This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3127873, IEEE Internet of Things Journal

Deep Reinforcement Learning for Flocking Motion of Multi-UAV systems: Learn from a Digital Twin

Gaoqing Shen, Lei Lei, Zhilin Li, Shengsuo Cai, Lijuan Zhang, Pan Cao, Xiaojiao Liu

***Abstract*—Over the past decades, unmanned aerial vehicles (UAVs) have been widely used in both military and civilian fields. In these applications, flocking motion is a fundamental but crucial operation of multi-UAV systems. Traditional flocking motion methods usually designed for a specific environment. However, the real environment is mostly unknown and stochastic, which greatly reduces the practicality of these methods. In this paper, deep reinforcement learning (DRL) is used to realize the flocking motion of multi-UAV systems. Considering that the sim-to-real problem restricts the application of DRL to the flocking motion scenario, a digital twin (DT) enabled DRL training framework is proposed to solve this problem. The DRL model can learn from DT and be quickly deployed on the real-world UAV with the help of DT. Under this training framework, this paper proposes an actor-critic DRL algorithm named behavior-coupling deep deterministic policy gradient (BCDDPG) for the flocking motion problem, which is inspired by the flocking behavior of animals. Extensive simulations are conducted to evaluate the performance of BCDDPG. Simulation results show that BCDDPG achieves a higher average reward and performs better in terms of arrival rate and collision rate compared with existing methods.**

***Index Terms*—multi-UAV systems, flocking motion, deep reinforcement learning, digital twin.**

I. Introduction

OVER the past two decades, unmanned aerial vehicles (UAVs) have experienced unprecedented growth in both civilian and military applications, such as aerial photography, search and rescue, target tracking, forest fire-prevention, crowd monitoring, and agriculture spraying [1]–[5]. Most of these tasks could be performed more efficiently if there are multi-UAVs working cooperatively. For a multi-UAV system, a fundamental and crucial challenge is how to maintain the coordination of UAVs to achieve a team goal and behavior in an autonomous manner without external guidance. The flocking motion is the important premise to realize the coordination of multi-UAVs, as shown in Fig. 1, for which a group of UAVs moves from start to destination in a collective way.

Flocking was originally used to describe the collective behavior of a group of natures with a common goal. For example, birds, ants, bees, fish, and wolves often exist in flocking form to maximize the team's benefits. Inspired by the natural flocks, Reynolds proposed the well-known Boids model to simulate the flocking behaviors of natural animals in 1986 [6]. The Boids model contains three rules: (i) Collision Avoidance: avoid collisions with the neighborhood; (ii) Matching: try to match velocity with the neighborhood; (iii) Flocking Centering: try to match velocity with the neighborhood; (iii)Flocking Centering: try to stay close to the neighborhood. In the decades after the Boids model was proposed, it has been wildly used to guide the design of the flocking motion algorithm of the multi-agent system.

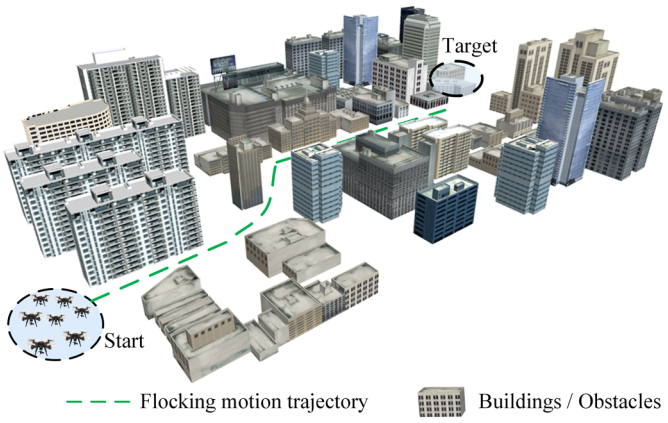


Fig. 1. The considered distributed multi-UAV flocking motion scenario.

Vicsek *et al.* [7] first analyzed the velocity consistency in the Boids model and proposed a dynamics model to investigate the self-ordered motion in particle systems. Olfati-Saber [8] put forward a leader-follower theoretical framework for the design of distributed flocking algorithm for multi-agent systems. It was assumed in this framework that every agent is informed of the information of the virtual leader. Su *et al.* [9] provided another framework based on Olfati-Saber's work with only a fraction of agents being informed. Inspired by homing pigeon hierarchical strategies, Luo *et al.* [10] proposed a distributed control algorithm to solve the flocking motion problem of multi-UAV systems, which adopts the hierarchical leadership network. Zhao *et al.* [11] proposed a self-organism collective motion control algorithm that enables multi-UAVs to move along a preplanned path based on the artificial potential field (APF) method.

Reviewing the above literature about flocking motion, it is obvious that almost all algorithms are based on control theory. However, these algorithms usually are designed for a specific scenario, and their performance will degrade rapidly when the

This work is partly supported by the National Natural Science Foundation of China (61572254, 61902182); the Qing Lan Project of Jiangsu Province of China. *(corresponding author : Lei Lei)*

The authors are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China (e-mail: shengaoqing@nuaa.edu.cn; leilei@nuaa.edu.cn; lizhilin@nuaa.edu.cn; caishengsuo@nuaa.edu.cn; lijuan.zhang@nuaa.edu.cn; caopan@nuaa.edu.cn; liuxjiao@nuaa.edu.cn).

scenario changes. It is usually assumed that the environment information is available for all agents, which is not practical in the real world.

In recent years, machine learning, especially deep learning (DL) [12], has been increasingly used to solve some complex problems [13] with the growth of computing power, providing another alternative solution for the flocking motion problem of multi-UAV systems. As a kind of machine learning, reinforcement learning (RL) is suitable for handling sequential decision-making problems. Recently, the emergence of deep reinforcement learning (DRL) [14], [15] which combines DL and RL, has greatly improved the ability of RL to solve large-scale complex problems and achieved great success in electronic games [16]–[18].

The flocking motion problem of multi-UAV is also a sequential decision-making problem just like electronic games. However, there are some differences between them. Firstly, it is hard for us to model a high-fidelity simulation environment for the flocking motion scenario due to the nonlinear and uncertain factors. The policy learned from DRL methods in the simulation environment can not directly deploy on the real-world UAV. Secondly, if we use the real-world UAV to train the DRL model, the training speed may be unacceptable for the reason of inefficient data acquisition. The sim-to-real problem restricts the application of DRL to the flocking motion problem of multi-UAV systems [19].

To solve the above issue, we turn our attention to digital twin (DT) technology. DT is a high-fidelity mirror of the real world in cyberspace that reflects the states of the real world based on the historical data, sensor data, and physical objects in a timely manner [20]. Recently, DT has been wildly applied in smart city, smart manufacturing, and health management [21]. More and more people use machine learning in the data processing of DT to provide better service for the physical objects. According to [22], with the help of a DT, machine learning methods can easily get high-fidelity state information of the real world for model training. In other words, machine learning methods learn from the DT. As a result, we can establish a DT of the multi-UAV system to realize flocking motion with the control of DRL methods, which use the high-fidelity data from DT for training. In addition, by monitoring the variations of the real-world multi-UAV system, the DRL model can be continuously updated over time. Therefore, DT may solve the sim-to-real problem, and promote the application of DRL to the non-stationary real-world control problem. Nevertheless, how to apply DT in the DRL architecture for the flocking motion of multi-UAV systems has not been studied.

In this paper, we focus on how to combine DRL and DT to realize the flocking motion of multi-UAV in unknown and stochastic environments. We aim to improve the cooperative ability of the multi-UAV system and explore a more efficient training paradigm for DRL methods. Therefore, the digital twin of the multi-UAV system is established for the training of flocking motion methods based on DRL. To the best of our knowledge, this is the first work that applies DT to the flocking motion problem. The main contributions of this paper can be summarized as follows:

- We propose a digital twin enabled DRL training framework for the flocking motion of multi-UAV systems. With the help of DT, the DRL model can obtain high-fidelity state information of the flocking motion scenario for model training. The trained model can be quickly deployed on the real-world UAV to evaluate the performance through the connections between the physical entity and digital model. Moreover, the trained model can be continuously updated by monitoring the state changes of multi-UAV systems in the real world.
- We propose an improved version of deep deterministic policy gradient (DDPG) [23] called behavior-coupling DDPG (BCDDPG). Inspired by the flocking behavior of animals, BCDDPG uses a decomposing first and then coupling actor network architecture, which helps the UAV to generate higher-quality actions for the flocking motion problem.
- The long short-term memory (LSTM) network is adopted in the actor network of BCDDPG to solve the partial observability of the flocking motion problem. The use of the LSTM network improves the comprehension ability of the actor network to the environment state, resulting in better policy.

The rest of this paper is organized as follows: Section II overviews the related work of DRL on UAV navigation. Section III defines the system model and formulates the flocking motion problem of multi-UAV systems. The proposed DRL algorithm is demonstrated in section IV. In section V, simulation results are presented to validate the effectiveness of BCDDPG. Section VI concludes this paper and outlooks the future work.

II. Related Work

In this section, we briefly review the related work of DRL on UAV navigation and point out the inadequacy of these works in the flocking motion of multi-UAV systems.

UAV navigation has been a hot topic in UAV control in the past decades. With the development of DRL, more and more researchers use DRL to solve the UAV navigation problem. Wang *et al.* [24] considered the navigation in complex environments that UAV might get in trouble with a local dilemma. They combined the recurrent neural network with DRL and use historical observations as the state input to solve the partial observability. In [25], Wang *et al.* further proposed a DRL algorithm to solve the autonomous UAV navigation problem with sparse rewards, which uses a nonexpert prior policy to guide the agent to get close with the goal and explore the state space at the same time. Singla *et al.* [26] focused on UAV navigation in unstructured and unknown indoor environments. They used a deep neural network (DNN) to model the depth maps of the indoor environment and proposed a deep recurrent Q-network with temporal attention to learn the control policy. Autonomous navigation problems often have high-dimensional state space and multiple targets to be optimized. In [27], the authors proposed a hybrid hierarchical reinforcement learning method consisting of three sub-layers to alleviate the "curse of dimensionality" for online UAV navigation with partial

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3127873, IEEE Internet of Things Journal

observability. Furthermore, multi-UAV autonomous navigation or path planning receives more attention recently. Qie *et al.* [28] adopted the multi-agent deep deterministic policy gradient (MADDPG) algorithm to solve multi-UAV target assignment and path planning simultaneously. Liu *et al.* [29] proposed a DRL approach to achieve long-term communication coverage for ground users by a multi-UAV system, which takes energy efficiency, fairness, and connectivity into consideration. The authors in [30] also used the MADDPG algorithm to manage the trajectory design of UAVs for multi-UAV assisted mobile edge computing environments.

However, the above works are either single-UAV navigation or multi-UAV navigation, but they do not consider flocking behavior. There is little research dedicated to the flocking motion of multi-UAV using RL. In [31], a Q-learning based approach for flocking motion of small fixed-wing UAVs in stochastic environments was proposed. The authors used a leader-follower mechanism to realize the coordination among the leader and followers, which also determines that it is not a distributed algorithm. Xu *et al.* [32] proposed a DRL approach for a multi-vehicle flocking control system to guide the way-points tracking, collision avoidance, and communication preserving. Yan *et al.* [33] adopted the same leader-follower mechanism as [31] to deal with the flocking control of fixed-wing UAVs. The difference is that they used a DRL method instead of RL and validated the feasibility of their method in the semi-physical simulation. Although these works have studied the flocking motion of multi-UAV systems based on DRL. Most of them only consider the discrete state and action space which limits the precision of the optimal policy. Another aspect that needs to be improved is how to solve the sim-to-real problem in the application of DRL to the flocking motion problem. Therefore, we propose a digital twin enabled DRL algorithm for distributed flocking motion of multi-UAV systems which considers continuous state and action space.

III. System Model and Problem Statement

In this section, we define the system model of the flocking motion problem and formulate the flocking motion problem into a mathematical problem.

A. System Model

We consider an area of interest (AoI) where there are N UAVs, M obstacles, and one target as shown in Fig. 2. Each UAV is equipped with a GPS device and is aware of its own position. UAVs can communicate with a wireless transceiver to exchange information such as their position and velocity. The position and velocity of UAVs are denoted as $\mathbf{U}_t = \{\mathbf{u}_{1,t}, \mathbf{u}_{2,t}, \mathbf{u}_{3,t} \dots \mathbf{u}_{N,t}\}$ and $\mathbf{V}_t = \{\mathbf{v}_{1,t}, \mathbf{v}_{2,t}, \mathbf{v}_{3,t} \dots \mathbf{v}_{N,t}\}$ respectively. Similarly, the positions of obstacles and target are denoted as $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3 \dots \mathbf{o}_M\}$ and $\mathbf{G}$. The UAVs are initially located at the starting point according to a preset distribution and intend to move to the target along a real-time planning trajectory. Obstacles are randomly distributed in the AoI. Once the distance between the UAV and obstacle is less than the perception distance, the UAV could determine the position of the obstacle by sensors. In order to satisfy the requirements of RL, a discrete time-scale is adopted. Then the trajectory of UAVs can be denoted as $\mathbf{T}_r = \{\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3, \dots\}$. The magnitude of the velocity of each UAV is adjustable with a range of $[v_{\min}, v_{\max}]$, and the UAV moves with a constant velocity when $v_{\min} = v_{\max}$. We assume that the mobility of UAV follows the same model proposed in [34], which can be described as

$$\begin{cases} \mathbf{v}_{i,t+1} = \dfrac{clip\left(\left\|\mathbf{v}_{i,t} + \frac{\mathbf{F}_{i,t}}{m}\Delta t\right\|_2, v_{\min}, v_{\max}\right)}{\left\|\mathbf{v}_{i,t} + \frac{\mathbf{F}_{i,t}}{m}\Delta t\right\|_2}\left(\mathbf{v}_{i,t} + \dfrac{\mathbf{F}_{i,t}}{m}\Delta t\right), \\ \mathbf{u}_{i,t+1} = \mathbf{u}_{i,t} + \mathbf{v}_{i,t+1}\Delta t \end{cases} \tag{1}$$

where $\mathbf{F}_{i,t}$ represents the external control input, m is the default mass of UAV, Δt is the time step, $clip()$ represents the truncation function.

To support the efficient training of DRL methods and provide intelligent decisions in a timely manner for flocking motion problems, we propose a digital twin-enabled DRL framework. As illustrated in Fig. 2, the framework consists of four parts: the physical entity that constitutes the basis of the framework; the digital model which is a mirror of the physical entity in cyberspace; the DRL model that provides intelligent decision-making services; and connections bridges physical entity and digital model. Next, we will introduce each part in detail.

Physical Entity: The multi-UAV system which consists of low-cost, small-sized UAVs and the task environment is called the physical entity. The UAVs are computation and storage constrained and are considered hardly for DRL training. Each UAV is equipped with multiple sensors and senses the environment state over time.

Digital model: With the received data from the real world, the central server establishes an ultra-fidelity digital model of the multi-UAV system through simulation and modeling. To keep the fidelity of the digital model, the central server updates the digital model at each time step with the new data from the real-time sensing of the real-world UAVs.

DRL model: The DRL model is used for providing decision-making services for flocking motion problems. It extracts the state information from the digital model which is needed by the training process. Making use of the powerful computing performance of the central server, the DRL model can output the control policy for the flocking motion of multi-UAV systems in a timely manner.

Connections: The connections bridge the physical space and cyberspace. It can be established through 4G/5G, mobile AP or satellite, and so on. The connections are bidirectional to provide a real-time service for flocking motion problems. On one hand, the physical entity transmits the sensor data to the central server for digital model construction. On the other hand, the central server output the control policy generated by the DRL model to the physical entity for UAV cooperation.

The above training-execution framework of DRL has the following advantages: (1) The central server can establish multiple workers that run the copies of the digital model to general more sample data at the same time. (2) The digital model can obtain the global state information, which is useful

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3127873, IEEE Internet of Things Journal

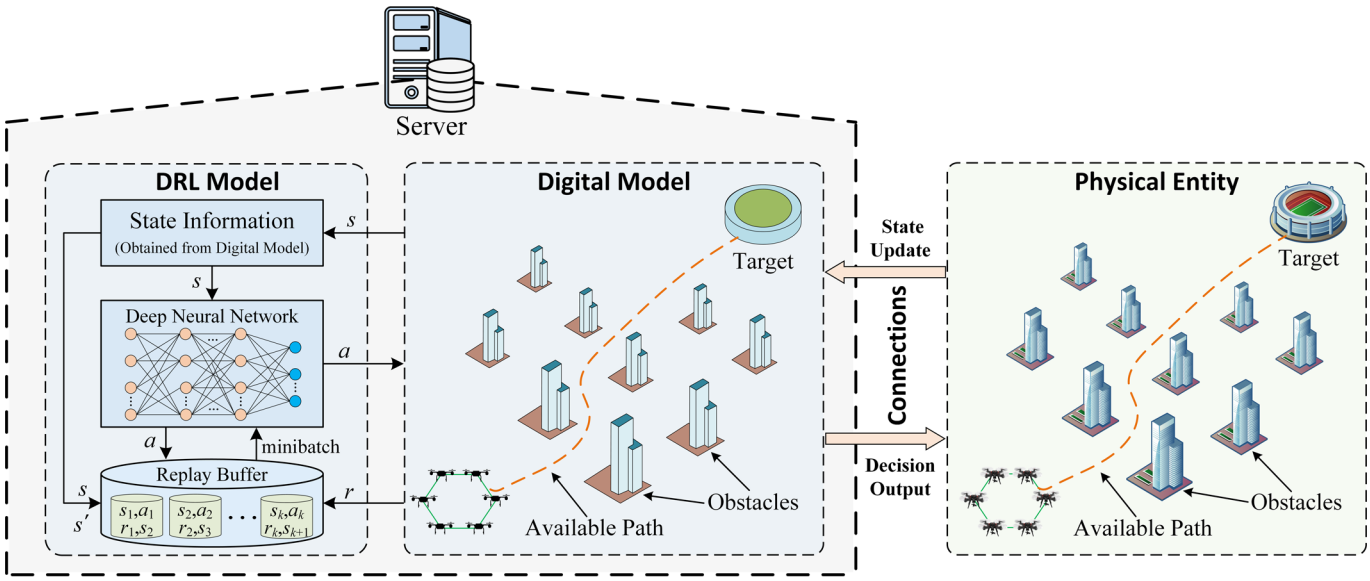


Fig. 2. System model

to improve the training speed and effectiveness of the DRL model. (3) Once the DRL model has completed the training stage, it can be deployed to the real-world multi-UAV system and executed in a distributed way. In other words, this is a centralized training and decentralized execution architecture. (4) In addition, the DRL model can be continuously updated and improved with the help of DT in the execution stage, which realizes the continuous evolution of the DRL model.

B. Problem Statement

The main goal of the flocking motion of multi-UAV systems can be summarized into three sub-goals:

Firstly, the travel distance of the flocking motion of multi-UAV systems should be as smaller as possible. Suppose it takes T steps to reach the target, the above goal is equal to minimize

$$\sum_{t=1}^{T-1}\sum_{i=1}^{N}\|\mathbf{U}_{i,t+1}-\mathbf{U}_{i,t}\|_2. \tag{2}$$

Secondly, the UAV is required to avoid collisions with obstacles and its neighbors in an automatic way. Suppose that the safe distance between UAV i and UAV j is d_1, and the safe distance between UAV i and obstacle o is d_3, the position of UAV i must satisfy

$$\begin{aligned}\|\mathbf{u}_{i,t}-\mathbf{u}_{j,t}\|_2 \geq d_1, \forall j \in \mathcal{N}_i \\ \|\mathbf{u}_{i,t}-\mathbf{o}_m\|_2 \geq d_3, \forall m \in \mathcal{O}\end{aligned}, \tag{3}$$

where $\mathcal{N}_i$ represents the neighbor set of the UAV i, $\mathcal{O}$ represents the set of all obstacles.

Finally, the distance between two UAVs must be less than the maximum communication range to maintain connectivity. Suppose that the maximum communication range of UAV is d_2, the position of UAV i should satisfy

$$\|\mathbf{u}_{i,t}-\mathbf{u}_{j,t}\|_2 \leq d_2, \forall j \in \mathcal{N}_i. \tag{4}$$

In addition, the motion of UAV should also satisfy its power and the boundary condition constraints. In this paper, we aim to solve the problem in a machine-learning way without professional domain knowledge.

IV. The Proposed BCDDPG Algorithm

In this section, we first introduce why the flocking motion problem can be solved by DRL. Then we define the state and action space of DRL. Next, we design the reward function for the flocking motion problem according to task requirements. Finally, we present the BCDDPG algorithm and give the training procedures in detail.

A. Reinforcement Learning for Flocking Motion Problem

In classical RL, the problem to be solved is described as a Markov decision process (MDP) [35]. An MDP must satisfy the Markov property, i.e., the next state of the process only depends on the current state of the process and the selected action of the decision-maker. If we consider the UAV as the decision-maker, the flocking motion process also satisfies the Markov property. The next state of the flocking motion process only depends on the current state of the flocking motion process and the selected action of the UAV. Therefore, we can use RL to solve the flocking motion problem.

An MDP is often defined with a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}^a_{s,s'}, \mathcal{R}^a_{s,s'})$, where:

- $\mathcal{S}$ is a set of states called state space;
- $\mathcal{A}$ is a set of actions called action space;
- $\mathcal{P}^a_{s,s'}$ represents the probability of transition from state space s to state space s' after executing action a.
- $\mathcal{R}^a_{s,s'}$ represents the immediate reward received after transitioning from state s to state s' with action a.

The difference between RL and supervised learning is that RL learns a policy from the interactions between the agent and the environment, which does not require a training set of labeled examples in advance. In other words, RL learns the mapping relationship between state space $\mathcal{S}$ and action space $\mathcal{A}$. Considering a discrete-time sequence t = 0, 1, 2, ..., the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3127873, IEEE Internet of Things Journal

agent observes a state $s_t \in \mathcal{S}$ of the environment and selects an action $a_t \in \mathcal{A}$ based on s_t at each time step t. Then the agent transitions to the next state s_{t+1} with probability $p^{a_t}_{s_t,s_{t+1}} \in \mathcal{P}$ and receives the immediate reward $r_t \in \mathcal{R}$. The mapping relationship between s_t and a_t is denoted as the policy π, where $\pi_t(s|a)$ is the probability that $a_t = a$ if $s_t = s$. The goal of RL is to learn a policy π to maximize the cumulative discounted reward:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{5}$$

where $\gamma \in$ [0,1) is the discount rate. In general, we denote the value (i.e., the expected return) of taking action a at state s under a policy π as the action-value function:

$$Q_\pi(s,a) = \mathbb{E}_\pi(G_t) = \mathbb{E}_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a\right). \tag{6}$$

According to Bellman's principle of optimality [35], the optimal policy is equivalent to maximizing the action-value function:

$$\pi_*(s) = \arg\max_a Q_\pi(s,a). \tag{7}$$

For most RL problems, the transition probability $\mathcal{P}$ of the environment is unknown to the agent. The agent needs to approximate the action-value function to learn the optimal policy. The most wildly used approximation method is the temporal difference (TD) learning, in which the action-value function is updated with

$$Q_\pi(s,a) = Q_\pi(s,a) + \alpha\left(r + \gamma \max_a Q_\pi(s',a') - Q_\pi(s,a)\right), \tag{8}$$

where α is the learning rate, $r + \gamma \max Q_\pi(s',a')$ is known as the TD-target, and $r + \gamma \max Q_\pi(s',a') - Q_\pi(s,a)$ is known as the TD-error. The action-value function is updated every time step according to (8) until it converges. Then the optimal policy is determined by (7) with the convergent action-value function.

B. State and Action Space Representation

For an MDP problem, the state information usually refers to all the possibilities of the agent and environment. However, in the flocking motion scenario, each UAV cannot sense the whole information of the environment because of the limitations of the sensors and communications. In fact, it is a partial observation Markov decision process (POMDP). For each UAV i at time step t, its observation $s_{i,t}$ consists of three parts. The first part is the target state information, which indicates the destination for the flocking motion task. In order to weaken the sensitivity of the model to the absolute position of the target, we actually adopt the relative position $\{\mathbf{G} - \mathbf{u}_{i,t}\}$ instead of absolute position $\mathbf{G}$, and this trick is also adopted in the second and third parts. The second part is the obstacle state information. Assuming that there are k obstacles within the perceived range of UAV i, the obstacle information of UAV i at time step t is $\{\mathbf{o}_1 - \mathbf{u}_{i,t}, \mathbf{o}_2 - \mathbf{u}_{i,t}, \ldots, \mathbf{o}_k - \mathbf{u}_{i,t}\}$. The third part is the neighbor information, including the position and velocity. The neighbor information of UAV i at time step t is $\{\mathbf{u}_{1,t} - \mathbf{u}_{i,t}, \mathbf{u}_{2,t} - \mathbf{u}_{i,t}, \ldots, \mathbf{u}_{j,t} - \mathbf{u}_{i,t}\}$ where $j \in \mathcal{N}_i$. To sum up, the state space of UAV i at time step t can be represented as $s_{i,t} = \{\mathbf{G} - \mathbf{u}_{i,t}, \mathbf{o}_1 - \mathbf{u}_{i,t}, \mathbf{o}_2 - \mathbf{u}_{i,t}, \ldots, \mathbf{o}_k - \mathbf{u}_{i,t}, \mathbf{u}_{1,t} - \mathbf{u}_{i,t}, \mathbf{u}_{2,t} - \mathbf{u}_{i,t}, \ldots, \mathbf{u}_{j,t} - \mathbf{u}_{i,t}\}$.

For the purpose of making the trajectory of UAV smoother, the continuous action space is adopted in our model. Considering the AoI is restricted in a two-dimensional plane, the action space of UAV i at time step t is characterized by $a_{i,t} = \{v, \theta\}$, where v represents the magnitude of the velocity, ranging from $v_{\min}$ to $v_{\max}$, and θ represents the direction of the velocity, which is limited by the maximum steering angle of the UAV.

The main goal of BCDDPG is to make UAVs learn how to fly in a collective way. In other words, we hope that our model has generalization ability. Therefore, we represent the state space by using relative position relation to reduce the influence of the environment, and we will use a completely random environment setting for every training episode. In our experiments, we will test the generalization ability of our model in detail.

C. Reward Function Design

Reward signal will reinforce the action of agents. A good reward function could shorten the convergence time of the algorithm. For the flocking motion of multi-UAV systems, its main purpose consists of three aspects: achieving the target as soon as possible, not colliding with obstacles and neighbors, and maintaining a proper distance with neighbors. Based on these purposes, we define the reward functions as follows:

- **Approaching the target**:This reward function is to guide the UAV towards the target direction. The basic idea is that the travel distance towards the target direction in one step as large as possible. Consequently, the positive reward of UAV i approaching the target at time step t is defined as

$$r_1 = \omega_{\text{appro}} \|\mathbf{u}_{i,t} - \mathbf{G}\|_2, \tag{9}$$

  where ω_{appo} is a positive constant.
- **Collision avoidance**: This reward function is to guide the UAV to keep a safe distance from the obstacles and neighbors. The negative reward of UAV i colliding with obstacles or neighbors at time step t is defined as

$$r_2 = -r_{\text{obs}} - r_{\text{nei}}, \tag{10}$$

  where

$$r_{\text{obs}} = \begin{cases} \omega_{\text{obs}} & \text{if} \|\mathbf{u}_{i,t} - \mathbf{o}_m\|_2 < d_3, \forall m \in \mathcal{O} \\ 0 & \text{otherwise} \end{cases}, \tag{11}$$

$$r_{\text{nei}} = \begin{cases} \omega_{\text{nei}} & \text{if} \|\mathbf{u}_{i,t} - \mathbf{u}_{j,t}\|_2 < d_1, \forall j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases}, \tag{12}$$

  where ω_{obs} and ω_{nei} are positive constants.
- **Connectivity maintenance**: This reward function is to guide the UAV to maintain connectivity with neighbors, so as to build better cooperation during the flocking

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3127873, IEEE Internet of Things Journal

motion process. The positive reward of UAV i for cooperation with its neighbors at time step t is defined as

$$r_3 = \begin{cases} \omega_{\text{connec}} & \text{if } d_1 \leq \|\mathbf{u}_{i,t} - \mathbf{u}_{j,t}\|_2 \leq d_2, \forall j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases}, \tag{13}$$

where ω_{connec} is a positive constant.

In addition to the above reward functions which are closely related to the flocking motion behavior, we also define the other two reward functions to make the UAV performs better. One is the step reward for guiding the UAV to use less time to complete the flocking motion task, defined as

$$r_4 = -\omega_{\text{step}}, \tag{14}$$

where ω_{step} is a positive constant.The other is the boundary reward to give punishment when the UAV is too close to the boundaries, defined as

$$r_5 = \omega_{\text{hor}} \left[d_{\text{hor}} - 0.05\left(x_e - x_s\right)\right] + w_{\text{ver}} \left[d_{\text{ver}} - 0.05\left(y_e - y_s\right)\right], \tag{15}$$

where ω_{hor} and ω_{ver} are positive constants, x_s, x_e, y_s, and y_e are the boundaries of the AoI, d_{hor} and d_{ver} are the minimum distance of the position of UAV i to the horizontal and vertical boundaries, and are set to zero when they are bigger than 0.05 times the side length.

Therefore, the whole reward function of UAV i at time step t can be derived as

$$r_{i,t} = r_1 + r_2 + r_3 + r_4 + r_5. \tag{16}$$

D. Behavior-coupling BCDDPG

For traditional model-free RL methods, the action-value function is usually stored in a Q-table (e.g., Q-learning and SARSA) [35].However, the use of Q-table determines these methods only can solve problems with discrete and low dimensional state and action space. Recently, people combine DL and RL, resulting in the Deep Q-network (DQN) algorithm, which uses a Q-network (with parameters θ) to approximate the action-value function of continuous state space [14]. Although the use of Q-network enables DQN to solve problems with high dimensional state space, it can only handle discrete and low dimensional action space. In this paper, we propose a novel DRL method called behavior-coupling DDPG (BCDDPG) to solve the flocking motion problem with continuous action space.

The digital twin enabled BCDDPG algorithm is presented in Fig. 3. Each agent has its own decision model. The decision model uses the state information extracted from the digital model as the input, and outputs the action to be performed in the next time step to the digital model. The decision model of BCDDPG uses the actor-critic network architecture [35]. Instead of using a Q network to output the action-value of all possible actions like DQN, BCDDPG uses an actor network (with parameters θ_μ) to directly output the expected action. The actor network builds a mapping between the current state and specific action, which enables DDPG to deal with continuous action space. That is to say, the actor network is just the policy network. Since the output action is deterministic at

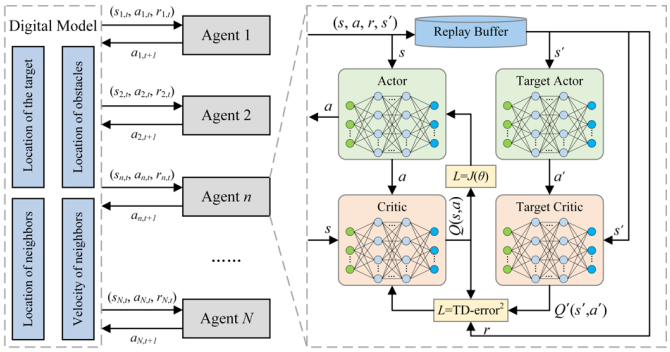


Fig. 3. Digital twin enabled distributed decision framework of BCDDPG.

every time step, the stochastic policy π becomes deterministic policy μ, which also explains the meaning of "deterministic" in BCDDPG. To train the actor network, BCDDPG adopts another network called critic network (with parameters θ_Q) to approximate the action-value function $Q(s,a)$. The critic network is used to evaluate the quality of the action generated by the actor network and reinforces the actor network to output better actions according to Bellman's principle of optimality.

The critic network is trained using gradient descent. Define the loss function:

$$L\left(s,a|\theta^Q\right) = \left(r + \gamma \max_{a'} Q\left(s',a'\right) - Q\left(s,a|\theta^Q\right)\right)^2. \tag{17}$$

The parameters of the critic network are updated by minimizing the loss function:

$$\theta^Q = \theta^Q + \alpha \nabla_\theta L\left(s,a|\theta^Q\right). \tag{18}$$

The parameters of the actor network are updated by applying the chain rule to the expected return from the start distribution J:

$$\nabla_{\theta^\mu} J = \mathbb{E}\left[\nabla_a Q\left(s,a|\theta_Q\right)|_{s=s_t,a=\mu(s_t)} \nabla_{\theta_\mu} \mu\left(s|\theta_\mu\right)|_{s_t}\right]. \tag{19}$$

However, if we only use one neural network to approximate $Q(s,a)$, it may make the policy unstable. Since the update of the network $Q(s,a|\theta_Q)$ may lead to the overestimation of the TD-target, and eventually, lead to oscillations or divergence of the model. In order to improve the stability of the policy, the target networks (i.e., target actor network and target critic network) are used to calculate the target values, which have the same network architecture as the learned networks (i.e., actor network and critic network). The parameters of these target networks are updated by slowly track the learned networks, that is

$$\begin{cases} \theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\,\theta^{\mu'} \\ \theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\,\theta^{Q'} \end{cases}, \tag{20}$$

where $\tau \ll 1$. Accordingly, the target values are constrained by the learned networks to change slowly, which greatly improves the stability of the model.

Another useful trick used in BCDDPG is the experience replay mechanism. As we all know, the sample set of neural networks need to be independent identically distributed. Otherwise, it may lead to low accuracy and even divergence of the model. BCDDPG stores the sample set in the replay buffer and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3127873, IEEE Internet of Things Journal

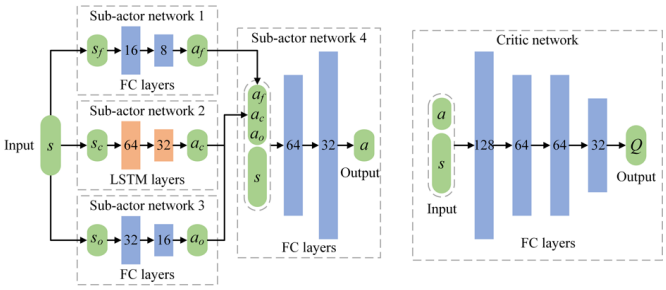


Fig. 4. The network architecture of actor and critic networks in BCDDPG.

the networks are trained by random sampling, which greatly reduces the correlation among data and improves the stability of the model.

The biggest innovation of BCDDPG is that the actor network consists of more than one sub-actor network. Inspired by the flocking behavior of animals, for the flocking motion problem, the state information of a given UAV can be divided into three categories:

1) The position of the target at the current time step. It determines the main forward direction of the UAV at the next time step, which is denoted as s_f.
2) The positions and velocities of its neighbors at the current time step. They determine whether the UAV should move closer to its neighbors to avoid falling behind or move away from its neighbors to avoid collisions, and drive the UAV to maintain a consistent velocity with neighbors as possible. In other words, it helps the UAV to maintain the connectivity with its neighbors, which is denoted as s_c.
3) The positions of obstacles at the current time step. They determine whether the UAV should move away from the obstacles to avoid collisions, which is denoted as s_o.

The final behavior of the UAV depends on the comprehensive impact of these three categories of state information, and the impact weight is time-varying during the flocking motion process. If we directly input all the state information into one actor network, it may hardly distinguish them correctly and output undesirable policy. In fact, the final behavior of the UAV can be regarded as a coupling of three kinds of behavior, i.e., forward behavior, connectivity maintenance behavior, and obstacle avoidance behavior. As shown in Fig. 4, BCDDPG uses three different sub-actor networks to process three categories of state information. Sub-actor network 1, 2 and 3 take s_f, s_c, and s_o as the input, and output a_f, a_c, and a_o as the corresponding action. Then sub-actor network 4 combines a_f, a_c, a_o, and s into a new vector as the input, and outputs the final action a. This way of decomposing first and then coupling can help the actor better understand the environment state of the UAV to generate higher-quality policy. Therefore, we call this method behavior-coupling DDPG.

Another innovation of BCDDPG is that BCDDPG uses the recurrent neural network (RNN) in the actor network. It is important to correctly predict the trajectory of the neighbor UAV in the next steps to maintain connectivity with them. However, the trajectory prediction of neighbor UAVs not only depends on their previous state information, but also is related to their historical state information. RNN is a kind of neural network that takes sequence information as input and is very effective for mining temporal information in historical data. From the point of view of complexity, RNN has a higher complexity than FC network due to the more complex network architecture. According to [36], the complexity of RNN is $O(n \cdot d^2)$, where n is the sequence length and d is the representation dimension. In order to minimize the complexity of our model, we set the sequence length to 4 (i.e., the state information of the previous four steps are used for prediction) of RNN and only use RNN in sub-actor network 2 to reduce n and d respectively. In order to avoid the vanishing gradient problem during learning recurrent neural networks using backpropagation, the LSTM network is adopted which introduces the gate mechanism to control the flow of features. As shown in Fig. 4, LSTM is used in sub-actor network 2, whereas other sub-actor networks only use the fully connected (FC) network. Different network architectures are adopted for different sub-state information, which further improves the comprehension ability of the actor network to the state information, thereby generating better policy.

E. Model Training

Pseudocode of BCDDPG is presented in algorithm 1. Since BCDDPG is a distributed algorithm, we only descript the training process of the DRL model of one UAV as an example. BCDDPG works as the following parts:

The first part is mainly for parameter initialization (Line 1 to 3). In the beginning, BCDDPG randomly initializes the actor network $\mu(s|\cdot)$ and critic network $Q(s,a|\cdot)$ with parameter θ_μ and θ_Q (Line 1). As we mentioned above, BCDDPG adopts the target network to alleviate the oscillation of the model. Therefore, the algorithm initializes the target actor $\mu'(s|\cdot)$ and target critic network $Q'(s,a|\cdot)$ with parameter $\theta_{\mu'}$ and $\theta_{Q'}$, where $\theta_{\mu'}$ equals θ_μ and $\theta_{Q'}$ equals θ_Q. (Line 2). Moreover, the replay buffer is also initialized in the first part (Line 3).

The second part is for the agent to explore the environment information (Line 4 to 13). It is worth mentioning that the training process contains E episodes and each episode contains T steps (Line 4 and 7). First, the agent receives the environment state s_t, and uses the behavior-coupling method to generate the action a (Line 8 to 10). Then the agent adds random noise to the action a to increase explorations and gets the actual action a_t (Line 10 to 11). Finally, the agent executes action a_t, and stores transition (s_t, a_t, r_t, s_{t+1}) into the replay buffer D (Line 12 to 13).

The third part is for the agent to exploit the environment information (Line 14 to 20). First, sample a minibatch of transitions from the reply buffer D (Line 14). Then, calculate the TD-error to update the critic network parameter θ_Q by minimizing the loss $L(\cdot)$ (Line 15 to 16), and update the actor network parameter θ_μ according to the chain rule (Line 17). Finally, the parameter $\theta_{\mu'}$ and $\theta_{Q'}$ of the target are updated slowly by using a soft-copy of θ_μ and θ_Q (Line 18).

V. Experiments

In this section, extensive simulations are conducted to evaluate the performance of BCDDPG. First, we present the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3127873, IEEE Internet of Things Journal

Algorithm 1 BCDDPG

Initialize the actor $\mu(s|\theta_\mu)$ and critic $Q(s,a|\theta_Q)$ with parameter θ_μ and θ_Q.
Initialize the target $\mu'(s|\theta_{\mu'})$ and $Q'(s,a|\theta_{Q'})$ with $\theta_{\mu'} \leftarrow \theta_\mu$ and $\theta_{Q'} \leftarrow \theta_Q$.
Initialize replay buffer D.
for episode = 1:E **do**
  Initialize a random process Ψ as the action noise.
  Receive the original observation state s_1.
  for step = 1:T **do**
    Decompose s_t into s_f, s_c and s_o.
    Input s_f, s_c, s_o to sub-actor network 1,2,3, and output a_f, a_c, a_o.
    Coupling (a_f, a_c, a_o, s_t) as the input of sub-actor network 4, output a.
    Obtain action $a_t = a + \Psi_t$ according to the exploration noise.
    Execute action a_t, receive reward r_t and observe new state s_{t+1}.
    Store transition (s_t, a_t, r_t, s_{t+1}) into D.
    Sample a random minibatch of λ transitions (s_i, a_i, r_i, s_{i+1}) from D.
    Calculate the TD-error as
    $r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta_{\mu'})|\theta_{Q'}) - Q(s_i, a_i|\theta_Q)$
    Update critic by minimizing the loss:
    $$L = \frac{1}{\lambda}\sum_i \text{TD-error}_i^2$$
    Update the actor according to the chain rule:
    $$\nabla_{\theta^\mu} J = \frac{1}{\lambda}\sum_i \nabla_a Q(s,a|\theta_Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta_\mu}\mu(s|\theta_\mu)|_{s_i}$$
    Update the target:
    $$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$
    $$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
  end for
end for

simulation environment and parameters. Next, we compare the convergence of BCDDPG and DDPG. Then, we define the evaluation metrics for the flocking motion of multi-UAV systems. Finally, we compare the performance of BCDDPG and other existing methods.

A. Simulation Environment

We implemented BCDDPG using PyTorch and Python 3.6 on an Ubuntu 20.04 server with 4 Nvidia 2080Ti graphic cards. We use a relative value to describe the scale relationship among AoI, UAV, target, and obstacles. Both the vertical and horizontal boundaries of AoI are set to [-1, 1], the size of the UAV is set to 0.01, the size of the target is set to 0.15, and the size of the obstacle is randomly distributed in the range [0.10, 0.15]. The task of the multi-UAV system is to move from the start to the target in a flocking motion way and avoid collisions with obstacles at the same time. The positions of obstacles are random for each episode. The maximum number of steps that a UAV can move in one episode is 150, and the whole training process contains 2×10^6 steps. The other simulation parameters are presented in Table I.

TABLE I
SIMULATION PARAMETERS

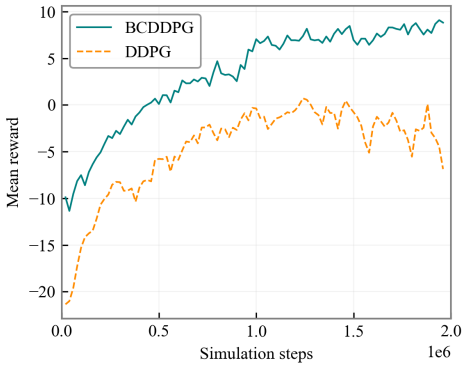
Parameters	Value
ω_{appo}	100
ω_{obs}	50
ω_{nei}	50
ω_{connec}	1
ω_{step}	1
ω_{hor}	10
ω_{ver}	10
τ	0.01
The maximum speed of the UAV (v_{max})	0.3
The safe distance with neighbors (d_1)	0.02
The communication distance (d_2)	0.04
The safe distance with obstacles (d_3)	0.02
The number of obstacles (M)	6
The learning rate of the actor	0.001
The learning rate of the critic	0.005
Replay buffer size (D)	100000
Batch size (λ)	500
The discount rate (γ)	0.9
Activation function	Sigmoid

B. Convergence Comparison

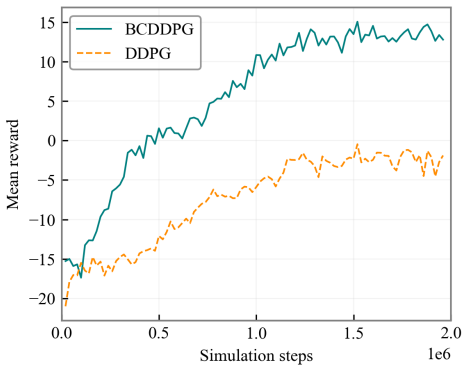
To validate the effectiveness of our model, we first make a convergence comparison between DDPG and BCDDPG by showing the curves of mean reward versus simulation steps. In addition to the different network architecture, DDPG and BCDDPG use the same simulation parameters. As shown in Fig. 5, we train DDPG and BCDDPG with different numbers of UAVs for $N = 6$, $N = 9$, and $N = 12$ respectively. As the number of UAVs increases, the environment complexity increases, too. It shows that BCDDPG converges to a better convergence performance regardless of the number of UAVs, whereas DDPG cannot converge when the number of UAVs rises to 12. More specifically, as the number of UAVs increases, the performance gap between BCDDPG and DDPG becomes larger and larger. When the environment is relatively simple ($N = 6$), both BCDDPG and DDPG can handle the cooperation among UAVs well and drive the multi-UAV system to move to the target in a flocking way. However, when the environment becomes complicated ($N = 12$), DDPG fails to deal with the contradiction among different rewards, i.e., approaching the target, collision avoidance, and connectivity maintenance. Finally, it causes DDPG to not converge.

The main reasons for the performance improvement of BCDDPG over DDPG come from the following two aspects: First, BCDDPG uses four sub-actor networks to generate the actions to be executed in the next step, whereas DDPG only uses one actor network. This network architecture helps BCDDPG extract valid information from the environment state for different sub-behaviors and generate higher-quality actions in a behavior-coupling way. Second, BCDDPG uses an LSTM network in the sub-actor network, whereas DDPG only uses the full-connected network. As we all know, the LSTM network is a kind of recurrent neural network and performs better in processing time sequence data. LSTM helps the UAV better understand the motion trend of its neighbors and generate better actions in coordination with them.

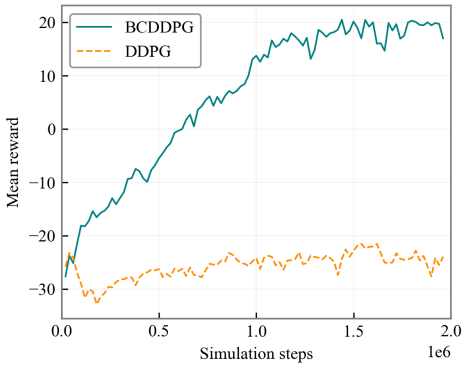
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3127873, IEEE Internet of Things Journal



(a) $N = 6$



(b) $N = 9$



(c) $N = 12$

Fig. 5. Convergence comparison between DDPG and BCDDPG for $N = 6$, $N = 9$, and $N = 12$ respectively.

C. Performance Metrics

After the training process is finished, some performance evaluation experiments are needed to validate the model. Intuitively, the goal of the methods for flocking motion of multi-UAV systems is to reach the target in the shortest path and time. But how to quantitatively analyze the performance of these methods and the influence factors behind them? Before we make a comparison between BCDDPG and other flocking motion methods, we define the following metrics.

- **Average arrival rate (F_1)**: This metric is used to evaluate the successful arrival rate of the flocking motion methods. Suppose that N_{arri} UAVs have reached the target, the average arrival rate can be denoted as the ratio of N_{arri} and N (the initial number of UAVs). In order to reduce the experiment error, every performance metric is averaged after multiple experiments. Due to limited space, no more tautology below.
- **Velocity consistency (F_2)**: The velocity consistency is one of the three rules in Boids model, it is defined in our experiment as

$$F_2 = \frac{\left\| \mathbf{v}_i + \sum_{j \in \mathcal{N}_i} \mathbf{v}_j \right\|}{\|\mathbf{v}_i\| + \left\| \sum_{j \in \mathcal{N}_i} \mathbf{v}_j \right\|}. \tag{21}$$

  The size of velocity consistency ranges from 0 to 1, and the better the velocity consistency, the bigger its size.
- **Average collision rate (F_3)**: This metric is used to evaluate the proportion of UAVs colliding with obstacles and neighbors in a whole episode. It is usually the key reason why UAVs cannot reach the target successfully.
- **Average stray rate (F_4)**: Sometimes there is no collision between UAVs and obstacles or neighbors, but there are still a few UAVs that did not reach the target. That is because UAVs are stray into the local minimum point of the environment. We define the number of these stray UAVs as N_{stra}, then the average stray rate can be denoted as the ratio of N_{stra} and N.
- **Average collision rate (F_5)**: This metric is used to calculate the average travel distance that it takes for UAVs to reach the target. The travel distance is proportional to energy consumption. That is to say, the shorter the average travel distance, the higher the energy efficiency.
- **Average collision rate (F_6)**: This metric is to evaluate the total steps it takes for UAVs to reach the target. It is a very important performance metric for the flocking motion of multi-UAV systems, and we always hope it can be smaller.
- **Average collision rate (F_7)**: In addition to velocity matching, the key of another two rules of Boids model, i.e., flocking centering and collision avoidance, is to maintain a proper distance with neighbors. In other words, average neighbor distance is an intuitive reflection of the synergy effect. Therefore, it is used to evaluate the synergy effect of multi-UAV systems.
- **Average collision rate (F_8)**: This metric is to evaluate whether the UAV takes full advantage of its maximum velocity or not. It is defined as the average of the ratio of the magnitude of the velocity at each step and $v_{\max}$.

D. Performance Comparison

In addition to the RL method DDPG, we choose a traditional flocking motion method APF proposed in [11] as another baseline. As the performance of DDPG drops dramatically with the increasement of UAVs and even does not converge, we choose the trained model with 6 UAVs of DDPG and BCDDPG as our testing model. For the implementation of APF, we refer to the approach in [11] and make some fine-tuning to adapt to our simulation environment. To illustrate

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3127873, IEEE Internet of Things Journal

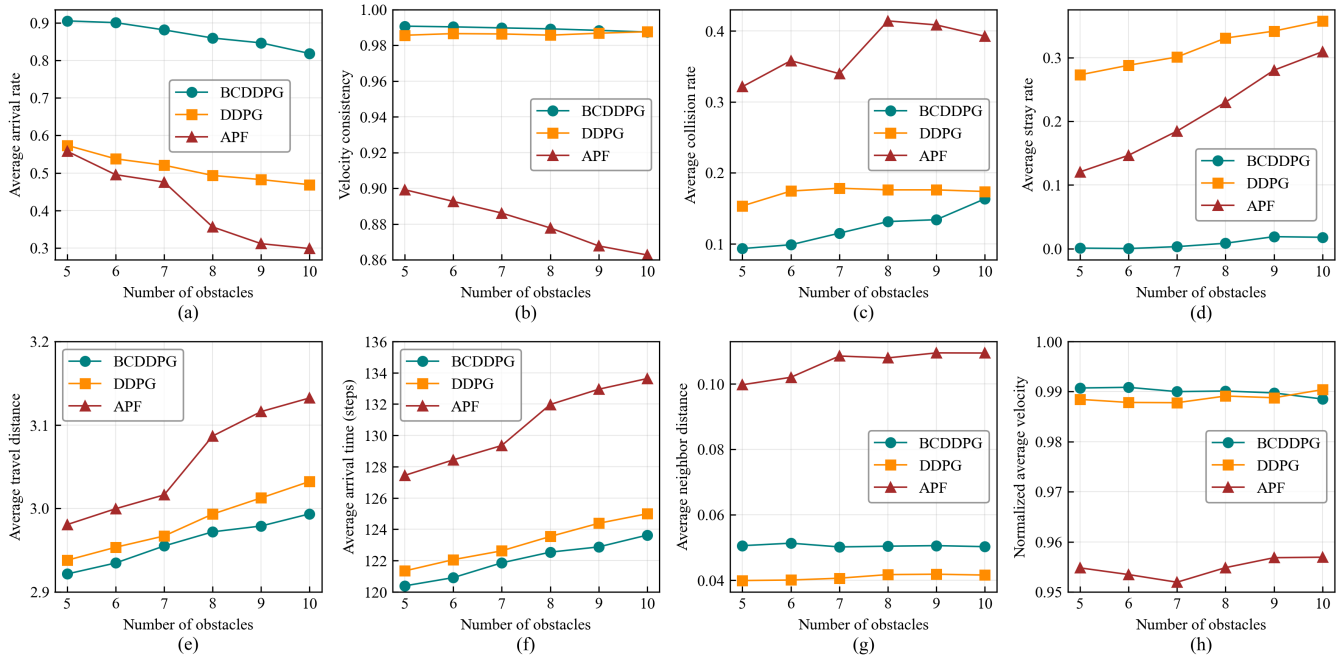


Fig. 6. The comparison of average arrival rate, velocity consistency, average collision rate, average stray rate, average travel distance rate, average arrival time, average neighbor distance and normalized average velocity among BCDDPG, DDPG and APF.

the generalization ability of our model, we use the models which are trained with 6 obstacles to test the performance of BCDDPG, DDPG, and APF in the case of 5 to 10 obstacles. Fig. 6 gives the performance comparison among BCDDPG, DDPG, and APF. Every piece of data in Fig. 6 is the average of 500 testing episodes.

As shown in Fig. 6(a), with the increasement of obstacles from 5 to 10, the performance of BCDDPG decreases slowly and always maintains an average arrival rate above 80%. Although the performance of DDPG also decreases slowly, it achieves a poor average arrival rate of less than 60%. APF performs the worst in this round of comparison, and the average arrival rate only is 30% when the number of obstacles is 10. It is unqualified as a flocking motion method in this environment.

Fig. 6(b) shows the curves of velocity consistency versus the number of obstacles. Both BCDDPG and DDPG achieve a higher velocity consistency than APF, which is close to 1. In other words, the RL methods perform well in the cooperation among neighbors. That is because the RL methods give reward to the joint actions of agents that can improve the velocity consistency. However, the velocity consistency of APF is linearly decreasing with the rise of the number of obstacles. The biggest problem of APF is that the repulsive force will degrade the velocity consistency, especially in the environment with more obstacles.

Fig. 6(c) and Fig. 6(d) give the results of average collision rate and average stray rate. Although DDPG has a relatively low collision rate, its average stray rate is higher than BCDDPG and APF. As a result, DDPG may lead to the UAV stray into the local minimum point, and eventually, lead to a low average arrival rate. The average collision rate and average stray rate of BCDDPG always maintain a low level, whereas APF both keeps a relatively high value. The above analysis of Fig. 6(c) and Fig. 6(d) can also partly explain the result of Fig. 6(a).

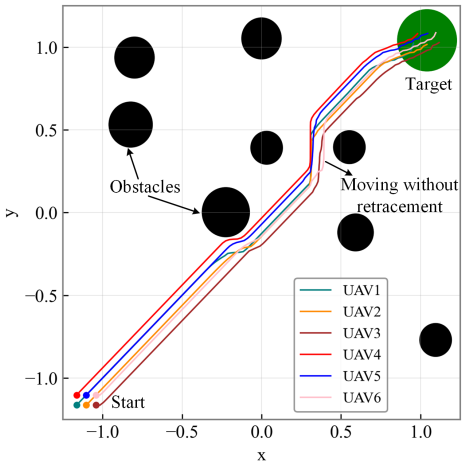
Average travel distance and average travel time are another two important metrics of flocking motion methods. The variation trends of these two metrics for BCDDPG, DDPG, and APF are similar with the rise of the number of obstacles as shown in Fig. 6(e) and Fig. 6(f). All three show a linear growth trend and the sort of them is APF, DDPG, and BCDDPG. The performance of BCDDPG and DDPG is close, whereas APF is significantly higher than them. In other words, BCDDPG uses less energy and less time to achieve the flocking motion task than DDPG and APF.

Fig.6 (g) shows the variation of average neighbor distance of BCDDPG, DDPG, and APF versus the number of obstacles. As we can see, the average neighbor distance of BCDDPG stays around 0.04, DDPG stays around 0.05, but APF is over 0.1. The larger the neighbor distance of APF, the sparser the topology of the multi-UAV system, which leads to the degradation of the synergy effect of the multi-UAV system. BCDDPG and DDPG always drive the UAV to keep a proper distance from neighbors, which is conducive to the information interaction between UAVs.

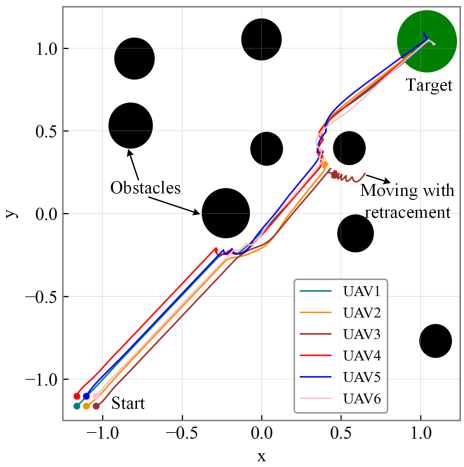
Fig.6 (h) describes the normalized average velocity of BCDDPG, DDPG, and APF. Regardless of the number of obstacles, DDPG and BCDDPG almost take full advantage of the maximum velocity, whereas APF only can use 95% around the maximum velocity. The main reason for this result of APF is that the velocity of the UAV will drop rapidly due to the repulsive force when the UAV is close to the obstacles. There is no repulsive force in RL methods and they can achieve higher velocity utilization.

To summary, BCDDPG uses less time, higher energy effi-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3127873, IEEE Internet of Things Journal



(a) The trajectories generated by BCDDPG



(b) The trajectories generated by APF

Fig. 7. The trajectories of UAVs in a specific testing environment.

ciency, and better synergy effect to achieve the flocking motion task than DDPG and APF. And BCDDPG performs better in average arrival rate, average collision rate, average stray rate, and other performance metrics. To better illustrate the advantage of BCDDPG over the traditional APF method, we design a specific environment to observe the trajectories and motion state of UAVs that adopt BCDDPG and APF as the flocking motion method, respectively.

Fig. 7 shows the trajectories of 6 UAVs in a specific testing environment with 8 obstacles that the UAVs never have meet in the training stage. Fig. 7(a) shows the UAVs trained by BCDDPG move to the target in a highly cooperative way. All the UAVs successfully arrive at the target and there is no retracement during the movement process. The neighbor distance of UAVs always keeps a proper value regardless of the position of the UAV. Fig. 7(b) shows the trajectories of UAVs generated by APF. As we can see, not all the UAVs have arrived at the target and UAV3 falls into the local minimum point of the environment. Each UAV has experienced different degrees of retracement when they attempt to avoid obstacles. This also shows that the obstacle avoidance way of APF is inefficient compared to BCDDPG.

Fig. 8 and Fig. 9 give the motion state of multi-UAV systems controlled by BCDDPG and APF at different times in the same environment as Fig. 7. The number of steps in one episode is 150 and there is no obstacle within the first few steps. Therefore, we choose the motion state of UAVs at 60s, 75s, 90s, and 120s as our observation objects (one step is just one second). As we can see, when the motion time is 60s, the UAVs meet the first obstacle that needs to be avoided. Both BCDDPG and APF successfully drive the UAVs to avoid the obstacle. The topology of UAVs generated by BCDDPG is well-organized, whereas APF is chaotic. When the motion time reaches 75s, the topology of UAVs generated by BCDDPG still maintains well-organized, and the topology of UAVs generated by APF shows no improvement. When the motion time is 90s, the topology of UAVs generated by BCDDPG becomes slightly chaotic and the distance among UAVs becomes shorter, but there is no collision among UAVs. However, the topology of UAVs generated by APF becomes very chaotic at this time, and UAV1 and UAV2 are colliding with each other. When the motion time reaches 120s, the UAVs driven by BCDDPG almost all have arrived at the target. On the contrary, there is no UAV driven by APF that has arrived at the target, and UAV3 and UAV4 have fallen far behind the team. The above results show that BCDDPG algorithm has a higher generalization ability than APF. It can be applied to a new environment without parameter finetuning.

VI. Conclusion and Future Work

In this paper, we focus on how to apply DRL to the flocking motion problem of multi-UAV systems. We propose a digital twin enabled DRL method named BCDDPG for this problem, where the digital twin of the multi-UAV system is built in the central server to train the DRL model. After the training stage, the DRL model can be quickly deployed to the real-world UAV through the connections between the physical entity and the digital model.

BCDDPG reduces the entry barrier of designing a flocking motion algorithm for those engineers without professional domain knowledge. Inspired by the flocking behavior of animals, BCDDPG uses a behavior-coupling network architecture to help UAVs understand the complexity of the dynamic environment. Besides, BCDDPG adopts the LSTM network in its sub-actor network to help UAVs mine more useful information in historical data. Simulation results show that driven by BCDDPG, the multi-UAV system moves to the target in a flocking way successfully. Extended simulations also compare BCDDPG with DDPG and the traditional flocking motion method, and BCDDPG performs better than them in all metrics.

In our future work, we will explore the methods of building a high-fidelity digital twin model of the multi-UAV system. A basic assumption of the digital twin enabled DRL training framework is that the update rate of the digital model and the computation resources are adequate. In fact, both of them will affect the performance of DRL methods in non-ideal environments. Therefore, we will study the influence of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3127873, IEEE Internet of Things Journal

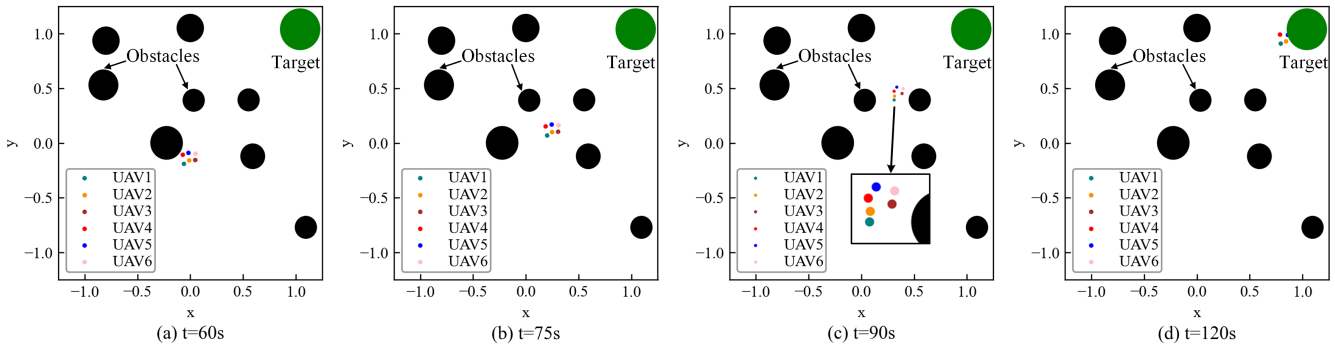


Fig. 8. The motion state of UAVs controlled by BCDDPG at different times in a specific testing environment with 8 obstacles.

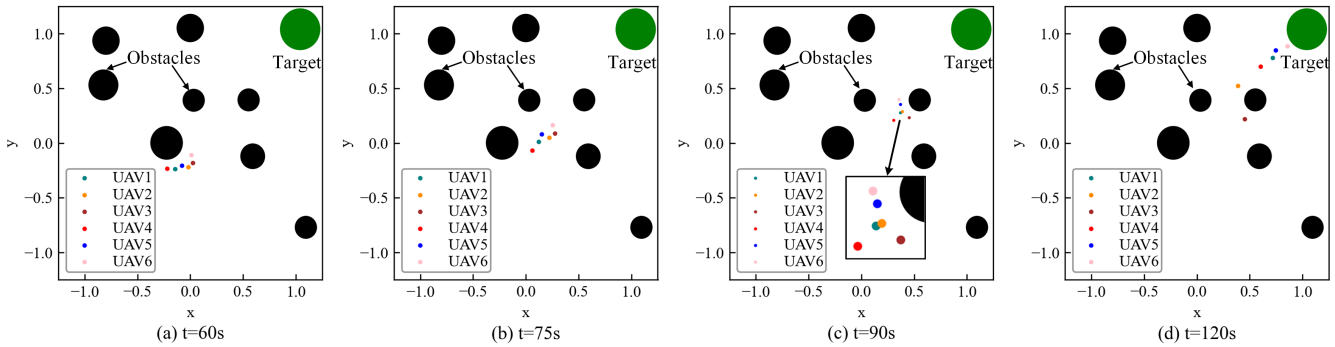


Fig. 9. The motion state of UAVs controlled by APF at different times in a specific testing environment with 8 obstacles.

communication on the performance of DRL methods in real-world environments and consider how to integrate DT with distributed training methods for more practical application. In addition, we will study the continuous evolution ability of our DRL model in the dynamic environment of the high dynamic ultra- dense device-to-device (D2D) cell-free network to enhance the generalization ability of our model.

REFERENCES

[1] Q. Kuang, X. Jin, Q. Zhao and B. Zhou, "Deep Multimodality Learning for UAV Video Aesthetic Quality Assessment," *IEEE Trans. Multimedia*, vol. 22, no. 10, pp. 2623-2634, Oct. 2020.

[2] M. Atif, R. Ahmad, W. Ahmad, L. Zhao and J. J. P. C. Rodrigues, "UAV-Assisted Wireless Localization for Search and Rescue," *IEEE Syst. J.*, vol. 15, no. 3, pp. 3261-3272, Sept. 2021.

[3] H.-M. Chuang, D. He, and A. Namiki, "Autonomous Target Tracking of UAV Using High-Speed Visual Feedback," *Appl. Sci.*, vol. 9, no. 21, p. 4552, Oct. 2019.

[4] W. Xiao, M. Li, B. Alzahrani, R. Alotaibi, A. Barnawi and Q. Ai, "A Blockchain-Based Secure Crowd Monitoring System Using UAV Swarm," *IEEE Netw.*, vol. 35, no. 1, pp. 108-115, Jan. 2021.

[5] Indu, R. P. Singh, H. R. Choudhary and A. K. Dubey, "Trajectory Design for UAV-to-Ground Communication With Energy Optimization Using Genetic Algorithm for Agriculture Application," *IEEE Sens. J.*, vol. 21, no. 16, pp. 17548-17555, Aug. 2021.

[6] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Comput. Graph.*, vol. 21, no. 4, Jul. pp. 25–34, Jul. 1987.

[7] T. Vicsek, A. Cziroók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-deriven particles," *Phys. Rev. Lett.*, vol. 75, no. 6, pp. 1226–1229, Aug. 1995.

[8] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. Automat. Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[9] H. Su, X. Wang and Z. Lin, "Flocking of multi-agents with a virtual leader," *IEEE Trans. Automat. Control*, vol. 54, no. 2, pp. 293-307, Feb. 2009.

[10] Q. Luo and H. Duan, "Distributed UAV flocking control based on homing pigeon hierarchical strategies," *Aerosp. Sci. Technol.*, vol. 70, pp. 257–264, Nov. 2017.

[11] H. Zhao, H. Liu, Y.-W. Leung, and X. Chu, "Self-adaptive collective motion of swarm robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 4, pp. 1533-1545, Oct. 2018.

[12] A. Carrio, C. Sampedro, A. Rodriguez-Ramos, and P. Campoy, "A review of deep learning methods and applications for unmanned aerial vehicles," *J. Sens.*, vol. 2017, no. 3296874, pp. 1–13, Aug. 2017.

[13] M. Pan, C. Zhang, P. Li and Y. Fang, "Joint routing and link scheduling for cognitive radio networks under uncertain spectrum supply," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 2237-2245.

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

[15] D. Shi, J. Ding, S. M. Errapotu, H. Yue, W. Xu, X. Zhou, and M. Pan, "Deep *Q*-Network-Based Route Scheduling for TNC Vehicles With Passengers' Location Differential Privacy," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7681-7692, Oct. 2019.

[16] J. Schrittwieser *et al.*, "Mastering Atari, Go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, Dec. 2020.

[17] O. Vinyals *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350-354, Oct. 2019.

[18] C. Xu, S. Liu, C. Zhang, Y. Huang, Z. Lu, and L. Yang, "Multiagent reinforcement learning based distributed transmission in collaborative cloud-edge systems," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1658-1672, Feb. 2021.

[19] W. Zhao, J. P. Queralta, T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2020, pp. 737–744.

[20] E. Glaessgen and D. Stargel, "The digital twin paradigm for future NASA and U.S. Air Force vehicles," in *Proc. 53rd AIAA/ASME/ASCE/AHS/ASC Struct. Struct. Dyn. Mater. Conf.*, Apr. 2012, p. 1818.

[21] F. Tao, H. Zhang, A. Liu and A. Y. C. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2405-2415, Apr. 2019.
[22] L. Lei, G. Shen, L. Zhang, and Z. Li, "Toward Intelligent Cooperation of UAV Swarms: When Machine Learning Meets Digital Twin," *IEEE Netw.*, vol. 35, no. 1, pp. 386-392, Jan. 2021.
[23] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, May 2016, pp. 1-14.
[24] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2124-2136, Mar. 2019.
[25] C. Wang, J. Wang, J. Wang, and X. Zhang, "Deep reinforcement learning-based autonomous UAV navigation with sparse rewards," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6180-6190, Jul. 2020.
[26] A. Singla, S. Padakandla and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge," *IEEE Trans. Intell. Transport. Syst.*, vol. 22, no. 1, pp. 107-118, Jan. 2021.
[27] Zhou Y, van Kampen E J, and Chu Q. "Hybrid hierarchical reinforcement learning for online guidance and navigation with partial observability," *Neurocomputing*, vol. 331, pp. 443-457, Feb. 2019.
[28] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li and L. Wang, "Joint optimization of multi-UAV target assignment and path planning based on multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 146264-146272, Sep. 2019.
[29] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.
[30] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam and L. Hanzo, "Multi-agent deep reinforcement learning based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 73-84, Mar. 2021.
[31] S. M. Hung, S. N. Givigi, "A Q-learning approach to flocking with UAVs in a stochastic environment," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 186-197, Jan. 2017.
[32] Z. Xu, Y. Lyu, Q. Pan, J. Hu, C. Zhao, and S. Liu, "Multi-vehicle flocking control with deep deterministic policy gradient method," in *Proc. IEEE 14th Int. Conf. Control Autom. (ICCA)*, Jun. 2018, pp. 306–311.
[33] C. Yan, X. Xiang, and C. Wang, "Fixed-Wing UAVs flocking in continuous spaces: A Deep reinforcement learning approach," *Rob. Auton. Syst.*, vol. 131, p. 103594, Jun. 2020.
[34] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," 2018, *arXiv:1706.02275*. [Online]. Available: https://arxiv. org/abs/1706.02275
[35] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Hoboken, NJ, USA: MIT Press, 2018.
[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

Gaoqing Shen received the B.S. degree in information engineering and the M.S. degree in communication and information system from Nanjing University of Aeronautics and Astronautics, Nanjing, China in 2016 and 2019, respectively. He is currently working toward the Ph.D. degree in communication and information systems with Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests are in reinforcement learning and swarm intelligence.

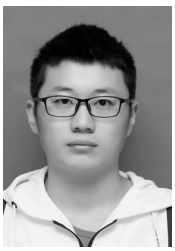
Lei Lei received the B.S. degree in electronic and information engineering from Northwestern Polytechnical University, Xi'an, China in 2002, and the Ph.D. degree in communication and information systems from Nanjing University of Aeronautics and Astronautics, Nanjing, China in 2008. He is currently a professor with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics. His current research interests are in wireless ad hoc networks and UAV swarms.

Zhilin Li received the B.S. degree in electronic information science and technology from Central South University, Changsha, China, in 2016, and the M.S. degree in communication and information systems from Nanjing University of Aeronautics and Astronautics, Nanjing, China in 2019. He is currently working toward the Ph.D. degree in communication and information systems with Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include multi-fidelity modeling and data fusion.

Shengsuo Cai was born in Shandong province, China, in 1988. He is a Lecturer in the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics. He received his Bachelor and Master's degrees in electronic information science and technology from Nanjing University of Aeronautics and Astronautics in 2011 and 2014, respectively. His current research interests are in the area of aeronautical ad hoc networks.

Lijuan Zhang received the B.Eng. degree in information security from the University of Southwest Jiaotong University, Chengdu, China, in 2010, and the Ph.D. degree in electronic engineering from James Cook University, Cairns, Australia, in 2018. She is currently an associate researcher in the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China. Her research interests include RFID communication and MAC protocol design.

Pan Cao received the B.S. degree in electronic information engineering from Huaiyin Normal University, Huaian, China, in 2017, and the M.S. degree electronic and communication engineering from Nanjing University of Posts and Telecommunications, Nanjing, China in 2020. She is currently working toward the Ph.D. degree in communication and information systems with Nanjing University of Aeronautics and Astronautics, Nanjing, China. Her research interests include UAV swarm collaborative control and swarm intelligence.



Xiaojiao Liu received the B.S. degree in information engineering in communication and information system from Nanjing University of Aeronautics and Astronautics, Nanjing, China in 2020. He is currently working toward the M.S degree in communication and information systems with Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests are in Ad Hoc Networks and UVA swarms.