Cooperative Behavior on Limited Resource Using Deep Reinforcement Learning in Multi-Agent System

Yining Li[1,a)] Toshiharu Sugawara[1,b)]

Abstract: We studied that several cooperative agents in a multi-agent system (MAS) try to achieve a common goal but with scarce resources to share. Recent years have seen more researchers use deep reinforcement learning (DRL),more specifically,deep Q-network(DQN) to study cooperative behaviors in a distributed system. In a multi-agent environment, which is an unsupervised scenario, we usually apply reinforcement learning to deal with large observation space problems. Thus our research investigated that DRL techniques make self-autonomous agents to learn how to solve the problem efficiently and effectively by avoiding or reducing conflicts. In this paper,we conducted experiments by changing agents' view scope of the world, and compared the learning performance of cooperative behaviors among these learning agents.

Keywords: deep reinforcement learning, cooperation, conflicts resolution, scarce resource

1. Introduction

Multi-agent system (MAS) is a more realistic abstraction model of our real world comparing to a single agent model which only study individual behaviors without considering interference among agents. The main application domains of MAS are ambient intelligence, grid computing, electronic business, the semantic web, bio-informatics and computational biology, monitoring and control, resource management, education, space, military, manufacturing and Internet of Things (IoT) [1]. In such complex systems, handcrafted rules which are used to apply to cooperate and coordinate with other agents' behaviors and control the workflow of the system are complicated and thus very difficult to be implemented in real-world systems.

A MAS is a complex system in which conflict or collision may not be able to be avoided. This problem happens even more often when the common resource is scarce. Many conflicts can influence the whole system's working performance that induces large amounts of unnecessary economic loss. Therefor many researchers studied how to reduce conflicts in a large distributed system and a few studies mainly focus on path-planning strategy. For example Jingjin Yu *et al.* [3] studies the problem of optimal multi-robot path planning on graphs focusing on structural and computational complexity issues. Xie *et al.* [14] acquired agent's cooperative behavior by using extended Q-learning in which the Q table is shared among agents. Wolfgang *et al.* [2] put forward a method named multi-agent path-finding (MAPF) to reduce conflicts in MAS. Their approach focus on path planning dynamically according to CBS-TA, which creates a search forest on demand, and got complete and optimal results.

[1] Waseda University, Tokyo, Japan
[a)] liyining@ruri.waseda.jp
[b)] sugawara@waseda.jp

Although path-planning methods are used to solve conflict problems, such approaches are primarily implemented in a supervised learning scenario, which require large efforts on algorithm design. In a more realistic world environment, unsupervised learning like reinforcement learning (RL) seems to be a more practical solution for some specific problems. RL is aimed to implement human-level control in a real world [13].And in recent years, research on multi-agent reinforcement learning (MARL) with good results were reported. However, natural extension from single agent environments methods to MARL has been often proven failure [4], [5] , mainly due to the high-dimensional state-action spaces that force agents to be explored.

On the other hand, deep reinforcement learning (DRL) has produced many successful results in fields such as robotics and games, and DRL for single agents has enabled learning in many sophisti- cated domains [15]. In 2015, DQN beat human experts in many Atari games, and in 2017, a professional team beat a DeepMind AI program in Starcraft 2 easily [7], [8], [9].Many previous research using deep reinforcement learning have shown satisfying results in multi-agent system.To cope with the above-mentioned problems caused by MAS large state-action space along with achieving coordination, Miyashita *et al.* [6] proposed an efficient method for multiple and self-autonomous agents operating in a distributed and large environment by using the DRL approaches in MAS (DMARL). It is desirable for the agents themselves to learn and identify the appropriate cooperative actions and form a regime for cooperation in accordance with the many environmental factors such as task structures, frequency of task occurrence, and environmental characteristics. However, In DMARL, we need a large training set and random data noises can affect convergence and correctness of the learning results. Even though DMARL can obtain satisfying results in promoting cooperation, large amount of dataset should not be ignored when in real implementation. For example, Diallo *et al.* [10] shows that a large number of agents can behave cooperatively with large amount of input data under long time experiment. Other strategies with DQN like Zhang-Wei Hong*et al.* [12], who proposed *a deep policy inference Q-network*(DPIQN) that targets multi-agent systems composed of controllable agents, collaborators, and opponents that interact with each other by studying policy with DQN. Miyashita *et al.* examined what kinds of strategic cooperative behaviors emerge in multiple agents by changing the observation view scope of each agent as the inputs to the DQNs.

Thus, in this paper, we implement our problem model which considers to reduce conflicts regarding with scarce resources in the system based on the previous model proposed by Miyashita *et al.* [6] . In Miyashita's model, each agent has their own DQN to act according to current observation. They also introduced *relative view*, a part of global state as input in some of their experiments. We also adopt *relative view* as one of our agents' attributes. In this way, we study the cooperative agents' behaviors using DQN in a distributed environment.

2. Background and Related Work

2.1 Q-learning in Multi-Agent Environments

In a MAS, the environment state is affected by the joint action of all agents, so we should formulate Q-fucntion by being integrated with other agents actions in MAS. So Q-functions formulation should be reconsidered in MAS. As one RL algorithm, Q-learning is to maximize cumulative rewards of an agent considering discounted factor γ at step t $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}, \gamma \in [0, 1)$. Q-learning is based on Bellman Equation, whose form is as follows:

$$Q^{\pi}(s, a) = \mathbb{E}[R_t | s = s_t, a = \pi(s_t) = a_t],$$

where at step t, the environment state s is s_t, a is the action got from policy π.

Intuitively, the optimal action value $Q^*(s, a)$ can be described by :

$$Q^*(s, a) = \max_{\pi} \mathbb{E}[R_t | s = s_t, a = \pi(s_t) = a_t].$$

So the main task for Q-learning is to create the optimal Q-value table for every possible (s, a) , we can find an optimal solution for a certain problem.

However, when (*state, action*) space is too large, like in a broad environment with multiple agents, it is almost impossible to create such a Q-table using conventional Q-learning methods, because it will not be converge and training will not reach to end. Therefore, we try to use deep Q-Network to resolve large-space issue.

2.2 Deep Q-Network

DQN is a model-free approach to reinforcement learning based on deep neural networks for estimating the Q-function over high-dimensional and complex state space. DQN is parameterized by a set of network weights θ, which can be updated by a variety of RL algorithms. The parameters θ are learned by gradient descent or other optimization methods that iteratively minimizes the loss function ($L(\theta)$) using samples (s, a, r, s'),which is a state transition tuple from step t to next step.

At time t, to get optimal Q-value approximation from the network, parameters $\theta_{i,t}$ in the network of agent i are updated to minimize the mean squared loss function $L_{i,t}(\theta_{i,t})$, which is defined as

$$L_{i,t}(\theta_{i,t}) = \mathbb{E}_{(s_i, a_i, r_i, s_i')}[(r_i + \gamma \max_{a_i'} Q_i(s_i', a_i'; \theta_{i,t}^-) - Q_i(s_i, a_i; \theta_{i,t}))^2],$$

Note that we use the double DQN, in which target network parameters $\theta_{i,t}^-$ are copied from θ periodically to stabilize learning for deep Q-network;

2.3 Neural Network - CNN

Neural network has a long history in studying artificial intelligence and is still a cutting-edge field in AI, mainly because of the rise of deep neural network (DNN). In theory, a deep neural network can approximate any function as long as more layers are added. Sample data's different local features are extracted when they passing through different layers of a DNN.

Convolutional neural network (CNN) is such a DNN that points at input samples with spacial local features, and often used in image recognition.

3. Problem Formulation

Inspired by Toru Ishida [16], our problem is set in a grid world (or game board), which is a $N \times N$ game board described by the Fig.1. In this figure, red dots are agents, blue dots are tasks, the only black dot is a hole to which agents can push tasks inside it. We assume that this hole is the shared scarce resource because there are only four cells from which agents can directly access to.

The environment has several independent and autonomous agents located in the game board. Besides the agents, who work as task performers, there is one hole located in a fixed position and some certain number of tasks scattered in the game board. The agents can move around on the game board and collect the tasks. We can see this problem model as a game that the autonomous agents are players who want to collect the scattered balls (tasks) and then transport them into the only hole as quickly as possible. Thus, we set $r_m = -0.1$. By setting each movement to an empty cell(no task on it) a negative reward, agents are pushed to move to the terminal status as quickly as possible in order to get more positive rewards so that increase its total reward. The agents are cooperative and the shorter execution time for all the agents to transport all the balls into the hole, the higher scores (or rewards the agents can get). The whole environment can be defined as tuple $\langle I, m, N, \{A_i\}, \{\Omega_i\}\rangle$, where $I = \{1, \cdots, n\}$ is a set of finite number of agents, m is the number of tasks, N is the edge size of the environment, A_i is the action set of agent i, $\{A_i\}$ denotes all the agents' action set in the environment, and Ω_i is the observation space of agent i.

In our model, all agents are independent self-interested and autonomous with their own deep Q-networks, so we cannot consider communications for joint actions to reduce unnecessary complexity. The cooperative behavior is represented by reward function, that is, when an agent picks up a task, it gets reward $r_p = 1$. Because the grid world environment is a discrete reinforcement learning model, so we introduce discrete time $t = 0, 1, 2, \ldots$ for

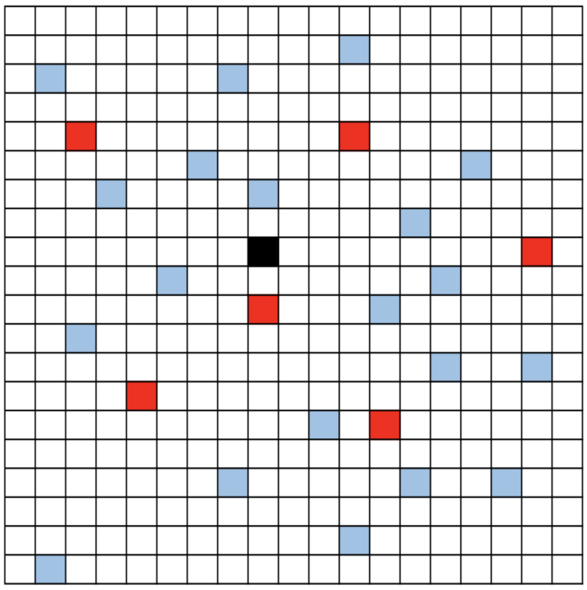

Fig. 1 Example of experiment environment.

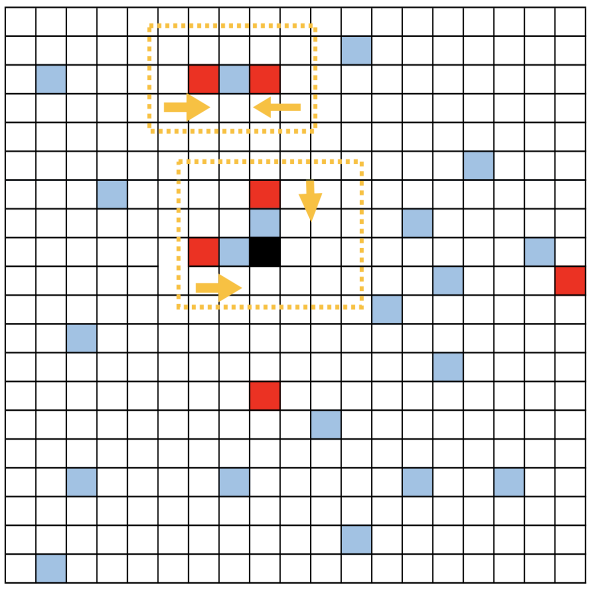

Fig. 2 Illustration of system's conflict.

discrete execution steps.

Action : Agents possible actions are *UP, DOWN, LEFT, RIGHT*. We also define the action to collect a ball as when an agent with no balls in hand (or empty agent) arrives at the cell in which a ball is located. Similarly, we define the action to push a ball into the hole as when an agent who is carrying a ball in hand (a non-empty agent) arrives at the holes cell. The corresponding rewards of *push into* and *pick up* are $r_h = +5$ and $r_p = +1$ respectively.

We assume that each agent has only a limited local view around itself, i.e., it can only observe the contents in the surrounding V(view scope, radius) cells around it. Besides the local view scope, we assume that each agent knows the number of uncollected tasks on the game board. In our problem model, each agent has its own DQN network whose structure is identical to others. The observation of agent i at time step t is denoted by: $o_{i,t} \in \Omega_i$

Policy : State : Initially ($t = 0$), agents $i \in I$ and tasks ψ_u, $(1 \leq u \leq m)$ are scattered in the environment. Tasks are in fixed positions once they are generated from initially in one episode while agents can consecutively move around in the environment. The hole is in the fixed position throughout all episodes. In our experiment, the observation of each agent is the input to Q-network, and the goal of all the agents is to collect all the scattered tasks and push them all into the hole. Therefore the terminal state is when the hole has m tasks in it. From the start of each game episode, all agents take the following steps simultaneously.

(1) At time t, agent $\forall i \in I$ decides action $a_{i,t}$ based on the their *policy*, so $a_{i,t} = \pi_i(o_{i,t}) \in A_i$,where $o_{i,t} \in \Omega_i$

(2) When i moves to a cell where task ψ_u exists, i holds ψ_u and ψ_u is deleted from the grid. Then, i receives reward $r_{i,t} = r_p$.

(3) Agents holding tasks in hand (non-empty agents) pass by the hole, and if no other agents are using the hole, it can push all its tasks into the hole, and get corresponding reward $r_{i,t} = 5$.

(4) If $t \geq T$, or *when the hole has m tasks* then an episode of the game ends; otherwise, $t = t + 1$ and go back to Step (1), where T is a positive integer.

(5) After one episode ends, the environment is initialized and then another episode will start from Step (1).

While executing their tasks, agents would get a negative reward to make them to move as quickly as possible. Each agent has its own reward, and We use total rewards to evaluate the performance and efficiency of the whole multi-agent system.

There are two main conflicts in our model. And we set a *severe punishment* when conflicts happen, which is by setting a negative value whose absolute value is very large.

(1) Conflicts over task:
This type of conflicts could happen when two or more agents try to pick up the same task at the simultaneously as shown in Fig. 2. The upper scenario shows that two agents are trying to pick up the same task between them .

(2) Conflicts over hole:
This conflict occurs when two or more agents try to push their task inside the hole simultaneously, however, the hole can only be accessed by one agent at each time as shown in Fig. 2.

4. Learning Methods

4.1 View representation

As shown in Fig.3, the agent at the center of the yellow pane can only see the local cell's state inside the yellow pane. So it's policy is only base on what it can perceive from the local view.

4.2 Experience Replay

In this experiment, we use DDQN with experience replay in order to avoid overfitting. Agent i stores the experienced data $c_{i,t} = (s_{i,t}, a_{i,t}, r_{i,t}, s_{i,t+1})$ into its own memory $D_{i,t-1} = \{c_{i,t-d}, \cdots, c_{i,t-1}\}$, where $d > 0$ is memory capacity, at t steps. By using experience replay, agents can efficiently use previous experience by learning with it multiple times. Gaining real-world experience is usually costly, agents can get full use of historical experience data and get good learning results. Secondly, by using experience replay, we

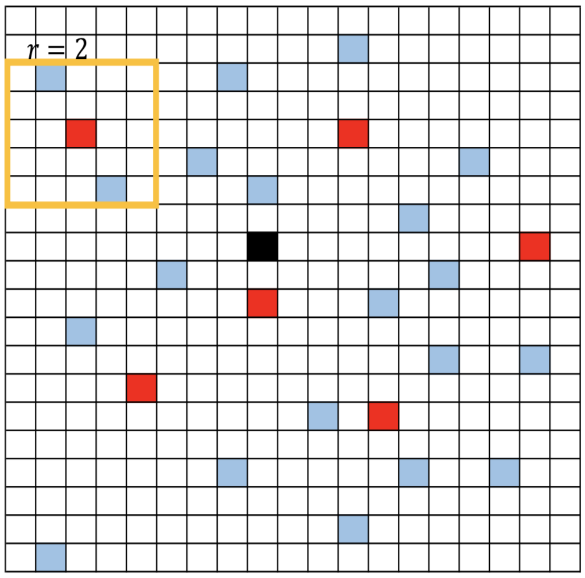

Fig. 3 Illustration of agent's local view.

Table 1 Environment

parameter	value
grid edge length L	20
agent number N	6
pick up a task r_p	1
push a task inside the hole r_h	5
move to an empty cell r_m	-0.1
conflicts r_c	-5
steps per episode T	500
episode E	5000
numbers of each experiment	4

Table 2 Network Structure.

Layers	Inputs	Filter	Kernel size	stride	activation	Output
Convolutional	20 x 20 x 3	16	2 x 2	2 x 2	ReLU	10 x 10 x 16
Convolutional	10 x 10 x 16	32	2 x 2	1 x 1	ReLU	10 x 10 x 32
Convolutional	10 x 10 x 16	32	2 x 2	1 x 1	ReLU	10 x 10 x 32
FCN	10 x 10 x 32				Linear	128
FCN	128				ReLU	4

will have better convergence behaviour when training a function approximation.

4.3 ϵ decay

In reinforcement learning, ϵ is used when we are selecting specific actions base on the Q values we already have. Balancing the ratio of exploration/exploitation is a great challenge in reinforcement learning (RL). Setting ϵ artificially has a great bias on learning time and the quality of learned policies [11]. Merely exploiting uncertain environment knowledge prevents from maximizing the long-term reward because selected actions may not be optimal. There are many ways to decay ϵ like linear way of decaying. In this paper, we adopt ϵ exponential decay to optimize learning performance and try to make a balance between exploit and explore.

5. Experiment and Discussion

5.1 Experimental Setting

Our network structure is shown in Tables 1, 2 and 3.

The observation space for each agent can be abstracted as a

Table 3 Agent parameter.

parameter	value
steps update T_{train}	8
view scope V	2,4
ε decay rate γ_ε	0.9995
Discount rate γ_Q	0.95
RMSprop learning rateη	0.0002
replay memory size	2000
mini batch size	128
θ_t^- update step T_{copy}	100

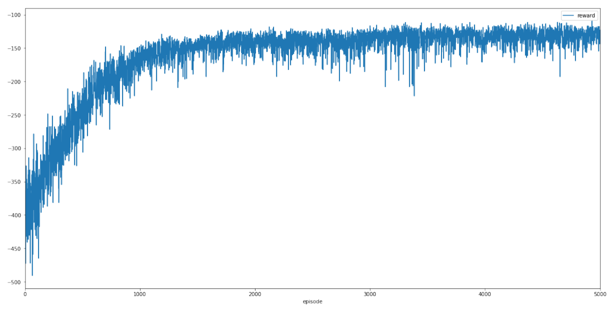

Fig. 4 Total rewards per episode, r=2

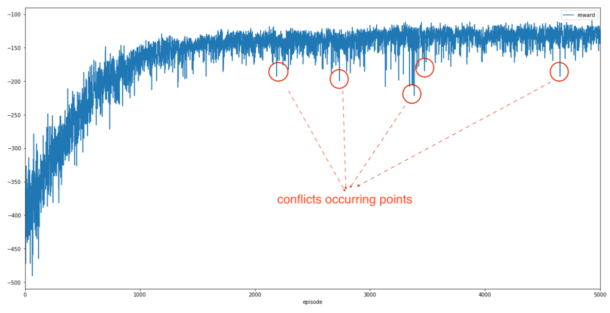

Fig. 5 Conflicts occurrence.

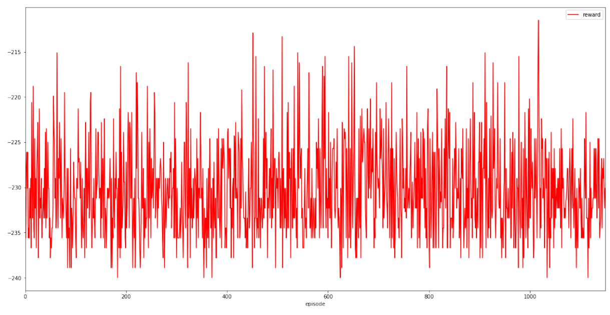

Fig. 6 Total reward per episode when V = 4.

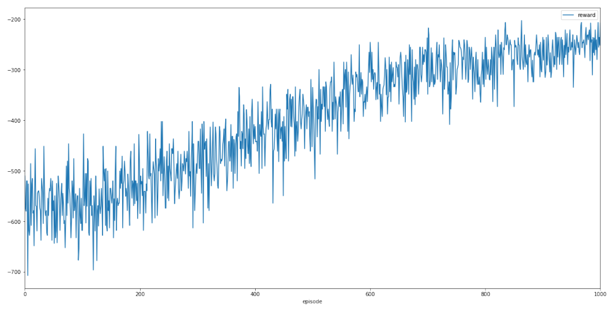

Fig. 7 Total reward per episode when V = 2.

tuple with shape $(height, width, 3)$, in each cell, with or without *agent, task, empty status* can use binary values to represent.

Considering all the agent, then the global observation is $(n, height, width, 3)$, where n is agent's number.

5.2 Results

We conducted two experiments in regard to agents' local view scope with $V = 2$ and $V = 4$, and got relation graphs of average total reward and episode. Fig. 4 shows whole system's performance when agents with view scope $V = 2$. From this figure, we can see that in the first 1000 episodes, total reward grows explicitly, and grows slowly after episode 1200, which means that it has reached its potential highest reward under the specific parameter setting.

Fig. 6 shows whole system's performance when agents with view scope $V = 4$. under 1000 episodes in contrast to Fig. 7 with agent's view scope $V = 2$ under 1000 episodes. With larger view scope $V = 4$, there is no explicit learning growth as we can see from Fig. 6 that the total reward curve fluctuates all the time. While in contrast, under the setting of $V = 2$, agents can have obvious growth of total reward by learning.

5.3 Discussion

First, we can see from Fig. 4, this MAS can learn to optimize the whole system's total reward, in other words, cooperative structure has formed among these agents in an acceptable short time period. With cooperative and coordinate behavior, agents can share to use the scarce resource properly to increase total reward.

On the other hand, according to Fig. 5, there are some low points in the graph occurring periodically, which means that there are considerable amounts of conflicts happening in this episode. But every time when the whole episode is ruined by conflicts, in later episodes, agents can learn to avoid such cases to happen again, thus total reward rises gradually in later episodes.

By comparing Fig. 6 and Fig. 7, we can see that when view scope expands, agents' learning performance is not as good as when view scope is smaller. In the first 1000 episode, agents with $V = 4$ don't show any learning progress, which means that more useful policy should be adopted. Such difference is mainly resulted from the fact that when agents' view scope is larger, their observation space grows exponentially, which need more time to converge and a more powerful computing machine to deal with such large input matrices. In theory, the curve will have a similar shape as when $V = 2$, however, such time delay is not practical in real use, so we should find better solution to cope with larger view scope input.

6. Conclusion

We investigated if agents in multi-agent system can work cooperatively by sharing scarce resource to achieve a common goal using deep reinforcement learning method (MA-DQN). We conducted two experiments to compare the performance with different size of view scopes as input to each agent's DQN, our experiment result indicates that there is a significant difference between smaller view scope and larger view scope, when the view scope is smaller, which in our experiment $V = 2$, agents can have a good learning performance and total reward can converge quickly. However, under the setting of larger scope, which in our experiment $V = 4$, agents' learn speed is pretty low, and that is for our future work to explore why and how to improve it so that it can work well as the smaller view scope.

We would like to extend this topic in many aspects for our future research. For example, even though under the view scope $V = 2$, agents have a good learning performance, however, whether it is the optimal result in solving conflict problems in regard to scarce common resource in MAS or not, more experiments by using different methods are needed in our future study. So we plan to put forward a generalized multi-agent cooperative structure regardless of view scope size by changing agent's learning policy.

When we were experimenting even on GPU, waiting for results after all episodes of one experiment to finish is quite a long time, so we wonder to find effective ways to reduce training time as another aspect of our future work. As an extension, we also want to discuss deadlock problem. Deadlock is common to see when conflicts happen frequently in a busy environment, there are many ways to solve deadlocks in other system like operating system. How to apply the approaches for those systems to multi-agent system which is distributed and large-scale or create new solutions for deadlocks in MAS, there are still a lot more to be explored in our future work.

References

[1] Mihaela Oprea: Applications of Multi-Agent Systems, *Reis R. (eds) Information Technology. IFIP International Federation for Information Processing*, Vol.157, pp.239–270 (2004).

[2] Wolfgang Honig, Dadoun, Scott Kiesel, Andrew Tinka, Joseph W. Durham and Nora Ayanian: Conflict-Based Search with Optimal Task Assignment, *In Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems, Stockholm, Sweden*, pp.757–765 (2018).

[3] Jingjin Yu, Steven M. LaValle : Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics, *Journal IEEE Transactions on Robotics*, Vol.32, No.5, pp.1163–1177 (2016).

[4] Jakob Foerster, Richard Y. Chen,Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel,Igor Mordatch : Learning with Opponent-Learning Awareness, *Proceeding AAMAS '18 Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp.122–130 (2018).

[5] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, Shimon Whiteson : Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning, *arXiv preprint arXiv:1702.08887*, (2017).

[6] Yuki Miyashita, Toshiharu Sugawara: Emergence of Multi-Agent Cooperation By Deep Q-Network for Decentralized Search Problem, *Journal of Information Processing (JAWS 2018)*, Vol.59, No.1, pp.1–7 (2018).

[7] Shixiang Gu, Ethan Holly, Timothy Lillicrap, Sergey Levine : Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics, *arXiv:1610.00633*, (2016).

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller : Playing Atari with Deep Reinforcement Learning, *arXiv:1312.5602*, (2013).

[9] Maximilian Httenrauch, Adrian, Gerhard Neumann : Guided Deep Reinforcement Learning for Swarm Systems, *arXiv:1709.06011*, (2017).

[10] Elhadji Amadou Oury Diallo,Toshiharu Sugawara : Learning Strategic Group Formation for Coordinated Behavior in Adversarial Multi-Agent with Double DQN, *PRIMA 2018: Principles and Practice of Multi-Agent Systems. PRIMA 2018*, Vol.11224, pp.458–466 (2018).

[11] Michel Tokic : Adaptive ε-greedy Exploration in Reinforcement Learning Based on Value Differences, *KI 2010: Advances in Artificial Intelligence. KI 2010. Lecture Notes in Computer Science*, Vol.6359,

pp.203–210 (2010).

[12] Zhang-Wei Hong, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, Chun-Yi Lee : A Deep Policy Inference Q-Network for Multi-Agent Systems, *arXiv:1712.07893 [cs.AI]*, (2017).

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg Demis Hassabis : Human-level control through deep reinforcement learning, *Nature*, Vol.518, pp.529–533 (2015).

[14] MC Xie and A Tachibana : Cooperative behavior acquisition for multi-agent systems by Q-learning, *Foundations of Computational Intelligence,FOCI 2007. IEEE Symposium on. IEEE*, pp.424–428 (2007).

[15] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel : Optimal Multirobot Path Planning on Graphs: Sim-to-real transfer of robotic control with dynamics randomization, *International Conference on Robotics and Automation (ICRA). IEEE*, pp.1–8 (2018).

[16] Toru Ishida : Real-Time Bidirectional Search: Coordinated Problem Solving in Uncertain Situations., *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp.617–628 (1996).