Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Generation of Roles in Reinforcement Learning Considering Redistribution of Reward between Agents

Masayuki Nakahara
Graduate School of Bionics, Computer and Media Sciences
Tokyo University of Technology
Hachioji, Tokyo

Yuko Osana
School of Computer Science
Tokyo University of Technology
Hachioji, Tokyo
osana@cs.teu.ac.jp

***Abstract*—In this paper, we propose a method for generating roles by redistribution of reward between agents in reinforcement learning of multiagent system. In the proposed method, there are two kinds of reward; (1) reward obtained from environment when agents reach the goal and (2) reward received from other agents. Agents can learn actions to work cooperatively with other agents by the reward received from other agents, and the roles of agents are generated. In the proposed method, agents can decide how much reward to which agent to give from the obtained reward by using the history that records actions at the time when the change of environment are recognized. We carried out computer experiments for two tasks; (1) path finding problem and (2) transportation problem, and confirmed that roles of agents are generated by redistribution of reward between agents in the proposed method. Moreover, we confirmed that the proposed method can learn as similar as when the designer decides conditions for redistribution of reward.**

***Index Terms*—Generation of Roles, Redistribution of Reward, Multiagent System, Reinforcement Learning**

I. Introduction

Recently, much research on multiagent system is carried out as one of the distributed and cooperative systems. Multiagent system is a system that solves tasks by plural agents, and it has possibilities that can achieve tasks efficiently by cooperation of multiagent. In multiagent systems, cooperation among agents can be realized by the interaction such as communication between agents, division of roles and so on.

Much research on cooperation among agents has been carried out. For example, in refs.[1] and [2], each agent learns the own working area by reward distribution. However, in these methods, the condition that the agent gives the reward to the other agents is decided beforehand. When division of roles are realized in multiagent systems, agents have to learn own role by themselves. Since the actions that each agent should take are usually unknown, reinforcement learning is suitable.

In this paper, we propose the method for generating roles by redistribution of reward between agents in reinforcement learning of multiagent system. In the proposed method, there are two kinds of reward; (1) reward obtained from environment when agents reach the goal and (2) reward received from other agents. Agents can learn actions to work cooperatively with other agents by the reward received from other agents, and the roles of agents are generated. In the proposed method, agents can decide how much reward to which agent to give from the obtained reward by using the history that records actions at the time when the change of environment are recognized.

II. Reinforcement Learning Considering Redistribution of Reward between Agents

Here, we explain the proposed reinforcement learning considering redistribution of reward between agents.

A. Outline

In the proposed method, the roles are generated by redistribution of reward between agents. There are two kinds of reward; (1) reward obtained from environment when agents reach the goal and (2) reward received from other agents. Agents can learn actions to work cooperatively with other agents by the reward received from other agents, and the roles of agents are generated. In the proposed method, agents can decide how much reward to which agent to give from the obtained reward by using the history that records actions at the time when the change of environment are recognized.

B. Flow of Proposed Method

In the proposed method,

- action of the agent x at the nth state change ($H^A_{n,x}$)
- time of the nth state change ($H^{T_s}_n$)
- last time when one of the agents receives positive reward from environment before the time $H^{T_s}_n$ ($H^{T_r}_n$)

are used as the history.

(1) Initialization

The environment and the history lists are initialized.

$$t = 0 \tag{1}$$

$$n = 0 \tag{2}$$

$$H^{T_r}_1 = 0 \tag{3}$$

If the Boltzmann selection is used in (2), the Boltzmann temperature T is updated as follows:

$$T \leftarrow T - \Delta T. \tag{4}$$

(2) Action Decision

Each agent observes the state s_t, and decides the action. The action decided by the roulette selection or the Boltzmann selection.

In the roulette selection, the probability that the agent x selects the action a_t^x when the state s_t is observed, $\pi^x(s_t, a_t^x)$ is given by

$$\pi^x(s_t, a_t^x) = \frac{Q^x(s_t, a_t^x)}{\sum_{a \in A^x(s_t)} Q^x(s_t, a)} \tag{5}$$

where $Q^x(s_t, a_t^x)$ is the action-value function that the agent x takes the action a_t^x for the state s_t, $A^x(s_t)$ is the set of possible actions of the agent x.

In the Boltzmann selection, the probability that the agent x chooses the action a_t^x when the state s_t is observed $\pi^x(s_t, a_t^x)$ is given by

$$\pi^x(s_t, a_t^x) = \frac{\exp(Q^x(s_t, a_t^x)/T)}{\sum_{a \in A^x(s_t)} \exp(Q^x(s_t, a)/T)} \tag{6}$$

where T is the Boltzmann temperature.

(3) Observation of Environment Change

Each agent takes the action chosen in (2), and the state s_t transits s_{t+1}. Here, whether the environment change is observed. If the environment change is observed at the time $t+1$, n is updated as

$$n \leftarrow n + 1. \tag{7}$$

And, the history lists are updated as follows:

$$H_{n,x}^A = a_t^x \tag{8}$$
$$H_n^{T_s} = t + 1 \tag{9}$$
$$H_{n+1}^{T_r} = H_n^{T_r}. \tag{10}$$

(4) Reward Redistribution

If there is an agent that obtained the positive reward from the environment, the agent decides how much reward to which agent to give.

(a) If the agent X_0 receives the positive reward from the environment, the agent X_1 that the agent X_0 gives a part of reward is decided by

$$X_1 = \underset{x}{\text{argmax}}(C^n(H_{n,x}^A)) \tag{11}$$

$$C^n(a) = \left(\sum_{i:0<H_i^{T_s}<H_n^{T_r}} \sum_x \delta_{H_{i,x}^A, a} \right) \times \left(\sum_{i:H_n^{T_r} \le H_i^{T_s}} \sum_x \delta_{H_{i,x}^A, a} \right) \tag{12}$$

where $\delta_{i,j}$ is the Kronecker delta, and is given by

$$\delta_{i,j} = \begin{cases} 1, & (i = j) \\ 0, & (i \neq j). \end{cases} \tag{13}$$

(b) When the agent oneself was chosen as a partner handing a reward ($(X_0 = X_1)$), the agent X_0 does not give a part of reward to any other agents, the quantity of the reward that the agent X_0 gives to the other agent from the reward received at the time $t+1$, $r_{t+1}^{X_0 \rightarrow}$ is set to

$$r_{t+1}^{X_0 \rightarrow} = 0, \tag{14}$$

and go to (5). Otherwise, go to (c).

(c) The agent X_0 give a part of the reward to the agent X_1. The quantity of the reward that the agent X_0 gives to the agent X_1 is given by

$$r_{t+1}^{X_0 \rightarrow} = r_{H_n^{T_s}}^{\rightarrow X_1} = \begin{cases} 0.5 r_{t+1}^{\rightarrow X_0} \dfrac{C^n(H_{n,X_1}^A)}{n'(n-n')}, \\ \qquad \left(\dfrac{C^n(H_{n,X_1}^A)}{n'(n-n')} < 1 \right) \\ 0.5 r_{t+1}^{\rightarrow X_0}, (\text{otherwise}) \end{cases} \tag{15}$$

$$n' = \max_i \left(i : H_i^{T_s} < H_n^{T_r} \right) \tag{16}$$

(d) The agent X_i that receives the reward from the other agent judges whether it gives to a part of reward to the other agent. The agent X_{i+1} that agent X_i gives a part of reward to is decided as follows:

$$X_{i+1} = \underset{x}{\text{argmax}} \left(C^{n-i}(H_{n-i,x}^A) \right) \tag{17}$$

(e) When the agent oneself was chosen as a partner handing a reward ($(X_i = X_{i+1})$), the agent X_i does not give a part of reward to any other agents, the quantity of the reward that the agent X_i gives to the other agent from the reward received at the time $H_{n-i}^{T_s}$ is set to

$$r_{H_{n-i}^{T_s}}^{X_i \rightarrow} = 0, \tag{18}$$

and go to (5). Otherwise, go to (f).

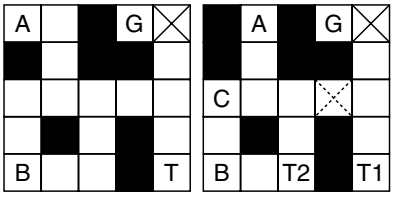
(f) The agent X_i give a part of the reward to the agent X_{i+1}. The quantity of the reward that the agent X_i gives to the agent X_{i+1} is given by

$$r_{H_{n-i+1}^{T_s}}^{X_i \rightarrow} = r_{H_{n-i}^{T_s}}^{\rightarrow X_{i+1}} = \begin{cases} 0.5 r_{H_{n-i}^{T_s}}^{\rightarrow X_i} \dfrac{C^{n-i}(H_{n-i}^A, X_{i+1})}{(n-i)'((n-i)-(n-i)')}, \\ \qquad \left(\dfrac{C^{n-i}(H_{n-i}^A, X_{i+1})}{(n-i)'((n-i)-(n-i)')} < 1 \right) \\ 0.5 r_{H_{n-i}^{T_s}}^{\rightarrow X_i}, \quad (\text{otherwise}) \end{cases} \tag{19}$$

(g) Back to (d).

If there is an agent that receives the positive reward, the history is updated as follows:

$$H_{n+1}^{T_r} = t + 1 \tag{20}$$



(a) (b)

Fig. 1. Map used in Path Finding Problem.

If the agent x receives the negative reward from the environment, $r_{t+1}^{x\rightarrow}$ is set to

$$r_{t+1}^{x\rightarrow} = 0. \tag{21}$$

(5) Update of Action-Value Function
The action-value function is updated.

(a) Profit Sharing The action-value function of the agent X_i, $Q^{X_i}(s_\tau, a_\tau^{X_i})$ is updated as follows:

$$Q^{X_i}(s_\tau, a_\tau^{X_i}) \leftarrow Q^{X_i}(s_\tau, a_\tau^{X_i}) + r^{X_i}(\tau) \tag{22}$$

where $r^{X_i}(\tau)$ is the reward function, and is given by

$$r^{X_i}(\tau) = \begin{cases} r_{t_r^{X_i}}^{\rightarrow X_i} - r_{t_r^{X_i}}^{X_i\rightarrow}, & (\tau = t_r^{X_i}) \\ \dfrac{1}{M} r^{X_i}(\tau+1), & (\tau < t_r^{X_i}) \end{cases} \tag{23}$$

where $r_{t_r^{X_i}}^{\rightarrow X_i}$ is the reward that the agent X_i receives at the time $t_r^{X_i}$, $r_{t_r^{X_i}}^{X_i\rightarrow}$ is the reward that the agent X_i gives to the other agent, and M is the number of possible actions.

(b) Q-learning The action-value function of the agent x $Q^x(s_{t_r^x}, a_{t_r^x})$ is updated.

$$\begin{aligned} Q^x(s_{t_r^x}, a^{t_r^x}) \leftarrow \quad & Q^x(s_{t_r^x}, a_{t_r^x}) + \alpha[r_{t_r^x}^{\rightarrow x} - r_{t_r^x}^{x\rightarrow} \\ & + \gamma \max_a Q^x(s_{t_r^x}, a) - Q^x(s_{t_r^x}, a_{t_r^x})] \end{aligned} \tag{24}$$

(6) Judgment
If the state s_{s+1} is the goal state, back to (1). Otherwise go to (2).

III. Computer Experiment Results

Here, we show the computer experiment results to demonstrate the effectiveness of the proposed method.

A. Path Finding Problem (Two Agents)

We applied the proposed method to the path finding problem. Figure 1 (a) shows the map which was used in the experiment. In Fig.1 (a), white squares show the floor, black squares show wall. A and B show the start points of the agents A and B, and G shows the goal. T is the switch, and $\times$ shows the obstacle which disappears when an agent reaches T. Each agent can move up/down/left/right, or stay.

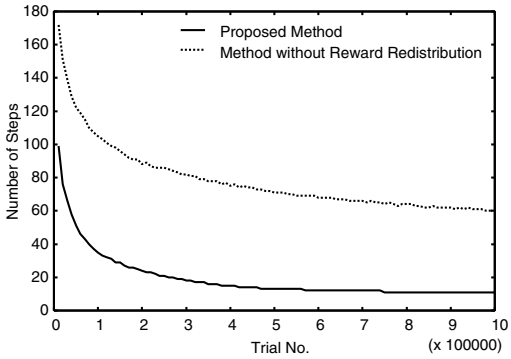


Fig. 2. Variation of Steps (Profit Sharing) (1).

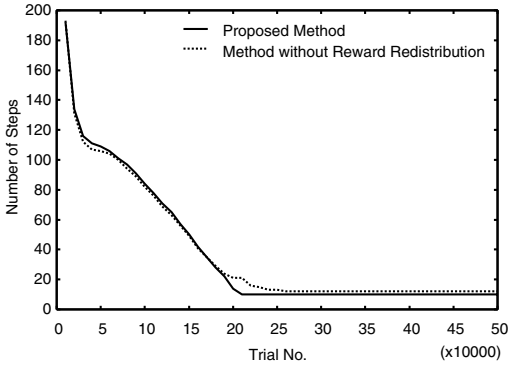


Fig. 3. Variation of Steps (Q-learning) (1).

1) Comparison with Method without Reward Redistribution: Here, we compared the proposed method with the method without reward redistribution. In the method without reward redistribution, only an agent that reaches the goal receives the reward.

Figures 2 and 3 shows the variation of steps that the agent reaches the goal. As shown in Fig.2, the agent in the proposed method reached the goal in 11 steps at the 1000000th trail, and the agent in the method without reward redistribution reached the goal in 60 steps. In the same way, in Fig.3, the agent in the proposed method reached the goal in 10 steps at the 1000000th trail, and the agent in the method without reward redistribution reached the goal in 12 steps.

As shown in Figs.2 and 3, the proposed method can learn faster than the method without reward redistribution. This is because that the proposed method can learn the agent push the switch by the reward redistribution.

2) Comparison with Method with Reward Redistribution: Here, we compared the proposed method with the method with reward redistribution. In the method with reward redistribution, the agent that reaches the goal gives a half of the reward to the agent that pushes the switch.

Figures 4 and 5 show the variations of steps that the agent reaches the goal. As shown in Fig.4, the agent in the proposed

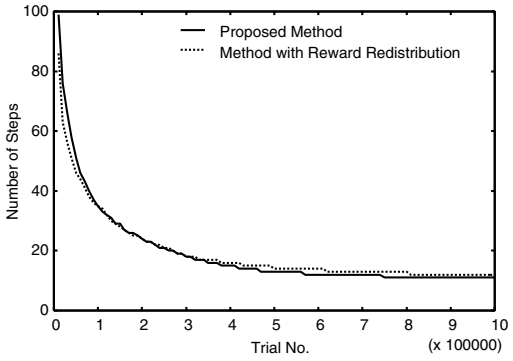


Fig. 4. Variation of Steps (Profit Sharing) (2).

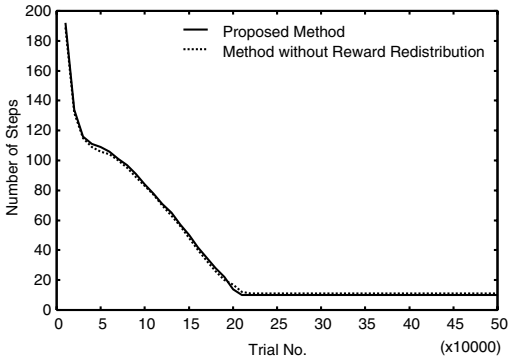


Fig. 5. Variation of Steps (Q-learning) (2).

method reached the goal in 11 steps at the 1000000th trail, and the agent in the method with reward redistribution reached the goal in 12 steps. In the same way, in Fig.5, the agent in the proposed method reached the goal in 10 steps at the 1000000th trail, and the agent in the method with reward redistribution reached the goal in 11 steps.

As shown in Figs.4 and 5, the proposed method can learn fast as the method with reward redistribution. So, we can see that the agents can decide how much reward to which agent to give in the proposed method.

3) Trained Actions in Proposed Method: Figure 6 shows the trained actions in the proposed method. As shown in this figure, the agent that arrives at the fork in the path first pushes the switch, and the agent that arrives there later goes to the goal.

B. Path Finding Problem (Three Agents)

We applied the proposed method to the path finding problem of three agents. Figure 1 (b) shows the map which was used in the experiment.

In Fig.1 (b), white squares show the floor, black squares show wall. A, B and C show the start points of the agents A, B and C, and G shows the goal. T1 and T2 are the switches, and $\times$ shows the obstacle which disappears when an agent reaches T. Each agent can move up/down/left/right, or stay.

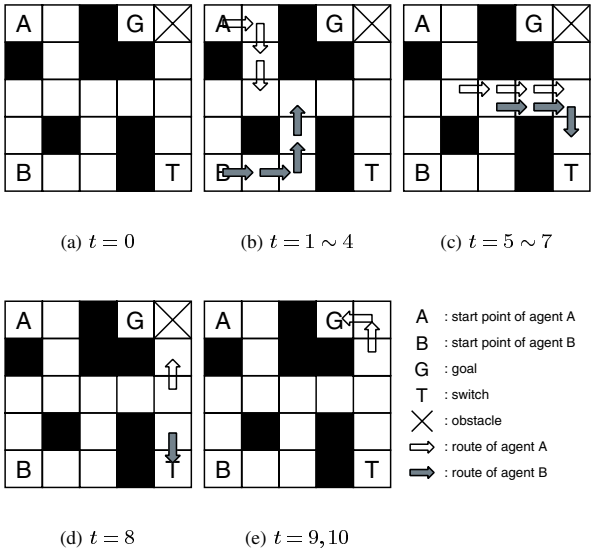


Fig. 6. Trained Actions in Proposed Method (Path Finding Problem).

1) Comparison with Method without Reward Redistribution: Here, we compared the proposed method with the method without reward redistribution. In the method without reward redistribution, only an agent that reaches the goal receives the reward.

Figures 7 and 8 show the variation of steps that the agent reaches the goal. As shown in Fig.7, the agent in the proposed method reached the goal in 48 steps at the 1000000th trail, and the agent in the method without reward redistribution reached the goal in 103 steps. In the same way, in Fig.8, the agent in the proposed method reached the goal in 18 steps at the 1000000th trail, and the agent in the method without reward redistribution reached the goal in 46 steps.

As shown in Figs.7 and 8, the proposed method can learn faster than the method without reward redistribution. This is because that the proposed method can learn the agent pushes the switch by the reward redistribution.

2) Comparison with Method with Reward Redistribution: Here, we compared the proposed method with the method with reward redistribution. In the method with reward redistribution, the agent that reaches the goal gives a half of the reward to the agent that pushes the switch T1, and the agent that reaches the switch T1 gives a half of the reward to the agent that pushes the switch T2.

Figures 9 and 10 show the variations of steps that the agent reaches the goal. As shown in Fig.9, the agent in the proposed method reached the goal in 35 steps at the 1000000th trail, and the agent in the method with reward redistribution reached the goal in 10 steps. In the same way, in Fig.10, the agent in the proposed method reached the goal in 18 steps at the 1000000th trail, and the agent in the method with reward redistribution reached the goal in 11 steps.

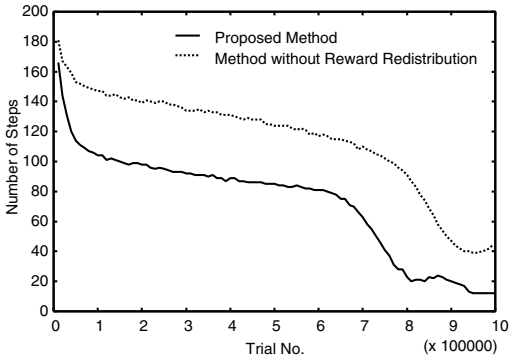


Fig. 7. Variation of Steps (Profit Sharing) (3).

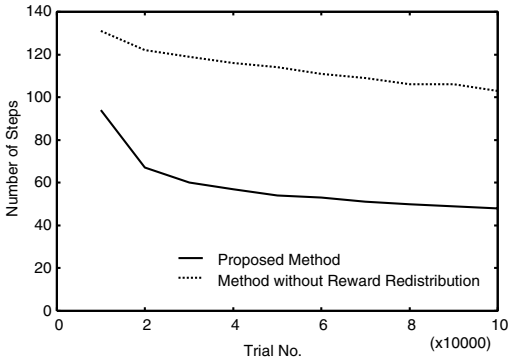


Fig. 8. Variation of Steps (Q-learning) (3).

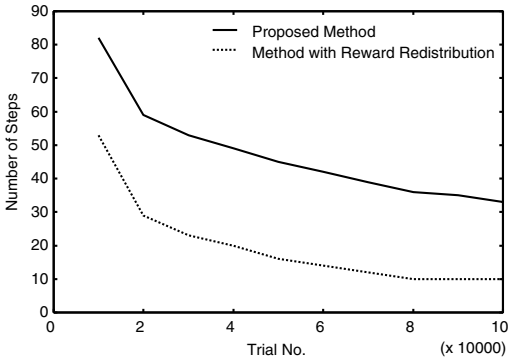


Fig. 9. Variation of Steps (Profit Sharing) (4).

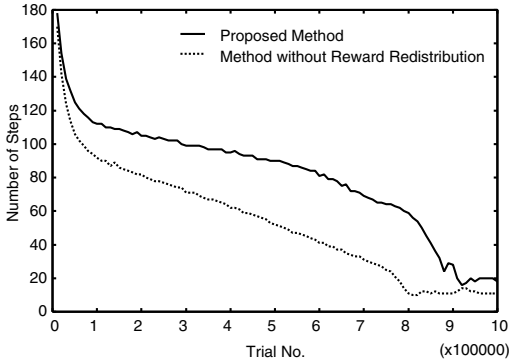


Fig. 10. Variation of Steps (Q-learning) (4).

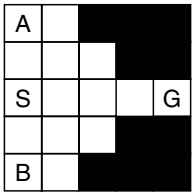


Fig. 11. Map in Transportation Problem.

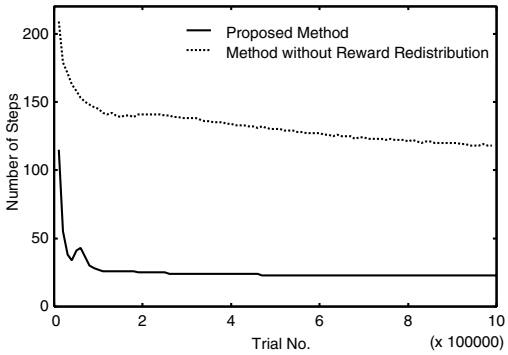


Fig. 12. Variation of Steps (Profit Sharing) (5).

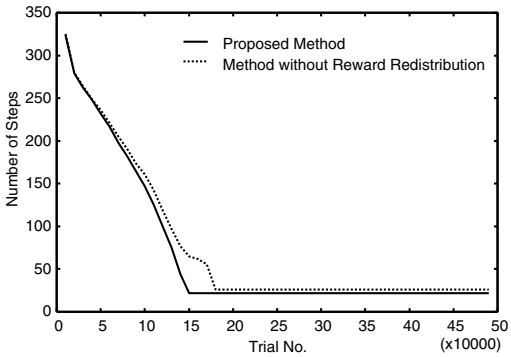


Fig. 13. Variation of Steps (Q-learning) (5).

C. Transportation Problem (Two Agents)

We applied the proposed method to the transportation problem. Figure 11 shows the map which was used in the experiment.

In Fig.11, white squares show the floor, black squares show wall. A and B show the start points of the agents A and B, S is the load depot (five loads are there), and G is the goal. Each agent can move up/down/left/right, pass a load to the other agent, or stay.

1) Comparison with Method without Reward Redistribution: Here, we compared the proposed method with the method without reward redistribution. In the method without reward redistribution, only an agent that reaches the goal with a load receive the reward.

Figures 12 and 13 show the variation of steps that the agent reaches the goal. As shown in Fig.12, the agent in the proposed method reached the goal in 23 steps at the 1000000th trail, and the agent in the method without reward redistribution reached

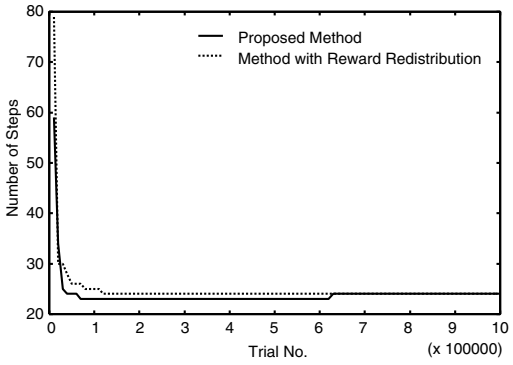


Fig. 14. Variation of Steps (Profit Sharing) (6).

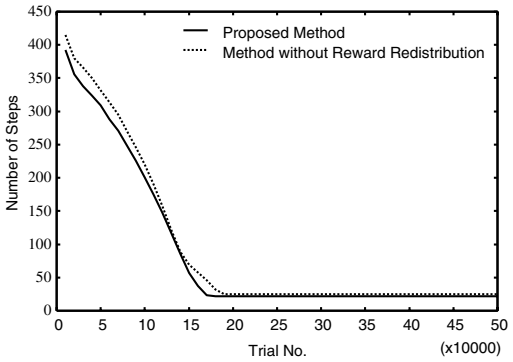


Fig. 15. Variation of Steps (Q-learning) (6).

the goal in 118 steps. In the same way, in Fig.13, the agent in the proposed method reached the goal in 22 steps at the 1000000th trail, and the agent in the method without reward redistribution reached the goal in 26 steps.

2) Comparison with Method with Reward Redistribution: Figures 14 and 15 show the variations of steps that the agent reaches the goal. As shown in Fig.14, the agent in the proposed method reached the goal in 24 steps at the 1000000th trail, and the agent in the method with reward redistribution reached the goal in 24 steps. In the same way, in Fig.15, the agent in the proposed method reached the goal in 22 steps at the 50000th trail, and the agent in the method with reward redistribution reached the goal in 25 steps.

3) Trained Actions in Proposed Method: Figure 16 shows the trained actions in the proposed method.

IV. Conclusion

In this paper, we have proposed the method for generating roles by redistribution of reward between agents in reinforcement learning of multiagent system. In the proposed method, there are two kinds of reward; (1) reward obtained from environment when agents reach the goal and (2) reward received from other agents. Agents can learn actions to work cooperatively with other agents by the reward received from other agents, and the roles of agents are generated. In the proposed method, agents can decide how much reward to

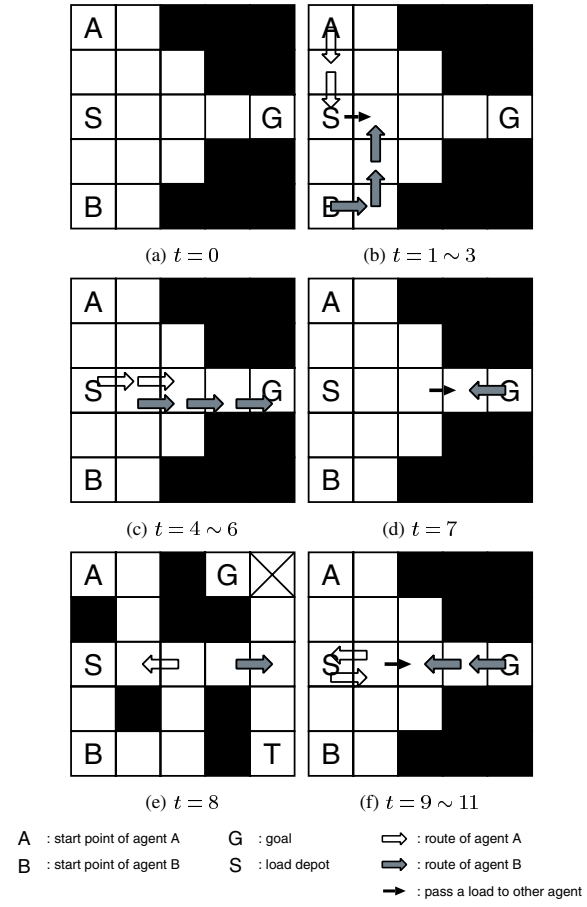


Fig. 16. Trained Actions in Proposed Method (Transportation Problem).

which agent to give from the obtained reward by using the history that records actions at the time when the change of environment are recognized. We carried out computer experiments for two tasks; (1) path finding problem and (2) transportation problem, and confirmed that roles of agents are generated by redistribution of reward between agents in the proposed method. Moreover, we confirmed that the proposed method can learn as similar as when the designer decides conditions for redistribution of reward.

References

[1] M. Saitoh and Y. Oyama: "Economy-like reward distribution for division of labor," Proceedings of the 24th IASTED International Multi-Conference on Artificial Intelligence And Applications, Innsbruck, pp.499–506, 2006.

[2] M. Saitoh: "Impacts of team size on role learning in multiagent systems," Proceedings of the 26th IASTED International Conference on Artificial Intelligence And Applications, Innsbruck, pp.252–257, 2008.

[3] R.S.Sutton and A.G Barto: Reinforcement Learning: An Introduction, The MIT Press, 1998.

[4] J. J. Grefenstette: "Credit assignment in rule discovery systems based on genetic algorithms," Machine Learning, Vol.3, pp.224–245, 1988.

[5] C. J. H. Watkins and P. Dayan: "Q-learning," Machine Learning, Vol.8, pp.55–68, 1992.