Greedy Action Selection and Pessimistic Q-Value Updating in Multi-Agent Reinforcement Learning with Sparse Interaction

Toshihiro KUJIRAI* and Takayoshi YOKOTA*

Abstract: Although multi-agent reinforcement learning (MARL) is a promising method for learning a collaborative action policy, enabling each agent to accomplish specified tasks, MARL has a problem of exponentially increasing state-action space. This state-action space can be dramatically reduced by assuming sparse interaction. We previously proposed three methods (greedily selecting actions, switching between Q-value update equations on the basis of the state of each agent in the next step, and their combination) for improving the performance of coordinating Q-learning (CQ-learning), a typical method for multi-agent reinforcement learning with sparse interaction. We have now modified the learning algorithm used in a combination of these two methods to enable it to cope with interference among more than two agents. Evaluation of this enhanced method using two additional maze games from three perspectives (the number of steps to a goal, the number of augmented states, and the computational cost) demonstrated that the modified algorithm improves the performance of CQ-learning.

Key Words: reinforcement learning, multi agent, sparse interaction, fully cooperative, maze games.

1. Introduction

Multi-agent reinforcement learning (MARL) is a promising method for learning a collaborative action policy, enabling each agent to accomplish specified tasks [1],[2]. Each agent tries to learn an optimal action policy, one that maximizes the expected cumulative rewards, while sharing the environment with other agents. Agents that learn their own action policy by taking the states and actions of other agents into account are called joint action learners. Those that learn it independently are called independent learners [3].

If each agent shares the same reward for a task, i.e., a fully cooperative task, independent learners can sometimes learn a collaborative action policy without taking the states and actions of other agents into account because a random exploration strategy may enable them to learn collaborative actions coincidentally [4],[5]. While joint action learners may perform better because they take information about other agents (sensation, episodes, learned policies, etc.) into account, they suffer an exponential increase in the state-action space for learning, thereby reducing the learning speed and increasing the cost of communication and estimation of information about other agents [6].

In some real-world tasks, agents may behave independently most of the time and sometimes must learn how to cooperate. For example, consider a task involving multiple robots working together to move a heavy box to a specific position. A rational approach is for them to independently approach it and then cooperatively move it in the same direction to the final position. Each should decide its actions taking other agents’ positions and actions into account only when they are close to each other.

The basic idea of MARL with sparse interaction is to reduce the state-action space by taking information about other agents into account only when necessary because a smaller state-action space makes the learning process more efficient. This means that identifying when cooperative actions are necessary is a key function in MARL with sparse interaction. Melo and Veloso [7] reported a method in which a pseudo action, COORDINATE, is added to the action space of each agent. The agents learn when they should take information about other agents into account by estimating the Q-value for the COORDINATE action for each state. Hauwere et al. [8],[9] proposed the coordinating Q-learning (CQ-learning) concept. Each agent determines when it should take information about other agents into account by comparing the immediate rewards in a single-agent environment with those in a multi-agent environment. In CQ-learning, the state-action space is partially augmented when an agent judges that it observes a difference in the immediate rewards using Student’s t-test.

We previously reported three methods for improving the performance of CQ-learning: greedily selecting actions (GCQ-learning), switching between Q-value update equations on the basis of the state of each agent in the next step (PCQ-learning), and a combination of both (GPCQ-learning) [10]. Evaluation using several maze games validated their effectiveness, especially that of GPCQ-learning.

We have now modified the learning algorithm of GPCQ-learning to enable it to cope with interference among more than two agents. Evaluation of this enhanced method using two additional maze games from three perspectives (the number of steps to a goal, the number of augmented states, and the computational cost) demonstrated that the modified algorithm improves the performance of CQ-learning.

The remainder of this paper is organized as follows. Section 2 gives an overview of MARL and discusses related work. Section 3 discusses MARL with sparse interaction. In Section 4 we describe our three methods and the modified algorithm for
coping with interference among more than two agents. Section 5 describes how we evaluated our methods using maze games and compares their performances with those of existing methods. We conclude in Section 6 with a summary of the key points and a look at future work.

2. Multi-Agent Reinforcement Learning

2.1 MDP and Reinforcement Learning

A Markov decision process (MDP) is formalized as a problem in which an agent optimizes its action policy by maximizing the expected cumulative rewards from its environment resulting from the actions it takes. The MDP is defined as a tuple \((S, T, R, \pi)\), where \(S\) stands for the state space of the agent, \(T = p(s'|s, a)\) and \(R = r(s, a, s')\) stand for the transition probability matrix and immediate reward matrix for combinations of state \(s\), action \(a\), and next state \(s'\), and \(\pi = p(a|s)\) stands for the action policy of the agent. The optimal policy, i.e., one that maximizes the expected cumulative reward, is described as \(\pi^*\).

Reinforcement learning is one method for iteratively estimating \(\pi^*\). Q-learning [11],[12] is a typical reinforcement learning method. Instead of estimating \(\pi^*\), Q-learning estimates the optimal Q-value \(Q^*\) by iteratively updating the Q-value using (1)

\[
Q(s, a) \leftarrow (1 - \alpha_t)Q(s, a) + \alpha_t[r(s, a) + \gamma \max_{a'}Q(s', a')].
\]

(1)

2.2 Extended Multi-Agent Systems

As shown in Table 1, an MDP can be extended for multi-agent systems in at least four ways. A natural extension is multi-agent MDP (MMDP), in which agents share all the system states (full observability) and rewards from the environment. The agents share all their states and actions and obtain the same rewards from the environment as a result of their joint actions [13].

Another extension is decentralized MDP (DEC-MDP), in which each agent can observe only its own states, and the

agents obtain the same rewards [14]. If they can know the complete state of the environment by sharing their observations, the agents are said to have full joint observability.

These two extensions are called fully cooperative games because the agents obtain the same rewards.

In contrast, in a Markov game (MG) and a decentralized Markov game (DEC-MG), each agent has an independent reward function. This results in a competitive situation [15].

2.3 Related Work

The focus here is on fully cooperative games, which have at least one optimal action policy for each agent. Even if all the agents must collaborate to perform optimally, they can serendipitously learn a cooperative behavior without having any information about the other agents. This is because coincidental actions that lead to a high reward are reinforced [5].

For example, assume that an agent randomly selects an action at a certain position in a maze game, and the action results in the agent obtaining a reward because performing the action coincidentally prevents the agent from colliding with another agent. The agent may thereby learn a cooperative action policy without having any information about the other agents because the selection of that action at that position is reinforced.

An agent learns a more precise cooperative action policy if it has knowledge of not only its own state-action combinations but also those of other agents. However, the resulting exponential increase in the state-action space and communication cost between agents slows down the learning process.

In maze games, each agent \(i\) tries to find an optimal path from start position \(S_i\) to goal \(G_i\). We used the five maze games shown in Fig. 1, which were previously used by Hauwere et al. [9]. In the TunnelToGoal and TunnelToGoal3 games, each agent has the same goal and starts from a different position. They collide
if they simply take the shortest path. At least one of them needs to wait while the others proceed. However, if they all wait, more steps will be needed to reach the goal. The optimal solution, i.e., one that minimizes the number of steps it takes for both agents to reach their goals, is for one agent to take the shortest path while another agent waits for the first agent to proceed for both games. In addition, the last agent needs to wait for the first two agents to proceed in the TunnelToGoal3 game. In the ISR, CIT, and CMU games, the goal for each agent is the start position of the other agent. Simply waiting will not prevent a collision. The optimal solution is for the agent taking a detour to do so immediately before a collision would occur. However, finding this solution requires extensive exploration of potential detours because there are a number of unsuitable detour routes.

Figure 2 shows the average number of steps needed to complete a game for every 100 episodes using three different existing methods for the five games. These methods are straightforward extensions of Q-learning for a multi-agent environment. The first method is independent-learning, which is Q-learning itself. Each agent acquires its own action policy without having any information about the other agents. The second one is joint-state learning (JSQ-learning), in which each agent knows the states of the other agents at all times and decides its actions independently on the basis of its own policy. The third is joint-state-action learning (JSAQ-learning), in which one super-agent observes all the states and decides a combination of actions for all the agents.

As shown in Fig. 2, even the agents trained using
3. Multi-Agent Reinforcement Learning with Sparse Interaction

In some fully cooperative games, agents can decide their actions without knowing any information about the other agents for most states. That is, an independent action policy might be optimal for most states. In the other states, each agent needs information about the other agents in order to coordinate. Therefore, an agent should take into account information about the other agents only when necessary in order to avoid exponential increases in the state-action space and in the communication cost. This type of framework is called a decentralized sparse interaction MDP (DEC-SIMDP) [14] and is a special case of DEC-MDP. Several methods have been proposed for agents to learn their action policy for DEC-SIMDP.

As mentioned in the introduction, Melo and Veloso [7] suggested adding a pseudo-action, COORDINATE, to the action space of each agent. When COORDINATE is selected as an action by an agent, the agent obtains information about the other agents and behaves on the basis of that information while the agent takes a penalty as a communication cost. Because the Q-value for selecting COORDINATE can be obtained with Q-learning, the agent can decide when to take the other agents into account. Setting the cost of COORDINATE for a specific game is a difficult issue because, if the cost is zero, agents will always choose COORDINATE, and, if the cost is too high, the agents will seldom choose it.

Hauwere et al. [8],[9] proposed a method in which the state of an agent is augmented to include the state of another agent if the two agents are likely to interfere with each other. Each agent behaves on the basis of Q-values learned in advance in a single-agent environment. Each agent can identify potential interference with other agents by comparing the distribution of a state’s immediate rewards to those for a single-agent environment. Once the state is augmented, the agent selects an action on the basis of Q-values corresponding to the augmented joint state when the agent and another agent are in an augmented joint state. This method is called CQ-learning.

To be more specific, CQ-learning augments the state of agent \( k \) to \( s_k = (s_k, s_i) \), creating augmented joint state \( s_k^r = (s_k, s_i) \) when Student’s t-test rejects the hypothesis that the distribution of immediate rewards in the state comes from that for a single-agent environment.

This partial augmentation of joint states dramatically reduces the state-action space in sparse interaction games compared to JSQ-learning and JSAQ-learning, and it improves the efficiency and optimality of the learned action policy.

4. Proposed Methods

Because agents trained using CQ-learning always \( \epsilon \)-greedily select an action, they may take random actions even if no interference with another agent is likely to occur. This causes unnecessary exploration and interferences. In addition to that, taking a random action might coincidently prevent interference with another agent, resulting in a lost opportunity for the agent to identify the difference between a single-agent environment and multi-agent environment.

Moreover, CQ-learning optimistically updates the Q-values of an augmented joint state assuming that, after taking the selected action, the agent can behave on the basis of independent Q-values without subsequent interference. This optimistic assumption may cause subsequent collisions.

4.1 GCQ-, PCQ-, AND GPCQ-Learning

To improve the performance of CQ-learning, we previously proposed three methods: (i) greedily selecting actions, (ii) switching between Q-value update equations on the basis of the state of the agent in the next step, and (iii) application of both [10].

In our greedy coordinating Q-learning (GCQ-learning) method, an action is greedily selected when it uses independent Q-values while an action is \( \epsilon \)-greedily selected when it uses Q-values of an augmented joint state. This helps each agent more quickly identify the states in which it should take the information of the other agents into account and thereby achieve more efficient learning.

In our pessimistic coordinating Q-learning (PCQ-learning) method, switching between update equations for the Q-values of the augmented state is based on whether the agent will be in an interference state with another agent in the next step. Although the assumption in CQ-learning that an agent will not be in an interference state in the next step seems rational due to the sparse interaction assumption, it is, in fact, too optimistic because the probability of subsequent interference is too high to ignore. Our PCQ-learning method prevents optimistic estimation of the Q-values and properly evaluates the effect of subsequent interference, resulting in more accurate estimation
of optimal Q-values.

Application of both GCQ-learning and PCQ-learning together is referred to as GPCQ-learning. We previously showed that GPCQ-learning achieves the best performance for mazes (a), (b), and (c) [10].

4.2 Modified Algorithm of GPCQ-Learning for More Than Two Agents

When more than two agents are in an interference state, each agent has multiple candidates for $Q^\text{aug}_k(s_k, a_k)$. Hauwere [8],[9] and we [10] did not explicitly describe how an agent selects an action on the basis of multiple candidates for $Q^\text{aug}_k(s_k, a_k)$.

We have now modified the GPCQ-learning algorithm to cope with interference among more than two agents. In the modified algorithm, agent $l$ is selected randomly from the set of $\mathcal{I}_l = (s_l, s_l)$ when more than two agents are in an interference state. This modification is not expected to improve the performance of GPCQ-learning because interference among more than two agents must be rare due to the assumption of sparse interaction.

The modified GPCQ-learning algorithm is shown in Algorithm 1. The differences from CQ-learning are indicated by underlining; $E(R_k(s_k, a_k))$ stands for the expected values of the immediate rewards when agent $k$ takes action $a_k$ in state $s_k$ in a single-agent environment, and $W_k(s_k, a_k)$ stands for samples of the immediate rewards when agent $k$ takes action $a_k$ in state $s_k$ in a multi-agent environment.

5. Evaluation

In this study, we evaluated the previously proposed GCQ-learning and PCQ-learning methods and the modified GPCQ-learning method using a more complex maze (the CMU game) and a maze game with three agents (the TunnelToGoal3 game) as well as the three previously used maze games (the TunnelToGoal, ISR, and CIT games). All the five games (Fig. 1) were used by Hauwere et al. [9]. The start position for agent $k$ is indicated by $S_k$, and the goal for agent $k$ is indicated by $G_k$. An episode ends when all the agents have reached their goals; each agent then receives a reward of $+10$. Each agent receives a reward of $-1$ both for a successful movement and for a collision with a wall. An agent cannot move after colliding with a wall. If multiple agents try to move to the same position or cross each other’s paths, each agent receives a reward of $-10$ due to the colliding with another agent. They then return to their previous position. Once an agent reaches its goal, it waits for the other agents to reach their goals and does not receive a reward until the episode ends.

For the agents trained using CQ-, GCQ-, PCQ-, and GPCQ-learning, the length of the window used to calculate the distribution of immediate rewards was set to 20. Student’s $t$-test was performed for the distribution between the immediate rewards in a single-agent environment and in a multi-agent environment only when 20+ immediate reward samples were obtained for a state in a multi-agent environment. The threshold of the $t$-test, $p_{th}$, was set to 0.01, as was done by Hauwere et al. [8],[9]. Only when the null hypothesis (the mean of the immediate reward samples in a certain state in a multi-agent environment is the same as the expected immediate rewards in the same state in a single-agent environment) was rejected was the state augmented because it could be an interfered state.

CQ-learning and our proposed methods require pre-learning of independent Q-values for each agent. For each game, all the agents first learned the Q-values $\varepsilon$-greedily in a single-agent environment for 200,000 episodes with $\varepsilon=0.3$ to ensure that the agents could sufficiently explore the environment. An agent using CQ-learning and our proposed methods uses independent Q-values to select an action. Because all the mazes were designed so that the agents would interfere with each other if they selected their actions on the basis of independent Q-values learned in a single-agent environment, the agents were likely to collide as they made their way towards their goals.

As shown in Table 2, the number of state-action combinations increased exponentially with the number of states and Table 2. Number of state-action combinations in games.

| Game                  | No. of states | JSQ | JSAQ | SI |
|-----------------------|---------------|-----|------|----|
| TunnelToGoal          | 25            | 5,000 | 10,000 | 200|
| ISR                   | 43            | 14,792 | 29,384 | 344|
| CIT                   | 69            | 38,088 | 76,176 | 552|
| CMU                   | 133           | 141,512 | 283,024 | 1,064|
| TunnelToGoal3         | 55            | 1,996,500 | 10,648,000 | 660|
agents with JSQ-learning (JSQ) and JSAQ-learning (JSAQ). In contrast, the initial number of state-action combinations increased linearly with the number of states and agents with CQ-learning and our proposed methods (referred to as SI(sparse interaction) in the table). The number of augmented joint states will be discussed later in this section.

We compared our proposed methods with CQ-learning as well as with joint-state learning (JSQ) and joint-state-action learning (JSAQ). The learning rate $\alpha$ was set to 0.1, the discount rate of the reward $\gamma$ was set to 1.0, and the random action selection rate $\epsilon$ was set to 0.1. Fifty independent trials, each consisting of 10,000 episodes, were run for each method. Because CQ-learning and our methods need pre-learning in a single-agent environment, JSQ and JSAQ with longer episodes, i.e., 200,000 episodes (described in Table 3 as JSQ0.2M and JSAQ0.2M respectively), were also evaluated to be fair.

### 5.1 Number of Average Steps and Augmented Joint States

The results shown in Table 3 include the average number of steps to the goal, the standard deviation during the last 100 episodes, the average number of augmented states, and the standard deviation after 10,000 episodes. The dark gray and light gray cells indicate the methods that resulted in the smallest number of steps to the goal and the smallest number of augmented joint states, respectively.

The agents trained using joint-state learning had difficulty learning the optimal policy for the CMU and TunnelToGoal3 games when the number of episodes was limited to 10,000 due to a large number of state-action combinations. When they had more time to learn better policies, i.e., in JSQ0.2M, they performed better than the CQ- and GCQ-learning agents in most of the games and had the best score in the TunnelToGoal3 game.

If an agent trained using JSAQ-learning, which practically controls all the agents, had sufficient time to explore all the state-action space many times thoroughly, it was able to learn the optimal joint action policy. It had trouble doing this in our evaluation experiment because of the exponentially increasing state-action space in the CMU and TunnelToGoal3 games (Table 2). In the simplest game (the ISR game), which needs only six steps to complete, it demonstrated the best performance when it had sufficient learning time.

The agents trained using CQ-learning did not have the best performance for any game and had the second-best performance only for the TunnelToGoal3 game.

The agents trained using GCQ- and PCQ-learning did not have the best performance for any game; those trained using GCQ-learning had the second-best performance only for the CMU game. They exhibited unstable behavior in the ISR and CIT games, resulting in a deviation in the path taken to the goal. This is because they took greedy actions on the basis of the independent Q-values after they avoided a collision, which resulted in subsequent collisions. The number of augmented states in GCQ-learning was smaller than in CQ- and PCQ-learning for most of the games, indicating that the agents could not sufficiently explore the state-action combinations when a good route was not found due to the greedy action selection.

The agents trained using modified GPCQ-learning, which uses both methods, achieved the best performance in terms of the average number of steps to the goal in the TunnelToGoal, CIT, and CMU games and had the second-best performance in the ISR game. This is attributed to their ability to find the differences between the single-agent environment and the multi-agent environment as well as to estimate the probability of sequential interference.

In the TunnelToGoal3 game, the performances of all the methods were far from being optimal. Even modified GPCQ-learning did not exhibit improved performance because the maze does not conform to the assumption of sparse interaction as the agents frequently interfered with each other near the entrance of the tunnel to the goal.

Figure 3 visualizes the average number of steps and the deviations given in Table 3 for JSQ0.2M, JSAQ0.2M, CQ, and GPCQ. JSQ0.2M and JSAQ were chosen to be fair, and modified GPCQ was chosen as a representative proposed method. Note the negative sides of the standard deviations are omitted in the figure.

### 5.2 Example of Augmented Joint States

Figure 4 shows an example of augmented joint states in the TunnelToGoal game with agents trained using GPCQ- and CQ-learning. The circled numbers represent the locations of the agents where joint states were augmented in the state space of agent 1. As shown in the figure, augmented joint states can be categorized as augmented by collision and augmented by waiting. In an augmented by collision state, if both agents move in the directions indicated by the arrows (i.e., (c-1)(c-2)(c-3)) or an agent moves in a direction indicated by an arrow and the other does not move (i.e., (c-4)(c-5)), a collision occurs, and each agent receives a reward of $-10$. In a single-agent environment, the agent received a reward of $-1$ for the same action taken in the same state. The difference in rewards is detected using Student’s t-test, and the state is augmented. Augmented by waiting has a different mechanism. In a single-agent environment, the agent receives a reward of $+10$ simply for reach-
Fig. 3 Comparison of average number of steps.

Fig. 4 Example of augmented joint states in TunnelToGoal game.
ing the goal. In contrast, in a multi-agent environment, even if an agent selects the same action, it receives a reward of −1 if another agent has not yet reached the goal. In this case, whatever action agent 2 takes at any position (i.e., (w-1)(w-2)), a difference in immediate rewards between the two environments is observed.

As shown in Fig. 4, the number of augmented joint states for both categories is higher with CQ-learning than with GPCQ-learning. This indicates that $\epsilon$-greedy action selection in CQ-learning leads to unnecessary exploration.

5.3 Example of Selected Paths

Figure 5 shows examples of the paths taken when each agent selected greedy actions on the basis of the action policies learned using JSQ-, JSAQ-, CQ-, and our proposed GPCQ-learning in the TunnelToGoal game. The actions of agent 1 are depicted with solid lines, and those of agent 2 are depicted with dotted lines. Thin arrows represent actions taken on the basis of $Q_k$, and thick arrows present actions taken on the basis of $Q_{k}^{aug}$. In all the games, agent 2, which started from $S_2$, took the shortest path to the goal in the examples. In contrast, agent 1, took a detour route, thereby avoiding a collision with agent 2. The agents trained using JSQ- and JSAQ-learning did not take the optimal path. Although the agents trained using the CQ- and GPCQ-learning learned the optimal action policy, those trained using CQ-learning augmented more joint states (i.e., 7, 8, and 9 in the black circles), so more episodes were required for the agents to learn the optimal Q-values for the augmented joint states.

5.4 Computational Cost

None of the proposed methods require extra computational cost other than the cost for selecting the Q-values and the equations for updating Q-values, as shown in Algorithm 1. For example, the average computational time for each step in the TunnelToGoal game was 0.346 ms for CQ-learning and 0.330 ms for GPCQ-learning. GPCQ-learning required less computational time for each step because it avoids unnecessary exploration of a joint state. Moreover, it sometimes causes subsequent collisions, including subsequent interference after avoiding a collision. Detailed analysis on the basis of this assumption remains for future work.

Another issue for future work is the effect of the number of episodes in pre-learning. In this work, 200,000 episodes were used for pre-learning simply because each agent seemed to reach policy convergence in terms of the number of steps to reach the goal. More specific criteria for determining the number of episodes used for pre-learning is necessary.

Finally, evaluation should be done using different types of tasks. A manufacturing task would be a good first target because manufacturing processes are designed so that each process is independent; however, unexpected interference between processes can cause problems.

References

[1] D. Bloembergen, K. Tuyls, D. Hennes, and M. Kaisers: Evolutionary dynamics of multi-agent learning: A survey. *Journal of Artificial Intelligence Research*, Vol. 53, No. 1, pp. 659–697, 2015.

[2] N. Vlassis: A concise introduction to multiagent systems and distributed artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2007.

[3] C. Claus and C. Boutilier: The dynamics of reinforcement learning in cooperative multiagent systems, *Proceedings of the 15th National Conference on Artificial Intelligence*, pp. 746–752, 1998.

[4] M. Lauer and M. Riedmiller: An algorithm for distributed reinforcement learning in cooperative multi-agent systems, *Pro-
ceedings of the 17th International Conference on Machine Learning, pp. 535–542, 2000.

[5] S. Sen, M. Sekaran, and J. Hale: Learning to coordinate without sharing information, Proceedings of the 12th National Conference on Artificial Intelligence, pp. 426–431, 1994.

[6] M. Tan: Multi-agent reinforcement learning: independent vs. cooperative agents, Proceedings of the 10th International Conference on Machine Learning, pp. 330–337, 1993.

[7] F. Melo and M. Veloso: Learning of coordination: Exploiting sparse interactions in multiagent systems, Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 773–780, 2009.

[8] Y. Hauwere, P. Vrancx, and A. Nowé: Learning multi-agent state space representations, Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, pp. 715–722, 2010.

[9] Y. Hauwere: Sparse Interactions in Multi-Agent Reinforcement Learning, Ph.D. Thesis, Vrije Universiteit Brussel, 2011.

[10] T. Kujirai and T. Yokota: Greedy action selection and pessimistic Q-value updates in cooperative Q-learning, Proceedings of the SICE Annual Conference, pp. 821–826, 2018.

[11] C.J.C.H. Watkins: Learning from Delayed Rewards, Ph.D. Thesis, Cambridge University, 1989.

[12] C.J.C.H. Watkins and P. Dayan: Q-learning, Machine Learning, Vol. 8, No. 3-4, pp. 279–292, 1992.

[13] C. Boutilier: Planning, learning and coordination in multiagent decision processes, Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge, pp. 195–210, 1996.

[14] F. Melo and M. Veloso: Decentralized MDPs with sparse interactions, Artificial Intelligence, Vol. 175, No. 11, pp. 1757–1789, 2011.

[15] R. Aras, A. Dutech, and F. Charpillet: Cooperation through communication in decentralized Markov games, Proceedings of the International Conference on Advances in Intelligent Systems - Theory and Applications, 2004.

Toshihiro KUJIRAI (Student Member)

He received the B.S. and M.S. degrees from the University of Tokyo, Japan, in 1995 and 1997, respectively. He joined Hitachi’s Central Research Laboratory in 1997. He is now a senior researcher responsible for Hitachi’s research and development of artificial intelligence applications for industrial systems. He has been a doctoral candidate at Tottori University since 2016. His research interests include big data analysis and reinforcement learning. He is a member of JSAI.

Takayoshi YOKOTA (Member)

He is a professor in the Department of Information and Electronics, Graduate School of Engineering, Tottori University, Tottori, Japan. He received the B.S., M.S., and Ph.D. degrees from Tokyo Institute of Technology in 1979, 1981 and 1984, respectively. He was born in 1956 in Ibaraki prefecture, Japan. He joined Hitachi’s Hitachi Research Laboratory in 1984. He was a chief researcher responsible for Hitachi’s research and development of intelligent transport systems and car navigation systems from 1993 to 2008.