, the optimal action of each robot depends not only on its own state but also on the states and actions of other robots. As all robots perform behavior learning concurrently in our framework, the action policy of each robot always changes. Therefore, our problem is quite difficult and it might cause unstable learning. In order to realize acceleration and stabilization of learning in multi-robot system, we apply the curriculum learning, where the task difficulty level is gradually raised.

As the first stage, learning of DQN is performed in the environment where only a single robot exists. The problem environment is same as for multi-robot system except the number of robots. Then, other two robots learn their DQNs while the first robot stops learning and selects the action using the parameter values of DQN which were acquired through the initial learning. By this curriculum learning, it is expected to improve the stability of learning, because the change of action policies of the robots is mitigated.

3. Target Problem.

3.1. **Problem setting.** The objective of this research is to realize that multiple mobile robots acquire behavior to avoid walls and other robots by using DQN in simulation environment. Our simulation environment is illustrated in Figure 3, which is constructed by using Cyberbotic robot simulator Webots. We have already manufactured three real mobile robots and problem environment in real environment. Three mobile robots have same size with diameter of 22[cm] and height of 34[cm]. The simulation model was constructed in exactly the same size as this real robot.



(a) Problem environment (from top veiw) (b) Wheeled mobile robot

FIGURE 3. Problem environment in simulation and wheeled mobile robot

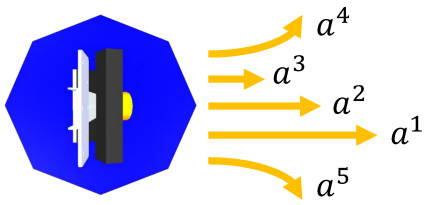
Each robot decides the action using only the image of the camera mounted on the robot. We use the wide view camera with 120[deg] view angle just same as in the real robot. The camera images are translated into three-valued images. The three-value images are generated by converting the road region to white (pixel value 255), the wall region to black (pixel value 0), and robot body region to gray (pixel value 127).

The robot also has distance sensors with eight directions, which enable to measure the distance between robots and between the robot and the wall. The measured values of distance sensors are used only for calculation of the rewards. Their values are not used for the inputs of DQNs. In the simulation environment, three mobile robots exist. Each robot executes its own action independently and synchronously.

3.2. **Definition of action and reward.** The robot takes an action a_t at each time step t among the five candidate actions as shown in Figure 4. Here, by each action, the robot moves the following distance. By the action "go straight (high speed)", the robot moves forward 4.8[cm]. Similarly, the robot moves forward 3.2[cm] by the action "go straight (middle speed)", and it moves 1.6[cm] by the action "go straight (low speed)". By the action "turn right", the robot moves forward 2.7[cm] and moves to the left 0.12[cm]. After this action, the robot turns around 5.2[deg] to the clockwise direction. The action "turn left" is opposite move of the action "turn right".

$$r = \begin{cases} 15 & \text{in case A} \\ 5 & \text{in case B} \\ 10 & \text{in case C} \\ -20 & \text{in case D} \\ 0 & \text{in otherwise.} \end{cases}$$

Here, case A indicates the case when robot is far away from the wall and other robots and robot selects the action "go straight (high speed)". Case B means the case when robot is far away from the wall and other robots and robot selects the action "go straight (low speed)". Case C means the case when robot is far away from the wall and other robots and robot selects the action except the above actions. Case D shows the case when robot collides to the wall or other robots. It is expected that each robot selects the action "go straight (high speed)" in many situations so as to get more and more reward. It is also expected that the robot selects the action "go straight (low speed)" when it gets close to the other robots.



$$\mathcal{A} = \begin{cases} a^1\text{: go straight (high speed)} \\ a^2\text{: go straight (middle speed)} \\ a^3\text{: go straight (low speed)} \\ a^4\text{: turn left} \\ a^5\text{: turn right} \end{cases}$$

FIGURE 4. Candidate actions of each robot (from top veiw)

4. Simulation Experiment.

4.1. **Experimental method.** In our target problem, three mobile robots exist in problem environment. We set the moving speed of one of the three robots to half of other two robots. Except the moving speed of robots, all settings such as the kinds of candidate actions and reward function are same. All robots move in clockwise in the problem environment. It is expected that the fast two robots acquire behavior to overtake the slow robot. When one of the three robots collides to walls or other robots, all robots stop at once and the robot with collision executes the restart action by which the robot moves backward and rotates in clockwise or in counter-clockwise. This restart action is designed so as to execute the action by using only images of the camera mounted of the robot in real robot environment.

Inputs for DQN are pair of current image and one step previous image of the camera mounted on the robot. The camera images are three-values as explained in Section 3.1 and their image size is set to 48×27[pixel] in order to make learning and calculation easier. DQN has two convolutional and pooling layers. Convolutional layers have 32 weight filters with size of 8×8 and with stride of 1. Pooling layers execute 2×2 Max-pooling. We decided the structure of CNN based on the preliminary experiments.

We adopt ϵ-greedy method as action selection method. In ϵ-greedy method, a robot selects an action randomly with small probability ϵ, otherwise selects an action with the highest action value. In this research, value of ϵ linearly decreases from 0.1 to 0 during 12,500 steps.

We apply Profit Sharing method [10] to DQN for the acceleration of leaning. Rewards are updated by the following equation when a robot collides to walls or other robots,

$$r_{t-i} \leftarrow r_{t-i} + \beta^i r_t \quad (i = 1 \sim K) \tag{4}$$

where β is the damping factor ($0 < \beta < 1$), r_t is the terminal reward at time step t, K is the maximum number of steps to distribute the reward at once. In this experiment, we set the value K to 5 and the value β to 0.7.

During the initial 500 steps, the robots only collect experiences data to use in experience replay. The size of replay memory is 100,000 and minibatch size is 32.

After the initial 500 steps, we evaluate the learning performance with 10 test runs every 500 steps of learning. In test runs, the starting points of the robots are selected among the three positions as shown in Figure 5. In this test run phase, each robot selects an action with the highest action value, using the greedy method. Each test run finished when a robot collides to wall or other robots, or 500 steps have passed without any collisions.

4.2. **Experimental results.** We executed 10 sets of experiment without curriculum learning. Figure 6 shows the change of average number of collisions at every 500 steps. The number of collisions with walls decreased at the beginning of learning, and the number of collisions with other robots decreased after the middle stage of learning. These

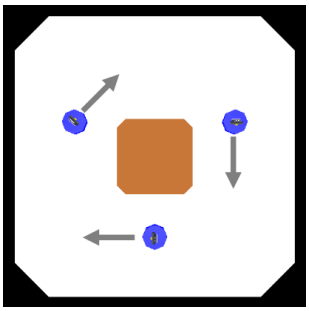


FIGURE 5. Starting points and directions of three robots in test run phase (evaluation phase)

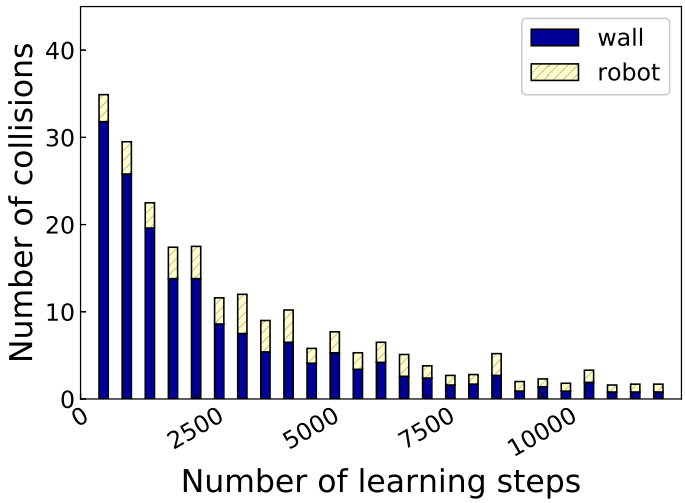


FIGURE 6. Change of number of collisions at every 500 steps in learning phase (training phase)

results suggested that our robots acquired behavior to avoid walls at first, and then acquired behavior to avoid other robots. In the later stage of learning, the main causes of collisions to walls are actions to avoid other robots.

Figure 7 and Figure 8 show examples of actions after learning. In Figure 7, the fast robot B executed actions to change the course and overtake the slow robot C. Figure 8 shows a collision example in a state in which the three robots are very close. In our definition of reward function, the reward which a robot receives is not affected by the rewards which other robots get. Hence, it is difficult to acquire actions perfectly to avoid collisions between other robots without inter-robot communication.

Figure 9 shows the comparison of with and without curriculum learning. In curriculum learning of this research, one slow robot learns at first by itself. After that, the slow robot stops learning DQN and other two fast robots start learning. After 6000 steps have passed since the two fast robots start learning, the slow robot restarts learning. We have two reasons why the slow robot learns at first. First, the slow robot takes more and more action steps until it collides to the wall than the fast robots. It means the slow robot takes long time to acquire behavior to avoid the wall. Second, in this problem setting, overtaken robots cannot detect overtaking robots because there is no rear camera on the robot. It is necessary for fast robots to acquire behavior according to the action of the slow robot in front of them, therefore it is desirable that the slow robot's policy does not change. From Figure 9, we can see that curriculum learning accelerates behavior learning. We think it is possible to realize stable learning by this curriculum leaning. The search

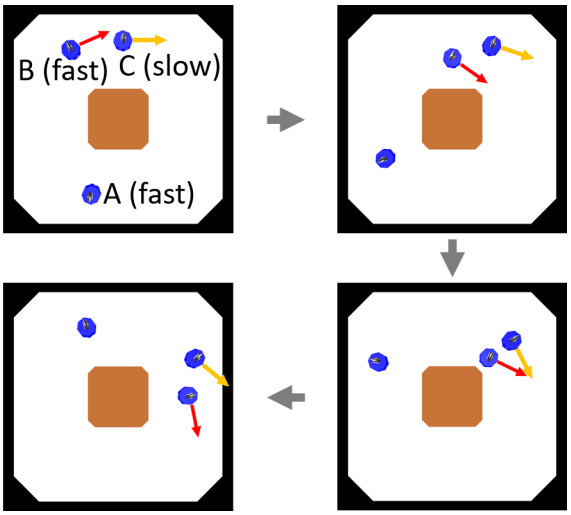


FIGURE 7. An example of overtaking behavior after learning (drawing at every 20 action steps)

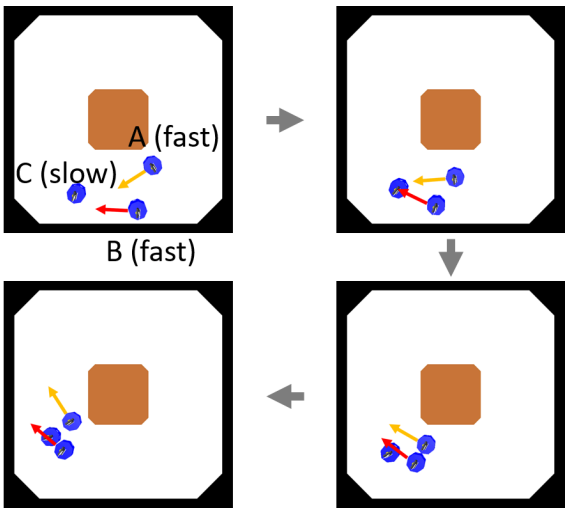


FIGURE 8. An example of collision between robots after learning (drawing at every 10 action steps)

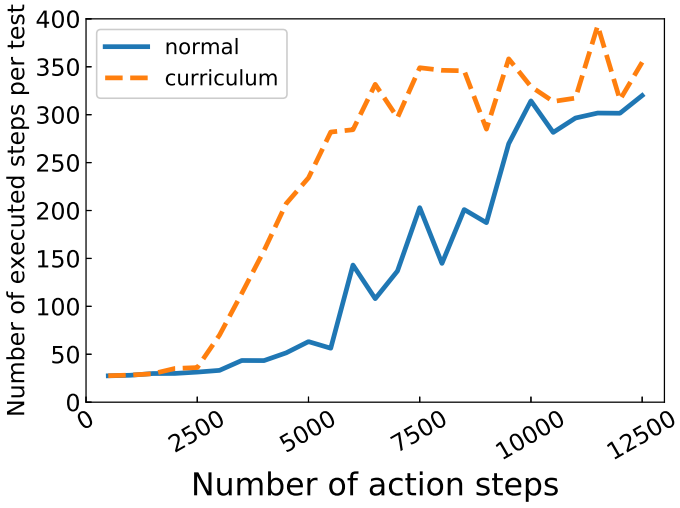


FIGURE 9. Comparison of learning performance with and without curriculum learning

space of action policies of fast robots gets narrow because the slow robot does not change its action policy.

5. **Conclusions.** In this research, we applied Deep Q-Network (DQN) to behavior acquisition in multi-robot environment. We realized that each robot acquired behavior to avoid the wall and other robots using only the camera mounted on the robot.

- Each robot reduces the number of collisions to walls or other robots using DQN method in simulation environment where all robots rotate in same direction.
- We set the moving speed of one of the three robots to half that of other two robots. The fast two robots acquired behavior to overtake the slow robot.
- We applied curriculum learning framework to DQN. At first, only slow robot learned, and then the slow robot stopped learning and other two fast robots started learning. By this method, we could expect that the search space of action policies became narrow and learning became stable.

We aim to realize that robots move and work in environments where humans live. In the environment we used, the wall and road have simple shape and are separated by certain colors (black, brown and white). We plan to make the environment more realistic and apply methods to reducing the complexity of the environment.

REFERENCES

[1] V. Mnih et al., Human-level control through deep reinforcement learning, *Nature*, vol.518, pp.529-533, 2015.
[2] D. Silver et al., Mastering the game of Go with deep neural networks and tree search, *Nature*, vol.529, pp.484-489, 2016.
[3] Y. LeCun et al., Gradient-based learning applied to document recognition, *Proc. of the IEEE*, vol.86, no.11, pp.2278-2324, 1998.
[4] H. Sasaki, T. Horiuchi and S. Kato, Experimental study on behavior acquisition of mobile robot by deep Q-network, *J. of Advanced Computational Intelligence and Intelligent Informatics*, vol.21, no.5, pp.840-848, 2017.
[5] D. Kato, H. Sasaki, T. Horiuchi and T. Aodai, A study on vision-based behavior learning of real mobile robot by deep Q-network, *Proc. of the SICE Annual Conference 2018*, pp.1035-1038, 2018.
[6] Y. Maruyama et al., Autonomous movement acquisition by deep reinforcement learning using semantic segmentation, *Proc. of the 35th Annual Conference of the RSJ*, 2017 (in Japanese).
[7] https://research.preferred.jp/2015/06/distributed-deep-reinforcement-learning/, Accessed on November 20, 2019.
[8] C. J. Watkins and P. Dayan, Technical note: *Q*-learning, *Machine Learning*, vol.8, pp.279-292, 1992.
[9] A. Graves, Generating sequences with recurrent neural networks, *arXiv*, arXiv:1308.0850, 2013.
[10] K. Miyazaki, H. Kimura and S. Kobayashi, Theory and applications of reinforcement learning based on profit sharing, *Trans. of the Japanese Society for Artificial Intelligence*, vol.14, no.5, pp.800-807, 1999 (in Japanese).