Transformer-Based Reinforcement Learning for Scalable Multi-UAV Area Coverage

Dezhi Chen, Qi Qi, *Senior Member, IEEE*, Qianlong Fu, Jingyu Wang, *Senior Member, IEEE*, Jianxin Liao, and Zhu Han, *Fellow, IEEE*

***Abstract*— Compared with terrestrial networks, unmanned aerial vehicles (UAVs) have the characteristics of flexible deployment and strong adaptability, which are an important supplement to intelligent transportation systems (ITS). In this paper, we focus on the multi-UAV network area coverage problem (ACP) which require intelligent UAVs long-term trajectory decisions in the complex and scalable network environment. Multi-agent deep reinforcement learning (DRL) has recently emerged as an effective tool for solving long-term decisions problems. However, since the input dimension of multi-layer perceptron (MLP)-based deep neural network (DNN) is fixed, it is difficult for standard DNN to adapt to a variable number of UAVs and network users. Therefore, we combine Transformer with DRL to meet the scalability of the network and propose a Transformer-based deep multi-agent reinforcement learning (T-MARL) algorithm. Transformer can adapt to variable input dimensions and extract important information from complex network states by attention module. In our research, we find that random initialization of Transformer may cause DRL training failure, so we propose a baseline-assisted pre-training scheme. This scheme can quickly provide an initial policy model for UAVs based on imitation learning, and use the temporal-difference(1) algorithm to initialize policy evaluation network. Finally, based on parameter sharing, T-MARL is applicable to any standard DRL algorithm and supports expansion on networks of different sizes. Experimental results show that T-MARL can make UAVs have cooperative behaviors and perform outstandingly on ACP.**

***Index Terms*— Unmanned aerial vehicles, area coverage, multi-agent deep reinforcement learning, transformer.**

Manuscript received 8 May 2023; revised 2 December 2023; accepted 8 January 2024. This work was supported in part by the National Natural Science Foundation of China under Grant U23B2001, Grant 62171057, Grant 62101064, Grant 62201072, Grant 62001054, and Grant 62071067; in part by the National Post-Doctoral Program for Innovative Talents under Grant BX20230052; in part by China Post-Doctoral Science Foundation under Grant 2023TQ0039; in part by the Ministry of Education and China Mobile Joint Fund under Grant MCM20200202 and Grant MCM20180101; in part by the BUPT-China Mobile Research Institute Joint Innovation Center; and in part by the Beijing University of Posts and Telecommunications (BUPT) Excellent Ph.D. Students Foundation under Grant CX2021107. The Associate Editor for this article was L. Wang. *(Dezhi Chen and Qi Qi are co-first authors.) (Corresponding authors: Jingyu Wang; Jianxin Liao.)*

Dezhi Chen, Qi Qi, Qianlong Fu, Jingyu Wang, and Jianxin Liao are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with EBUPT.COM, Beijing 100191, China (e-mail: chendezhi@bupt.edu.cn; qiqi8266@bupt.edu.cn; fuqianlong@bupt.edu.cn; wangjingyu@bupt.edu.cn; liaojx@bupt.edu.com).

Zhu Han is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 446-701, South Korea (e-mail: hanzhu22@gmail.com).

Digital Object Identifier 10.1109/TITS.2024.3358010

I. Introduction

UNMANNED aerial vehicles (UAVs) are recognized as an important part of 6G network and intelligent transportation systems (ITS) to support a wider range and more diverse services [1], [2], [3], [4], [5]. UAVs can be flexibly deployed at high altitudes to reduce terrain interference, and are suitable for missions in areas lacking infrastructure and as a supplement to existing networks. Unfortunately, due to the complex environment and the lacking terrestrial networks, and UAVs with limited coverage capabilities for large-scale monitoring, how to optimize the flight trajectory of UAVs has become an urgent problem to be solved.

The problem can be modeled as a multi-UAV area coverage problem (ACP) [6], [7], [8], [9], as shown in Fig. 1. In this paper, we focus on the new scenario where UAVs are used as air base stations to provide auxiliary communication coverage for ground users. Standard ACPs, such as wireless sensor networks (WSNs) [10], have a relatively fixed network topology that requires pre-designed fixed cruise locations. Conventional solvers like greedy and heuristic algorithms based on the traveling salesman problem (TSP) [11] depend heavily on human expertise. While expert design can narrow down the TSP solver's search space, it adds to the complexity and is hard to scale up. In addition, the fixed location designed by experts limits the flexibility of UAVs. Heuristic algorithms that rely on iterative solutions are time-consuming, hindering the UAVs ability to respond to emergencies. Recently, some studies [12] have discovered the potential of multi-agent deep reinforcement learning (MA-DRL) [13], [14], [15], [16], [17] in the field of long-term decision-making multi-player games, such as AlphaGo Zero and StarCraft II. Deep reinforcement learning uses a deep neural network (DNN) to fit control policies and is suitable for complex state problems. However, due to the variable number of UAVs and users, the complex dynamics of the environment, and the long-term decision-making requirements, it is a challenge to train a DNN directly through MA-DRL to form a reliable cooperate strategy.

In this paper, we consider two of the key challenges, the scalability of multi-UAV networks and the random initialization of DNN in a complex long-term task. Although MA-DRL has achieved certain successes in multi-player games with a specific number of players, existing MA-DRL methods remain an open problem in the area of scalability. In order to be unified with the original DRL work, we use "actor" to refer

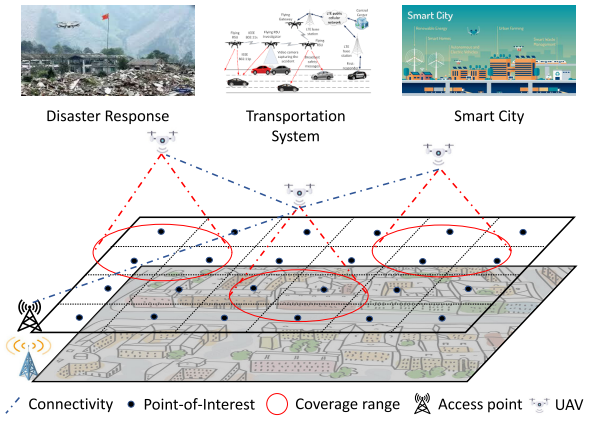

Fig. 1. Multi-UAV area coverage problem.

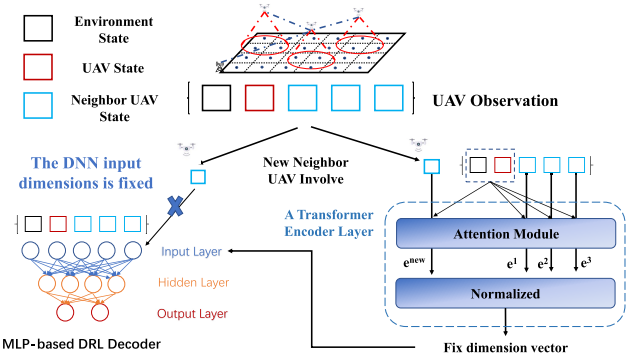

Fig. 2. MLP-based deep neural network and Transformer in the multi-UAV networks.

to policy DNN, and "critic" to refer to policy evaluation network. Existing standard multi-agent reinforcement learning methods can be divided into two categories: one is to have a centralized critic to evaluate the contribution of all agents in the group to the cooperative environment, such as COMA [15] and VDN [16]; another is that each agent retains its own actor and critic to address environmental issues facing competition or a mix of competition and cooperation, such as MADDPG [14]. However, both approaches suffer from inherent training problems in environments with large numbers of scalable agents, making it difficult for researchers to train overly large centralized critics, or to train unique policy networks for each agent. We propose a Transformer-based deep multi-agent reinforcement learning (T-MARL) algorithm. Our algorithm is based on parameter sharing [18], which can extend the standard DRL algorithm to large-scale agents in the same state action space, and provide a promising solution to the scalability problem in the network.

In multi-UAV networks, the standard DNN based on the fixed dimensional MLP is not applicable due to the variable dimensionality of the input state space, as shown in Fig. 2. We introduce the Transformer encoder [19] into the DNN, which can weight the observable neighbor UAVs based on the UAV's current state and convert the observed state into a fixed-dimensional vector for subsequent DNN input. The complexity of long-term task and the random initialization of the actor can lead to a large number of invalid actions during the sampling phase, resulting in training failure. To address this, we decouple the standard DRL training process and the DNN

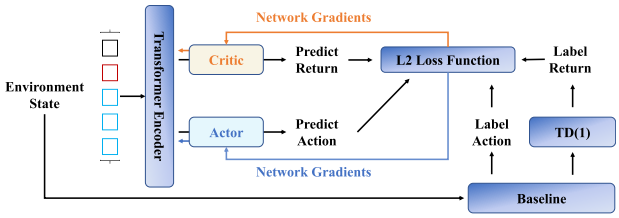

Fig. 3. Baseline-assisted pre-training scheme.

initialization process. We use imitation learning to initialize the actor by fitting a small number of baseline samples, effectively limiting the actor's output range and avoiding invalid samples in DRL training, as shown in Fig. 3. The random initialization of the critic can lead to incorrect evaluation of the UAV policy. To solve this, we choose a supervised learning method based on temporal-difference (1) as a label and train a critic to match the actor. Simulations demonstrate that under the combined effect of Transformer and pre-training, T-MARL improves the multi-UAV network performance compared to the baseline algorithm in the literature.

The main contributions of this paper are as follows:

- We propose a Transformer-based multi-agent reinforcement learning algorithm to address the area coverage problem in multi-UAV networks.
- In the multi-UAV network area coverage problem, DNNs with fixed input dimensions cannot be deployed on a variable number of UAVs. We introduce a Transformer and parameter sharing scheme to solve the scalability problem in the network by variable input dimensions and sharing policies on homogeneous agents.
- Effectively training multi-agent cooperative policies in networks remains an open problem. We found that the random initialization of DNN will lead to the agent cannot generate valid samples during the training phase. To address this issue, we use baseline-assisted pre-training scheme to initialize the actor and critic of UAVs.
- The experimental results show that the T-MARL algorithm can effectively improve the performance of UAVs. Compared with the baseline algorithm, the average coverage score is improved by 39.7%, and the fairness index is improved by of 45.3%. Ablation experiments show that Transformer and baseline-assisted pre-training schemes are effective in improving the performance respectively.

The rest of this paper is organized as follows: Section II lists relevant literature on joint design of UAV communication and trajectory. Section III describes our system model of the multi-UAV area coverage problem. Section IV defines the Transformer-based multi-agent deep reinforcement learning for multi-UAV networks. Section V provides the simulation results. Finally, Section VI concludes this paper.

II. Related Work

A. UAV Area Coverage Problem

UAVs, with their flexible deployment and robust adaptability, offer a significant enhancement to traditional terrestrial networks and are poised to play a crucial role in the future of

ITS [20], [21], [22]. One of the key challenges in UAV-assisted ITS is multi-objective optimization for UAV path planning, with the UAV area coverage problem being a notable example [6], [9], [12], [23], [24], [25].

The single-agent DRL-based energy-efficient control for coverage and connectivity (DRL-EC^3) proposed in [6] was a pioneering effort in defining the ACP, demonstrating superior performance over random and greedy algorithms. The ACP was further extended to three-dimensional positioning in [23], and the impact of mutual interference between UAVs on ACP was explored in [24]. However, these early works relied on a central DNN for supporting multi-UAV systems, which posed scalability challenges. The works in [9] and [25] addressed these issues by extending the DRL-EC^3 algorithm to a distributed control scheme using a dedicated decision network for each agent. While this modification addresses the issue of an overly large central network, it does not eliminate the need for training a new policy network when the number of agents changes. Additionally [9] and [25] use The structure of a multi-layer perceptron (MLP) with fixed-dimensional environment state input. When the number of UAVs increases, the MLP-based dedicated DNN needs to be retrained, and when the number of UAVs decreases, some required dimensions have no input, which may lead to DNN decision errors.

Further advancements in system scalability were realized in [12] with the incorporation of another conventional DRL algorithm, TRPO [26], and an innovative MLP-based DNN state input technique, feature embedding [27]. In the process of [12] reproduction, we found that the premise for the work of [12] to take effect is that the initial positions of the UAVs are evenly distributed within the map. This is because any arbitrary flight action can easily discover uncovered user nodes, thereby obtaining reward signals. However, with a slight modification, such as placing the initial position in a corner of the map, the UAV needs more orderly actions, such as accelerating towards the other side of the map, to obtain reward signals to support subsequent training. But due to the random initialization of DNN model parameters, the actions of the UAV are disordered at the beginning, making it impossible to obtain reward signals, which leads to the failure of RL training.

In T-MARL, Transformer avoids the problem of the fixed number of agents caused by the fixed input dimension of the neural network and can be easily extended to large-scale networks through the parameter sharing method. Besides, We introduce a pre-training method that utilizes a baseline algorithm to provide initial parameters for the Transformer network. This method further enhances network performance through deep reinforcement learning.

B. Transformer in UAVs

Vaswani et al. proposed Transformer [19], and it has shown amazing performance in computer vision and natural language processing (NLP) tasks [28]. Recently, some methods of applying Transformer in DRL have appeared in reinforcement learning demonstration scenes such as Atari [29], Gym [30], and StarCraft [31]. Transformer consists of two parts: encoder and decoder. Transformer encoder use attention module which can weight other parts of the sentence according to the current word to generate the corresponding fixed dimensional vector to facilitate the processing of the next layer of DNN. Similar to sentences of different lengths in NLP, Transformer encoder can handle variable input dimensions caused by the number of variable network nodes.

With the expansion of Transformer from NLP tasks to computer vision (CV) tasks, Transformer have also appeared in the UAV field. Vision-based target tracking is an important task in the UAV field. How to deploy large-scale Transformers in vision tasks to UAV devices with limited resources has become a key challenge [32], [33], [34], [35], [36]. Different from the above-mentioned mainstream challenge: how to deploy Transformer, this paper focuses on the Transformer's ability to flexibly expand the input dimension (such as the number of network nodes), and applies Transformer to UAV trajectory planning. Zhu et al. [37] used the Transformer in the single UAV scenario to solve the age of information minimization problem in data collection. Wang et al. [38] used Transformer to solve the dynamic network states changes problem in single UAV assisted wireless sensor networks. Similar to the work of Zhu [37], Wang [38] also used Transformer to embed the network states. In this paper, we use Transformer to embed dynamical network states and extend the Transformer to multi-agent field through parameters sharing. However, our investigation find that random initialization hinders Transformer network training. Therefore, we introduce the pre-training method which use the baseline algorithm to provide initial Transformer network parameters and further improving network performance by deep reinforcement learning.

III. System Model

This section defines the multi-UAV area coverage problem [6], [9], [12], and the evaluation metrics of multi-UAV networks. We explain how to model the multi-UAV ACP as MDP to meet the requirements of MA-DRL in Section III-A. Besides, the UAV channel model and energy model are described in Section III-B. Table I lists the important functions and variables in this paper.

A. Multi-UAV Area Coverage Problems

We consider N UAVs flying on a specific altitude plane and provide communication services for users in a target area. UAVs make autonomous decisions, and allow mutual communication to obtain environmental information. UAVs are indexed by i. The target area is divided into K cells. UAVs provide services to all users in a certain range at the same time. To simplify the problem, each cell is represented by its geometric center, called Point-of-Interest (PoI), as shown in Fig. 1. When the PoIs fall into the coverage of a UAV, the cells are considered to be covered. UAVs need to cover PoIs as much as possible and reduce energy consumption through motion control within the specified time range T.

To evaluate the performance of the multi-UAV network, we mainly focus on three parts: (1) covering users for as

TABLE I
LIST OF FUNCTIONS AND VARIABLES

Functions	Meaning	
N, K	Number of UAVs and PoIs	
$i(j), k$	The index of UAVs and PoIs	
c_t	Corresponding coverage score	
f_t	Corresponding fairness index	
E, e	Energy function of UAVs	
n_i	UAV coverage PoIs number	
m_i	The number of neighbor UAVs	
π	Policy	
PL	Path loss	
I	Channel interference	
C	A2G Transmission rate	
R	Achievable throughout	
Parameters	**Meaning**	**Default value**
γ	Discount factor	0.99
σ	Learning rate	0.01
h	UAV flight height	200m
f_0	A2A carrier frequency	2GHz
$f_c, f_{i,k}$	A2G carrier frequency	20MHZ
N_s	Gaussian noise power	10e-8W
D_0	Delay threshold	100ms
P_T	Transmission power of UAVs	200mW
P_H	Hovering power of UAVs	168W
$v_{\min}, P_{F,\min}$	UAV minimum consumption	10m/s, 126w
P_c	UAV control system power	30W
P_{F0}	The blade profile power	79.86W
P_{F1}	The induced power	88.63W
v_{tip}	UAV rotor tip speed	120 m/s
$\tilde{v}_0$	Mean rotor induced velocity	4.03 m/s
d_0	The fuselage drag ratio	0.6
ρ	The density of the air	1.225 kg/m^3
s	The rotor solidity	0.05
A_{disc}	The rotor disc area	0.503 s^2
E_{limit}	The maximum energy of UAV	1kWh
δ_F	The flying time	5s
δ_T	The transmission time	25s

long as possible; (2) covering each user in as fair a time as possible; (3) performance consumption should be less than battery capacity. We use the corresponding coverage score c_k to represent the coverage metric of PoI k:

$$c_{t,k} = \begin{cases} 1, \text{ PoI is covered by a UAV} \\ \qquad \text{and channel is active,} \\ 0, \text{ otherwise} \end{cases}$$
$$c_k = \frac{\sum_{t=1}^{T} c_{t,k}}{T}, \quad k \in \{1, \cdots, K\}. \tag{1}$$

PoI k may be covered by multiple UAVs at the same time. PoIs average coverage score defines the coverage effect of multi-UAV networks by

$$\bar{c} = \frac{\sum_{k=1}^{K} c_k}{K}. \tag{2}$$

However, under the definition of average coverage score, UAVs can achieve the same performance by covering a part of PoIs for a long time. This makes UAVs give up PoIs that require power to achieve coverage. The UAV team needs to take care to avoid unfair coverage results. In other words, it is undesirable that some PoIs are covered for a long time and some PoIs are ignored, even if the average coverage score is equal. Therefore, we use the corresponding fairness index

$$f = \frac{\left(\sum_{k=1}^{K} c_k\right)^2}{K \sum_{k=1}^{K} (c_k)^2} \tag{3}$$

to represent the quality of experience in covering fairness.

Multi-UAV networks need to maximize the total coverage score and fairness index of PoIs while satisfying constraints. At the time t, the overall performance in a single time slot t can be expressed as:

$$\max \ \zeta = f \cdot \bar{c},$$
$$\text{s.t. } E_i \geq \sum_{i=1}^{N} \Delta e_{i,t}, \quad i \in \{i, \cdots, N\}, \tag{4}$$

where $\Delta e_{i,t} = E_{i,t} - E_{i,t-1}$ is the incremental energy consumption of UAV i, and E_i is the battery capacity of the UAV i.

In our scenario, the next state of the environment is only related to the current state and UAVs actions, so it can be modeled as a Markov decision process (MDP) [39]. MDP is defined by a tuple $(\mathbf{N}, \mathbf{S}, \mathbf{A}, \mathbf{R}, \mathbf{P}, \gamma)$. $\mathbf{N} = \{\text{agent } 1, \ldots, \text{agent } i, \ldots, \text{agent } N\}$ denotes the set of N agents, and each of them is deployed in a UAV. For convenience, we use UAVs instead.

$\mathbf{S}$ is the state of the environment including all UAVs. In real-world applications, a single UAV i can only observe a part of the state $\mathbf{S}$ and make decisions based on the observed information $\mathbf{S}_{i,t}$ at time slot t. In recent research, the MDP based on observation state $\mathbf{S}_{i,t}$ is also called decentralized partially observed Markov decision process (Dec-POMDP). In this paper, $\mathbf{S}_{i,t}$ consists of $\left(e_{i,t}, e_{j,t}, p_{i,t}, p_{j,t}, p_{k,t}, \chi_{i,t}, c_{k,t}, n_{k,t}\right)$, where $e_{i,t}, e_{j,t}$ is the current energy of UAV i and neighbor UAV j, $p_{i,t}, p_{j,t}, p_{k,t}$ is the position information of UAV i, neighbor UAV j and PoI k, $\chi_{i,t}$ is used to indicate whether the UAV has flew out of the target area, $c_{k,t}$ is the current coverage score of PoI k, and $n_{k,t}$ is the number of UAVs covering PoI k. It is worth noting that the UAV can observe the states of the variable number of neighbor UAVs and PoIs within its communication range.

$\mathbf{A} = \left\{\mathbf{A}_{1,t}, \ldots, \mathbf{A}_{N,t}\right\}$ is the joint action set of all UAVs. In some recent research [40], [41], [42], the authors use predefined methods to provide alternative target locations for UAVs to simplify the action output problem. However, in real-world scenarios, UAVs have the need to deal with emergencies and face an unknown environment. Simple predefined target methods are no longer applicable under these conditions. Therefore, the UAVs need to output continuous linear velocity and flight angle $\mathbf{A}_{i,t} = \left(v_{i,t}, \omega_{i,t}\right)$ to control the flight trajectory. Direct output actions can reduce the iterative calculation time to one DNN inference, and provide more path options to increase the upper limit of the algorithm.

$\mathbf{R}$ is a set of predefined function $r_{i,t}\left(\mathbf{S}_t, \mathbf{A}_{i,t}\right)$ which represent the reward of UAV i outputting action $\mathbf{A}_{i,t}$ in state $\mathbf{S}_t$. $\mathbf{T}$ is the transition probability function $\mathbf{T} : \mathbf{S}_t \times \mathbf{A}_t \rightarrow \mathbf{S}_{t+1}$ which represents the dynamic characteristic of the environment. $\gamma \in (0, 1]$ is the discounted factor that represents the attention of the UAVs to future rewards. If $\gamma = 0$, UAVs do not consider

any future rewards. If $\gamma = 1$, the value of UAV future rewards will not decrease over time.

The standard DRL use the long-term discount cumulative rewards

$$R_\pi = \sum_{t}^{T} \gamma^t r\left(\mathbf{S}_t, \mathbf{A}_t\right) \tag{7}$$

to evaluation policy π, where $\pi\left(\mathbf{A}_t|\mathbf{S}_t\right)$ is agent decision policy function. $r\left(\mathbf{S}_t, \mathbf{A}_t\right)$ is global reward which is related to the single reward function $r_{i,t}\left(\mathbf{S}_t, \mathbf{A}_{i,t}\right)$ of each agent, such as linear combination [16], and non-linear combination [43]. In DRL, there are three common variants of R_π, value function $V_\pi\left(\mathbf{S}_t\right)$, action-value function $Q_\pi\left(\mathbf{S}_t, \mathbf{A}_t\right)$, and advantage function $A_\pi\left(\mathbf{S}_t, \mathbf{A}_t\right) = Q_\pi\left(\mathbf{S}_t, \mathbf{A}_t\right) - V_\pi\left(\mathbf{S}_t\right)$.

The reward r is based on (4) and is related to the entire multi-UAV networks. We further design the reward function related to the individual to avoid the mismatch between the reward and the action. We retain f_t, the fairness index of the network in the current time slot t, as the induced cooperation variable. The increase of f_t depends on the cooperative behavior between UAVs. In the early stage of the mission, if UAVs consume more energy and speed up to reach the uncovered area faster, the overall fairness will increase, and UAVs will collectively receive more rewards. The definition of the reward function is as follows

$$\begin{aligned} r_i &= f_t\left(\sum_{k=1}^{K_i} \Delta c_{k,t}\right)/K - p, \\ \Delta c_{k,t} &= \frac{1}{n_i}. \end{aligned} \tag{8}$$

$\Delta c_{k,t} = c_{k,t} - c_{k,t-1}$ is the incremental coverage score of PoI k K_i is the number of PoIs covered by UAV i. n_i is the number of UAVs covering PoIs. p is used as a large punishment to avoid situations such as complete power consumption or loss of connection in the mission.

B. UAV Model

1) Channel Model: We refer to the 3GPP specifications [44] to consider two kinds of channel models in the multi-UAV network, such as the air-to-air (A2A) channel model and the air-to-ground (A2G) channel model.

The A2A channel between UAVs is dominated by Line-of-Sight (LoS) links, whose path loss formulate as

$$\mathrm{PL}_{i,0} = 30.9 + \left[22.25 - 0.5 \cdot \log_{10}\left|h_i - h_0\right|\right] * \log_{10} d_{i,0} + 20 \cdot \log_{10} f_0 \tag{9}$$

according to 3GPP TR 36.777 [44]. The $d_{i,0}$ and $|h_i - h_0|$ in (9) are the distance and the height difference between UAV 0 and UAV i, respectively. f_0 is the carrier frequency of the A2A link. In this paper, to simplify the problem, we consider that UAVs can communicate with each other.

The A2G channel uses a down-link emergency communication network [44]. UAV base stations can connect to users (PoIs in this paper) either through LoS or Non-Line-of-Sight (NLoS) with different probabilities. The A2G path loss function formulate as (5), shown at the bottom of the page, where $d_{i,k}$ is the distance between UAV i and PoI k, and f_c is the carrier frequency of the A2G link. The probability of the LoS link formulate as (6), shown at the bottom of the page, and the probability of the NLoS is $\mathrm{PNLOS}_{i,k} = 1 - \mathrm{PLOS}_{i,k}$. If a PoI is covered by at least two UAVs, such as UAVs i and j, the interference I_k is formulated as

$$\begin{aligned} I_k &= I_{k,\mathrm{LoS}} + I_{k,\mathrm{NLoS}} \\ &= \sum_{j \in U_k, j \neq i} \mathrm{PLoS}_{j,k} \cdot P_T \cdot 10^{-\mathrm{PL}_{j,k}^{\mathrm{LoS}}/10} \\ &\quad + \sum_{j \in U_k, j \neq i} \mathrm{PNLoS}_{j,k} \cdot P_T \cdot 10^{-\mathrm{PL}_{j,k}^{\mathrm{NLoS}}/10}, \end{aligned} \tag{10}$$

where P_T is the transmission power of the UAV [45], and U_k is the set of UAVs covered by PoI k. The transmission rate allocated to UAV i and Poi k A2G channels are

$$\begin{aligned} C_{i,k}^{\mathrm{LOS}} &= \log_2\left(1 + \frac{P_T \cdot 10^{-PL_{i,k}^{\mathrm{LOS}}/10}}{I_k + N_s}\right), \\ C_{i,k}^{\mathrm{NLOS}} &= \log_2\left(1 + \frac{P_T \cdot 10^{-PL_{i,k}^{\mathrm{NLOS}}/10}}{I_k + N_s}\right), \end{aligned} \tag{11}$$

where N_s is the additive white Gaussian noise. The achievable throughput [46] of PoI k from UAV i is

$$R_{i,k} = \mathrm{PLOS}_{i,k} \cdot C_{i,k}^{\mathrm{LOS}} + \mathrm{PNLOS}_{i,k} \cdot C_{i,k}^{\mathrm{NLOS}}. \tag{12}$$

The transmission delay D depends on the data size $l_{i,k}$, the link bandwidth resource $f_{i,k}$ and the achievable throughput $R_{i,k}$, and it is described as

$$D_{i,k} = \frac{l_{i,k}}{f_{i,k} \cdot R_{i,k}}. \tag{13}$$

Channels with $D_{i,k}$ less than $D_0 = 100$ms are considered active, and in order to simplify the problem, $l_{i,k}$ and $f_{i,k}$ are set as constants.

$$\mathrm{PL}_{i,k} = \begin{cases} 30.9 + \left(22.25 - 0.5\log_{10} h\right)\log_{10} d_{i,k} + 20\log_{10} f_c, & \mathrm{LoS} \\ \max\left\{\mathrm{PL}_{i,k}^{\mathrm{LOS}}, 32.4 + \left(43.2 - 7.6\log_{10} h\right)\log_{10} d_{i,k} + 20\log_{10} f_c\right\}, & \mathrm{NLoS} \end{cases} \tag{5}$$

$$\mathrm{PLOS}_{i,k} = \begin{cases} 1, & \text{if } \sqrt{d_{i,k}^2 - h^2} < d_0 \\ \dfrac{d_0}{\sqrt{d_{i,k}^2 - h^2}} + \exp\left\{\left(-\dfrac{\sqrt{d_{i,k}^2 - h^2}}{p_1}\right)\left(1 - \dfrac{d_0}{\sqrt{d_{i,k}^2 - h^2}}\right)\right\}, & \text{if } \sqrt{d_{i,k}^2 - h^2} > d_0 \end{cases} \tag{6}$$

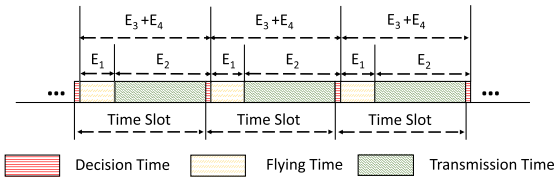

Fig. 4. The structure of time slot.

2) Energy Model: In this paper, we use rotary-wing UAVs in the network. Referring to [47], the UAV propulsion power is model as

$$P_F(v) = P_{F0}\left(1+\frac{3v^2}{v_{\text{tip}}^2}\right) + P_{F1}\left(\sqrt{1+\frac{v^4}{4\tilde{v}_0^4}}-\frac{v^2}{2\tilde{v}_0^2}\right)^{1/2} + \frac{1}{2}d_0\rho s A_{\text{disc}}v^3, \tag{14}$$

where v is the speed of the UAV, v_{tip} is the tip speed of the UAV's rotor blade, and $\tilde{v}_0$ is the mean rotor induced velocity in hover. P_{F0} and P_{F1} are the blade profile power and induced power in a hovering state respectively. d_0 denotes the fuselage drag ratio, s denotes the rotor solidity, A_{disc} is the rotor disc area, and ρ is the density of the air. From (14), when the UAV is hovering (UAV speed $v=0$), the UAV power model is $P_H = P_F(0) = P_{F0} + P_{F1}$. The transmission energy power of the UAV is P_T.

It is worth noting that the training process of the model is separated from the execution process, which is called CTDE (centralized training distribution execution) [18] in MA-DRL. The training part of the heavy load is carried out on the central server, and then the trained model is loaded onto the UAV. The UAV only performs the DNN inference part according to the input state. Similarly, in the field of computer vision, training is done in the data center, and the client only needs to input the image into the DNN once and does not need to train again. The power of the DNN inference is incorporated into the UAV control system consumption power P_c. The default values of UAV parameters are shown in table I.

The overall mission time T of UAVs to provide communication services can be divided into equal-length time slots (256 in this paper). We assume that all UAVs operate in a synchronized manner. The time axis is partitioned into equal non-overlapping time slots. In a time slot, UAVs need to be moved to cover PoIs and hovered to execute a mission. A time slot is divided into three parts: decision time, flying time δ_F and hovering time δ_T, as shown in Fig. 4. In flying time phase, a UAV flies speed is v, and the flying consumption is $E_1 = P_F(v)\delta_F$. In transmission time phase, the UAV i hover and connect to n_i PoIs, and UAV i consumption is $E_2 = (P_H + n_i P_T)\delta_T$. Decision time is the time for UAVs to perform a DNN inference, which is negligible compared to flying time and hovering time. The UAV communication with m_i neighbor UAVs with energy consumption $E_3 = m_i P_T(\delta_F+\delta_T)$. Besides, the UAV control system consumption is $E_4 = P_c(\delta_F+\delta_T)$.

IV. Algorithm

In this section, we describe the Transformer-based deep multi-agent reinforcement learning algorithm. We define the Transformer in Section IV-A, propose the Transformer-based MARL update process in Section IV-B, and design the baseline-assisted pre-training scheme in Section IV-C.

A. Solution of Scalability Challenge

Due to the variable dimensional input state caused by the variable number of UAVs in multi-UAV networks, the standard DNN based on the fixed dimensional multi-layer perceptron (MLP) is no longer applicable. If UAVs use an MLP-based DNN, sufficient input dimensionality needs to be preserved in the input layer. Otherwise, when the number of UAVs increases, the DNN needs to be retrained, and when the number of UAVs decreases, some required dimensions have no input, which may lead to network errors. Besides, based on the perspective of a single UAV, it is easy for researchers to find that UAVs need to process a large amount of heterogeneous information, such as UAVs in different states, PoIs in different locations, and impassable obstacles. The importance of this information is not the same. For example, a UAV pays great attention to the distance from obstacles to avoid accidents, but the distance from PoIs only needs to be within the coverage distance.

Vaswani et al. proposed Transformer [19], and it has shown amazing performance in computer vision and natural language processing tasks [28]. Transformer is an emerging technology that can effectively solve the problem of variable input dimensions and extract key information. Based on the Transformer architecture, agents pay attention to the environmental information related to themselves and can notice the important parts of complex heterogeneous information. Similar to sentences of different lengths in NLP, Transformer can handle variable input dimensions caused by the number of variable network nodes. Transformer can weight other parts of the sentence according to the current word to generate the corresponding fixed-dimensional vector to facilitate the processing of the next layer of DNN. In addition, MLP-based DNN requires the absolute position of the state in the input vector to remain unchanged, while in the network, the agent neighbors are dynamically changing, and it is impossible to reserve input positions for all possible neighbors. Transformer is permutation invariant to the input vector, which enables UAVs to dynamically process neighbor information. In this paper, we combine Transformer with standard DRL, propose a Transformer-based reinforcement learning algorithm, and extend it to a multi-agent environment to achieve area coverage tasks, as shown in Fig. 2.

The most important technology of the Transformer is the attention mechanism, as shown in Fig. 5. The calculation process of the attention mechanism in multi-UAV networks is as follows. First, key embedding neural network take the observation neighbor states $(s_1,\ldots,s_{i-1},s_{i+1},\ldots,s_N)$ of UAV i as input, then outputs a set of key vectors $(\mathcal{K}_1,\ldots,\mathcal{K}_{i-1},\mathcal{K}_{i+1},\ldots,\mathcal{K}_N)$. Similarly, value embedding neural network also take the observation neighbor

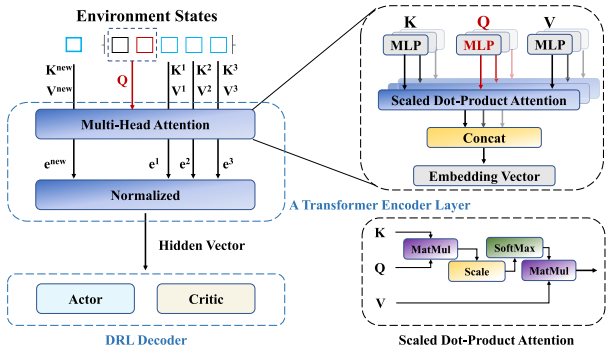

Fig. 5. The structure of attention.

states as input and output a set of value vectors $(\mathcal{V}_1, \ldots, \mathcal{V}_{i-1}, \mathcal{V}_{i+1}, \ldots, \mathcal{V}_N)$. Query embedding neural network take the UAV i state s_i as input, and outputs a query vector $\mathcal{Q}_i$. Second, agent computes attention weights $(\mathcal{W}_1, \ldots, \mathcal{W}_{i-1}, \mathcal{W}_{i+1}, \ldots, \mathcal{W}_N)$,

$$\mathcal{W}_j = \frac{\exp\left(\mathcal{Q}_i \mathcal{K}_j^T\right)}{\sqrt{d_k} \sum_{j \neq i}^{N} \exp\left(f\left(\mathcal{Q}_i, \mathcal{K}_j\right)\right)}, \tag{15}$$

where d_k is the dimension of keys. $\mathcal{W}_j$ are used to represent the importance of $\mathcal{V}_j$. Finally, the attention output vector $\mathcal{A}$ is computed as

$$\mathcal{A}(\mathcal{K}, \mathcal{Q}, \mathcal{V}) = \mathcal{W} * \mathcal{V}. \tag{16}$$

In addition, Transformer uses the multi-head attention to find different state representation spaces. Multi-head attention divides the $\mathcal{K}, \mathcal{Q}, \mathcal{V}$ vector into multiple sub-vectors of equal length to form multiple sub-spaces (or using more embedding neural networks to generate sub-vectors), allowing the model to pay attention to different aspects of information and generates head vectors $\mathcal{H}_h$,

$$\mathcal{H}_h = \mathcal{A}(\mathcal{K}_h, \mathcal{Q}_h, \mathcal{V}_h) \tag{17}$$

where h is the head index. Then concatenate these head vectors:

$$\mathcal{A}(\mathcal{K}, \mathcal{Q}, \mathcal{V}) = \text{Concat}\left(\mathcal{H}_1\left(\mathcal{K}, \mathcal{Q}, \mathcal{V}\right), \ldots, \mathcal{H}_H\left(\mathcal{K}, \mathcal{Q}, \mathcal{V}\right)\right), \tag{18}$$

where H is the number of heads.

It should be noted that in our scenario, in addition to the variable UAV neighbors, the UAV also needs to process the PoI neighbors that will change as the UAV itself moves. In other words, there are heterogeneous network nodes in our network. UAVs need to extract the state of PoIs to meet their communication needs and pay attention to the state of neighbor UAVs to maximize network utility. Fortunately, we find that the Transformer encoder has excellent scalability, and the neural network structure of this paper is shown in Fig. 6. We have designed Transformer encoder for each type of agent that match its state vector. Through the Transformer encoder, UAVs can effectively extract the information needed for decision-making and are suitable for more diversified network scenarios.

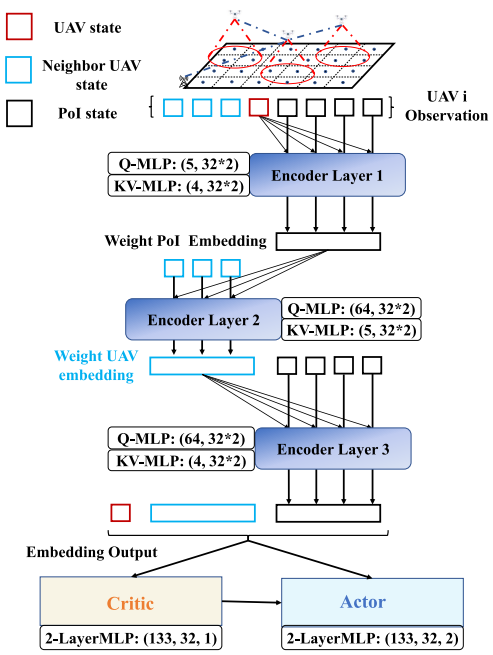

Fig. 6. In the proposed algorithm, each UAV network consists of tow part: the input Transformer encoder layer using attention module with 2 heads and 32 hidden units followed by the ReLU layer; the output critic and critic/actor layer using a MLP network with 32 hidden units. 5 is the UAV i state dimension $\left(e_{i,t}, p_{i,t}(x, y), \chi_{i,t}, t\right)$, and 4 is the PoI k state dimension $\left(p_{k,t}(x, y), c_{k,t}, n_{k,t}\right)$.

Unlike the Transformer [19] decoder that also uses the attention module, the decoder in this paper chooses the MLP layer. First, Vaswani et al. classic paper [19] on Transformer applied to the field of natural language processing (NLP). In NLP, attention module is often used to find the keywords in the previous text (the attention weight is larger) to assist the selection of the current word. The attention module is often used to distinguish the importance of different state inputs. However, the decoder in this paper does not need to use the attention module (Transformer decoder) to assign weights to the states of different sources, but only needs to process a hidden state vector embedded by the encoder. MLP can handle such hidden state vectors. Second, the attention module requires more parameters than the MLP. Attention not only includes the MLP network corresponding to the three K, Q, V, but also needs to perform dot product and normalization calculations on all K, Q, V vectors, which is contrary to the goal of reducing UAV energy consumption in this paper. In fact, in this paper we obtain acceptable results using a decoder module consisting of only one layer of MLP (32 hidden units).

B. Transformer-Based MARL

Next, we propose the T-MARL algorithm to solve the multi-UAV area coverage problem. T-MARL uses the Transformer architecture to fit action-state value by critic network θ and fit the policy by actor network ϕ. T-MARL expands among multi-agents based on parameter sharing methods and uses standard DRL for DNN training, such as deep deterministic policy gradient (DDPG) [48] and trust region policy optimization (TRPO) [26]. The agents policy function $\vec{\pi}_\theta\left(a|s\right)$ by the DNN directly, and the DNN parameter is

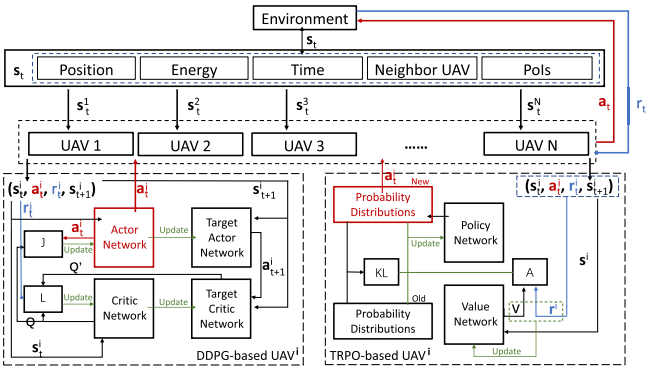

Fig. 7. Distributed control process and PG-based DRL model for each UAV.

θ. The policy gradient method updates the policy function parameters according to the loss function J_θ. J_θ can be defined according to different algorithms and application scenarios, such as $J_\theta = R_{\tilde{\pi}_\theta}(s_0)$. The updated goal of the policy function is to maximize the loss function, and its parameter update gradient is:

$$\theta_{t+1} = \theta_t + \sigma \nabla_\theta J_{\theta_t}, \tag{19}$$

where σ is a hyperparameter. In this paper, we discuss the application of two types of state-of-the-art PG methods: DDPG and TRPO in the multi-UAV area coverage problem. The algorithm is shown in Fig. 7. DDPG and TRPO are the PG methods based on the actor-critic structure. The actor network is used for action generation, and the critic network is used to evaluate the state with actor-selected action. We use centralized training and distributed execution to train UAVs, and each UAV deploys a DNN for autonomous decision-making.

DDPG can directly output deterministic actions, which enables it to deal with continuous space problems. The critic network θ_{dc} of DDPG directly outputs the action value function $Q_{\theta_{dc}}$, and its loss function is to minimize $J_{\theta_{dc}} = L_{\theta_{dc}}$ as

$$\begin{aligned} L_{\theta_{dc}} &= \mathbb{E}_{s,a_{t+1}}\left[\left(y_t - Q_{\theta_{dc}}(o_t, a_t)\right)^2\right], \\ y_t &= r_t(o_t, a_t) + \gamma Q'_{\theta'_{dc}}(o_{t+1}, a_{t+1}), \end{aligned} \tag{20}$$

where $Q'_{\theta'_{dc}}$ is the target critic network. The target critic network is used to assist in evaluating the action value function and has the same network structure as the critic network. a_t and a_{t+1} are selected by the actor network and the target actor network, respectively. DDPG actor network θ_{da} can be updated by:

$$\nabla_{\theta_{da}} J(\theta_{da}) = \mathbb{E}_{s,a}\left[\nabla_{\theta_{dc}} Q_{\theta_{dc}}(o_t, a_t)\right]. \tag{21}$$

DDPG updates the parameters of the target network through a soft method

$$\theta' \leftarrow \sigma\theta + (1-\sigma)\theta', \tag{22}$$

where σ is a hyper-parameter. In the experiments, we also implemented two popular variants of the DDPG algorithm, soft actor-critic (SAC) [49] and twin delayed DDPG (TD3) [50], but due to the space limitations, we will not repeat them here.

In TRPO, $\pi_\theta(a_i|o_i)$ is a probability distributions. For example, assuming that the policy itself satisfies the Gaussian distribution, the output of the neural network is the mean and variance that are satisfied the distribution, and the action is sampled from the distribution. TRPO iteratively solves the following constraint problems, and then obtains an optimized stochastic policy $\tilde{\pi}_{\tilde{\theta}}$:

$$\begin{aligned} &\max_{\tilde{\pi}_{\tilde{\theta}}} \quad \mathbb{E}_{s,a}\left[\frac{\tilde{\pi}_{\tilde{\theta}}(a|o)}{\pi_\theta(a|o)} A_\pi(o,a)\right], \\ &s.t. \quad \mathbb{E}_s\left[D_{KL}(\pi_\theta(\cdot|o)||\tilde{\pi}_{\tilde{\theta}}(\cdot|o))\right] \le \delta_{KL}, \end{aligned} \tag{23}$$

where D_{KL} is the Kullback-Leibler (KL) divergence between the two policy distributions, A is the advantage function

$$A_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r(s_t, a_t) + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)\right], \tag{24}$$

and δ_{KL} is a step size parameter that controls the maximum change in the policy per optimization step.

C. Baseline-Assisted Pre-Training

In standard DRL, agents need to generate training data through interaction with environment, and hope to find actions that can obtain higher rewards through exploration. Although RL allows to improve DNN models by exploring actions (occasional high reward samples), the premise of effective exploration is that the agent's action policy (actor) can provide stable and effective action selection. However, in long-term tasks like ACP, we find that randomly initialized actor may cause the DRL agent to be filled with a large number of invalid actions during sampling phase, resulting in training failure. The performance improvement brought about by occasional effective exploration actions will be offset by a large number of invalid actions (such as the random output of a randomly initialized action policy network), and no obvious system rewards can be obtained. For example, if the randomly initialized UAV linear velocity action output is always less than 0, this will make the UAV keeps hovering, which obviously cannot obtain good system rewards. We depict three examples of the UAV decision-making process in Fig. 8 to show the effect of randomly initializing the network on training.

In Fig. 8, the black circle represents the maximum flight distance of the UAV. The numbers on UAVs mark the action index. Since the initial policy is initialized by random parameters, we assume that the probability of UAV taking all actions in the feasible region is uniformly distributed. In time slot 1, the initial UAV coverage area is the red circle. It is easy to think that based on the initial position, the UAV flying to the upper left can get a better reward. Action 1 is a good choice for time slot 1. UAV covers the two new PoIs and gets more rewards. Action 2 causes the UAV to exceed the target area, and the UAV will receive the corresponding revenue penalty. Action 3 maintains the same reward, but is closer to the boundary and tends to exceed the target area. It should be noted that even if corresponding rules can be designed to prevent UAVs from flying out of the border, such flying trends are not allowed, and therefore still need to be punished. Therefore, actions 2 and 3 are poor action choices. A simple observation of Fig. 8 shows that the probability of a UAV

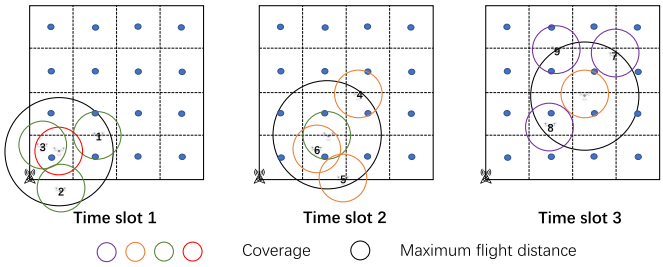

Fig. 8. The three kinds of the time slot of the UAV decision-making process with the randomly initialize actor.

taking a bad action is greater than the probability of taking a good action.

At the same time, we noticed two other interesting actions. In time slot 2, UAV is already in a relatively good position, and keeping hovering has similar rewards to action 4. In time slot 3, although action 7 moved to the upper left, it did not cover PoIs, which resulted in low rewards. In addition to the low probability of choosing a good action, the agent also needs to solve the problem: even if a good action is explored, it may not necessarily get a high instant reward. The existence of the above problems has led to extremely high requirements for the design of the reward function. In addition, in the MA-DRL scenario that also involves coordination among UAVs, which makes the training of UAVs policy more difficult. New methods are needed to solve the inherent problem of randomly initialized DNN in complex environments.

The mainstream model initialization methods can be roughly divided into two types: random initialization and pre-training. However, as mentioned earlier, the training data of RL agents is typically a function of the current policy. Randomly initialized DNN models do not support RL training. The pre-training method uses the data collected from similar tasks to perform supervised or semi-supervised learning to initialize DNN model, and then adjust the model parameters on the current task through fine-tune. The pre-training method is more common in natural language processing (NLP) tasks. But unfortunately, the UAV field does not yet have a recognized large pre-training model like the NLP field (such as BERT [51]). In fact, the lack of existing data and the high cost of exploration make the progress of DRL technology in the control field difficult, and there is still a lack of an accepted method for model initialization.

In this paper, we use a baseline-assisted pre-training scheme to initialize actors and critics of UAVs, as shown in Fig. 3. Baseline algorithms are a natural way to obtain data. Although baseline algorithms are often environment dependent and are only an acceptable solution, compared to randomly initialized actors, the baseline algorithm can generate labeled data and assist in actor initialization. In the actor initialization phase of the baseline-assisted pre-training scheme, we perform supervised training on actors based on the baseline generated labeled actions. It should be noted that baseline algorithm data may introduce a bias in the DNN model, which may make the UAV policy fall into a sub-optimum. But in this paper, such bias is acceptable. First, our pre-training solves the problem that DRL cannot start training caused by the randomly initialized DNN model, and the policy performance improvement is still carried out in the DRL stage. Second, in many applications, theoretically optimal solutions are difficult to obtain. As a trade-off, a good enough sub-optimal solution is acceptable in many applications, and researchers often hope to get a better solution. Third, Some strategies exist for both sub-optimal and theoretically optimal solutions. We only require a small amount of data to generate the initialization model and avoid over-fitting the baseline algorithm.

We apply the "early stopping" trick during pre-training. In traditional deep learning, "early stopping" is often used to prevent DNN from "over-fitting", so that the model maintains generalization ability on the test set. We want policies to learn common parts of optimal and sub-optimal policies. After that, we feed the pre-trained model into the subsequent DRL stage. It should be noted that our pre-training solves the problem that DRL cannot start training caused by the randomly initialized DNN model, and the policy performance improvement is still carried out in the DRL stage.

The critic determines the actor update gradient. temporal-difference(TD)(λ) is a classic algorithm in the field of reinforcement learning, which estimates the expected returns (7) through n-step bootstrapping [52]. Here n is the traditional notation used to represent the number of steps planned for the future. The formula for expected returns in TD(λ) is

$$R_t^{\lambda} = \lambda^{T-t-1} R_t^{(0)} + (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)}, \tag{25}$$

where ∞ represents the termination state, and $R_t^{(n)}$ is the state s_{t+n} expected returns. The critic is updated based on R_t^{λ} as label,

$$V(s_t) \leftarrow V(s_t) + \alpha \left(R_t^{\lambda} - V(s_t) \right), \tag{26}$$

where α is the TD learning rate hyper-parameter. There are two things to note in the (26). First, updating the value function $V(s_t)$ of each state requires a large number of samples to access $R_t^{(n)}$, which brings a huge computational load. Second, although both λ and γ formally discount the expected returns, λ is a normalization of the effect of the n-steps $R_t^{(n)}$ effect, while γ is a discount on future single-step rewards as part of the expected returns. In addition, in standard deep reinforcement learning, to make the training data satisfy the assumption of independent and identical distribution, the sampling data is often shuffled deliberately, and only 1-step transition information is retained, which is stored in the form of quadruples (s_t, a_t, r_t, s_{t+1}). Standard DRL uses TD(0) for critic training, and the formula is

$$R_t^{\lambda=0} = R_t^{(1)} = r_t + \lambda V(s_{t+1}). \tag{27}$$

Note that λ in TD(0) is just a constant multiplier, so λ is ignored in a lot of standard DRL literature, such as (7). $V(s_{t+1})$ is estimated by input s_{t+1} to target critic. Designing a target critic is a benchmark method to avoid the overestimation problem in DRL, and interested readers can further read the related literature of standard DRL [50]. To generate an accurate critic network, the standard DRL method assumes that the number of task steps is infinite and satisfies the MDP [52]. This assumption can usually be satisfied in game scenarios.

However, there are some specific requirements in network scenarios. First, data is expensive in networks. And in the network, data is often saved in the form of a stream (or called a trajectory), which retains the context information and can perform n-step return access. Compared with the resource consumption of computational load, making full use of expert data and reducing sampling to avoid network risks are key factors in the networks. Second, due to the hyperparameter $\gamma = 0.99$, the state action representing the current time step will only be affected by the effective length of $1/(1-\gamma) = 100$, which also means that at the end of our task. The last $1/(1-\gamma) = 100$ timeslots do not meet the assumption of infinite length and influences overall critic update gradients. In practical applications, the problem manifests itself as an explosion of critic output values based on TD(0). Therefore, we use TD(1) for critic evaluation, and the formula for TD(1) expected returns is

$$R_t^{\lambda=1} = R_t^{(0)} = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t} r_T, \qquad (28)$$

where $R_t^{\lambda=1}$ and (7) are the same. Although TD(1) loses the advantages of low variance in TD(0)-based standard DRL, constructing a usable critic is the primary problem for a critic.

V. Simulation

A. Simulation Settings

We design a square area of 10×10 units as the target area, and the center of each unit is PoI. The side length of each unit is 100 meters. In the target area, we have designed 6 UAVs to provide communication services for PoIs by default. The communication coverage range of UAVs is 250 meters by default. The default communication range among UAVs is 500 metes. The UAVs start the task from the access point area, and the total time of the coverage task is 256 time slots. In each time slot, the maximum flight speed of UAVs is $25m/s$, and the most energy-efficient flight speed is $10m/s$. If a UAV runs out of power or loses connection while it is moving, the task is considered to have failed. The network structure of T-MARL in the experiment as shown in in Fig. 6. We set the following measures to evaluate the performance of the trained model:

- **Average Coverage Score (ACS)**: The coverage performance of all PoIs during the entire test phase can be expressed as (2).
- **Fairness Index (F)**: The fair performance of all PoIs covered throughout the test phase is consistent with (3).

We compare T-MARL with the following baseline algorithms to perform ablation experiments to verify the effectiveness of the Transformer and pre-training scheme:

- **A-Star**: The A-Star algorithm is indeed recognized as one of the most effective direct search methods for finding the shortest path in a static road network. It is a heuristic algorithm. The A-Star algorithm needs to search for nodes in the map and set a suitable heuristic function (we use reward function in this paper) for guidance. By evaluating the cost of each node, the next best node that needs to be expanded is obtained until the final target point is reached.
- **Expert**: Drawing inspiration from the greedy concept inherent in the A-star algorithm, we have designed two expert baseline solutions for multi-UAV ACP. **(i) ACS-First**: In each time slot, UAVs fly straight to the nearest uncovered PoIs within the observable range, and each UAV flies at cruising speed to avoid violating energy constraints, as shown in Fig. 9(a). **(ii) F-First**: This method is based on a specific topology for artificial UAV trajectory design, so that UAVs can cover as many PoIs as possible. In the regular PoIs topology, we keep the UAV at an energy-efficient cruise speed, and each UAV covers a sector of $\pi/24$, as shown in Fig. 9(d).
- **Pre-training**: We use 10 episodes (8 ACS-First and 2 F-First trajectories) of the expert algorithm to generate data to initialize the model by a baseline-assisted pre-training scheme.
- **DRL-Trans**: In the result analysis, to facilitate the representation of the legend, we mark the T-MARL scheme based on the DDPG algorithm as DDPG-Trans and the scheme that does not use the Transformer architecture as DDPG-MLP. In addition, we use DDPG-Trans* to mark the model without pre-training. Similarly, we use TRPO-Trans, SAC-Trans, and TD3-Trans to denote experiment results based on other general RL algorithms (TRPO, SAC, and TD3). For convenience, we use DRL-Trans to collectively refer to these algorithms. In addition, these DRL-Trans essentially replicate some basic reinforcement learning algorithms at the algorithm level, for example, DDPG-MLP (without pre-training) is equivalent to work in [6], and TRPO-MLP (without pre-training) is equivalent to works in [9] and [12]. We have chosen the four most mainstream benchmark DRL algorithms. However, since this paper focuses on DNN model initialization, this part is decoupled from the model gradient update process that DRL algorithms focus on. Therefore, other DRL algorithms can also be conveniently replaced within the T-MARL framework.

B. Multi-UAV Control Performance

1) UAVs Flight Trajectory: We train models with 3 to 8 UAVs, coverage ranges from 1.5 to 2.75 (distance units are 1 kilometer), and different numbers of PoIs from 25 to 100. The rest of the parameter settings are default values in Section V-A. For each parameter group, we used five different random seeds for training. In the experimental results, Figs. 10, 11, and 14 are the best model for demonstration and Figs. 12 and 13 are the average performance of the five seed models to represent the performance of the algorithm. In practical work, we found that after supervised learning with a large amount of data initialization, the effect of random seeds is small, and the model initialization performance is close to the baseline performance.

Fig. 9 shows a demonstration obtained by using T-MARL based DDPG and baseline algorithm with UAV number 6 and PoI number 100. The experimental results show that DDPG-Trans has the best network performance. DDPG-Trans has an ACS improvement of 39.7%, and a fairness index improvement of 45.3% relative to the expert coverage-first algorithm. In Fig. 9(a), we show the reason why the effect of the expert coverage-first algorithm is not as good as the

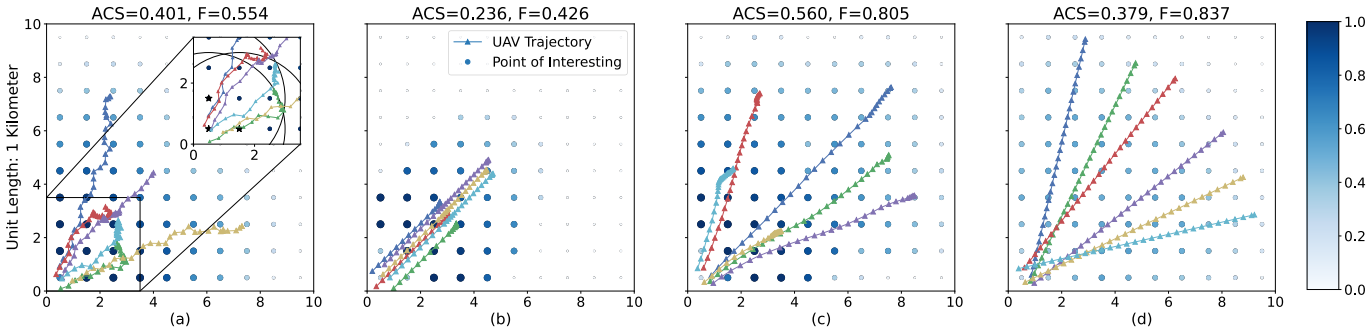

Fig. 9. The flight trajectory of UAVs based on (a) ACS-first, (b) Pre-training, (c) DDPG-Trans, and (d) F-First solution with the number of UAVs 6 and PoIs 100. The blue color bar indicates the coverage time of each PoI after normalization. The black arc in the upper right sub-figure of (a) is a circle whose star mark indicates that the PoI position is the center and the coverage range is the radius.

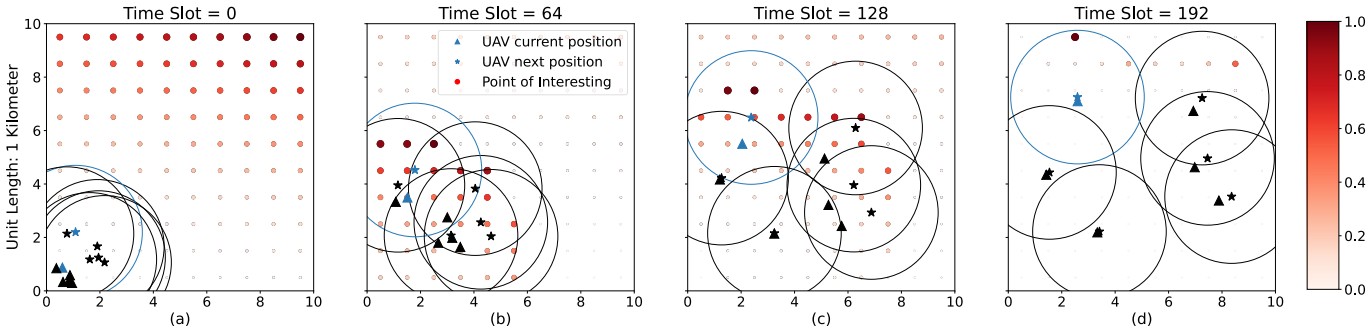

Fig. 10. This figure shows the attention trajectories of the DDPG-Trans UAV (indicated in sky blue) over different time periods. The triangle represents the position in (a) $t = 0$, (b) $t = 64$, (c) $t = 128$, (d) $t = 192$. The star mark represent position in (a) $t = 32$, (b) $t = 96$, (c) $t = 160$, (d) $t = 224$. The color bars represent the normalized attention weights of the sky blue UAV for each PoI. The circle indicates the coverage of star mark UAVs. The black UAVs is a neighbor of the sky blue UAV.

design expectations. In the enlarged area at the top right of the figure 9(a), the black arc is a circle whose star mark indicates that the PoI position is the center and the coverage range is the radius. The UAV covered in the black circle provides coverage services for the center PoIs. It is easy to find that the three UAV trajectories represented by sky blue, green, and red oscillate on the circumference, instead of one UAV covering three star mark PoIs at the same time. Through follow-up verification, we found that this is caused by the initial position of the UAV, and the problem is common. This shows the sensitivity of the artificial design algorithm to the initial state, lack of complex state transfer ability, and scalability.

DNN can generalize to unseen complex input states that training data are in the same state space. We did not show the trajectory of DDPG-Trans*, because the UAV cannot collect the available data under this method, resulting in an extreme output of the actor, such as static in the initial position or rushing out of the target area at the maximum speed, causing the mission to fail. Thus, we collect 8 episodes from the coverage-first algorithm and 2 episodes from the fairness-first algorithm to train the initialization pre-training DNN model. The UAVs flight trajectory based on pre-training DNN is shown in Fig. 9(b), and pre-training DNN has only learned a very rough strategy, moving slowly to the upper right. However, such an initial policy is sufficient to support subsequent DRL training, and agents cannot collect available samples. The UAV flight trajectory based on DDPG-Trans shows obvious cooperative behavior. As shown in Fig. 9(c), UAVs accelerate in the early stage of the mission to increase the fairness index, and then UAVs cooperated to cover more PoIs.

2) UAV Attention Trajectory: The red color bar in Fig. 10 is used to represent the change of the normalized attention weight of the red UAV in Fig. 9(c) at different time slot t. In Fig. 10(a), UAVs are in the initial position, with sufficient power, and their attention is focused on the upper right corner. Through Fig. 9(c), we can see that UAVs are in the acceleration phase to the upper right to increase the fairness index. From Figs. 10(b), 10(c), and 10(d), we can find that the PoIs with higher UAV attention weight at time slot t have been covered by UAVs at time slot $t + 32$. The experimental phenomenon shows that attention has played a guiding role in UAVs.

3) Network Performance and Ablation Experiments: We have verified the effectiveness of the T-MARL algorithm under various conditions through experiments, and the numerical results are shown in Figs. 11 and 12. In Fig. 11, we show the average coverage score performance under different conditions. Both T-MARL based on different standard reinforcement learning algorithms (TRPO, DDPG, SAC, and TD3) outperform the baseline algorithm ACS-first. This shows that T-MARL can effectively leverage existing standard reinforcement learning results and extend them to network scenarios. Furthermore, we note that the experimental results of DRL-Trans* without the pre-training scheme, and the experimental results of DRL-MLP without Transformer are not satisfactory. This ablation experiment demonstrates the

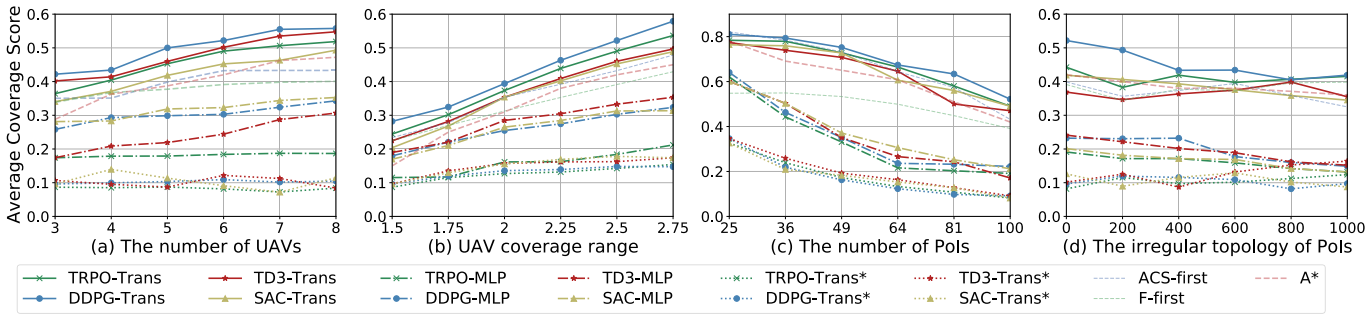

Fig. 11. The impact of (a) the number of UAVs on average coverage score, (b) the communication coverage range of UAVs, (c) the number of PoIs, and (d) the irregular PoIs topology on average coverage score.

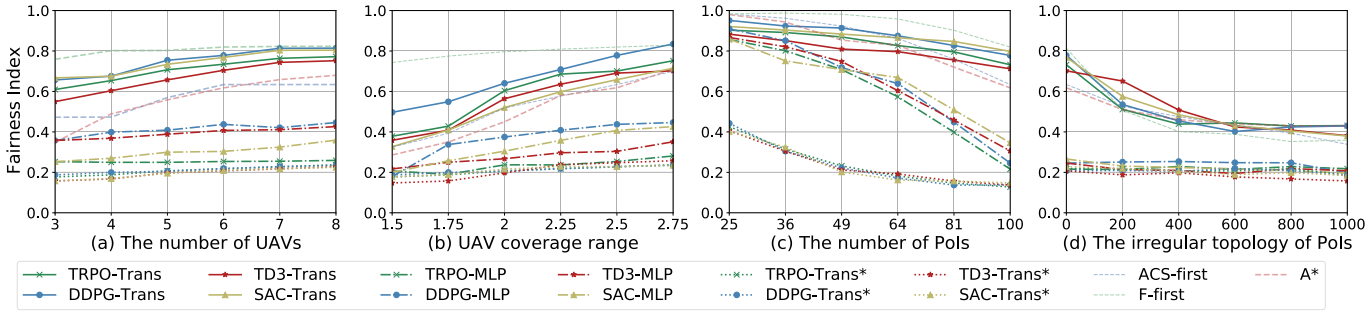

Fig. 12. The impact of (a) the number of UAVs, (b) the communication coverage range of UAVs, (c) the number of PoIs, and (d) the irregular PoIs topology on fairness index.

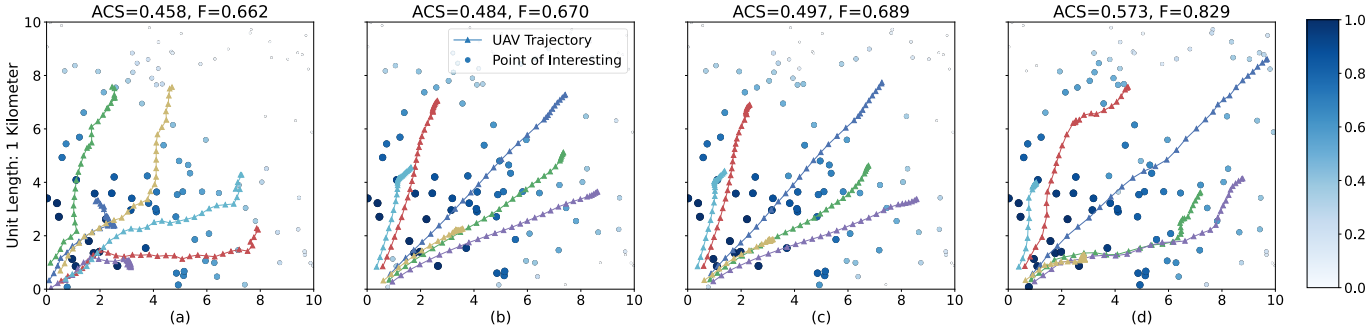

Fig. 13. The flight trajectory of UAVs based on (a) Expert solution, (b) execute initial DDPG-Trans directly, (c) DDPG-Trans after 1 fairness-first episode pre-training, and (d) DDPG-Trans.

necessity of the Transformer and baseline-assisted pre-training scheme. In Fig. 12, DRL-trans also show acceptable performance on fairness, although they fail to surpass the well-designed baseline algorithm F-first. This is because UAVs should fly farther to serve more PoIs, making the average coverage score lower to improve fairness, such as a small number of PoIs in edge areas. Conversely, when considering average coverage score, UAVs will stay in areas with dense PoIs, which leaves PoIs in sparse areas uncovered, making fairness lower. In fact, when DRL-Trans optimizes two objectives at the same time, it needs to trade-off between average coverage score and fairness. The experimental results are shown in Figs. 9(a) and 9(d).

4) From Regular to Irregular: Fig. 13 shows a demonstrate in a random initialization irregular PoIs distribution. Figure 13(b) is a trajectory diagram of directly executing the model learned by DDPG-Trans on the irregular topology. Figure 13(c) is a model based on a Fairness-first episode pre-training after the DDPG-Trans model is used as a seed. Figure 13(d) is the trajectory of the DDPG-Trans model after DRL training on the topology. Compared with the Fig. 13(a) generated based on coverage-first, directly applying the DDPG-Trans model can get slightly better network performance. Using the existing regular DDPG-Trans as the initialization model, after training with the T-MARL algorithm under this irregular topology, the network performance of the model has an ACS improvement of 25.1%, an improvement of 25.2% relative to the expert coverage-first algorithm. We further verified the transfer ability of the model learned by the T-MARL algorithm in irregular topology, and the numerical results are shown in Fig. 11(d) and Fig. 12(d) with different number of PoIs.

5) The Evolution of Policy: Taking the experimental results in this paper as an example, we end the pre-training phase

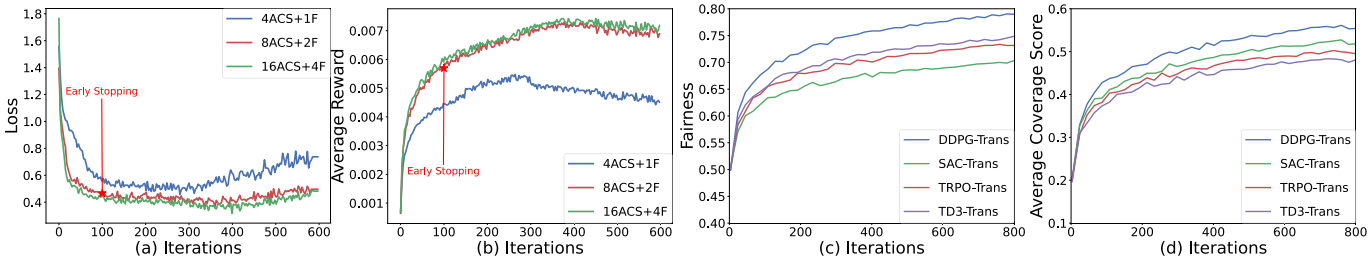

Fig. 14. The evolution of model performance over the number of supervised training iterations: (a) training loss in pre-training, (b) average reward in pre-training. Observing the pre-training experiment results, we select the model indicated by the red star mark to DRL training phase. (c) average coverage score in DRL training, and (d) fairness in DRL training.

very early, as shown in Figs. 14(a) and 14(b). In each iteration, 16 time slots of data are randomly selected from replay buffer for training (batch size = 16). It can be clearly found that UAVs only learn simple flight strategies in the "early stopping" model, as shown in Fig. 9(b), and due to the sharing of models between UAVs, the behavior trajectories between UAVs are similar, and no high-order cooperative behavior appears. We also show the evolution of system performance over the number of DRL training iterations in Fig. 14(c) and 14(d). In DRL training phase, every 24 iterations, the UAVs interacts with the environment to generate a new set of trajectories (256 time slots). The experimental results show that DRL can further improver the network performance and converge.

Subsequently, we will provide a theoretical validation that the T-MARL is capable of converging to a Nash equilibrium. The Nash equilibrium is a crucial concept that assesses the optimality of a single agent's policy value function under the collective policy of all agents. The verification of this claim is based on the subsequent assumptions:

Assumption 1: Each action-value pair is encountered an infinite number of times, and the reward is limited by a constant K.

Assumption 2: Agent's policy is Greedy in the Limit with Infinite Exploration (GLIE).

Assumption 3: For each game $[Q_{\boldsymbol{\pi}}^1(s), \cdots, Q_{\boldsymbol{\pi}}^N(s)]$ at time t and in state s in training, for all t, s, $i \in \{1, \cdots, N\}$, the Nash equilibrium $\boldsymbol{\pi}_* = [\pi_*^1, \cdots, \pi_*^N]$ is recognized either as (29) the global optimum or (30) a saddle point expressed as:

$$\mathbb{E}_{\pi_*}[Q_t^i(s)] \geq \mathbb{E}_{\pi}[Q_t^i(s)], \forall \boldsymbol{\pi} \in \Omega(\prod_k \mathcal{A}^k), \tag{29}$$

$$\begin{aligned} &\mathbb{E}_{\pi_*}[Q_t^i(s)] \geq \mathbb{E}_{\pi^i}\mathbb{E}_{\pi_*^{-j}}[Q_t^i(s)], \forall \pi^i \in \Omega(\prod_k \mathcal{A}^k), \\ &\mathbb{E}_{\pi_*}[Q_t^i(s)] \leq \mathbb{E}_{\pi_*^i}\mathbb{E}_{\pi^{-j}}[Q_t^i(s)], \forall \boldsymbol{\pi^{-i}} \in \Omega(\prod_{k\neq i} \mathcal{A}^k), \end{aligned} \tag{30}$$

where π^{-j} is a joint policy for all agents except agent j.

The proof is also built upon the two lemmas as follows:

Lemma 1: Under Assumption 3, define the Nash operator

$$\mathscr{H}^{Nash}\, \boldsymbol{Q}(s, \boldsymbol{a}) = \mathbb{E}_{s' \sim p}\left[\boldsymbol{r}(s, \boldsymbol{a}) + \gamma \boldsymbol{v}^{Nash}\left(s'\right)\right], \tag{31}$$

where $\boldsymbol{Q} \triangleq \left[Q^1, \cdots, Q^N\right]$ and $\boldsymbol{r} \triangleq \left[r^1, \cdots, r^N\right]$. $\mathscr{H}^{Nash}$ forms a contraction mapping on the complete metric space from $\mathcal{Q}$ to $\mathcal{Q}$ with the fixed point being the Nash Q-value of the entire game, such as $\mathscr{H}^{Nash}\, \boldsymbol{Q}_* = \boldsymbol{Q}_*$.

Proof: See Theorem 17 in [53].

In simple scenarios (such as tabular value functions), Nash RL [53] outlines a procedure for generating Nash equilibrium policies via an alternating two-step iteration process: (1) The Lemke-Howson algorithm is employed to determine the Nash equilibrium policy of the current game stage. (2) The estimation of the value function under the new Nash equilibrium is enhanced. This mechanism facilitates the dynamic modification and optimization of strategies within the game setting. The action-value function will eventually stabilize at the value derived from a game's Nash equilibrium, referred to as the Nash Q-value. Similar to Nash RL, T-MARL also iteratively updates the agent's policy (actor) and value function evaluation (critic), but it opts for an update method based on DNN gradients. The policy iteration and enhancement of an agent can be facilitated through RL predicated on value functions or Q-functions. Throughout the learning phase, the RL agent cyclically updates the value function.

$$Q_{t+1}(s, a) = (1 - \alpha) Q_t(s, a) + \alpha \left[r + \gamma v_t(s')\right], \tag{32}$$

where α is learning rate. With the advancement of deep learning technologies, the use of DNN to approximate value function and policy functions in intricate problems has surfaced as a new paradigm. Consequently, a variety of DRL variants have been introduced, such as DQN [54], DDPG [48], SAC [49], TD3 [50], and PPO [55] (TRPO [26]).

Lemma 2: The random process $\{\Delta_t\}$ define in $\mathbb{R}$ as

$$\Delta_{t+1}(x) = (1 - \alpha_t(x))\Delta_t(x) + \alpha_t(x) F_t(x) \tag{33}$$

converges to zero with probability 1 when

1. $0 \leq \alpha_t(x) \leq 1$, $\sum_t \alpha_t(x) = \infty$, $\sum_t \alpha_t^2(x) < \infty$;

2. $x \in \mathscr{X}$, the set of possible states, and $|\mathscr{X}| < \infty$;

3. $\|\mathbb{E}[F_t(x)|\mathscr{F}_t]\|_W \leq \gamma \|\Delta_t\|_W + c_t$, where $\gamma \in [0, 1)$ and c_t converges to zero w.p.1;

4. $\boldsymbol{var}[F_t(x)|\mathscr{F}_t] \leq K(1 + \|\Delta_t\|_W^2)$ with constant $K < 0$

Here $\mathscr{F}_t$ denotes the filtration of an increasing sequence of σ-fields including the history of processes; α_t, Δ_t, $F_t \in \mathscr{F}_t$ and $\|\cdot\|_W$ is a weighted maximum norm.

Proof: See Theorem 1 in [56] and Corollary 5 [57] for detailed derivation.

The convergence theorem for T-MARL is articulated as follows:

Theorem 1: If the first and second conditions of Lemma 2 and Assumptions 1-3 are met, the Q-values calculated using the update rule (31) will converge to the Nash Q-value $\boldsymbol{Q}_*$.

Proof: Assumption 1 is self-evident given the design of a substantial number of training samples and reward functions. When Assumption 1 is met, if Assumption 2 is also established, it should additionally satisfy:

$$\lim_{k\to\infty} \pi_k(a|s) = 1$$
$$a = \underset{a'\in\mathcal{A}}{argmax}\ Q_k(s,a'). \tag{34}$$

In RL, the policy is perpetually updated towards the direction of the maximum Q-value (e.g., DQN [54], DDPG [48], TD3 [50]) or selects a direction that will not deteriorate the policy (e.g., TRPO [26], PPO [55]), causing the algorithms to cease updating automatically. At this juncture, a certain policy approach that satisfies Assumption 2 will exist.

In accordance with the common Q-value update rule of reinforcement learning in (32) and equation (33) in Lemma 2, we formulate the difference between the current policy Q-value and the Nash Q-value Q_{π_*} *as follows*:

$$\begin{aligned}\boldsymbol{\Delta}_t(x) &= \boldsymbol{Q}_\pi(s,\boldsymbol{a}) - \boldsymbol{Q}_{\pi_*}(s,\boldsymbol{a}),\\ \boldsymbol{F}_t(x) &= \boldsymbol{r}_t + \gamma \boldsymbol{v}_t^{\mathrm{DRL}}(s_{t+1}) - \boldsymbol{Q}_{\pi_*}(s_t,\boldsymbol{a}_t),\end{aligned} \tag{35}$$

where $x \triangleq (s_t, a_t)$ *denotes the visited state-action pair at time* t. *In (33)* $\alpha(t)$ *represents the learning rate, where* $\alpha_t(s',a') = 0$ *for all* $(s',a') \neq (s_t,a_t)$. *This is because each agent only updates its Q-function for the state* s_t *and actions* a_t *that visited at time* t. *Lemma 2 indicates that* $\Delta_t(x)$ *will converge to zero, implying that if it holds, the sequence of Q will asymptotically approach the Nash* Q_*.

Let $\mathscr{F}_t$ *represent the* σ*-field generated by the cumulative random variables of the stochastic game until time* t: $(s_t, \alpha_t, \boldsymbol{a}_t, r_{t-1}, \cdots, s_1, \alpha_1, \boldsymbol{a}_1, \boldsymbol{Q}_0)$. *Notice that* $\boldsymbol{Q}_t$ *is a random variables originating from the historical trajectory until time* t. *Considering that all* $\boldsymbol{Q}_\tau$ *with* $\tau < t$ *are* $\mathscr{F}_t$*-measurable, it follows that both* $\boldsymbol{\Delta}_t$ *and* $\boldsymbol{F}_{t-1}$ *are also* $\mathscr{F}_t$*-measurable, which met the measurability condition of Lemma 2.*

To apply Lemma 2, we need to demonstrate that the DRL operator $\mathscr{H}^{DRL}$ *satisfies the third and fourth conditions of Lemma 2. For third condition of Lemma 2, we begin with (35) that*

$$\begin{aligned}\boldsymbol{F}_t(s_t,\boldsymbol{a}_t) &= \boldsymbol{r}_t + \gamma \boldsymbol{v}_t^{DRL}(s_{t+1}) - \boldsymbol{Q}_*(s_t,\boldsymbol{a}_t)\\ &= \boldsymbol{r}_t + \gamma \boldsymbol{v}_t^{Nash}(s_{t+1}) - \boldsymbol{Q}_*(s_t,\boldsymbol{a}_t)\\ &\quad + \gamma\left[\boldsymbol{v}_t^{DRL}(s_{t+1}) - \boldsymbol{v}_t^{Nash}(s_{t+1})\right]\\ &= \left[\boldsymbol{r}_t + \gamma \boldsymbol{v}_t^{Nash}(s_{t+1}) - \boldsymbol{Q}_*(s_t,\boldsymbol{a}_t)\right] + C_t(s_t,\boldsymbol{a}_t)\\ &= \boldsymbol{F}_t^{Nash}(s_t,\boldsymbol{a}_t) + \boldsymbol{C}_t(s_t,\boldsymbol{a}_t).\end{aligned} \tag{36}$$

Notice that $\boldsymbol{F}_t^{Nash}$ *in (36) fundamentally corresponds to* $\boldsymbol{F}_t$ *in Lemma 2, which is crucial for establishing the convergence of the Nash reinforcement learning algorithm. From Lemma 1, it is evident that* $\boldsymbol{F}_t^{Nash}$ *constitutes a contraction mapping with the norm* $||\cdot||_\infty$ *representing the maximum norm on* $\boldsymbol{a}$. *Therefore, we have the following equation for all* t *that*

$$\left\|\mathbb{E}\left[\boldsymbol{F}_t^{Nash}(s_t,\boldsymbol{a}_t)\,|\mathscr{F}_t\right]\right\|_\infty \leq \gamma \left\|\boldsymbol{Q}_t - \boldsymbol{Q}_*\right\|_\infty = \gamma \left\|\boldsymbol{\Delta}_t\right\|_\infty \tag{37}$$

To satisfy the third condition of Lemma 2, we derive from (36) the subsequent equation

$$\begin{aligned}\|\mathbb{E}[\boldsymbol{F}_t|\mathscr{F}_t]\|_\infty &\leq \left\|\boldsymbol{F}_t^{Nash}|\mathscr{F}_t\right\|_\infty + \|\boldsymbol{C}_t|\mathscr{F}_t\|_\infty\\ &\leq \gamma\|\boldsymbol{\Delta}_t\|_\infty + \|\boldsymbol{C}_t(s_t,\boldsymbol{a}_t)|\mathscr{F}_t\|_\infty.\end{aligned} \tag{38}$$

We now need to demonstrate that $c_t = \|\boldsymbol{C}_t(s_t,\boldsymbol{a}_t|\mathscr{F}_t)\|$ *converges to zero with probability* 1. *According to Assumption 3, for each game stage, all the saddle point equilibrium(s) share the same Nash value, so does the globally optimal equilibrium(s). Lemma 1 indicates that the policy based on the action-value function constitutes a contraction mapping. With homogeneous agents (parameter sharing in T-MARL) and all optima/saddle points share the same Nash value,* $\boldsymbol{v}_t^{DRL}$ *will asymptotically converge to* $\boldsymbol{v}_t^{Nash}$ *and satisfy the third condition of Lemma 2.*

Regarding the fourth condition of Lemma 2, we use the conclusion that the aforementioned $\boldsymbol{v}_t^{DRL}$ *will asymptotically converge to* $\boldsymbol{v}_t^{Nash}$ *and* $\mathscr{H}^{DRL}$ *thus also forms a contraction mapping, such as* $\mathscr{H}^{DRL}\boldsymbol{Q}_* = \boldsymbol{Q}_*$, *which leads to*

$$\begin{aligned}\mathrm{var}[\boldsymbol{F}_t(s_t,\boldsymbol{a}_t)\,|\mathscr{F}_t] &= \mathbb{E}\left[\left(\boldsymbol{r}_t + \gamma \boldsymbol{v}_t^{\mathrm{DRL}}(s_{t+1}) - \boldsymbol{Q}_*(s_t,\boldsymbol{a}_t)\right)^2\right]\\ &= \mathbb{E}\left[\left(\boldsymbol{r}_t + \gamma \boldsymbol{v}_t^{\mathrm{DRL}}(s_{t+1}) - \mathscr{H}^{\mathrm{DRL}}(\boldsymbol{Q}_*)\right)^2\right]\\ &= \mathrm{var}\left[\boldsymbol{r}_t + \gamma \boldsymbol{v}_t^{\mathrm{DRL}}(s_{t+1})\,|\mathscr{F}_t\right]\\ &\leq K\left(1 + \|\boldsymbol{\Delta}_t\|_W^2\right)\end{aligned} \tag{39}$$

In the final step of (39), we apply Assumption 1, which states that the reward $\boldsymbol{r}_t$ *is consistently bounded by a certain constant. Finally, with all conditions met, it follows Lemma 2 that* $\boldsymbol{\Delta}_t$ *converges to zero with probability* 1 *such as* $\boldsymbol{Q}_t$ *converges to* $\boldsymbol{Q}_*$.

VI. Conclusion

In this paper, we consider the long-term decision-making problem of multi-UAV networks. In a multi-UAV area coverage problem, UAVs need to interact with complex networks and handle a variable number of heterogeneous network nodes to optimize the average coverage score and fairness index at the same time. We propose a Transformer-based deep multi-agent reinforcement learning algorithm to solve the UAV control problems. Transformer-based deep multi-agent reinforcement learning introduces Transformer and parameter sharing to standard reinforcement learning to solve the scalability problem of networks. We also discover the problem of random initialization of deep neural network and add the baseline-assisted pre-training scheme to solve this dilemma. Experimental results show that our algorithm has an ACS improvement of 39.7%, and a fairness index improvement of 45.3% relative to the baseline algorithm in the literature.

REFERENCES

[1] M. Asim, M. El Affendi, and A. A. A. El-Latif, "Multi-IRS and multi-UAV-assisted MEC system for 5G/6G networks: Efficient joint trajectory optimization and passive beamforming framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4553–4564, Apr. 2023.

[2] R. Fu, Q. Quan, M. Li, and K.-Y. Cai, "Practical distributed control for cooperative multicopters in structured free flight concepts," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4203–4216, Apr. 2023.

[3] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 740–759, Feb. 2022.

[4] L. Ni, B. Sun, X. Tan, and D. H. K. Tsang, "Mobility and energy management in electric vehicle based mobility-on-demand systems: Models and solutions," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 3702–3713, Apr. 2023.

[5] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and A. Nallanathan, "Deep reinforcement learning based dynamic trajectory control for UAV-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3536–3550, Oct. 2022.

[6] 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.

[7] N. Zhao et al., "UAV-assisted emergency networks in disasters," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 45–51, Feb. 2019.

[8] Y. Zhang, Z. Zhuang, F. Gao, J. Wang, and Z. Han, "Multi-agent deep reinforcement learning for secure UAV communications," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2020, pp. 1–5.

[9] C. H. Liu, X. Ma, X. Gao, and J. Tang, "Distributed energy-efficient multi-UAV navigation for long-term communication coverage by deep reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1274–1285, Jun. 2020.

[10] Y. Gu, Y. Jiao, X. Xu, and Q. Yu, "Request–response and censoring-based energy-efficient decentralized change-point detection with IoT applications," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6771–6788, Apr. 2021.

[11] J. Chen, F. Ye, and Y. Li, "Travelling salesman problem for UAV path planning with two parallel optimization algorithms," in *Proc. Prog. Electromagn. Res. Symp. (PIERS-FALL)*, Singapore, 2017, pp. 832–837.

[12] D. Chen, Q. Qi, Z. Zhuang, J. Wang, J. Liao, and Z. Han, "Mean field deep reinforcement learning for fair and efficient UAV control," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 813–828, Jan. 2021.

[13] 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.

[14] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 6380–6391.

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

[16] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. Auto. Agents Multiagent Syst.*, Stockholm, Sweden, Jul. 2018, pp. 2085–2087.

[17] E. Wei, D. Wicke, D. Freelan, and S. Luke, "Multiagent soft Q-learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, Sep. 2018, pp. 1–7.

[18] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, Sao Paulo, Brazil, May 2017, pp. 66–83.

[19] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, Dec. 2017, pp. 6000–6010.

[20] Z. Ye, K. Wang, Y. Chen, X. Jiang, and G. Song, "Multi-UAV navigation for partially observable communication coverage by graph reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4056–4069, Jan. 2022.

[21] Z. Mou, F. Gao, J. Liu, and Q. Wu, "Resilient UAV swarm communications with graph convolutional neural network," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 393–411, Jan. 2022.

[22] S. Rahmani, A. Baghbani, N. Bouguila, and Z. Patterson, "Graph neural networks for intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 8, pp. 8846–8885, Mar. 2023.

[23] H. Qi, Z. Hu, H. Huang, X. Wen, and Z. Lu, "Energy efficient 3-D UAV control for persistent communication service and fairness: A deep reinforcement learning approach," *IEEE Access*, vol. 8, pp. 53172–53184, 2020.

[24] H. V. Abeywickrama, Y. He, E. Dutkiewicz, B. A. Jayawickrama, and M. Mueck, "A reinforcement learning approach for fair user coverage using UAV mounted base stations under energy constraints," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 67–81, 2020.

[25] I. A. Nemer, T. R. Sheltami, S. Belhaiza, and A. S. Mahmoud, "Energy-efficient UAV movement control for fair communication coverage: A deep reinforcement learning approach," *Sensors*, vol. 22, no. 5, p. 1919, Mar. 2022.

[26] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. 32nd Int. Conf. Mach. Learn.*, Lille, France, Jul. 2015, pp. 1889–1897.

[27] M. Huttenrauch, A. Šošić, and G. Neumann, "Deep reinforcement learning for swarm systems," *J. Mach. Learn. Res.*, vol. 20, no. 54, pp. 1–31, Feb. 2019.

[28] K. Han et al., "A survey on vision transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 87–110, Jan. 2023.

[29] C. Lili et al., "Decision transformer: Reinforcement learning via sequence modeling," in *Proc. 35th Conf. Neural Inf. Process. Syst.*, Dec. 2021, pp. 15084–15097.

[30] M. Xu et al., "Prompting decision transformer for few-shot policy generalization," in *Proc. 39th Int. Conf. Mach. Learn.*, Baltimore, MD, USA, Jul. 2022, pp. 24631–24645.

[31] S. Hu, F. Zhu, X. Chang, and X. Liang, "UPDet: Universal multi-agent reinforcement learning via policy decoupling with transformers," in *Proc. 9th Int. Conf. Learn. Represent.*, May 2021, pp. 1–15.

[32] S. N. Ferdous, X. Li, and S. Lyu, "Uncertainty aware multitask pyramid vision transformer for UAV-based object re-identification," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Bordeaux, France, Oct. 2022, pp. 2381–2385.

[33] L. Bashmal, Y. Bazi, and N. Alajlan, "Space time attention transformer for non-event detection in UAV videos," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Kuala Lumpur, Malaysia, Jul. 2022, pp. 1920–1923.

[34] W. Lu et al., "A CNN-transformer hybrid model based on CSWin transformer for UAV image object detection," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 16, pp. 1211–1231, 2023.

[35] S. Kumar, A. Kumar, H.-S. Hong, and D.-G. Lee, "Encoder-decoder based segmentation model for UAV street scene images," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Las Vegas, NV, USA, Jan. 2023, pp. 1–4.

[36] T. Ye et al., "Real-time object detection network in UAV-vision based on CNN and transformer," *IEEE Trans. Instrum. Meas.*, vol. 72, 2023, Art. no. 2505713.

[37] B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton, and Z. Gao, "UAV trajectory planning for AoI-minimal data collection in UAV-aided IoT networks by transformer," *IEEE Trans. Wireless Commun.*, vol. 22, no. 2, pp. 1343–1358, Feb. 2023.

[38] Y. Wang, M. Yan, G. Feng, and S. Qin, "Autonomous learning based proactive deployment for UAV assisted wireless networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Rio de Janeiro, Brazil, Dec. 2022, pp. 1320–1325.

[39] R. Bellman, "A Markovian decision process," *J. Math. Mech.*, vol. 6, no. 5, pp. 679–684, Jan. 1957.

[40] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, "Distributed multi-agent meta learning for trajectory design in wireless drone networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 3177–3192, Oct. 2021.

[41] Z. Mou, Y. Zhang, F. Gao, H. Wang, T. Zhang, and Z. Han, "Deep reinforcement learning based three-dimensional area coverage with UAV swarm," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 3160–3176, Oct. 2021.

[42] C. Zhao, J. Liu, M. Sheng, W. Teng, Y. Zheng, and J. Li, "Multi-UAV trajectory planning for energy-efficient content coverage: A decentralized learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 3193–3207, Oct. 2021.

[43] T. Rashid et al., "QMix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 2018, pp. 7234–7284.

[44] *Study on Enhanced LTE Support for Aerial Vehicles (Release 15)*, Sophia Antipolis, France, document 3GPP Rep. TR 36.777, Dec. 2017.

[45] K. Liu and J. Zheng, "UAV trajectory optimization for time-constrained data collection in UAV-enabled environmental monitoring systems," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24300–24314, Dec. 2022.
[46] Q. Zhang, H. Wang, Z. Feng, and Z. Han, "Many-to-many matching-theory-based dynamic bandwidth allocation for UAVs," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9995–10009, Jun. 2021.
[47] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2329–2345, Apr. 2019.
[48] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, San Juan, Puerto Rico, May 2016, pp. 1–14.
[49] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 2018, pp. 1861–1870.
[50] F. Scott, H. Herke, and M. David, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 2018, pp. 1587–1596.
[51] D. Jacob, C. Ming, L. Kenton, and T. Kristina, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, Minneapolis, MN, USA, Sep. 2019, pp. 4171–4186.
[52] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, Nov. 2018.
[53] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Nov. 2003.
[54] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[55] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," Jul. 2017, *arXiv:1707.06347*.
[56] T. Jaakkola, M. I. Jordan, and S. P. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, Denver, CO, USA, Dec. 1994, pp. 703–710.
[57] C. Szepesvári and M. L. Littman, "A unified analysis of value-function-based reinforcement-learning algorithms," *Neural Comput.*, vol. 11, no. 8, pp. 2017–2060, Nov. 1999.

Dezhi Chen is currently pursuing the Ph.D. degree with the Beijing University of Posts and Telecommunications, Beijing, China. His research interests include UAV control, game theory, and next-generation mobile communication networks, using reinforcement learning and artificial intelligence technologies.

Qi Qi (Senior Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2010. She is currently an Associate Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She has authored or coauthored more than 30 articles in the international journals. Her research interests include edge computing, cloud computing, the Internet of Things, ubiquitous services, deep learning, and deep reinforcement learning. She was a recipient of two National Natural Science Foundations of China.

Qianlong Fu is currently pursuing the master's degree with the Beijing University of Posts and Telecommunications, Beijing, China. His research interests include network routing and management for next-generation network infrastructures, using machine learning and artificial intelligence techniques.

Jingyu Wang (Senior Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2008. He is currently a Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. He has published more than 100 articles in such as the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, CVPR, ACL, MM, ICDE, and AAAI. His research interests include broad aspects of intelligent networks, edge/cloud computing, machine learning, AIOps and self-driving networks, the IoV/IoT, knowledge-defined networks, and intent-driven networking. He is a Senior Member of CIC and selected for Beijing Young Talents Program.

Jianxin Liao received the Ph.D. degree from the University of Electronics Science and Technology of China, Chengdu, China, in 1996. He is currently the Dean of the Network Intelligence Research Center and a Full Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. He has authored or coauthored 100's of research articles and several books. His main research interests include cloud computing, mobile intelligent networks, service network intelligence, networking architectures and protocols, and multimedia communication. He has won several prizes in China for his research achievements, which include the Premiers Award of Distinguished Young Scientists from National Natural Science Foundation of China in 2005 and the specially invited Professor of the Yangtse River Scholar Award Program by the Ministry of Education in 2009.

Zhu Han (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University in 1997 and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively.

From 2000 to 2002, he was a Research and Development Engineer with JDSU, Germantown, Maryland. From 2003 to 2006, he was a Research Associate with the University of Maryland. From 2006 to 2008, he was an Assistant Professor with Boise State University, ID, USA. Currently, he is a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department and the Computer Science Department, University of Houston, TX, USA. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grids. He received the NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the *EURASIP Journal on Advances in Signal Processing* in 2015, IEEE Leonard G. Abraham Prize in the field of Communications Systems (Best Paper Award in IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS) in 2016, and several best paper awards in IEEE conferences. He was an IEEE Communications Society Distinguished Lecturer from 2015 to 2018, an AAAS fellow since 2019, and an ACM Distinguished Member since 2019. He is a 1% Highly Cited Researcher since 2017 according to Web of Science.