

Action-Based Intrinsic Reward Design for Cooperative Behavior Acquisition in Multi-Agent Reinforcement Learning

Iori Takeuchi¹ and Keiki Takadama^{2,3} ^a

¹*Department of Informatics, University of Electro-Communications, Tokyo, Japan*

²*Information Technology Center, The University of Tokyo, Tokyo, Japan*

³*Department of Information & Communication Engineering, The University of Tokyo, Tokyo, Japan*

Keywords: Intrinsic Rewards, Multi-Agent System, Reinforcement Learning, Cooperative Behavior.


Abstract: In recent years, research has been conducted in multi-agent reinforcement learning that aims at efficient agent exploration in complex environments by using intrinsic rewards. However, such intrinsic rewards may inhibit the learning of behaviors necessary for acquiring cooperative behavior, and may not be able to solve the task of the environment. In this paper, we propose two types of internal reward designs to promote agents' learning of cooperative behaviors in multi-agent reinforcement learning. One is to use the average of the values of the actions selected by all agents to promote the learning of actions necessary for cooperative behavior but difficult to increase in value. The other is to provide an individual intrinsic reward when the value of the action selected by each agent is lower than the average of the values of all the actions at the time, aiming to escape from the local solution. The results of the experiment with StarCraft II scenario *6h_vs_8z* showed that by adding the proposed intrinsic reward to the intrinsic reward that encourages agents to explore unexplored areas, cooperative behavior can be obtained in more cases than before.

1 INTRODUCTION

Reinforcement Learning (RL) (Sutton and Barto, 1999) is a method of machine learning in which an agent repeatedly interacts with its environment to learn strategies that adapt to the environment and are used for risk management (Paragliola et al., 2018) and itinerary planning optimization (Chen et al., 2020) (Coronato et al., 2021). In addition, Multi-Agent Reinforcement Learning (MARL), a type of RL in which multiple agents exist in the same environment, aims to acquire cooperative behavior in complex environments by having the agents learn policies simultaneously. However, in many MARL environments, the agents cannot learn policies that take into account other agents using only the rewards they receive from the environment, making it difficult to acquire cooperative behavior. Against this background, research on MARL that applies intrinsic rewards has been progressing in recent years. Intrinsic rewards are rewards given by the agent's reward function. Rewards received from the environment in contrast to intrinsic rewards are called extrinsic rewards. In previous

studies, intrinsic rewards applied to MARL are often designed to improve the agent's exploration ability. Exploiting Policy Discrepancy for Efficient Exploration (EXPODE) (Zhang and Yu, 2023) is one of the MARL methods that applies intrinsic rewards, and the intrinsic rewards of EXPODE are designed to encourage agents to explore unexplored areas. Specifically, in EXPODE, an agent learns a different policy simultaneously in addition to the policy it learns through interactions with the environment, and the prediction error of the value function of that policy is given as an intrinsic reward. This gives agents high rewards according to how infrequently they reach a state in the environment, and is expected to enable efficient exploration even in environments with a vast state-action space, leading to the acquisition of effective cooperative behavior. However, intrinsic rewards used in EXPODE to improve the agents' exploration capabilities are not directly designed to acquire cooperative behavior, and there is a problem that the number of times the agents select actions necessary to acquire cooperative behavior for exploration is reduced, which stagnates learning and hinders the acquisition of cooperative behavior.

Considering this problem, this paper proposes two

^a  <https://orcid.org/0009-0007-0916-5505>

types of intrinsic reward designs to acquire cooperative behavior. In the first intrinsic reward design, the average of the values corresponding to the actions selected by all agents respectively is used as the collective intrinsic reward for all agents. The second design is more limited and aims to provide intrinsic rewards. The value of the selected action for each agent is compared with the average value of all actions at that time, and if the value of the selected action is lower than the average, an individual intrinsic reward is given to each agent. To verify the effectiveness of the proposed intrinsic reward, we applied the proposed intrinsic reward as an extension of the conventional method EX-PODE and conducted experiments using the StarCraft II (Vinyals et al., 2017) scenario *6h_vs_8z* as an environment.

This paper is structured as follows: Section 2 presents the problem formulation and related work. Section 3 presents the proposed method. Section 4 presents experiments and their results. Finally, Section 5 concludes this paper.

2 RELATED WORKS

2.1 Dec-POMDP

In this research, we consider a problem formulated by Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek et al., 2016). In Dec-POMDP, a problem is defined as a tuple $(I, S, A, P, R, Z, O, \gamma)$. I is a set of agents i , $s \in S$ is a set of true states of the environment, and A is a set of agent actions. When agent i selects action a_i , $\mathbf{a} = (a_1, a_2, \dots, a_i)$ is represented as the joint action of all agents. At this time, the next state of agents is determined by transition function $P(s'|s, \mathbf{a})$. In addition, each agent obtains individual observations $o_i \in Z$ using observation function $O(s, a)$. Each agent has its behavioral observation history $\tau_i \in \mathcal{T} \equiv (Z \times A)^*$, and acts according to policy $\pi_i(a_i|\tau_i)$ based on that history. Agents obtain extrinsic rewards r using a collective reward function $R(s, \mathbf{a})$. $\gamma \in [0, 1]$ is the discount rate, and agents learn policies to maximize the expected discounted reward $\sum_{t=0}^T \gamma^t r_t$, where T is the length of the episode.

2.2 Deep Reinforcement Learning

2.2.1 Deep Q-Network

Deep Q-Network (DQN) (Mnih et al., 2015) is a deep reinforcement learning algorithm that expresses an action-value function using a neural network based

on a parameter θ . In DQN, the history of actions and state transitions is stored in a replay buffer, and learning is performed by sampling several histories from the replay buffer. θ is learned to minimize the squared TD error:

$$\mathcal{L}(\theta) = [y - Q(s, a; \theta)]^2, \quad (1)$$

where $Q(s, a; \theta)$ is the action value represented by the network of parameters θ for state s and action a . $y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$. θ^- is the parameter of the target network, which is periodically copied from θ and kept constant for several iterations.

2.2.2 Deep Recurrent Q-Network

Deep Recurrent Q-Network (DRQN) (Hausknecht and Stone, 2015) is a deep reinforcement learning algorithm that uses a recurrent neural network to adapt to a partially observable environment. In addition, DRQN promotes learning that considers longer-term time series by using a collective action observation history of one episode as a sample during learning.

2.3 Multi-Agent Deep Reinforcement Learning

2.3.1 Centralized Training with Decentralized Execution

In recent years, MARL has often used Centralized Training with Decentralized Execution (CTDE) (Oliehoek et al., 2008) to approach partially observable environments. In CTDE, agents act based on their local observations and action values during the execution process, and in the training process, agents learn using global information by sharing the true state of the environment and the action values of all agents. As a result, CTDE realizes the interaction between agents and the environment that takes partial observability into account and the optimization of policies using global information.

2.3.2 QMIX

QMIX (Rashid et al., 2018) is a MARL algorithm based on CTDE. In QMIX, each agent has an individual action value Q_i , and in the execution process, each agent selects an action using its action value Q_i . In the learning process, several samples of one episode's history obtained in the execution process are taken from the replay buffer, and the joint action value Q_{tot} is calculated from the individual action value Q_i using a mixing network, and learning is performed to minimize the following loss function. The weights of the

Mixing Network are trained by a hypernetwork (Ha et al., 2016) that uses the true state as input. In addition, the individual action value Q_i is also updated by backpropagating the error arising from this loss function to the network that expresses the individual value function Q_i :

$$\mathcal{L}(\theta) = [y_{tot} - Q_{tot}(\tau, \mathbf{a}, s; \theta)]^2, \quad (2)$$

where $y_{tot} = r + \gamma \max_{\mathbf{a}'} Q_{tot}(\tau', \mathbf{a}', s'; \theta^-)$. θ, θ^- are parameters of the network, and the target network, respectively, as in the equation 1.

In this case, the joint action value and the individual action value have the relationship shown in the following formula.

$$\operatorname{argmax}_{\mathbf{a}} Q_{tot}(\tau, \mathbf{a}) = \begin{pmatrix} \operatorname{argmax}_{a_1} Q_1(\tau_1, a_1) \\ \vdots \\ \operatorname{argmax}_{a_i} Q_i(\tau_i, a_i) \end{pmatrix}. \quad (3)$$

Equation (3) shows that when each agent selects the action with the highest individual action value based on its action observation history, the joint action value also takes the highest value. To satisfy the relationship in equation (3), QMIX imposes the constraints expressed in the following equation (4) through the Mixing Network.

$$\frac{\partial Q_{tot}}{\partial Q_i} \geq 0, \quad \forall i \in I. \quad (4)$$

2.4 Intrinsic Rewards for MARL

2.4.1 Exploiting Policy Discrepancy for Efficient Exploration

Exploiting Policy Discrepancy for Efficient Exploration (EXPODE) (Zhang and Yu, 2023) is a MARL algorithm that improves the efficiency of learning in an environment with a huge observed action space by encouraging agents to explore unexplored areas with intrinsic rewards. EXPODE consists of three components called Exploiter, Explorer, and Predictor.

Exploiter learns the policy for agent-environment interaction. It extends QMIX based on the Double Q-Learning (Hasselt, 2010) approach and is configured to have two networks similar to QMIX. This allows the Exploiter to output two joint action values for one sample during the training process. Applying the smaller two joint values to update the policy prevents overestimation of the value. The loss function of Exploiter is as shown in the equation (5).

$$\mathcal{L}_{exploiter}(\theta_k) = [y_{exploiter} - Q_{tot}(\tau, \mathbf{a}, s; \theta_k)]^2, \quad (5)$$

where $y_{exploiter} = r^{ext} + \alpha r^{EXPODEint} + \gamma \min_{k'=1,2} Q_{tot}^{\theta_{k'}}(\tau', \mathbf{a}', s'; \theta_{k'})$. r^{ext} is the extrinsic reward, $r^{EXPODEint}$ is the intrinsic reward, and α is a hyperparameter related to the intrinsic reward. k and k' indicate that there are two learning networks and two target networks, respectively, and the joint action value with the smallest value is adopted during learning.

Explorer is a component with the same structure as Exploiter and learns the policy used to calculate intrinsic rewards. The process of policy update is the same as that of Exploiter. The individual action value of each agent included in the network that outputs the larger value of the two joint action values output by Explorer is passed to Predictor for the calculation of intrinsic rewards. The loss function of Explorer is as shown in the equation (6).

$$\mathcal{L}_{explorer}(\phi_k) = [y_{explorer} - Q_{tot}(\tau, \mathbf{a}, s; \phi_k)]^2, \quad (6)$$

where $y_{explorer} = r^{ext} + \gamma \min_{k'=1,2} Q_{tot}^{\phi_{k'}}(\tau', \mathbf{a}', s'; \phi_{k'})$.

Predictor calculates the intrinsic reward using the individual action value given by Explorer. Predictor consists of a network for each agent that expresses the individual action value included in Expitter and Explorer and outputs the individual action value for each agent from one sample. In addition, Predictor calculates the intrinsic reward $r^{EXPODEint}$ using the following equation (7) from the individual action value given by the Explorer and the individual action value output by Predictor.

$$r_i^{EXPODEint} = \|Q_i^{\psi}(\tau'_i, \cdot) - \max_{k'=1,2} Q_i^{\phi_{k'}}(\tau'_i, \cdot)\|^2. \quad (7)$$

The intrinsic reward of each agent is summarized as follows:

$$\mathcal{L}_{predictor}(\psi) = r^{EXPODEint} = \frac{1}{I} \sum_{i=1}^I r_i^{int} \quad (8)$$

EXPODE calculates the intrinsic reward based on the frequency of the agent's achievement in a different policy than the one used to select an action in the environment. This is expected to lead to exploration in more diverse directions than if a single policy was used.

3 PROPOSED METHOD

From the previous chapter, EXPODE is expected to acquire cooperative behavior in a task with a huge state action space using intrinsic rewards that encourage agents to explore various areas. However, such inherent rewards may inhibit the learning of measures

by prioritizing the exploration of unexplored regions over the utilization of actions required for cooperative behavior and may lead to the teaching of measures that are stuck in local solutions. In this chapter, we propose two types of intrinsic reward designs to acquire cooperative behavior, based on the issues of inherent rewards in EXPODE described above.

3.1 Collective Intrinsic Reward Design for all Agents

In MARL, cooperative behavior is a combination of actions performed by multiple agents over a certain period, and acquiring cooperative behavior requires repeated execution of the primitive actions that constitute it to increase their action values. However, in environments where the acquisition of complex cooperative behavior is required, the actions necessary for cooperative behavior may not always directly lead to the acquisition of external rewards. In environments formulated with Dec-POMDP, a collective reward is given to all agents, so even actions that do not yield external rewards for a single agent can earn some external reward. However, since agents generally try to acquire more rewards, they may increase the value of actions that yield immediate external rewards and learn local policies that fail to acquire cooperative behavior. In such cases, EXPODE gives a lot of intrinsic rewards during initial state transitions, so it can increase the value of actions that do not temporarily yield external rewards. However, the intrinsic reward obtained by repeating transitions decreases, so if the agent cannot find the effectiveness of the action and increase its value by then, it will fall into a local solution. As an approach to this problem, Equation (9) shows the proposed formula for calculating intrinsic rewards $r_{collectiveint}$.

$$r_{collectiveint} = e^{\frac{1}{T} \sum_{i=1}^T Q_i(\tau_i, a_i)}. \quad (9)$$

From the equation (9), the proposed intrinsic reward uses the average of the individual action values corresponding to the actions selected by the agent. In other words, the proposed intrinsic reward is given taking into consideration the value of not only the agent's actions as a whole but also the actions selected by other agents at the same time. This helps to avoid learning local policies and aims to acquire cooperative behavior by providing rewards that also consider the value of actions chosen by other agents when the value of actions necessary for acquiring cooperative behavior is difficult to increase.

3.2 Individual Intrinsic Rewards for Each Agent

The intrinsic reward proposed in section 3.1 is intended to promote learning of actions necessary for some agents' cooperative behavior but difficult to value. However, since this intrinsic reward is given for every action that an agent chooses, it may promote learning of actions that are more likely to increase in value. In this case, if intrinsic rewards are given to behaviors that are difficult to increase in value, no value reversal occurs, and the possibility arises that the agent may not escape from the local solution.

Therefore, in this section, we propose an intrinsic reward given to each agent according to the value of that agent's chosen action. Specifically, each agent compares the value corresponding to its own selected action at each step with the average of the values of all actions in that state. If the value of the chosen action is lower than the value of all actions, an intrinsic reward is given using that average. The formula for calculating the intrinsic reward $r_{individualint}$ is shown in equation (10).

$$r_i^{individualint} = e^{\frac{1}{A} \sum_a Q_i(\tau_i, a_i)}. \quad (10)$$

To give this individual intrinsic reward only to the corresponding agent, a phase to update each agent's Q-Network is added to the EXPODE component, Exploiter. The loss function used to train the Q-Network of each agent is shown in equation (11). This is done just before the learning by loss function of Exploiter shown in equation (5).

$$\mathcal{L}_{individual}(\theta_{k_i}) = [y_{individual} - Q_i(\tau_i, a_i; \theta_{k_i})]^2, \quad (11)$$

where $y_{individual} = \frac{1}{T}(r^{ext} + \alpha r^{EXPODEint}) + \alpha r_i^{individualint} + \gamma \min_{k'=1,2} Q_i^{k'}(\tau_i, a_i; \theta_{k'})$. As a result, the total Exploiter loss function in this method is revised to the following equation (12).

$$\mathcal{L}_{exploiter}(\theta) = \mathcal{L}_{exploiter}(\theta) + \sum_{i=1}^I \mathcal{L}_{exploiter}(\theta_i). \quad (12)$$

4 EXPERIMENT

4.1 StarCraft II

In this study, we conduct experiments to verify the effectiveness of the proposed method and use StarCraft II as the environment. StarCraft II is a real-time strategy game provided by Blizzard Entertainment, and in

recent years it has been used as an environment for the MARL algorithm by a benchmark called SMAC (Samvelyan et al., 2019). Several scenarios are available in StarCraft II, and in this study, we conduct experiments using a scenario called $6h_vs_8z$.

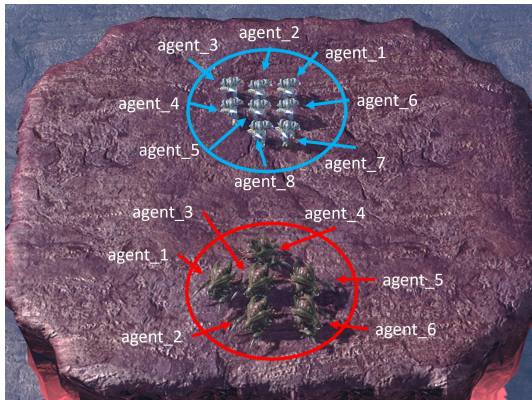


Figure 1: Agent's initial position in $6h_vs_8z$ (red: ally, blue: enemy).

The initial state of $6h_vs_8z$ is shown in the Figure 1. $6h_vs_8z$ is a scenario in which six ally agents (red circles in the Figure 1) aim to acquire a strategy to defeat eight enemy agents (blue circles in the Figure 1). The enemies are controlled by NPCs in the game. One episode is defined as one trial from the start to the end of the scenario, and one step is defined as the unit time of an episode in which all agents, enemy and ally, act once. The end condition of $6h_vs_8z$ is when the ally's victory or defeat conditions are met. The ally's victory condition is when all enemies are incapacitated, and the defeat condition is when all allies are incapacitated or 150 steps have passed. In $6h_vs_8z$, allies have less physical strength and less damage per attack than enemies, but they can attack from a certain distance even when they are not adjacent to the enemy. Also, enemies have more physical strength and more damage per attack than allies, but they can only attack allies adjacent to them. From the above, allies in $6h_vs_8z$ need to learn cooperative behavior that can efficiently defeat the enemy under the constraints of these characteristics and numerical disadvantage.

In this study, the agent obtains only positive extrinsic rewards from the environment for each step. There are three extrinsic rewards: damage dealt to the enemy, defeating the enemy (10 points), and winning for allies (200 points). The total value of these rewards is normalized to be in the range of 0 to 20. In all comparison methods, intrinsic rewards are given for each step, just like extrinsic rewards, but are not included in the normalization.

4.2 Experiment Setup

There are three comparison methods: EXPODE, the proposed intrinsic reward for the whole agent plus the EXPODE Exploiter (EXPODE+collective), and the proposed intrinsic reward for each agent plus the EXPODE Exploiter (EXPODE+ individual). For each method, α is set to 0.01. The winning rate of the allies at the time of evaluation is used as the evaluation index of the measure to be acquired. Each evaluation is performed in 32 episodes. The training length is 2,100,000 steps. We use ϵ -greedy for action selection, where ϵ is 1 at the beginning of the experiment and decreases linearly to reach a minimum value of 0.02 at 400,000 steps. The learning rate is 0.0005 and the discount rate is 0.99. Note that these parameters are the same for all methods.

4.3 Result

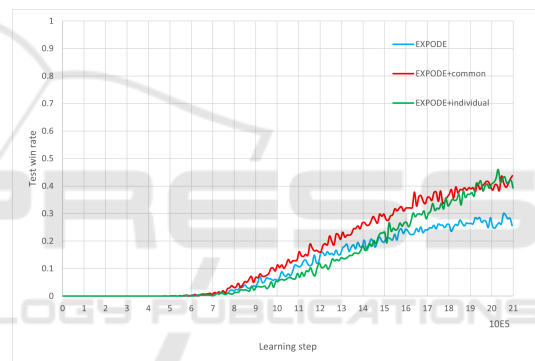


Figure 2: Test win rate of each method in $6h_vs_8z$.

Figure 2 shows the win rate of the allies during the training of each method in $6h_vs_8z$. The values in Figure 2 are the 30 seed average of the win rate at each measure evaluation. From Figure 2, it can be seen that the win rate for all methods increases from about 700,000 steps. In addition, EXPODE+collective has a higher win rate than EXPODE, indicating that the learning process continues to progress. On the other hand, EXPODE+individual has exceeded the win rate of EXPODE since about 1,500,000 steps, and finally, the win rate has remained at the same level as that of EXPODE+collective. From this, we can say that adding the proposed intrinsic reward to EXPODE improves the learning performance of EXPODE in $6h_vs_8z$ and enables it to acquire more effective cooperative behavior.

Figure 3 shows the distribution of win rates at the last evaluation for 30 seeds for each method. The crosses in Figure 3 indicate average values. Table 1 compares the win rate at the final evaluation for each

Table 1: Distribution of the number of seeds with large and small test win rates of the policies at the final evaluation.

	> EXPODE	= EXPODE	< EXPODE
collective intrinsic reward	17	2	11
individual intrinsic reward	18	0	12

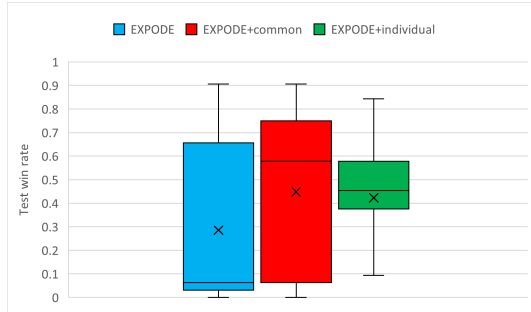


Figure 3: Distribution of test win rates at the final evaluation of each method.

method for each seed and summarizes the relationship between the win rate at the final evaluation and the win rate at the final evaluation for each method. From Figure 3 and Table 1, EXPODE+collective has a higher win rate than EXPODE in many of the selections, and more than half of the selections finally obtain a strategy with a win rate of more than 50%. EXPODE+individual also has a higher win rate in many seeds compared to EXPODE. In addition, EXPODE+individual has a much smaller interquartile range than the other two methods, and it is found that EXPODE+individual obtains measures with a win rate of 40-60% in many of the seeds. From the above, it can be said that the proposed intrinsic reward can achieve a more stable acquisition of cooperative behavior than EXPODE in many cases.

4.4 Discussion

4.4.1 Cooperative Behavior

The actions of the episode are shown. (a) is EXPODE, (b) is EXPODE+collective, and (c) is EXPODE+individual. Table 2 shows the flow of the episode in snapshots at four important time points. The leftmost Figure in Table 2 is the state at the start of the episode, and as the step progresses, the state moves to the right Figure, and finally reaches the state at the end of the episode shown in the rightmost Figure.

From Table 2, it can be seen that in this episode, the policy with EXPODE results in the defeat of the allied agent, whereas EXPODE+collective and EXPODE+individual result in victory. Furthermore, comparing the actions of the allied agent when winning and losing, the following differences in actions

can be confirmed in each Event: (1) Episode start - Encounter: With EXPODE, the ally encounters the enemy at the same location as the initial position, whereas with EXPODE+collective and EXPODE+individual, the ally transitions backward or to the left or right to encounter the enemy. Also, at this time, with EXPODE and EXPODE+collective, the ally's attacks are concentrated on a single enemy, and one enemy is incapacitated at the time of encounter. With EXPODE+individual, one enemy has not been incapacitated at the time of encounter, but since the health of one of the four enemies heading towards the ally is extremely low, it can be seen that a similar concentrated attack is being carried out. (2) Encounter - One ally is unable to fight: With EXPODE, the numerical disadvantage of allies continues at this point, whereas with EXPODE+collective, the number of allies and enemies is 5 to 5, and it can be seen that the numerical disadvantage against the enemies has been resolved by the allies' attacks since the time of encounter. Also, with EXPODE+individual, the numerical disadvantage has not been resolved as with EXPODE, but it can be seen that several enemies are located away from the allies. This is because one of the two agents that were divided at the time of the Encounter attracted several enemies, and in the meantime, the remaining allies and enemies were fighting in a numerically equal situation. As a result, it is considered that even though the allies in EXPODE+individual were unable to resolve the numerical disadvantage at this point, they were ultimately able to win due to the advantage gained in the previous battles. From the above, the characteristics of cooperative behavior at the time of victory that can be acquired through the proposed intrinsic reward can be summarized as follows. (1) State transitions to keep a distance from the enemy and concentrated attacks on specific enemies. (2) Attacks to eliminate numerical inferiority. Furthermore, comparing the actions of EXPODE+collective and EXPODE+individual after (2), it is considered that these cooperative behaviors are largely dependent on the cooperative behaviors taken in (1). Therefore, in cooperative behavior in *6h_vs_8z*, it is considered that an important point is what kind of formation is used to keep a distance from the enemy while continuing concentrated attacks at the start of the episode.

Table 2: Example of behavior in one episode of 6h_vs_8z (a: EXPODE, b: EXPODE+collective, c: EXPODE+individual).

(a)				
(b)				
(c)				
Event	Episode start	Encounter	One ally is unable to fight	Episode finish

4.4.2 Acquiring Cooperative Behavior Through Intrinsic Rewards

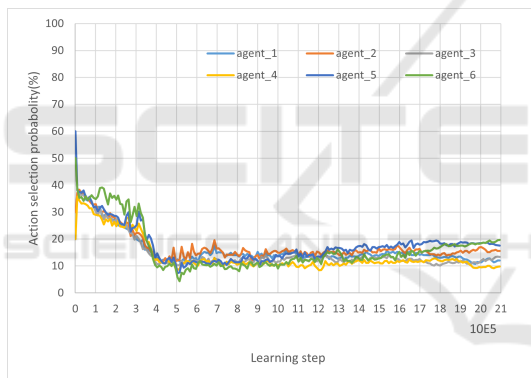


Figure 4: State transition probability for each agent in EXPODE from 1 to 10 steps.

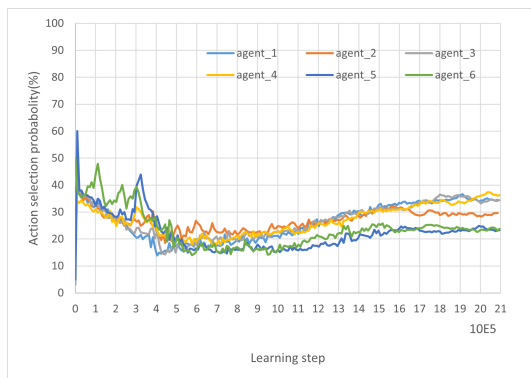


Figure 5: State transition probability for each agent in EXPODE+collective from 1 to 10 steps.

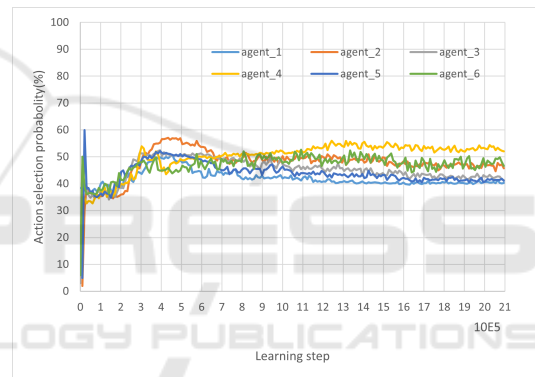


Figure 6: State transition probability for each agent in EXPODE+individual from 1 to 10 steps.

Figures 4, 5, and 6 respectively show the probability of each agent transitioning states between 1 and 10 steps in the execution process between each evaluation point during learning of each method with a certain seed. Figure 4 shows that in EXPODE, the state transition probability of any agent does not increase after 400,000 learning steps, where the value of ϵ reaches the minimum value and action selection becomes most dependent on individual action value. On the other hand, Figure 5 shows that in EXPODE+collective, the state transition probability of the agents increases even after 400,000 learning steps. From this, it is considered that in EXPODE+collective, the proposed intrinsic reward increases the value of actions necessary for cooperative behavior, and as a result, the number of times they are selected can be increased. Also, from Figure 6, it can be seen that in EXPODE+individual, the state transition probability does not increase after 400,000

learning steps, but the state transition probability of the agents is higher than that of the other methods. This is thought to be because the cooperative behavior that can be acquired by EXPODE+individual is significantly different from that of the other methods, but since the high probability is maintained even after 400,000 learning steps, it is considered that the value of state transitions is sufficiently increased by the proposed intrinsic reward.

5 CONCLUSIONS

This paper proposed two types of intrinsic rewards aimed at acquiring cooperative behavior. One utilizes the average value of actions selected by agents at each step, aiming to further increase the value of actions necessary for cooperative behavior, and the other aims to avoid local solutions by giving individual rewards to each agent when they meet certain conditions. In addition, we conducted experiments to verify the effectiveness in *6h_vs_8z*, a StarCraft II scenario, by adding the proposed intrinsic reward design to a conventional method (EXPODE) with an intrinsic reward design that promotes agents' exploration of unexplored regions. As a result, we confirmed that by applying the proposed intrinsic reward, EXPODE can learn policies with a higher win rate for allies in many cases compared to before application, and can stably acquire cooperative behavior. Furthermore, we confirmed that applying the proposed intrinsic reward can increase the probability of selecting actions necessary for cooperative behavior in *6h_vs_8z* compared to the conventional method, contributing to the learning of cooperative behavior. Future work includes experiments using other scenarios of StarCraft II to verify the generality of the effect of the proposed intrinsic reward design.

REFERENCES

- Chen, S., Chen, B.-H., Chen, Z., and Wu, Y. (2020). Itinerary planning via deep reinforcement learning. In *Proceedings of the 2020 International Conference on Multimedia Retrieval, ICMR '20*, page 286–290, New York, NY, USA. Association for Computing Machinery.
- Coronato, A., Di Napoli, C., Paragliola, G., and Serino, L. (2021). Intelligent planning of onshore touristic itineraries for cruise passengers in a smart city. In *2021 17th International Conference on Intelligent Environments (IE)*, pages 1–7.
- Ha, D., Dai, A. M., and Le, Q. V. (2016). Hypernetworks. *CoRR*, abs/1609.09106.
- Hasselt, H. (2010). Double q-learning. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.
- Hausknecht, M. and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 AAAI fall symposium series*.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.
- Oliehoek, F. A., Amato, C., et al. (2016). *A concise introduction to decentralized POMDPs*, volume 1. Springer.
- Oliehoek, F. A., Spaan, M. T., and Vlassis, N. (2008). Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353.
- Paragliola, G., Coronato, A., Naeem, M., and De Pietro, G. (2018). A reinforcement learning-based approach for the risk management of e-health environments: A case study. In *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 711–716.
- Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. (2018). QMIX: Monotonic value function factorisation for deep 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 4295–4304. PMLR.
- Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C., Torr, P. H. S., Foerster, J. N., and Whiteson, S. (2019). The starcraft multi-agent challenge. *CoRR*, abs/1902.04043.
- Sutton, R. S. and Barto, A. G. (1999). Reinforcement learning: An introduction. *Robotica*, 17(2):229–235.
- Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J. P., Schrittwieser, J., Quan, J., Gaffney, S., Petersen, S., Simonyan, K., Schaul, T., van Hasselt, H., Silver, D., Lillicrap, T. P., Calderone, K., Keet, P., Brunasso, A., Lawrence, D., Ekeremo, A., Repp, J., and Tsing, R. (2017). Starcraft II: A new challenge for reinforcement learning. *CoRR*, abs/1708.04782.
- Zhang, Y. and Yu, C. (2023). Expode: Exploiting policy discrepancy for efficient exploration in multi-agent reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '23*, page 58–66, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.