Utilizing Observed Information for No-Communication Multi-Agent Reinforcement Learning toward Cooperation in Dynamic Environment

Fumito UWANO * and Keiki TAKADAMA *

Abstract: This paper proposes a multi-agent reinforcement learning method without communication toward dynamic environments, called profit minimizing reinforcement learning with oblivion of memory (PMRL-OM). PMRL-OM is extended from PMRL and defines a memory range that only utilizes the valuable information from the environment. Since agents do not require information observed before an environmental change, the agents utilize the information acquired after a certain iteration, which is performed by the memory range. In addition, PMRL-OM improves the update function for a goal value as a priority of purpose and updates the goal value based on newer information. To evaluate the effectiveness of PMRL-OM, this study compares PMRL-OM with PMRL in five dynamic maze environments, including state changes for two types of cooperation, position changes for two types of cooperation, and a combined case from these four cases. The experimental results revealed that: (a) PMRL-OM was an effective method for cooperation in all five cases of dynamic environments examined in this study; (b) PMRL-OM was more effective than PMRL was in these dynamic environments; and (c) in a memory range of 100 to 500, PMRL-OM performs well.

Key Words: reinforcement learning, multi-agent system, dynamic environment, memory management.

1. Introduction

Multi-agent systems have become an effective model to simulate certain aspects of human society; they can be used to solve certain problems by agents cooperation. Iwashita et al. aimed to provide support to make a security plan by solving an urban road network security problem, in which guards and criminals are modeled as agents [1]. Multi-agent reinforcement learning (MARL) is a reinforcement learning technique designed to solve problems in a multi-agent system, where the agents generally utilize information from other agents to cooperate with each other. Tan explored the types of information from other agents that contribute to increasing the performance in a multi-agent system [2], though it found two interesting issues. First, it is difficult for agents to handle the information required to cooperate with each other, especially when there are a large number of agents, whereby a large amount of information cannot be effectively processed by the agents. Second, it is difficult to acquire the current information of all other agents without delay or noise as the field of agents becomes large or the number of relay agents increases. From this fact, profit minimizing reinforcement learning (PMRL) [3] was proposed to theoretically force multi-agent cooperation under the condition of no random actions by agents. This method is useful for enabling agents to cooperate with each other without communication. This is important in multi-agent systems whose agents have to learn complex behaviors. For example, when agents exist as transport vehicles, PMRL might perform well on the problem of transporting goods in a disaster situation because PMRL makes all agents spend a minimum amount of time to transport the goods, and PMRL prevents some agents from spending too much time transporting the goods. However, PMRL cannot guarantee multi-agent cooperation if the environment, the first state, and the purpose can be changed in every iteration. Since real-world problems are dynamically changed [1], a no-communication method for agents to learn cooperative behaviors with each other in dynamic environments for MARL may have certain advantages. In problems related to the transportation of goods, situations where roads are closed or cut off, the goal positions are changed, and supply positions are changed as the system dynamics change. To tackle this issue, this study proposes PMRL with oblivion of memory (PMRL-OM), whereby the agents can cooperate with each other while maintaining the guarantee associated with PMRL in a dynamic environment. This study further employs a series of maze problem whose states, starts, and goals change dynamically in the same manner as the goods transportation problem in the disaster situation. The remainder of this paper is organized in the following manner. Section 2 explains a maze problem and the dilemmas as premised in this study. Section 3 explains Q-learning and PMRL, and the proposed method is introduced in terms of architecture and a mechanism in Section 4. Section 5 describes the experiment and analyzes the obtained results, and finally, our conclusions are summarized in Section 6.

2. Problem and Cooperation

To consider a goods transportation problem, a maze problem was employed to validate whether an agent can change the behavior through its learning. The maze has several states, starts, and goals, and the agents are transports on tracks. The agents depart from the start and continuously observe the states until they reach the objective of transporting the goods from the start to the goal, and the agents must reach the goal to acquire the reward. During this cycle, the agents learn to reach the goal to acquire the maximum gain per unit time. In this paper,
the cycle from an agent’s departure to reaching the goal is referred to as the “iteration”. This study focuses on whether the agent can learn multi-agent cooperation in a dynamic environment based on the goods transportation problem in a disaster situation. For this focus, we developed the problem based on the above maze problem, wherein two rules were added to the maze. The first rule is that there are the same number of starts and goals, and the second rule is that the largest step is the smallest among all combinations, and the combination including these types of pairs with the same agent or the same goal is considered a conflict in this study. In short, each pair does not have the same agent or same goal, and the combination including these pairs with the same agent or the same goal is considered a conflict in this study. In short, the conflict is the agents learn to reach the same goal. The arbitrary combination is \( C^* \), the optimal combination is \( C^\ast \), and the combination set is \( C \).

The purpose is to find the optimal combination \( C^\ast \) whose largest step is the smallest among all combinations, and to force the agents to learn to reach the goals by following the optimal combination \( C^\ast \), wherein the agents’ behavior by following the optimal combination \( C^\ast \) is referred to as cooperation in this study, and cooperation is a policy whereby all agents can acquire the rewards for the shortest step.

\[
C^* = \underset{C \in C}{\text{argmin}} \max_{s \in S} t_1(C').
\]  

Figure 1 shows an example of the problem with two agents. Agent A and Agent B represent the start locations of the agents A and B, respectively. Goal X and Goal Y represent the goal locations of goals X and Y, respectively. The combination set is shown in Eq. (2). This equation has two combinations: \( C_1 = [(A, X), (B, Y)] \) and \( C_2 = [(A, Y), (B, X)] \). In each combination, A and B represent the agents A and B, and X and Y represent the goals X and Y, respectively. The pairs of all agents and all goals are shown as \( (A, X), (A, Y), (B, X), \) and \( (B, Y) \), respectively. In Eq. (1), \( t_1(C_1) \), \( t_2(C_2) \), \( t_1(C_1) \), and \( t_2(C_2) \) are the minimum number of steps which the agents A and B spend to reach the goals by following \( C_1 \) and \( C_2 \), respectively.

\[
\begin{align*}
C & = [C_1, C_2], \\
C_1 & = [(A, X), (B, Y)], \quad C_2 = [(A, Y), (B, X)].
\end{align*}
\]  

Since \( t_1(C_1) = 7, t_2(C_2) = 9, t_1(C_2) = 5, \) and \( t_2(C_1) = 7 \) in this problem, the optimal combination \( C^* \) is calculated by Eqs. (4) and (5). Equation (5) derives \( C_1 \) as the optimal combination. Therefore, the agents A and B have to learn to reach the goals X and Y through cooperation.

\[
\begin{align*}
C^* & = \underset{(c_1, c_2)}{\text{argmin}} \max_{(A,B)} \{ t_1(c_1), t_2(c_1) \}. \\
& = \underset{(c_1, c_2)}{\text{argmin}} \{ t_1(c_1) = 7, t_2(c_2) = 9 \} = C_1.
\end{align*}
\]  

In the dynamic environment, \( t_1(C^\ast), C^\ast \), and other variables change after several iterations, and the optimal combination changes by repeating the above process.

### 3. Background

#### 3.1 Q-Learning

Reinforcement learning (RL) [4] is a trial-and-error method which seeks to maximize an acquired reward per unit time. As its general framework, an RL agent interacts with an environment. In other words, it observes a state from the environment, selects an action, receives a reward from the environment as the result of that action, and then learns from the reward. Note that this cycle from the current observation to the next observation is considered a “step” for the purpose of this study. Among many RL methods, Q-learning [5] is a very popular RL method for a single-agent task. A Q-learning agent estimates state-action values (called Q-value) for the possible state-action pairs in the environment (i.e., the agent estimates a discounted expected reward that it will receive when its action \( a \) is executed in its state \( s \)). The agent learns to acquire a policy \( \pi(s,a) \) to decide which action should be executed to maximize the reward they receive. Technically, the policy \( \pi \) can be composed of probabilities in selecting any action \( a \) in any state \( s \) and is calculated by \( Q(s,a) \), \( a \in A \), where \( Q(s,a) \) is the Q-value when the state and action become \( s \) and \( a \), and \( A \) is a set of actions that includes possible actions. To maximize a received reward, \( Q(s,a) \) is updated using Eq. (6).

\[
Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')] - Q(s,a),
\]  

where \( s' \) is the next-state when \( a \) is executed in \( s \), \( a' \) is the next-action executed in \( s' \), \( r \) is the reward received from the environment, and \( \max_{a' \in A} Q(s',a') \) is the largest Q-value when executing the action \( a' \in A \) in the \( s' \). In addition to these variables, \( \alpha \) is the learning rate, while \( \gamma \) is the discount factor. More specifically, \( \alpha \) is a real number from 0 to 1 which indicates the learning speed, while \( \gamma \) is a real number from 0 to 1 which indicates how much the future rewards are considered as important.

#### 3.2 Profit Minimizing Reinforcement Learning

To learn multi-agent cooperation without communication, Uwano et al. proposed profit minimizing reinforcement learning (PMRL) [3], which forces the cooperation by controlling
the reward. The basic idea of PMRL is that the more advantageous agent has to learn to reach the farther goal to help the disadvantageous one. Figure 2 shows an example of PMRL. In PMRL, the situation in the upper part of Fig. 2 occurs initially, and finally changes to that in the lower part. In each maze, there are two goals for two agents. The agents A and B have to reach the goals X and Y through cooperation, respectively. PMRL forces the agents to learn to reach the farthest goal, that is, both agents learn to reach the goal Y in this maze. Since this situation constitutes a conflict, each agent must learn to reach the other goal. In PMRL, the agent farther from the goal Y is forced to learn to reach the goal X as the other goal, because the agent B is closer to the goal Y than the agent A is. Concretely, each agent determines that by checking whether the other agent is already at the same goal or not through iterations. If the other agent is already at the goal, the first agent cannot be closer to the goal; otherwise, the agent is closer to the goal than the other agent is. By following these rules, a cooperative situation occurs. To realize the basic idea, PMRL employed an internal reward and a goal value. The internal reward is the reward inside the agent. That is changed from an ordinary (external) reward, and the agent learns from that in an agent. That is changed from an ordinary (external) reward, and the agent learns from that in an agent. Since the value of the internal reward is the internal reward of the goal where the agent has reached. After the internal reward is set, the agent estimates the Q-value using Eq. (8). This equation is the same as the update function of Q-learning; however, the reward r is changed to irg. In addition, the states s and s′ in Eq. (8) are the state before one step from the goal and the goal, respectively. Since the value irg′y′ indicates the Q-value in the start to reach the goal g, irg′y′rs′ represents the gap between Q-values to reach the goals g and g′. Equation (7) searches the largest gap and sets the largest gap and a constant value of δ on the internal reward irg to learn to reach the goal g. If there is this gap, the agent learns to reach the goal g′. PMRL removes this gap and adds the certain value of δ to learn to reach the goal g using Eq. (7). The value δ must be a positive value; however, other conditions do not exist.

\[
irg = \max_{g' \in G, g} irg_{g' y' s' - 1} + \delta, \quad \text{Eq. (7)}
\]

\[
\phi(s, a) \leftarrow Q(s, a) + \alpha [irg + \gamma \max_{a' \in A} Q(s', a') - Q(s, a)]. \quad \text{Eq. (8)}
\]

3.2.2 Goal value

The goal value is the value by which the farther goal is evaluated based on the minimum number of the steps to reach the goals. Equation (9) is employed to update the goal value. The goal value is updated once every iteration. The agent calculates the weighted average of the minimum number of steps. The weights are indicated by the function φ(m) which includes 0 or 1 for any inputs. The subscript m is the number of iterations from 0 to the current iteration n. Since φ(m) is 1 in only the situation where the agent reaches the goal earliest, bidg is the value where the minimum number of steps is multiplied by the probability to reach the goal g the soonest. This suggests that the agent learns to reach the farthest goal among the goals it might reach. If φ(m) is always 0, the agent does not have to learn to reach the goal because other agents learn to reach it. This is the detail and the policy of the goal value updating. Using goal value updating, PMRL realizes the basic idea.

\[
bidg = \frac{1}{n \sum t \phi(m)} \quad \text{Eq. (9)}
\]

3.2.3 Algorithm

Algorithm 1 is the PMRL algorithm. In this algorithm, tgc is the minimum number of steps until the agent reaches the arbitrary goal gc, and ng is the number of agent reaching the goal g. The agent observes their own state, selects the action, receives a reward (the reward is 0 in PMRL) and updates the Q-value from the reward (lines 3 to 6), which follows the same flow as that of Q-learning. The agent repeats this cycle several times and stops if it has reached the goal or the several numbers of steps are used (lines 2 and 8). Thereafter, if the agent is at the goal g and the number of the current steps is smaller than before, it saves this step to tgc (lines 9 and 10). In the end, the agent updates the goal value bidg (lines 12 to 20), and the internal reward (lines 21 and 22). Concretely, if the agent has reached the selected goal g, it adds 1 to ng. After that the agent sets the function φ(n) to 1 if it had reached the goal g fastest in the current iteration; otherwise, it sets it to 0. From those parameters, the agent updates the goal value bidg. After updating, the agent reselects the appropriate goal g′ based on the updated goal values and sets the internal reward to reach the goal g′. The agent selects the appropriate goal g′ based on the goal value bidg. Concretely, the agent selects the goal with
the largest goal value in some probability and selects that randomly in other probability (15% in this paper). This is because the agent can estimate the accurate goal values for all goals and prevents from repeating selecting the first selected goal like the greedy selection. Note that the agent selects the goal randomly in other probability (15% in this paper). This is because the agent can estimate the accurate goal values for all goals and prevents from repeatedly selecting the first selected goal like the greedy selection. After that, the agent goes to the next iteration and repeats this process. At the next iteration, if the agent has reached the selected goal, the agent stores the memory, whereas PMRL-OM does not. Since PMRL-OM stores the goal that the agent reached and the steps it took to reach the goal, the agent can estimate the accurate goal values for all goals and prevents from repeatedly selecting the first selected goal like the greedy selection.

### 4. PMRL with Oblivion of Memory

To achieve cooperation in a dynamic environment, we propose PMRL with oblivion of memory (PMRL-OM) as extended PMRL. The agent’s architecture of PMRL-OM is the same as that of PMRL. However, PMRL-OM is different from PMRL in two notable ways, which includes how the minimum number of the steps and how to calculate the goal value. Both of these aspects and how PMRL is improved are explained below.

#### 4.1 Difference of the Minimum Number of the Steps

Figure 3 shows the difference between the minimum number of steps for PMRL and PMRL-OM. In this figure, the top shows the PMRL method and the bottom is PMRL-OM. The bottom arrow indicates the progress of the iterations. This figure depicts the difference from iteration 50 to iteration 51. The matrix inside the agent indicates the memory for the minimum number of the steps. The matrix of PMRL stores the minimum number of the steps. The steps in the memory are the minimum number of steps based on the PMRL-OM memory. PMRL has the minimum number of steps inside the memory, whereas PMRL-OM does not. Since PMRL-OM stores the goal that the agent reached and the steps it took to reach the goal, the agent can estimate the accurate goal values for all goals and prevents from repeatedly selecting the first selected goal like the greedy selection.

#### 4.2 Goal Value Calculation Mechanism

In the process of calculating the goal values, the agents utilize Eq. (10) to make convergence to the goal value easy than PMRL. Concretely, $(\frac{1}{\lambda})^p \gamma^m$ exists to prevent the divergence of the goal value and converges that to the minimum number
of the steps. Since Eq. (10) is the geometric progression and the geometric ratio is \( \frac{\xi}{\xi - 1} \) as the positive real value under 1, this equation converges even if \( n_{g} \) becomes infinite. The converged value can be calculated by Eq. (11). Note that this converged value might be little smaller than \( \beta t_{g} \). Until \( t_{g} \) has converged to the minimum number of the steps or when \( \phi(m) = 0 \),

\[
\begin{align*}
\text{bid}_{g} &= \frac{\beta}{\xi} \sum_{m=0}^{n_{g}} (\frac{\xi - 1}{\xi})^{n_{g}-m} t_{g}\phi(m), \\
\lim_{n_{g} \to \infty} \text{bid}_{g} &= \frac{\beta}{\xi - 1} - \frac{t_{g}}{\xi - 1} = \beta t_{g}.
\end{align*}
\]

In addition, it is not easily influenced by the minimum number of the steps acquired in one iteration. For example, if the current iteration is 1000 (i.e., \( n_{g} = 1000 \)), then \( t_{g} = \frac{1000}{1000} \) is added to the goal value of goal \( g \). This suggests that the goal value cannot be changed in the latter half of the iterations. Since PMRL-OM has to find appropriate goal values after an environment change, the goal value must be calculated by emphasizing the new information. In Eq. (10), \( \text{bid}_{g} \), \( t_{g} \), \( \phi(m) \), and \( n_{g} \) are the same as those of PMRL. The variable \( \xi \) is a constant value, and \( \beta \) is a discount rate for the goal value. Equation (10) becomes easily influenced by the newly added \( t_{g}\phi(m) \) by changing the denominator from \( n_{g} \) to the constant positive integer value \( \xi \). In addition, we add \( \beta \) to \( t_{g}\phi(m) \) to promote this outcome. Note that \( \beta \) is a real value from 0 to 1 as a discount rate. The discount rate \( \beta \) has a role to change the converged goal value. How to use this \( \beta \) is that the agents set \( \beta \) to the small positive value in order to prevent from the influence of the inaccurate minimum number of the steps at the beginning iteration. For example, if the current iteration is 1000 (i.e., \( n_{g} = 1000 \)), \( \xi = 500 \), and \( \beta = 0.9 \), then \( \frac{0.9(\phi(m))}{500} \) is added to the goal value \( \text{bid}_{g} \). From this result, PMRL-OM can add the latest \( \frac{0.9\phi(m)}{500} \) to the goal value regardless of the iteration, and obtain a small beginning goal value by \( \frac{\xi - 1}{\xi} \) and \( \beta \). In this situation, \( \beta \) changes the upper range of \( t_{g} \) smaller and promotes the convergence of the goal value. From the above explanation, PMRL-OM can resolve problems in which the minimum number of the steps cannot influence the goal value in the later iteration.

Figure 4 shows the difference between the goal value updating of PMRL and PMRL-OM. The parameters are the same as those of the experiment. The function \( \phi(m) \) is always 1, \( t_{g} = 3 \) before the 500th iteration and \( t_{g} = 1 \) after the 500th iteration. The vertical and horizontal axes represent the goal value and iteration, respectively. The marked line and the solid line indicate the goal value updated by PMRL and PMRL-OM from the first to the 1000th iteration, respectively. From this figure, the goal value of PMRL changes rapidly, while that of PMRL-OM changes gradually before the 500th iteration. In addition, the goal value of PMRL-OM converges earlier than that of PMRL after 500 iterations. This suggests that PMRL-OM can update the goal value to prevent it from being influenced by the wrong minimum number of the steps in the earlier iteration. Note that since the agent might not reach the goal by the minimum number of steps in the early iteration, it calculates the goal value using the wrong steps. PMRL-OM can update the goal value from the minimum number of the steps in the earlier iteration. In addition, PMRL-OM can consider the minimum number of the steps changing earlier than PMRL; that is, PMRL-OM can handle a changing environment earlier than PMRL can.

4.3 Algorithm

Algorithm 2 is the algorithm of PMRL-OM. In this algorithm, \( t_{g} \) is the array, including the minimum numbers of steps until the agent reaches the goal \( g \) in each iteration, and \( n_{g} \) is the number of the agent reaching the goal \( g \).

The agent observes its own state, selects the action, receives a reward (the reward is 0 in PMRL-OM), and updates the Q-value from the reward (lines 3 to 6). This is the same flow as that of Q-learning. The agent repeats this cycle for several times, and stops this cycle if it has reached the goal or several steps are spent (lines 2 and 8). Thereafter, if the agent is on the goal \( g \) and the number of the current step is smaller than before, it saves this step into \( t_{g} \) (lines 9 and 10); otherwise, it saves the maximum number of steps into \( t_{g} \) (lines 11 to 13). After this process, if the length of \( t_{g} \) is over the threshold \( \text{Cycle} \), the agent erases the first element in \( t_{g} \) (lines 14 to 16). In the end, the agent updates the goal value \( \text{bid}_{g} \) (lines 17 to 22) and the internal reward (lines 23 and 24). Stated simply, the agent adds 1 to \( n_{g} \), and the function \( \phi(g) \) to 1 if it had reached the goal \( g \) fastest at the current iteration; otherwise, it sets 0 on that. From those parameters, the agent updates the goal value \( \text{bid}_{g} \). After this updating, the agent reselects the appropriate goal \( g' \) based on the updated goal values, and sets the internal reward to reach the goal \( g' \). Since this
process indicates the flow of PMRL-OM in one iteration, the agent goes to the next iteration and repeats the process.

5. Experiment

5.1 Experimental Design and Setting

To investigate the effectiveness of PMRL-OM, we verified that PMRL-OM is applicable to dynamic changes in $3 \times 8$ grid mazes. There were four cases as follows, Cases (1-1), (1-2), (2-1), and (2-2). Some walls were placed in the maze after a certain number of iterations in Cases (1-1) and (1-2). On the other hand, the agents’ and the goals’ location were changed after a certain number of iterations in Cases (2-1) and (2-2). Each case required different cooperation with each other for changes of the walls and the positions, respectively: the agents do not have to change selecting the goals in Cases (1-2) and (2-2), while they have to change selecting the goals in Cases (1-1) and (2-1), and three examples were employed for each case. Figure 5 shows the maze in each case as examples of the four environmental changes. The arrow in the bottom of the figure indicates the sequence of the iterations, the dotted line indicates the period of the environmental change, the maze before the environmental change is on the left side, and the mazes after the change is on the right side. The mazes on the right side are in order of Cases (1-1), (1-2), (2-1), (2-2) from the top. Note that the other two mazes change their own environment in the same manner for each case.

This study evaluated the steps spent until all agents reached their goals. The total number of experiments was determined by the number of trials (e.g., 300 trials with 30 different seeds in 10 types of mazes for each case). Table 1 summarizes the parameters. Learning iterations and steps were limited to 50000 and 100 as the threshold, respectively. Q-values of all states were initialized to 0, and $\alpha$ and $\gamma$ were set to 0.1 and 0.9, respectively. All methods employed $\epsilon$-greedy selection in the learning phase, while they employed the greedy selection in the evaluation phase. More specifically, the agents selected their actions according to the $\epsilon$-greedy selection method with $\epsilon = 0.7$, and evaluate the learning result according to the greedy selection. We set $\epsilon = 0.7$ to find the minimum number of the steps from the start to all goals. The ordinary (external) reward was set to 10, and $\delta$ was set to 10. The memory range $\text{Cycle}$ was set to 50000, the constant $\xi$ was set to 100, and the discount rate $\beta$ was set to 0.9 in PMRL-OM.

5.2 Experimental Result

Figure 6 shows the results of all cases. In each graph, the vertical axis indicates the step spent by the agents, while the horizontal axis indicates the number of iterations. A line with circle marks, with square marks, and a dotted line indicate the results of PMRL-OM, PMRL, and the minimum number of steps, respectively. Note that each line indicates the average step among steps with 30 seeds. From these results, PMRL-OM performed well after environmental changes. In particular, the steps in all cases converged to the minimum number of steps in PMRL-OM. Note that the results of the three types of mazes were the same in each case. In addition, we used the Wilcoxon signed-rank sum test to verify two types of data. These two verifications were designated as tests 1 and 2 in this response. In test 1, the data included a minimum of 30 steps for 30 random seeds in the last iteration for each maze. Each experiment was executed 30 times with different random seeds to prevent it from being influenced by randomization. These data did not behave according to a certain distribution (i.e., non-parametric test is the best), and these data exist as pairs. This suggests that the Wilcoxon signed-rank sum test was appropriate for test 1. We verified that PMRL-OM performed better than PMRL did with all random seeds in each maze by test 1. Table 2 includes the $p$-values for test 1 results in all mazes for all cases. The rows represent the cases in order from Cases (1-1), (1-2), (2-1), and (2-2). The columns represent Maze1, Maze2, and Maze3, and note that Maze 1 is the maze in Fig. 5. The results were good because the $p$-value was very small without those of Maze 1 and Maze 2 in Case (1-2). These results were “NaN” because the data used for PMRL-OM and PMRL were identical. This suggests that PMRL-OM and PMRL demonstrated the same level of effectiveness for these two mazes. From these results,
PMRL-OM performed as well as or better than PMRL did in each maze. In test 2, the data included a minimum of 12 steps for the average minimum numbers of steps among 30 random seeds for three mazes in four cases. These data did not behave according to a certain distribution (i.e., non-parametric test is the best), and these data exist as pairs. This suggested that the Wilcoxon signed-rank sum test was appropriate for test 2. We verified that PMRL-OM performed as well as or better than PMRL did in all mazes in test 2. The $p$-value was 5.83e-3 for the comparison between PMRL and PMRL-OM in all mazes. Since this $p$-value is very small, PMRL-OM performs as well as or better than PMRL did in all mazes.

5.3 Discussion

From the result of Fig. 6, PMRL-OM performs well in the dynamic environment. In addition, the agents were found to learn cooperative behavior and reached all goals with the minimum number of steps. In this figure, PMRL-OM performed better than PMRL did before the environment changed. This difference was due to the difference in the goal value update equations of PMRL-OM vs. PMRL. PMRL-OM is not easily influenced by the minimum number of steps acquired from the earlier iteration. Since the agents might not store the appropriate minimum number of the steps at the earlier iteration, PMRL might calculate the wrong goal value, whereas PMRL-OM prevented the calculation of the wrong goal value. This suggested that PMRL-OM can repair the goal value calculated by the wrong minimum number of steps earlier than PMRL can. Therefore, PMRL-OM performed better than PMRL did before the environment changed based on the experimental result. Figure 7 shows the goal values of all agents with PMRL-OM in all cases. The upper part of this figure shows the agent A’s goal values, while the lower part shows that the agent B’s goal values. In this figure, the vertical axis represents the goal values, and the horizontal axis represents the iterations. The solid line represents the goal value of the goal X, while the dotted line represents that of the goal Y. The agent set the internal reward to reach the goal with the maximum number of goal values. This figure indicates that the agents can change the large and small relationship of the goal values, and they can apply it in a dynamic environmental, or in other words, the agents have to learn to reach different goals after the environmental change in Cases (1-1) and (2-1). On the other hand, the agents should keep the relationship in Cases (1-2) and (2-2). From these results, the agents can set the goal values and learn to apply them to the environmental change using PMRL-OM.

5.3.1 Comparison between PMRL-OM and PMRL

Both methods can enable the agents to learn cooperative behaviors with the minimum number of steps before the environmental change in all cases, and PMRL-OM can perform optimally after that change. However, PMRL cannot apply to the environment after the change. In particular, the steps converged to other values in Cases (1-1) and (2-1), the steps converged to the minimum number of steps in Case (1-2), and the number of steps was 100 in Case (2-2). Note that when the step was 100, this situation indicated that the agents failed to cooperate with each other. PMRL can only apply to Case (1-2). In other words, because the agents do not have to change the large and small relationship, and they should learn to reach the same goals of those before the environmental change. However, PMRL might not perform well in Case (1-2). The agents stored the minimum number of steps, and these were the same as those
stored before the change. This suggested that the agents might not calculate the internal reward well using PMRL. In this experiment, PMRL performed well because $\delta$ was large enough to decrease the influence of this problem. On the other hand, PMRL-OM can store the minimum number of steps after the change, and this always enabled the agents to learn cooperative behavior with the minimum number of the steps. In other cases, the agents using PMRL cannot learn to reach the appropriate goals for cooperation, unlike PMRL-OM. From the above, PMRL-OM was found to be better than PMRL was in a dynamic environment.

5.3.2 How to utilize the information

Figure 8 shows the results when $\xi$ becomes other values in Case (1-1). The vertical axis represents the step spent by the agents, while the horizontal axis represents the number of iterations. The line represents the result of PMRL-OM, and the stable line under that represents the minimum number of steps. In this figure, when $\xi$ is large, it is difficult for the steps to converge to a certain value, whereas if $\xi$ is over 5000, the step converges to a certain value without the minimum number of steps, i.e., PMRL-OM cannot apply to the environmental change. From these results, $\xi$ has to be a value under 1000. On the other hand, Fig. 9 depicts the goal values when $\xi$ becomes other values in Case (1-1). The figures in the upper part are the behavior of the agent A’s goal value, while those in the lower part are the behavior of the agent B’s goal value. There are three kinds of the graphs of $\xi = 10, 500, \text{and } 10000$ in order from left to right. The vertical axis represents the goal values, and the horizontal axis represents the iterations. The solid line represents the goal value of the goal X, while the dotted line represents that of the goal Y. If $\xi = 10$, the agents can set appropriate goal values, but the goal values of the agent A did not converge because $\xi$ is small, and the goal value becomes easily influenced by the latest update. If the acquired information is imbalanced, the agents cannot completely estimate the optimal goal values. If $\xi = 500$, the goal values can converge to a certain value and apply it to the environmental changes. If $\xi = 10$, the goal value cannot converge. This is the influence of agent B. Since the minimum numbers of the steps do not change, the relationship of the goal values is not
changed. From these results, $\xi = 500$ was the best. Note that the results of the other values of $\xi$ cannot be better than $\xi = 500$; however, they are not all bad.

The variable $\xi$ is the parameter used to calculate the goal value, which cannot be influenced by steps before the environmental change. The problem of the goal value calculation using PMRL is the agents added steps divided by the large value (update count) to the goal value as the learning proceeded. The goal value calculation of PMRL-OM was that the agents always added steps to the goal value at each learning iteration. This is an important aspect of PMRL-OM, and this is established if $\xi$ is any positive value. For this reason, $\xi$ is not sensitive, though if $\xi$ is the huge value, the goal value calculations become almost the same between PMRL-OM and PMRL, as shown in Fig. 8, and $\xi$ should be under 500. In addition, Fig. 9 shows the changing goal value at each iteration in $\xi = 10$, $\xi = 500$, and $\xi = 10000$. From the result of $\xi = 10000$, since the goal value of one goal is always larger than that of the other through all iterations, $\xi$ should be under 500. Although PMRL-OM performs bad with $\xi = 1000$, $\xi = 5000$, and $\xi = 10000$, that PMRL-OM can acquire the optimal result in almost all trials, that is, the PMRL-OM performs badly for certain seeds. The difference in the performance among the different values of $\xi$ occurs along with the number of the learning iterations. This suggests that if the number of iterations is large, $\xi$ can be many values; otherwise, it should be a small value. However, the performance does not become the worst even if $\xi$ is the large value in the situation of small iterations. Note that the agents cannot completely correspond to the environment before the change, but they can correspond to the environment after the changing in Fig. 9.

5.4 Additional Experiment for a More Complicated Case

To validate the effectiveness of PMRL-OM in an environment consisting of all cases, we employed one case, as shown in Fig. 10. Figure 10 shows the situation of Case (3). The arrow in the bottom of the figure represents the sequence of iterations, the dotted line represents the period of environmental change, the maze before the environmental change is on the left side, and the mazes after that are on the right side. The experimental design and setting were performed in the same manner as the other experiment. Figure 11 shows the result, and Figs. 12 and 13 show the goal values of each agent. The axes and the lines of these figures are the same as those in Figs. 6 and 7. From the results, PMRL-OM can perform well in the combined environment. Specifically, the agents can learn to reach the goals with the minimum number of steps, and they can change the large and small relationship of the goal values after the environmental change. This suggests that PMRL-OM enable the agents to learn cooperative behaviors along with several environmental changes. In addition, we executed two tests from the
results of five cases. In test 1, the $p$-values were 4.62e-8 for all mazes in Case (3). In test 2, the $p$-value was 1.62e-3 for the comparison between PMRL and PMRL-OM for all mazes in five cases. Since this $p$-value was very small, PMRL-OM performed better than PMRL did in all mazes.

6. Conclusion

This paper proposes a multi-agent RL method without communication with respect to dynamic environments, called PMRL with oblivion of memory (PMRL-OM). PMRL-OM is an extension of PMRL which changes the ordinary (external) reward to the internal reward for the cooperation, and it defined a memory range to only utilize the valuable information from the environment. Since agents do not require information observed before an environmental change, the agents utilized the information acquired after a certain iteration by the memory range. In addition, PMRL-OM improved an update function for a goal value as a priority of purpose, and updated the goal value based on newer information. To evaluate the effectiveness of PMRL-OM, this paper compares PMRL-OM with PMRL in five cases of dynamic maze environments: starts, goals, and normal states are replaced and a comprehensive case of the replace. From the experimental result, we found that:
(a) PMRL-OM was an effective method for cooperation in all five cases of dynamic environments; (b) PMRL-OM was better than PMRL in the dynamic environment; and (c) if the memory range was between 100 and 500, PMRL-OM performed well. This study employed the factor of dynamism. Since real-world problems are more complex, we are going to improve the proposed method to apply to situations where environmental change happens randomly in terms of timing in the future.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP17J08724, and the authors would like to thank Enago (www.enago.jp) for the English language review.

References

[1] H. Iwashita, K. Ohori, H. Anai, and A. Iwasaki: Simplifying urban network security games with cut-based graph contraction, Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems, AAMAS ’16, pp. 205–213, Richland, SC, 2016.
[2] M. Tan: Multi-agent reinforcement learning: Independent vs. cooperative agents, Proceedings of the 10th International Conference on Machine Learning, pp. 330–337, 1993.
[3] F. Uwano and K. Takadama: Comparison between reinforcement learning methods with different goal selections in multi-agent cooperation, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 21, No. 5, pp. 917–929, Sep. 2017.
[4] R.S. Sutton and A.G. Barto: Introduction to Reinforcement Learning, 1st edition, MIT Press, Cambridge, MA, USA, 1998.
[5] C.J.C.H. Watkins: Learning from Delayed Rewards, Ph.D. thesis, King’s College, 1989.

Fumito Uwano (Member)
He received a Master’s degree from the University of Electro-Communications, Japan, in 2017. He is currently a doctoral student at the University of Electro-Communications and a research fellow at the Japan Society for the Promotion of Science. His research interests include multi-agent systems and reinforcement learning.

Keiki Takadama (Member)
He received a Doctor of Engineering degree from the University of Tokyo, Japan, in 1998. He worked as a visiting researcher at the Advanced Telecommunications Research Institute International (ATR) between 1998 and 2002 and then worked at the Tokyo Institute of Technology as a lecturer between 2002 and 2006. He became an associate professor at the University of Electro-Communications in 2006 and has been a professor there since 2011. His research interests include multi-agent systems, reinforcement learning, and evolutionary computing.