SICE Annual Conference 2008
August 20-22, 2008, The University Electro-Communications, Japan

Role differentiation process by division of reward function in multi-agent reinforcement learning

Tadahiro Taniguchi[1] , Kazuma Tabuchi[2] and Tetsuo Sawaragi [2]

[1]Dept. of Human and Computer Intelligence, Ritsumeikan University, Shiga, Japan
(Tel : +81-77-561-5839; E-mail: taniguchi@ci.ritsumei.ac.jp)
[2]Dept. of Mechanical Eng. and Sci., Kyoto University, Kyoto, Japan
(Tel : +81-75-753-5266; E-mail: sawaragi@me.kyoto-u.ac.jp)

Abstract: There are several problems which discourage an organization from achieving tasks, e.g. partial observation, credit assignment, and concurrent learning in the multi-agent reinforcement learning domain. In many conventional approaches, each agent estimates hidden states, e.g., other agents' sensor inputs, positions, and policies, and reduces the uncertainty in the partially-observable Markov decision process (POMDP), and solves the multiagent reinforcement learning problem. In contrast, people reduce uncertainty in human organizations in the real world by autonommously dividing the roles played by each agent. In a framework of reinforcement learning, roles are mainly represented by goals for each agent. This paper presents a method for generating internal rewards from manager agents to worker agents. It also explicitly divides the roles, which can change a POMDP task for each agent into a simple MDP task under certain conditions. Several situational experiments are also described and the validity of the proposed method is evaluated.

Keywords: multi-agent reinforcement learning, organization, reole differentiation, POMDP

1. INTRODUCTION

In a multi-agent environment, an agent should adapt to diverse dynamics caused by changes in physical properties of the task environment and in social situations concerning how other agents behave and change their behavior. Owing to the socially dynamic property of a multi-agent system, it is difficult for participating agents to achieve multi-agent reinforcement learning tasks. In contrast, it seems that people can solve such kinds of problems in our society as a whole.

There are several computational problems, which discourage an organization consisting of several agents from achieving multi-agent reinforcement learning tasks, e.g., partial observation, credit assignment, and concurrent learning problems. To overcome these problems, many researchers investigating multi-agent reinforcement learning problem have tried to solve this problem.

In most conventional approaches to the partial observation problem in a multi-agent reinforcement learning domain, each agent estimates hidden states in a socially dynamic system, e.g., other agents' sensor inputs, positions, and policies to reduce the uncertainty in partially-observable Markov decision process (POMDP). However, such methods require a large amout of computational resources and time to solve this problem.

In contrast, people reduce uncertainty in human organizations by constructing a division of roles played by each agent. In the context of multi-agent reinforcement learning, the division of roles are considered to be only a result of a multi-agent learning task. However, the division of roles are generated not only from the bottom-up, but also from the top-down in human organizations. We describe a method which enables an agent who manages an organization to rationally divide participating agent's roles, and examine the its effectiveness.

In the framework of reinforcement learning, roles are mainly represented by goals for each agent. We present a method for generating individual rewards which are provided by a manaer agent to worker agents. The explicit rational division of roles can change a POMDP task for each agent into a simple MDP task under certain conditions.

Several experiments are described and the validity of the proposed method is evaluated.

2. MULTI-AGENT REINFORCEMENT LEARNING

2.1 Reinforcement learning

Reinforcement learning is a learning method that enables animals and learning machines to obtain an optimal policy so as to maximize averaged cumulative rewards in a task environment. In reinforcement learning tasks, the optimal action output is not provided to the learner, which is in contrast to supervised learning methods. Therefore, the learner has to modify its policy through trial and error.

Most reinforcement learning methods assume that the environmental dynamics are described as a Markov decision process (MDP). In an MDP, after an agent observes its state $\mathbf{x_t} \in \mathbb{X}$ and outputs action $\mathbf{u_t} \in \mathbb{U}$ at time step t, the agent observes the next state $\mathbf{x_{t+1}}$ and obtains a reward r_{t+1}. The MDP assumes that the transition probability is defined as $p(\mathbf{x_{t+1}}|\mathbf{x_t}, \mathbf{u_t})$, and the expectation of the reward is $E[r_{t+1}|\mathbf{x_t}, \mathbf{u_t}]$.

2.2 Multi-agent reinforcement learning

In a multi-agent reinforcement learning process, multiple agents are engaged in a reinforcement learning task. In this paper, we deal with multi-agent reinforcement learning problems. We assume that the environmental dy-

namics are described as an entire MDP. The state of the whole multi-agent system is defined as $\mathbf{x^G} \in \mathbb{X^G}$. The action output, rewards, and the transition probability are $\mathbf{u^G} \in \mathbb{U^G}$, $r^G : \mathbb{X}^G \times \mathbb{U}^G \to \mathbb{R}$, and $p\left(\mathbf{x^{G\prime}}|\mathbf{x^G}, \mathbf{u^G}\right)$, respectively.

We assume that the i-th agent observes states $\mathbf{x^i} \in \mathbb{X^i}$ and outputs $\mathbf{u^i} \in \mathbb{U^i}$ in this paper, and that $\mathbb{X}^G = \mathbb{X}^1 \otimes \mathbb{X}^2 \otimes \cdots \otimes \mathbb{X}^N$,. $\mathbb{U}^G = \mathbb{U}^1 \otimes \mathbb{U}^2 \otimes \cdots \otimes \mathbb{U}^M$. A group of agents obtain rewards based on their collaborative behavior. We assume that the i-th agent learns the function $\mu^i_{\theta_i}\left(\mathbf{u^i}|\mathbf{x^i}\right)$, which calculates $\mathbf{u^i}$ from $\mathbf{x^i}$, by referring to r^i given to each agent, where θ_i is a parameter of the i-th agent's policy function. The total superimposed policy, which is generated based on each agent's individual learning, is defined as $\mu^G_{\theta_G}\left(\mathbf{u^G}|\mathbf{x^G}\right)$, where $\theta_G = \{\theta_1, \theta_2, \cdots \theta_N\}$.

2.3 Problems in multi-agent reinforcement learning

The following problems in multi-agent reinforcement learning are widely known in the research area [2].

2.3.1 Partial observation problem

In a multi-agent task environment, each agent rarely observes all the other agents' state variables. Therefore, partial observation problem is a multi-agent reinfocement learning problem. Although the whole multi-agent system satisfies the MDP, the actual environmental dynamics of the reinforcement-learning problem for each agent often becomes a partially observable Markov decision process (POMDP) because each learner cannot observe $\mathbf{x^G}$ by itself.

Several previous studies were performed to overcome POMDP. The can be sclassified into two approaches. In the first approach, the learning methods do not assume that the environmental dynamics satisfy MDP. In the second approach, the learning method includes participating agent to estimate the hidden variables and recovers the MDP [3, 4].

A method utilizing profit-sharing, which has a certatin robustness when the transition probability involves much uncertainty, can be considered as part of the first approach [10]. Arie [3] proposed a reinforcement-learning method using recurrent neural network (RNN). An RNN keeps some past information and utilizes it to supplement the hidden variables. This is an example of the second approach.

However, if the agent identifies all the hidden states in a multi-agent system, the total number of state spaces increases dynamically because a participating agent has to take all the other agents' states into consideration to decide its outputs and policy.

2.3.2 Concurrent learning problem

Generally, the i-th agent's transition probability $p\left(\mathbf{x^{i\prime}}|\mathbf{x^G}, \mathbf{u^G}\right)$ is at least influenced by the other agents' states and actions. In fact,

$$p\left(\mathbf{x^{i\prime}}|\mathbf{x^G}, \mathbf{u^G}\right) = p\left(\mathbf{x^{i\prime}}|\mathbf{x^G}, \mathbf{u^i}, \left\{\mu^{\mathbf{j}}_{\theta_{\mathbf{j}}}, \mathbf{j} \neq \mathbf{i}\right\}\right). \quad (1)$$

When the other agents also learn and change their policies $\mu^j_{\theta_j}$ concurrently, the transition probability becomes unstable and the MDP cannot be fixed temporally. This makes the convergence of the reinforcement learning for each agent difficult.

2.3.3 Credit assignment problem

It is difficult for agents participating in a multi-agent reinforcement learning problem to learn based on only the global reward r^G. If $r^i = r^G$ when some agents perform adequately and the others perform badly, all the agents' actions are equally encouraged or discouraged. However, if we give a reward to an agent who directly contributes to the acquisition of the final reward, agents who assist that agent cannot be evaluated properly. To overcome such a problem, properly assigning a reward to each agent is important.

To overcome all these problems, we developed a division of reward function. This approach is inspired by human organizations.

3. DIVISION OF ROLES BY DIVIDING REWARD FUNCTIONS

3.1 Importance of designing reward functions in multi-agent system

In most computational reinforcement learning processes, the task environments are assumed to be described using MDPs whose observable states, action outputs, and rewards are represented by $\mathbf{x}$,.$\mathbf{u}$ and r, respectively. It is important for each participating agent to adequately achieve tasks. However, multi-agent reinforcement learning inevitably involves a POMDP problem.

Our approach aims to recover the MDP by redesigning each agent's reward function based on the structure of the target multi-agent task. If the agents learn several possible optimal policies and each optimal policy is mutually related, the agent's leaning process is inevitably affected by the other agent's leaning processes. Therefore, to make each agent's goal that a participating agent tries to achieve fixed during a task is valuable.

If an organization has to obtain a global optimal policy of the original multi-agent reinforcement-learning task, each agent has to solve the POMDP based on the original reward function r^G. However, if the POMDP can be transformed into an MDP by redesigning the i-th agent's reward function r^i so that its variables can be observed by the i-th agent, each agent can learn smoothly based on the MDP. Moreover, if the optimal policies of the modified reinforcement-learning task prove to be almost the same as the optimal policies of the original task, the design of the reward functions for each agent must be useful even if the optimal performance of the organization becomes a little worse.

A similar approach uses factored MDPs [7]. Factored MDPs are used to describe the relationship between state and action by using Bayesian networks, and they define each node's linear value function which has variables effected on the node. Therefore, each node effectively ac-

quires an optimal policy. By assuming each node in the network to be an agent in a multi-agent system, Guestrin et al. [6] proposed an effective multi-agent reinforcement learning method. However, Guestrin's approach is actually a top-down approach. We are focusing on the emergence of role differentiation in an autonomous agent system. We focus on the division of the reward function, and discuss the design method of r^i under the condition that each agent is dynamically independent.

3.2 Division of rewards under the condition of independent transition probability

If two main conditions are satisfied, the total MDP, which represents a multi-agent system, can be mathematically divided into several independent MDPs, (1) the dynamics of each participating agents are independent and (2) the reward function given to the organization can be represented by a product of elemental reward functions corresponding to participating agents. In other words, the reward function is multiplicative. Each elemental reward function has each agent's sensory inputs and motor outputs as input variables.

Konda [8] formulated a reinforcement learning task as a problem in which an agent maximizes the following target evaluation function $\bar{\alpha}^G$.

$$\bar{\alpha}^G\left(\theta^G\right) = \int_{\mathbb{X}^G,\mathbb{U}^G} r^G\left(\mathbf{x^G}\right)\eta^G_{\theta_G}\left(\mathbf{x^G},\mathbf{u^G}\right)d\mathbf{x^G}\mathbf{du^G}. \quad (2)$$

Here, we assume that each agent observes a different part of $\mathbf{x^G}$ and their total observation exhausts the variables. Therefore, the state and action spacea are described as $\mathbb{X}^G = \mathbb{X}^1 \otimes \mathbb{X}^2 \otimes \cdots \otimes \mathbb{X}^N$ and $\mathbb{U}^G = \mathbb{U}^1 \otimes \mathbb{U}^2 \otimes \cdots \otimes \mathbb{U}^N$, respectively. We assume that the transition probability of each agent is independent and behaves based on its own observation. The global state transition probability is $p\left(\mathbf{x^{G\prime}}|\mathbf{x^G},\mathbf{u^G}\right)$, and the global policy $\mu^G_{\theta_G}\left(\mathbf{u^G}|\mathbf{x^G}\right)$ becomes

$$p\left(\mathbf{x^{G\prime}}|\mathbf{x^G},\mathbf{u^G}\right) = \prod_i p\left(\mathbf{x^{i\prime}}|\mathbf{x^i},\mathbf{u^i}\right), \quad (3)$$

$$\mu^G_{\theta_G}\left(\mathbf{u^G}|\mathbf{x^G}\right) = \prod_i \mu^i_{\theta_i}\left(\mathbf{u^i}|\mathbf{x^i}\right). \quad (4)$$

Therefore, $p_{\theta_G}\left(\mathbf{x^{G\prime}}|\mathbf{x^G}\right)$ becomes

$$\begin{aligned} p_{\theta_G}\left(\mathbf{x^{G\prime}}|\mathbf{x^G}\right) &= p\left(\mathbf{x^{G\prime}}|\mathbf{x^G},\mathbf{u^G}\right)\mu^G_{\theta_G}\left(\mathbf{u^G}|\mathbf{x^G}\right) \\ &= \prod_i p\left(\mathbf{x^{i\prime}}|\mathbf{x^i},\mathbf{u^i}\right)\prod_i \mu^i_{\theta_i}\left(\mathbf{u^i}|\mathbf{x^i}\right) \\ &= \prod_i p\left(\mathbf{x^{i\prime}}|\mathbf{x^i},\mathbf{u^i}\right)\mu^i_{\theta_i}\left(\mathbf{u^i}|\mathbf{x^i}\right) \\ &= \prod_i p_{\theta_i}\left(\mathbf{x^{i\prime}}|\mathbf{x^i}\right). \end{aligned} \quad (5)$$

In this case, stationary distribution $\eta^G_{\theta_G}\left(\mathbf{x^G},\mathbf{u^G}\right)$ can be divided, as follows

$$\eta^G_{\theta_G}\left(\mathbf{x^G},\mathbf{u^G}\right) = \prod_i \eta^i_{\theta_i}\left(\mathbf{x^i},\mathbf{u^i}\right), \quad (6)$$

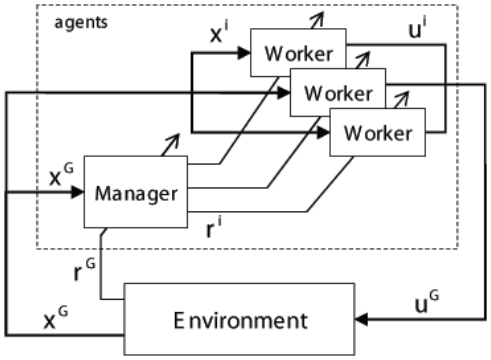


Fig. 1 Organization of agents

Therefore, if the global reward function r^G satisfies

$$r^G\left(\mathbf{x^G}\right) = \prod_i r^i\left(\mathbf{x^i}\right), \quad (7)$$

(2) becomes

$$\bar{\alpha}^i\left(\theta^i\right) = \int_{\mathbb{X}^i,\mathbb{U}^i} r^i\left(\mathbf{x^i}\right)\eta^i_{\theta_i}\left(\mathbf{x^i},\mathbf{u^i}\right)d\mathbf{x^i}\mathbf{du^i}, \quad (8)$$

$$\bar{\alpha}^G\left(\theta^G\right) = \prod_i \bar{\alpha}^i. \quad (9)$$

The target multi-agent task is divided into partial problems. For each problem, each agent is required to maximize its obtaining indivisual reward (8). Each problem is simply an MDP and can be solved by using a conventional reinforcement learning method.

Generarly, a reward function cannot be divided like (9). By replacing r^G with r^M, we can divide a reward function based on (9). If r^M has a similar optimal policy to r^G, the division of the reward function will improve the learning process.

Additionally, the independent state transition probabilities for each agent are generally rarely found. However, this approach will improve the learning process in a loosely coupled multi-agent system whose connection effect is considered as only noise.

3.3 Architecture of the learning organization

A new agent who manages an agent group by designing r^M is added to the group of the multi-agent system. We name this agent "manager". Correspondingly, we name the ordinal agent, who interacta with the environment by outputing $\mathbf{u^i}$, "worker". In the model described in this paper, the manager completely observes $\mathbf{x^G}$. However, the manager cannot directly engage in the task. The manager can only interact with the task environment indirectly by outputting a reward function $\{r^i\}$ to the workers.

3.4 Example of global reward function

For example, we assume $r^G\left(\mathbf{x^G}\right)$ can be described with the sum of the radial basis functions.

$$g\left(\mathbf{x};\mu,\mathbf{S}\right) = \exp\left\{-\frac{1}{2}\left(\mathbf{x}-\mu\right)^T S\left(\mathbf{x}-\mu\right)\right\}, \quad (10)$$

where S is a positive symmetric matrix.

$$r^G\left(\mathbf{x^G}\right) = \sum_j f_j\left(\mathbf{x^G}\right), \quad (11)$$

$$f_j\left(\mathbf{x^G}\right) = w_j g\left(\mathbf{x^G}; \mu_\mathbf{j}, \mathbf{S_j}\right), \quad (12)$$

where $\{w_j, \mu_\mathbf{j}, \mathbf{S_j}\}$ are parameters. If the peaks of the multi-peaked function r^G are separated sufficiently, the stationary distribution based on the optimal policy is expected to be localized around the highest peak. Therefore, sufficient performance can be obtained by the learning organization even if we take only the highest Gaussian function into consideration.

$$j^* = \underset{j}{\operatorname{argmax}}\, w_j \quad (13)$$

$$r^M = w_{j^*} g(x^G; \mu_{j^*}, S_{j^*}) \quad (14)$$

The r^M satisfies the condition of (9), and the optimal policy acquired under the divided reward function is expected to be almost the same as that acquired under r^G.

However, r^G is unknown before the learning task starts. Therefore, the manager has to construct r^M.

3.5 Local approximation of reward function by using M-estimation

How to construct r^M is the next problem. We try to obtain r^M by estimating r^G locally based on M-estimation, which is a type of robust estimation method. We assume the observed input variables to be x_t, and output variables to be y_t. The relation between these variables is modeled by $h\left(x;\theta\right)$, where θ is a parameter.

Cost function J is usually defined to be

$$J = \sum_t \|\varepsilon_t\|^2$$

based on the least-square criteria. In this case, the bigger the residual error is, the more the impact on the cost function is. A few outliers will harm the performance of the approximator. To overcome such a problem, $\rho\left(\varepsilon_t\right)$ is introduced.

$$J = \sum_t \rho\left(\varepsilon_t\right) \quad (15)$$

The M-estimation aims to minimize J. If we assume $\rho\left(\varepsilon_t\right) = \|\varepsilon_t\|^2$, the M-estimation becomes a least-square method. Therefore, M-estimation is a generalized method of the least-square method. By adequately designing ρ, we can reduce the negative effects of the outliers.

The M-estimation has several possibilities of ρ. In this paper, we used the biweight method. The ρ function is described as follow.

$$\rho\left(\varepsilon_t\right) = \begin{cases} \frac{c^2}{6}\left\{1 - \left[1 - \left(\frac{\varepsilon_t}{c}\right)^2\right]^3\right\} & \left(|\varepsilon_t| \le c\right) \\ \frac{c^2}{6} & \left(|\varepsilon_t| > c\right) \end{cases}, \quad (16)$$

where c is a parameter.

We made a single-peaked function which approximates a The global reward function near $\mu_{\mathbf{j}^*}$ by using the biweight method.

$$\varepsilon_t = r_t^G - r^M, \quad (17)$$

$$r^M = wg\left(\mathbf{x^G}; \mu, \mathbf{S}\right), \quad (18)$$

$$r^i = w^i g\left(\mathbf{x^i}; \mu^\mathbf{i}, \mathbf{S^i}\right), \quad (19)$$

where $\theta = \{w, \mu, \mathbf{S}, \mathbf{b}\}$ are the parameters of a Gauussian function. The learning rule is derived based on the steepest decent method. The partial differential of the evaluation function J is

$$\frac{\partial J}{\partial \theta} = \sum_t \frac{\partial \rho}{\partial \varepsilon_t}\frac{\partial \varepsilon_t}{\partial \theta} = -\sum_t \frac{\partial \rho}{\partial \varepsilon_t}\frac{\partial r^\dagger}{\partial \theta}. \quad (20)$$

Therefore, the updated rule becomes

$$\theta_{\tau+1} = \theta_\tau + \beta \sum_{t=1}^{D_\tau} \frac{\partial \rho}{\partial \varepsilon_t}\frac{\partial \varepsilon_t}{\partial \theta_\tau}, \quad (21)$$

where β is a learning rate, $\theta_\tau = \{w_\tau, \mu_\tau, \mathbf{S}_\tau, \mathbf{b}_\tau\}$ are the parameter at a time τ, and D_τ is the number of data observed from τ to$\tau + 1$.

$$\frac{\partial \rho}{\partial \varepsilon_t} = \begin{cases} \varepsilon_t\left\{1 - \left(\frac{\varepsilon_t}{c}\right)^2\right\}^2 & \left(|\varepsilon_t| \le c\right) \\ 0 & \left(|\varepsilon_t| > c\right) \end{cases} \quad (22)$$

However, the obtained data, whose residuals are large, rarely has an effect on the evaluation function J. Therefore, the learning result from the biweight method strongly depends on the initial parameters. In other words, the biweight method only searches locally. Therefore, we have the manager search globally until $t = initialize$. In the global search, the maximal reward $r^G_{\max}$ and the state $\mathbf{x}^\mathbf{G}_{\max}$ at that time are memorized, and the parameters are updated as $w \leftarrow r^G_{\max}, \mu \leftarrow \mathbf{x}^\mathbf{G}_{\max}$ at $t = initialize$. Additionally, to stabilize the learning process, the manager stores the incoming data to the memory whose size is D. When the memory becomes full, the parameters are updated and the data are erased.

By using the biweight method, $r^\dagger$ can be used to finally approximate a part of the target reward function around the specific peak of that function.

4. EXPERIMENT

We evaluated the proposed framework in a simple continuous multi-agent reinforcement-learning environment.

4.1 Conditions

In this experiment, two workers were required to reach given goals (Fig. 2). There were two goals, and they were required to reach different goals, i.e., if the first agent breached the second goal, the second agent would have to reach the first goal. A global reward was given only after both agents reached the different goals. This is a typical problem in multi-agent reinforcement learning. If an agent cannot observe other agents, this problem involves POMDP. Where the first agent should go depends on where the second agent goes. Therefore, concurrent learning problems have to be taken into consideration. Additionally, though an agent reaches the goal, the agent cannot obtain the reward until the other agent reaches its

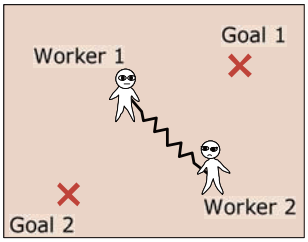
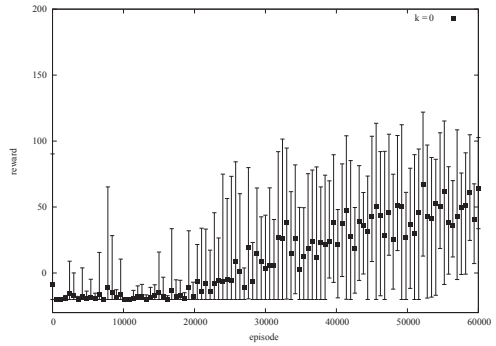
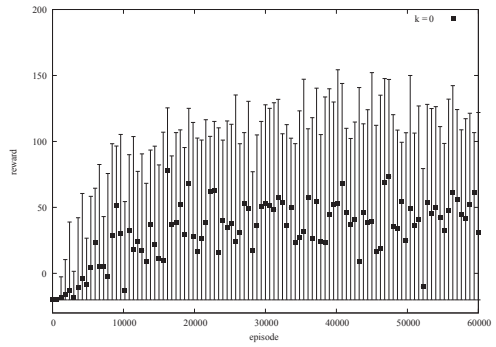


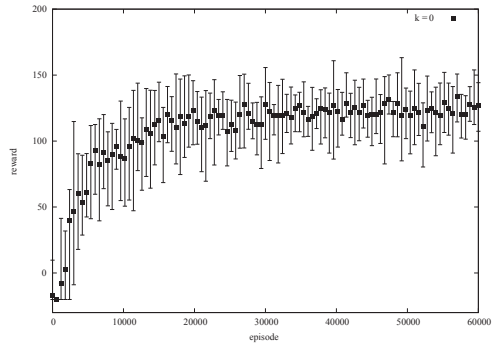
Fig. 2 Overview of the task environment



(a) Fully observable agents with a global reward function



(b) Partially observable agents with a global reward function



(c) Partially observable agents with a devided reward function

Fig. 3 Transitions of acquired reward under condition that state transition probability of each agent is independent

Table 1 Parameters of task environment

T	10	ζ	1.1
Δt	0.01	η	-0.1
κ	5	$\xi^{\mathbf{1}}$	$(3\,3)^T$
$\max u^i$	4	$\xi^{\mathbf{2}}$	$(-3\ -3)^T$
$\min u^i$	-4	A^1, A^2	unit matrix

Table 2 Parameters of workers

A^i_j	$\begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix}$
Learning rate α	0.01
Discount factor γ	0.9
Variance of the exploration noise	0.5

Table 3 Parameters of manager

β	0.1
c	1
K	10
$initialize$	500
$interval$	10

goal. Therefore, credit assignment problems are also related to this simple task.

Each worker's state variable is $\mathbf{x}^i = (x_i\ y_i)^T$, $i = 1, 2$, where x_i, y_i represent the position of the i-th agent. Action outputs were $\mathbf{u^i} = \left(\mathbf{u^i}\ \mathbf{v^i}\right)^{\mathbf{T}}$, $\mathbf{i} = \mathbf{1}, \mathbf{2}$, where u^i, v^i are the velocity toward x, y directions, respectively. The workers observed their own positions, and the manager observed the two workers' position. The state of the total system $\mathbf{x^G} = \left(\mathbf{x^{1^T}}\ \mathbf{x^{2^T}}\right)^{\mathbf{T}}$ transits based on

$$\mathbf{x^G_{t+1}} = \mathbf{x^G_t} + \mathbf{u^G_t \Delta t}, \tag{23}$$

where $\mathbf{u^G} = \left(\mathbf{u^{1^T}}\ \mathbf{u^{2^T}}\right)^{\mathbf{T}}$ and Δt is a time step constant. Every agent observed the states and output its action at intervals of $\kappa\Delta t$ $(\kappa \in \mathbb{N})$. After T seconds passed, the workers were relocated and started their tasks. We call the segment between the relocations as 1 episode.

The global reward function r^G is defined as

$$\begin{aligned} r^G \quad = \quad & \zeta \left\{ g\left(\mathbf{x^1}; \xi_1, \mathbf{A_1}\right) \mathbf{g}\left(\mathbf{x^2}; \xi_2, \mathbf{A_2}\right) \right. \\ & \left. + g\left(\mathbf{x^1}; \xi_2, \mathbf{A_2}\right) \mathbf{g}\left(\mathbf{x^2}; \xi_1, \mathbf{A_1}\right) \right\} + \eta. \end{aligned} \tag{24}$$

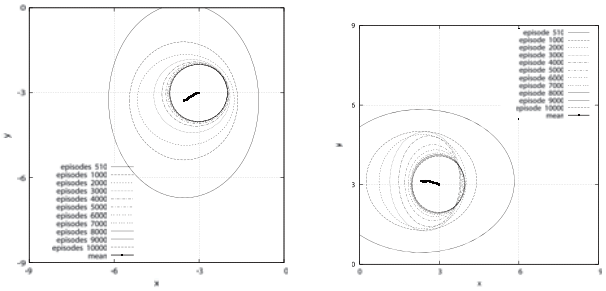
When the two workers remained at $\xi_\mathbf{1}$ and $\xi_\mathbf{2}$ respectively, the total obtained reward was maximized. Therefore, $\mathbf{x^G}$ finally reached $\left(\xi_\mathbf{1}^\mathbf{T}\ \xi_\mathbf{2}^\mathbf{T}\right)^\mathbf{T}$ or $\left(\xi_\mathbf{2}^\mathbf{T}\ \xi_\mathbf{1}^\mathbf{T}\right)^\mathbf{T}$ under the condition that the workers had optimal policies.

The parameters of the task environment are defined in table 1.

The workers were engaged in a reinforcement-learning task by using the actor-critic method [9], which is an on-policy reinforcement learning method. Each worker has two approximators for policy and state value functions as following.

$$\mathbf{u^i}\left(\mathbf{x^i}\right) \quad = \quad \sum_{j=1}^{B} \phi^\mathbf{i}_\mathbf{j}\, \mathbf{g}\left(\mathbf{x^i}; \nu^\mathbf{i}_\mathbf{j}, \mathbf{A^i_j}\right) \tag{25}$$

$$V^i\left(\mathbf{x^i}\right) \quad = \quad \sum_{j=1}^{B} \psi^\mathbf{i}_\mathbf{j}\, \mathbf{g}\left(\mathbf{x}; \nu^\mathbf{i}_\mathbf{j}, \mathbf{A^i_j}\right) \tag{26}$$



(a) worker 1 (b) worker 2

Fig. 4 Reward functions given the corresponding workers

The workers optimaize these functions under the reward functions r^i designed by the managers. The exploration noise was produced by using a normal distribution.

Before the number of episodes reach $initialize$, the center of r^M is relocated to where the biggest reward is observed. After that, the manager modifies r^M by estimating $\left(\mathbf{x^G}, \mathbf{r^G}\right)$ with the M-estimation method at intervals of $interval$.

The workers' basis functions were set in a reticular pattern. The x, y values of their centers ν_j^i were set at $0, \pm 5, \pm 10$. The number of basis functions is $B = 5^2 = 25$. Other parameters were set as shown in table 2, 3.

Two other settings were prepared for comparison. In the first setting, workers observed other agent's state variables, i.e. position, and obtained a global reward $(r^i = r^G, \mathbf{x^i} = \mathbf{x^G})$. We call the setting "fully observable agents with global reward function". In the second setting, workers observed only their own states, but directly obtained a global reward $(r^i = r^G)$. We call the setting "partially observable agents with global reeward function". In the first case, the dimension of the state space was 4. Therefore, B became $5^4 = 625$. The other parameters were set as shown in table 2. We have to pay attention to that the increase in the state space makes the task difficult for the workers to continue their learning process.

4.2 Results 1

Figure 3 shows the averaged cumulated reward for each condition. The horizontal axis shows the number of episodes, and the vertical axis shows the cumulative global reward obtained through an episode. The cumulative reward is the averaged value of ten trials. The error bar represents the best and the worst value in the ten episodes. Figure 4 shows the reward functions given to each worker by the manager. This shows the transition of the designed reward function $\{r^i\}$. The broad line represents the transition of the center of the radial basis function, and the ellipses shows $\left(\mathbf{x^i} - \mu^\mathbf{i}\right)^T S^{ii} \left(\mathbf{x^i} - \mu^\mathbf{i}\right) = 1$ in each episode.

Figure 3 shows that the proposed method enabled the workers to learn more quickly and smoothly. In contrast, the workers who learned based on r^G scored badly and the results were unstable. In this model, the typical POMDP and the concurrent learning problem harmed the learning process.

Moreover, fully observable agents did not score well. This owes to the fact that the trials were stopped after 60000 episodes. If the trials continued, the workers would have reached almost the same level as that for the proposed method. However, the learning speed would be terribly slow because of the large size of its state space. Therefore, the proposed method has the advantage with respect to the speed and smoothness of the learning process. The results show that role differentiation reduces uncertainity in an organization, which enables the organization to achive the target task more easily.

4.3 Results 2

We also evaluated the proposed method under the condition that the agent's dynamics are not independent. We inserted a virtual spring between the two workers to design the dynamical dependency.

$$\mathbf{x_{t+1}^i} = \mathbf{x_t^i} + \left\{\mathbf{u_t^i} + \mathbf{k}\left(\mathbf{x_t^j} - \mathbf{x_t^i}\right)\right\}\mathbf{\Delta t} \tag{27}$$

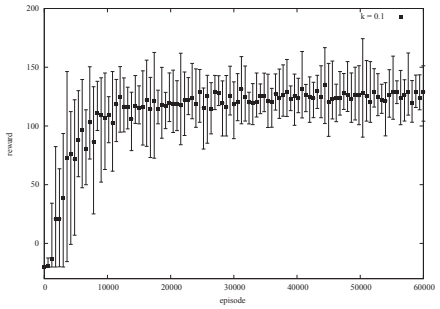
k represents how much unobservable states affect the state transition probability. Three experiments with the proposed model whose $k = 0.1, 0.3, 0.5$ (Figs. 5(a), 5(b), and 5(c), respectively) were performed. The results are shown in Fig. 5 in the same way as in Fig. 3.

Figure 5(c) shows that our proposed method did not work when the condition of independence of transition probability was not satisfied. However, the learning architecture shown in Fig. 5(a) performed no worse than that with $k = 0$. In addition, the results from $k = 0.3$ seemed to be stable and scored well. Therefore, it seems that our proposed model can be applied not only to a completely independent multi-agent system, but also to a loosely-coupled multi-agent system.

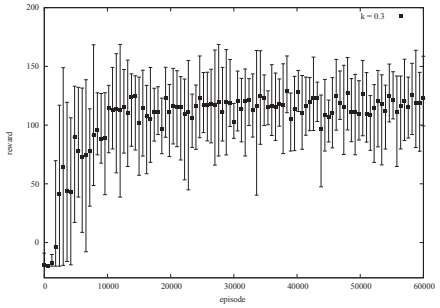
5. CONCLUSIONS

We described multi-agent reinforcement learning appraoch. In this approach, a local reward function, which is given to each agent, is defined differently from the original reward function, and the input variables of each local reward function have to be observed by the corresponding agent. It was shown that a multi-agent reinforcement learning task can be divided into several independent agent's learning tasks by dividing a global reward function into several local reward functions if the dynamics of task environment for each agent is independent and a global reward function is described as a product of such a local reward functions. Through several simulation experiments, the division of roles was found to be effective even if a little dependency was found among the participating agents.

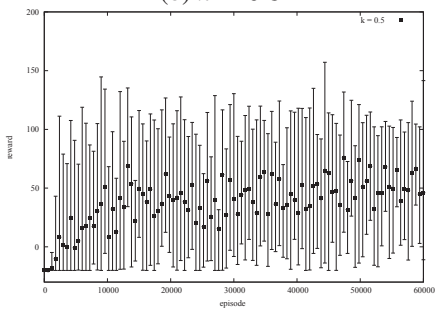
However, the method of designing the multiplicative global reward function has not been derived theoretically. The derivation based on the biweight method is only a heuristic approach to designing dividable global reward functions. More reliable methods should be studied. In



(a) $k = 0.1$



(b) $k = 0.3$



(c) $k = 0.5$

Fig. 5 Transition of cumulative reward when workers were connected with a virtual spring

addition, our approach is applicable to the limited domain of multi-agent reinforcement learning processes as we mentioned. The applicable domain should be clarified in future work.

It is clear that the proposed method does not when each agent's actions affects the other agent's state transitions. In such a case, other existing multi-agent learning methods should be taken into consideration.

In human organizations, many kinds of divisions of roles can be found. The framework described in this paper is only a simple one. More complex role divisions, including several kinds of collaborative works, cannot be achieved based on this framework. For example, "pass" and "shoot" in a football game. How such closely dependent collaborative behavior and division of roles can emerge in a bottom-up fashion is a challenging problem. The research described in this paper does not cover such a division of roles. However, to clarify the domain of role differentiation that can be easily understood mathematically must be clarified to go on to the next step. In future work, we would like to reveal the role differentiation process in such a closely dependent collaborative behavior from the viewpoint of multi-agent reinforcement learning.

REFERENCES

[1] Richard S. Sutton, Andrew G. Barto: Reinforcement Learning: An Introduction, MIT Press (1998)

[2] Sachiyo Arai: Multiagent Reinforcement Learning Frameworks: Steps toward Practical Use, Journal of Japanese Society for Artificial Intelligence, Vol. 16(4), pp. 476-481 (2001) (in Japanese)

[3] Hiroaki Arie, Tetsuya Ogata, Jun Tani, Shigeki Sugano: Reinforcement learning of a continuous motor sequence with hidden states, Advanced Robotics, Vol. 21(10), pp. 1215-1229, VSP, an imprint of Brill (2007)

[4] Stuart J. Russell, John Binder, Daphne Koller, Keiji Kanazawa, Tani Jun, Sugano Shigeki: Local learning in probabilistic networks with hidden variables, IJCAI, pp. 1146-1152 (1995)

[5] Sachiyo Arai, Kazuteru Miyazaki, Shigenobu Kobayashi: Methodology in Multi-Agent Reinforcement Learning: Approaches by Q-Learning and Profit Sharing, Journal of Japanese Society for Artificial Intelligence, Vol. 13(4), pp. 609-618, (1998) (in Japanese)

[6] Carlos Guestrin, Daphne Koller, Ronald Parr: Multiagent Planning with Factored MDPs, In 14th Neural Information Processing Systems (NIPS-14) (2001)

[7] Dale Schuurmans, Relu Patrascu: Direct value-approximation for factored MDPs, In 14th Neural Information Processing Systems (NIPS-14) (2001)

[8] Vijay R. Konda, John N. Tsitsiklis: Actor-Critic Algorithms, In 12th Neural Information Processing Systems (NIPS-12). MIT Press (2000)

[9] Kenji Doya: Reinforcement Learning in Continuous Time and Space, Neural Computation, Vol. 12(1), pp.219–245, (2000)

[10] S. Arai, Sycara, K., and Payne, T.R.,Experience-based reinforcement learning to acquire effective behavior in a multi-agent domain, Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence, pp. 125–135, (2000)