Decentralized Exploration of a Structured Environment Based on Multi-agent Deep Reinforcement Learning

Dingjie He, Dawei Feng, Hongda Jia, Hui Liu
College of Computer
National University of Defense Technology
Changsha, China
Email: dingjiehe@gmail.com

***Abstract*—Multi-robot environment exploration is one of the widely discussed topics in the field of robotics. It is the foundation for many real-world robotic applications. Many decentralized methods (that is, without a centralized controller) have been proposed in the past decades. Most of them focus on improving collaboration efficiency by utilizing low-level heuristic information, such as distances to obstacles and robot positions. In contrast, although a human being can make decisions on a similar task, he/she exploits high-level knowledge, such as the building's common structure pattern. This paper proposes a novel distributed multi-robot exploration algorithm based on deep reinforcement learning (DME-DRL) for structured environments that enables robots to make decisions on the basis of this high-level knowledge. DME-DRL is a distributed algorithm that uses deep neural networks to extract the structural pattern of the environment, and it can work in scenarios with or without communication. The experimental results showed that this approach can decrease the travel distance by approximately 10.84% on average, compared with those of traditional heuristic methods and can significantly reduce the communication cost in the exploration process.**

***Keywords*-multi-robot system; distributed environment exploration; deep reinforcement learning;**

I. Introduction

Exploring an unknown area efficiently with a group of robots is a fundamental problem in mobile robotics[1]. It lays the foundation for many real-world multi-robot applications such as unmanned aerial vehicle reconnaissance, robot-assisted disaster relief, and planet exploration. Depending on whether there is a central coordinator, exploration algorithms can be divided into two classes: centralized and decentralized. In the latter class, every robot controller makes decisions on the basis of its local observations and communication with neighboring robots. By excluding a central control unit, this type of algorithm is more robust, has the potential to promote exploration efficiency by enabling the robots to behave asynchronously, and fits large-scale scenarios better[2]. However, it is challenging to perform high-quality coordination behaviors in the absence of a coordinator, especially in a large, stochastic, and unknown environment.

A number of decentralized algorithms for multi-robot environment exploration have been proposed. Most of them focus on improving collaboration efficiency by utilizing the raw information observed by the robots. For example, the main branch of these algorithms is frontier-based exploration, where "frontier" means the boundary between known and unknown areas[3]. A robot, on observing a frontier, can utilize programming algorithms, such as greedy algorithm, pairing, or brute-force searching, to achieve fast exploration and minimize the overlaps in the explored regions. Such methods generally require an evaluation function to estimate the selection of frontiers and to provide heuristic information. However, this evaluation function can only utilize simple information, such as distances, positions, and unexplored regions. In contrast, a human being benefits from a previous similar task and he/she can also take advantage of abstract knowledge, such as the structural pattern of a building. This high-level knowledge helps in forming a general understanding of environments and in designing an effective exploration policy.

Inspired by these, and by combining the success of deep neural networks in pattern recognition[4], we propose a novel decentralized approach based on multi-agent deep reinforcement learning (MADRL), which contains several agents in the task, to guide multiple robots in exploring unknown environments. In our method, deep neural networks are used to extract high-level knowledge, such as the structural patterns of the environment. This approach adopts a variant of the multi-agent deep deterministic policy gradient (MADDPG) algorithm [5] to improve the coordination ability in decentralized scenes. This algorithm is a centralized training and decentralized execution (CTDE) algorithm in which robots obtain all information in the training phase, while them make decisions based on their own local observations and the messages from neighboring robots in the execution phase. To reduce the transmitted data, we introduce layers of communication with a recurrent network to use partial information sequences. Robots can estimate the change in environment state with limited sequential information. We examined this algorithm in several simulation experiments, and the obtained results showed the excellent performance of the algorithm. Compared with traditional heuristic methods, our approach can reduce the travel distance by approximately 10.84% on average, which means it can reduce the energy consumption of the multi-robot system in the whole exploration process. As for the communication, although its performance is approximately 10.78% lower than that of the complete communication method, our approach reduced the transmitted data by approximately 76.49%.

The rest of the paper is structured as follows. Section 2 introduces the related works on robot environment exploration. Section 3 introduces our model and algorithm for the multi-robot environment exploration problem. Section 4 presents the experiment results. Finally, Section 5 concludes the paper.

II. Related Works

The multi-robot exploration problem was proposed in the early 1990s. As a foundation to mobile robotics, multi-robot exploration requires the coordination of robots and enabling of the exploration process efficient[6]. These need to reduce the overlaps in explored areas to minimize the distance (or time) cost. For this problem, a naive idea is to explore the environment randomly, or choose a random point as each robot's target in path planning. Lpez-Snchez et al. conducted an early work, where they made the robots choose from one of four behaviors with different probabilities[7], [8]. This random method is quite simple, but a lack of coordination mechanism can result in a significant overlap in the explored regions and in lower efficiency. Approaches based on heuristic information have been proposed to improve the performance of multi-robot systems. The main branch adopts the frontier, i.e., the boundary of visited regions, to guide the exploration. In Yamauchi's seminal paper[3], the author firstly adopted a strategy based on the distance to the nearest frontier. Based on this seminal paper, a number of market-based approaches have been proposed to improve the coordination of the robots. A representative work is Zlot et al.'s exploration strategy[9]. In their approach, every robot needs to find the regions with the largest unknown area, and these regions are allocated to the robots by using a bidding protocol. Sheng et al.[2] combined a variety of heuristic information and considered the communication problem in coordination. Recently, Corah and Michael[10] used raw data from sensors as heuristic information and a Monte-Carlo tree to speed up the exploration in real-world settings. This approach can be extended to $4 \sim 32$ robots; however, it may lead to suboptimality. Zhou et al.[11], [12] use the genetic algorithm to solve the coverage problem in the exploration but mainly direct at known environments. All these methods are based on heuristic information and mainly focus on utilizing low-level information from sensor information, such as distances and positions. In contrast, a human being can use high-level knowledge, such as structural patterns, to accelerate the exploration. However, it is difficult to utilize this knowledge in the above methods.

With the development of machine learning, especially deep learning and reinforcement learning, researchers have tried to utilize deep reinforcement learning (DRL) to solve problems in multi-robot systems. For example, Long et al. [13] adapted the proximal policy optimization algorithm, a reinforcement learning algorithm, to avoid collision in decentralized multi-robot systems. Sartoretti et al. used a variant of the asynchronous advantage actor-critic (A3C) algorithm to solve the multi-robot construction problem[14]. Everett et al.[15] utilized a deep neural network with long short-term memory and a reinforcement learning algorithm to help robots navigate in a dynamic environment. To solve the exploration problem in multi-robot systems, researchers have tried to introduce deep reinforcement learning. Bai et al. [16] used a supervised learning algorithm to imitate human behavior under different observations. Tai et al. [17] used environment information as input to the neural network and trained agents using the asynchronous deep deterministic policy gradient algorithm. To extract the structural patterns of the environment and use them, Zhu et al. [18] introduced convolution networks to encode the structure information and used the A3C algorithm in the training phase. Most of these works aim at a single-robot scenario or known areas, but lack solutions to multi-robot scenarios in an unknown environment. Using the method proposed by Zhu et al. [18], Luo et al. [19] extended it to multi-robot exploration. They transformed the environment into a graph and used neural networks to extract information from this graph. This method reduces the average distance travelled, but requires that the environment be known.

In summary, traditional heuristic methods can be efficient and straightforward in specific environments, but they cannot use high-level knowledge. Inspired by the application of deep reinforcement learning on other fields of robotics, many methods have been proposed to solve the exploration problem. They are mainly directed at single-robot scenarios, or known areas even with multiple robots. Therefore, studies on multi-robot exploration in unknown environments are still relatively lacking. To our best knowledge, our work is the first to guide multiple robots to explore unknown environments directly by using structural patterns and history of observations.

III. Distributed Multi-robot Exploration Algorithm Based on MADRL

In this section, we formulate the multi-robot environment exploration problem and build a model for it. Subsequently, we propose our algorithm, the distributed multi-robot exploration algorithm based on deep reinforcement learning (DME-DRL). On the one hand, we introduce the deep network to utilize the high-dimensional information in the DME-DRL; on the other hand, we change the structure of the replay buffer to store a period of timesteps' trajectories as a record, which helps to overcome Dec-POMDP in the exploration task. The details will be discussed in the following parts.

A. Problem Formulation

To simplify the problem, we use a grid map[20] to store the real map. The grid map is a way to translate the continuous map into a set of discrete points (cells), and every point represents a specific region. In order to simulate the limitation of real sensors, we assume that the environment can only be **partially observed** by the robots, which means that robots' local observations are only parts of the global state.

In multi-robot exploration, the goal is to find an optimal path with the shortest distance travelled or the minimal time spent to explore the whole environment. In this paper, we try to minimize the distance cost. On the one hand, reducing distance costs can reduce energy consumption; on the other hand, when tasks are evenly allocated, a shorter distance means less time spent. On the grid map, the exploration problem can be described as finding an optimal path P^* consisting of several points to minimize the exploration distance cost. Its formulation is as follows:

$$P^* = \underset{p\in\mathbb{P}}{\arg\min} \sum_{i=1}^{N} \sum_{t_i=1}^{T_i} (g(p_i^{t_i})). \tag{1}$$

Here, $\mathbb{P}$ is the set of all points on the grid map, T_i is the total timesteps of robot i, N is the number of robots, $p_i^{t_i}$ is the point chosen by robot i at time t_i, and $g(p)$ is a function for calculating the distance from the robot's current position to the point p.

B. Model for Multi-robot Explorations

To use reinforcement learning to solve the exploration problem, we need to build a Markov decision process (MDP) model for it. When the observation is only a part of the global state, the reinforcement learning cannot be applied directly because partial observations do not satisfy the Markov property. A recurrent neural network is usually used to store time sequences[21] to deal with the problem in this partially-observed MDP case, and these sequences provide enough information to remedy the lack of state information. Lastly, the floor plan of buildings usually has a structured pattern and convolution networks can extract this pattern from the floor plan. From the above, it can be deduced that deep reinforcement learning provides a way to solve the multi-robot exploration problem.

To apply reinforcement learning on multi-robot exploration, we formulate an MDP model for this problem as follows:

State: S is a global state set of the multi-robot system, including the state of the environment and those of the robots.

Action: A is a set of actions taken by the robots every time. Especially, $A^t = (a_1^t, \cdots, a_N^t)$ and a_i is the action of robot i at time t. In this paper, an action means choosing a direction to move.

Observation: o_i is the observation that robot i can sense and store, and it is a part of the global state. Because of the limitation of communication, different robots usually have different observations. When the robots get close at a certain distance, they can exchange their observations and merge them with their own explored maps.

Reward: When a robot takes action according to its controller, the environment will return a reward r. It is a signal from the environment to the robot. It can guide the robot to complete the task, and, therefore, the reward should be well designed. In this paper, to encourage the robots to explore the environment as quickly as possible, we designed the reward to contain two parts. The first part is the area c explored every time, making the robots explore the whole environment as much as possible. The other part is the distance d the robots move, which minimizes the distance cost in the exploration. Therefore, for robot i, its reward r_i^t at time t can be represented as

$$r_i^t = w_1 c_i^t + w_2 d_i^t, \tag{2}$$

where w_1 and w_2 are weight factors, and c_i^t and d_i^t are the area reward and distance reward of robot i at timestep t. In this paper, we set them as $w_1 = 2e-4, w_2 = -1e-3$

γ: γ is a discount factor in the range $[0, 1]$ of the reward, and it controls the importance of the current and future rewards. In this paper, we set γ to 0.95, so that the algorithm pays more attention to future rewards.

C. DRL-based System Architecture

1) Modules: In this paper, the multi-robot exploration system contains three modules. The **environment** contains a 2D map with obstacles. It also provides a scenario for the robot to move and an interface for interactions. Every **robot** has two neural networks: the *actor* and the *critic* network. The *actor* network generates an exploration policy, and the *critic* network estimates the value of this policy. Subsequently, the robots carry out actions to explore the environment according to the generated policy and obtain a reward signal from the environment. The **replay buffer** is a module for storing the experience in the training phase, and it also provides a data batch to the neural network for updating the exploration policy. All data are saved in the form of the tuple $< o, a, r, o' >$, where o' is the next observation sequence. Especially, the observation contains explored maps and the robots positions.

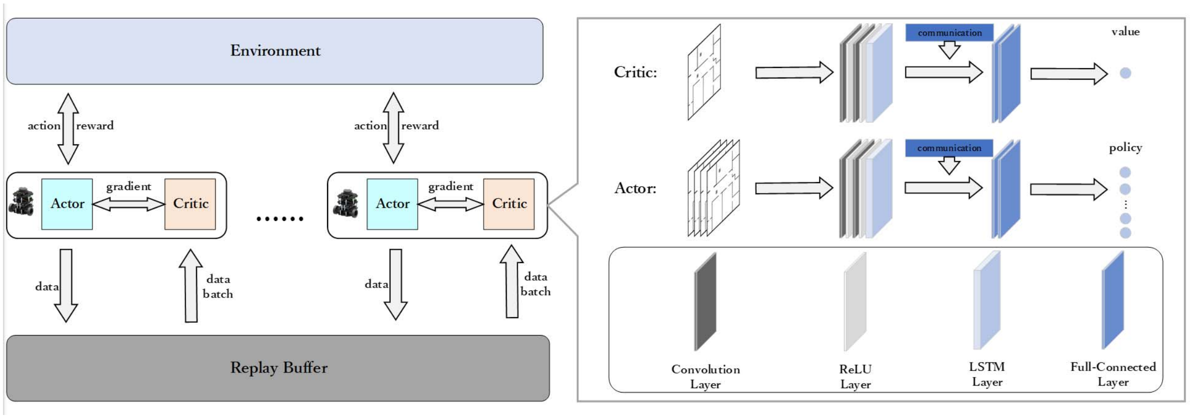

Figure 1. DRL-based System Architecture

2) Neural Networks: The training method used in DME-DRL is a variant of the MADDPG algorithm[5]. This algorithm uses the CTDE architecture. The CTDE architecture trains a *critic* neural network together and provides robots with global information (including the other robots' observations) in an ideal environment such as a laboratory. This network plays the role of a *critic* to guide the *actor* network. The *actor* neural network generates an exploration policy, and the trained *critic* network evaluates the value of this policy. Then, the *actor* uses this value to update the parameters of its network. Especially, the *actor* only takes its observation as an input, which means that every robot can make a distributed decision and does not depend on other robots' information.

D. Distributed Multi-robot Exploration Algorithm Based on Multi-Agent Deep Reinforcement Learning

DME-DRL can work in no-communication scenes by using the CTDE architecture; however, we also propose a communication version that considers real-world settings.

1) Training: For every robot i, it has no information about the environment at first, so its *critic* network $Q_i(o_{1:N}, a_{1:N}|\theta_i^Q)$ with weight θ_i^Q, which is used to estimate the value of actions $a_{1:N}$ when observations are $o_{1:N}$, and *actor* network $\mu_i(o_i|\theta_i^\mu)$ with weight θ_i^μ, which is known as policy and generates actions taken by robots, are all initialized randomly at the beginning; then, we make a copy Q'_i, μ'_i as robot i's target networks, which are really used in the exploration phase.

All robots explore the environment for a certain period to collect enough experiences to store in a replay buffer, and this progress is more like a random exploration because of the untrained μ. Then, the robots start their training process: the *critic* samples a data batch from the replay buffer and calculates the estimated values of every data record j for every robot i:

$$y^j = r_i^j + \gamma Q'_i(o'^j, a'_{1:N})|_{a'_k=\mu'_k(o_k^j)}. \tag{3}$$

Here,r_i^j is the reward got at the current step, and $Q'_i(o'^j, a'_{1:N})$ is a future reward, so the y^j is the real value of robot i's current state. Then, the *critic* tries to update its weight θ_i^Q by minimizing the difference (called the "loss") between the real value and the estimated value $Q_i(o^j, a_{1:N}^j)$):

$$\mathcal{L}(\theta_i^Q) = \frac{1}{N}\sum_j (y^j - Q_i(o^j, a_{1:N}^j))^2. \tag{4}$$

After several iterations, the *critic* network can give a better estimation, and the *actor* tries to maximaze the value from *critic* and update its weight θ_i^μ by using a sampled policy gradient:

$$\nabla_{\theta_i^\mu} J \approx \frac{1}{N}\sum_j \nabla_{\theta_i^\mu}\mu_i(o_i^j)\nabla_{a_i}Q_i(o_{1:N}^j, \sigma_{a_i}^{a_i^j}(a_{1:N}^j)), \tag{5}$$

where $\sigma_y^x(Z)$ is a function that replaces the x in Z with y and $a_i = \mu_i(o_i^j)$. DME-DRL updates all robots' weights at every timestep until the environment is explored or timestep $t = T$ (T is the maximum timestep in an episode). Finally, we perform a soft update on the target network by using τ to avoid huge changes in the target network. This paper shows the details of DME-DRL in Algorithm 1.

2) Exploration: When the training phase concludes, only the *actor* network is needed to guide the exploration process. In this phase, the robots' actors take the current map being explored and the observation sequences as inputs to the neural networks and receive the output as the action command (i.e., a direction of movement). Then, the robots choose the nearest boundary point in this direction. Finally, every robot plans its path by using the A^* algorithm. These operations are repeated until 95% of the environment has been explored.

3) Layered Communication: In the exploration process, robots can always communicate with other robots at a certain distance d. This assumption accelerates the exploration process; however, the process usually consumes more bandwidth for information transmission, especially the whole map. We change robots' communication behaviors and divide the communication range into two layers to reduce the transmitted data, and the robots send different data in different layers. Using the communication method, we propose a **layered communication** model for DME-DRL. In this model, we use D_1 and D_2 ($D_1 < D_2$) to represent the respective radii of two different layers. When the distance is in the range $(0, D_1]$, the robots exchange complete information, and when the distance is in the range $(D_1, D_2]$, only the position information is exchanged. The details are shown in Algorithm 2.

Algorithm 1: Distributed Multi-robot Exploration Algorithm Based on Deep Reinforcement Learning

```
1  Initialize networks $Q, \mu$ and target networks $Q', \mu'$ ;
2  for episode=1 to $M$ do
3      Initialize the environment, robots $\mathbb{R}$, and observation sequences $o$ ;
4      repeat
5          Initialize a Gauss random process $\mathcal{N}$ ;
6          Choose an action $a_i$ by $\mu_{\theta_i}(o_i) + \mathcal{N}_t$ for every robot $i$;
7          Choose the nearest frontier $g_i$ in $a_i$'s direction ;
8          Plan a path $P$ to $g_i$ by using the A* algorithm for every robot $i$;
9          Move along $P$ to explore the environment for all robots ;
10         Obtain new observations $o'$ and rewards $r$;
11         Communication ;
12         Update observations $o'_i$ and rewards $r$ for every robot $i$ ;
13         Store the experience $(o, a, r, o')$ in replay buffer $\mathcal{D}$;
14         Update the observation sequences $o \leftarrow o'$;
15         for robot $i$ in $\mathbb{R}$ do
16             Sample a batch of experiences $(o^j, a^j, r^j, o'^j)$ from $\mathcal{D}$ ;
17             Update the critic network's weight $\theta_i^Q$ by minimizing the loss in equation (4) ;
18             Update the actor network's weight $\theta_i^\mu$ by using the sampled policy gradient in equation (5) ;
19         end
20         Update the target network weights $\theta'_i$ for every robot $i$ by $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$;
21     until $95\%$ or more of the environment $m$ is explored or the timestep $t = T$;
22 end
```

Algorithm 2: Layered Communication

```
1  for robot $i$ in $\mathbb{R}$ do
2      Read distance data $d$ with data record $j$ from the sensors;
3      if $d$ in $(0, D_1]$ then
4          Exchange the explored maps $m_i, m_j$ between $i$ and $j$;
5      end
6      if $d$ in $(D_1, D_2]$ then
7          Exchange the position data $p_i, p_j$ between $i$ and $j$;
8      end
9  end
10 Merge the explored maps and positions;
11 Recalculate observations $o'$ and rewards $r$;
```

IV. Experiments and Results

To evaluate the performance of the proposed algorithm, we conducted relevant experiments in this study. We used an improved version[1] of the simulator from [22]. The simulator contains a scalable robot team and hundreds of 2D maps generated from the SUNCG dataset[23]. The robot in this simulator has a 360° field of view and can move to any free (without obstacles) space on the map. Moreover, the robots can communicate with other robots in a limited range. Some hyperparameters in this paper are given in Table I.

Table I
Hyperparameter Settings

symbol	meaning	value
γ	discount factor	0.95
w_1	weight of area reward	$2e-4$
w_2	weight of distance reward	$-1e-3$
M	maximum episodes	6000
T	timestep limit per episode	50
τ	weight of soft update	0.01

A. Distance Cost of DME-DRL

As stated in Section III.*A.*, our major goal is to minimize the distance cost. Therefore, to verify the performance of DMR-DRL, we performed an experiment to compare its distance cost with those of the following three other methods:

Random Frontier (RF): Every robot chooses a point on the boundary of the explored area randomly and moves there.

Nearest Frontier (NF): The robots choose the nearest frontier point as a target.

[1]The code can be get from https://github.com/hedingjie/DME-DRL.git

MADDPG: The robots use the raw distance data as the inputs (without explored maps and observation sequences) and train the neural networks by using the raw MADDPG algorithm.

These methods are comparable because they have two common properties: a) they have no central controllers, which means that they are distributed, and b) they all communicate in the entire exploration process. We tested these methods on 10 different maps, 10 times, and the robots had different initial positions every time. The results are shown in Figure 2.

From Figure 2, we can observe that DME-DRL has the shortest distance travelled on nine maps. Compared with the second one, DME-DRL has a significant improvement, with a 10.84% reduction in distance travelled. The different performance between DME-DRL and the raw MADDPG algorithm (without structural information and time sequences) shows that structural information does help the robots accelerate the exploration process.

B. Time Cost of DME-DRL

Additionally, the time cost is an important indicator to measure the speed of exploration, so we performed an experiment to evaluate the performance of DME-DRL in terms of the time cost: In the experiment, we tested the three algorithms on 20 maps, 10 times, and the time spent on completing the exploration (95% of areas explored) is shown in Table II.

Table II
TIME SPENT ON COMPLETING THE EXPLORATION

Map ID	**Methods**			
	DME-DRL	***MADDPG***	***RF***	***NF***
0	1613.6	1746.7	2065.4	**1433.2**
1	988.0	886.5	997.1	**869.0**
2	**804.9**	838.7	874.9	928.2
3	**1132.6**	1239.0	1201.6	1135.6
4	899.4	**851.2**	921.4	1054.2
5	**958.7**	1106.7	1197.4	1166.9
6	1393.6	**1299.8**	1586.0	1357.6
7	**723.1**	891.6	865.0	752.0
8	**806.0**	888.5	923.7	1009.1
9	**992.1**	1069.9	1140.7	1118.7
10	712.3	794.7	986.4	**672.8**
11	922.7	**848.9**	1208.6	1018.3
12	**778.2**	904.5	881.2	792.5
13	**1095.7**	1269.1	1319.8	1336.0
14	690.1	740.9	953.3	**645.1**
15	**772.4**	786.2	980.7	850.5
16	818.1	**682.0**	993.0	850.0
17	897.5	1010.2	1293.5	**796.5**
18	917.5	821.0	1069.8	**792.9**
19	920.0	1064.5	1114.9	**872.3**
Average	**941.8**	987.0	1128.7	972.6

*The **bold data** are the best result.

From Table II, we can observe that DME-DRL also shows good performance in reducing the time cost. In the 20 different maps, it spent the least time on nine maps. Moreover, it saved approximately 3.27% of time compared with that of the sub-optimal one, i.e., the NF method. The improvement in time cost is not vast because it is not the primary target of optimization, and it even is not contained in the reward. The DME-DRL can obtain better results if we design the reward function carefully.

The results of the two aforementioned experiments showed that DME-DRL did increase the efficiency of the exploration and decrease the distance cost; however, there were several cases (such as map 5) where it did not obtain the best performance. There are three reasons for its sub-optimal performance in these cases. First, the buildings' structures are uncomplicated, which means that the structure information is sparse and difficult to extract. Second, a wrong initial position may also lead to bad performance. The reason is that, when robots are set very close to each other at the beginning, they may have similar observations, which will lead to the same actions and overlaps in the explored areas. Lastly, the reward function prefers to encourage robots to decrease the distance travelled, which may ignore the performance in time cost. However, this problem can be solved by a carefully designed reward function.

C. Communication Costs

To evaluate the performance of DME-DRL on different communication methods, we compared it with those of the following three communication models on 10 other maps:

Layered Communication (LC): LC divides the communication range into two different layers. When the distance between the robots is in the range $(0, D_1]$, the robots exchange complete information. However they need to spend more bandwidth on data transmission. In the range $(D_1, D_2]$, the robots only exchange partial information, such as positions. This design is for the case where the signal degrades with the distance.

No Communication (NC): In this case, the robots do not have any communication, and there is no information exchanged in the process.

Complete Communication (CC): The robots exchange all information (such as the huge map information) when they are within communication distance with other robots. The size of the transmitted data was calculated in the process, and the results are shown in Table III.

From Table III, we can observe that the CC method always shows the best performance on the distance cost. The reason is that this method maintains the robot data synchronized, which always leads to good coordination. The synchronization of the robot information leads to effective coordination. Compared with this method, the layered communication saved approximately 77.73% data transmission, although it travelled approximately 10.78% more. The result of this experiment showed that DME-DRL can coordinate robots even if only a small amount data are exchanged, and that it is suitable for a weak network case.

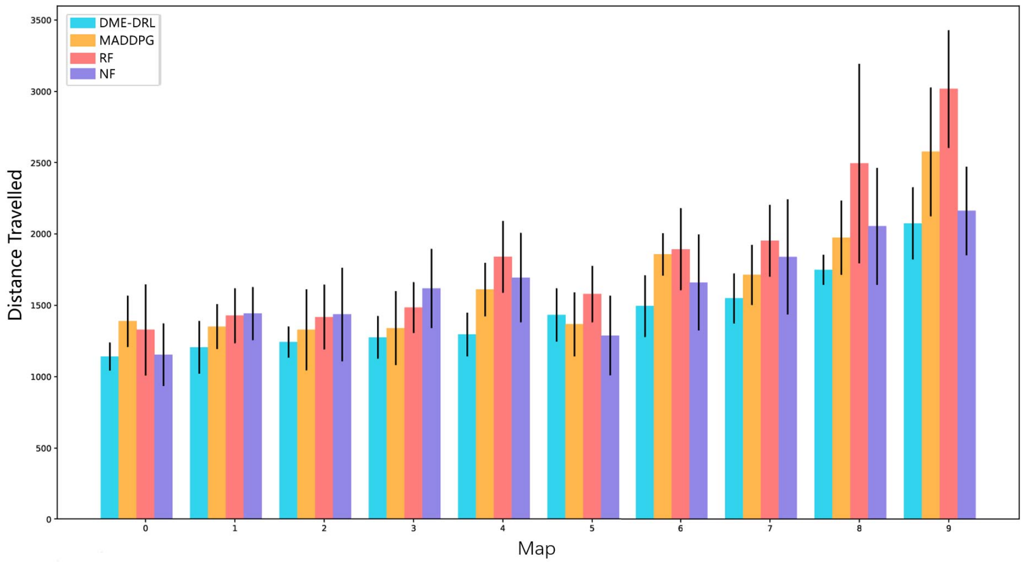

Figure 2. Comparison of Distance Travelled among DME-DRL, MADDPG, RF, and NF.

Table III
COMPARISON OF THE DATA TRANSMITTED BETWEEN THE DIFFERENT COMMUNICATION MODELS

Map ID	**Distance Travelled**			**Data Transmitted**	
	NC	***CC***	***LC***	***CC***	***LC***
0	1633.0	1296.3	1579.9	0.256	0.046
1	1510.0	1265.5	1269.1	0.298	0.065
2	1015.1	775.3	1142.8	0.202	0.057
3	1286.8	1016.3	1106.3	0.141	0.061
4	1368.3	1180.6	1230.4	0.221	0.050
5	1179.7	1002.6	775.4	0.206	0.034
6	1895.7	1277.9	1442.8	0.206	0.061
7	1432.5	1141.9	1383.4	0.191	0.027
8	1199.8	1064.0	1188.9	0.202	0.034
9	1275.8	1224.5	1337.9	0.187	0.034
Average	1379.7	1124.5	1245.7	0.211	0.047

*The unit of the transmitted data is megabytes **MB**.

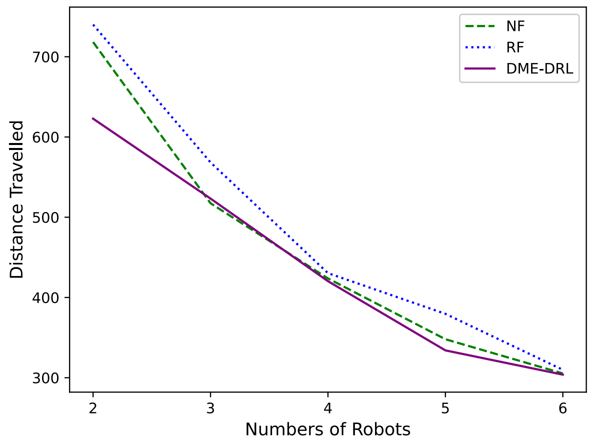

Figure 3. Average Distance Travelled with Different Numbers of Robots.

D. Scalability

Finally, we tested the performance of DME-DRL using different numbers of robots, and the results are shown in Figure 3. Compared with the nearest frontier and random frontier methods, DME-DRL usually has the shortest average distance travelled, which further decreased with the increase in the number of robots. This result indicates that DME-DRL has good scalability. Thus it is suitable for a small-scale multi-robot system. We notice that, on the one hand, the system with three robots has a sub-optimal result, which may be caused by bad initial positions; on the other hand, the distance travelled using DME-DRL converges with the increase in the number of robots. When the number of robots increased to 6, the three methods showed similar performance owing to the fixed-sized maps.

V. CONCLUSION

This paper proposed an algorithm known as DME-DRL, which utilizes structural information and time sequences to accelerate the exploration of multiple robots in an environment. It obtained better results on the time cost, distance cost, and data transmitted than those of the traditional heuristic methods. We also found that several factors may cause its performance to degrade, such as sparse structural information and wrong initial positions. A solution to this problem is having well-designed reward functions and good initial positions, and we leave this as our future work.

VI. Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (no. 61751208) and by the Advanced Research Program (no. 41412050202).

References

[1] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 476–481.

[2] W. Sheng, Q. Yang, J. Tan, and N. Xi, "Distributed multi-robot coordination in area exploration," *Robotics and Autonomous Systems*, vol. 54, no. 12, pp. 945–955, 2006.

[3] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE International Symposium on Computational Intelligence in Robotics & Automation, Cira*, 1997.

[4] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep machine learning - a new frontier in artificial intelligence research [research frontier]," *Computational Intelligence Magazine IEEE*, vol. 5, no. 4, pp. 13–18, 2010.

[5] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.

[6] J. Banfi, A. Q. Li, I. Rekleitis, F. Amigoni, and N. Basilico, "Strategies for coordinated multirobot exploration with recurrent connectivity constraints," *Autonomous Robots*, vol. 42, no. 4, pp. 875–894, 2018.

[7] R. López de Màntaras, J. Amat, F. Esteva, M. López, and C. Sierra, "Generation of unknown environment maps by cooperative low-cost robots," in *Proceedings of the first international conference on autonomous agents*, 1997, pp. 164–169.

[8] M. Lpez-Snchez, F. Esteva, R. L. D. Mntaras, C. Sierra, and J. Amat, "Map generation by cooperative low-cost robots in structured unknown environments," *Autonomous Robots*, vol. 5, no. 1, pp. 53–61, 1998.

[9] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 3. IEEE, 2002, pp. 3016–3023.

[10] M. Corah and N. Michael, "Efficient online multi-robot exploration via distributed sequential greedy assignment." in *Robotics: Science and Systems*, vol. 13, 2017.

[11] X. Zhou, H. Wang, and B. Ding, "How many robots are enough: A multi-objective genetic algorithm for the single-objective time-limited complete coverage problem," pp. 2380–2387, 2018.

[12] X. Zhou, H. Wang, B. Ding, W. Peng, and R. Wang, "Multi-objective evolutionary computation for topology coverage assessment problem," *Knowledge Based Systems*, vol. 177, 2019.

[13] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6252–6259.

[14] G. Sartoretti, Y. Wu, W. Paivine, T. S. Kumar, S. Koenig, and H. Choset, "Distributed reinforcement learning for multi-robot decentralized collective construction," in *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 35–49.

[15] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[16] S. Bai, F. Chen, and B. Englot, "Toward autonomous mapping and exploration for mobile robots through deep supervised learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2379–2384.

[17] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[18] D. Zhu, T. Li, D. Ho, C. Wang, and M. Q.-H. Meng, "Deep reinforcement learning supervised autonomous exploration in office environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7548–7555.

[19] T. Luo, B. Subagdja, D. Wang, and A.-H. Tan, "Multi-agent collaborative exploration through graph-based deep reinforcement learning," in *2019 IEEE International Conference on Agents (ICA)*. IEEE, 2019, pp. 2–7.

[20] C. Luo, J. Gao, X. Li, H. Mo, and Q. Jiang, "Sensor-based autonomous robot navigation under unknown environments with grid map representation," in *Swarm Intelligence*, 2014.

[21] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *Computer Science*, 2015.

[22] T. Li, D. Ho, C. Li, D. Zhu, C. Wang, and M. Q. H. Meng, "Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots." *arXiv: Robotics*, 2019.

[23] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.