Stigmergic Independent Reinforcement Learning for Multiagent Collaboration

Xing Xu, Rongpeng Li, Zhifeng Zhao, and Honggang Zhang

***Abstract*— With the rapid evolution of wireless mobile devices, there emerges an increased need to design effective collaboration mechanisms between intelligent agents to gradually approach the final collective objective by continuously learning from the environment based on their individual observations. In this regard, independent reinforcement learning (IRL) is often deployed in multiagent collaboration to alleviate the problem of a nonstationary learning environment. However, behavioral strategies of intelligent agents in IRL can be formulated only upon their local individual observations of the global environment, and appropriate communication mechanisms must be introduced to reduce their behavioral localities. In this article, we address the problem of communication between intelligent agents in IRL by jointly adopting mechanisms with two different scales. For the large scale, we introduce the stigmergy mechanism as an indirect communication bridge between independent learning agents, and carefully design a mathematical method to indicate the impact of digital pheromone. For the small scale, we propose a conflict-avoidance mechanism between adjacent agents by implementing an additionally embedded neural network to provide more opportunities for participants with higher action priorities. In addition, we present a federal training method to effectively optimize the neural network of each agent in a decentralized manner. Finally, we establish a simulation scenario in which a number of mobile agents in a certain area move automatically to form a specified target shape. Extensive simulations demonstrate the effectiveness of our proposed method.**

***Index Terms*— Artificial intelligence, collective intelligence, multiagent collaboration, reinforcement learning, stigmergy.**

I. Introduction

With the rapid development of mobile wireless communication and Internet of things technologies, many scenarios have arisen in which collaboration between intelligent agents is required, such as in the deployment of unmanned aerial vehicles (UAVs) [1]–[3], distributed control in the field of industry automation [4]–[6], and mobile crowdsensing and computing (MCSC) [7], [8]. In these scenarios, traditional centralized control methods are usually impractical due to limited computational resources and the demand for ultralow latency and ultrahigh reliability. As an alternative, multiagent collaboration technologies can be used in these scenarios to relieve the pressure on the centralized controller.

Guiding autonomous agents to act optimally through trial-and-error interaction with the corresponding environment is the primary goal in the field of artificial intelligence and is regarded as one of the most important objectives of reinforcement learning (RL) [9]–[11]. Recently, deep RL (DRL), which combines RL and deep neural networks, has improved the ability to obtain information from high-dimensional input, such as high-resolution images, and has demonstrated extraordinary learning ability across a wide array of tasks [12]. There are a number of advanced DRL algorithms that can direct a single agent to improve its behavioral policy by continuously learning from the environment [13], [14]. However, the extension of single-agent DRL to multiagent DRL is not straightforward, and many challenging problems remain to be solved in the application of multiagent RL (MARL) [15], [16]. In particular, in a completely distributed multiagent system (MAS), each agent is usually limited to partially observe the global environment, and its learning process following this local observation can thus be nonstationary, as other agents' behavioral policies may change temporally. In addition, the assignment of an individual reward is another challenging problem, as there is only one global reward for feedback in most cases. As an alternative, independent RL (IRL) has been proposed to alleviate the problem of a nonstationary learning environment, where each agent undergoes an independent learning process with only self-related sensations [17].

In IRL, most behavioral policies learned by intelligent agents are self-centered, aiming to maximize their own interests. Thus, the target of mutual communication is to integrate these individual behaviors effectively for the same task. Several studies have attempted to solve the problem of mutual communication in IRL. Foerster *et al.* [18] proposed differentiable interagent learning (DIAL), in which an additional communication action is added to the action set of each agent. In addition to the selection of the current action, a piece of interagent message is also generated and sent to other agents through a specified communication channel.

Manuscript received 1 November 2019; revised 29 September 2020 and 29 December 2020; accepted 27 January 2021. Date of publication 15 February 2021; date of current version 2 September 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1804804, in part by the National Natural Science Foundation of China under Grant 61731002 and Grant 62071425, in part by the Zhejiang Key Research and Development Plan under Grant 2019C01002 and Grant 2019C03131, in part by the Huawei Cooperation Project, the Project sponsored by Zhejiang Lab under Grant 2019LC0AB01, and in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LY20F010016. *(Corresponding author: Rongpeng Li.)*

Xing Xu, Rongpeng Li, and Honggang Zhang are with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: hsuxing@zju.edu.cn; lirongpeng@zju.edu.cn; honggangzhang@zju.edu.cn).

Zhifeng Zhao is with Zhejiang Lab, Hangzhou, China (e-mail: zhaozf@zhejianglab.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3056418.

Digital Object Identifier 10.1109/TNNLS.2021.3056418

Thereafter, a back-propagated error from the receiver of the communication messages is sent back to the sender to regulate the communication action. However, this type of communication channel exists between all pairs of independent learning agents, making DIAL very complex as the number of agents increases. Tan [19] tested to share different types of messages to coordinate intelligent agents on the basis of independent Q-learning (e.g., sensations, episodes, and learned policies). Despite its improvement in the final performance, the method has several over-optimistic assumptions that limit its potential application. For example, given the limited resources in a harsh environment, it is usually impractical for a single agent to transmit large messages in mobile wireless scenarios. As an improvement, Mao *et al.* [20] proposed to utilize a coordinator network to aggregate compressed local messages and then share them among all agents. Because the shared messages contain joint information from all agents, the expected result of this design is to make each agent act optimally considering the other agents' behaviors. However, it is difficult to obtain a well-trained coordinator and local networks. In summary, appropriate communication mechanisms must be introduced between independent learning agents to reduce their behavioral localities [21].

As another approach, the concept of stigmergy was first introduced by French entomologist Pierre-Paul Grassè in the 1950s when studying the behavior of social insects [22], [23]. Recently, stigmergy has experienced rapid diffusion across a large number of application domains together with the popularization of distributed computing, collective intelligence, and broadband Internet [24]–[27]. In particular, stigmergy has demonstrated advantages in various scenarios requiring distributed control where the generation of messages is closely related to the environment, space, and time, such as the management of traffic lights [28], [29]. A key component of stigmergy is called the medium, which acts as the information aggregator in multiagent collaboration. Benefiting from the existence of a medium, effective stigmergic interaction can be established between agents and their surrounding environment, and distributed agents can also interact with each other indirectly to reduce their behavioral localities. Therefore, stigmergy is a potential technique to solve the problem of mutual communication in IRL.

In addition to reap the large-scale merits of stigmergy, we also propose a conflict-avoidance mechanism executed at smaller scales between adjacent agents to further reduce their behavioral localities. With this mechanism, we evaluate and assign a corresponding action priority to each agent, and a larger number of decision-making opportunities are provided for agents with higher action priorities. In particular, the action priority value is efficiently calculated by an additionally embedded neural network within each agent. Furthermore, to synchronously optimize the neural network of each agent, we apply a federal training method along with average optimization by improving the asynchronous advantage actor-critic (A3C) algorithm [13]. We summarize all the aforementioned techniques and propose the stigmergic IRL (SIRL) algorithm. Based on the simulation scenario in [30], in which a number of mobile agents move automatically to form a specified target shape, we examine the effectiveness of the proposed SIRL algorithm through an in-depth performance comparison with other available methods.

Focusing on the SIRL algorithm, the contributions of this article can be summarized as follows:

1) First, we introduce the stigmergy mechanism into MARL and provide an effective cooperative algorithm. We also demonstrate that the stigmergy mechanism can decompose a global objective into small tasks that can be more efficiently perceived by individual agents.
2) Second, we propose a conflict-avoidance mechanism to further reduce the behavioral localities of agents, whose foundation is an additionally embedded neural network in each agent.
3) Third, we provide a federal training method by enhancing the A3C algorithm to synchronously optimize the neural network of each independent learning agent in MAS.

The remainder of this article is organized as follows. In Section II, we discuss the related work from the perspective of combining the stigmergy mechanism and MARL and clarify the novelty of our work. In Section III, we present the system framework followed by a description of the stigmergy mechanism and the proposed conflict-avoidance mechanism. Finally, we introduce the federal training method. In Section IV, we mathematically analyze the details of the proposed SIRL algorithm. In Section V, we describe the simulation scenario, compare the numerical simulation results, and present key insights into these results. Finally, in Section VI, we conclude this article.

II. Related Work

A prototype of the stigmergy mechanism can be widely observed in natural colonies. Pagán [31] described a colony of social insects as a super-organism with brain-like cognitive abilities. This super-organism consists of a large number of small insect brains coupled with appropriate cooperative mechanisms. Despite its limited size, the small brain of each insect is capable of performing an adaptive learning process, which is similar to RL [11], [32]. As a classical mechanism explaining the cooperative behavior of social insects [26], stigmergy also includes a large number of small-scale learning processes [33], [34] and records their effect on the surrounding environment with the involved medium.

The application of stigmergy has been studied in the field of MAS [24]–[29]. Stigmergy is generally used to coordinate the behavior of multiple agents to accomplish the target task more efficiently. In particular, the coordination process in most applications focuses on the maintenance of digital pheromone, which is an important part of stigmergy, while the involved agent itself lacks the ability to learn the behavioral policy. For example, the coordination process in the classical ant colony optimization (ACO) algorithm [35] leads to an increased concentration of the correct pheromone; however, the behavioral policy of the involved agent is predetermined, that is, choosing among several concentrations in a probabilistic manner. In practice, this is effective in a bottom-to-up designed MAS [36] in which the behavioral policy of the

involved agent can generally be predetermined and it is easier to predict the system's final performance. However, in many practical scenarios, the behavioral policies of the involved agents cannot be predetermined, and the agents must adjust their own policies while maintaining coordination.

In MARL, each agent can learn its behavioral policy through interaction with the surrounding environment. However, MARL also faces several challenges, such as the partial observation problem and credit assignment problem [15], [16]. Several studies have attempted to combine stigmergy with MARL. Aras *et al.* [37] conceptually described how certain aspects of stigmergy can be imported into MARL, and defined an interagent communication framework. Verbeeck and Ann [38] investigated the impact of an ant algorithm (AA) on the extension of RL to an MAS, and explained that stigmergy is essential in systems in which agents do not fully cooperate. However, the above studies all lack available algorithms. For an agent to perform well in a partially observable environment, it is usually necessary to require its actions to depend on the history of observations [39]. Peshkin *et al.* [40] explored a stigmergic approach in which an external memory is created and included as part of the input to the involved agent. In addition, to optimize packet scheduling in routers, Bourenane *et al.* [41] presented a pheromone-Q learning approach that combines the Q-learning algorithm [42] with a synthetic pheromone that acts as a communication medium. Specifically, the synthetic pheromone is introduced into the Q-learning updating equation through a belief factor. However, the effectiveness of these methods should be verified in a more complex MAS. Verbeeck and Nowe [43] first proposed using learning automaton to design and test stigmergic communication. Furthermore, Masoumi and Meybodi [44] proposed to use the concept of stigmergy to calculate the reward received by each learning automaton (i.e., agent) to accelerate the learning process in Markov games. Similarly, Xu *et al.* [30] proposed to use the digital pheromone to coordinate the behavior of multiple agents attempting to form a target shape. In particular, the distribution of digital pheromones helps provide guidance for the movement of agents. However, none of the above-mentioned works incorporate the advantageous capability of RL into the involved agents' decision-making processes. In contrast to the above-mentioned studies, our work focuses on introducing DRL into the decision-making process of each involved agent.

Stochastic gradient descent based distributed machine learning algorithms have been studied in the literature in terms of both theoretical convergence analysis [45] and real-world experiments [46]. Moreover, in addition to the traditional distributed algorithms, Hu *et al.* [47] proposed to use the federated learning algorithm to reduce the communication cost to train one neural network. Federated learning distributes the training task (e.g., face recognition) among several devices and obtains a common neural network through the integration process. Similarly, Sartoretti *et al.* [48] proposed a distributed RL algorithm for decentralized (DC) multiagent collection. This algorithm allows multiple agents to learn a homogeneous, distributed policy, where various agents work together toward a common target without explicit interaction. Sartoretti *et al.* [48] also demonstrated that the aggregation of experience from all agents can be leveraged to quickly obtain a collaborative behavioral policy that naturally scales to smaller and larger swarms. However, the above-mentioned distributed learning algorithms usually prohibit interactions among agents to stabilize the learning process, which may be harmful, especially in cases where agents do not fully cooperate. Therefore, in this study, the stigmergy mechanism is specifically incorporated into the proposed distributed learning algorithm to achieve effective collaboration between agents while allowing each agent to retain efficient RL ability.

III. System Framework

We present the framework of the SIRL mechanism in Fig. 1. In particular, each agent is designed to learn independently during the training phase and is required to act automatically during the DC execution phase. Note that each agent can only observe the environment partially and locally. Therefore, as illustrated at the bottom of Fig. 1, we deploy stigmergy as an indirect communication bridge between independent learning agents, which represents an explicit feedback loop between agents and the medium. Furthermore, as illustrated in the center of Fig. 1, a conflict-avoidance mechanism is deployed among adjacent agents to further reduce their behavioral localities. At the top of Fig. 1, we introduce the federal training method by appending a virtual agent to effectively optimize the neural network of each agent. To more easily clarify the framework and process of the SIRL mechanism, we primarily consider the flight formation of UAVs to monitor a specific target area as a typical example. We assume that multiple UAVs (i.e., agents) are flying and collaborating to form a specific team shape to hover above the final target area and that each UAV must determine its flying policy independently based on its limited local environmental information, such as its relative position to other UAVs. In this scenario, the proposed framework can be applied to improve the learning effectiveness in terms of the final similarity between the target team shape and the shape formed by the UAVs or the aggregated cost of the UAVs.

A. Stigmergy Mechanism

In general, stigmergy consists of four main components (i.e., medium, trace, condition, and action), which together form a feedback loop between agents and their surrounding environment [22], as illustrated in Fig. 1. Note that the medium can also be regarded as part of the entire environment. Here, the environment and medium are represented separately by different parts in Fig. 1 to distinguish the traditional learning environment in RL from that utilized in the stigmergy mechanism. In addition, a trace (i.e., digital pheromone) is normally left by an agent in the medium as an indicator of the environmental change resulting from its action. Several traces left by different agents in the medium can diffuse and further mix in a spontaneous manner [26]. Then, the variation pattern of these digital pheromone traces is returned as the interinfluence to other agents for their subsequent actions, while the amplitude of this interinfluence is largely related to the interdistance between agents [30].

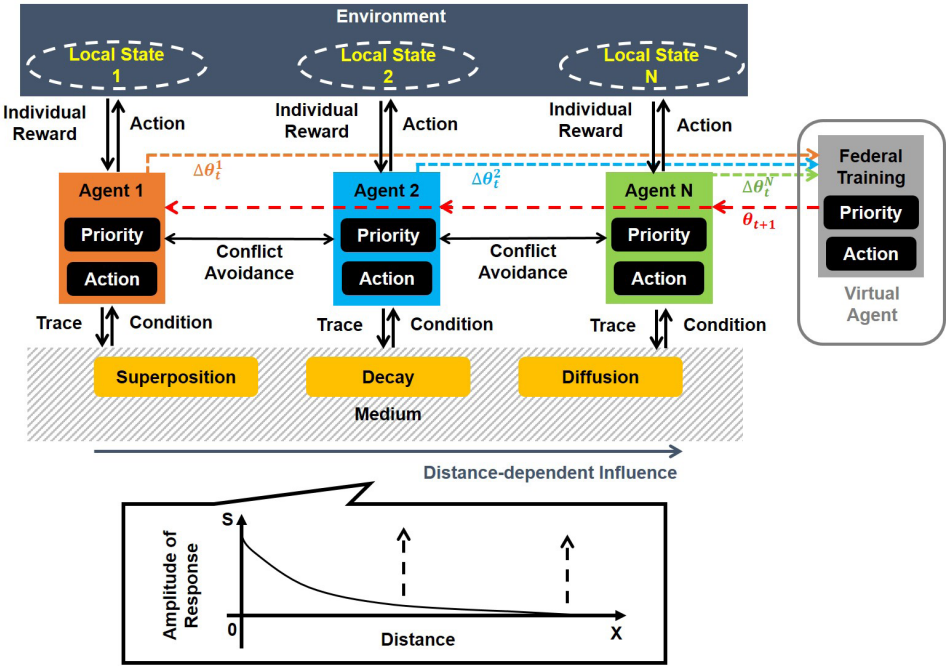


Fig. 1. Framework of SIRL.

In addition, stigmergy can also serve as a potential solution to the decomposition of a global objective. For instance, the construction of a termite nest requires the cooperation of the entire colony, which usually passes through several generations. However, a single termite in this colony is unaware of the global objective (i.e., building a termite nest), due to the limited size of its brain. The cooperative mechanism utilized by this colony must be able to decompose the global objective into several small tasks that can be perceived by a single termite. Therefore, in addition to realize the indirect communication, stigmergy can also achieve decomposition of the global objective implicitly to help obtain individual reward in multiagent collaboration.

B. Conflict-Avoidance Mechanism

Conflicts between the actions of different agents may arise from the competition for limited task resources during multiagent collaboration. To reduce the number of conflicts and minimize the amount of data that agents need to transmit during the collaboration process at the same time, we propose a conflict-avoidance mechanism by calculating action priorities for different agents. As illustrated in Fig. 2, Agent 1 (or 2) represents an independent learning agent in Fig. 1. In particular, there are two different internal neural network modules in each agent: the evaluation module and the behavior module. The evaluation module is used to efficiently calculate the action priority of an agent at the current local state, which is further used to compete for the action opportunity. The behavior module is used to select appropriate actions for an agent according to the local input state when obtaining the action opportunity. Note that the action policy from the behavior module may be self-centered, signifying that each agent may be trapped in its local optimality while ignoring the global

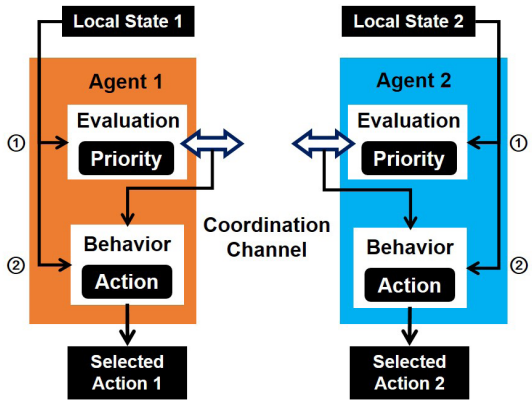


Fig. 2. Intuitive schematic of the conflict-avoidance mechanism.

objective. Thus, the conflict-avoidance mechanism enabled by the evaluation module can help eliminate self-centered action policies to facilitate collaboration.

With the conflict-avoidance mechanism, each agent passes through two steps to determine the action at the current state. As illustrated in Fig. 2, in the first step, the input related to the current state of an agent is first sent to the evaluation module to calculate the action priority. The action priority value is then compared with that of nearby agents through a coordination channel, and a priority list within a small range can be obtained. In the second step, the same input can be sent to the behavior module for selecting appropriate actions only when an agent has the highest action priority in the first step. Otherwise, the agent remains silent.

C. Federal Training Method

Despite its success in the training of deep Q-learning networks [12], [49], experience replay may not be as effective

in an MAS. Since a single agent in an MAS faces different tasks or situations, samples stored in the experience pool may not adapt to these changes. Recently, an asynchronous method for the advantage actor-critic (A2C) algorithm has been proposed and its advantages have been verified through training human-level players for Atari games [13], [50], [51]. Here, we further propose a federal training method by improving the asynchronous A2C (i.e., A3C) algorithm to synchronously optimize the neural network of each agent through the average optimization of neural network gradients.

In this federal training method, each agent attempts to optimize its neural network not only through self-related experience but also through neural network gradients from other collaborative teammates. Suppose that the number of active agents participating in the collaboration at time step t is N_t, and $N_t \leq N$, where N denotes the total number of agents. Moreover, N_t can also represent the number of agents that have obtained action opportunities through the conflict-avoidance mechanism. Gradients from these participating agents naturally form a mini-batch whose functionality is similar to that in the experience replay and maybe even more uncorrelated because they are sampled from different situations. Therefore, as illustrated in the right part of Fig. 1, a virtual agent is designed and added into SIRL, aiming to collect various local gradients of the involved agents for the average optimization. This virtual agent has the same neural network structure as other agents but takes no action.

TABLE I
MAIN NOTATIONS USED IN THIS ARTICLE

Notation	Definition
N	Number of agents
N_t	Number of active agents at time t
$\mathcal{S}$	Local state space
$\mathcal{A}$	Individual action set
s	Local state
a	Selected action
π	Action policy
r	Individual reward
R	Accumulated individual reward
$\widetilde{r}$	Deterministic individual reward
$\widetilde{R}$	Accumulated deterministic individual reward
l	Learning rate
e	State value network within Evaluation Module
$\overline{e}$	Target state value network within Evaluation Module
p	Policy network within Behavior Module
b	State value network within Behavior Module
$\overline{b}$	Target state value network within Behavior Module
V	State value
θ	Neural network parameters
d	Distance between agent and attractor
D	Distance-dependent function
ε	Amount of digital pheromone
ξ	Attractor set
r_m	Individual reward from medium
v	Virtual agent
SI	Similarity

IV. SIRL ALGORITHMS FOR MULTIAGENT COLLABORATION

In this section, we provide additional details and mathematical formulations regarding the three above-mentioned mechanisms. We assume that N agents are located in an area space and are dedicated to collaboratively fulfilling a specific task (e.g., UAVs to form a particular position shape). As illustrated in Figs. 1 and 2, at each time step t, each agent i receives a local state $s_t^{(i)}$ from the environment's local state space $\mathcal{S}$. Depending on the local state, an action $a_t^{(i)}$ is selected from the individual action set $\mathcal{A}$ by each of N_t agents. After the selected action is performed, an individual reward $r_t^{(i)}$ is returned to each participating agent to calculate the neural network gradients according to the loss function, $\text{Loss}(\theta_t^{(i)})$, so as to adjust its neural network parameters $\theta_t^{(i)}$. Specifically, the local state received by each agent is impacted by some particular environmental attractors, which are selected according to the digital pheromone, as discussed in Section IV-A. Furthermore, because the selected action of each agent is influenced by its priority at the current local state, we discuss the related calculation method of this action priority in Section IV-B. Finally, we present the entire training process of the two neural network modules of each agent using the federal training method. The main notations used in this article are listed in Table I.

A. Attractor Selection Within the Stigmergic State Space

The effectiveness of stigmergy can be improved by utilizing the digital pheromone [2]. Unlike chemical pheromones left by ants in natural colonies, digital pheromones generated by intelligent agents are virtual and can be represented by several informative records in memory with attributes such as value, time, and location [52]. Furthermore, during swarm foraging, most ants are attracted to these chemical signals, whose distribution naturally forms a pheromone map between the food destination and nest. Similarly, a digital pheromone map that contains the distribution of digital pheromones for providing relevant information of the state space in the entire activity area is also deployed in SIRL. The entire digital pheromone map can be stored in a centralized manner in the virtual agent or can be split into several parts and stored in a DC manner in several specified agents [2]. Moreover, the digital pheromone map is continuously updated by mutual communication between its maintainer (e.g., virtual agent) and other agents in the activity area.

In SIRL, the digital pheromone is regarded as the trace left by an agent in the medium, while the digital pheromone map is regarded as the corresponding medium. As indicated by the dynamic features of the medium in Fig. 1, the digital pheromone experiences different evolution processes in the medium, which should be carefully designed to make the returned conditions more effective for each agent. Inspired by phenomena in natural colonies, in which chemical pheromones left by different ants can be superposed to increase the total influence, we model the accumulation of digital pheromones with different sources as a linear superposition. Moreover, instead of being restricted to a single area, the digital pheromone with a larger amount will diffuse into surrounding areas. Furthermore, the amount of digital pheromone will decay over time. Therefore, the maintainer of the digital pheromone map should follow the three key

principles: 1) linearly superposing digital pheromones with different sources in the same area, 2) diffusing digital pheromones into surrounding areas at a small scale with a fixed diffusion rate after a new digital pheromone has been left, and 3) decreasing the amount of digital pheromone at positions already occupied by agents with a fixed decay rate. Note that the decay and diffusion rate are both constants between 0 and 1.

With a digital pheromone map for the stigmergic state space, each agent can sense the amount of digital pheromone within a certain range. Without loss of generality, we take the aforementioned UAV application scenario as an example. Here, we regard any block (i.e., unit area) filled with the digital pheromone as an attractor in the local environment, which has an attractive effect on any nearby mobile agent for efficiently observing the local state space. Similar to the classical ACO algorithm, within the local state space, each intelligent agent can independently perform its action (i.e., approaching an attractor) by selecting a suitable attractor within its sensing range from several potential attractor candidates, which can be expressed by

$$C_{i,j}(t) = \frac{D\big(d_{i,j}(t)\big) \cdot \varepsilon_j(t)}{\sum_{j \in \xi_i(t)} D\big(d_{i,j}(t)\big) \cdot \varepsilon_j(t)} \tag{1}$$

where $C_{i,j}(t)$ is the probability of agent i selecting attractor j; $\varepsilon_j(t)$ is the total amount of digital pheromone in attractor j at time t; $\xi_i(t)$ is the set of attractors within the sensing range of agent i; $d_{i,j}(t)$ is the Euclidean distance between agent i and attractor j; and $D(\cdot)$ is a monotone function used to reduce the effect of the digital pheromone as the interdistance $d_{i,j}(t)$ increases [30], which is intuitively illustrated at the bottom of Fig. 1. This attractor selection method is inspired by the classical ACO algorithm [35]. However, we use function $D(\cdot)$ to replace the heuristic factor, which is commonly used to consider the constraints of an optimization problem, of the original ACO algorithm to improve the selection process. Function $D(\cdot)$ can cause an agent to pay more attention to nearby influential attractors while avoiding the so-called ping-pong effect in the local environment. In addition, selecting attractors in a stochastic manner can cause agents to perform actions (i.e., approaching target positions) with a smaller amount of digital pheromone, and can prevent a large number of agents from crowding in a small local environment.

The location of the selected attractor serves as an informative part of the input local state for each agent. Depending on the two neural network modules of each agent, an action is selected according to the input local state. Furthermore, following the stigmergic principle, any agent, which has performed the selected action accordingly leaves an additional digital pheromone in the medium, to provide new condition information for the subsequent selection of attractors. This process can be expressed by

$$\varepsilon_j(t+1) = \begin{cases} \varepsilon_j(t) + a_1, & \text{if } \varepsilon_j \text{ is in the labeled area} \\ \varepsilon_j(t) \cdot b_1, & \text{otherwise} \end{cases} \tag{2}$$

where a_1 represents the fixed amount of digital pheromone left by an agent at a time, and b_1 is a discount constant between 0 and 1 that helps gradually remove useless attractors. The labeled area indicates that the agent has partially fulfilled a small task.

B. Action Priority Determination

In this subsection, we discuss the methods to calculate the action priority for the conflict-avoidance mechanism. Corresponding to the two neural network modules in Fig. 2, we exploit two algorithms to optimize their parameters, respectively. First, for the evaluation module, we define an internal state value network whose output is the expected accumulated deterministic individual reward at $s_t^{(i)}$

$$\begin{aligned} V_e\Big(s_t^{(i)}; \theta_e^{(i)}\Big) &= \mathbb{E}\Big[\widetilde{R}_t^{(i)} \Big| s_t^{(i)} = s, a_t^{(i)} = \big(a; \theta_p^{(i)}\big)\Big] \\ a_t^{(i)} &= \arg\max_{a \in \mathcal{A}} \pi\Big(s_t^{(i)}, a; \theta_p^{(i)}\Big) \end{aligned} \tag{3}$$

where $s_t^{(i)}$ represents the local state observed by agent i at time t. Subscript e represents the state value network in the evaluation module, and $\theta_e^{(i)}$ denotes the related parameters. Moreover, V_e is the state value of $s_t^{(i)}$, which is regarded as the action priority of agent i at the current local state. $a_t^{(i)}$ denotes the selected action toward the chosen attractor of agent i at time t. Furthermore, subscript p represents the policy network in the behavior module, while π represents its action policy. Note that the evaluation of the action priority at $s_t^{(i)}$ is based on the deterministically executed action at the same local state, and the returned individual reward during the training process of the state value network in the evaluation module is also deterministic. Therefore, we define $\widetilde{R}_t^{(i)}$ as the accumulated deterministic individual reward of agent i at time t, which is calculated by

$$\widetilde{R}_t^{(i)} = \begin{cases} \widetilde{r}_t^{(i)} + \gamma_2 \cdot V_{\overline{e}}\Big(s_{t+1}^{(i)}; \theta_{\overline{e}}^{(i)}\Big), & \text{if } s_{t+1}^{(i)} \text{ is nonterminal;} \\ \widetilde{r}_t^{(i)}, & \text{otherwise} \end{cases} \tag{4}$$

where γ_2 is a discount factor, and $\widetilde{r}_t^{(i)}$ is the returned deterministic individual reward. Subscript $\overline{e}$ represents the target state value network of agent i, whose parameters and output are represented by $\theta_{\overline{e}}^{(i)}$ and $V_{\overline{e}}$, respectively. The target state value network is used to calculate the state value of the new input local state $s_{t+1}^{(i)}$ and further help calculate the accumulated deterministic individual reward, as illustrated in the first line of (4). Moreover, the target state value network is almost the same as the state value network in the evaluation module except that its parameters are periodically copied from the original state value network [12]. Finally, the loss function of the state value network in the evaluation module can be expressed as

$$\text{Loss}\big(\theta_e^{(i)}\big) = 0.5 \cdot \Big[\widetilde{R}_t^{(i)} - V_e\Big(s_t^{(i)}; \theta_e^{(i)}\Big)\Big]^2. \tag{5}$$

For the behavior module, we use the A2C algorithm to optimize its parameters [50]. In particular, there is a policy and a state value network in the behavior module that share the same input local state from the local environment, including the stigmergic state space with attractor indexing. Their loss

functions, which are used to calculate the gradients of the neural network parameters, are, respectively, expressed as

$$\text{Loss}\left(\theta_p^{(i)}\right) = -\log\pi\left(a_t^{(i)}\middle|s_t^{(i)};\theta_p^{(i)}\right)\left(R_t^{(i)} - V_b\left(s_t^{(i)};\theta_b^{(i)}\right)\right) \tag{6}$$

$$\text{Loss}\left(\theta_b^{(i)}\right) = 0.5\cdot\left[R_t^{(i)} - V_b\left(s_t^{(i)};\theta_b^{(i)}\right)\right]^2 \tag{7}$$

where subscripts p and b represent the policy and state value network in the behavior module, respectively. The local state $s_t^{(i)}$ is sent to two different neural networks (i.e., the policy network and the state value network) to calculate the corresponding output. For the policy network, the output is the action policy π. We select each action in a probabilistic manner during the training phase of the policy network; thus, the returned individual reward is stochastic. For the state value network, the output is the state value of $s_t^{(i)}$ (i.e., V_b), which is used to calculate the advantage (i.e., $R_t^{(i)} - V_b(s_t^{(i)};\theta_b^{(i)})$) to speed up the convergence of the action policy in the parallel policy network. Accordingly, $R_t^{(i)}$ is the accumulated individual reward of agent i, which can be expressed as

$$R_t^{(i)} = \begin{cases} r_t^{(i)} + \gamma_1 \cdot V_{\overline{b}}\left(s_{t+1}^{(i)};\theta_{\overline{b}}^{(i)}\right), & \text{if } s_{t+1}^{(i)} \text{ is nonterminal;} \\ r_t^{(i)}, & \text{otherwise} \end{cases} \tag{8}$$

where $r_t^{(i)}$ is the individual reward received by agent i at time t, and γ_1 is a discount factor that can normally be set to a constant between 0 and 1 (e.g., 0.9). Similarly, subscript $\overline{b}$ represents another target state value network of agent i. Note that under the conflict-avoidance mechanism, any agent with the highest action priority will have the opportunity to perform the selected action. Because the execution order of different actions is arranged by their accumulated deterministic individual rewards, this estimation method of action priority is expected to obtain a large global reward.

C. Training Neural Network Modules

Under the conflict-avoidance mechanism, the evaluation and behavior modules work together to obtain an individual reward and forge a cooperative relationship. Furthermore, according to (3), the estimation of the action priority is also based on the deterministic action policy of the behavior module. Therefore, there are two successive sessions in each training round (i.e., training epoch). In the left part of Fig. 3, we freeze the parameters of the behavior module in the training session of the evaluation module. Similarly, we freeze the parameters of the evaluation module in the training session of the behavior module, which is indicated in the right part of Fig. 3. The federal training method is applied in both sessions, where a virtual agent is also employed. Furthermore, we set a terminal condition for both training sessions. Each session is stopped as the number of internal time steps reaches its maximum, or the global reward is positive. The performance of multiagent collaboration is represented by the global reward, whose improvement can implicitly indicate the convergence of the current training of the neural network modules. In Fig. 3, t denotes the index of time steps while $t_{\max}$ represents its maximum. t is set to 0 at the beginning of a training session. During each training round, a sample is sent to both sessions to optimize different neural network modules.

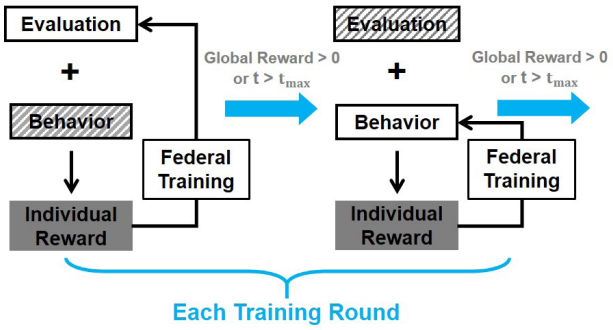


Fig. 3. Two successive sessions in each training round.

In MARL, the global reward can typically be used to indicate the performance of multiagent collaboration. However, it usually cannot be directly used as the individual reward received by an agent, as the global reward is determined by a series of actions from different agents. In general, the individual reward can be obtained through a reasonable decomposition of the global objective. In SIRL, after the selected action is performed, each agent receives an individual reward from the medium. Because the objective of each agent is to approach its selected attractor in the stigmergic state space, we define the returned individual reward as the Euclidean measure (i.e., interdistance change) between the position of each agent and its selected attractor, which can be expressed as

$$r_m^{(i)}(t) = \rho_1 \cdot \max\left(\left[d_{i,j}(t-1) - d_{i,j}(t)\right], 0\right) \tag{9}$$

where the subscript m represents the medium in SIRL, and $r_m^{(i)}(t)$ represents the individual reward received by agent i from the medium at time t. In addition, ρ_1 is a scalar factor, while $d_{i,j}(t)$ represents the interdistance between agent i and its selected attractor j, where $j \in \xi_i(t-1)$. Note that the reward $r_m^{(i)}(t)$ is obtained due to the implementation of the digital pheromone, which indicates the decomposition process of the stigmergy mechanism for the global objective. In particular, during each training round, $r_m^{(i)}(t)$ is set to $\widetilde{r}_t^{(i)}$ in (4) during the training session of the evaluation module, and $r_t^{(i)}$ in (8) during the training session of the behavior module.

Furthermore, as illustrated in Figs. 1 and 3, the federal training method is used to optimize the parameters of different neural network modules through the average optimization

$$\theta_{t+1}^{(v)} = \theta_t^{(v)} + v_t^{(v)} \tag{10}$$

$$v_t^{(v)} = \rho \cdot v_{t-1}^{(v)} - l_t \cdot \frac{1}{N_t}\sum_{i=1}^{N_t}\frac{\partial \text{Loss}\left(\theta_t^{(i)}\right)}{\partial \theta_t^{(i)}} \tag{11}$$

where the superscript v represents the virtual agent. At time step $t = 0$, $v_0^{(v)}$ is set to 0. ρ is a momentum factor, while l_t denotes the learning rate of parameters. The federal training method is inspired by the A3C algorithm; however, it applies a synchronous updating method, which has been demonstrated

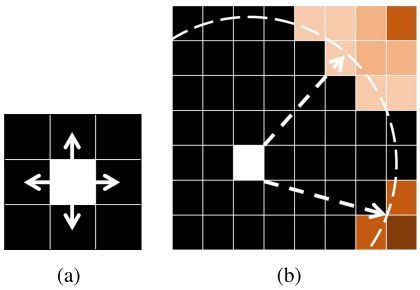


Fig. 4. Moving mode and digital pheromone sensing range of a single mobile agent in the activity area.

to have a lower error bound [53]. Moreover, we add a momentum factor in the updating process to accelerate the convergence. Because a virtual agent has the same neural network structure as other agents, it can be used to optimize the parameters of different neural network modules. For simplicity, we use $\theta_t^{(i)}$ in (11) to represent the parameters of either the evaluation or behavior module for agent i, and use $\theta_t^{(v)}$ to represent the parameters of the same neural network module of the virtual agent. In particular, for the current training of the neural network module, gradients of the involved neural network parameters are first calculated in these N_t agents according to the corresponding loss function, Loss$(\theta_t^{(i)})$, and are then sent to the virtual agent for the average optimization. Finally, the newly calculated parameters $\theta_{t+1}^{(v)}$ are sent back to all agents to update their neural network modules. The entire SIRL algorithm can be divided into a training and testing part, which are detailed in Algorithms 1 and 2, respectively.

V. Experimental Simulation Settings and Numerical Results

In this section, we create a simulation scenario in which a certain number of mobile agents (e.g., UAVs) in an activity area automatically form a specified team shape. In this simulation scenario, the target team shape is transferred through the binary normalization of an image that is taken from the standard Modified National Institute of Standards and Technology (MNIST) data set [54]. The MNIST data set is a large set of handwritten digits that is commonly used for training various image processing systems. In addition, it is widely used for training and testing in the field of machine learning. Specifically, each image in this data set has a standard size (i.e., 28×28). As illustrated in Fig. 4, an agent is represented by a nonzero block (i.e., a pixel, as in a digitized binary image). The white block (i.e., unit area) represents an agent, while the black blocks represent the potential positions the mobile agent can visit. We set the distance between any two adjacent blocks to 1. In the beginning, all mobile agents are distributed randomly across the entire activity area (i.e., the entire image). The experimental objective in this scenario can be mapped to many real-world multiagent collaboration tasks, such as the flight formation of UAVs monitoring a certain target area.

First, the simulation settings for the involved agents and the activity area are described as follows:

1) Each agent is supposed to move a block at each time step toward one of four directions: 1) up, 2) down, 3) left, and 4) right.
2) Each agent can leave the digital pheromone in the current position and sense the amount of digital pheromones in the surrounding blocks within a certain range.
3) Each agent can identify whether the occupied position is labeled or unlabeled.
4) Each agent can sense the existence of neighbors in the up, down, left, and right directions to avoid a collision.
5) Each block in the activity area can only be occupied by one agent at a time.
6) The activity area can be classified as labeled and unlabeled, where the former corresponds to the target team shape to be formed.

The moving mode of a single agent is illustrated in Fig. 4(a). In Fig. 4(b), the white dotted line represents the boundary of the digital pheromone sensing range of that agent. Brown blocks indicate the presence of digital pheromones, and colors with different shades represent distinct amounts of the pheromone. In addition, the coordination channel for the conflict-avoidance exists between any central agent and its eight possible neighbors, which are also called Moore neighbors in mobile cellular automation [55]. In the experimental simulation, we use a Gaussian function to play the role of $D(\cdot)$ in (1), which can be expressed as

$$D\left(d_{i,j}(t)\right) = a_2 \cdot \exp\left(-\frac{\left(d_{i,j}(t) - b_2\right)^2}{2c_1^2}\right) \tag{12}$$

where a_2 represents the peak value and is set to 1; b_2 is the mean value and is set to 0; and c_1 represents the standard deviation and is normally set to 0.25.

The similarity, SI, is calculated by the ratio of the number of agents that end up in the labeled area to the total number of agents in the activity area, which is further determined by the number of nonzero pixels required to form the target shape. In the training part, we define the increase in similarity after each time step (i.e., ΔSI) as the global reward, which can be positive or negative. In addition, similar to the settings in [30], we use the position distribution of all agents after a certain number of iterations as the swarm's initial state, which can also be regarded as a sample. Approximately 7500 samples are extracted from different iterations in this simulation during the formation of the target shape "4." In particular, the new position distribution of all agents is returned after each time step to calculate both the global and individual rewards so as to further optimize the neural network modules. Note that the neural network modules trained by this sample set can also be used in the testing process to form another shape (e.g., "1," "2," "0," "6," and "8").

Furthermore, the evaluation and behavior modules share the same input local state, which is denoted by a vector with seven elements. The first four elements represented by bit numbers are used to confirm whether there are neighbors in the following four adjacent positions: up, right, down, and left. The fifth and sixth elements are used to describe the relative position of the selected attractor in a two-dimensional plane, and the

Algorithm 1 Training Part of SIRL

Input: Agents with the neural network modules, the sample set, the target shape, number of training rounds;
Output: Agents with the well-trained neural network modules;

Initialize the whole activity area, the digital pheromone map, the labeled area, the neural network modules within each agent, diffusion rate, decay rate, $t_{\max}$, time step t, the range of sensing digital pheromone, the range of coordination channel;
for each training round **do**
  //* Training session of Evaluation Module *//
  Select a sample randomly from the sample set and initialize the location of agents according to the selected sample;
  while $t \leq t_{\max}$ **do**
    for each agent **do**
      Select an attractor according to (1) and form the input local state;
      Send the local state to Evaluation Module;
      Send the same local state to Behavior Module and select an action with the largest probability;
      Perform the action;
      Modify the digital pheromone at current position according to (2);
      if the extra digital pheromone is left **then**
        Diffuse the digital pheromone to the surrounding areas with the fixed diffusion rate;
        Superpose the amount of digital pheromone at the same position linearly;
      end if
      Calculate the individual reward according to (9);
      Select a new attractor according to (1) and form the new input local state;
    end for
    Decay the amount of digital pheromone at positions already occupied by agents with the fixed decay rate;
    if the calculated global reward > 0 **then**
      Break;
    else
      for each agent **do**
        Calculate the gradients $\frac{\partial Loss(\theta_e^{(i)})}{\partial \theta_e^{(i)}}$ of the state value network within Evaluation Module according to (4) - (5) with the new input local state;
        Send the calculated gradients to the virtual agent;
      end for
      The virtual agent receives the gradients from agents and optimizes the internal state value network within Evaluation Module according to (10) - (11);
      The virtual agent sends back the calculated parameters $\theta_{t+1}^{(v)}$ to all agents;
      Each agent updates the state value network within Evaluation Module with $\theta_{t+1}^{(v)}$;
    end if
  end while
  //* Training session of Behavior Module *//
  Initialize the location of agents according to the selected sample;
  while $t \leq t_{\max}$ **do**
    for each agent **do**
      Select an attractor according to (1) and form the input local state;
      Send the local state to Evaluation Module and calculate the action priority;
      Send out the action priority through the coordination channel and receive the returned priority list;
      if the own action priority is the largest **then**
        Send the same local state to Behavior Module and select an action in a probabilistic manner;
        Perform the action;
        Modify the digital pheromone at current position according to (2);
        if the extra digital pheromone is left **then**
          Diffuse the digital pheromone to the surrounding areas with the fixed diffusion rate;
          Superpose the amount of digital pheromone at the same position linearly;
        end if
        Calculate the individual reward according to (9);
        Select a new attractor according to (1) and form the new input local state;
      end if
    end for
    Decay the amount of digital pheromone at positions already occupied by agents with the fixed decay rate;
    if the calculated global reward > 0 **then**
      Break;
    else
      for each agent getting the action opportunity **do**
        Calculate the gradients $\frac{\partial Loss(\theta_p^{(i)})}{\partial \theta_p^{(i)}}$ and $\frac{\partial Loss(\theta_b^{(i)})}{\partial \theta_b^{(i)}}$ of the policy and state value networks within Behavior Module according to (6) - (8) with the new input local state;
        Send the calculated gradients to the virtual agent;
      end for
      The virtual agent receives the gradients from agents and optimizes the internal policy and state value networks within Behavior Module according to (10) - (11);
      The virtual agent sends back the calculated parameters $\theta_{t+1}^{(v)}$ to all agents;
      Each agent updates the policy and state value networks within Behavior Module with $\theta_{t+1}^{(v)}$;
    end if
  end while
end for
Return agents with the well-trained neural network modules;

Algorithm 2 Testing Part of SIRL

Input: Agents with the well-trained neural network modules, the location of each agent, the target shape, number of iterations;
Output: The final similarity;

Initialize the whole activity area, the digital pheromone map, the labeled area, diffusion rate, decay rate, time step t, the range of sensing digital pheromone, the range of coordination channel;
for each iteration **do**
  for each agent **do**
    Select an attractor according to (1) and form the input local state;
    Send the local state to Evaluation Module and calculate the action priority;
    Send out the action priority through the coordination channel and receive the returned priority list;
    if the own action priority is the largest **then**
      Send the same local state to Behavior Module and select an action with the largest probability;
      Perform the action;
      Modify the digital pheromone at current position according to (2);
      if the extra digital pheromone is left **then**
        Diffuse the digital pheromone to the surrounding areas with the fixed diffusion rate;
        Superpose the amount of digital pheromone at the same position linearly;
      end if
    end if
  end for
  Decay the amount of digital pheromone at positions already occupied by agents with the fixed decay rate;
end for
Calculate the final similarity according to the target shape;
Return the final similarity;

seventh element is used to confirm whether the current occupied position is labeled or unlabeled. The individual action set $\mathcal{A}$ contains five different actions: up, right, down, left, and stop. However, since the local state received by each agent contains the relative positions of adjacent agents, the recorded new input local state $s_{t+1}^{(i)}$ may be inaccurate at time step $t+1$ due to the possible movement of adjacent agents at time step t, thus leading to inaccurate estimation of $V_{\overline{e}}(s_{t+1}^{(i)}; \theta_{\overline{e}}^{(i)})$ in (4). In the following simulation, we use different values of γ_2 to test this phenomenon.

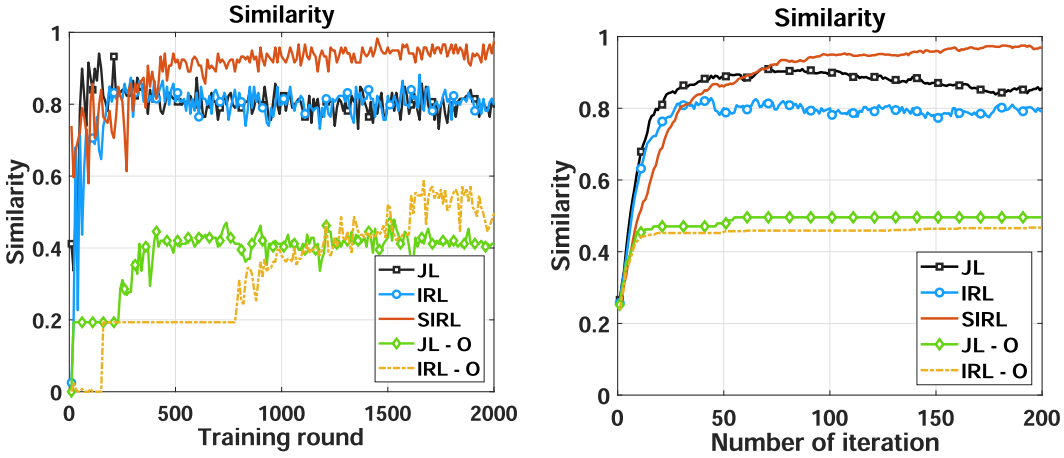
We first present the convergence of SIRL and compare it with that of other typical methods. The first method, joint learning (JL), attempts to learn the optimal behavior from the joint information using only the behavior module, whose input is a cascaded vector containing all the input local states of surrounding agents within the Moore neighborhood, while the conflict-avoidance mechanism is disabled. The second method, IRL, has almost the same settings as JL except that its input vector only contains the self-related local state. The third method, joint learning-origin (JL-O), and the fourth method, IRL-origin (IRL-O), are modified from the JL and IRL methods, respectively, by further disabling the stigmergy mechanism and replacing the attractors by the exact coordinates of the agents. In this situation, each agent receives a nonzero individual reward only when it enters the labeled area, and the global reward is also considered afterward. The received individual reward in this situation is indicated in Table II. Here, the transition $0 \rightarrow 1$ represents an agent moving from the unlabeled area to the labeled area. a_3 and b_3 are both positive constants.

TABLE II
RECEIVED INDIVIDUAL REWARD FOR JL-O AND IRL-O

Transition	Reward
Unlabeled area to Unlabeled area: $0 \rightarrow 0$	0
Unlabeled area to Labeled area: $0 \rightarrow 1$	a_3
Labeled area to Unlabeled area: $1 \rightarrow 0$	0
Labeled area to Labeled area: $1 \rightarrow 1$	$b_3 \cdot \max(\triangle SI, 0)$

The training and testing performance of the above-mentioned five methods is presented in Fig. 5, and an intuitive illustration of the formed shapes with respect to the iteration index in SIRL is presented in Fig. 6. In Fig. 5(a), the neural network modules are tested every 10 training rounds. It can be observed from Fig. 5 that there is an evident performance difference between methods with and without the stigmergy mechanism, as indicated by the curves of SIRL, JL, and IRL, since stigmergy can better decompose the global objective and achieve a higher final similarity. In addition, although the joint information is obtained in JL, it is simply treated as a noisy signal by each agent without appropriate utilization. Therefore, despite different inputs, the performance of JL and IRL is almost identical. Moreover, SIRL performs better than JL or IRL, benefiting from the conflict-avoidance mechanism, which can reconcile the contradictions caused by the actions of different agents and further improve the cooperation efficiency.

We also present the training performance of SIRL in Fig. 7 when the discount factor γ_2 takes different values. We can observe that the training performance of SIRL declines as the value of γ_2 increases to 1. In SIRL, based on the current local observations of the environment and the behavioral policy,



(a) (b)

Fig. 5. Training and testing performance of different methods in the task to form the shape "4".

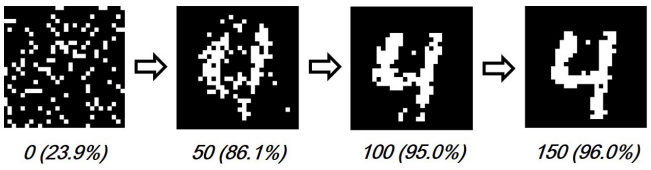


Fig. 6. Shapes formed at different iterations using SIRL.

the evaluation module of each agent is used to calculate the action priority among the action candidates. However, due to the existence and influence of other agents, the transition process from one state to another becomes unpredictable for each individual agent. In this situation, the reward mapped from the future state by the discount factor γ_2 becomes inaccurate. A smaller γ_2 can generally reduce this estimation error. Therefore, in the following simulation, we set the discount factor $\gamma_2 = 0$ in the training session of the evaluation module of each agent to limit the accumulation of $\widetilde{R}_t^{(i)}$ to only one step.

We further test the well-trained neural network modules using the above-mentioned five methods in the tasks to form shapes "1" and "2." The final similarity and the number of involved agents in each task are listed in Table III, where each value is an average of five replications. Note that the task complexity is largely related to the number of agents involved. With an increase in task complexity, the number of iterations required to reach convergence also increases, whereas the final similarity normally declines. In the first five rows of Table III, we can observe that the neural network modules fully trained in the task to form shape "4" can also be used to form other shapes (i.e., "1" and "2"), and the methods with stigmergy yield superior portability to those without it.

Next, we compare the performance with the centralized selection (CS) method presented in [30], in which a certain number of agents are selected synchronously at each iteration through a threshold-based function [26] to perform actions, while the other agents stop. Furthermore, we convert the CS method into a DC method called DC, in which each agent can automatically decide to perform an action or not through

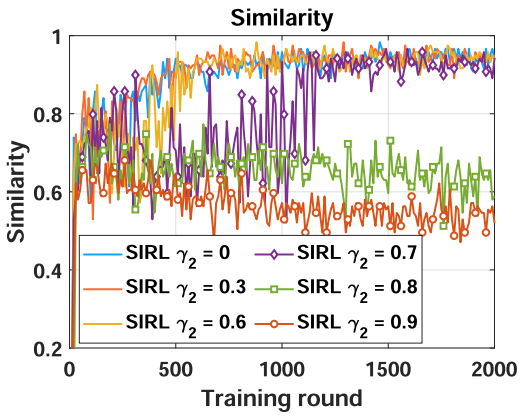


Fig. 7. Training performance of SIRL when γ_2 takes different values.

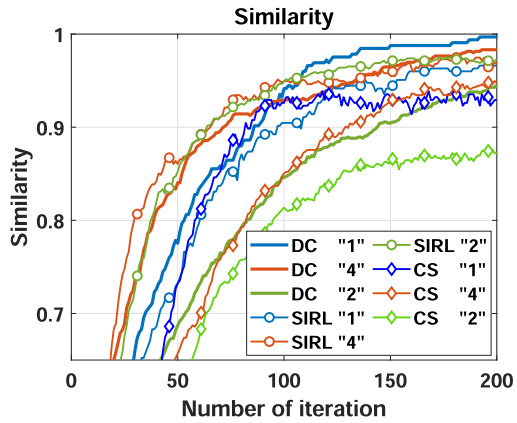
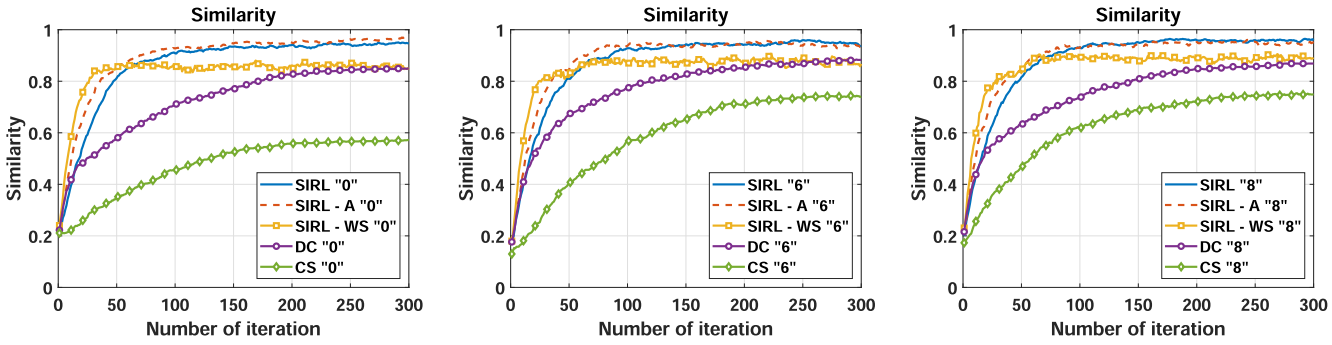


Fig. 8. Performance comparison between CS, DC, and SIRL in three different tasks.

comparison with surrounding neighbors. Thus, the number of active agents may vary within each task or iteration. Instead of competing based on the action priority, as in SIRL, agents in DC can directly determine their moving priorities in terms of the received rewards. In particular, an agent can move in the next time step $t+1$ only when it receives the maximum reward

TABLE III
FINAL SIMILARITIES USING DIFFERENT METHODS

Stigmergy	Conflict-Avoidance	Federal Training	Input Local State	Algorithm	Shape (Number of involved agents)					
					1 (65)	4 (119)	2 (161)	0 (179)	6 (128)	8 (163)
Applied	Applied	Applied	Self-related	SIRL	**96.6%**	**97.5%**	**97.5%**	**95.1%**	**96.0%**	**96.4%**
Applied	None	Applied	Joint	JL	84.3%	85.7%	82.1%	N/A	N/A	N/A
Applied	None	Applied	Self-related	IRL	84.9%	83.2%	81.9%	N/A	N/A	N/A
None	None	Applied	Joint	JL - O	35.4%	49.6%	41.0%	N/A	N/A	N/A
None	None	Applied	Self-related	IRL - O	33.9%	46.7%	37.9%	N/A	N/A	N/A
Applied	Applied	None	Self-related	DC	99.7%	98.3%	94.5%	84.8%	88.4%	86.9%
Applied	Applied	None	Self-related	CS	94.2%	95.1%	87.8%	57.2%	74.4%	75.3%
Applied	Applied	Applied	Self-related	SIRL - A	N/A	N/A	N/A	97.0%	95.4%	95.3%
Applied	None	Applied	Self-related	SIRL - WS	N/A	N/A	N/A	85.7%	87.8%	88.7%



(a) (b) (c)

Fig. 9. Performance comparison using different methods in the tasks to form shapes "0," "6," and "8".

TABLE IV
MANUALLY SET RECEIVED REWARD FOR THE DC METHOD

Number of neighbors	4	3	3	2	1	1	0
In the labeled area	*	1	0	*	1	0	*
Reward	0	4	12	8	8	12	12

in time step t within the comparison range, which has the same size as the Moore neighborhood. As illustrated in Table IV, the reward received by each agent in DC is carefully tuned and closely related to its surrounding conditions. Here, each agent should consider the existence of up to four nearest neighbors in the up, down, left, and right directions. The label "1" (or "0") in the second line signifies that this agent is in the labeled (or unlabeled) area, whereas "*" indicates to ignore this condition. We observe that agents in the unlabeled area appear to receive greater rewards and thus obtain more action opportunities. In addition, similar to CS, each agent in DC is also designed to approach its selected attractor in a circular path so as to make most agents in the unlabeled area move around the labeled area to accelerate the convergence [30].

In Fig. 8, we present a performance comparison between CS, DC, and SIRL in the task of forming three different shapes (i.e., "1," "4," and "2"). Their final similarities are listed in the first, sixth, and seventh rows of Table III. We can observe that the performance of CS and DC tends to decline sharply with an increase in task complexity and the number of involved agents. Moreover, in CS and DC, the value of received individual rewards and the moving manner of each agent must be determined manually, which is impractical in more complex scenarios. In contrast, the performance of SIRL reaches a level comparable to that of CS or DC, and achieves a superior convergence rate in more complex tasks, such as the task to form the shape "2."

Next, we increase the task complexity and add a ring structure into the shape to be formed so as to give it a more complex topology. The performance comparison between CS, DC, and SIRL in the tasks to form shapes "0," "6," and "8" is provided in Fig. 9. The numerical results are also listed in the first, sixth, and seventh rows of Table III. The neural network modules utilized in SIRL are still fully trained in the previous task to form shape "4." In Fig. 9, there is a large performance difference between CS or DC and SIRL. The reason is that it is generally easier for agents in the unlabeled area to win the moving opportunities; however, they move around the labeled area repeatedly in CS or DC. Consistent with our previous discussion, both the CS and DC methods perform better in less complex scenarios, such as the task to form the shape "1." However, they reach a bottleneck for complex shapes that contain ring structures because agents in the unlabeled area are blocked by other agents located at the edge of the labeled area.

We present the final shapes formed by the three methods in Fig. 10. It can be clearly observed that many agents in CS or DC are detained on the periphery of the labeled area, ultimately reducing the final similarity. In contrast, the performance of SIRL remains stable regardless of the task complexity, which benefits from the learning process of the federal training method. Compared with DC, the number of

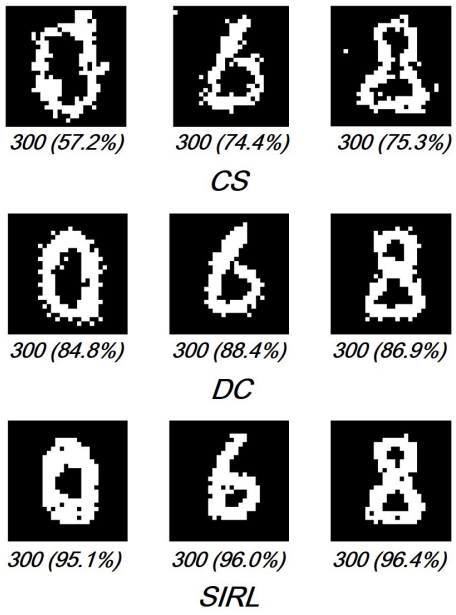


Fig. 10. Final shapes formed by CS, DC, and SIRL.

agents that are selected at each iteration must be predetermined in CS. However, the most appropriate number may depend on the specific task or iteration. Thus, CS would obtain lower final similarities or require a larger number of total steps in different tasks. In addition, the performance of CS also declines dramatically when the shape to be formed contains a ring structure. This phenomenon can also be observed in DC, as the two methods share the same manner of motion.

We further test the effect of different ranges of the coordination channel of the conflict-avoidance mechanism on the final performance based on SIRL. Instead of relying on the Moore neighbors, in the SIRL-A method, we reduce the maximum range of the coordination channel to 1; thus, only four neighbors are added in the comparison of action priority. Furthermore, in the SIRL-WS method, we disable the conflict-avoidance mechanism (i.e., well-trained evaluation module) and provide each agent with the action opportunity at each time step, which can be regarded as the maximum range of the coordination channel being set to 0. The performance of the two methods is illustrated in Fig. 9. The numerical results are also listed in the last two rows of Table III. It can be seen that the performance of SIRL-A reaches a similar level to that of SIRL, and achieves an even higher convergence rate since more agents can obtain action opportunities and participate in the task at an early stage. In addition, the curves of SIRL-WS in various tasks grow faster but stop at similarly lower levels. It can be concluded that the conflict-avoidance mechanism plays an important role in multiagent collaboration and can reduce the behavioral localities of agents and achieve a higher final similarity.

As an important metric, the number of total steps required by all involved agents to form the target shape should especially be considered. In Fig. 11, we present the total number of steps when the performance reaches convergence using four different methods. In the Oracle method, it is assumed that

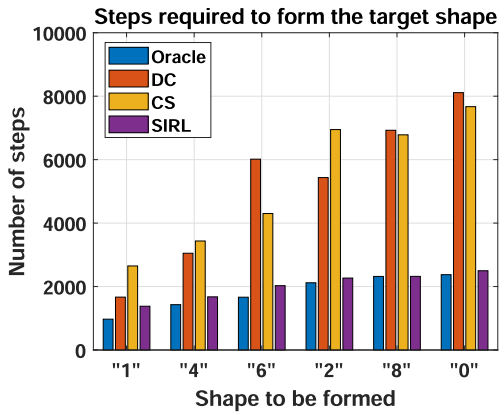


Fig. 11. Number of steps required using different methods.

the location of each agent and the target shape is known in advance. Therefore, for each vacancy in the labeled area, the nearest agent is moved in sequence to fill it greedily. This scheme represents a simple but effective control method in the extraordinary case in which all information is known, even though the greedy algorithm does not necessarily produce the optimal result. In Fig. 11, the formed shapes are arranged from small to large based on the number of involved agents. There is a trend that as the number of involved agents increases, the number of total steps required also increases. We can observe that the performance of SIRL reaches a level comparable to that of Oracle. Moreover, since the agents are controlled in parallel, SIRL may spend less time completing tasks in real scenarios.

VI. Conclusion

This study utilizes the advantages of the stigmergy mechanism rooted in a natural MAS and contributes to integrating stigmergic collaboration into the DRL-based decision-making process of each involved intelligent agent. In particular, in our proposed SIRL algorithm, various agents are coordinated through the stigmergic functionality and conflict-avoidance mechanism in fulfilling the corresponding tasks. As an enhancement to stigmergy, the proposed conflict-avoidance mechanism can further improve the coordination performance, especially when agents do not fully cooperate. In addition, the RL process of each agent is strengthened through the proposed federal training method. Compared with a traditional distributed learning algorithm, our learning method allows agents to optimize their internal neural networks while maintaining their external interaction processes required for effective interagent collaboration. Furthermore, due to the introduction of the stigmergic RL process, our proposed SIRL scheme can be further applied to more complex cooperative tasks for which the agent's behavioral policy cannot be predetermined. The results of numerical simulations verify the effectiveness of our proposed scheme with significant improvements in both efficiency and scalability.

In future work, we plan to implement the proposed SIRL scheme in real-world application scenarios. As a preliminary step, our scheme has recently been implemented on multiple

mobile robots in team formation tasks, which are similar to those presented in this article [56], and achieve satisfactory results. We also plan to investigate the performance of the SIRL scheme in other coordination scenarios, such as the control of autonomous vehicles in intelligent urban transportation. In addition, determining how to improve the network training efficiency while considering the communication delay and cost among various agents is also a valuable direction that we will consider in future research.

REFERENCES

[1] J. Schwarzrock, I. Zacarias, A. L. C. Bazzan, R. Q. de Araujo Fernandes, L. H. Moreira, and E. P. de Freitas, "Solving task allocation problem in multi unmanned aerial vehicles systems using swarm intelligence," *Eng. Appl. Artif. Intell.*, vol. 72, pp. 10–20, Jun. 2018.

[2] H. V. D. Parunak, S. A. Brueckner, and J. Sauter, "Digital pheromones for coordination of unmanned vehicles," in *Proc. Int. Workshop Environ. Multi-Agent Syst.*, in Lecture Notes in Computer Science, vol. 3374, 2005, pp. 246–263.

[3] M. G. C. A. Cimino, A. Lazzeri, and G. Vaglini, "Combining stigmergic and flocking behaviors to coordinate swarms of drones performing target search," in *Proc. 6th Int. Conf. Inf., Intell., Syst. Appl. (IISA)*, Corfu, Greece, Jul. 2015, pp. 1–6.

[4] P. Valckenaers, H. V. Brussel, M. Kollingbaum, and O. Bochmann, "Multi-agent coordination and control using stigmergy applied to manufacturing control," in *Proc. 9th ECCAI Adv. Course ACAI, Agent Link's 3rd Eur. Agent Syst. Summer School Multi-Agent Syst. Appl.*, Berlin, Germany, Jul. 2001, pp. 317–334.

[5] Hadeli, P. Valckenaers, M. Kollingbaum, and H. Van Brussel, "Multi-agent coordination and control using stigmergy," *Comput. Ind.*, vol. 53, no. 1, pp. 75–96, Jan. 2004.

[6] J. Werfel and R. Nagpal, "Extended stigmergy in collective construction," *IEEE Intell. Syst.*, vol. 21, no. 2, pp. 20–28, Mar. 2006.

[7] B. Guo, C. Chen, D. Zhang, Z. Yu, and A. Chin, "Mobile crowd sensing and computing: When participatory sensing meets participatory social media," *IEEE Commun. Mag.*, vol. 54, no. 2, pp. 131–137, Feb. 2016.

[8] A. L. Alfeo, M. G. C. A. Cimino, S. Egidi, B. Lepri, A. Pentland, and G. Vaglini, "Stigmergy-based modeling to discover urban activity patterns from positioning data," in *Social, Cultural, and Behavioral Modeling*. Cham, Switzerland: Springer, Apr. 2017, pp. 292–301.

[9] G. Tesauro, "Temporal difference learning and TD-gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, Mar. 1995.

[10] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, New Orleans, LA, USA, vol. 3, May 2004, pp. 2619–2624.

[11] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[12] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," Feb. 2016, *arXiv:1602.01783*. [Online]. Available: http://arxiv.org/abs/1602.01783

[14] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn.*, Bejing, China, Jun. 2014, vol. 32, no. 1, pp. 387–395.

[15] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[16] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*. San Francisco, CA, USA, Jul. 1994, pp. 157–163.

[17] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems," *Knowl. Eng. Rev.*, vol. 27, no. 1, pp. 1–31, Feb. 2012.

[18] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," May 2016, *arXiv:1605.06676*. [Online]. Available: http://arxiv.org/abs/1605.06676

[19] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th Int. Conf. Mach. Learn. (ICML)*, May 1993, pp. 330–337.

[20] H. Mao, Z. Gong, Y. Ni, and Z. Xiao, "ACCNet: Actor-coordinator-critic net for 'learning-to-communicate' with deep multi-agent reinforcement learning," Jun. 2017, *arXiv:1706.03235*. [Online]. Available: http://arxiv.org/abs/1706.03235

[21] M. J. Mataric, "Using communication to reduce locality in distributed multiagent learning," *J. Exp. Theor. Artif. Intell.*, vol. 10, no. 3, p. 13, 1998.

[22] F. Heylighen, "Stigmergy as a universal coordination mechanism I: Definition and components," *Cognit. Syst. Res.*, vol. 38, pp. 4–13, Jun. 2016.

[23] F. Heylighen, "Stigmergy as a universal coordination mechanism II: Varieties and evolution," *Cognit. Syst. Res.*, vol. 38, pp. 50–59, Jun. 2016.

[24] R. Schoonderwoerd, O. E. Holland, J. L. Bruten, and L. J. M. Rothkrantz, "Ant-based load balancing in telecommunications networks," *Adapt. Behav.*, vol. 5, no. 2, pp. 169–207, Jan. 1997.

[25] G. D. Caro and M. Dorigo, "AntNet: Distributed stigmergetic control for communications networks," *J. Artif. Intell. Res.*, vol. 9, no. 1, pp. 317–365, Dec. 1998.

[26] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Gener. Comput. Syst.*, vol. 16, no. 8, pp. 851–871, Jun. 2000.

[27] C. Maru, M. Enoki, A. Nakao, S. Yamamoto, S. Yamaguchi, and M. Oguchi, "QoE control of network using collective intelligence of SNS in large-scale disasters," in *Proc. IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, Nadi, Fiji, Dec. 2016, pp. 57–64.

[28] J. Takahashi, R. Kanamori, and T. Ito, "A preliminary study on anticipatory stigmergy for traffic management," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, Macau, China, vol. 3, Dec. 2012, pp. 399–405.

[29] R. Kanamori, J. Takahashi, and T. Ito, "Evaluation of anticipatory stigmergy strategies for traffic management," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Seoul, South Korea, Nov. 2012, pp. 33–39.

[30] X. Xu, Z. Zhao, R. Li, and H. Zhang, "Brain-inspired stigmergy learning," *IEEE Access*, vol. 7, pp. 410–424, 2019.

[31] O. R. Pagán, "The brain: A concept in flux," *Phil. Trans. Roy. Soc. B, Biol. Sci.*, vol. 374, no. 1774, Jun. 2019, Art. no. 20180383.

[32] O. E. Holland, "Multiagent systems: Lessons from social insects and collective robotics," in *Proc. Adaptation, Coevol. Learn. Multiagent Syst., Papers AAAI Spring Symp.*, 1996, pp. 57–62.

[33] A. Ricci, A. Omicini, M. Viroli, L. Gardelli, and E. Oliva, "Cognitive stigmergy: Towards a framework based on agents and artifacts," in *Proc. 3rd Int. Workshop Environ. Multi-Agent Syst.*, Berlin, Germany, May 2006, pp. 124–140.

[34] C. H. Yong and R. Miikkulainen, "Coevolution of role-based cooperation in multiagent systems," *IEEE Trans. Auton. Mental Develop.*, vol. 1, no. 3, pp. 170–186, Oct. 2009.

[35] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[36] V. Crespi, A. Galstyan, and K. Lerman, "Comparative analysis of top-down and bottom-up methodologies for multi-agent system design," in *Proc. 4th Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS)*, New York, NY, USA, Jul. 2005, pp. 1159–1160.

[37] R. Aras, A. Dutech, and F. Charpillet, "Stigmergy in multi agent reinforcement learning," in *Proc. 4th Int. Conf. Hybrid Intell. Syst. (HIS)*, New York, NY, USA, Dec. 2004, pp. 468–469.

[38] K. Verbeeck and N. Ann, "Stigmergy and bootstrapping: Using ant algorithms as a case study for learning in MAS," in *Proc. 2nd Int. Workshop Ant Algorithms*, Sep. 2000.

[39] 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, vol. 70, Aug. 2017, pp. 2681–2690.

[40] L. Peshkin, N. Meuleau, and L. P. Kaelbling, "Learning policies with external memory," in *Proc. 16th Int. Conf. Mach. Learn.*, Williamstown, MA, USA, Mar. 2001.

[41] M. Bourenane, A. Mellouk, and D. Benhamamouch, "Reinforcement learning in multi-agent environment and ant colony for packet scheduling in routers," in *Proc. 5th ACM Int. Workshop Mobility Manage. Wireless Access (MobiWac)*, New York, NY, USA, Oct. 2007, pp. 137–143.

[42] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[43] K. Verbeeck and A. Nowe, "Colonies of learning automata," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 32, no. 6, pp. 772–780, Dec. 2002.

[44] B. Masoumi and M. R. Meybodi, "Speeding up learning automata based multi agent systems using the concepts of stigmergy and entropy," *Expert Syst. Appl.*, vol. 38, no. 7, pp. 8105–8118, Jul. 2011.
[45] O. Shamir and T. Zhang, "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes," in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, Jun. 2013, vol. 28, no. 1, pp. 71–79.
[46] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA, vol. 1, Dec. 2012.
[47] R. Hu, Y. Guo, and Y. Gong, "Concentrated differentially private and utility preserving federated learning," Mar. 2020, *arXiv:2003.13761*. [Online]. Available: http://arxiv.org/abs/2003.13761
[48] G. Sartoretti, Y. Wu, W. Paivine, and T. K. S. Kumar, "Distributed reinforcement learning for multi-robot decentralized collective construction," in *Distributed Autonomous Robotic Systems*. Cham, Switzerland: Springer, 2019, pp. 35–49.
[49] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," Dec. 2015, *arXiv:1511.05952*. [Online]. Available: http://arxiv.org/abs/1511.05952
[50] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.
[51] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, no. 1, pp. 253–279, Jun. 2013.
[52] M. Mamei and F. Zambonelli, "Programming stigmergic coordination with the TOTA middleware," in *Proc. 4th Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS)*, New York, NY, USA, Jul. 2005, pp. 415–422.
[53] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar, "Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD," 2018, *arXiv:1803.01113*. [Online]. Available: http://arxiv.org/abs/1803.01113
[54] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[55] O. Miramontes, R. V. Solé, and B. C. Goodwin, "Collective behaviour of random-activated mobile cellular automata," *Phys. D, Nonlinear Phenomena*, vol. 63, nos. 1–2, pp. 145–160, Feb. 1993.
[56] K. Chen, R. Li, J. Crowcroft, Z. Zhao, and H. Zhang, "The implementation of stigmergy in network-assisted multi-agent system," in *Proc. ACM Mobicom*, London, U.K., Sep. 2020, pp. 1–2.

Xing Xu is currently pursuing the Ph.D. degree with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China.

His research interests include collective intelligence, reinforcement learning, and deep learning.

Rongpeng Li He was a Research Engineer with the Wireless Communication Laboratory, Huawei Technologies Company, Ltd., Shanghai, China, from August 2015 to September 2016. He was a Visiting Scholar with the Department of Computer Science and Technology, Cambridge University, Cambridge, U.K., from February 2020 to August 2020. He has been with Zhejiang University (ZJU), Hangzhou, China, since November 2016 and is now an Associate Professor with the College of Information Science and Electronic Engineering, ZJU. His research interest currently focus on networked intelligence for communications evolving (NICE). He received the sponsorship "National Post-Doctoral Program for Innovative Talents" in 2016.

Mr. Li serves as an Editor for *China Communications*.

Zhifeng Zhao received the bachelor's degree in computer science, the master's degree in communication and information system, and the Ph.D. degree in communication and information system from the PLA University of Science and Technology, Nanjing, China, in 1996 1999, and 2002, respectively.

From 2002 to 2004, he acted as a Post-Doctoral Researcher with Zhejiang University, Hangzhou, China, where his researches were focused on multimedia next-generation networks (NGN) and soft-switch technology for energy efficiency. From 2005 to 2006, he acted as a Senior Researcher with the PLA University of Science and Technology, where he performed research and development on advanced energy-efficient wireless router, Ad Hoc network simulator, and cognitive mesh networking test-bed. From 2006 to 2019, he was an Associate Professor with the College of Information Science and Electronic Engineering, Zhejiang University. Currently, he is with Zhejiang Lab, Hangzhou. His research area includes software defined network (SDN), wireless network in 6G, computing network, and collective intelligence.

Dr. Zhao is the Symposium Co-Chair of ChinaCom 2009 and 2010. He is the Technical Program Committee (TPC) Co-Chair of the 10th IEEE International Symposium on Communication and Information Technology (ISCIT 2010).

Honggang Zhang was an Honorary Visiting Professor with the University of York, York, U.K., and an International Chair Professor of Excellence with the Université Européenne de Bretagne and Supélec, France. He is currently a Full Professor with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China. He has coauthored and edited two books: *Cognitive Communications-Distributed Artificial Intelligence (DAI), Regulatory Policy and Economics, Implementation* (John Wiley & Sons) and *Green Communications: Theoretical Fundamentals, Algorithms and Applications* (CRC Press), respectively. He is active in the research on cognitive green communications.

Mr. Zhang was the leading Guest Editor of the *IEEE Communications Magazine* special issues on Green Communications. He served as the Series Editor for the *IEEE Communications Magazine* for its Green Communications and Computing Networks Series from 2015 to 2018 and the Chair of the Technical Committee on Cognitive Networks of the IEEE Communications Society from 2011 to 2012. He is an Associate Editor-in-Chief of *China Communications*.