Learning Communication for Cooperation in Dynamic Agent-Number Environment

Weiwei Liu, Shanqi Liu, Junjie Cao, Qi Wang, Xiaolei Lang, and Yong Liu

***Abstract*—The number of agents in many multiagent systems in the real world, such as storage robots and drone cluster systems, continually changes. Still, most current multiagent reinforcement learning (RL) algorithms are limited to fixed network dimensions, and prior knowledge is used to preset the number of agents in the training phase, which leads to a poor generalization of the algorithm. In addition, these algorithms use centralized training to solve the instability problem of multiagent systems. However, the centralized learning of large-scale multiagent RL algorithms will lead to an explosion of network dimensions, which in turn leads to very limited scalability of centralized learning algorithms. To solve these two difficulties, in this article propose a group centralized training and decentralized execution-unlimited dynamic agent-number network (GCTDE-UDAN). First, since we use the attention mechanism to select several leaders and establish a dynamic number of teams, and the UDAN performs a nonlinear combination of all agents' *Q* values when performing value decomposition, it is not affected by changes in the number of agents. Moreover, our algorithm can unite any agent to form a group and conduct centralized training within the group, avoiding network dimension explosion caused by the global centralized training of large-scale agents. Finally, we verified on the simulation and experimental platform that the algorithm can learn and perform cooperative behaviors in many dynamic multiagent environments.**

***Index Terms*—Attention mechanism, multiagent reinforcement learning (RL), multiagent system, recurrent neural network (RNN).**

I. Introduction

Although both model-based [1] and model-free [2] reinforcement learning (RL) has reached a human level of control in many complex single-agent tasks, such as Atari video games [3], Go games [4], and complex continuous control scenarios. However, most of the real environments are multiagent systems, agents change their strategies based on actions taken by other agents, so the multiagent environment is complex and dynamic, which brings great difficulties to the learning process [5]–[7].

One of the difficulties is that in the current multiagent RL algorithm, only a fixed number of agents can be trained [8]–[9], which creates a serious contradiction with the reality of the ever-changing number of agents in the real world, especially in a cooperative environment. Agents need to change their strategies according to changes in the environment continually. Some scholars have proposed some solutions to this problem to a certain extent. Peng *et al.* [8] proposed the BiCNet algorithm, which can handle different types of battles under different terrains, and both sides have different numbers of artificial intelligence (AI) agents during the battle. However, due to the recurrent neural network (RNN) network's characteristics, the number of agents has a fixed upper limit. Jiang *et al.* [10] use a graph convolutional neural network to deal with the problem of the uncertain number of neighbors of an agent. As the convolutional layer increases, the perceptual domain of each agent expands. However, the number of convolutional layers still needs to be set in advance. The number of agents that need to communicate cannot be dynamically changed according to environmental changes. Although different numbers of agents can use the aforementioned methods under a clear upper limit to complete the cooperation task, they did not solve learning cooperation with an unlimited number of agents.

Another difficulty of the multiagent system is that most of the current multiagent RL algorithms are centralized training and centralized execution (CTCE) or decentralized execution (CTDE), which makes each agent need the local information of all agents in the training phase to estimate the value function. As a result, these algorithms are complicated to extend to large-scale agent systems. For example, CommNet [11] needs to communicate between agents in the training and execution stages; that is, the network input is the local information of all agents, so this is a CTCE algorithm. Multi-Agent Deep Deterministic Policy Gradient (MADDPG) requires local observations and actions of all agents in the training stage. However, only each agent's local information is needed in the execution phase, namely the CTDE algorithm.

In order to solve the aforementioned problems, we proposed the unlimited dynamic agent-number network (UDAN), which can group centralized training and decentralized execution (GCTDE), we named this algorithm as GCTDE-UDAN.

Manuscript received December 22, 2020; revised February 22, 2021; accepted March 28, 2021. Date of publication April 27, 2021; date of current version August 13, 2021. Recommended by Technical Editor D. Shi and Senior Editor X. Chen. This work was supported in part by the National Key R&D Program of China under Grant 2017YFB1302003, and in part by the National Natural Science Foundation of China under Grant 61836015. *(Corresponding authors: Junjie Cao and Yong Liu)*

The authors are with the Advanced Perception on Robotics, and Intelligent Learning Lab, College of Control Science, and Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: 11932061@zju.edu.cn; 637lsq@gmail.com; cjunjie@zju.edu.cn; 1265251823@qq.com; jerry_locker@zju.edu.cn; yongliu@iipc.zju.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TMECH.2021.3076080.

Digital Object Identifier 10.1109/TMECH.2021.3076080

Our method is not affected by the increase in the number of agents, and can still consider the information of agents inside and outside the group to make more cooperative actions. The contributions of this article are as follows:

1) We use the attention mechanism to build dynamic number groups and give the attention network training method. The UDAN, effectively trained in the dynamic number of agents, is proposed, enabling agents to learn the communication protocol within the group and nonlinearly fit the Q value of each group in the number of dynamic agents to solve the problem of the agent's credit assignment.
2) The UDAN is GCTDE. Since this algorithm only needs the agent information in the group for group centralized training. Our method solves the problem that ordinary algorithms are difficult to expand as the number of agents grows.
3) We have carried out simulation and physical experiments in the MAgent environment and the real-world environment, and compared our methods with various multiagent RL algorithms. The result shows that our method achieves excellent performance. All experimental demonstrations can be viewed from the following link.[1]

II. Related Work

The research on cooperation and competition between multiagents has a long history [12],[13]. They are called random games, and RL has been a feasible method to promote cooperation between multiagents for a long time. However, as the multiagents environment's complexity continues to increase, these traditional methods are not effective. With the development of artificial neural networks in recent years, scholars have begun to pursue an end-to-end solution to multiagents problems, typically in deep RL [14]. In addition, the increasing number of multiagents and the complexity of the environment have brought about the necessity of communication [15] between agents.

Independent Q-Learning (IQL) [16] is a strategy that treats each agent as an independent individual; each learns its policy independently and treats other agents as part of the environment. Although IQL avoids the scalability problem of centralized training and has good results in some scenarios, it has caused environmental fluctuations since other agents are regarded as part of the environment. There is no proof of convergence. Under the global reward condition, the algorithm effect is feeble.

In order to solve the instability problem of multiagent RL, some scholars believe that agents should learn to communicate. Foerster *et al.* [17] are the first to introduce communication learning in deep multiagent RL, where each agent only has its partial observations. The article assumes that the communication channel is discrete; only discrete information can be transmitted between agents. Kim *et al.* [18] believe that the bandwidth of communication channels, in reality, is limited. If all agents send information to this narrow bandwidth channel, information loss or blockage will occur once the capacity is exceeded. Kim proposed SchedNet, which introduced the medium access control (MAC) method in the communication field into multiagent RL to solve this problem. The Deep Distributed Recurrent Q-Networks (DDRQN) [19] can solve the problem of communication and cooperation between multiple agents so that agents can reach a communication agreement from scratch.

[1][Online]. Available: https://youtu.be/xeFmfK9zgMU

Although the learning and communication between agents have achieved good results when each agent has an independent reward function, the DDRQN and SchedNet only use global rewards for learning and cannot distinguish whether each agent is working hard. Sunehag *et al.* proposed that the Value-Decomposition Networks (VDN) [5] perform value decomposition of global rewards to solve the agent's credit assignment problem. However, the VDN only performs a simple summation for the joint Q-value decomposition. The QMIX [9] algorithm believes that this approach will make the learned local Q function expression limited, and there is no way to capture the more complex interrelationships between agents. QMIX generalizes the joint Q function decomposition method to a larger family of monotonic functions. Also, QMIX believes that each agent only depends on local observations and may not estimate its local Q function accurately, so it introduces the global state as an auxiliary input. Counterfactual Multi-Agent Policy Gradients (COMA) [20] introduced a counterfactual baseline function. This method solves multiagent credit allocation by comparing the global reward obtained by the agent following the current strategy for decision making and the global reward obtained by following a certain default strategy.

In addition, due to restrictions on the bandwidth, large-scale agent communication is challenging. Recently, scholars have started to use the attention mechanism to enable agents to communicate in small areas. G2ANet [21] uses a graph attention neural network to extract the relationship between agents. Multi-Actor-Attention-Critic (MAAC) [22] learns a centralized critic with a soft-attention mechanism. The mechanism can dynamically select which agents to attend at each time step. However, these works can only be used in environments that have a fixed number of agents.

To expand the applicability of the algorithm, a few works consider training in an arbitrary-sized setting. The DGN [10] propose a graph structure to extract features from the scalable number of neighbors. Agarwal *et al.* [15] improve the scalability of the algorithm through course learning. However, for example, the DGN requires prior knowledge to set the number of communication layers of agents to expand the number of agents that can be communicated and inflexible in a dynamic number of multiagent systems. The algorithm proposed by Agarwal *et al.* is not an end-to-end solution to the problem of changes in the number of agents. Furthermore, these algorithms do not consider the plan of centralized training execution within the group, As a result, a large amount of bandwidth is required for communication during the training phase.

Since most multiagent RL algorithms are globally centralized training, all agents' information is required in the training phase, making it difficult for these algorithms to be extended to large-scale agent scenarios. In addition, unlike the algorithm mentioned previously applied to scenarios with a variable number of agents, the UDAN is an end-to-end RL algorithm that can be used to cooperate with unlimited dynamic agent number.

III. Background

Single-agent RL is described by the Markov decision process, while multiagent RL needs to be described by Markov game [23]. Among them, Markov means that the state of the multiagent system conforms to Markov, that is, the state of the next moment is only related to the current moment, and has no direct relationship with the previous moment. In this article, we focus on partially observable stochastic games (POSGs) [24], that is, each agent can only obtain part of the information in the environment.

The POSG can be described by a tuple $\{n, S, A_1, \ldots, A_n, T, \gamma, R_1, \ldots, R_n, O_1, \ldots, O_n\}$, where n is the number of agents, and the number of n in this article is constantly changing. S is the system state, that is, the joint state of each agent. A_i is the set of actions available to the agent i ($A = A_1 \times A_2 \times \cdots \times A_n$ is the joint action space), T is the state transition function, which refers to the probability distribution of the next state when the current state and joint behavior of the agent are given. which is: $S \times A_1 \times A_2 \times \cdots \times A_n \times S \to [0, 1]$. Discount factor $\gamma \in [0, 1)$. Its size indicates the importance of future returns in the value function. The larger the value, the more important the future returns. On the contrary, the agents pay more attention to the current returns. R_i is the reward function for the agent i, $S \times A_1 \times \cdots \times A_n \to R$. The algorithm in this article only uses the overall reward, that is, the sum of all agent rewards. At last, O_i is the observation set of the agent i.

For the agent i, the corresponding policy is $\pi_i : S \to \Omega(A_i)$, where $\Omega(A_i)$ is the collection of probability distributions over A_i. Each agent i, according to the current state S, chooses an action, or outputs an action distribution. The joint policy of all agents is $\pi \overset{\triangle}{=} [\pi_1, \ldots \pi_n]$. The state value function of the agent i: $v_\pi^i(s) = v_i(s; \pi) = \Sigma_{t=0}^{\infty} r^t E_{\pi,p}[r_i^t | s_0 = s, \pi]$. The state-action value function Q_π^i of the agent i: $S \times A_1 \times \cdots \times A_n \to R$. $Q_\pi^i = r(s, a) + \gamma E_{s' \sim p} v_\pi^i(s')$, where $a = [a_1, \ldots, a_n]$.

IV. Methods

The GCTDE-UDAN judges each agent every timestep T, whether it becomes a group leader, initiates communication with other agents, and selects which agents become group members to communicate. The communication group consists of a different number of agents according to different tasks, and the size of the communication group is also different. Although the communication group's size is fixed in the same task, the number of communication groups is constantly changing. Each communication group changes dynamically throughout the episode and only exists when needed. Furthermore, when multiple group leaders select an agent simultaneously, it will continue to participate in the information coding of different groups, and the code will be updated by itself, cyclically. It will work in all communication groups at the same time. At this time, this agent acts as a communication link in different communication groups.

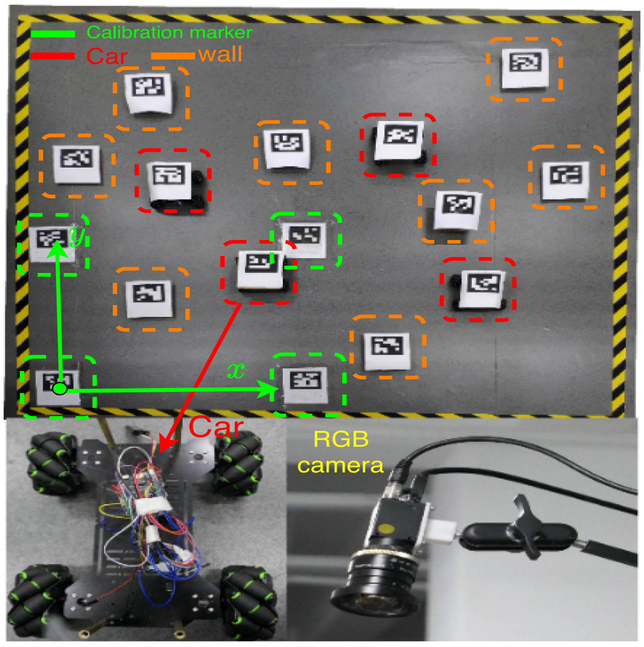


Fig. 1. Map of the experimental site and equipment, the map size is 2.8 × 2.8 m, rasterized to 9 × 9. Use the calibration marker to establish a coordinate system. The car uses mecanum wheels to enable it to move in any direction.

The GCTDE-UDAN's network structure is shown in Fig. 2, which includes an evaluation network, a communication channel, an attention unit [25], and a mixing network. First, the attention network takes the observation o_i of the agent i as the input to determine whether the agent i can become the group leader. Second, the group leader selects different group members to communicate in the communication channel, outputs communication information g_t^k, the agent obtains more comprehensive perception information, understands and infers other agents' behavior, and cooperates with other agents in decision making and mutual assistance.

A. Unlimited Dynamic Agent-Number Network (UDAN)

This chapter proposes a novel network UDAN that can communicate with any number of agents (not preset). In reality, the number of agents should change with changes in tasks and environments, which is very intuitive. Therefore, our algorithm randomly changes the number of agents in each training episode to adapt to the task change.

In previous algorithms, the preset network dimensions need to be consistent with all agents' splicing observation dimensions, the number of agents between training episodes is fixed. Also, some other algorithms set an upper limit on the number of agents. When the number of agents is lower than the upper limit, 0 will be used to fill the vector to ensure consistent dimensions. However, the upper limit of the agent's number for this task needs to be known in advance, and prior knowledge is required. Also, this may lose advanced structural information between

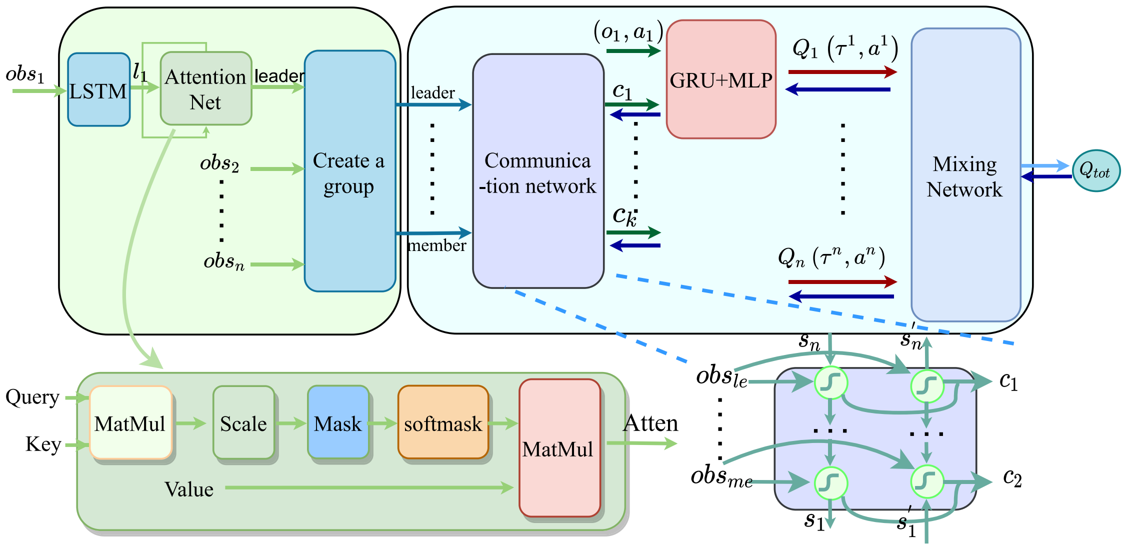


Fig. 2. GCTDE-UDAN architecture. The block diagram in the lower left corner is the attention mechanism. This article uses a self-attention network. The block diagram in the lower right corner is a bidirectional RNN. Among them, obs_{le} and obs_{me}represent the group leader and group members' observation values. s and s' are hidden status.

agents. Moreover, since in a large-scale agent environment, if the agent communicates globally, it will greatly reduce the communication efficiency and even cause the agent to move away from the task. The agent should focus on the surrounding agent's information situation. And in the process of large-scale agent training, to avoid the dimensional explosion of neural networks, the algorithm should adopt group centralized training. So, inspired by the aforementioned three points, we proposed UDAN.

1) Attention Net: Like on a football field, players generally pay more attention to where the ball is located and the player in possession. Therefore, the multiagent system agents should also focus on the agents that are closely related to themselves. We use the attention mechanism to select each group leader in the agents and build the group with the few agents closest to the group leader. We train the attention network by changing whether the communication team helps with the task.

Similar to [25], we use the self-attention mechanism. After the observation value of the agent i is processed by long short-term memory (LSTM) [26], that is, $l_i = L(o_i)$, Order: $\mathrm{Q(Query)} = \mathrm{K(Key)} = \mathrm{V(Value)} = l_i$. Calculate the dot product between Q and K, use the softmax operation to normalize the result to a probability distribution, and then, multiply it by the matrix V to get the weight sum as

$$\mathrm{Attention(Q, K, V)} = \mathrm{softmax}\left(\frac{\mathrm{QK^T}}{\sqrt{\mathrm{d_k}}}\right)\mathrm{V}. \quad (1)$$

Among them, the scale of $\sqrt{d_k}$ prevents the result of the aforementioned formula from being too large, and d_k is the dimension of a query and key vector. Finally, a fully connected layer is used to determine whether the agent i becomes the group leader and establish a group.

The group established here does not require prior knowledge, it does not need to be preset. Therefore, in each episode, each agent may become the group leader and establish a communication group. The communication groups change dynamically in an episode. However, cooperation requires a certain time step to be effective, so we set the group creation interval T. Limited by the communication bandwidth and communication distance, etc., the communication between agents in the real world is also changing dynamically, so this is consistent with the real world.

2) Communication Network: When the agent i becomes the group leader, he will select group members from the surrounding agents and establish a communication group. When multiple groups select an agent j simultaneously, this agent will become a bridge between different communication groups, allowing information to flow in different groups. For example, the agent i is a member of different groups, in the group P

$$\left(o^i_{G_t}, \ldots, o^j_{G_t}\right) = C\left(o^i_t, \ldots, o^j_t\right) \quad (2)$$

in the group Q

$$\left(o^i_{G_t}, \ldots, o^k_{G_t}\right) = C\left(o^i_{G_t}, \ldots, o^k_t\right). \quad (3)$$

Here, C represents the communication network. Equations (2) is equivalent to the agent i's information updated once in the P group. Then, it participates in the Q communication group, updates its own encoding again. The last updated encoding also affects the rest of the agent's encoding update in the Q group. The communication network here uses a bidirectional LSTM [27] unit. Unlike CommNet and BiCNet, which share information through arithmetic average and weighted average integration agents, the bidirectional LSTM unit can selectively output information that promotes cooperation, helping the agent make decisions based on understanding and predicting the actions of other agents.

3) Mixing Network: In a multiagent system, all agents share a global reward function. Once an agent learns some useful strategies earlier, the rest will choose lazier strategies, making the overall reward decline. To solve credit assignment among agents caused by all agents sharing a reward function, we introduce a mixing network from QMIX. The mixing network input is the local Q function of each agent, and the output is the global Q_{tot}. Since each agent only depends on local observations and may not accurately estimate its local Q function, QMIX needs to take the global state at each moment as an additional input to the mixing network. Unlike QMIX, the GCTDE-UDAN has integrated the relevant information of other agents at the bottom. It can accurately estimate the local Q function, so here we omit the global state.

B. Group Centralized Training and Decentralized Execution (GCTDE)

Although the joint action-value function can naturally deal with cooperation problems and avoid the nonstationarity in multiagent RL, as the number of agents grows, centralized training methods cannot adapt to large-scale agent systems. Therefore, the GCTDE is proposed, which divides the agents into groups, performs group centralized training within the group, and performs decentralized execution globally. First, in groups with high agent relevance, centralized training can solve the nonstationarity problem in the multiagent RL and promote cooperation between agents; second, since the number of groups for centralized training is limited, this avoids scalability problems caused by centralized training in a large-scale agent system.

The GCTDE-UDAN is an RL method based on value iteration. It initializes a cooperative task and has N agents, where N is random between each episode. The experience buffer $\mathcal{R}$ contains the tuples $(O_t, A_t, R_t, O_{t+1}, C_t)$, which represents the observations, actions, and rewards of all agents at time t and $t+1$. Among them, $O_t = (o_1^t, \ldots, o_N^t)$, $A_t = (a_1^t, \ldots, a_N^t)$, R_t, $O_{t+1} = (o_1^{t+1}, \ldots, o_N^{t+1})$, and C is an $N \times N$ matrix that records the communication groups. Here, o_i^t and o_i^{t+1} represent the observation of the agent i at moment t and $t+1$, respectively, a_i^t represents the action of the agent i at time t, and R_t represents global reward at time t.

The attention network is trained as a binary classifier to select the leader of each group. Use the communication information in the group as the auxiliary input of the local value network. At this time, the input of the local value network is the communication information o_G and the agent's own observation o, namely: $Q_c = Q(o_j, o_{G_i} | \theta^Q)$. Only the observation of the agent itself is used as the input of the local value network, namely: $Q_s = Q(o_j | \theta^Q)$. Calculate all groups' average local value of the difference ΔQ_i between Q_c and Q_s as

$$\Delta Q_i = \frac{1}{|G_i|} \left(\sum_{j \in g_i} Q\left(o_j, o_{G_i} | \theta^Q\right) - \sum_{j \in G_i} Q\left(o_j | \theta^Q\right) \right) \quad (4)$$

among them, j is the jth agent and i is the ith group. θ^Q is the parameter of the local value network.

The two-classifier network will finally go through a Sigmoid function and output a probability value. This probability value reflects the possibility of predicting the agent as the leader—the greater the probability value, the greater the possibility of becoming the leader. Define the output of the Sigmoid function to represent the probability that the current agent is the leader of the group as

$$p\left(o_i | \theta^p\right) = P\left(y = 1 \mid x\right). \quad (5)$$

Conversely, the probability of not being the group leader is

$$1 - p\left(o_i | \theta^p\right) = P\left(y = 0 \mid x\right). \quad (6)$$

From the maximum likelihood estimate, it can be derived as

$$P\left(y \mid x\right) = p^y\left(o_i | \theta^p\right) \cdot \left(1 - p\left(o_i | \theta^p\right)\right)^{1-y} \quad (7)$$

Then, introducing the log function gives

$$\begin{aligned} \log P(y|x) &= \log\left(p^y\left(o_i | \theta^p\right) \cdot \left(1 - p\left(o_i | \theta^p\right)\right)^{1-y}\right) \\ &= y \log p\left(o_i | \theta^p\right) + (1 - y)\log\left(1 - p\left(o_i | \theta^p\right)\right). \end{aligned} \quad (8)$$

Substituting $y = \Delta\hat{Q}_i$ and two classification network uses cross entropy loss function, the result is:

$$\mathcal{L}(\theta^p) = -\Delta\hat{Q}_i \log(p\left(o_i | \theta^p\right)) - (1 - \Delta\hat{Q}_i) \log\left(1 - p\left(o_i | \theta^p\right)\right) \quad (9)$$

where $\Delta\hat{Q}_i$ is the min–max normalized ΔQ_i, $\Delta\hat{Q}_i \in [0, 1]$. θ^p is the parameters of the attention network.

The interaction between the agent and the environment can be modeled as a Markov decision process, the current state of the agent is

$$s_t = f(\mathbf{o}_1, r_1, \mathbf{a}_1, \ldots, \mathbf{a}_{t-1}, \mathbf{o}_t, r_t) \quad (10)$$

estimate the next state(s_{t+1}) as

$$P(s_{t+1} | s_t) = P(s_{t+1} | s_1, \ldots, s_t). \quad (11)$$

The agent can be rewarded every time it interacts with the environment. The long-term return of the agent is

$$G_t = r_{t+1} + r_{t+2} + \cdots = \sum_{k=0}^{\infty} \lambda^k r_{t+k+1} \quad (12)$$

where λ is a discount factor, use this long-term return to measure the pros and cons of the agent's strategy.

According to the current state of the agent, use the state value function to estimate this future return as follows:

$$V(s) = \mathbb{E}[G_t | S_t = s]. \quad (13)$$

Similarly, the state-action value function is as follows:

$$Q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]. \quad (14)$$

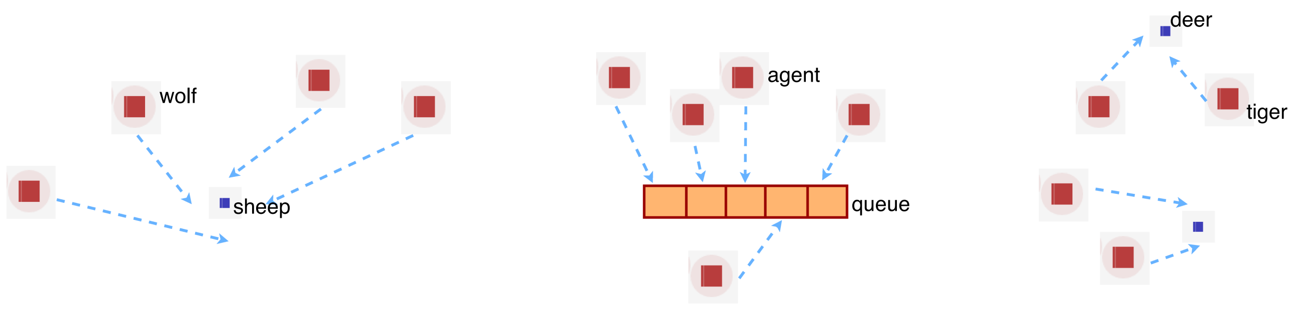


Fig. 3. Simulation scenarios. (Left) Cooperative pursuit. (Mid) Cooperative queue. (Right) Cooperative tiger.

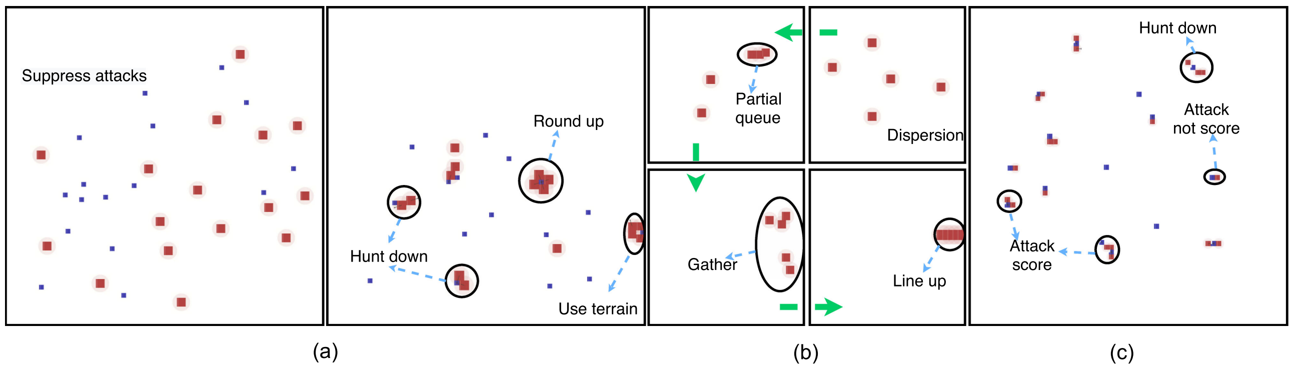


Fig. 4. Illustration of multiagents learning to cooperate in three scenarios. (a) Cooperative pursuit. (b) Cooperative queue. (c) Cooperative tiger.

Derivation of the Bellman equation give the state value function as

$$\begin{aligned} V(s) &= \mathbb{E}[G_t|S_t = s] \\ &= \mathbb{E}[r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \cdots |S_t = s] \\ &= \mathbb{E}[r_{t+1} + \lambda(r_{t+2} + \lambda r_{t+3} + \cdots)|S_t = s] \\ &= \mathbb{E}[r_{t+1} + \lambda G_{t+1}|S_t = s] \\ &= \mathbb{E}[r_{t+1} + \lambda V(s_{t+1})|S_t = s]. \end{aligned} \tag{15}$$

Similarly, the state-action value function is

$$\begin{aligned} Q(s,a) &= \mathbb{E}[G_t|S_t = s, A_t = a] \\ &= \mathbb{E}[r_{t+1} + \lambda Q(s_{t+1}, a_{t+1})|S_t = s, A_t = a]. \end{aligned} \tag{16}$$

When $\pi(a|s)$ is deterministic policy

$$V(s) = \max_{a \in A} Q_\pi(s,a). \tag{17}$$

The value function is an evaluation of the strategy π as

$$\forall s, \pi^* = \underset{\pi}{\operatorname{argmax}} V^\pi(s). \tag{18}$$

The GCTDE-UDAN is trained as follows:

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[\left(y_i^{\text{tot}} - Q_{\text{tot}}(\mathbf{o}, \mathbf{a}; \theta) \right)^2 \right] \tag{19}$$

where b is the batch size of transitions sampled from the replay buffer. The target value function is

$$y^{tot} = r + \gamma \max_{\mathbf{a}'} Q_{tot}(\boldsymbol{\tau}',\mathbf{a}'; \theta^-) \tag{20}$$

where θ^- is the parameter of the target value network. According to QMIX, $Q_{\text{tot}}(\mathbf{o}, \mathbf{a})$ is

$$\underset{\boldsymbol{\tau}}{\operatorname{argmax}} \mathrm{Q}_{\text{tot}}(\boldsymbol{o}, \mathbf{a}) = \begin{pmatrix} \operatorname{argmax}_{\mathrm{a}_1} \mathrm{Q}_1(\mathrm{o}_1, \mathrm{a}_1) \\ \vdots \\ \operatorname{argmax}_{\mathrm{a}_n} \mathrm{Q}_n(\mathrm{o}_n, \mathrm{a}_n). \end{pmatrix} \tag{21}$$

where $Q_i(o_1, a_1) = Q_c^i = Q(o_t^i, o_{G_t}^i)$, Q is local value network, and $(o_{G_t}^i, \ldots, o_{G_t}^j) = C(o_t^i, \ldots, o_t^j)$.

V. Simulations and Experiments

To verify the algorithm's effect in the multiagent cooperative environment, we design both simulation and physical experiments. The simulation experiment is based on MAgent [28] to enable multiple agents to round up, maintain a line, and contact prey together, as shown in Fig. 3. The physical experiment uses multiple vehicles to round up or look for water over obstacles. The specific situation is shown in Fig. 6. In the same scenario, all agents share a reward function. Since the number of agents needs to be constantly changing during the test phase of the following experiment, and all agents share a global reward, which requires credit assignment, only the algorithm that is both CTDE and value decomposition can be used as a benchmark here. The experimental baselines are QMIX, VDN [5], DQN [29], and COMA [20].

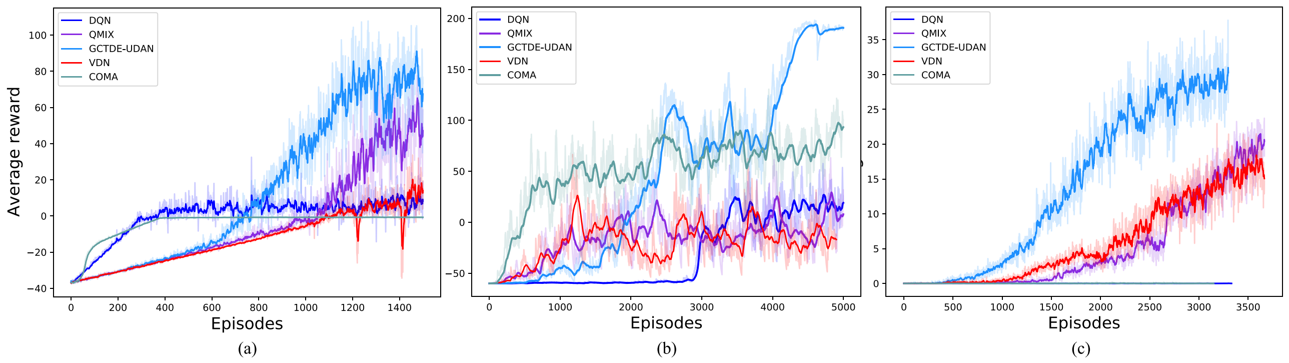


Fig. 5. Average reward's comparison diagram of the GCTDE-UDAN and baseline algorithms in three scenarios. (a) Cooperative pursuit. (b) Cooperative queue. (c) Cooperative tiger.

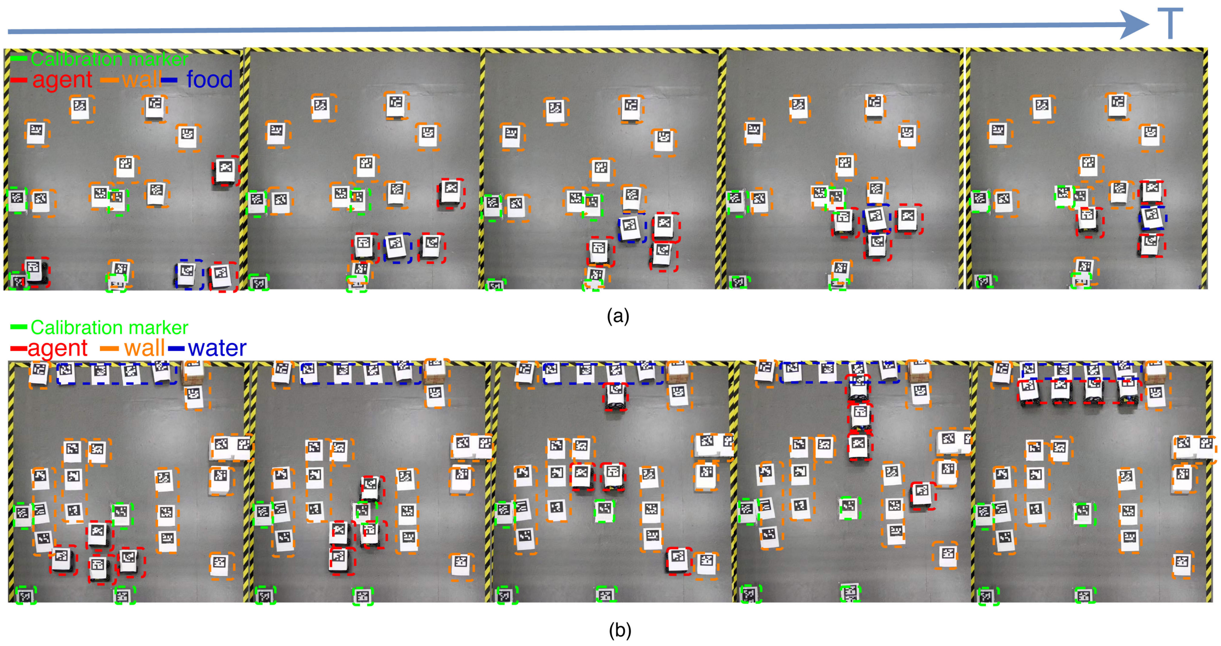


Fig. 6. Illustration of multiagents learn cooperative in two real-world scenarios. (a) Surrounded (b) Find-water.

A. Magent

Magent is a huge grid world. Agents are controlled by groups. In each step, each agent can choose to move to the surrounding grid or attack the enemy. Each agent has local information around its cell, including ID embedding, last action, last reward, and normalized position.

1) Cooperative Pursuit: In this scenario, wolves round up sheep. Each wolf will get 1 point after attacking the sheep once, and the sheep will lose 1 point. However, wolves move slower than sheep, so wolves need to learn to surround a sheep so that it cannot move to keep scoring. Regardless of whether the wolf attacks the sheep, as long as the wolf attacks, the wolf will lose -0.2 points. This is to reduce the number of useless attacks by wolves and to surround the sheep as soon as possible. Each episode is 300 steps. $\mathrm{mapsize} = 100$. The specific settings are shown in Table I. In this scene, the number of each wolf and sheep changes simultaneously, which aligns with nature's laws. Although the algorithm can be used for unlimited agent training, it is limited to computing speed and memory size. Here, the number of wolves and sheep is set to vary between 8 and 20. Since the baseline algorithms cannot adapt to the change in the number of agents during training, the number of agents during the training period is set to 10. Four wolves can catch a sheep, therefore, each team leader can form a four-person team.

TABLE I
COOPERATIVE PURSUIT SCENARIO SETTING PARAMETERS

	view range	attack range	speed	attack penalty
wolf	5	2	1	-0.2
sheep	4	0	1.5	0

In Fig. 5(a), the average reward is each agent's reward in each episode. Both the COMA and DQN algorithms have achieved large rewards in the initial stage, and the rate of rewards has

TABLE II
AVERAGE REWARD DURING TRAINING PHASE WITH DIFFERENT NUMBER OF AGENTS

Nums		8-20		
Rewards	***GCTDE-UDAN***	***71***		
Nums		10	15	20
Rewards	DQN	8.5	7.91	6.26
	COMA	-0.71	-0.68	-0.73
	VDN	7.5	14.32	3.03
	QMIX	43.98	46.18	34.35

TABLE III
COOPERATIVE TIGER SCENARIO SETTING PARAMETERS

	view range	attack range	speed	step recover	hp
tiger	4	1	1	0	8
deer	1	0	1	0.2	5

increased rapidly. This is because these two algorithms suppress the attack of the agent and obtain rewards. Still, as the training progresses, this algorithm makes the wolves lose the ability to attack. Therefore, even when the wolves encounter sheep, they will not attack, and its rewards are stable at around 0. As shown in Fig. 4(a)-left, the agent suppresses attacks and has not learned to cooperate. However, other algorithms rarely suppress attacks, and a large part of their reward boost comes from attacks on sheep. Since the wolves did not learn how to track and round up sheep at the beginning of training, their rewards increase very slowly. As the training progresses, the cooperation ability between the wolves strengthens, which can trap the sheep, continuously attack and score, and the rewards will also increase quickly. As shown in Fig. 4(a)-right, the agents learn to track, round up, and use terrain. Comparing these benchmark algorithms, as the GCTDE-UDAN has unlimited changes in the number of agents in the training phase, its performance is not affected by changes in the number of agents during testing. That is, it has a good generalization ability to get greater rewards. Moreover, due to the added communication function, it considers related agents' information when constructing each agent's Q value, making the agents more inclined to cooperate.

Nums in Table II represents the number of agents in the training phase, but in the testing phase, the number of agents in each episode is randomly generated between 8 and 20. Rewards represent the average return of five experiments. As can be seen from the table, as the number of agents in the training phase increases, the COMA and DQN algorithm rewards will not change much. It is because such algorithms suppress the attacks of wolves and make them unable to score. However, unlike intuition, the reward of the QMIX algorithm first rises, and then, falls. This is because 15 agents take into account the situation encountered between 8 and 20 agents. Since the number of agents in the GCTDE-UDAN is dynamic in the training phase, its generalization ability is more robust than these baselines. Its average reward for each agent reaches 71.

2) Cooperative Queue: In this scenario, the agents need to be arranged in a line, as shown in Fig. 3(b). The agent has a field of view of 5 and a speed of 1. To complete the task as soon as possible, arrange in a team, the agent will lose -0.2 points every step. All agents line up in a line and get reward 1. $\mathrm{mapsize} = 30$. Each episode is 300 steps.

As shown in Fig. 5(b), the algorithm fluctuates significantly in the *Cooperative Queue* scenario. The reason is that compared with the *Cooperative Pursuit* task, this task is challenging to transfer knowledge between different numbers of agents. For example, the strategy in the three-agent scenario is challenging to use in the five-agent scenario. Also, under this task, the COMA algorithm has achieved good results. On the contrary, DQN and other benchmark algorithms perform poorly. This is because this task is entirely dependent on the cooperation between all agents. No agent can be lazy and not work; otherwise, all agents will lose their rewards. The COMA algorithm mainly solves the multiagent credit assignment problem in partially observable Markov decision. This is consistent with the *Cooperative Queue* task.

Compared with the benchmark algorithms, the GCTDE-UDAN still achieves state-of-the-art. The reason is that, on the one hand, we have achieved intragroup communication, which is essential for fully cooperative tasks. On the other hand, similar to the *Cooperative Pursuit* of tasks, we learn to cooperate in a dynamic number of agents during the training process. When the number of agents in the training phase is different, the knowledge transfer speed will be faster.

3) Cooperative Tiger: As shown in Fig. 3(c), if two tigers attack a deer at the same time, both tigers get 1 point. Otherwise, no points are scored. Each episode is 200 steps. $\mathrm{mapsize} = 50$. Tiger's attack damage is 1. Other parameters are shown in Table III.

In this scenario, because a single tiger attacks the deer, no points are scored, but the deer will lose blood, and the overall score will drop. Therefore, the UDAN algorithm learned that even a single tiger would not attack even if it is near the deer. He will follow the deer and call another tiger to cooperate with the attack to get rewards. At every step, the deer recovers its blood, the tiger even learns to stock the deer. For example, after attacking for a while, stop the attack to restore the deer's blood to a healthy level, and then, attack again. As shown in Fig. 5(c), the number of interactions is as high as 5000 episodes. This is because, at the beginning of the training, the tiger can easily wipe out all the deer without learning to cooperate, so there is no return. Also, the DQN is challenging to decompose all rewards and cannot solve the problem of credit distribution. agents need to cooperate in pairs, this does not match COMA's view of agents as cooperation between all agents to decompose rewards. Finally, the intragroup communication of the UDAN algorithm and the powerful generalization ability in multiple scenarios make it far better than QMIX and VDN.

B. Physical Experiments

The physical experiment platform is shown in Fig. 1, the camera and router parameters are shown in Table IV. We use the Mecanum wheel car as an agent for experimentation to ensure that the car can move in four directions. The experimental platform's data flow process is as follows: The high-altitude camera determines the coordinates of the car and obstacles and sends

TABLE IV
CAMERA AND ROUTER PARAMETER TABLE

Camera Model	Effective Pixels	Image Resolution	Frame Rate
MindVision MV-GE231GM-T	2300000	1920X1200	40FPS
Router Model	Frequency Range	Transmission Rate	Ransmission Protocol
Huawei WS7200	2.4GHz&5GHz	2.4GHz: 574Mbps, 5GHz: 2402Mbps	802.3, 802.3u, 802.3ab

them to the local computer. The UDAN and baseline algorithms calculate agents' next action (up, down, left, right, and waiting) according to the coordinate information, and finally, sends the next instruction to the car through the router. After obtaining the specified message through ESP8266-WIFI on the car, it is sent to the serial port module to determine the car's direction and the angle of the body. The network delay time is within 10 ms, communication and other functions are realized by the Robot Operating System (ROS) platform, and the underlying control algorithm of the car is written in C++. It should be noted that although the camera obtains the global map information, in the local computer, through the masking algorithm, each vehicle has only a part of the field of view. The GCTDE-UDAN only makes decisions based on the part of the field of view around each car, that is, the partially observable Markov decision process.

In the following experiment, the number of obstacles varies between 0 and 10 to test the cooperative ability of the agent in different terrains. As the GCTDE-UDAN algorithm can adapt to different numbers of agents in training, for the same task, the following experimental results are obtained from one training session.

1) Surrounded: Like the *Cooperative Pursuit* task, the agents surround the goal to get the team's maximum score. Each agent appears in the adjacent grid of the goal to get a score. However, the agents will cooperate to surround the target and lose its mobility to obtain the maximum return because the target will move. As shown in Fig. 6(a), the agent moves in the grid map and needs to avoid obstacles, keep approaching the target, track the target, and even use obstacles to surround the target for maximum return. The moving speed of the agent and the target is 1, and the field of view is 3 and 1. Each episode is 50 steps.

Table V shows the average score of each agent under different numbers of agents and obstacles. As shown in Table V, when the number of obstacles remains the same, as the number of agents increases, the agents' average score will decrease. This is because the speed of the agent and the target are the same, even if it cannot form an effective envelope of the target, the agent can still track the target score. When the number of agents does not change, the number of obstacles increases, the agents' average score first increases, and then, decreases. As the number of obstacles is small, the agent can use obstacles to track and surround the target. However, too many obstacles will affect the agent's action space, making it difficult to track the target. Second, when the number of obstacles is small, the score when the number of agents is four is higher than when the number is three, because four agents can surround the target without the advantage of terrain. However, when the number of obstacles is large, for example, seven or eight, the agent learns to use the terrain. At this time, four agents are not needed to surround the target, and the extra agent may have nothing to do, resulting in a

TABLE V
SCORE OF THE AGENT AND THE TARGET UNDER THE SURROUNDED TASK

Agent nums / Wall nums	1	2	3	4
	Average agent score			
0	29.329	28.277	26.937	28.677
1	30.077	29.502	27.844	29.727
2	30.279	29.620	28.025	28.959
3	30.101	29.434	27.841	28.182
4	29.634	29.128	27.756	26.640
5	29.305	29.004	27.288	26.164
6	29.095	28.069	27.207	25.370
7	28.585	27.770	26.798	25.147
8	27.691	27.377	26.565	24.527
9	28.194	27.530	26.238	25.080
	Goal score			
0	-36.838	-35.369	-34.354	-34.289
1	-36.873	-36.045	-34.806	-34.907
2	-38.788	-37.626	-35.659	-36.771
3	-37.999	-37.547	-36.698	-36.281
4	-38.390	-37.578	-35.942	-34.514
5	-37.238	-36.996	-35.842	-33.820
6	-37.364	-36.697	-34.796	-32.854
7	-36.974	-35.799	-34.545	-32.430
8	-35.447	-35.290	-34.319	-32.231
9	-37.737	-35.367	-34.150	-32.514

decrease in the average score. The lower part of Table V shows the average loss score of each agent.

The experimental results are shown in Fig. 7(top). Compared with other baseline algorithms, the UDAN scores higher returns, and the difference in returns between different numbers of agents is also more negligible. This is because the UDAN has acquired cooperative knowledge of different numbers of agents during the training phase. In other baseline algorithms, the number of agents in the training phase is set to 3. That is, only three agents' knowledge of hunting is learned. However, when the number of agents is also set to 3 during the test, UDAN's score is still significantly highest. This shows that the UDAN performance is higher than the baseline algorithm even without considering the generalization ability of the UDAN. Fig. 7(below) shows the average loss score of each agent. When the agent is not in the adjacent grid of the target, it will lose 1 point every four steps. This is to punish the agent for not taking effective measures to complete the task as soon as possible. Therefore, the target loss situation more intuitively reflects the pros and cons of the agent's behavior.

2) Find Water: As shown in Fig. 6(b), in the task of finding the water source, the map is divided into three parts according to the y-axis, which are the agent's starting area, the obstacle area, and the water source area. The agent needs to avoid obstacles, and reach the water source area from any position in the departure area. The number of water sources is the same as the number of agents. Due to the limited number of water sources, agents need to cooperate and keep in formation to make each agent drink

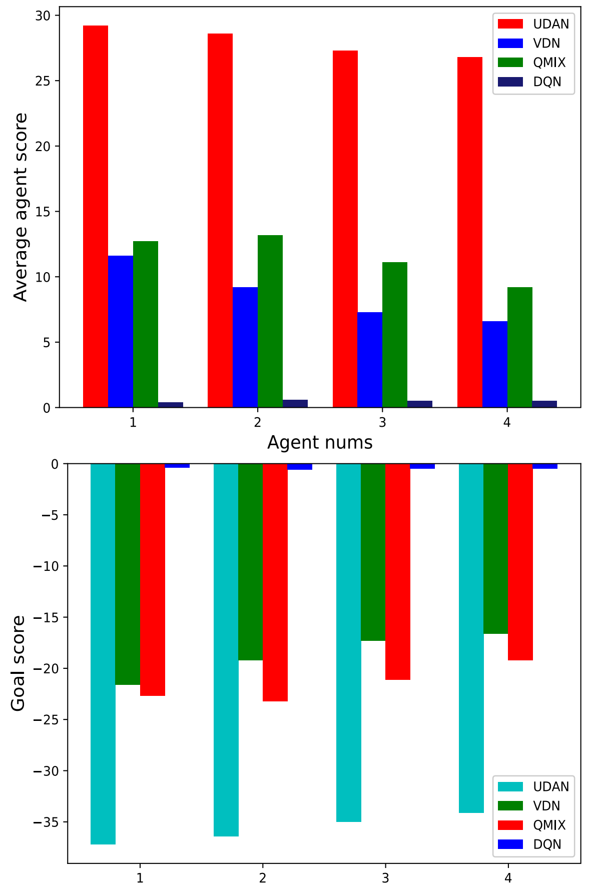


Fig. 7. Score of the agent and the target under the surrounded task.

TABLE VI
SCORE OF THE AGENT UNDER THE FIND-WATER TASK

Agent nums / Wall nums	1	2	3	4	5
	Average agent score				
0	42.0	41.5	41.954	42.220	41.749
1	42.0	41.413	41.850	41.990	41.548
2	41.904	41.404	41.760	41.828	41.569
3	41.922	41.47	41.532	41.539	41.532
4	41.647	41.25	41.365	41.45	41.132
5	41.68	41.30	41.085	41.137	40.710
6	41.137	40.962	40.872	40.466	40.426
7	41.68	38.627	40.908	40.43	39.875
8	40.865	39.71	40.071	39.03	39.708
9	40.0	40.127	38.960	39.038	38.364
10	40.490	39.343	39.933	36.53	38.145
11	37.865	38.0	35.648	36.615	37.180
12	38.491	37.27	35.660	35.135	36.345
13	37.314	35.598	37.679	34.647	35.944
14	32.588	32.63	32.549	30.656	32.992

water and get the maximum return for the team. In this task, each agent has a field of view of 3. In each episode, the number of obstacles, agents, and water sources are randomly generated. If the agent is in the grid adjacent to the water source, it is deemed to have obtained the water source and will be rewarded 1. Each episode is 50 steps.

As shown in Table VI, similar to the surrounded task, when the number of agents remains the same, as the number of obstacles grows, the average reward of the agents continues to decrease. However, when the number of obstacles changes within a small range, it has little effect on the agent's average reward. When the number of obstacles is small, there are still many roads from the departure area to the obstacle area. However, when the number of obstacles is too large, the agent's average reward will drop rapidly. Due to the limited area of the map and many obstacles, the road will be blocked, and only a few agents or even no agents can pass, which significantly hinders the agent from passing through the obstacle area. As shown in Fig. 6(b), both ends and the back of the water source in the water source area are surrounded, and only the vacant seats with the same number of front and agent can drink water. In order to let all agents drink water, agents need to line up tightly. Therefore, with the increase in the number of agents, it is more difficult for agents to form up, and team returns have decreased. Finally, in general, the UDAN algorithm obtained the highest return of 42 in the task scenario with the maximum average reward of 50, indicating that the algorithm performed exceptionally well.

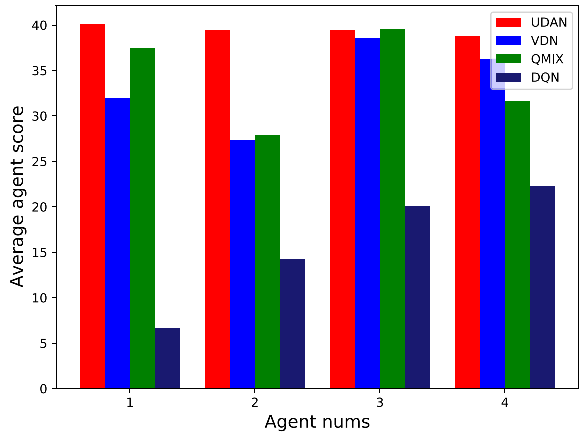


Fig. 8. Score of the agent under the find-water task.

As shown in Fig. 8, unlike the surround task, the increase in the number of agents has no consistent effect on the experimental results. First of all, from the whole picture, because the baseline algorithm only learns the cooperative knowledge of three agents, the algorithm's generalization ability is feeble, and its score fluctuates wildly. On the contrary, the UDAN has all the cooperative knowledge, the algorithm generalization ability is powerful, and its score has almost no fluctuation. Second, only in terms of the three agents' cooperation, UDAN's score is almost the same as the best-performing QMIX algorithm. UDAN has the best performance while maintaining the stability of cooperation between different numbers of agents.

VI. CONCLUSION

This article addresses the problem that CTCE and CTDE algorithms cannot be extended to large-scale agent scenarios. An algorithm for the centralized training within the group and global decentralized execution is proposed, which greatly reduces the cost of the centralized training in large-scale agent scenarios. Second, since the number of agents in a multiagent system is continually changing in reality, the proposed algorithm can dynamically adapt to changes in the number of agents. Finally, this article verifies the GCTDE-UDAN and other benchmark algorithms in three simulation tasks and two physical experiments. The GCTDE-UDAN algorithm performs exceptionally well in all environments.

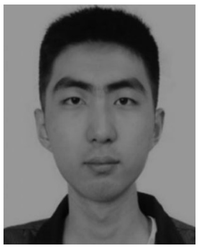
Although the GCTDE-UDAN algorithm we have proposed can adapt to the task of changing the number of global agents, the number of agents communicating in the group needs to be preset with prior knowledge. Therefore, in future work, we hope that each agent can establish groups with different numbers of members to communicate according to its state.

References

[1] V. Mnih*et al.* "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[2] D. Silver*et al.* "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[3] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2829–2838.
[4] S. Bansal, R. Calandra, K. Chua, S. Levine, and C. Tomlin, "Mbmf: Model-based priors for model-free reinforcement learning," 2017, *arXiv:1709.03153*.
[5] P. Sunehag *et al.* "Value-decomposition networks for cooperative multi-agent learning based on team reward.," in *Proc. 17th Int. Conf. Auton. Agents MultiAgent*, 2018, pp. 2085–2087.
[6] L. Busoniu, R. Babuska, and B. De Schutter, "Multi-agent reinforcement learning: A survey," in *Proc. 9th Int. Conf. Control, Automat., Robot. Vis.*, 2006, pp. 1–6.
[7] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," 2019, *arXiv:1906.04737*
[8] P. Peng *et al.*, "Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games," in Workshop on Neural Information Processing Systems(NIPS), 2017.
[9] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in International Conference on Machine Learning (ICML), pp. 4295–4304, *PMLR*, 2018.
[10] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," in International Conference on Learning Representations (ICLR), 2018.
[11] S. Sukhbaatar *et al.* "Learning multiagent communication with backpropagation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2244–2252.
[12] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 1994, pp. 157–163.
[13] J. Schmidhuber, "A general method for multi-agent reinforcement learning in unrestricted environments," in *Proc. AAAI Spring Symp. Adaptation, Coevolution Learn. Multiagent Systems*, 1996, pp. 84–87.
[14] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in International Conference on Learning Representations (ICLR), 2016.
[15] A. Agarwal, S. Kumar, K. Sycara, and M. Lewis, "Learning transferable cooperative behavior in multi-agent team," in International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 1741–1743, 2020.
[16] T. Ardi *et al.*, "Multiagent cooperation and competition with deep reinforcement learning," *Plos One*, vol. 12, no. 4, 2017, Art. no. e0 172395.
[17] J. N. Foerster, Y. M. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in Neural Information Processing Systems (NIPS), pp. 2145–2153, 2016.
[18] D. Kim *et al.*, "Learning to schedule communication in multi-agent reinforcement learning," in International Conference on Representation Learning (ICLR), 2019.
[19] J. N. Foerster2, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate to solve riddles with deep distributed recurrent q-networks," 2016, *arXiv:1602.02672*.
[20] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *32nd Proc. AAAI Conf. Artif. Intell.*, pp. 2974–2982, 2018.
[21] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, pp. 7211–7218.
[22] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.
[23] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. 11th Int. Conf. Mach. Learn. Proc.*1994, pp. 157–163.
[24] E. A. Hansen, D. S. Bernstein, and S. Zilberstein, "Dynamic programming for partially observable stochastic games," *in Proc. 19th Nat. Conf. Artif. Intell.*, 2004, vol. 4, pp. 709–715.
[25] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[26] S. Yan, "Understanding LSTM networks," vol. 11, Aug. 2015.
[27] P. Zhou *et al.*, "Attention-based bidirectional long short-term memory networks for relation classification," in *Proc. 54th Annu. Meeting Assoc.Comput. Linguistics*, 2016, pp. 207–212.
[28] L. Zheng *et al.*, "MAgent: A many-agent reinforcement learning platform for artificial collective intelligence," in *Proc. 32nd AAAI Conf. Artif. Intell.*, pp. 8222–8223, 2018.
[29] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," in Deep Learning Workshop on Neural Information Processing Systems (NIPS) 2013.

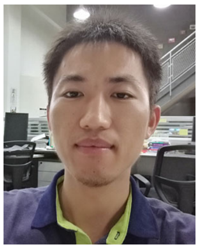
Weiwei Liu is currently working toward the Ph.D. degree at the Advanced Perception on Robotics and Intelligent Learning Lab, the College of Control Science and Engineering, Zhejiang University.

At present, his main research directions are artificial intelligence, reinforcement learning and robotics.



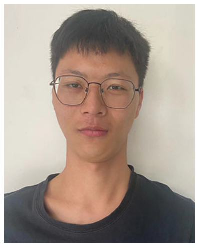
Shanqi Liu received the B.S. degree in control science and engineering, in 2019, from Zhejiang University, Hangzhou, China, where he is currently working toward the Ph.D. degree with the institute of Cyber Systems and Control, Department of Control Science and Engineering.

His research interests include reinforcement learning and robotics.



Junjie Cao received the B.S. degree in mechanical engineering and automation from Nanjing Tech University, Nanjing, China, in 2014, and the M.S. degree in mechanical engineering (Mechatronics), in 2017, from Zhejiang University, Zhejiang, China, where he is currently working toward the Ph.D. degree with the College of Control Science and Engineering.

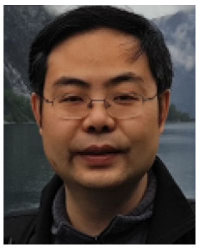
His current research interests include machine learning, sequential decision making, and robotics.



Qi Wang received the B.S. degrees in information engineering from Hangzhou Dianzi University, Hangzhou, China, in 2020.

He is currently a postgraduate student studying in Zhejiang University.

XiaoLei Lang received the B.S. degree in electrical engineering from the Zhejiang University of Technology, Hangzhou, China, in 2020. He is currently working toward M.S. degree at the College of Control Science and Engineering, Zhejiang University.



Yong Liu (Member, IEEE) received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2001 and 2007, respectively.

He is currently a Professor with the Department of Control Science and Engineering, Institute of Cyber Systems and Control, Zhejiang University. He has authored and coauthored more than 30 research papers in machine learning, computer vision, information fusion, and robotics. His current research interests include machine learning, robotics vision, information processing, and granular computing.