Autonomous Learning of Reward Distribution for Each Agent in Multi-Agent Reinforcement Learning

Katsunari SHIBATA† and Koji ITO††
† Dept. of Electrical & Electronic Engineering, Oita Univ., Oita 870-1192, Japan,
†† Dept. of Comp. Intel. & Sys. Sci., Tokyo Inst. of Tech., Yokohama 226-8502, Japan
† shibata@cc.oita-u.ac.jp, ††ito@ito.dis.titech.ac.jp

Abstract. A novel approach for the reward distribution in multi-agent reinforcement learning is proposed. The agent who gets a reward gives a part of it to the other agents. If an agent gives a part of its own reward to the other ones, they may help the agent to get more reward. There may be some cases in which the agent gets more reward than that it gave to the other ones. In this case, it is better for the agent to give the part of the reward to the other ones. Based on this principle, each agent learns the reward distribution ratio to the other agents autonomously based on the selfish value function. Some simulations have been demonstrated that a rational reward distribution ratio is obtained by each agent depending on the given task.

1. Introduction

In multi-agent systems, the avoidance of the conflicts and the formation of the cooperative behaviors among the agents are both important subjects. In some tasks, such behaviors can be formed by reinforcement learning based on the agent's own reward. For example, let us suppose that a conflict happens between two agents, and both agents cannot get any reward. If both agents can get some rewards even though one agent gives its way, it chooses to give its way rather than the conflict state[1]. However, in the tasks with more serious conflict, such that when one agent gets the reward, the other cannot get any rewards, the "giving its way" behavior cannot be formed only by its own reward.

In order to realize the cooperative behaviors in such tasks, there exists a popular method in which the rewards of each agent are distributed to the other agents. However, it has the problem how are the rewards distributed. The easiest way is to distribute all the rewards uniformly to all the agents including the agent itself like [2]. In this case, however, even if there is an agent who disturbed the task accomplishment, it also obtains the reward. Accordingly the disturbing behavior is reinforced temporally. Furthermore, the reward of each agent, who tried to accomplish the task, becomes small relatively. Then the learning is irrational and slow in general. The other method is that a part of the reward is given to the agent who gets the reward and the remaining is distributed uniformly to all the other agents[3]. In this case, it is difficult to decide the distribution ratio in advance because the ideal ratio may depend on the task, and further it is still irrational that all the other agents get the reward uniformly not depending on their behaviors.

We think that these problems are caused by the top-down approach in which the reward distribution ratio is given to the agents in advance. Reinforcement learning is a very autonomous learning, and is useful especially when the environment is unknown. So

the top-down approach may spoil the advantage of the reinforcement learning. When we look back at ourselves, we give some reward to the persons who help us, and expect indirectly to help us again in the next chance. Then a novel approach inspired by our human reward distribution is introduced here. The reward distribution ratio is obtained autonomously by each agent through its experiences. And the value function, which each agent utilizes to learn the ratio, includes only "selfish" value and no "altruistic" value.

2. Classification of Conflicts

Here it is introduced that the conflicts can be classified according to its seriousness. The seriousness is measured by how much reward decreases for the agent by giving its way to the other one to avoid the conflict. Simply writing, if the reward is larger when it gives its way than when it is in the conflict, the conflict is called "weak", otherwise, it is called "strong".

At first, the value function at the time t is written as $V(t)$. According to the conventional way, the state of the agent can be evaluated by the value function as

$$V(t) = \sum_{i=1}^{\infty} \gamma^{i-1} r(t+i). \quad (1)$$

Conflict states can be written as

$$V_{no\_opp} > V_{conflict} \quad (2)$$

where V_{no_opp} indicates the state evaluation value when no opponent agent exist, and $V_{conflict}$ indicates the value when the conflict happens, and no agent gives its way. Further $V_{altruistic}$ is supposed to be the value when the agent gives its way to the other, and $V_{selfish}$ is supposed to be the value when the opponent agent gives its way. In the weak conflict, the reward of the agent, who changes its action to avoid the conflict, does not decrease in comparison with the selfish one's as

$$V_{no\_opp} \geq V_{selfish} = V_{altruistic} > V_{conflict}. \quad (3)$$

In the second case, the agent, who changes its action, cannot get the same reward as the selfish agent, but gets more reward than in the case of the conflict as

$$V_{no\_opp} \geq V_{selfish} > V_{altruistic} > V_{conflict} \quad (4)$$

It is called a medium conflict here. In the last case, if one agent gives its way, it does not get more reward than in the conflict as

$$V_{no\_opp} \geq V_{selfish} > V_{conflict} \geq V_{alturistic}. \quad (5)$$

It is called a strong conflict here. When there is only one target, and only the first-come agent can get it, this problem belongs to this category. In the case of the weak conflict or medium conflict, each agent can acquire the behavior to avoid the conflict by reinforcement learning in each agent, because it can get more reward when it gives its way than in the conflict. While, in the case of the strong conflict, the reward distribution between the agents is required for the agents to acquire such behaviors by reinforcement learning in each agent. However, the condition as

$$V_{selfish} + V_{alturistic} > 2V_{conflict} \quad (6)$$

is required. In this paper, the strong conflict is focused, and the autonomous acquisition of the reward distribution is proposed.

3. Principle and Value Function

Though the reward distribution seems "altruistic" at a glance, we think that it emerges only by the "selfish" value function. Here one-reward task, which finishes when one agent reaches its goal, is employed as an example of strong conflicts. Suppose that the goal of the agent A is located behind the agent B and the goal of the agent B is located behind the agent A, and the conflict happens as shown in Fig. 1 (a). In this case, both agents get no reward unless one of them goes to the opposite direction to its goal. However, if the agent A gives a part of the reward to the agent B, the agent B learns to give its way and to get the part of the reward as shown in Fig. 1 (b). Since the agent A becomes to be able to get the reward by the reward distribution, it can learn the distribution. Even if the agent A gives a part of its reward to the agent C, the performance of the agent A does not change so much. So it does not learn the distribution. This can be thought of a kind of negotiation.

Precisely, not only the reward but also the time steps until the goal should be considered to learn the reward distribution. The value function R for learning the reward distribution is designed so as to keep the consistency with Q-learning, which is formulated as Eq. (1) becomes the maximum. It is calculated from the total rewards *total_rein* and total time steps *total_step* in some fixed number of trials (here 100 is employed) as

$$R = total\_rein\ \gamma^{total\_step}, \tag{7}$$

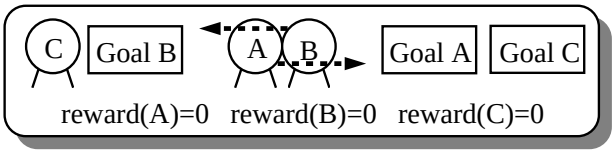
where

$$total\_rein = \sum_{n=1}^{100} \{(1 - dist_{self})\ r_{self}[n] + dist_{opp}\ r_{opp}[n]\}, \tag{8}$$

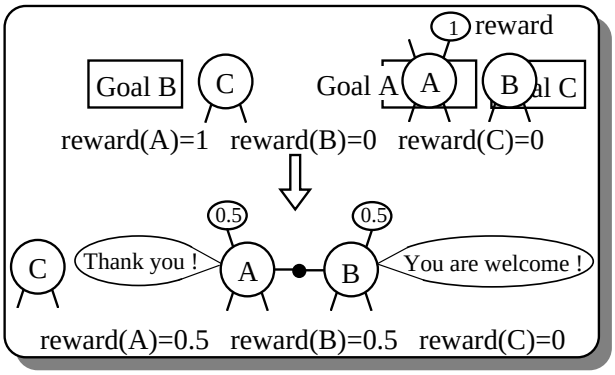
$$total\_step = \sum_{n=1}^{100} step(n), \tag{9}$$

where $dist_i$: the ratio of the reward given to the other agent (opponent) when the agent i reaches its goal, r_i: the reward which the agent i gets, *opp*: the suffix that indicates the opponent, γ: a discount factor.

A random number is added as $\Delta dist$ to the distribution ratio $dist$ in the range of $0.0 < dist + \Delta dist < 1.0$ before the beginning of the 100 trials. After the trials, the present R is compared with the previous one. If the present one is larger than the previous one,



(a) conflict state



(b) when the reward is distributed

Fig. 1 The principle of the proposed reward distribution learning.

$dist = dist + \Delta dist$. If the previous one is larger, the ratio is not changed.

When the number of agents is more than 3, $dist_{ji}$ is defined as the reward distribution ratio from the agent j to the agent i including $i=j$. So there is the relation as

$$\sum_{i=1}^{N} dist_{ji} = 1.0, \tag{10}$$

where N: the number of the agents. Then $\Delta dist_{ji}$ is calculated using random numbers rnd_{ji} as

$$\Delta dist_{ji} = rnd_{ji} - \frac{\sum_{i=1}^{N} rnd_{ji}}{N}. \tag{11}$$

If $(dist+\Delta dist)<0.0$ or $(dist+\Delta dist)>1.0$, the difference from 0.0 or 1.0 is distributed to the other $\Delta dist_{ji}$ uniformly.

At the beginning, the problem in which a task is iterated with two or some fixed members, is adopted here as a simple case, and the reward distribution ratio to each of the other agents is trained in each agent.

4. Formulation of Multi-agent Q-learning

Each agent is supposed to move sequentially in a fixed order. One trial finishes when one of the agents reaches its goal, and a conflict happens between the two agents. They learn their actions based on one-step Q-learning. Here the learning of two-agent case is explained as an example. The state transition is represented as shown in Fig. 2, where $a(t)$: the action of an agent, $a_{opp}(t)$: the action of the other agent, $s(t)$: the state. The state transition is supposed to be deterministic. At first, the value of the state $V(s(t+1))$ after the agent's own action $a(t)$ is calculated as Eq.(6), considering the both cases, when the other agent reaches its goal or not just after the state $s(t+1)$.

$$V(s(t+1)) = P_{oppgoal}(s(t+1))\, \check{r}_{opp}(s(t+1)) + (1 - P_{oppgoal}(s(t+1)))\, \gamma\, maxQ(s(t+1)), \tag{12}$$

where

$$P_{oppgoal}(s(t+1)) \leftarrow (1-\alpha)\, P_{oppgoal}(s(t+1)) + \alpha \quad \textit{if the opponent reaches its goal}$$
$$\leftarrow (1-\alpha)\, P_{oppgoal}(s(t+1)) \quad \textit{otherwise}, \tag{13}$$

$$\check{r}_{opp}(s(t+1)) \leftarrow (1-\alpha)\, \check{r}_{opp}(s(t+1)) + \alpha\, dist_{opp}\, r_{opp}(t) \quad \textit{if the opponent reaches its goal}, \tag{14}$$

$$maxQ(s(t+1)) \leftarrow (1-\alpha)\, maxQ(s(t+1)) + \alpha \max_i(Q(s(t+2), a_i)), \tag{15}$$

where $P_{oppgoal}(s)$: the probability that the opponent reaches its goal just after the state s, $\check{r}_{opp}(s)$: the expected reward given from the opponent when it reaches its goal, $maxQ(s(t+1))$: the expectation of the maximum Q-value at the time $t+2$, α: a learning rate$(0<\alpha<1)$. The reason why $maxQ$ is calculated instead of using the actual maximum Q-value is that the

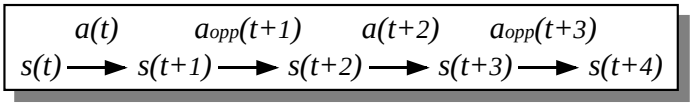


Fig. 2 State transition in two-agent case.

state at the time t+2 changes depending on the opponent's action. Then Q-value is trained by the following equation using $V(s(t+1))$ as

$$Q(s(t),a(t)) \leftarrow (1-\alpha)\, Q(s(t),a(t)) + \alpha\, \{(1-dist_{self})\, r(s(t),a(t)) + \gamma\, V(s(t+1))\}. \quad (16)$$

The state transition by the opponent's action is also supposed to be one step, and if the opponent does not reaches its goal, the ideal Q-value has the relation as $Q(t) = \gamma^2\, Q(t+2)$. If the obtained reward is always the same value and every agent distributes its reward uniformly to the opponent and itself, it is better for the agent that the opponent reaches its goal at time t+1 than that the agent itself reaches its goal at time t+2.

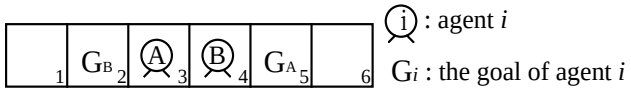


Fig. 3 A simple environment with the conflict.

5. Simulation

5.1 Two agent Simulation

In order to validate the proposed learning method, some simple simulations were done. At first, a simple environment with two agents, A and B, and two goals, G_A and G_B was assumed as Fig. 3. Here, learning rate α=0.1, discount factor $\gamma=\sqrt{0.9}$, and reward r=1.0. The agent's action can be one of "stay", "go to the right", "go to the left". The agent selects its action according to the Boltzman distribution. The temperature is reduced gradually from 1.0 to 0.01 as shown in Fig. 4. The distribution ratio is added by a random number between ±0.05 until 4000 iterations, between ±0.02 until 10000 iterations, between ±0.005 otherwise.

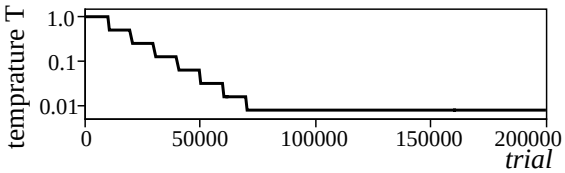


Fig. 4 Change of the temperature

At first, after the learning of each agent's actions with some fixed reward distribution ratios for the agent A $dist_A$, the value function R was observed. The $dist_B$ was fixed at 1.0. Fig. 5 shows the result. In all the cases, the agent A leans the action to its goal except for the case of $dist_A$=1.0. It can be seen that the maximum R is obtained in the case that $dist_A$ is around 0.2 for the agent A. Since the $dist_A$ is not 0.0, it is known that the distribution of the obtained reward is useful for the agent, and the agent can learn the

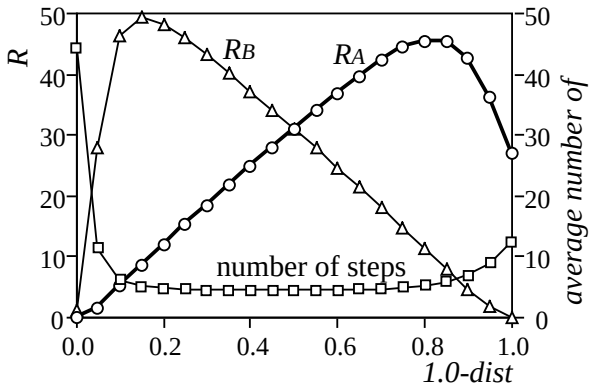
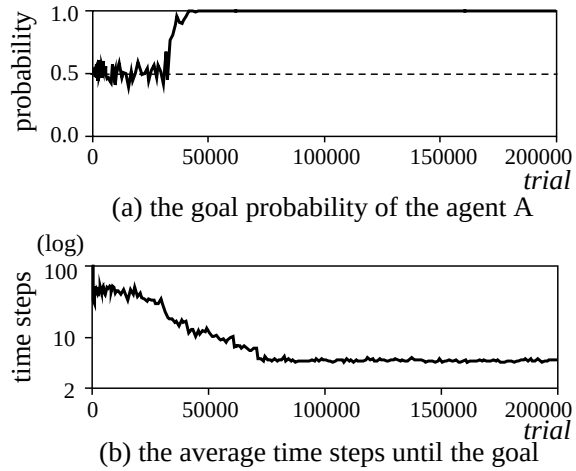


Fig. 5 The value function R as a function of *dist*



(a) the goal probability of the agent A

(b) the average time steps until the goal

Fig. 6 The change of the goal probability of the agent A and the average time steps and average times steps when *dist* changes.

reward distribution based on the "selfish" value function.

When the both *dist*s are trained, one of the agents becomes to go its goal and the average time steps until its goal is reduced as shown in Fig 6. And the *dist* of the agent who gets the reward converges not depending on the initial value as shown in Fig. 7. The maximum *dist* of the agent who gets the reward is 0.313 and the minimum is 0.238 during 100 simulation runs. It is close to 0.2, but slightly larger than 0.2. The reason is as follows. If *dist* is fixed for many trials, the value function *R* becomes as Fig. 5, but just after *dist* changes when *dist* is around 0.25, *R* changes the other direction as shown in Fig. 8. That is because when the distributed reward is reduced, the opponent agent feels it as a penalty and the time steps becomes large.

Next, the asymmetrical environments were introduced as shown in Fig. 9, and the probability that the agent B reaches its goal over some simulation runs, was observed as shown in Fig. 10. It can be seen that the probability becomes larger when the distance from the agent A to its goal becomes farther. That is rational because when the distance is far, it is better for the agent A to give its way to the agent B and get the distributed reward from the agent A.

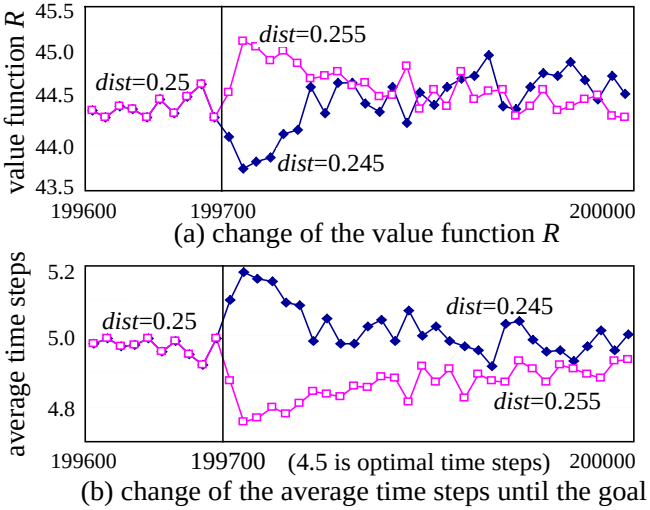


Fig. 8 The change of the value function *R*

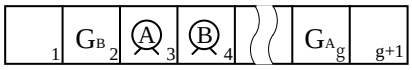


Fig. 9 An asymmetrical environment

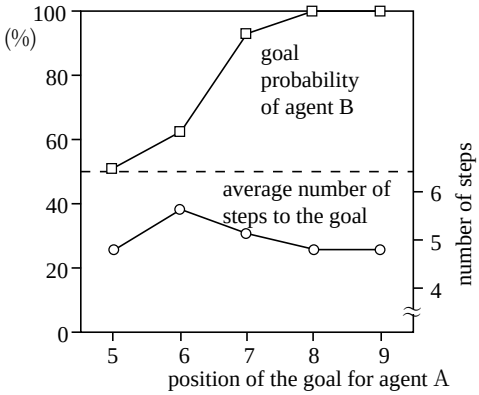


Fig. 10 Probability of the agent B's goal as a function of the goal position for the agent A.

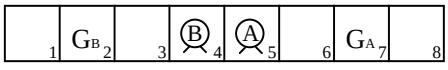


Fig. 11 An environment with no conflicts

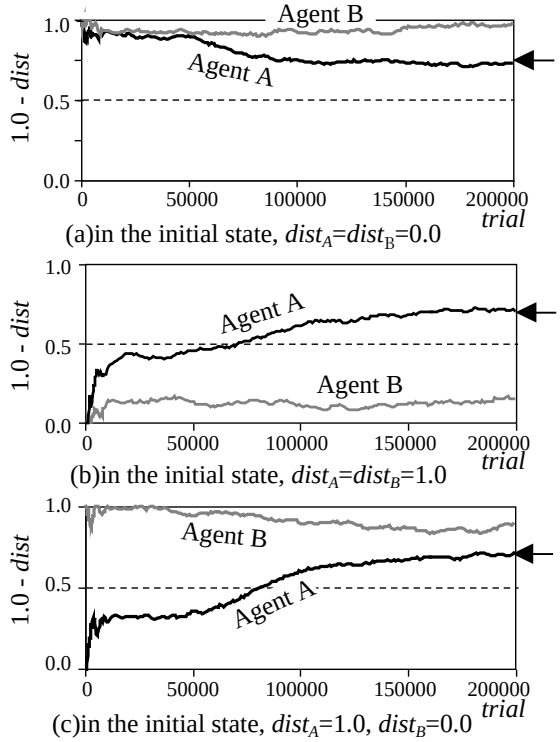


Fig. 7 The change of *dist* in some cases with different initial values.

Finally, a simulation was done on the environment with no conflicts as shown in Fig. 11. In this case, *dist* of the both agents becomes around 0.0, and it looks also a rational result.

5.2 Three Agent Simulation

In this subsection, it is examined whether the proposed algorithm works in the case of three agents. The employed task is shown in Fig. 12. The main different points from the previous task are the number of the agents and the environment. Since the environment is a circle and has no dead end, each agent can reach its goal in either direction. However, the agent cannot reach its goal without the other agents' cooperation. The discount factor γ=0.96, and the reward r=1.5. One simulation run consists of 500000 trials. The temperature changes from 1.0 to 0.01 continuously during the first half of the learning, and is constant during the latter half. The range of the random number that is added to the distribution ration also reduced continuously from ±0.05 to ±0.005 during whole the simulation. 100 simulation runs are executed. The initial distribution ration is 1.0 for itself 0.0 for the others. The minimum number of steps which one of the agents reaches its goal is 7. In this case, the first move agent has to reach its goal. But actually the number of steps cannot be 7.0 due to the stochastic action selection.

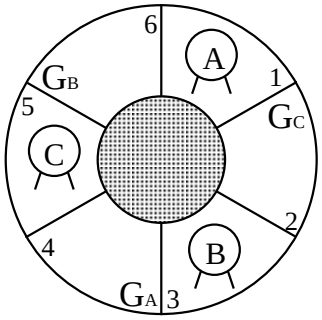


Fig. 12 Three agent simulation

When the average number of steps during last 1000 trials in each simulation was observed, the maximum was 11.41, the minimum was 7.26, and the average was 8.45 in 100 simulation runs. The average distribution ratio of the agent who reached its goal over 100 simulation runs was 0.60 for itself (in almost all cases within ±0.1), 0.21 for the next agent in the direction that the agent moves more often, and 0.19 for the other agent. From this result, it is seen that the agent learns to distribute its reward to the others. It was interesting that in many cases, the ratio to the agent in the moving direction was larger than that to the other. In 73 simulation runs, one agent always reached its goal not depending on who moves first (case 1). In 14 runs, one agent reached its goal in about two thirds of the trials, and another agent reached its goal in about one third (case 2). In 11 runs, each agent reached its goal when it moves first (case 3). In the other two runs, the situation is transition state when the simulation finished. Generally, the learning is not so stable as the case of two-agent case. Furthermore, it was sometimes observed in the case 1 or 2, that one agent disturbed another agent's goal, and that resulted in the other agent's goal. For example, even if the agent B came to the location 2, the agent A did not move until the agent C came to the location 3. Then agent C always reached its goal not depending on the moving order. It is interesting that for the disturbing agent, the distributed ratio from the goal agent was larger than that from the disturbed agent.

When the distribution ratio is fixed to 1/3 for all the agents including the goal agent itself, the maximum number of steps is 7.54, the minimum is 7.39, and the average is 7.46. In all runs, the agent who moves first reached its goal, and the learning is more stable. On the other way, when the distribution ration is fixed to 1.0 for itself and 0.0 for the others, the first move agent reached its goal, but the average is 101.7 and even the minimum is 78.39. Finally when one agent is removed from the circle, the distribution ratio for the agent becomes almost 0.0. From these results, by the learning of the reward distribution ratio, the optimal solution for the system cannot be obtained, but rational solution can be obtained without any advance ideas.

6. Discussion

In the game theory, the conflict avoidance problems have been discussed such as the prisoner's dilemma problem. This task can be taken as one of the non-cooperative non-zero-sum games, but there are two main differences between this task and the game. The first one is that in the game theory, the gain matrix is fixed and the possibility to converge the rational solution is discussed. On the other way, the proposed approach corresponds to the autonomous modification of gain matrix by each agent under the condition that the sum of the reward is constant. The second point is that in this research, time makes an important role. The agent has to do some actions until it arrives at its goal. Further, the number of time steps until its goal is reflected to the evaluation.

7. Conclusion

A novel approach for the autonomous reward distribution in multi-agent systems has been introduced. Each agent learns the reward distribution ratio to the other agents autonomously based on the selfish value function through some iterations of the task, while in each trial of the task, it learns its action by Q-learning.

The simulations with two agents including a strong conflict have demonstrated the followings. (1) Actually there exists a case in which the agent can learn the reward distribution from the selfish value function. (2) In the environment with some conflicts, the agent who gets the reward, learns to give a part of the reward to the other agent through the learning. (3) In the asymmetrical environment, the agent who can reach its goal faster, learns the action to the goal, and the other learns the action to give its way. (4) In the environment with no conflicts, both agent does not give its reward to the other. All the results seem rational.

Through the simulation with three agents, it is known that the proposed algorithm can be utilized, and some interesting strategies of each agent can be found.

8. Problems and Future Works

The problem of this algorithm is the slow learning. The reason of the slow learning is that the learning is based on trial-and-errors, and each agent cannot evaluate the distribution ratio even though it experiences many trials. This problem becomes more serious when the number of the agents becomes large. One of the ideas is that the agent decides the ratio in each trial by observing the other agents' behaviors in the trial.

Acknowledgement

A part of this research was supported by the Scientific Research Foundation of the Ministry of Education, Science, Sports and Culture of Japan (#10450165) and by Research for the Future Program by The Japan Society for the Promotion of Science (JSPS-RFTF96I00105).

Reference

[1] Shibata, K. and Ito, K., " Emergence of Individuality and Sociality by Reinforcement Learning in Multi-Agent Systems ", Proc. of AROB(Int'l Sympo. on Artificial Life and Robotics) 5th '00, Vol 2, pp.589-592 (2000)

[2] Ono, N. and Fukumoto, K., "Multi-agent Reinforcement Learning: A Modular Approach", Proc. of ICMAS-96, pp.252-258 (1996)

[3] Shirakawa, H., Kimura, H. and Kobayashi, S., "Experimental study on Emergence of cooperative action using Reinforcement Learning", Proc. of the 5-th Intelligent Systems Sympo., pp.119-124 (1998) in Japanese