Received May 27, 2020, accepted June 15, 2020, date of publication June 29, 2020, date of current version July 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3005734

Strategic Interaction Multi-Agent Deep Reinforcement Learning

WENHONG ZHOU, JIE LI, YITING CHEN, AND LIN-CHENG SHEN, (Member, IEEE)
College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China

Corresponding author: Jie Li (lijie09@nudt.edu.cn)

ABSTRACT Despite the proliferation of multi-agent deep reinforcement learning (MADRL), most existing typical methods do not scale well to the dynamics of agent populations. And as the population increases, the dimensional explosion of joint state-action and the complex interaction between agents make learning extremely cumbersome, which poses the scalability challenge for MADRL. This paper focuses on the scalability issue of MADRL with homogeneous agents. In a natural population, local interaction is a more feasible mode of interplay rather than global interaction. And inspired by the strategic interaction model in economics, we decompose the value function of each agent into the sum of the expected cumulative rewards of the interaction between the agent and each neighbor. This novel value function is decentralized and decomposable, which enables it to scale well to the dynamic changes in the number of large-scale agents. Hereby, the corresponding strategic interaction reinforcement learning algorithm (SIQ), is proposed to learn the optimal policy of each agent, wherein a neural network is employed to estimate the expected cumulative reward for the interaction between the agent and one of its neighbors. We test the validity of the proposed method in a mixed cooperative-competitive confrontation game through numerical experiments. Furthermore, the scalability comparison experiments illustrate that the scalability of the SIQ algorithm outperforms the independent learning and mean field reinforcement learning algorithms in multiple scenarios with different and dynamically changing numbers.

INDEX TERMS Multi-agent deep reinforcement learning, scalability, local interaction, large scale.

I. INTRODUCTION

Multi-agent reinforcement learning (MARL) sophisticatedly combines the game theory, multi-agent system and reinforcement learning (RL). Recently, with the introduction of deep neural networks, the performances of multi-agent deep reinforcement learning (MADRL) in many tasks are impressive, such as learning communication protocol [1], [2], playing computer games [3]–[5], motion planning and control [6]–[8] and trajectory prediction [9], [10].

The research scope of MADRL includes training schemes [11], collaborative relationships [12], multi-task learning [13] and scalability [14], [15]. Here, the scalability is defined as the ability to adapt to dynamic changes in the number of agents during training and execution while ensuring effective interactions. It is a basic capability required by MADRL (or MARL) in practical applications where a large number of agents may be involved, or the number may change dynamically due to possible contingencies and confrontations.

In MARL, agents not only interact with the environment, but also interact with each other. The interactions of agents are bidirectional, such that their action decision processes are coupled: each agent is affected by the others, and vice versa [11]. Some works directly or indirectly integrate the states of all agents into a joint state to make a joint action decision [16], [17] for all agent. Although this method takes all the interactions into consideration, it would cause dimension explosion since the joint state-action space exponentially increases with the growth in the number of agents, as well as their potential interactions. Moreover, when the number of agents changes (increases or decreases), their joint dimensions also change accordingly, but these methods do not support dynamic changes in the input and output dimensions. So the learned policy can only be applied to the scenarios that the number of agents is identical to that of the training one.

As a result, most methods for MADRL are just appropriate for the scenarios with a small and fixed number of agents,

The associate editor coordinating the review of this manuscript and approving it for publication was Chenguang Yang.

and the scalability is a bottleneck that MADRL needs to address. This work tries to explore a solution to the scalability problem of MADRL, which not only enables MADRL to actively adapt to dynamic changes in the number of agents, but also enables the learned policy to be applied to scenarios with abundant agents. Especially, We restrict the agents to be homogeneous and partially observable, the observation and communication scopes are limited, so each agent cannot obtain perfect global information except its own private state, local observation or communication. And, each agent's policy should be irrelevant with the permutation of its neighbors.

A. RELATED WORKS

Although several works have been published recently summarizing the development and current status of MADRL, including MADDPG [11], QMIX [17],COMA [18], value-decomposition networks [19], QTran [20], there is little research on the scalability of MADRL in scenarios with a large number of agents, so here we summarize and compare the existing scalable methods from the perspective of whether to consider the interaction of agents, while some other MARL methods are detailed in [21]–[25].

First, the simplest scalable MADRL method is independent learning (IL), which simplifies the problem by assuming no interactions among agents. Independent learning can handle the scalability problem naturally because it just takes private perception into consideration when making a decision without any interaction, so the input and output dimensions of the neural network have nothing to do with other agents [8]. Fan *et al.* [7] and Long *et al.* [6] proposed collision avoidance algorithms based on independent Q learning for multi-robot. Each robot used a radar to sense the surrounding environment to construct its input variable of the network. An independent Q learning algorithm was embedded in the multi-agent platform to make action decisions for large-scale agents. Although IL may work for MADRL, it is challenged by the greedy resulting from treating each other as part of the domain. It has been proved that the initiative interaction in MADRL is conducive to improving cooperation or competition faster [26], [27].

Then, some methods based on interaction are presented. Khan *et al.* [14] proposed a scalable centralized algorithm for homogeneous agents, which collects the policy network parameters of all agents to update the global parameters, and the global parameters are used to initialize the policy network of all agents in the next round of training. This algorithm can handle the scalability problem of MADRL to some extent, but ignores the fact that the others' policies are also being updated synchronously during the training process. Everett *et al.* [28] proposed a method to encode the joint state of finite agents as a fixed-length vector by utilizing the temporal association ability of Long Short-Term Memory (LSTM) to solve the scalability problem of MADRL. Zhang *et al.* [29] proposed two decentralized actor-critic algorithms for networked agents, which are applicable to large-scale MARL problems. But the global state and joint actions of all agents are required to train the critic networks, so the algorithms may fail when the global state or joint actions are not available, which is often the case in large-scale agents. However, the dimensions of the neural network must be broad enough to formulate the relationship of all agents and the order of the joint state is fixed, which indicates that this method is not efficient and flexible in practical application. The mean field reinforcement learning (MFQ) algorithm [15] solved the scalability problem by modeling the agents' interactions as those between each agent and the mean action from its neighbors, which extremely simplifies the multi-agent interaction problem. This algorithm also provides an idea to solve the scalability problem of MADRL from the perspective of modeling the interactions of agents.

B. CONTRIBUTION

This paper studies the scalability issue of MADRL with abundant homogeneous agents whose scale varies dynamically. Our solution is a value-based method that can be executed in a decentralized fashion, and the learned policy can be shared to homogeneous agents.

Generally, local interaction is a natural and practical interplay way for natural swarms, such as ants, bees, and birds, rather than global interaction. We introduce the strategic interaction model of economics to formulate the MARL problem. Inspired by this model, the behavior of an agent selecting an action in response to the play of a neighbor is called an interaction pair between them. The interaction function is defined to evaluate the expected cumulative reward of the interaction pair between each agent and one of its neighbors. Since the interaction function cannot be easily and analytically formulated, we design a deep neural network to approximate it. And the sum of the interaction functions between the agent and all its neighbors is defined as the agent's state-action value (Q) function. The purpose of calculating the Q function is to establish a learning algorithm to learn the optimal policy from the data to maximize the reward of the agent. Accordingly, following centralized training and decentralized execution paradigm, a corresponding strategic interaction reinforcement learning algorithm (SIQ) based on double Q networks is proposed, wherein a policy is learned from the experiences of all homogeneous agents and shared by all these agents. Hereby, each agent makes its action decision to interplay with others based on the learned policy to maximize its expected cumulative reward.

The major contribution in this paper are: A decentralized and scalable MARL formulation based on strategic interaction model is presented to handle the scalability issue of MADRL by approximately decomposing each agent's complex interactions with others into the sum of interaction pairs between the agent and every neighbor. In particular, this novel Q function applies not only to traditional MARL but also to MADRL. Experiments illustrate that the scalability of the corresponding SIQ algorithm is better than the existing IL and MFQ algorithms in scenarios with different numbers of agents and dynamic changes in agents' population.

This paper is organized as follows. Section II briefly introduces some basic knowledge of MARL. Section III details the scalable MADRL formulation and the deductive process of the SIQ algorithm. In Section IV, the experimental details and results are presented and analyzed. Finally, we conclude the paper and consider avenues for future research in Section V.

II. PRELIMINARIES AND NOTATION

In this section we introduce the relevant background and notations that we build on in the remainder of the paper. The mathematical model of MARL can be formulated as a Markov Decision Process with the tuple:

$$(n, S, A_1, \dots, A_n, P, r_1, \dots, r_n, \gamma)$$

Assuming there are n agents in the multi-agent system, where

- S is the state space, and $s \in S$ is a state. Specifically, s could be a joint state formed by all agents' local states or a panorama of the environment.
- A_i denotes the action space of agent i, $i \in [1, \dots, n]$, and $a_i \in A_i$ represents its action. The dimension of action space A_i is assumed to be m_i. $a \triangleq (a_1, \dots, a_n)$ is defined as the joint action of all agents, so $\boldsymbol{a} \in (A_1 \times \dots \times A_n)$.
- $P : S \times (A_1, \dots, A_n) \times S \rightarrow [0, 1]$ is the transition function that characterizes the probability distribution of the next state given the current state and joint action.
- r_i is the immediate reward of agent i, which is related to the reward space, such as $(r_1, \dots, r_n) \triangleq r(s)$ or $(r_1, \dots, r_n) \triangleq r(s, a)$. The reward space describes the relationship between agents, including cooperation (team-games, where agents receive an identical reward at each time step), competition (zero-sum games) and mixed cooperation-competition (general-sum games).
- γ is a constant discount factor for emphasizing the extent to which the system considers the short and long term consequences.

Without loss of generality, this paper assumes that the agents in the environment are homogeneous, so they can share a common action space as well as a policy. For each agent, the others who are within its communication range are called as its neighbors, and the agent can directly communicate with its neighbors without delay.

In MARL, the joint policy of all agents can be defined as $\boldsymbol{\pi} \triangleq \boldsymbol{\pi}(\boldsymbol{a}|\boldsymbol{s}) = (\pi_1, \dots, \pi_n)$. For agent i, its policy π_i is defined as a conditional probability distribution from joint state s and other agents' actions to its action a_i at time t:

$$\pi_i(a_i|s, \boldsymbol{a}_{-i}) = P[a_i(t) = a_i|s(t) = s, a_{-i}(t) = \boldsymbol{a}_{-i}]. \quad (1)$$

where $\boldsymbol{a}_{-i} \triangleq (a_i, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$. Policy π_i is determined by the joint state and other agents' actions, which indicates that action a_i is affected by other agents, and vice versa.

Following the joint policy $\boldsymbol{\pi}$, the long term expected cumulative reward of agent i when starting in the joint state s at time t is defined as the state value function (V function):

$$V_i(s) = E_{\boldsymbol{\pi}}[\sum_{k=0}^{\infty} \gamma^k r_i(t + k + 1)|s(t) = s], \quad (2)$$

where $k \in [0, \dots, \infty)$ is the discrete time sequence. Furthermore, the expected cumulative reward of agent i when starting in the joint state s and taking the joint action a is defined as state-action value function (Q function):

$$Q_i(s, a) = E_{\boldsymbol{\pi}}[\sum_{k=0}^{\infty} \gamma^k (r_i(t + k + 1)|s(t) = s, a(t) = a)]. \quad (3)$$

Following the joint policy $\boldsymbol{\pi}$, $V_i(s)$ can also be decomposed as:

$$V_i(\boldsymbol{s}) = \sum_{\boldsymbol{a}} \boldsymbol{\pi}(\boldsymbol{a}|\boldsymbol{s}) Q_i(\boldsymbol{s}, \boldsymbol{a}). \quad (4)$$

So (3) can be rewritten as:

$$Q_i(s, a) = r_i(t + 1) + \gamma \sum_{s\_} P^a_{ss\_} \sum_{\boldsymbol{a}\_} \boldsymbol{\pi}(\boldsymbol{a}\_|\boldsymbol{s}\_) Q_i(\boldsymbol{s}\_, \boldsymbol{a}\_), \quad (5)$$

where $\boldsymbol{a}\_$ is the next joint action and $\boldsymbol{s}\_$ is the next joint state, and $P^{\boldsymbol{a}}_{\boldsymbol{ss}\_} = P[s(t + 1) = \boldsymbol{s}\_|s(t) = \boldsymbol{s}, a(t) = \boldsymbol{a}]$.

The optimal Q function of agent i is:

$$Q_i^*(\boldsymbol{s}, \boldsymbol{a}) = r_i(t + 1) + \gamma \sum_{s\_} P^{\boldsymbol{a}}_{\boldsymbol{ss}\_} \max_{\boldsymbol{a}\_} Q_i^*(\boldsymbol{s}\_, \boldsymbol{a}\_) \quad (6)$$

And for each agent i, the optimal action a_i^* is satisfied with:

$$a_i^* = \arg\max_{a_i} Q_i^*(\boldsymbol{s}, a_i, \boldsymbol{a}^*_{-i}) \quad (7)$$

where $\boldsymbol{a}^*_{-i} \triangleq (a_i^*, \dots, a_{i-1}^*, a_{i+1}^*, \dots, a_n^*)$. It is obvious that the action of each agent in a multi-agent environment will affect others' action decisions and also be affected by others', so the optimal action solutions for all agents are coupled. As the number of agents increases, the computational complexity of optimal actions increases exponentially, so that there is almost no stable equilibrium that all agents can satisfy (7) [11].

III. STRATEGIC INTERACTION MADRL

Existing centralized MARL methods suffer from the curse of dimensonaltiy with regards to the joint state-action space. Furthermore, due to the action decisions of all agents being highly coupled, calculating a global equilibrium for all agents is currently intractable. Thus we resort to the strategic interaction model [30], which is a local interaction game model broadly used in coordinate game and economics.

The strategic interaction model is shown in Figure 1. On a vertex, each player is directly connected to finite neighbors, and those neighbors are collected by a nonempty set $\mathcal{N}$. Each player makes action decision in response to its neighbors' plays. When a player takes an action, it would receive instant payoff from each of its interacting neighbors, and each payoff is determined by the actions of the player itself and the corresponding neighbor. If the player chooses action ω while a neighbor k takes υ_k, the immediate payoff is formulated by $G(\omega, \upsilon_k)$, and the total payoff of primary player is the sum of

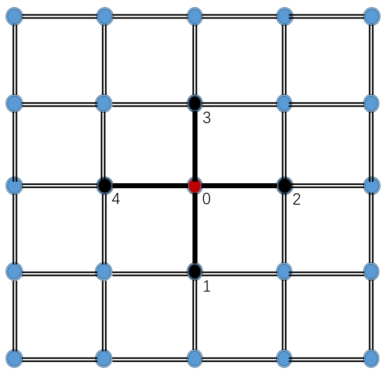


FIGURE 1. Strategic interaction model. Player 0 directly communicates with neighboring players 1, 2, 3, 4, and the interactions between agent 0 and its neighbors can be shaped as the sum of the interaction pairs between itself and each neighbor.

immediate payoffs from all its interacting neighbors, which is denoted as:

$$G(\omega) = \sum_{k \in \mathcal{N}} G(\omega, \upsilon_k) \tag{8}$$

The strategic interaction model formulates the payoff of each player with local interaction pairs without obtaining the joint state and joint action of all agents, which significantly reduces computational complexity while retaining direct interactions.

To break the dimension curse of MARL, we could adapt the idea of using local strategic interaction pairs to factorize the value function of each agent with the local state as well as the actions of each agent and its neighbors. Then each agent only directly interacts with its neighbors and makes its decentralized action decision according to its value functions.

Instead of passive observation [6], [8], agents use active local communication to achieve decentralized interactions with their neighbors. Communication enables agents to act as a group rather than as a collection of individuals, so as to achieve collaboration faster. In MARL, compared with global communication, local communication relaxes the agent's assumptions, such as communication bandwidth and computing ability, meanwhile increases the scalability.

For agent i, its local state is s_i and the action is a_i, set $\mathcal{N}_i$ contains its n_i neighbors. $c_i \triangleq (c_{i1}, \ldots, c_{in_i})$ denotes the communicate information from all its neighbors, where $c_{ij}(j = 1, \ldots, n_i)$ is the communication information from neighbor j to agent i and it could be task-dependent, such as position, speed and so on. Specially, c_{i0} means that there is no neighbor around agent i. We define (s_i, c_{ij}, a_i) as an interaction pair between agent i and neighbor j. And $Q_{ij}(s_i, c_{ij}, a_i)$ is the interaction function of the interaction pair between agent i and neighbor j. Then, as shown in Figure 2, the Q function of agent i is defined as the sum of all its interaction functions:

$$Q_i(s_i, c_i, a_i) = \sum_{j=1}^{n_i} Q_{ij}(s_i, c_{ij}, a_i). \tag{9}$$

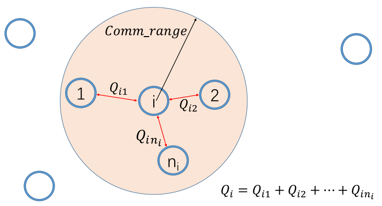


FIGURE 2. Illustration of the strategic interaction Q function under local interactions. Each agent only interacts with its neighbors within communication range. And the strategic interaction Q function is defined as the sum of the expected cumulative rewards of the interaction between the agent and each neighbor.

The major difference between (9) and others decomposition methods, such as QMIX [17], value-decomposition networks [19], QTran [20], is that it decomposes the value function of each agent into the sum of the expected cumulative rewards of the interaction between the agent and each neighbor, while the latter ones decompose the global value function into the sum of individual functions. Compared with IL in [6]–[8], [31], (9) not only formulates a novel Q function that considers the local interactions, but also enables agents to achieve a high degree of scalability on the basis of maintaining decentralized collaboration. So the MARL methods based on this formulation are able to adapt to the dynamic change of the number of agents, and then the learned policy can be flexibly applied to scenarios with different numbers of homogeneous agents.

Specifically, in order to collaborate effectively with neighbors, it is necessary to know the intentions of its neighbors [8]. For example, in a collision avoidance scenario, it is necessary to know the position and action of its neighbors. In this paper, to calculate the interaction function $Q_{ij}(s_i, c_{ij}, a_i)$, the local state and current action of neighbor j are required. Since all agents make action decisions synchronously and the neighborhood relationships of agents are symmetric, the action of agent i is also required for calculating Q_{ji}, which makes the action solving process of all agents coupled with each other. So it is impossible to obtain the current actions of the neighbors.

However, since the last actions of agents have been executing from the last time stamp till now, we can utilize the last action of neighbor j to indicate its intention by telling the agent what neighbor j just did. Then the communication information of agent i from neighbor j is redeclared as $c_{ij} = (s^c_{ij}, a'_j)$, where s^c_{ij} is the motional state of neighbor j, and a'_j is its last action. Note that $s^c_{ij} \neq s^c_{ji}$. Thus we calculate the Q function for agent i as follows:

$$Q_i(s_i, c_i, a_i) = \sum_{j \in \mathcal{N}_i} Q_{ij}(s_i, s^c_{ij}, a'_j, a_i). \tag{10}$$

In action space A_i, agent i can obtain its action decison by using the $\varepsilon - greedy$ policy to balance the exploitation and exploration during the training process. Given the local state s_i and communication information c_i, the policy of agent i

with strategic interactions could be defined as follows:

$$\pi(a_i|s_i, c_i) = \begin{cases} 1 - \varepsilon + \dfrac{\varepsilon}{|A_i|} & a_i = \underset{a_i}{\operatorname{argmax}}\, Q_i(s_i, c_i, a_i) \\ \dfrac{\varepsilon}{|A_i|} & otherwise. \end{cases} \tag{11}$$

Although this novel Q function applies not only to traditional MARL but also to MADRL, it is difficult to explicitly formulate $Q_{ij}(s_i, a_i, c_{ij})$ with an explicit formulation or several eigenvectors, so we appeal to the deep neural network to approximate the interaction function, which has powerful fitting ability [32]. The structure of the designed neural network is shown in Figure 3. In deep RL, a neural network is usually trained with the experience replay mechanism that is widely used to stabilize and converge the training process [4]. An experience is usually formed by tuple $(s, a, r, s\_)$ that includes state, action, immediate reward and the next state. In the interaction function, communication information is added to the input variables, so the experience of agent i is reformed as: $(s_i, c_i, a_i, r_i, s_{i\_}, c_{i\_})$, where c_i and $c_{i\_}$ are the current and next communication informations from neighbors. The dimension of each communication information is determined by the number of neighbors interacting with the agent at that moment, not the number of all agents, which breaks the dimensional curse.

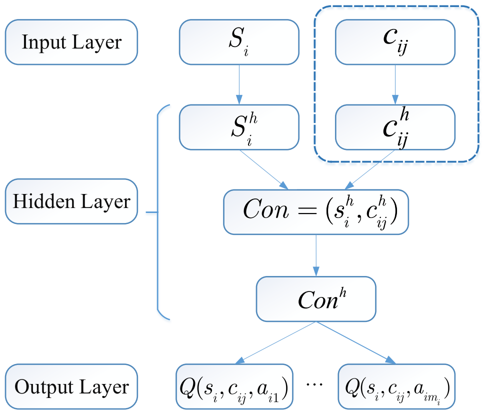


FIGURE 3. The structure of the neural network. The superscript *h* indicates that this is a hidden layer. *Con* is the concatenation of hidden layers of agent *i* and neighbor *j*. The portion in the dotted box is used to handle the communication information from neighbor *j*. The output dimension of the neural network is equal to the action space dimension of the agent *i*.

In an environment with large scale homogeneous agents, all agents share identical state space and action space. If they perform a cooperative mission, their policies are similar, so all agents can share a single interaction function to calculate their own Q functions respectively, which significantly improves the diversity of training data and is beneficial to improving the training efficiency [4]. Similar to [33], all agents' experiences are collected into a buffer to train the neural network of the interaction function in a centralized paradigm, and the trained network is distributed to each agent for solving their actions respectively.

However, the original deep Q learning may overestimate the Q value. In order to overcome this disadvantage, double Q networks algorithm is proposed by [4], and its validation is also verified by [3], [34]. So we opt for this algorithm to train the designed neural network. This algorithm employs an evaluation network Q_e and a target network Q_t, which are the instantiations of Q_{ij} in (10) using parameters ϕ_e and ϕ_t respectively. Thus, the two networks have the same architecture and similar parameters.

To train the double Q networks with experience $e_i = (s_i, c_i, a_i, r_i, s_{i\_}, c_{i\_})$, the loss function is defined as:

$$L_i(\phi_e) = \frac{1}{2}(y_i - Q_e(s_i, c_i, a_i, \phi_e))^2, \tag{12}$$

where $y_i = r_i + \gamma Q_t(s_{i\_}, \underset{a_{i\_}}{\operatorname{argmax}}\, Q(s_{i\_}, a_{i\_}, c_{i\_}, \phi_e), c_{i\_}, \phi_t)$ and y_i is called as Temporal-Difference target.

After that, the derivative of the loss function $L_i(\phi_e)$ with respect to the parameters ϕ_e can be obtained by:

$$\nabla_{\phi_e} L_i(\phi_e) = -(y_i - Q_e(s_i, a_i, c_i, \phi_e))\nabla_{\phi_e} Q_e(s_i, a_i, c_i, \phi_e). \tag{13}$$

The derivative term $\nabla_{\phi_e} L_i(\phi_e)$ is used to update the evaluation network parameters ϕ_e:

$$\begin{aligned} \phi_e &= \phi_e - \alpha \nabla_{\phi_e} L_i(\phi_e) \\ &= \phi_e + \alpha(y_i - Q_e(s_i, a_i, c_i, \phi_e))\nabla_{\phi_e} Q_e(s_i, a_i, c_i, \phi_e), \end{aligned} \tag{14}$$

where α is interpreted as the learning rate.

And the target network parameters ϕ_t can be updated with ϕ_e as follows:

$$\phi_t = \tau\phi_e + (1 - \tau)\phi_t \tag{15}$$

where τ is the target update rate.

All agents' experiences are stored and randomly sampled to update the network parameters iteratively until the loss function and the network parameters converge. Basing on the above deduction, we propose the scalable Multi-agent Q learning (SIQ) algorithm to learn a decentralized policy for a group of agents. More details about the algorithm can be found in Algorithm 1.

IV. NUMERICAL EXPERIMENTS

We train and evaluate the proposed method on a multi-agent confrontation platform that contains a mixed cooperative-competitive confrontation game [31]. In the following experiments, we firstly verify and analyze the proposed SIQ algorithm, and then compare the scalability of SIQ algorithm with independent learning and mean field reinforcement learning algorithms in several scenarios with different numbers of agents.

A. MULTI-AGENT ENVIRONMENT

A multi-agent environment that supports large-scale agent simulation and allows a wide variety of agents is essential for training and evaluating the proposed algorithm. MAgent is developed by Geek.AI organization and popularly used in

Algorithm 1 Strategic Interaction Q Learning (SIQ)

```
Initialize networks $Q_e$ and $Q_t$ with parameters $\phi_e$ and $\phi_t$
Initialize the experience replay buffer $D$
Initialize the environment with $n$ agents
Setup configuration: max training round $n_{round}$, max step $n_{step}$, minibatch size $d$
for $k = 1,\ldots,n_{round}$ do
    for $m = 1,\ldots,n_{step}$ do
        for $i = 1,\ldots,n$ do
            For each agent $i$, obtain its local state $s_i$ and communication information $c_i$.
            Calculate $Q_i$ with $Q_e(\phi_e)$ as:
                $$Q_i(s_i, c_i, a_i) = \sum_{j \in \mathcal{N}_i} Q_{ij}(s_i, s^c_{ij}, a'_j, a_i).$$
            Opt for the optimal action $a_i$ to maximize $Q_i$.
        end for
        Step with the actions $(a_1, \ldots, a_{n_i})$ or exploration operations to obtain rewards $(r_1, \ldots, r_{n_i})$
        for $i = 1,\ldots,n$ do
            For each agent $i$, obtain the next local state $s_{i\_}$ and new communication information $c_{i\_}$ from the updated environment
            Form the experience $(s_i, c_i, a_i, r_i, s_{i\_}, c_{i\_})$, then store it into $D$
        end for
        Sample a minibatch randomly from $D$
        for $j = 1,\ldots, d$ do
            For experience $e_j$, calculate $Q_{ej}$, next action decision $a_{j\_}$ and the Temporal-Difference target $y_j$, then the loss function is:
                $$L_j(\phi_e) = (y_j - Q_{ej})^2$$
        end for
        Average the loss functions:
            $$L(\phi_e) = \frac{1}{d}\sum_{j=1}^{d} L_j(\phi_e)$$
        Update the parameters $\phi_e$ by minimizing the average loss function $L(\phi_e)$
        if update target network then
            Update parameters $\phi_e$ with $\phi_e$:
                $$\phi_t = \tau\phi_e + (1 - \tau)\phi_t$$
        end if
    end for
end for
```

recent researches [14], [15], [31] because of its abundant scenarios, including confrontation, gathering, pursuit and tiger, as well as its cross-plarform ability and environmental munificence. Owing to the open source of the MAgent[1] platform, we can easily adapt the environment to train and test the proposed algorithms. As far as we know, this platform is the only one for now satisfying the requirements of large-scale quantity and scalability.

Here we choose the confrontation scenario and modify it according to our requirement. The modified scenario contains abundant homogeneous agents with identical state spaces and action spaces. The agents in the environment are divided into the red and blue teams, and each team is supposed to learn a cooperative policy to eliminate its enemy. Different algorithms can be respectively embedded into the two teams to make the action decisions for their members. With different algorithms and settings, the agents can learn skills such as attack, pursuit, escape, etc.

B. MODEL AND PARAMETERS SETTING

To calculate the Q function of each agent for making its action decision, the input variable should include the agent's local state and the received motional states and actions from its neighbors. In the neural network shown in Figure 3, the Convolutional Neural layer is used to handle its local state input, Multilayer Perceptrons are used to process its communication information and hidden layers, and ReLu functions are used as the activation functions.

The basic rewards of the confrontation scenario are shaped as follows: -0.005 for each movement, which encourages agents to accomplish task as soon as possible, 5 for diminishing an opponent and 0.2 for a valid attack. The hyper-parameters of the SIQ algorithm are given in Table 1. Specially, the exploration probability ε in (11) is set to be 1 at the beginning of the training, which will encourage agents to make action decisions randomly to explore the environment. Then the exploration probability gradually decreases to 0.05 as the training goes on, and the agents could exploit the trained neural networks to make action decisions to maximize their cumulative rewards.

TABLE 1. Hyper-parameters configuration.

Hyper-parameter	Value
Max Step	300
Exploration Probability	$1 \sim 0.05$
Experience Buffer Size	2^{10}
Minibatch Size	64
Target Network Update Rate	0.95
Learning Rate	1e-4
Discount Factor	0.95

C. RESULTS AND DISCUSSIONS

1) VALIDITY VERIFICATION OF SIQ ALGORITHM

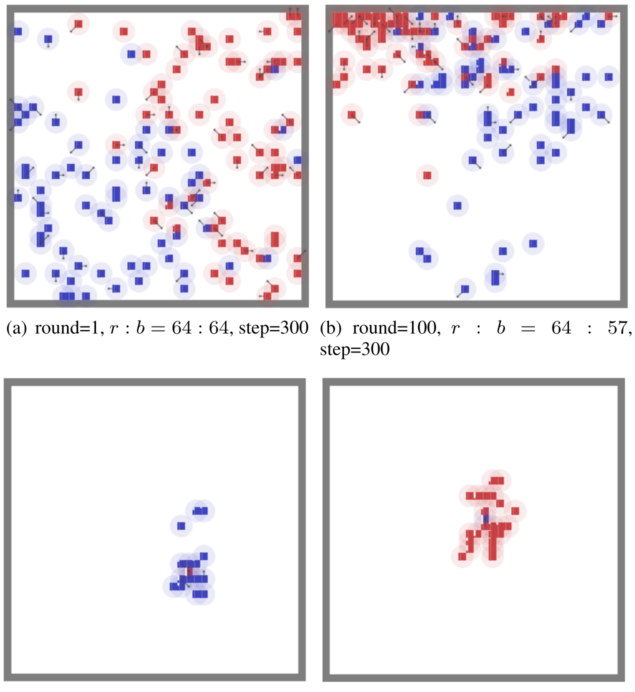
This experiment is supposed to verify the validity of the proposed SIQ algorithm. In the environment initialization, both red team and blue team consist of 64 agents and the map size is 40×40. The number of agents here far exceeds that in [11], [28]. Here, the number of agents in the training scenario is low, because we want to learn a policy that can be learned at a lower cost (including time and computing

[1]https://github.com/geek-ai/MAgent

resources) in an environment with fewer agents, but it can be well applied to those scenarios with more agents. And if realized, this also could prove that the proposed algorithm can address the MADRL scalability challenge.

Both teams use the SIQ algorithm to train their own neural networks, and each team shares its own neural network with all members to calculate their Q functions. The parameters of the neural networks for two teams are initialized randomly. Each training round would be terminated if the simulation step arrives the maximal or one team is annihilated.

One of the many trials is randomly chosen to illustrate the training process. There are four sub-figures in Figure 4, which are the results of simulation when the training terminates at round $n_{round} = 1, 100, 200, 300$ respectively. And the two curves in Figure 5 show the changes in single-step average rewards of the agents in both teams.


(a) round=1, $r : b = 64 : 64$, step=300 (b) round=100, $r : b = 64 : 57$, step=300

(c) round=200, $r : b = 1 : 15$, step=119 (d) round=300, $r : b = 23 : 1$, step=50

FIGURE 4. **Combat terminal results at different training rounds.**

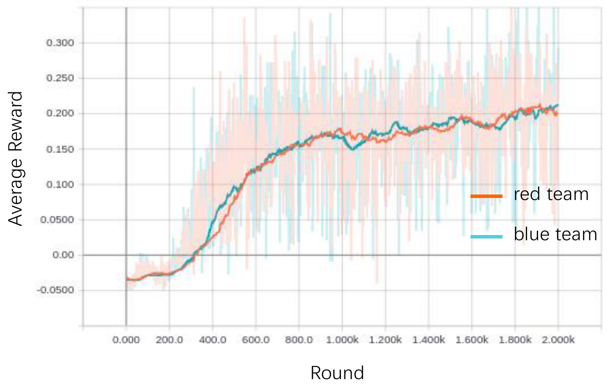


FIGURE 5. **Single-step average rewards of the two teams.**

During the preliminary training, due to the high exploration probability and the random initialization of neural networks' parameters, the actions of both teams are conducted from random sampling. So both teams do not aggressively combat against each other, which is proved by their survivors' numbers in sub-figure 4(a) when the training process reaches the maximum step. When $n_{round} = 100$, the comparison of their survivors displays that the red team has more survivors than the blue one, which indicates that the red team has learned better network parameters temporarily. At round $n_{round} = 200$, the blue team wins the match, while the red team wins when $n_{round} = 300$. It is obvious that both teams have gradually learned how to combat against their opponents, but the winning and losing between them is alternate, which can also be seen from the alternating rise between the two curves in Figure 5. As the training progresses, both the two teams can win the match before reaching the maximum step limit, and their single-step average rewards are getting higher and higher, so their neural networks are constantly evolving and their policies are gradually improving.

The results of this experiment show that although the number of agents in the training process is varying, the novel Q function can be competent for handling the scalability issue, and the corresponding SIQ algorithm can learn a effective policy to accomplish the cooperative confrontation task. More tests of the performance of the proposed method in the scenarios with agents in different scales will be carried out in subsection IV-C2.

2) SCALABILITY COMPARISONS

SIQ is a scalable MARL algorithm based on value function, we compare it with two other scalable baselines,IL and MFQ algorithms, on the MAgent platform. On the one hand, agents are partially observable and they may die due to the attack from their opponents, so we exclude algorithms that require global information and input variables with fixed dimension, such as QMIX [17] and QTran [20], etc. One the other hand, one may concatenate all agents' local information to replace the global information, such as MADDPG [11]. However, when the number of agents is large or the dimension of local information is very high, the joint dimension would be extremely high, which would be cursed by dimensionality. And the concatenation cannot avoid the permutation irrelevant issue. Also, Yang *et al.* [15] have proved that on the MAgent platform, MFQ outperforms IL, and the latter performs far better than the actor-critic and mean-filed actor-critic algorithms. Therefore, we exclude algorithms that are only applicable to a small number of agents and the number is fixed as well as those perform poorly.

Firstly, in the environment where its configuration is same as one in subsection IV-C1, we respectively train the SIQ, IL, and MFQ algorithms multiple times. The single-step average rewards of the three algorithms during training are shown in Figure 6. Obviously, as the training progresses, the average reward of the SIQ algorithm increases faster than the ones of IL and MFQ algorithms. It is reasonable since the SIQ

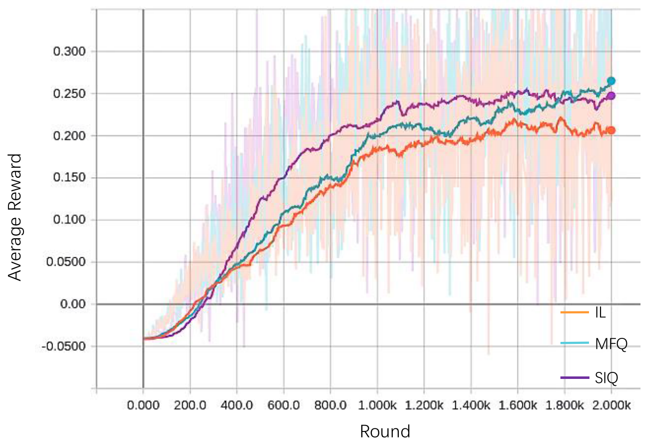


FIGURE 6. Single-step average rewards of the three algorithms.

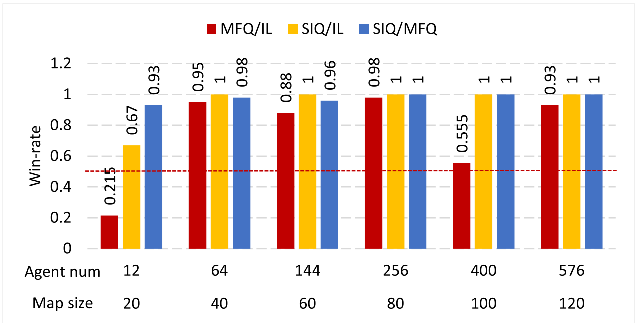


FIGURE 7. Win-rates of the three algorithms in different scenarios. Map size increases by 20 units each time, and the number of agents in each team will increase accordingly. The red dotted line indicates a draw.

algorithm takes a more complicated interaction model into consideration and uses a more complex network to formulate agents' interactions.

Qualitatively, the SIQ algorithm is the highest computational complexity, while the IL algorithm is the lowest one. Theorectical analysis is not easy, so we indirectly analyze it based on the training time of the algorithms. The average training time of the three algorithms are compared experimentally in Table 2. However, from the statistic results, we find that the MFQ algorithm spends the least time, and the SIQ algorithm spends the most but there is only a small increase. The possible reason is that in each training episode, if agents follow a better policy, they could complete the task faster and win the game, and their rewards would be higher, which is also illustrated by the average individual rewards in Figure 6.

TABLE 2. Average training time of the three algorithms.

Algorithm	Training Time
SIQ	8h54m
IL	8h20m
MFQ	**8h10m**

Furthermore, we also compare the scalability of the three algorithms in different confrontational scenarios, wherein the population of each team varies from 12 to 576 in different scenarios. In each confrontation scenario, the red and blue teams select two different ones of the three algorithms to make action decisions for all of their members. So the crosswise tests cover three antagonistic relations, including MFQ *vs* IL, SIQ *vs* IL and SIQ *vs* MFQ. And the network structures and parameters are loaded from the training results without modification or retraining. In all scenarios, each test is carried out for 100 rounds, and then we calculate the win-rates to evaluate the algorithms' performances. If there is a dead heat, the game will be declared a draw. The final win-rates of the crosswise comparisons are presented in Figure 7.

It can be seen roughly from Figure 7 that the comprehensive performance of the IL algorithm is the worst, while the SIQ algorithm performs best. However, the win-rate of MFQ *vs* IL is not stable in all the scenarios. Specifically, in the scenario with 12 agents, the win-rate is 0.215; in the other cases, it exceeds 0.5. This may imply that the scalability of the MFQ algorithm is not always superior to the IL algorithm in all scenarios. Therefore, the interaction with average actions of neighbors does not always promote cooperation.

In all these scenarios, the win-rates of SIQ *vs* IL and SIQ *vs* MFQ exceed 0.5, and when the number of agents is greater than 12, they are close to or equal to 1. This implies that the introduction of the strategic interaction model can significantly promote the collaboration of agents, thereby improving their ability to perform complex tasks. It is because the SIQ algorithm takes into account the more complex interactions between agents. And our method more accurately models the interaction between the agent and each of its neighbors, and can fully consider the potential cooperation or competition between the agents, so that the agent can make more reasonable decisions. In particular, this advantage can be maintained when the number of agents changes dramatically, because each agent only takes into account interactions with its neighbors who are directly determined by the local communication topology. Conversely, the MFQ algorithm simply considers the mean action of each agent's neighbors, which may ignore the intents of the neighbors in some complex situations. For example, in a collision avoidance mission, two neighbors moving in opposite directions may be mistaken for a fixed obstacle. Furthermore, when the number of neighbors is large enough, the mean action would be equal to a constant according to Law of Large Numbers, $\lim_{n\to\infty}(\sum_{1}^{n} a_n)/n = E(a_n) = constant$. Then the mean action would be meaningless and the MFQ algorithm would degenerate into the IL algorithm. In addition to these tests, we further test the win-rate of the SIQ algorithm in other scenarios, and the comparison results are similar.

In general, the outstanding performance of the SIQ algorithm demonstrates its scalability in different scenarios, and a well-trained SIQ network can be used in the scenarios with hundreds and even more agents.

V. CONCLUSION

This paper focuses on scalability issues of MADRL for homogeneous agents. We define the interaction pair to describe the local interplay of an agent with one of its neighbors. A novel MARL formulation based on the strategic interaction model of economics is presented, which approximates the Q function of each agent with the sum of the expected cumulative rewards of related interaction pairs. By transforming the multi-body decentralized interaction into the sum of multiple two-body local interactions, the novel Q function is scalable to the dynamic changes in the number of large-scale agents. Then the corresponding SIQ algorithm is proposed to train a neural network that is used to estimate the expected cumulative rewards of an interaction pair. The empirical experiments testify that the proposed method can learn a effective and sharing policy for homogeneous agents in cooperative tasks. Furthermore, the crosswise comparison experiment justifies the outstanding scalability of the proposed SIQ algorithm. This offers an effective solution for the scalability issue of MADRL, especially in scenarios that the population of agents is abundant and dynamic.

However, due to the design of a more complex interaction model, the computational complexity of our method is higher than the baselines, which should be further analyzed and simplified in future work. In the training process, we note that the performance of the neural network is sensitive to its hyper-parameters configuration, and at the same time the tuning work of the hyper-parameters is time-consuming and requires a lot of experience. We will further improve the performance of the SIQ algorithm via Population Based Training [35] that could be used to learn the hyper-parameters of the neural network and even to tune its structure.

REFERENCES

[1] J. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 2137–2145.

[2] J. N. Foerster, 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*. [Online]. Available: http://arxiv.org/abs/1602.02672

[3] C. Schulze and M. Schulze, "ViZDoom: DRQN with prioritized experience replay, double-Q learning and snapshot ensembling," in *Proc. Intell. Syst. Conf.*, London, U.K., vol. 1, Sep. 2018, pp. 1–17.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[5] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," 2016, *arXiv:1603.01121*. [Online]. Available: http://arxiv.org/abs/1603.01121

[6] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Brisbane, QLD, Australia, May 2018, pp. 6252–6259.

[7] T. Fan, P. Long, W. Liu, and J. Pan, "Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios," 2018, *arXiv:1808.03841*. [Online]. Available: http://arxiv.org/abs/1808.03841

[8] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Singapore, May 2017, pp. 285–292.

[9] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based LSTMs," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Changshu, China, Jun. 2018, pp. 1179–1184.

[10] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Changshu, China, Jun. 2018, pp. 1672–1678.

[11] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 6379–6390.

[12] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PLoS ONE*, vol. 12, no. 4, pp. 1–12, 2017.

[13] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, Aug. 2017, pp. 2681–2690.

[14] A. Khan, C. Zhang, D. D. Lee, V. Kumar, and A. Ribeiro, "Scalable centralized deep multi-agent reinforcement learning via policy gradients," 2018, *arXiv:1805.08776*. [Online]. Available: http://arxiv.org/abs/1805.08776

[15] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, 2018, pp. 5571–5580.

[16] V. Conitzer and T. Sandholm, "AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents," *Mach. Learn.*, vol. 67, nos. 1–2, pp. 23–43, May 2007.

[17] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, 2018, pp. 4295–4304.

[18] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. 32th AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, 2018, pp. 2974–2982.

[19] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning," in *Proc. Int. Joint Conf. Auton. Agents Multiagent Syst.*, Stockholm, Sweden, 2018, pp. 2085–2087.

[20] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, CA, USA, Jun. 2019, pp. 10329–10346.

[21] A. OroojlooyJadid and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," 2019, *arXiv:1908.03963*. [Online]. Available: http://arxiv.org/abs/1908.03963

[22] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Auton. Agents Multi-Agent Syst.*, vol. 33, no. 6, pp. 750–797, Nov. 2019.

[23] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, early access, Mar. 20, 2020, doi: 10.1109/TCYB.2020.2977374.

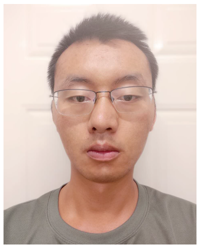
[24] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," 2019, *arXiv:1911.10635*. [Online]. Available: http://arxiv.org/abs/1911.10635

[25] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in Multi-Agent Systems and Applications*. Berlin, Germany: Springer, 2010, pp. 183–221.

[26] D. Szer and F. Charpillet, "Improving coordination with communication in multi-agent reinforcement learning," in *Proc. 16th IEEE Int. Conf. Tools Artif. Intell.*, Boca Raton, FL, USA, Nov. 2004, pp. 436–440.

[27] K. Cao, A. Lazaridou, M. Lanctot, J. Z. Leibo, K. Tuyls, and S. Clark, "Emergent communication through negotiation," in *Proc. Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, 2018, pp. 1–15.

[28] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Madrid, Spain, Oct. 2018, pp. 3052–3059.
[29] K. Zhang, Z. Yang, H. Liu, Z. Tong, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, 2018, pp. 9340–9371.
[30] L. E. Blume, "The statistical mechanics of strategic interaction," *Games Econ. Behav.*, vol. 5, no. 3, pp. 387–424, Jul. 1993.
[31] L. Zheng, J. Yang, H. Cai, M. Zhou, W. Zhang, J. Wang, and Y. Yu, "MAgent: A many-agent reinforcement learning platform for artificial collective intelligence," in *Proc. 32nd AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, 2018, pp. 8222–8223.
[32] H. He, J. Boyd-Graber, K. Kwok, and H. Daumé, III, "Opponent modeling in deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 1–10.
[33] Y. Yang, Y. Wen, L. Yu, W. Zhang, Y. Bai, and J. Wang, "A study of AI population dynamics with million-agent reinforcement learning," in *Proc. Int. Joint Conf. Auton. Agents Multiagent Syst.*, Stockholm, Sweden, 2018, pp. 2133–2135.
[34] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, 2016, pp. 2094–2100.
[35] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population based training of neural networks," 2017, *arXiv:1711.09846*. [Online]. Available: http://arxiv.org/abs/1711.09846



WENHONG ZHOU received the B.S. degree in flight vehicle design and engineering from the Harbin Institute of Technology, Harbin, China, in 2014, and the M.S. degree in control science and engineering from the National University of Defense Technology, Changsha, China, in 2017, where he is currently pursuing the Ph.D. degree in control science and engineering. His current research interests include artificial intelligence, swarm control, and corresponding applications to unmanned aerial systems.

JIE LI received the B.S. degree in automation and the M.S. and Ph.D. degrees in pattern recognition and intelligent systems from the National University of Defense Technology, Changsha, in 2006, 2008, and 2014, respectively. From 2014 to 2019, he was a Lecturer with the Unmanned Aerial Systems Laboratory. Since 2020, he has been an Assistant Professor with the College of Intelligence Science and Technology, National University of Defense Technology. His research interests include swarm intelligence, emergent behavior, multi-agent cooperation, distributed control, and collective decision-making.

YITING CHEN received the B.S. degree in automation from the National University of Defense Technology, Changsha, in 2017, where she is currently pursuing the Ph.D. degree in control science and engineering. Her research interests include multi-agent systems, deep reinforcement learning, and collective decision making.

LIN-CHENG SHEN (Member, IEEE) received the B.E., M.S., and Ph.D. degrees in automatic control from the National University of Defense Technology. He is currently a Full Professor and the Dean of the School of Postgraduates. He has initiated and organized several workshops and symposia, including the International Workshop on Bionic Engineering, in 2012, and the Chinese Automation Congress, in 2013. He has published over 100 technical papers in refereed international journals and academic conference proceedings. His current research interests include mission planning, autonomous and cooperative control, biomimetic robotics, and intelligent control. Since 2007, he has been serving as an Editorial Board Member of the *Journal of Bionic Engineering*.