SICE Journal of Control, Measurement, and System Integration, Vol. 0, No. 0, pp. 001–009, xxx 2006

Adaptive Design of Role Differentiation by Division of Reward Function in Multi-agent Reinforcement Learning

Tadahiro TANIGUCHI *Kazuma TABUCHI **Tetsuo SAWARAGI **

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), which partially solved 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 enables a POMDP task for each agentto be transformed into a simple MDP task under certain conditions. Several situational experiments are also described and the validity of the proposed method is evaluated.

Key Words : Multi-agent reinforcement learning, organization, role differentiation, POMDP

1. Introduction

Not only biological organizations but also human organizations, e.g., a company, a project team, or a community, adapt to their environments through interacting with them [1]. The concept of "organization" involves an organization structure. An organization structure defines each participant's task expected to be achieved by him/her. Such a kind of task is called role of the participant. From this perspective, an organization structure in a human organization is originally based on role differentiations. However, the relationship between role differentiation and task environment has not rarely been discussed from the computational viewpoint, especially in association with learning processes. An organizational learning process can be abstractly described as a multi-agent learning process. Therefore, we discuss the differentiation in an organization based on multi-agent reinforcement learning framework.

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 behaviors. Because of 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 proposed many ways to partially resolve those [2].

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 attributed to a result of a multi-agent learning task. However, the division of roles are generated not only in a bottom-up way, but also in a top-down way in human organizations. We describe a method which enables an agent who manages an organization to rationally divide participating agent's roles, and examine 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 manager 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 different from supervised learning methods. Therefore, the learner has to modify its policy by trial and error [3].

Most reinforcement learning methods assume that the envi-

* Department of Human and Computer Intelligence, Ritsumeikan University, Shiga, Japan ** Deptartment of Mechanical Engineering and Science Kyoto University, Kyoto, Japan
E-mail: taniguchi@ci.ritsumei.ac.jp
(Received xxx 00,)
(Revised xxx 00,)

ronmental 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 dynamics 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 $\mathbb{X}^G = \mathbb{X}^1 \otimes \mathbb{X}^2 \otimes \cdots \otimes \mathbb{X}^N$, $\mathbb{U}^G = \mathbb{U}^1 \otimes \mathbb{U}^2 \otimes \cdots \otimes \mathbb{U}^N$ where N is the number of agents. 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 derives $\mathbf{u}^i$ 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 $\theta_G = \{\theta_1, \theta_2, \cdots \theta_N\}$.

Konda [4] formulated a reinforcement learning task as a problem in which an agent maximizes the following target evaluation function $\overline{\alpha}^G$.

$$\overline{\alpha}^G\left(\theta^G\right) = \int_{\mathbb{X}^G, \mathbb{U}^G} r^G\left(\mathbf{x}^G\right) \eta^G_{\theta_G}\left(\mathbf{x}^G, \mathbf{u}^G\right) d\mathbf{x}^G d\mathbf{u}^G. \quad (1)$$

$$\theta*^G = \underset{\theta_G}{\text{argmax}}(\overline{\alpha}^G\left(\theta^G\right)) \quad (2)$$

where $\eta^G_{\theta_G}$ is a stationary distribution of all the participating agents who take a global policy $\mu^G_{\theta^G}$ as a whole. 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 call $\mu^G_{\theta*^G}$ global optimal policy. However, a problem is that each agent's resource for perception, action, computation and experience are limited. Therefore, a problem the i-th agent solves becomes a following problem.

$$\overline{\alpha}^i\left(\theta^i\right) = \int_{\mathbb{X}^i, \mathbb{U}^i} r^i\left(\mathbf{x}^i\right) \eta^i_{\theta_i}\left(\mathbf{x}^i, \mathbf{u}^i\right) d\mathbf{x}^i d\mathbf{u}^i. \quad (3)$$

$$\theta*^i = \underset{\theta_i}{\text{argmax}}(\overline{\alpha}^i\left(\theta^i\right)) \quad (4)$$

Additionally, even if an agent can observe global reward, it does not usually help him to learn an adequate behavior as long as his perception and action are limited. This problem is also mentioned in 2.3.3.

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 often occurs in a multi-agent reinfocement learning domain. 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. They can be classified 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 assumes agent be able to estimate the hidden variables, thus the MDP is recovered [5],[6].

A method utilizing profit-sharing, which has a certatin robustness when the transition probability involves much uncertainty, can be considered as a kind of the first approach [7]. Arie [5] proposed a reinforcement-learning method using recurrent neural network (RNN). An RNN keeps some past information and utilizes it to identify 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, a total number of state spaces increases enormously because a participating agent has to take all the other agents' states into consideration to decide its outputs and policy.

Another agent's intention often becomes a hidden variable and causes a partial observation problem in multi-agent reinforcement learning domain. Modular learning architecture including multiple state predictors enables a learning agent to discriminate another's intention and to achieve a cooperative task[8] or a competitive task[9].

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 influenced by the other agents' states and actions. In fact,

$$p\left(\mathbf{x}^{i\prime}|\mathbf{x}^G, \mathbf{u}^G\right) = p\left(\mathbf{x}^{i\prime}|\mathbf{x}^G, \mathbf{u}^i, \left\{\mu^j_{\theta_j}, j \neq i\right\}\right). \quad (5)$$

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.Additionally, even if the transition probabilities are independent, changes in other agent's policies will affect an agent's obtaining reward r^i when r^i depends on the other agents' action and state.

Ikenoue et al. proposed decentralized scheduling method which make each learning agent update policy[10]. They showed that such a asynchronous policy renewal scheme can avoid a concurrent learning problem in some cases.

2.3.3 Credit assignment problem

It is difficult for agents participating in a multi-agent reinforcement learning problem to learn based only on the global reward r^G. If $r^i = r^G$ and 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.

2.4 Utilization of constraints in a task domain

As we mentioned, there are several problems in multi-agent reinforcement learning domain. However, if there are several constraints in the target domain, some of the former problems can be reduced by utilizing the constraints. For example, if the target system can be described as a factored MDPs, Schuurmans et al. proved that they can reduce the learning cost[11]. 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 affected on the node. Therefore, each node effectively acquires an optimal policy. By assuming each node in the network to be an agent in a multi-agent system, Guestrin et al.[12] proposed an effective multi-agent reinforcement learning method. However, Guestrin's approach is actually a top-down approach. We are focusing on the emergence or generation of role differentiation in an autonomous agent system. Here, emergence of role differentiation means that an organization decides what elemental task each participant should achieve and generates each agent's task internally. This also means that an organization decides each prticipating agent's reward function automatically based on the global reward function experienced through interaction. To treat this problem, we assume that there is an agent who can observe all information of the multi-agent system in this paper. We call this agent a "manager '' in the organization.

In this paper, we treat a multi-agent system, whose participants are dynamically independent, but their action results are evaluated as a whole by a single global reward function r^G. We focus on the division of a reward function, and discuss the design method of individual reward function for each learning agent r^i under the condition that each agent is dynamically independent and the global reward function is potentially dividable. As we describe in the next section, our method assumes the reward function is locally multiplicative. This means that the target reward function is assumed to be goal-oriented positive reward function. Therefore, periodical movement where there's no goal and navigating with collision avoidance where negative reward should be taken into account are not targets of our proposed method. We focus on a goal oriented cooperative multi-agent reinforcement learning task.

There are several works relating our idea of dividing reward. However, to overcome the credit assignment problem, most previous works focus on how to share incoming reward under the condition that the sum of the distributed rewards is same as the incoming reward, e.g., distributed value function, distributed reward function and factored MDP [12],[13]. In other words, they assumed an additive reward function. In contrast, our proposed method treats multiplicatively dividable reward function. This paper shows that theoretically sound reward decomposition can be performed when we presume that each agent learns based on actor-critic method [4].

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 though the optimal performance of the organization might become a little worse.

3.2 Division of rewards under the condition of independent transition probability

If the following 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.

$$p\left(\mathbf{x}^{G\prime}|\mathbf{x}^G,\mathbf{u}^G\right) = \prod_i p\left(\mathbf{x}^{i\prime}|\mathbf{x}^i,\mathbf{u}^i\right), \tag{6}$$

$$\mu^G_{\theta_G}\left(\mathbf{u}^G|\mathbf{x}^G\right) = \prod_i \mu^i_{\theta_i}\left(\mathbf{u}^i|\mathbf{x}^i\right). \tag{7}$$

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

$$\begin{aligned} p_{\theta_G}\left(\mathbf{x}^{G\prime},\mathbf{u}^G|\mathbf{x}^G\right) &= p\left(\mathbf{x}^{G\prime}|\mathbf{x}^G,\mathbf{u}^G\right)\mu^G_{\theta_G}\left(\mathbf{u}^G|\mathbf{x}^G\right) \\ &= \prod_i p\left(\mathbf{x}^{i\prime}|\mathbf{x}^i,\mathbf{u}^i\right)\prod_i \mu^i_{\theta_i}\left(\mathbf{u}^i|\mathbf{x}^i\right) \\ &= \prod_i p\left(\mathbf{x}^{i\prime}|\mathbf{x}^i,\mathbf{u}^i\right)\mu^i_{\theta_i}\left(\mathbf{u}^i|\mathbf{x}^i\right) \\ &= \prod_i p_{\theta_i}\left(\mathbf{x}^{i\prime},\mathbf{u}^i|\mathbf{x}^i\right). \end{aligned} \tag{8}$$

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

$$\eta^G_{\theta_G}\left(\mathbf{x}^G,\mathbf{u}^G\right) = \prod_i \eta^i_{\theta_i}\left(\mathbf{x}^i,\mathbf{u}^i\right), \tag{9}$$

Therefore, if the global reward function r^G satisfies

$$r^G\left(\mathbf{x}^G\right) = \prod_i r^i\left(\mathbf{x}^i\right), \tag{10}$$

(3) becomes

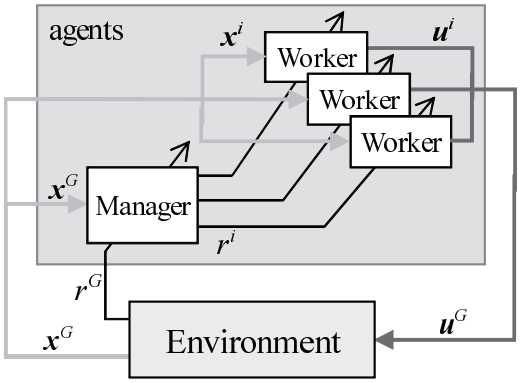


Fig. 1 Organization of agents

$$\bar{\alpha}^i\left(\theta^i\right) = \int_{\mathbb{X}^i,\mathbb{U}^i} r^i\left(\mathbf{x}^i\right)\eta^i_{\theta_i}\left(\mathbf{x}^i,\mathbf{u}^i\right)d\mathbf{x}^i d\mathbf{u}^i, \tag{11}$$

$$\bar{\alpha}^G\left(\theta^G\right) = \prod_i \bar{\alpha}^i. \tag{12}$$

The target multi-agent task is divided into partial problems. For each problem, each agent is required to maximize its obtaining indivisual reward (11). 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 (12). By replacing r^G with a tentative internal reward r^M, we can divide a reward function based on (12). The internal reward r^M is generated in order to determine respective reward function r^i. 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 interacts 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 $\left\{r^i\right\}$ 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.

$$g\left(\mathbf{x};\mu,S\right) = \exp\left\{-\frac{1}{2}\left(\mathbf{x}-\mu\right)^T S\left(\mathbf{x}-\mu\right)\right\}, \tag{13}$$

where S is a positive symmetric matrix.

$$r^G\left(\mathbf{x}^G\right) = \sum_j f_j\left(\mathbf{x}^G\right), \tag{14}$$

$$f_j\left(\mathbf{x}^G\right) = w_j g\left(\mathbf{x}^G;\mu_j,S_j\right), \tag{15}$$

where $\left\{w_j,\mu_j,S_j\right\}$ 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.

$$j^* = \operatorname*{argmax}_j w_j \tag{16}$$

$$r^M = w_{j^*} g(x^G;\mu_{j^*},S_{j^*}) \tag{17}$$

The r^M satisfies the condition of (12), 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 $h(x;\theta)$, 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, $\rho(\varepsilon_t)$ is introduced.

$$J = \sum_t \rho(\varepsilon_t) \tag{18}$$

The M-estimation aims to minimize J. If we assume $\rho(\varepsilon_t) = \|\varepsilon_t\|^2$, 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 possible definitions of ρ. In this paper, we used the biweight method. The ρ function is described as follow.

$$\rho(\varepsilon_t) = \begin{cases} \frac{c^2}{6}\left\{1-\left[1-\left(\frac{\varepsilon_t}{c}\right)^2\right]^3\right\} & (|\varepsilon_t| \le c) \\ \frac{c^2}{6} & (|\varepsilon_t| > c) \end{cases}, \tag{19}$$

where c is a parameter.

We made a single-peaked function which approximates a global reward function near μ_{j^*} by using the biweight method.

$$\varepsilon_t = r^G_t - r^M, \tag{20}$$

$$r^M = wg\left(\mathbf{x}^G;\mu,S\right), \tag{21}$$

$$r^i = w^i g\left(\mathbf{x}^i;\mu^i,S^i\right), \tag{22}$$

where $\theta = \{w,\mu,S\}$ 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

$$\frac{\partial J}{\partial\theta} = \sum_t \frac{\partial\rho}{\partial\varepsilon_t}\frac{\partial\varepsilon_t}{\partial\theta} = -\sum_t \frac{\partial\rho}{\partial\varepsilon_t}\frac{\partial r^M}{\partial\theta}. \tag{23}$$

Therefore, the updated rule becomes

Table 1 Parameters of task environment in experiment 1

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

Table 2 Parameters of workers in experiment 1

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 in experiment 1

β	0.1
c	1
K	10
initialize	500
interval	10

$$\theta_{\tau+1} = \theta_\tau + \beta \sum_{t=1}^{D_\tau} \frac{\partial \rho}{\partial \varepsilon_t} \frac{\partial r^M}{\partial \theta_\tau}, \tag{24}$$

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

$$\frac{\partial \rho}{\partial \varepsilon_t} = \begin{cases} \varepsilon_t \left\{ 1 - \left(\frac{\varepsilon_t}{c} \right)^2 \right\}^2 & (|\varepsilon_t| \le c) \\ 0 & (|\varepsilon_t| > c) \end{cases} \tag{25}$$

However, the obtained data, whose residuals are large, rarely have 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 $t = initialize$. *initialize* is a time constant that determines how long the manager search r^M globally before it starts local search. In the global search, the maximal reward $r^G_{\max}$ and the state $\mathbf{x}^G_{\max}$ at that time are memorized, and the parameters are updated as $w \leftarrow r^G_{\max}, \mu \leftarrow \mathbf{x}^G_{\max}$ at $t = initialize$. Additionally, to stabilize the learning process, the manager stores the incoming data to its 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^M can be used to finally approximate a part of the target reward function around the specific peak of that function.

4. Experiment 1

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

4.1 Assumptions

In this experiment, two workers were required to reach given goals (Fig. 2). There were two goals, and they were required to reach respective goals, i.e., if the first agent reached 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 kind of representative problem in multi-agent reinforcement learning. If an agent cannot observe other

[1] To apply the biweight method to obtain r^M, the function is differentiable by its parameters.

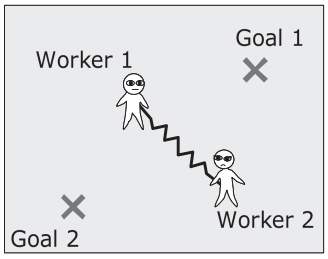
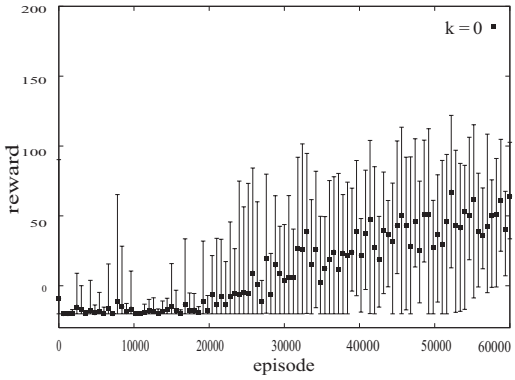
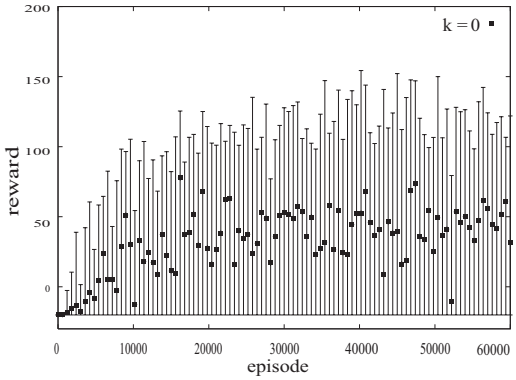


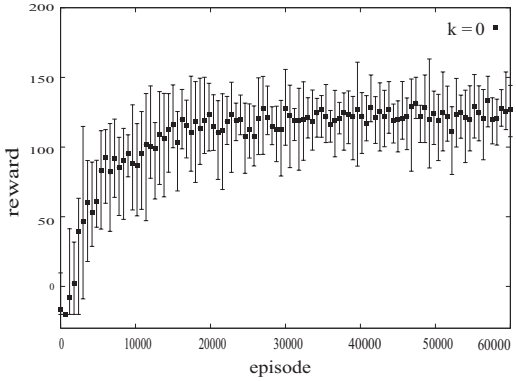
Fig. 2 Overview of the task environment



(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

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, this agent cannot obtain the reward until the other agent reaches another goal. Therefore, credit assignment problems are also taken into account in this simple task. This problem is not so concrete and practical multi-agent reinforcement learning problems, e.g. soccer and card games. However, to understand role differentiation in an organization constructively, treating such a theoretically tractable simple problem is important.

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

$$\mathbf{x}^G_{t+1} = \mathbf{x}^G_t + \mathbf{u}^G_t \Delta t, \tag{26}$$

where $\mathbf{u}^G = (\mathbf{u}^{1^T}\ \mathbf{u}^{2^T})^T$ 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 a single episode.

The global reward function r^G is defined as

$$\begin{aligned} r^G = \zeta \Big\{ & g(\mathbf{x}^1; \xi_1, A_1)\, g(\mathbf{x}^2; \xi_2, A_2) \\ & + g(\mathbf{x}^1; \xi_2, A_2)\, g(\mathbf{x}^2; \xi_1, A_1) \Big\} + \eta. \end{aligned} \tag{27}$$

When the two workers remain at ξ_1 and ξ_2 respectively, the total obtained reward is maximized. Therefore, $\mathbf{x}^G$ finally reaches either $(\xi_1^T\ \xi_2^T)^T$ or $(\xi_2^T\ \xi_1^T)^T$ under the condition that the workers have their optimal policies.

The parameters of the task environment are defined in table 1. As readers can easily notice, the reward function shown in Eq. 27 does not satisfy the condition of Eq. 10. A manager agent approximate the global reward function locally by using M-estimation described in section 3.5, and obtain a dividable reward function r^M.

The workers are engaged in a reinforcement-learning task by using the actor-critic method [15], which is an on-policy reinforcement learning method. Each worker has two approximators for policy and state value functions as following.

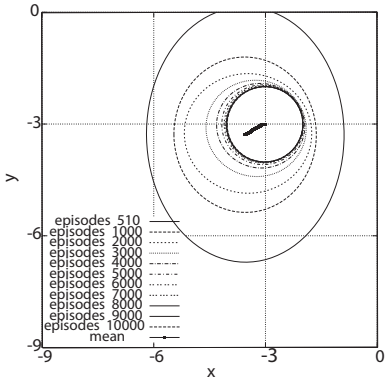
$$\mathbf{u}^i\left(\mathbf{x}^i\right) = \sum_{j=1}^{B} \phi^i_j\, g(\mathbf{x}^i; \nu^i_j, A^i_j) \tag{28}$$

$$V^i\left(\mathbf{x}^i\right) = \sum_{j=1}^{B} \psi^i_j\, g(\mathbf{x}; \nu^i_j, A^i_j) \tag{29}$$

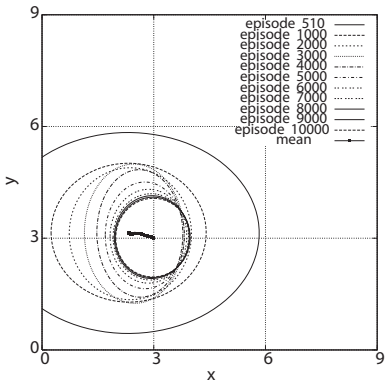
The workers optimize these functions under the reward functions r^i designed by the managers. The exploration noise was produced by using a normal distribution.

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 $\left(\mathbf{x}^G, r^G\right)$ 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^i_j 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, and table 3.



(a) worker 1



(b) worker 2

Fig. 4 Reward functions given the corresponding workers

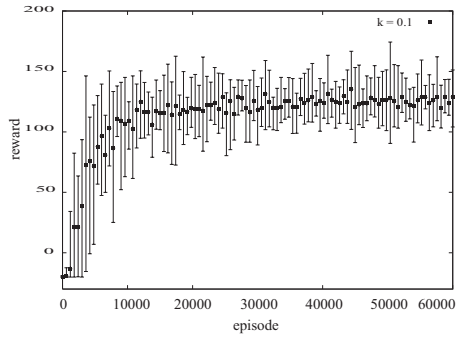
Two other settings are prepared for comparison. In the first setting, workers observed other agent's state variables, i.e., position, and obtained a global reward ($r^i = r^G, \mathbf{x}^i = \mathbf{x}^G$). We call this setting "fully observable agents with global reward function". In the second setting, 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 reward function". In the first case, the dimension of the state space is 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

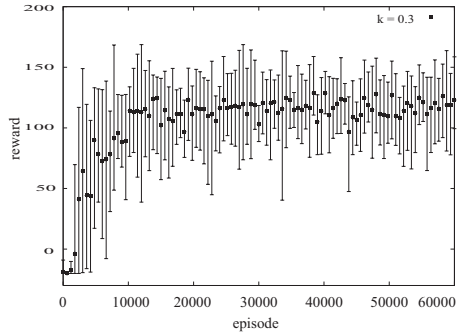
Figure 3 shows the averaged accumulated reward for each condition. The horizontal axis shows the number of episodes, and the vertical axis shows the cumulative global reward obtained through an episode. The cumulative reward is the averaged value of ten trials. The error bar represents the best and the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

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

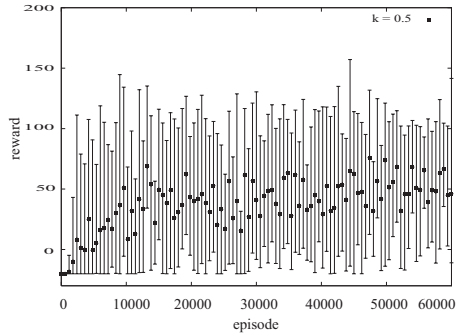
Moreover, fully observable agents does not score well. This



(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

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.

$$\mathbf{x}^i_{t+1} = \mathbf{x}^i_t + \left\{\mathbf{u}^i_t + k\left(\mathbf{x}^j_t - \mathbf{x}^i_t\right)\right\}\Delta t \tag{30}$$

Wherein, 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. This result shows that our proposed method cannot work well in a multi-agent system where participating agents are tightly connected, i.e., not dynamically independent. This result is natural considering our assumption.

5. Experiment 2

In the experiment of the section 4, we assumed that workers can obtain only internal rewards generated by a manager. However, if the manager estimates the global reward function wrongly or generates not so adequate internal reward function r^i, the workers' effort of reinforcement learning will become no use. Therefore, our proposed approach heavily depends on how accurate a manager can acquire a model of a global reward function. This high dependency to a manager's capability of learning and dividing a global reward function means that the total learning system is not so "robust".

To avoid such a problem, in human organization, workers themselves also observe the external environment and situations to behave so as to make the organization's performance better. This corresponds to that worker agents also observes global reward and learn to maximize both internal reward r^i and external reward r^G simultaneously in a multi-agent reinforcement learning environment.

Here, we define a mixed reward functions $\overline{r}^i$ as below.

$$\overline{r}^i = \rho r^G + (1-\rho) r^i \quad (0 \le \rho \le 1), \tag{31}$$

$\rho = 1$ corresponds to partially observable agents with a global reward function, and $\rho = 0$ corresponds to partially observable agents with a divided reward function in the experiment of section 4, respectively. ρ is a parameter which represent to what extent workers pay attention to global reward rather than internal reward given by its manager.We executed experiments based on the mixed reward function.

5.1 Task environment

The target task is almost same with that of the previous section. We increased the number of workers from 2 to 3. The environmental dynamics is

$$\boldsymbol{x}^i_{t+1} = \boldsymbol{x}^i_t + \left\{\boldsymbol{u}^i_t + \sum_{j\neq i} k_{ij}\left(\boldsymbol{x}^j_t - \boldsymbol{x}^i_t\right)\right\}\Delta t \tag{32}$$

where k_{ij} represents a strength of a virtual spring which connects worker i and j. A global function r^G was defined as

$$r^G = \zeta \sum_{i,j,k=1}^{3} \epsilon_{ijk}\, g(\boldsymbol{x}^i; \boldsymbol{\xi}_1, A_1)\, g(\boldsymbol{x}^j; \boldsymbol{\xi}_2, A_2)\, g(\boldsymbol{x}^k; \boldsymbol{\xi}_3, A_3) + \eta,$$

$$\epsilon_{ijk} = \begin{cases} 1 & (if(i \neq j) \wedge (j \neq k) \wedge (k \neq i))) \\ 0 & (otherwise) \end{cases} \tag{33}$$

where g is a function defined in (13). The parameters were defined as table 4. In this experiment, we also checked scalability of the proposed method.

Internal reward function r^i was generated at after *initialize* episodes passed. The other parameters were set as shown in table 6. They were almost similar to the previous experiment.

5.2 Result

We executed simulation experiments under 30 ($= 5 \times 6$) conditions , i.e., $initialize \in \{1000, 2000, 3000, 4000, 5000\}$ and

Table 4 Parameters of task environment in experiment 2

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

Table 5 Parameters of workers in experiment 2

A^i_j	$\begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix}$
Learning rate α	0.1
Discount factor γ	0.9
Variance of the exploration noise	0.5

Table 6 Parameters of manager in experiment 2

c	1
β	0.1
K	10
interval	10

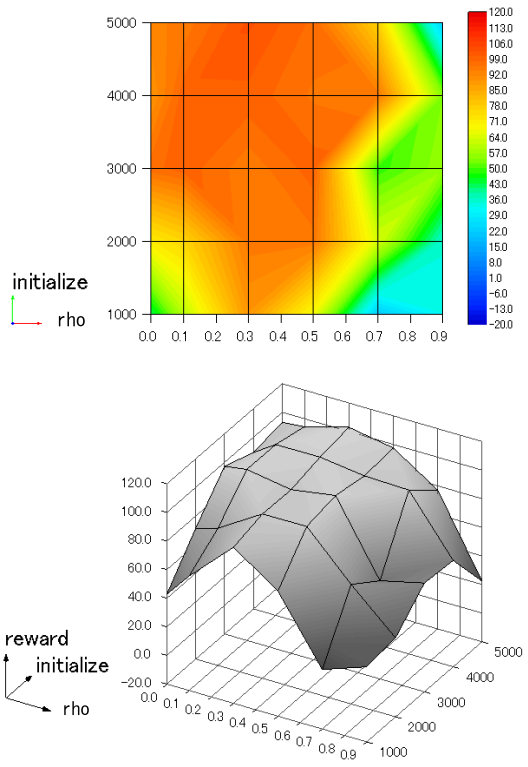


Fig. 6 Averaged reward obtained in the last 1000 episodes.

$\rho \in \{0.0, 0.1, 0.3, 0.5, 0.7, 0.9\}$. In this experiment, strength of virtual springs connecting agents each other were set $k_{12} = 0.1, k_{23} = 0.5, k_{31} = 0.0$. 10 experiments were executed for each condition, and each experiment continued until 10000 episodes passed. Averaged reward in the last 1000 episodes were evaluated. The averaged reward for each *initialize* and ρ are shown in Fig. 6 .

Fig. 6 shows that $\rho = 0$ was not the best way in a practical environment. In a practical environment, a manager has to adaptively design a role differentiation by dividing estimated global reward function during a finite period of time. If *initialize* is big enough, a manager can search globally and generate an adequate r^M and r^i, deliberately. However, it also delays the time for workers to start their learning processes based on organizational internal reward. In this case, $\rho = 0.3, initialize = 5000$ output the best performance.

This figure shows that there was the best mix of internal reward r^i generated by a manager and global reward r^G. In addition to that, the peak appeared clealy when *initialize* is small, i.e., manager's estimation of global reward function is often not accurate.

Theoretically, workers should not take r^G into consideration because learning process based on r^G becomes unstable as we see in section 4. However, a manager does not always succesfully generate internal reward functions. If we take this into consideration, workers should take r^G into consideration to some extent. The worker's consideration of external environment will make organization's leraning process more robust.

6. Conclusions

We described adaptive design of role differentiation in multi-agent reinforcement learning. The role differentiation is provided to avoid several problem in multi-agent reinforcement learning, e.g., partial observation problem, concurrent learning problem, and credit assignment problem. In our 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.

In the experiments, we took a task in which each agent climb up a single-peak reward function for example. This is a more restricted class of problems formalized in section 3. The proposed basic idea is not restricted to such a simple task. However, to apply this method to another complicated task, the function type of internal reward r^M should be modified depending on its target global reward function r^G.

A mixed reward $\vec{r^i} = \rho r^G + (1-\rho) r^i \quad (0 \le \rho \le 1)$ is also used and the mixed reward was proved to make the multi-agent learning process more robust in certain cases.

However, the method of designing the multiplicative global reward function from usual global reward which does not satisfy the condition 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 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 is not effective 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. We have to mention again that our approach solves only a small class of multi-agent reinforcement learning problem. To build an effective taxonomy of multi-agent reinforcement learning task is also an ongoing problem.

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] A.D. Chandler, Jr.: Strategy and structure, MIT Press, 1962

[2] S. Arai: Multiagent reinforcement learning frameworks: steps toward practical use, *Journal of Japanese Society for Artificial Intelligence*, Vol. 16, No. 4, pp. 476-481 , 2001, (in Japanese)

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

[4] V. R. Konda, J. N. Tsitsiklis: Actor-Critic algorithms, *12th Neural Information Processing Systems (NIPS-12)*, MIT Press , 2000

[5] H. Arie, T. Ogata, J. Tani, S. Sugano: Reinforcement learning of a continuous motor sequence with hidden states, *Advanced Robotics*, Vol. 21, No. 10, pp. 1215-1229, 2007

[6] S. J. Russell, J. Binder, D. Koller, K. Kanazawa: Local learning in probabilistic networks with hidden variables, *International Joint Conferences on Artificial Intelligence*, pp. 1146-1152, 1995

[7] S. Arai, K. Sycara, T.R. Payne: 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

[8] T. Taniguchi, K. Ogawa, and T. Sawaragi: Implicit estimation of another’s intention based on modular radeinforcement learning, Machine Learning, pp.381-400, In-Tech, 2009

[9] Y. Takahashi, K. Edazawa, M. Asada: Modular learning system and scheduling for behavior acquisition in multi-agent environment, *RoboCup 2004 Symposium team description*, 2004

[10] S. Ikenoue, M. Asada, K. Hosoda: Cooperative behaviour acquisition by asynchronous policy renewal that enables simultaneous learning in multiagent nnvironment, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.2728-2734, 2002

[11] D. Schuurmans, R. Patrascu: Direct value-approximation for factored MDPs, *14th Neural Information Processing Systems (NIPS-14)* , 2001

[12] C. Guestrin, D. Koller, R. Parr: Multiagent planning with factored MDPs, *In 14th Neural Information Processing Systems (NIPS-14)* , 2001

[13] J. Schneider, W.K. Wong, A. Moore, M. Riedmille: Distributed value functions, *In Proceedings of the Sixteenth International Conference on Machine Learning*, 1999

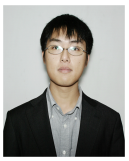
[14] S. Arai, K. Miyazaki, S. Kobayashi: Methodology in multiagent reinforcement learning: approaches by q-Learning and profit sharing, *Journal of Japanese Society for Artificial Intelligence*, Vol. 13, No. 4, pp. 609-618, 1998, (in Japanese)

[15] K. Doya: Reinforcement learning in continuous time and space, *Neural Computation*, Vol. 12, No. 1, pp.219-245, 2000

Tadahiro TANIGUCHI (Member)

He received his B.S., M.S., and Ph.D. degrees from Kyoto University, Japan, in 2001, 2003, and 2006, respectively. From April 2005 to March 2006, he was a Japan Society for the Promotion of Science (JSPS) Reserach Fellow (DC2) at the Department of Mechanical Engineering and Science, Graduate School of Engineering, Kyoto University. From April 2006 to March 2007, he was a JSPS Reserach Fellow (PD) at the same department. Since April 2007, he is a JSPS Research Fellow at the Department of Systems Science, Graduate school of Informatics, Kyoto University. In 2008, he joined the faculty of Ritsumeikan University, where he is currently an Assistant Professor of Department of Human and Computer Intelligence. His research interests include symbol emergence, complex system, and machine learning. He is a member of ISCIE, JSAI, and JNNS.

Kazuma TABUCHI



He received his B.S., and M.S. degrees from Kyoto University, Japan, in 2006, and 2008, respectively. From 2008, Currently, he is working at Yaskawa Electric Corporation. His research interests include machine learning and robot intelligence. He is a member of JSME.

Tetsuo SAWARAGI (Member)

He received his B.S., M.S. and Ph.D. degrees in Systems Engineering from Kyoto University in 1981, 1983 and 1988, respectively. From 1986 to 1994, he was a Research Associate at the Department of Precision Mechanics, Faculty of Engineering, Kyoto University, wherein he was an Associate Professor and a Professor in 1994 and 2002, respectively. In 2005 he was in the current department as a Professor. He is now a Professor at the Department of Mechanical Engineering and Science, Graduate School of Engineering, Kyoto University, From 1991 to 1992, he was a Visiting Scholar at the Department of Engineering-Economic Systems, Stanford University, USA. He has been engaged in research on systems engineering, cognitive science and artificial intelligence, particularly in the development of human-machine collaborative systems, modeling the transfer of human cognitive skills into machines. He was a chair of IEEE SMC Japan Chapter and a Board Member of the Institute of Systems, Control and Information Engineers, Human Interface Society and Japan Society for Fuzzy Theory and Systems. He is a member of the Japanese Society for Artificial Intelligence, JSME and IEEE.