LSTM-based Abstraction of Hetero Observation and Transition in Non-Communicative Multi-Agent Reinforcement Learning

Fumito Uwano[a]
Department of Computer Science, Okayama University, 3-1-1, Tsushima-naka, Kita-ku, Okayama, Japan

Keywords: Multiagent System, Reinforcement Learning, LSTM, Hetero-information, Hetero-transition.

Abstract: This study focuses on noncommunicative multiagent learning with hetero-information where agents observe each other in different resolutions of information. A new method is proposed for adapting the time dimension of the hetero-information from the observation by expanding the Asynchronous Advantage Actor–Critic (A3C) algorithm. The profit minimizing reinforcement learning with oblivion of memory mechanism was the previously used noncommunicative and cooperative learning method in multiagent reinforcement learning. We then insert an long short-term memory (LSTM) module into the A3C neural network to adapt to the time dimension influence of the hetero-information. The experiments investigate the performance of the proposed method on the hetero-information environment in terms of the effectiveness of LSTM. The experimental results show that: (1) the proposed method performs better than A3C. Without the LSTM module, the proposed method enabled the agents' learning to converge. (2) LSTM can adapt the time dimension of the input information.

1 INTRODUCTION

Multiagent Reinforcement Learning (MARL) controls some agents in groups to learn cooperative action, such as in warehouses where robot agents cooperate with each other to manage the delivery of supplies. In this case, MARL must decrease the complexity of communication to achieve the desired cooperation and enable the robots to solve real-world problems. In previous work, Kim et al. discussed a practical scenario for each agent to communicate with other agents in real-world reinforcement learning tasks and proposed a multiagent deep reinforcement learning (DRL) framework called SchedNet (Kim et al., 2019). Du et al. expanded the focus to the dynamic nature of communication and the correlation between agents' connections to propose a learning method to obtain the topology (Du et al., 2021). Those works are efficient and straightforward, but the agents themselves cannot do complex tasks based on real-world problems, especially in a dynamic environment. In contrast, Raileanu et al. proposed self–other modeling (SOM) method to enable agents to learn cooperative policy through predicting others' purpose or goals based only on the observation (Raileanu et al., 2018). Ghosh et al. argued that the premise of SOM requires the behaviors and types of all agents be presented as a problem and proposed AdaptPool and AdaptDQN as cooperative learning methods without using this premise (Ghosh et al., 2020). However, these methods are based on static environment and require all purposes to be given. Uwano et al. proposed a method called profit minimizing reinforcement learning with oblivion of memory (PMRL-OM) as a noncommunicative and cooperative learning method in a multiagent dynamic environment (Uwano and Takadama, 2019).

In addition, by communicating with other agents and predicting their behaviors, the agents can learn appropriate and cooperative actions (Raileanu et al., 2018; Ghosh et al., 2020). In MARL, however, agents can observe the same resolution information, e.g., the environmental situation and other agents' actions (i.e., if some agents can observe high-resolution information, then the other agents can also observe the same information). This can become an issue because all agents cannot be guaranteed to observe the same resolution of information about each other in the real world. For example, all vehicles using car navigation systems must cooperate (in planning routes) appropriately with each other because the optimal routes are influenced by other vehicles' route planning and traffic jams; therefore, the vehicles cannot observe

[a] https://orcid.org/0000-0003-4139-2605

Uwano, F.
LSTM-based Abstraction of Hetero Observation and Transition in Non-Communicative Multi-Agent Reinforcement Learning.
DOI: 10.5220/0010795700003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 1*, pages 172-179
ISBN: 978-989-758-547-0; ISSN: 2184-433X

information with different resolutions. Uwano described the different information resolutions as "hetero information" (Uwano, 2021), and discussed the relationship between the network topology of DRL and abstraction of input–output information. Although they discuss the hetero-information in inputs, the transition is important in MARL. Recently, the results for hetero-information did not converge; that is, the agents could not stably learn cooperative actions (Uwano, 2021).

Based on this background, this paper focuses on the hetero-transition caused by the hetero-information. Hetero-transition refers to the different transitions between observations and the actual situation. Thus the problem expands from the hetero-information to the hetero-transition. In addition, this paper proposes a noncommunicative and cooperative learning method using the unstable hetero-transition in a dynamic environment. In particular, this paper proposes a new method to expand the Asynchronous Advantage Actor–Critic (A3C) algorithm to adapt the hetero-information from observations using the PMRL-OM mechanism as the previous noncommunicative and cooperative learning method in MARL. We then an LSTM module into the A3C neural network to adapt to the time dimension influence on the hetero-information. In the experiment, this paper employed a maze problem where the starting and goal locations change after several steps in the grid world and either of the agents sense only the hetero-information and investigated the agents' performance in solving the problem.

This paper is organized as follows: A3C and PMRL-OM are introduced in Sections 2 and 3, respectively. The proposed new method combined both A3C and PMRL-OM as explained in Section 4. Furthermore, the problem and unstableness are explained in Section 5. The experimental details and discussions are described in Section 6. Finally, the conclusions are presented in Section 7.

2 ASYNCHRONOUS ADVANTAGE ACTOR–CRITIC

The A3C algorithm (Mnih et al., 2016) is a DRL method where the system copies the agents and environments, and then executes trials asynchronously to acquire an optimal policy immediately. This paper explains the details of A3C based on (Fujita et al., 2019), where the implementation is similarly employed in the experimental section of this paper as described previously (Mnih et al., 2016). The copied agents execute backpropagation of the primary agent with the loss of parameters as the learning result. They then initialize themselves and synchronize the current parameters from that of the primary agent and repeat the presented processes.

In DRL, state s or sensed information to detect the state is input to the network and the policy π or state–action value $Q(s,a)$ is output from the network. DRL approximates the true policy π or true state–action value $Q(s,a)$ and action a throughout the process to learn by backpropagation from the loss between the output and true value. As for A3C, the agent estimates the appropriate values of the policy $\pi(a_t|s_t;\theta)$ and state value $V(s_t;\theta_v)$ using the neural network, as well as shares them to the copied agents, which calculate the loss of the parameters θ and θ_v to update ones of the neural network. A3C learns from the state s or some information to detect s as input to output the policy $\pi(a_t|s_t;\theta)$ and state value $V(s_t;\theta_v)$ as respectively "Actor" and "Critic."

The policy loss $d\theta$ and state value loss $d\theta_v$ are defined as Equations (1) and (4), respectively.

$$d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi\left(a_i|s_i;\theta'\right) A(s_i,a_i;\theta,\theta_v) + \beta\nabla_{\theta'} H(\pi\left(s_i;\theta'\right)), \tag{1}$$

$$A(s_i,a_i;\theta,\theta_v) = \sum_{j=0}^{k-1} \gamma^j r_{i+j} + \gamma^k V(s_{i+k};\theta_v) - V(s_i;\theta_v). \tag{2}$$

$$R \leftarrow r_i + \gamma R, \tag{3}$$

$$d\theta_v \leftarrow d\theta_v + \frac{\partial\left(R - V(s_i;\theta'_v)\right)^2}{\partial\theta'_v}. \tag{4}$$

Where the losses θ' and θ'_v are for the copied agent. In a certain steps i, the state, action and reward are denoted by s_i, a_i, and r_i, respectively. Updating θ' uses the entropy function $H(\pi(s_i;\theta'))$ multiplied by the factor β. In the advantage function $A(s_i,a_i;\theta,\theta_v)$, the calculation includes the future reward multiplied by the discount factor γ. The copied agents update the original parameter using Equations (1) and (4).

3 PROFIT-MINIMIZING REINFORCEMENT LEARNING WITH OBLIVION OF MEMORY

To learn multiagent cooperation without communication in an environment of dynamic change, Uwano et al. proposed the PMRL-OM mechanism (Uwano and Takadama, 2019), which is based on Q-learning and enables cooperation by managing the reward in dynamic change environments. The agents can identify the largest spent steps when they reach all goals

through the optimal steps, e.g., the PMRL-OM mechanism enables all agents to reach the farthest goals in the maze problem. While that might result in each agent crashing into each other, the agent that reached the goal first yield it to other agents by trying to reach other goals and avoiding crashing and managing the goals whenever accidents occur.

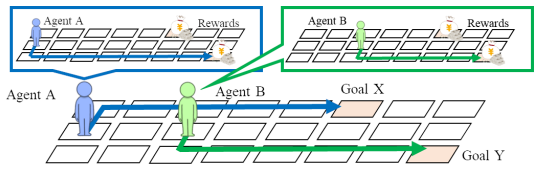

Figure 1: Overview of PMRL-OM.

The PMRL-OM mechanism updates the agent's memory as the environment changes and controls the reward values for each agent to reach all goals in the shortest time (Figure 1). In this figure, Agents A and B are learning to reach the appropriate goals with each other by tuning their own reward values. Thus, Agents A and B reached goals X and Y, respectively. Using an internal reward design as the reward tuning mechanism and updating the goal values as the goal selection are explained in the next subsection.

3.1 Internal Reward Design

Generally, an agent learns based on environmental rewards or a reward signal communicated by the agent when it achieves its purpose. However, the internal reward design can remake the reward, or the reward signal is used as the internal reward in the agents' learning process. PMRL-OM estimates the internal reward to enable agents to reach the appropriate goals by learning. The next subsection, "goal value updating," explains how to decide the appropriate goals. Let the goal g be the appropriate goal, then the internal reward is calculated using Equation (5) as follows:

$$ir_g = \max_{g' \in G, g' \neq g} r_{g'} \gamma^{t_{g'} - t_g} + \delta, \tag{5}$$

where ir_g is the internal reward of the goal g, γ is the discount rate in Q-learning, g' is the certain goal in the goal set G in which all goals are included, and $r_{g'}$ is the external reward of the goal g'. The variables t_g and $t_{g'}$ are the minimum number of steps until the agent has reached the goals g and g', respectively. Note that the internal reward is calculated for only the appropriate goal g and those of the other goals are set by the external rewards.

Equation (5) calculates discounted expected rewards as Q-values to each goal in the Q-learning process and sets the internal reward of the appropriate goal to be larger than any other goal. For example, Figure 2 shows three goals denoted by the money pouches, the arrows show the actions to achieve the goals, and the variables indicate the Q-values. An internal reward ir_g is set by the agent to facilitate other agents to achieve the goal g. Thus, ir_g should be set as follows:

$$ir_g \gamma^{t_g} > \max\left\{r_2 \gamma^{t_2}, r_3 \gamma^{t_3}\right\} \tag{6}$$

Let the difference between the left- and right-hand sides be δ; thus, the equation can be transformed as follows:

$$ir_g \gamma^{t_g} = \max\left\{r_2 \gamma^{t_2}, r_3 \gamma^{t_3}\right\} + \delta \tag{7}$$

$$ir_g = \max\left\{r_2 \gamma^{t_2 - t_g}, r_3 \gamma^{t_3 - t_g}\right\} + \delta \tag{8}$$

Let the arbitrary goal be g' and Equation (8) is the same as Equation (5). Therefore, the internal reward design can lead the agent to the appropriate goal.

3.2 Goal Value Updating

The goal value can be used by agents to select the appropriate goal. The value is converged to the minimum number of steps for each goal to achieve the goal that takes the longest time to achieve (i.e., the farthest goal in the maze problem). The agent then learns how to reach the goal with the maximum value using this internal reward. Equation (9) denotes the goal value update functions. Let t_g and ξ be the minimum number of steps for the appropriate goal g and constant value. Note that ξ is a positive integer value greater than 0 and ξ indicates how much emphasis is given to the current minimum number of steps in the goal value: if ξ is small, the minimum numbers of steps are emphasized; otherwise an increased minimum numbers of steps is emphasized in the goal value. At the end of every iteration, if the agent has received the reward, the goal value is updated by the top function; otherwise, it is updated by the bottom function. Using these equations, all agents aim to reach the farthest goal. After that the agents set their internal rewards to reach that goal:

$$\begin{cases} bid_g = \frac{\xi-1}{\xi} bid_g + \frac{t_g}{\xi} & if\ received\ rewards \\ bid_g = \frac{\xi-1}{\xi} bid_g + \frac{0}{\xi} & otherwise \end{cases} \tag{9}$$

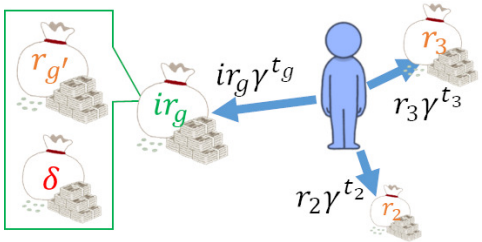

Figure 2: Internal reward design.

3.3 Memory Management

To use the minimum number of steps, PMRL-OM counts the steps to each goal throughout the entire iteration. However, PMRL-OM stores the steps in order because its memory is finite and older data might degrade. Figure 3 shows the memory management using PMRL-OM. The word balloon and the human figure at its tail denote the memory and agent, respectively. The bottom arrow indicates the continuous iterations using the iteration e to $e+1$. Given the memory length e, the agent repeats the actions of storing the current iteration, counting the current number of steps, and when it reached the goal until the stack length of the data is less than the memory length e. If the stack length is over e, the agent removes the oldest data from its memory.

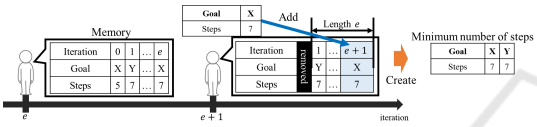

Figure 3: Memory management of PMRL-OM.

4 PROPOSED METHOD

The proposed method is expanded from A3C in the hetero-observation and hetero-transition using the PMRL-OM mechanism. The goal value and internal reward design are added to the A3C algorithm, which is modified to adapt to the hetero-observations. The A3C neural network is replaced with the LSTM-based network to adapt to the hetero-transitions. This section explains the presented three modifications and describes the shared parameters in the proposed method: that is, the goal values and minimum numbers of steps, which can promote learning and prevent any influence of the hetero-information by helping the agents to ervise their own information from the shared data.

4.1 Modification of Goal Value Design

The A3C is a policy-based algorithm, where the agent performs actions based on the policy π. Thus, the goal value function must be expanded in the case when the memory (explained in Subsection 3.3) is empty. In this situation, PMRL-OM sets the maximum number of steps to t_g in Equation (9) instead of the minimum number. Thus, the agent is influenced to reach the unmemorized goal. However, the maximum number of steps is too large for the A3C algorithm to enable the agent to avoid becoming absorbed in the goal because

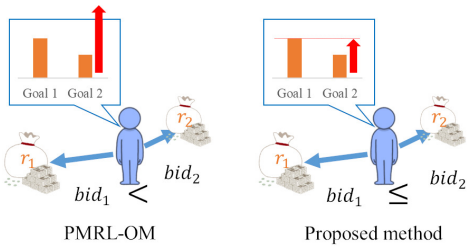

Figure 4: Difference in the goal value updating.

the algorithm does not employ ε-greedy action selection, unlike PMRL-OM. Thus, the proposed method modifies the goal value update function from Equation (9) to Equation (10) as follows:

$$bid_g = \frac{\xi - 1}{\xi} bid_g + \frac{\phi}{\xi}, \tag{10}$$

$$\phi = \begin{cases} \max_{g'} bid_{g'} & if\ memory\ is\ empty \\ t_g & else\ if\ received\ rewards \\ 0 & otherwise \end{cases} \tag{11}$$

This modification adds a process if the memory is empty, and the goal value is updated by the maximum value of all goal values. Figure 4 shows the effect of the proposed method when updating the goal value when the data for Goal 2 is empty. The left- and right-hand sides denote PMRL-OM and the proposed method. The word balloons are inside the agents and the graphs denote the goal values and their dynamics. PMRL-OM helps the agent to reach Goal 2 even if Goal 2 cannot be reached by overvaluation. The proposed method sets the goal values at the same level; thus, the goal value for Goal 2 cannot be greater than or equal to the other goal values. If Goal 2 is not valuable, the proposed method can recover quickly. Otherwise, the goal value for Goal 2 can exceed the goal value for the other goals.

4.2 Modification of Internal Reward Design

The internal reward design of the proposed method uses the same function of PMRL-OM as follows; however, the parameters differ:

$$ir_g = \max_{g' \in G, g' \neq g} r_{g'} \gamma^{l_{g'} - t_g} + \delta \tag{12}$$

where the discount rate γ is used in the A3C algorithm and the reward $r_{g'}$ is the external reward. As for A3C, the internal reward can maintain its rationality because of Equations (1) and (4). Equation (1) finally converges to the advantage value shown in Equation (2). Thus the internal reward can keep the gradient and large–small relationship between selecting actions in its policy. Equation (4) is used in the same manner.

4.3 LSTM-based Network

The proposed method replaces the A3C neural network to adapt the hetero-information from hetero-observations and hetero-transitions. An LSTM layer is then inserted into the network. Figure 5 shows the overview of the modified network. In the network model, the red line denotes an LSTM layer and the top graphic shows the difference between an agent's observation and the actual situation. Although the previous network comprises only dense layers, it can abstract the hetero-information in its inputs, but it cannot adapt to the hetero-information from the hetero-transition. This hetero-transition is because of the different transitions between the hetero-observation and the actual situation. At the top of this graphic, the agent cannot observe the movement to the left and inputs the same information.

LSTM (Hochreiter and Schmidhuber, 1997) is a kind of recurrent neural network used to learn about the current situation using one-step-back inputs to adapt to the time-sequential data. This paper replaces a part of the network to adapt to the hetero-transition. That is, the proposed method enables the agent to catch up with the actual change in the situation using LSTM to continuously learn the same input.

5 PROBLEM AND UNSTABLENESS

5.1 Maze Problem

In this paper, the agents train and practice on grid world mazes. The maze problems in the grid worlds are shown in Figure 6. On the left-hand side, the agent departs from the "Start" square to reach the "Goal" square based on the available rewards. On the right-hand side, the two agents depart from the squares labeled "Start A" and "Start B" to reach the goals named "Goal X" and "Goal Y." Although the agents can acquire the same reward value for each goal, the reward values accumulate when both agents reach different goals. The agents attempt to avoid hitting each other by avoiding being in the same square, including the goal square. The agents input information with one-hot vector using five dimensions: road, wall, goal, agent on road, and other in the even case, but only Agent A departs from "Start A."

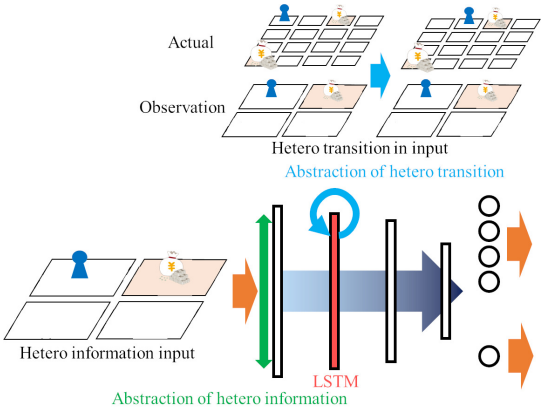

Figure 5: Vertical and horizontal abstraction in modification.

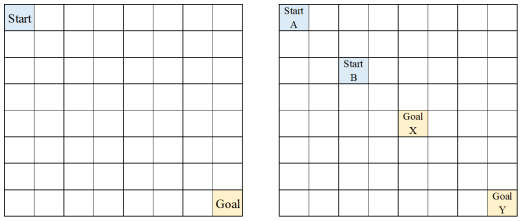

Figure 6: Environment.

5.2 Unstableness

There are two kinds of unstableness: hetero-information and the dynamic environment. This paper focuses on maze problems that include both kinds of unstableness.

5.2.1 Hetero-information

Hetero-information denotes the situation where agents have different input resolutions because of their individual sensor differences and different sensors in the same situation, among others. For vehicle navigation, each vehicle senses different resolution information; i.e., the same environmental states might be divided and the example becomes more difficult.

Figure 7 represents an example of hetero-information, with sensing in basic MARL on the right-hand side and sensing hetero-information on the left-hand side. The blue agent can move front, back, left, and right to reach the light-red goal square in the grid world. Although the agent can be given any information for all states in basic MARL, it uses one-fourth of all states because four states are observed as one state in the hetero-information case. Figure 7 shows that the agent can observe the back right-hand goal but cannot observe the front left-hand goal. Therefore, the hetero-information case means that agents cannot sense with low resolution, but can sense that something includes their goals.

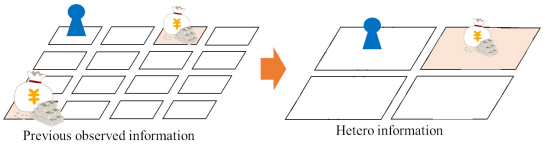

Figure 7: Hetero information observation.

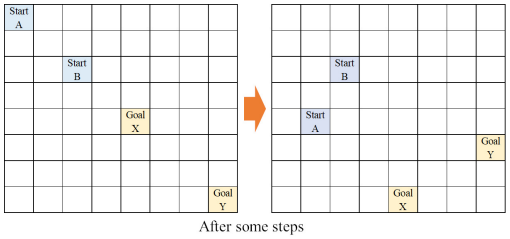

Figure 8: Dynamic environment.

5.2.2 Dynamic Environment

This paper focuses on the dynamics of the start and goal locations for the environmental change, which influence the state transition function. Figure 8 shows an example of the dynamic change. The location of agents and goals are changed after some steps and the reward is not changed. The environmental change occurred once in a trial and the modified method aims to learn a cooperative policy based on the environmental change.

6 EXPERIMENT

6.1 Experimental Setup

To investigate the effectiveness, this paper compared the proposed methods where the network has an LSTM module or not using the A3C algorithm in the maze problem (Figure 8). The environmental change happened after half of all the steps and Agent B only senses the hetero-information. The network topology differs for each agent: that is, Agent A has a hidden and dense layer of 16 nodes, while Agent B has 256 nodes in the proposed method without the LSTM module. The hidden layer of Agent B is replaced by an LSTM layer in the proposed method. Note that the A3C algorithm is based on (Fujita et al., 2019). The evaluation criteria are the spent step until all agents have reached the goal and the agents' acquired rewards.

6.2 Parameters

The setting parameters are summarized on Table 1. The total number of steps is 20 million (first line) and the copied agent learns every 25 steps as a maximum until 250 steps (second and third lines). The number of copied agents is 32 (fourth line). The parameters α, γ, and β are set by 0.0007, 0.99, and 0.01 (i.e., the fifth, sixth, and seventh lines, respectively). Finally, the internal reward gap δ is 1, and the number of external reward values is 10 for all goals (last lines).

Table 1: Experimental parameters.

Horizon of steps	20,000,000
Horizon of steps for copied agent	250
Horizon of steps in an iteration	25
Processes	32
Learning rate α	0.0007
Discount rate γ	0.99
Rate β	0.01
Internal reward gap δ	1
External reward value	10

6.3 Results

Figures 9, 10, and 11 show the spent steps until all agents have reached the goal and the acquired rewards of both agents using A3C and the proposed method, respectively. The vertical and horizontal axes denote the results and episodes, respectively. The blue, orange, and green lines indicate the resulting trajectories of 100 moving episodes using the A3C and the proposed methods (called "A3C," "Proposed method," and "Proposed method (no LSTM)" in Figures 9, 10, and 11, respectively).

Although the results are not clearly different from each other, they show that the proposed method is slightly better than any other method before the environmental change. After the change, the proposed method can converge to the minimum number of steps. In particular, the variance of the results using the A3C algorithm and the proposed method without the LSTM module is larger than that of the proposed method. As for the acquired rewards, the results are obtained in the same manner as the spent step and the convergence of the proposed method is better.

6.4 Discussion

Those results show that the proposed method performs better than PMRL-OM to enable the agents to learn optimal policies.

6.4.1 Effectiveness of LSTM

In this section, we discuss which layers should be inserted using LSTM through an experiment comparing

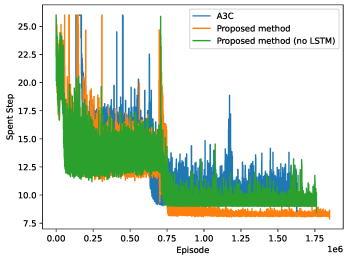

Figure 9: Result of spent step.

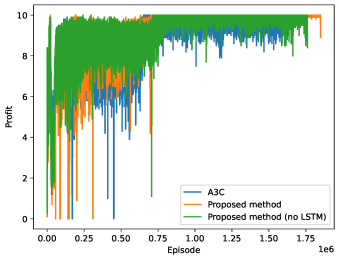

Figure 10: Result of profit (Agent A).

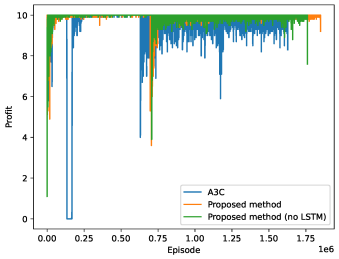

Figure 11: Result of profit (Agent B).

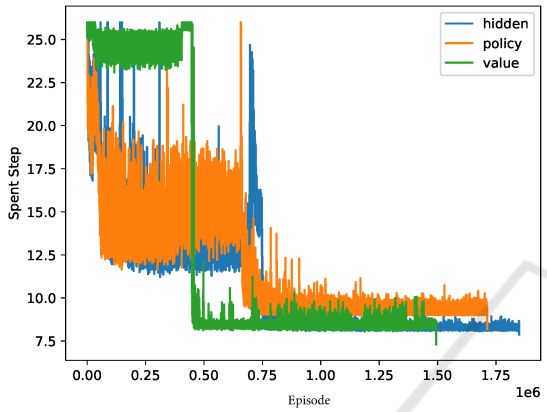

Figure 12: Results of spent steps (in the experiment of LSTM).

the agents' performance with different LSTM positions in the proposed method. There are three cases: (1) one is the proposed method; (2) another is which network uses the inserted LSTM in the previous policy output; and (3) the other is for the value output. Figure 12 shows the experiment results. The vertical and horizontal axes denote the spent steps and episodes, respectively. The blue, orange, and green lines indicate the resulting trajectories of 100 moving episodes for the cases (1), (2), and (3) ("hidden," "policy," and "value," respectively). The hidden result is the best, while the policy result is better than value. However, the value result is better than the policy result after the environmental change, it is the worst because the agents cannot learn a cooperative policy. This finding is because the state value enables the A3C algorithm for stable learning to converge the agents' learning results to the worst result. Therefore, case (1) enables the agents to abstract the hetero-transition information in their input. In case (2), LSTM cannot abstract the policy output by inserting the previous policy output. Because the environmental change occurred according to the total number of steps; this happened late during the situation where the agents using the proposed method can achieve the change using smaller steps than any other agents for each episode.

6.4.2 Limitation of Hetero Information

In Figure 9, the convergence is different between before and after the environmental change because the first maze is more difficult than the other mazes. In particular, it is difficult for Agent A to reach Goal Y for 14 translations using hetero-observation. Thus, the proposed method has a limitation of scale when using the hetero-transition. In this section, we examine the comparison of the proposed method and that without the LSTM module in the different mazes (Figure 13, where the setup is in the same manner as subsections 6.1 and 6.2.)

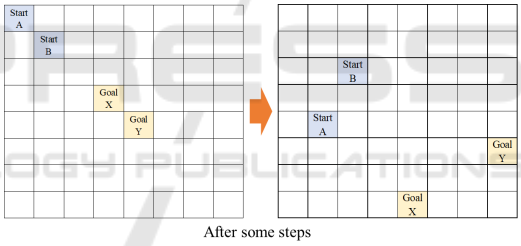

Figure 13: Simple environment.

Figure 14 shows the result. The vertical and horizontal axes denote the spent step and episodes, respectively. The blue and orange lines indicate the resulting trajectories of 100 moving episodes with the proposed method and without the LSTM module (called "LSTM" and "Dense"), respectively. From these results, the result of the proposed method without the LSTM module obtains some steps. On the other hand, the agents cannot cooperate with each other in some steps, e.g., in $0.75e+6$, before the experimental change, and the result could not converge to the minimum number of spent steps after that. On the other hand, the proposed method enables the agents to learn to cooperate and the optimal policy and convergence. The environmental change occurred by the total number of steps, which happened late in the situation where the agents using the proposed method can reach the change with fewer steps than any others for each episode.

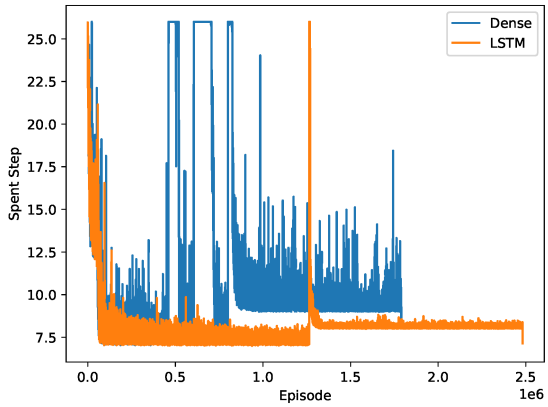

Figure 14: Spent step results (in the simple maze experiment).

7 CONCLUSION

This paper proposed a method based on the previous noncommunicative and cooperative learning method (PMRL-OM) based on DRL (A3C) in a multiagent system within an unstable environment in terms of hetero-transitions using different transitions according to the difference between observed and recent situations. The proposed method inserts an LSTM module into the A3C neural network. The experiments compared the proposed method with A3C and without the LSTM module. The derived results were as follows: (1) the proposed method performs better than the A3C algorithm and without the LSTM module. In particular, the proposed method enables the agents' learning to converge; (2) LSTM can adapt the time dimension of the input information.

This paper showed that the proposed method can not only adapt the hetero-transition of input information, but should also adapt the hetero-transition of output information. In particular, the hetero-transition should be assumed as a partially observable Markov decision process (POMDP), but the proposed method performs as a Markov decision process. Therefore, we will expand the proposed method to POMDP to adapt the hetero-observations in the future.

ACKNOWLEDGEMENTS

This research was supported by JSPS Grant on JP20K23326.

REFERENCES

Du, Y., Liu, B., Moens, V., Liu, Z., Ren, Z., Wang, J., Chen, X., and Zhang, H. (2021). *Learning Correlated Communication Topology in Multi-Agent Reinforcement Learning*, page 456–464. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC.

Fujita, Y., Kataoka, T., Nagarajan, P., and Ishikawa, T. (2019). Chainerrl: A deep reinforcement learning library. In *Workshop on Deep Reinforcement Learning at the 33rd Conference on Neural Information Processing Systems*.

Ghosh, A., Tschiatschek, S., Mahdavi, H., and Singla, A. (2020). *Towards Deployment of Robust Cooperative AI Agents: An Algorithmic Framework for Learning Adaptive Policies*, page 447–455. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Kim, D., Moon, S., Hostallero, D., Kang, W. J., Lee, T., Son, K., and Yi, Y. (2019). Learning to schedule communication in multi-agent reinforcement learning.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783.

Raileanu, R., Denton, E., Szlam, A., and Fergus, R. (2018). Modeling others using oneself in multi-agent reinforcement learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4257–4266, Stockholmsmassan, Stockholm Sweden. PMLR.

Uwano, F. (2021). A cooperative learning method for multi-agent system with different input resolutions. In *4th International Symposium on Agents, Multi-Agents Systems and Robotics*.

Uwano, F. and Takadama, K. (2019). Utilizing observed information for no-communication multi-agent reinforcement learning toward cooperation in dynamic environment. *SICE Journal of Control, Measurement, and System Integration*, 12(5):199–208.