Information Sciences 641 (2023) 119085

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Decomposing shared networks for separate cooperation with multi-agent reinforcement learning

Weiwei Liu [a], Linpeng Peng [a], Licheng Wen [a], Jian Yang [b], Yong Liu [a,*]

[a] The Advanced Perception on Robotics and Intelligent Learning Lab, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China

[b] China Research and Development Academy of Machinery Equipment, Beijing, China

ARTICLE INFO

Keywords:
Multi-agent reinforcement learning
Neural network
Multi-agent systems
Navigation planning

ABSTRACT

Sharing network parameters between agents is an essential and typical operation to improve the scalability of multi-agent reinforcement learning algorithms. However, agents with different tasks sharing the same network parameters are not conducive to distinguishing the agents' skills. In addition, the importance of communication between agents undertaking the same task is much higher than that with external agents. Therefore, we propose Dual Cooperation Networks (DCN). In order to distinguish whether agents undertake the same task, all agents are grouped according to their status through the graph neural network instead of the traditional proximity. The agent communicates within the group to achieve strong cooperation. After that, the global value function is decomposed by groups to facilitate cooperation between groups. Finally, we have verified it in simulation and physical hardware, and the algorithm has achieved excellent performance.

1. Introduction

Group cooperation is ubiquitous in nature. In recent years, the multi-agent deep reinforcement learning (MARL) algorithms have shown performance beyond human level in many cooperative environments, such as StarCraft [1,2], Multi-Agent Particle Environment [3][4] and Magent [5][6]. However, the MARL algorithms still have many enormous challenges, including scalability, the partial observability of the environment, and the credit assignment among agents.

The scalability problem is that agents have their networks, making algorithm training extremely difficult as the number of agents increases and the network parameters expand. A common trick to solve this problem in previous work [7] is to share network parameters. However, due to the coupling and similarity of network parameters, the indiscriminate application of network parameter sharing to all agents is not conducive to agents distinguishing self-tasks and affects the final convergence of the algorithm.

In addition, partial observability results in the agent being unable to obtain the actual state of the environment. Scholars currently use the information of all agents to establish an unbiased global critic network [8][9][10]; this also invisibly solves the problem of the agent's credit assignment. Moreover, MADDPG [8] adopts decentralized execution for facilitating algorithm deployment. That is the Centralized Training and Decentralized Execution (CTDE). However, global communication in training makes it difficult for agents to distinguish the valuable information helpful for cooperative decision-making. Therefore, global communication may even

* Corresponding author.
E-mail address: yongliu@iipc.zju.edu.cn (Y. Liu).

https://doi.org/10.1016/j.ins.2023.119085
Received 22 May 2022; Received in revised form 19 March 2023; Accepted 30 April 2023

Available online 9 May 2023

harm cooperative learning. Furthermore, the agent only makes decisions in deployment based on its observation and ignores other agents, leading to a sub-optimal joint action.

Agents with the same task share network parameters to help them learn similar behaviors. For example, it is difficult for football forwards and defenders to use the same network to fit two different modes of behavior, but the same group of agents using the same network to fit reduces the learning difficulty due to their similar strategies. For the agents to use the shared network discriminately, reduce the cost of agent communication and use valuable information to promote cooperation, we propose a novel algorithm called Dual Cooperation Networks (DCN). We observe that the agents' observations with the same task are similar or satisfy a specific rule. Thus, we introduce Graph Auto-Encoder [11] (GAE) to group according to the observations of the agents. The agents' actions in each group are output simultaneously by the Group Cooperation Networks (GCoNet). In addition, since the same reward function is still shared among groups, we introduce VDN [12] for value decomposition operation.

The main contributions are as follows:

- Dividing the group based on the information relationship between the agents rather than the traditional proximity. Due to agents with potential cooperative relations, the information they observe is internally consistent, such as consistent goals. Therefore, the graph autoencoder can encode this high-dimensional information and divide the agents into internal consistency cooperative groups based on the encoded information. We have not seen such methods used to divide agent groups in other papers.
- We propose Dual Cooperation Networks. In the same group, all agents will consider the state of other agents to make joint decisions, and this is the cooperation of agents within the group that significantly promotes the cooperation ability of the agents. In addition, the shared reward function is value-decomposed according to different groups' value functions for cooperation between groups.
- Finally, DCN is verified on simulation and physical hardware, and its performance is better than the baseline algorithm.

2. Related work

Although reinforcement learning [13,14] has made outstanding achievements in the field of single agents, there are many multi-agent systems in nature. However, the simple extension of the single-agent algorithm to the multi-agent system often has these two problems: the non-stationarity problem [15] of the multi-agent environment and the credit assignment problem [16].

2.1. Nonstationarity of multi-agent systems

In the multi-agent system, the agent only relies on its information to make decisions and is easily disturbed by other agents, resulting in non-stationarity problems. In order to solve it, some works propose to centrally train a joint value function to evaluate agent behavior using information from all agents. Depending on the information required for training and execution, multi-agent reinforcement learning algorithms can be divided into CTCE (Centralized Training and Centralized Execution) and CTDE. CommNet [17] trains and executes each agent's strategy, and the average value of the confidential information of other agents' observations is used as part of the network input. Wang et al. [1] propose to give the hidden information in the form of a maximum or sum value. So those are the typical CTCE algorithm. They let all agents share the same network parameters to solve the problem that the CTCE algorithm is difficult to scale. All agent information is averaged after circulating through the network to extract features. Therefore, even if the number of agents changes, effective communication can be achieved without changing the algorithmic network architecture. However, it is evident that this extraction method is too crude, and obtaining advanced spatial feature information between agents is challenging. In the CTDE algorithm, the scalability of the algorithm execution is naturally solved, because it only relies on the agent's local information during the execution phase. For example, MADDPG [8] uses all agent information to train the joint action-value function, but only relies on the agent's own information to make decisions. Furthermore, this also implicitly solves the multi-agent credit assignment problem.

However, communication among all agents would incur extremely high costs. Moreover, scholars have realized that the agent does not need to observe the state or behavior of all participants, but only needs to observe the state or behavior information of some related agents in the neighborhood. Therefore, agents only need to perform strong communication within the group after grouping. Some works have been done to model communication protocol to allow agents to obtain more information. LICA [16], MAAC [18] DGN [10] use different communication protocols for mutual communication between agents. The NCC-MARL [19] based on the neighborhood's cognitive consistency [20] is proposed to conduct strong cooperation with agents in the domain and weak cooperation with agents outside the domain. However, these methods of building groups based on agent distance are not applicable in some scenarios. It is worth noting that the application scenario of these algorithms is that the agent can obtain its reward function, and ignores the reputation distribution problem.

In this work, we use graph autoencoders [11] to extract high-level structural information between agents, and group agents based on it, so that agents performing the same task can complete the collaboration between different groups. Furthermore, the problem of credit assignment is solved through value decomposition [21,22] operation.

2.2. Graph neural network

Inspired by the prevalent cognitive consistency theory in the field of social psychology, it was found that multi-agent cognition of the environment is a necessary condition for good collaboration. The grouping between agents should start from the states of the

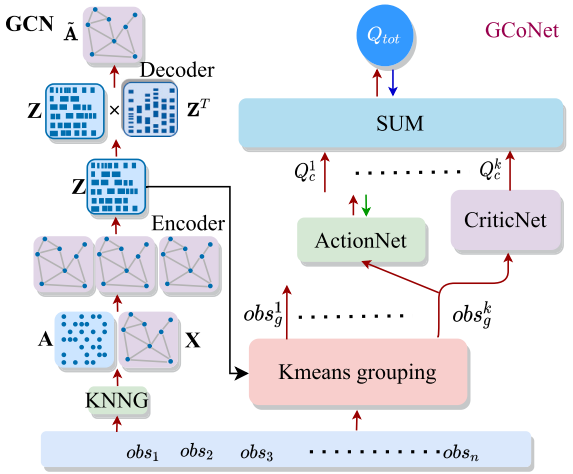


Fig. 1. Dual cooperation networks framework diagram. DCN can be divided into Graph Auto-encoder and Group Cooperation Networks (GCoNet).

agents not just the distance. Due to the non-independence between agents, each agent is an instance in the graph network, using GAE to capture the spatial dimension relationship between agents easily. In order to achieve both the dimensionality reduction and extraction of features and the mining of structural information, the Structural Deep Clustering Network [23] (SDCN) achieves the capture of high-order structural information by stacking multiple layers of Graph Neural Networks [24] (GNN). At the same time, it benefits from the self-encoder and the self-contained GNN supervised. The multi-layer GNN here does not have the so-called oversmooth phenomenon (as the number of layers increases, the node representations learned by GNN gradually become indistinguishable.). The previous deep clustering algorithm is divided into two steps: first, learn the feature representation of the data to represent embedding, and then cluster the data according to the feature representation. However, in this way, the learned data embedding is not task-oriented. On the contrary, deep attention embedding graph clustering [25] (DAEGC) learns node representation through graph attention networks [26] (GAN), and at the same time enhances the cohesion between nodes in the same cluster through self-training graph clustering and solves the clustering problem end-to-end.

In addition, graph networks are used in many reinforcement learning algorithms [27]. MAGNet [28] learns the relevant information of instances and agents in a multi-agent environment through a correlation graph, and incorporates it into the reinforcement learning process. HAMA [29] uses the hierarchical graph attention mechanism to extract dynamic quantitative agent features and inter-group information to improve the generalization ability of the algorithm. Compared with this work, most of the above work is used to extract the spatial features between agents, but ignores the problem of agent grouping and still uses the distance grouping between agents. However, this work relies on the graph network to judge whether the agents perform the same task based on the spatial feature information between the agents, thereby enabling task-oriented grouping.

3. Methods

In a football game, forwards and defenders have entirely different strategies. If the same network fitting is used for these two strategies to improve the algorithm's scalability, its performance will be affected due to the coupling and similarity of network parameters. However, since the agents' strategies for undertaking the same task are highly consistent, using the same network fitting will increase the learning speed. In addition, another starting point of this work is that in a multi-agent system, each agent should have strong cooperation for participants in the domain and weak cooperation between teams outside the domain. Inspired by this, we proposed a novel algorithm: DCN, whose structure is shown in Fig. 1. In addition, it is challenging to provide a separate reward function for each agent in a multi-agent system. Generally, all agents share a joint reward function, which also brings about the agent's credit assignment problem [30,31] —the rewards of lazy and hardworking agents are the same. According to the value function of each agent, Peter Sunehag proposed a VDN algorithm to decompose joint rewards, that is, value decomposition. However, the VDN algorithm can only be used in discrete action environments. It needs to splice the local observations of all agents as the algorithm input and is only suitable for the smallest amount of agent cooperation. The DCN algorithm transforms the joint value function based on the decomposition of the value function of a single agent into a value function based on the group. The proof will be given in the next section.

3.1. Group value decomposition

According to the VDN [12] algorithm, and taking the cooperation of four agents as an example:

$$r(\mathbf{s},\mathbf{a}) = r^1\left(o^1,a^1\right) + \cdots + r^4\left(o^4,a^4\right), \tag{1}$$

among them, $\mathbf{s}$ and $\mathbf{a}$ respectively represent the joint observation $(o^1,\dots,o^4)$ and joint action $\left(a^1,\cdots,a^4\right)$ of all agents. o^i, a^i respectively represent the local observation and action of agent i. r, r^i respectively represent the joint reward of all agents and the reward of agent i. Decompose the joint value function according to the value function of each agent to get:

$$\begin{aligned} Q^{\pi}(\mathbf{s},\mathbf{a}) &= \mathbb{E}[\sum_{t=1}^{\infty}\gamma^{t-1}r(\mathbf{s}_t,\mathbf{a}_t)|\mathbf{s}^1=\mathbf{s},\mathbf{a}^1=\mathbf{a};\pi] \\ &= \mathbb{E}[\sum_{t=1}^{\infty}\gamma^{t-1}r^1(o_t^1,a_t^1)|\mathbf{s}^1=\mathbf{s},\mathbf{a}^1=\mathbf{a};\pi]+\cdots \\ &+ \mathbb{E}[\sum_{t=1}^{\infty}\gamma^{t-1}r^4(o_t^4,a_t^4)|\mathbf{s}^1=\mathbf{s},\mathbf{a}^1=\mathbf{a};\pi] \\ &=: \bar{Q}_1^{\pi}(\mathbf{s},\mathbf{a})+\cdots+\bar{Q}_4^{\pi}(\mathbf{s},\mathbf{a}), \end{aligned} \tag{2}$$

where $\bar{Q}_i^{\pi}$ represents the state-action value function of agent i.
$\bar{Q}_i^{\pi}(\mathbf{s},\mathbf{a}) := \mathbb{E}[\sum_{t=1}^{\infty}\gamma^{t-1}r^i(o_t^i,a_t^i)|\mathbf{s}^1=\mathbf{s},\mathbf{a}^1=\mathbf{a};\pi], i=1,\cdots,4$. Similarly:

$$r_g^p(\mathbf{s},\mathbf{a}) = r^i\left(o^i,a^i\right)+r^j\left(o^j,a^j\right), \tag{3}$$

where $r_g^p(\mathbf{s},\mathbf{a})$ represents the joint reward of the p-th group of agents. g means group. Agent i,j is in group p, $i,j\in\{1,\cdots,4\}$ and $i\neq j$.

The agents are divided into two groups, p and q. It can be derived from formulas (1) and (2):

$$\begin{aligned} r(\mathbf{s},\mathbf{a}) &= r^1\left(o^1,a^1\right)+\cdots+r^4\left(o^4,a^4\right) \\ &= r_g^p(\mathbf{s},\mathbf{a})+r_g^q(\mathbf{s},\mathbf{a}). \end{aligned} \tag{4}$$

Similarly, according to the value function of each group, the joint-value function is decomposed:

$$\begin{aligned} Q^{\pi}(\mathbf{s},\mathbf{a}) &= \mathbb{E}[\sum_{t=1}^{\infty}\gamma^{t-1}r(\mathbf{s}_t,\mathbf{a}_t)|\mathbf{s}^1=\mathbf{s},\mathbf{a}^1=\mathbf{a};\pi] \\ &= \mathbb{E}[\sum_{t=1}^{\infty}\gamma^{t-1}r^p(o_t^p,a_t^p)|\mathbf{s}^1=\mathbf{s},\mathbf{a}^1=\mathbf{a};\pi] \\ &+ \mathbb{E}[\sum_{t=1}^{\infty}\gamma^{t-1}r^q(o_t^q,a_t^q)|\mathbf{s}^1=\mathbf{s},\mathbf{a}^1=\mathbf{a};\pi] \\ &=: \bar{Q}_p^{\pi}(\mathbf{s},\mathbf{a})+\bar{Q}_q^{\pi}(\mathbf{s},\mathbf{a}), \end{aligned} \tag{5}$$

where $\bar{Q}_p^{\pi}(\mathbf{s},\mathbf{a})$ represents the joint state-action value function of the p-th group of agents. $\bar{Q}_p^{\pi}(\mathbf{s},\mathbf{a}) := \mathbb{E}[\sum_{t=1}^{\infty}\gamma^{t-1}r_g^p(o_t^i,o_t^j,a_t^i,a_t^j)|\mathbf{s}^1=\mathbf{s},\mathbf{a}^1=\mathbf{a};\pi], p\in\{1,2\}$.

3.2. Use graph auto-encoder and Kmeans grouping

More concretely, it is considered that a game has N agents, which can be divided into k groups according to the tasks they need to complete. As shown in Fig. 1, to divide groups based on the global state rather than the positional relationship of agents, we introduce a graph autoencoder to extract high-dimensional spatial features between agents, and use this as the basis for grouping. The input of the graph autoencoder is an undirected graph structure. In order to use the state of the agent as the input of the autoencoder, the k-Nearest Neighbour Graph [32,33] (KNNG) algorithm is first used to construct the structure graph of the agent. The KNNG uses the distance between agents as the edges in the graph, and the agents are the vertices in the graph, which is:

Definition 1. *The graph is represented by $\mathcal{G}=(\mathcal{V},\varepsilon)$, where $\mathcal{V}$ represents the set of nodes and ε represents the set of edges. N is the number of nodes and also the number of agents. We introduce an adjacency matrix $\boldsymbol{A}$ of $\mathcal{G}$ and its degree matrix $\boldsymbol{D}$. d: the feature dimension of the node, $\boldsymbol{X}\in\mathbb{R}^{N\times d}$: the feature matrix of the node. f: dimensions of embedding. $\boldsymbol{Z}\in\mathbb{R}^{N\times f}$: node embedding.*

GAE uses Graph Convolutional Networks [34,35] (GCN) as encoder to get the latent representations of nodes:

$$\mathbf{Z} = GCN(\mathbf{X},\mathbf{A}), \tag{6}$$

among them:

$$GCN(\mathbf{X},\mathbf{A}) = \tilde{\mathbf{A}}ReLU(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0)\mathbf{W}_1, \tag{7}$$

among them: $\tilde{\mathbf{A}}=\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix. $\mathbf{W}_0$ and $\mathbf{W}_1$ are the parameters to be learned.

GAE uses inner-product as a decoder to reconstruct the original graph:

$$\tilde{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^T). \tag{8}$$

Algorithm 1 DCN for N agents.

Input: Randomly initialize GAE, actor and critic network G, π and Q with weights θ_g, θ_π and θ_q
for episode = 1, T **do**
  for steps $t = 1$, T **do**
    Get agents' observations $\mathbf{s} = \{o_1, \cdots, o_n\}$
    Get grouping matrix $\mathbf{G}$ according to Equation (11)
    Grouping according to Equation (12)
    Get each agent i's action $a_i = \pi_i(o_i) + \mathcal{N}_t$
    Perform actions, interact with the environment to get rewards r and next states $\mathbf{s}'$
    Store $(\mathbf{s}, r, \mathbf{a}, \mathbf{s}')$ in replay buffer $\mathcal{D}$
  end for
  for steps $k = 1$, train-steps **do**
    Randomly select a mini batch from the replay buffer $\mathcal{D}$
    Training graph autoencoders according to equation (9)
    Training joint action value network according to equation (5) , $y^{tot} = r + \gamma Q^{tot}(\mathbf{s}', \mathbf{a}')$ and $\mathcal{L} = (y^{tot} - Q^{tot}(\mathbf{s}, \mathbf{a}))^2$
    Train the action network for each agent i in group p according to equation $\mathcal{L}_i = -Q_p(o_i, \pi_i(o_i), \mathbf{s}_p, \mathbf{a}_p)$
    Update target network parameters
  end for
end for

In the above formula, $\tilde{\mathbf{A}}$ is the adjacency matrix reconstructed.

Because the adjacency matrix determines the structure of the graph, in order to obtain the optimal $\mathbf{Z}$, the reconstructed adjacency matrix is as similar as possible to the original adjacency matrix. Therefore, in the training process of GAE, cross entropy is used as the loss function:

$$\mathcal{L} = -\frac{1}{N} \sum \hat{y} \log y + (1 - y) \log(1 - \hat{y}). \tag{9}$$

In the above formula, y represents the value (0 or 1) of an element in the adjacency matrix $\mathbf{A}$, and $\hat{y}$ represents the value (between 0 and 1) of the corresponding element in the reconstructed adjacency matrix $\tilde{\mathbf{A}}$.

Based on the embedding obtained by the graph auto-encoder, Kmeans [36,37] algorithm is used to cluster it. It is defined as follows:

Definition 2. *Suppose that a given data sample X contains N objects $X = \{X_1, X_2, \cdots, X_n\}$. Each of these objects has attributes of m dimensions. Initialize k cluster centers $\{C_1, C_2, \cdots, C_k\}, 1 < k \leq n$.*

Calculate the Euclidean distance from each object to each cluster center, as shown in the following formula:

$$dis(X_i, C_i) = \sqrt{\sum_{t=1}^{m} \left(X_{it} - C_{jt}\right)^2}. \tag{10}$$

In the above formula, X_i represents the i-th object, C_j represents the j-th cluster center, $1 \leq j \leq k$, X_{it} represents the t-th attribute of the i-th object, $1 \leq t \leq m$. C_{jt} represents the t-th attribute of the j-th cluster center.

Compare the distance of each object to each cluster center in turn, assign the objects to the clusters of the nearest cluster center, and get k clusters $\{S_1, S_2, \cdots, S_k\}$. The Kmeans algorithm uses the center to define the prototype of the cluster. The center of the cluster is the average value of all objects in the cluster in each dimension. The calculation formula is as follows:

$$C_t = \frac{\sum_{X_i \in S_l} X_i}{|S_l|}. \tag{11}$$

In the formula, C_l represents the center of the l-th cluster, $1 \leq l \leq k$, $|S_l|$ represents the number of objects in the l-th cluster, and X_i represents the i-th object in the l-th cluster, $1 \leq i \leq |S_l|$.

Then through Kmeans algorithm, obtain the grouping matrix $\mathbf{G}$. The group joint observation value can be calculated:

$$[O_g^1, \cdots, O_g^k] = [o_1, \cdots, o_n] * \mathbf{G} \tag{12}$$

In the above formula, o_i are the local observation value of each agent, and O_g^i are the joint observation value of each group.

3.3. Network training

As shown in Fig. 1, the grouping matrix is obtained after task-oriented grouping by the graph autoencoder, where the graph autoencoder's loss function is shown in Equation (9). After that, all agents are grouped according to the grouping matrix. After the grouping is completed, the agents in the same group are output by the same network, and the joint value function trains the action network within the group. The joint value function between different groups is trained by Equation (5). The training process can be seen in Algorithm 1.

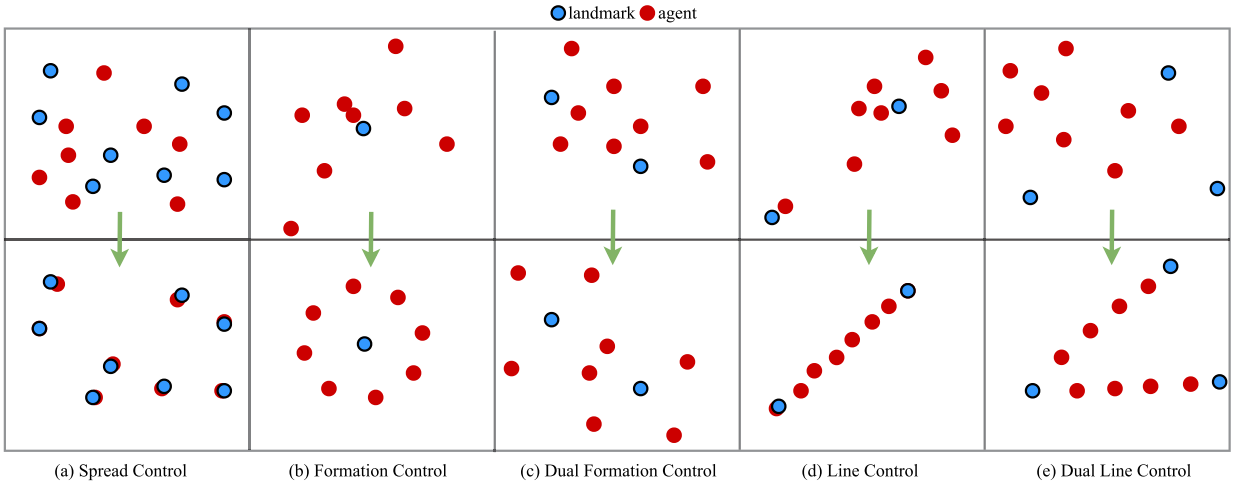


Fig. 2. Illustration of multi-agents learning to cooperate in five scenarios.

4. Experiment

The baseline algorithm for comparison uses the well-known MADDPG [8], ATOC [38] and MAAC [18]. All agents share the same reward function and network parameters. MADDPG implicitly solves the problem of multi-agent credit assignment; ATOC establishes fixed-size groups for cooperation based on the distance between agents; MAAC uses the attention mechanism to make each agent choose useful agent information from the global information to promote its capabilities. Finally, DCN can be divided into Graph Auto-encoder (GAE) and Group Cooperation Networks (GCoNet), to illustrate the necessity of dividing groups between agents according to the spatial structure, we also compared it with the Kmeans+GCoNet algorithm. In order to distinguish, the following experiment uses GAE+GCoNet to represent DCN. We have conducted multiple experiments on both simulation (Fig. 2) and physical platforms (Fig. 4) to verify the effectiveness of the algorithm. The experimental parameters and equipment platform information are as follows: The replay buffer size is $1 \times e6$, the training batch size is 256, the soft update parameter $\tau = 0.01$, and the network parameter learning rate of the actor and critic network is $1 \times e{-}4$. All experiments are performed on the same computer for simulation and calculation, equipped with i7-8700 CPU, 16GB RAM, and GeForce GTX 1080 Ti.

4.1. Simulations

We evaluate DCN through three standard multi-agent robotic tasks [39][40]. They are implemented through the open-source environment Multi-Agent Particle Environment [3]. According to the double integral dynamic model, agents can move in two-dimensional space [41]. The agent can control the acceleration or deceleration of the unit in the X and Y directions. As shown in Fig. 2, we briefly describe the following five environments:

Spread Control: As shown in Fig. 2(a), there are eight agents and landmarks on the map. The agent needs to reach landmarks from the chaotic state. Its reward is set to negative value of the distance between the agent and the nearest landmark. Namely, the further away from the landmark, the lower the score of the agent. In addition, to avoid collisions between agents, every collision between agents will cause them to lose a point.

Formation Control and Dual Formation Control: As shown in Fig. 2(b) and (c), this map has a landmark and eight agents. Agents need to arrange themselves neatly in a circle or two squares, and the landmark is the center of the circle or square. The reward setting is similar to the first task. The negative value of the distance between the agent and the nearest desired location is used as a reward through the bipartite matching algorithm. Note that the reward is regularized.

Line Control and Dual Line Control: As shown in Fig. 2(d) and (e), the map has two landmarks and eight agents. Agents need to be arranged equidistantly on the line between two or three landmarks in one or two columns. The reward setting is the same as the second task, but the desired position is different.

The map of the above five tasks is the standard area of MAPE: 2×2 sq. In the first task, we set each agent to perceive the positions of the four nearest agents and landmarks respectively; In the next four tasks, we set the communication distance to 1 unit. At this time, this situation forms the Partially Observable Markov Decision Process. Each episode lasts 25 time-steps in total. In the Dual Formation Control and Dual Line Control task, the agents of different groups have cross positions, the structural information between agents has a more obvious impact on the algorithm at this time, and the baseline algorithm only relies on distance to group and will fall into a local optimum. The following experimental results also prove this point.

4.1.1. Comparison with baseline algorithms in simulation

We conducted five different simulation experiments in Multi-Agent Particle Environment, and the results are shown in Fig. 3.

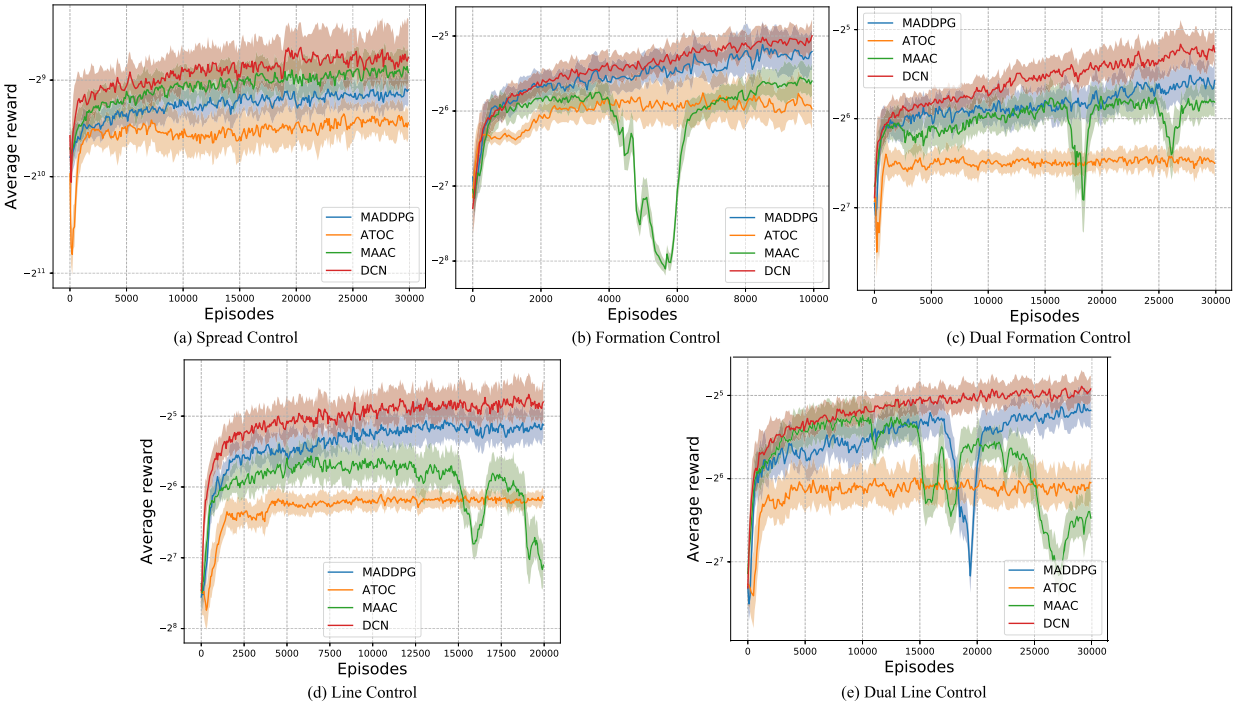


Fig. 3. Agent average reward comparison chart in the five control task.

Spread Control: In this task, the agent needs to move to occupy the landmark as soon as possible and need to understand the intentions of other agents to prevent multiple agents from using the same landmark as their target. Due to the physical dynamics of the agent, collisions will cause the agent to move uncontrollably, so it is necessary to try to avoid collisions during the movement. Fig. 3 shows the average score of the agent under each algorithm. ATOC is lower than DCN and even lower than MADDPG, which shows that grouping based on agents' relative position is not conducive to cooperation between them. The DCN and MAAC are superior to other algorithms. It shows that in a multi-agent system, the agent does not consider the information of all agents at the same time but only considers the information of the critical agents, which can promote cooperation between agents. This is because the information of the global agent is sometimes useless or even harmful to collaboration.

Formation Control and Dual Formation Control: As shown in Fig. 3, the scoring trend is generally the same as that of the first task. The rising speed of the scoring curve has a small change, and this is because we canceled the measure of losing points caused by the collision of the agent in this task. Moreover, compared to the line control task, the collision probability of the agent is smaller, so there is generally no uncontrolled movement. Different from task one, the multi-agent system needs closer cooperation within the group at this time. Especially in the dual formation control task, the multi-agent system obviously needs to be divided into two groups for cooperation. Since all agents share the same reward function, weak cooperation is still needed between groups to avoid local optimality. As shown in Fig. 3 (b) and (c), algorithms such as MAAC and ATOC that simply divide groups based on the relative positions of agents have significantly reduced performance. This is because although some agents in these two tasks are relatively close, due to their different mission objectives, they still cannot be grouped into the same group for collaboration.

Line Control and Dual Line Control: As shown in Fig. 3 (d) and (e), although the overall score trend remains unchanged, compared with other baseline algorithms, DCN scores the highest. Furthermore, the scoring curve rises faster at the beginning, and this is because all agents need to be arranged in a straight line within two points within a limited distance. Due to space constraints, agents are very prone to collisions, leading to uncontrolled movements and further away from the target location, resulting in loss of points. On the contrary, once the agent learns the knowledge of avoiding each other, although the task is not completed, the score increases faster.

4.1.2. Ablation experiments

In order to further observe the performance of DCN and to test the effect of general algorithm grouping according to the proximity of agents, we use the Kmeans algorithm to group agents and verify the necessity of multi-agent system grouping based on data that retains high-dimensional structural information. Comparing Kmeans+GCoNet with DCN is shown in Fig. 4. It can be seen from the figure that the performance of Kmeans+GCoNet is significantly lower than that of DCN, indicating that grouping according to the relative position of agents is not conducive to cooperation between them.

In addition, we observed that the cooperation between agents takes a certain amount of time to manifest, so we set the grouping frequency as 5, 10 and 15. As shown in Fig. 5, an appropriate grouping frequency is conducive to improving the algorithm's

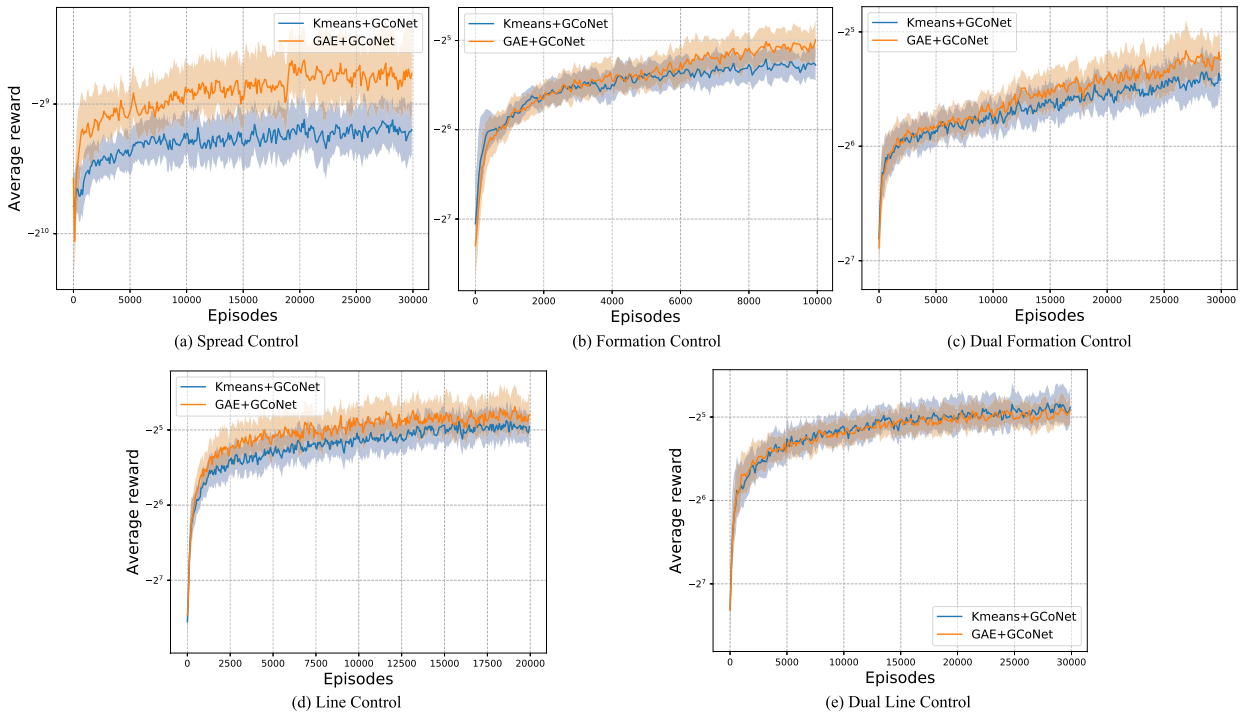


Fig. 4. Comparison of average rewards between DCN and Kmeans + GCoNet in the five control task.

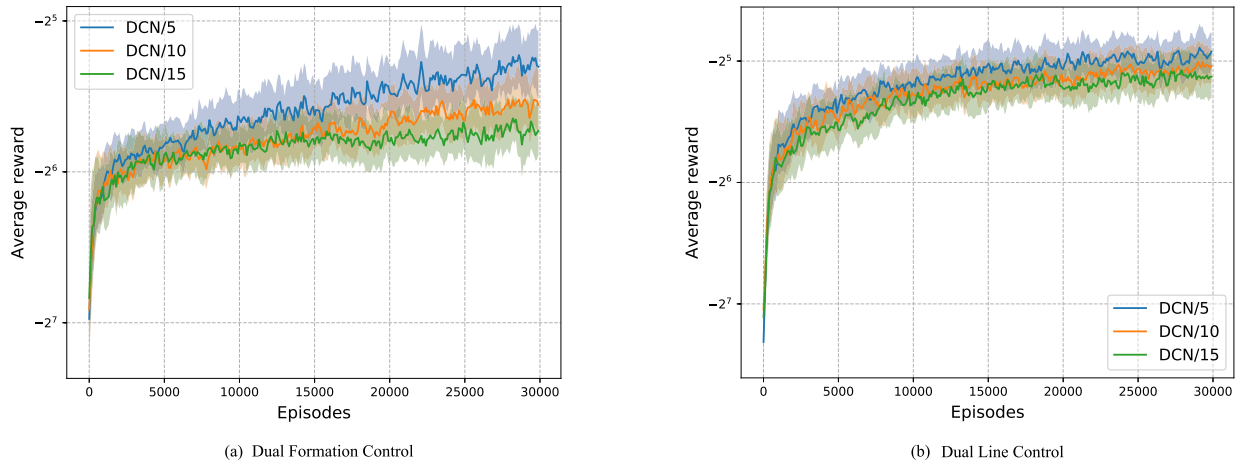
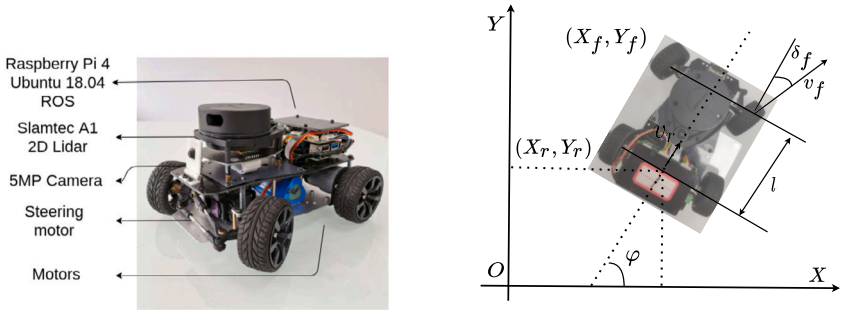


Fig. 5. The influence of grouping frequency on DCN, 5 in the figure means that the agent performs grouping at intervals of 5 steps, and other numbers have similar meanings.

Table 1
Experiment site and trolley parameter table, MLS: Maximum linear speed, MA: Maximum acceleration, MAA: Maximum angular acceleration, MLSP: Minimum line speed, MTR: Minimum turning radius.

Field length 4 m	MLS 0.6 m/s	MA 0.5 m/s^2	MAA 0.5 rad/s^2	Wheelbase 0.145 m
Field width 2.3 m	MLSP −0.2 m/s	MTR 0.375 m	Car length 0.22 m	Car width 0.185 m

performance, while a too low grouping frequency is not conducive to algorithm convergence because it cannot only follow the development of the situation.



(a) Experimental vehicles and on-board equipment. (b) Vehicle kinematics model

Fig. 6. Experimental vehicles and vehicle kinematics model.

4.2. Physical experiments

We use the Ackermann trolley for experiments, it is shown in Fig. 6(a). The main components include battery, motor, encoder, steering servo, inertial measurement unit (IMU), control board, raspberry pie, CMOS sensor interface (CSI) camera, lidar, etc. The cars communicate through the Robot Operating System (ROS). The parameters of the experiment site and trolley are shown in Table 1. We use the simultaneous localization and mapping (SLAM) [42] algorithm to give the location of each car, and through the local masking algorithm, each car can only know the location of the ten closest obstacles to it, and the fence is also calculated as a stack of obstacles of the same size. That is, to achieve partially observable Markov decisions. The algorithm outputs the acceleration and front-wheel deflection angle of each car, and obtains the next time position of the vehicle through the integration of the dynamic model of the Ackermann car, sends it to the vehicle, and navigates through the time elastic band (TEB) [43] local planner algorithm. Since the cost of training algorithms in real objects is too high, we pre-trained in simulation and fine-tuned in real objects.

4.2.1. Vehicle kinematics model

The vehicle steering model is shown in Fig. 6(b). In the inertial coordinate system XOY, (X_f, Y_f) and (X_r, Y_r) represent the coordinates of the center of the rear and front axles of the vehicle. φ indicates the heading angle of the vehicle, δ_f indicates the front wheel's deflection angle, and v_r indicates the speed of the center of the rear axle of the vehicle along the direction of the vehicle axis. v_f represents the speed of the center of the vehicle's front axle along the direction of the front wheel deflection, and l represents the distance between the front and rear axles of the vehicle.

$$v_r = \dot{X}_r * \cos(\varphi) + \dot{Y}_r * \sin(\varphi). \tag{13}$$

Kinematic constraints of the front and rear axles (the vehicle has no lateral sideslip):

$$\begin{cases} \dot{X}_r * \sin\left(\varphi + \delta_f\right) - \dot{Y}_r * \cos\left(\varphi + \delta_f\right) = 0 \\ \dot{X}_r * \sin\left(\varphi\right) - \dot{Y}_r * \cos\left(\varphi\right) = 0 \end{cases} \tag{14}$$

According to formula (16) and formula (17), we can get:

$$\begin{cases} \dot{X}_r = v_r \cos(\varphi) \\ \dot{Y}_r = v_r \sin(\varphi) \end{cases} \tag{15}$$

According to the position relationship of the front and rear wheels:

$$\begin{cases} X_f = X_r + l * \cos(\varphi) \\ Y_f = Y_r + l * \sin(\varphi) \end{cases} \tag{16}$$

From the above formula, the angular velocity ω can be obtained as:

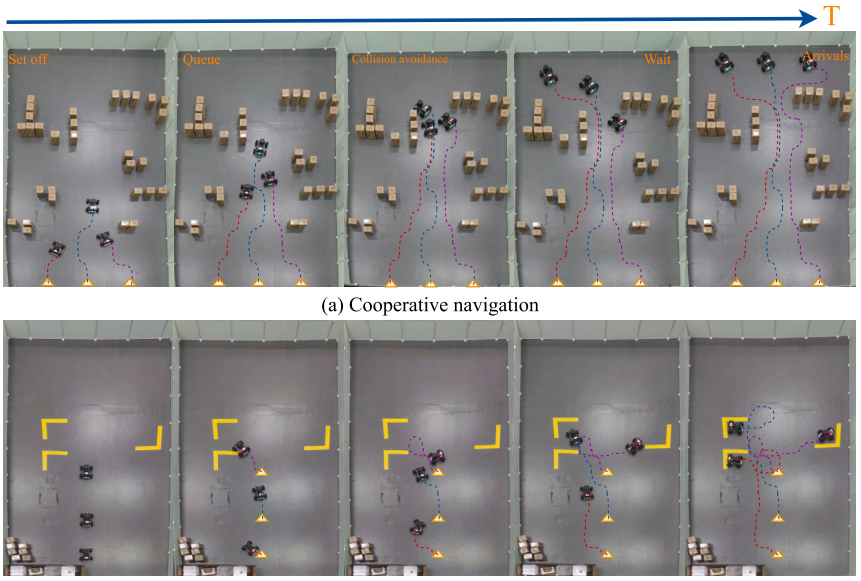
$$\dot{\varphi} = \omega = \frac{v_r}{l} * \tan(\delta_f) \tag{17}$$

Under the condition of different front-wheel steering angles, the corresponding different steering radius:

$$\begin{cases} R = v_r * \omega \\ \delta_f = \arctan\left(\frac{l}{R}\right) \end{cases} \tag{18}$$

The vehicle kinematics model can be drawn:

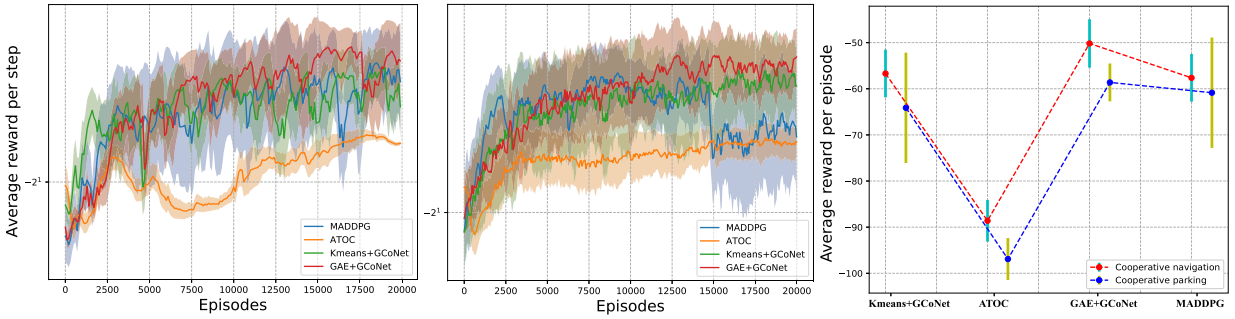
$$\begin{bmatrix} \dot{X}_r \\ \dot{Y}_r \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ \frac{\tan(\delta_f)}{l} \end{bmatrix} * v_r \tag{19}$$



(a) Cooperative navigation

(b) Cooperative parking

Fig. 7. Agent average reward comparison chart in the five control task.



(a) Cooperative Navigation (b) Cooperative Parking (c) Final rewards for the two tasks

Fig. 8. Each step and the final average reward of the two tasks.

4.2.2. *Experimental setup and result analysis*

Based on the above experimental platform, we designed two physical experiments-cooperative navigation and cooperative parking.

The process of the cooperative navigation task is that the three agents start from the starting point, traverse the obstacle zone, and reach the endpoints. In this process, the agent needs to avoid collisions and reach their respective endpoints. The task of cooperative parking is that the vehicle needs to park safely in the parking space, and the body and the lane line are consistent to avoid the collision. The two tasks reward function is the same as the Formation Control. In addition, the agent's collision will reduce the reward by five and stop the episode, with a maximum of 60 steps per episode. Since the number of steps in each round is inconsistent, for a fair comparison, we choose $\frac{r_e}{n_e}$ as the comparison object of different algorithms, where r_e represents the total reward of an episode, and n_e represents the number of steps in this episode.

Fig. 7(a) and (b) show the entire process of cooperative navigation and cooperative parking tasks. The agent learns the communication protocol during the training process and obtains the knowledge of avoiding each other, queuing in a narrow area, or even waiting in place in these two tasks to allow other agents to pass. Fig. 8(a) and (b) show that even in an environment where structural information is not apparent, the average score per step of DCN is still higher than other baseline algorithms. The final average score in Fig. 8(c) also verifies this point. Moreover, the performance of the MADDPG algorithm is similar to Kmeans+GCoNet, indicating that it is feasible to treat the agent in the group as multiple parts of an agent. In addition, the suspension of the game caused by the collision makes the average score fluctuate considerably. Finally, in the cooperative navigation task in Fig. 8(c), the score variance of each comparison algorithm is significant, and the score variance of the cooperative navigation parking task is slight and similar. This is because there are many obstacles in the first task, and the route of an agent is restricted by other agents and obstacles simultaneously, which makes it easy to collide and the process is easy to stop. At this time, the algorithm falls into a local optimum.

5. Conclusions

In the current multi-agent reinforcement learning work, to improve the algorithm's scalability, all agents perform network parameter sharing operations indiscriminately, and almost no one has paid attention to selectively sharing network parameters for different types of agents. Therefore, this paper proposes DCN, which is a selective network parameter sharing scheme for agents with common tasks or similar abilities. To more effectively utilize agents' spatial structure and high-dimensional information, we introduce graph autoencoders to group agents. DCN has two kinds of cooperation capabilities of agents inside and outside the group, which maintains the scalability of the algorithm and significantly improves the algorithm's performance.

It is applied to homogeneous agents and performs fully cooperative tasks to demonstrate that DCN can group agents with task orientation. However, because there are more significant differences in heterogeneous or competing task agents, and the same class of agents should perform the same task. In the future, we may extend the algorithm to heterogeneous agents or perform competing tasks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

This work is funded by the National Key R&D Program of China (Grant No: 2018AAA0101503) and the Science and technology project of SGCC (State Grid Corporation of China): fundamental theory of human-in-the-loop hybrid-augmented intelligence for power grid dispatch and control.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ins.2023.119085.

References

[1] W. Wang, T. Yang, Y. Liu, J. Hao, X. Hao, Y. Hu, Y. Chen, C. Fan, Y. Gao, From few to more: large-scale dynamic multiagent curriculum learning, Proc. AAAI Conf. Artif. Intell. 34 (2020) 7293–7300.
[2] O. Vinyals, I. Babuschkin, W.M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D.H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al., Grandmaster level in StarCraft II using multi-agent reinforcement learning, Nature 575 (7782) (2019) 350–354.
[3] I. Mordatch, P. Abbeel, Emergence of grounded compositional language in multi-agent populations, arXiv preprint, arXiv:1703.04908.
[4] A. Agarwal, S. Kumar, K. Sycara, Learning transferable cooperative behavior in multi-agent teams, arXiv preprint, arXiv:1906.01202.
[5] J.K. Terry, B. Black, M. Jayakumar, Magent, https://github.com/PettingZoo-Team/MAgent, 2020, gitHub repository.
[6] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, J. Wang, Mean field multi-agent reinforcement learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 5571–5580.
[7] J.K. Gupta, M. Egorov, M. Kochenderfer, Cooperative multi-agent control using deep reinforcement learning, in: International Conference on Autonomous Agents and Multiagent Systems, Springer, 2017, pp. 66–83.
[8] R. Lowe, Y.I. Wu, A. Tamar, J. Harb, O.P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, in: Advances in Neural Information Processing Systems, 2017, pp. 6379–6390.
[9] P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, J. Wang, Multiagent bidirectionally-coordinated nets: emergence of human-level coordination in learning to play StarCraft combat games, arXiv preprint, arXiv:1703.10069.
[10] J. Jiang, C. Dun, T. Huang, Z. Lu, Graph convolutional reinforcement learning, in: International Conference on Learning Representations, 2019.
[11] T.N. Kipf, M. Welling, Variational graph auto-encoders, arXiv preprint, arXiv:1611.07308, 2016.
[12] P. Sunehag, G. Lever, A. Gruslys, W.M. Czarnecki, V.F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J.Z. Leibo, K. Tuyls, et al., Value-decomposition networks for cooperative multi-agent learning based on team reward, in: AAMAS, 2018.
[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint, arXiv:1312.5602.
[14] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint, arXiv:1509.02971.
[15] A. Oroojlooy, D. Hajinezhad, A review of cooperative multi-agent deep reinforcement learning, Appl. Intell. (2022) 1–46.
[16] M. Zhou, Z. Liu, P. Sui, Y. Li, Y.Y. Chung, Learning implicit credit assignment for multi-agent actor-critic, arXiv preprint, arXiv:2007.02529.
[17] S. Sukhbaatar, R. Fergus, et al., Learning multiagent communication with backpropagation, Adv. Neural Inf. Process. Syst. 29 (2016) 2244–2252.
[18] S. Iqbal, F. Sha, Actor-attention-critic for multi-agent reinforcement learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 2961–2970.
[19] H. Mao, W. Liu, J. Hao, J. Luo, D. Li, Z. Zhang, J. Wang, Z. Xiao, Neighborhood cognition consistent multi-agent reinforcement learning, Proc. AAAI Conf. Artif. Intell. 34 (2020) 7219–7226.
[20] D. Simon, C.J. Snow, S.J. Read, The redux of cognitive consistency theories: evidence judgments by constraint satisfaction, J. Pers. Soc. Psychol. 86 (6) (2004) 814.
[21] P. Sunehag, G. Lever, A. Gruslys, W.M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J.Z. Leibo, K. Tuyls, et al., Value-decomposition networks for cooperative multi-agent learning, arXiv preprint, arXiv:1706.05296.

[22] T. Rashid, M. Samvelyan, C.S. De Witt, G. Farquhar, J. Foerster, S. Whiteson, Monotonic value function factorisation for deep multi-agent reinforcement learning, J. Mach. Learn. Res. 21 (1) (2020) 7234–7284.
[23] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, P. Cui, Structural deep clustering network, in: Proceedings of the Web Conference 2020, 2020, pp. 1400–1410.
[24] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, in: International Conference on Learning Representations, 2018.
[25] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, C. Zhang, Attributed graph clustering: a deep attentional embedding approach, in: International Joint Conference on Artificial Intelligence, 2019.
[26] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, Stat 1050 (2018) 4.
[27] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, Y. Gao, Multi-agent game abstraction via graph attention neural network, Proc. AAAI Conf. Artif. Intell. 34 (2020) 7211–7218.
[28] A. Malysheva, D. Kudenko, A. Shpilman, MAGNet: multi-agent graph network for deep multi-agent reinforcement learning, in: 2019 XVI International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY), IEEE, 2019, pp. 171–176.
[29] H. Ryu, H. Shin, J. Park, Multi-agent actor-critic with hierarchical graph attention network, Proc. AAAI Conf. Artif. Intell. 34 (2020) 7236–7243.
[30] R.S. Sutton, Temporal credit assignment in reinforcement learning, Ph.D. thesis, University of Massachusetts Amherst, 1984.
[31] G. Papoudakis, F. Christianos, A. Rahman, S.V. Albrecht, Dealing with non-stationarity in multi-agent deep reinforcement learning, arXiv preprint, arXiv: 1906.04737.
[32] P. Balister, B. Bollobás, A. Sarkar, M. Walters, Connectivity of random k-nearest-neighbour graphs, Adv. Appl. Probab. 37 (1) (2005) 1–24.
[33] V. Hautamaki, I. Karkkainen, P. Franti, Outlier Detection Using K-Nearest Neighbour Graph, Proceedings of the 17th International Conference on Pattern Recognition, 2004, ICPR 2004, vol. 3, IEEE, 2004, pp. 430–433.
[34] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.
[35] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: a review of methods and applications, AI Open 1 (2020) 57–81.
[36] Y. Cheng, Mean shift, mode seeking, and clustering, IEEE Trans. Pattern Anal. Mach. Intell. 17 (8) (1995) 790–799.
[37] D. Pelleg, A. Moore, Accelerating exact k-means algorithms with geometric reasoning, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 277–281.
[38] J. Jiang, Z. Lu, Learning attentional communication for multi-agent cooperation, in: Advances in Neural Information Processing Systems, 2018, pp. 7254–7264.
[39] M. Mesbahi, M. Egerstedt, Graph Theoretic Methods in Multiagent Networks, vol. 33, Princeton University Press, 2010.
[40] T. Balch, R.C. Arkin, Behavior-based formation control for multirobot teams, IEEE Trans. Robot. Autom. 14 (6) (1998) 926–939.
[41] V.G. Rao, D.S. Bernstein, Naive control of the double integrator, IEEE Control Syst. Mag. 21 (5) (2001) 86–97.
[42] A. Pajaziti, P. Avdullahu, SLAM–map building and navigation via ROS, Int. J. Intel. Syst. Appl. Eng. 2 (4) (2014) 71–75.
[43] T. Gu, J. Atwood, C. Dong, J.M. Dolan, J.-W. Lee, Tunable and stable real-time trajectory planning for urban autonomous driving, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2015, pp. 250–256.