Unmanned Aerial Vehicle Swarm Cooperative Decision-Making for SEAD Mission: A Hierarchical Multiagent Reinforcement Learning Approach

LONGFEI YUE\textsuperscript{1}, RENNONG YANG\textsuperscript{1}, JIALIANG ZUO\textsuperscript{1}, YING ZHANG\textsuperscript{1}, QIUNI LI\textsuperscript{2}, AND YIJIE ZHANG\textsuperscript{3}

\textsuperscript{1}Air Traffic Control and Navigation College, Air Force Engineering University, Xi’an 710051, China
\textsuperscript{2}Aeronautics Engineering College, Air Force Engineering University, Xi’an 710038, China
\textsuperscript{3}Xi’an Modern Control Technology Research Institute, Xi’an 710065, China

Corresponding author: Jialiang Zuo (correszjl@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 62106284; in part by the Natural Science Foundation of Shaanxi Province, China, under Grant 2021JQ-370; and in part by the Young Talent Fund of Association for Science and Technology in Shaanxi, China, under Grant 20220101.

\section*{ABSTRACT} Unmanned aerial vehicle (UAV) swarm cooperative decision-making has attracted increasing attentions because of its low-cost, reusable, and distributed characteristics. However, existing non-learning-based methods rely on small-scale, known scenarios, and cannot solve complex multi-agent cooperation problem in large-scale, uncertain scenarios. This paper proposes a hierarchical multi-agent reinforcement learning (HMARL) method to solve the heterogeneous UAV swarm cooperative decision-making problem for the typical suppression of enemy air defense (SEAD) mission, which is decoupled into two sub-problems, i.e., the higher-level target allocation (TA) sub-problem and the lower-level cooperative attacking (CA) sub-problem. A HMARL agent model, consisting of a multi-agent deep Q network (MADQN) based TA agent and multiple independent asynchronous proximal policy optimization (IAPPO) based CA agents, is established. MADQN-TA agent can dynamically adjust the TA schemes according to the relative position. To encourage exploration and promote learning efficiency, the Metropolis criterion and inter-agent information exchange techniques are introduced. IAPPO-CA agent adopts independent learning paradigm, which can easily scale with the number of agents. Comparative simulation results validate the effectiveness, robustness, and scalability of the proposed method.

\section*{INDEX TERMS} UAV swarm, suppression of enemy air defense, deep reinforcement learning, multi-agent, hierarchical reinforcement learning.

\section*{I. INTRODUCTION} Suppression of enemy air defense (SEAD) mission aims to suppress the enemy’s integrated air defense system (IADS), in which the active aerial electronic jamming or anti-radiation missiles are used for suppression, and fighters are used for attack by launching laser-guided weapons [1], [2], [3], [4]. SEAD is a typical offensive counter air (OCA) mission, an important prerequisite of a large-scale air attack. However, the powerful and high-threat IADS poses increasing challenges and risks to performing SEAD mission using manned aircrafts [5]. Over recent years, unmanned aerial vehicle (UAV) swarm techniques have seen great developments and have been widely used in jamming suppression, distributed strikes, and cooperative decision-making missions [6]. Thus, there exists a great potential in using autonomous and heterogeneous UAV swarm (i.e., to carry various payloads) cooperative decision-making to perform the complicated and high-threat SEAD mission.

Researchers on UAV swarm cooperative decision-making have paid their great efforts in rule-based methods [7], [8], influence diagram methods [9], [10], computational intelligence methods [11], search-based intelligent optimization
algorithms [3], [12], and learning-based reinforcement learning (RL) algorithms [13], [14], [15], [16], [17], [18].

Rule-based methods rely on expert knowledge to predefine decision-making rule databases [7], [8]. Influence diagram methods and computational intelligence methods require experts to establish a probabilistic reasoning network or design a specific heuristic objective function [9], [10], [11]. Although significant progress has been made by these aforementioned approaches, they all rely heavily on prior human experience.

The search-based intelligent optimization algorithms can find the optimal or suboptimal solution of the explicit objective function through the parallel iteration optimization mechanism. Zhang et al. [12] studied the UAV path planning problem in electronic warfare. Multi-objective particle swarm optimization algorithm was used to solve the optimal path and jamming array mode of the jammer. To solve the task assignment problem of UAV identification, attack, and evaluation, Darrach et al. [3] used a genetic algorithm to solve the optimal task assignment result. The search-based methods cannot generalize to dynamic uncertain environment, and the computational complexity increases exponentially with the growth in problem scale. Therefore, the search-based methods have limitations in the large-scale and uncertain combat problems where real-time performance is needed.

The learning-based RL algorithms are a data-driven optimization approach for decision-making problems, which solves the optimal policy by maximizing the sum of the rewards. Guo et al. [13] utilized HRL to train UAVs to flexibly and autonomously avoid obstacles in high dynamic environments. Hu et al. [14] proposed DRL with relevant experience replay to address UAV autonomous motion planning in unknown environments. Li et al. [15] and Pope et al. [16] researched the 1V1 air combat problem and found that DRL agents are more flexible and powerful than traditional rule-based agents. Sun et al. [17] trained air combat agents with tactics emergence capabilities using HRL and self-play techniques. Zhang et al. [18] studied the 3V3 air combat problem using MARL algorithm and employed a scenario transfer training strategy for air combat in a complex swarm scenario, improving learning efficiency. The learning-based method adopts offline training and online decision-making framework, so it has better real-time performance. Neural network can automatically extract important features from complex high-dimensional combat scenario, and RL ensures the optimality of decision-making. Therefore, the learning-based RL method has great potential in UAV swarm cooperative decision-making problems with dynamic uncertainties.

Recently deep reinforcement learning (DRL) and multi-agent reinforcement learning (MARL) [19], [20] have achieved great success in games and robotics, which show a bright prospect for multi-agent sequential decision-making problem. UAV swarm cooperative decision-making for SEAD mission is also a multi-agent sequential decision-making problem, involving multiple heterogeneous (attacking and jamming) UAVs, referred to as fighters and jammers, which cooperate to make dynamic and coupled decisions on target allocation (TA), route planning, jamming, and attacks. Thus, UAV swarm cooperative decision-making for air-to-surface [21] SEAD mission is naturally suited to be solved using MARL. Moreover, hierarchical reinforcement learning (HRL) gives agents hierarchical thinking and decision-making capabilities similar to humans, and can solve large-scale agents and sparse reward problems. Therefore, DRL shows bright prospect for the problem of large-scale UAV swarm cooperative decision-making in uncertain scenarios. This is the primary motivation of the present study.

In this paper, we build a hierarchical multi-agent reinforcement learning (HMARL) framework for UAV swarm cooperative decision-making for SEAD mission that involves multiple tasks, e.g., coordinated flight, jamming, and strike tasks. The hierarchical framework includes the higher-level target allocation (TA) agent and the lower-level the cooperative attacking (CA) agents, in which the CA agents are trained to attack the selected target, and the TA agent are optimized together with the CA agents to determine the optimal solution of target allocations. The lower-level CA agents are trained with an independent learning (IL) paradigm, and each agent utilizes an independent asynchronous proximal policy optimization (IAPPO) to control a fighter and jammer to conduct the jamming-attack kill chain, which can be distributed deployed online and flexibly scaled to large-scale scenarios. The higher-level TA agent employs the learning-based multi-agent deep Q network (MADQN) algorithm, which can consider the relative distance between the UAV and the target. Our method’s effectiveness, robustness, and scalability are tested and compared in simulation experiments. Figure 1 shows the offline training with online inference research framework in this paper.

The major contributions of this paper are:

1. A HMARL framework for UAV swarm cooperative decision-making is proposed. The higher-level MADQN-TA agent determines the optimal target allocation results, and the lower-level IAPPO-CA agent attacks the selected targets.

2. Metropolis criterion and inter-agent information exchange techniques are employed to improve the sample efficiency and learning speed. Metropolis criterion increases the action exploration probability and prevents the policy from falling into local minima. The inter-agent information exchange technique reduces the selection of invalid actions in multi-agent system.

3. A light UAV swarm cooperative decision-making environment is built for developing the DRL-based UAV swarm air-to-surface combat tactics for SEAD mission.

The remainder of this paper is organized as follows. Section II describes the SEAD problem and scenario. Section III summarizes several standard RL algorithms used in this paper. In Section IV, a cooperative decision-making model based on HMARL method is established for a UAV swarm in SEAD mission. Several experiments are carried out.
to verify the method’s effectiveness, robustness, and scalability in Section V. Section VI concludes the whole paper.

II. PROBLEM DESCRIPTION
Using a single aircraft to fight against an IADS system is vulnerable in the high-threat SEAD mission. Therefore, an effective tactic is to use a jammer to suppress the enemy’s IADS radar and a fighter to attack in the blind areas. This paper focuses on the cooperative tactics of heterogeneous UAV swarm.

Multiple IADS batteries exist in the real-world scenario, as shown in Figure 2, and thus cooperative decision-making is required. The combat procedure is divided into two phases: 1) first, to allocate targets, which determines the attacking sequence and selected target list; 2) second, to conduct a cooperative attack, which determines the route, jamming, and firing decisions. In particular, the jammer must approach the IADS and reduce the IADS’s range. The fighter can destroy the IADS and live when the radar range of IADS is reduced [22].

Therefore, we decouple the UAV swarm cooperative decision-making problem for SEAD mission into two hierarchical sub-problems, i.e., the higher-level TA sub-problem and the lower-level CA sub-problem. The solution of the whole problem is obtained by solving the two sub-problems first.

III. PRELIMINARIES
RL is a machine learning approach that agents can explore environments by trial and error to learn optimal policies. RL is usually modeled as a finite MDP, defined by the tuple \((S, A, R, P, \gamma)\), where \(S\) is a set of states, \(A\) is a set of actions, \(R\) is a reward function, \(P\) is a transition probability function, and \(\gamma\) is a discount factor. Assuming that at time step \(t\) the agent take action \(a \in A\) at the state \(s \in S\) according to policy \(\pi : S \rightarrow A\), the environment will feedback an instant reward.
where \( r \in R \) to the agent, and the agent will transition to a new state \( s' \in S \) until terminal. Tabular reinforcement learning evaluates the state-action value through a discrete Q table, but for continuous and high-dimensional problems, it encounters the “curse of dimensionality,” which spurs the development of DRL [23].

### A. DRL

DRL refers to the combination of RL with DL. DRL leverages a neural network to approximate policy or value function to solve the continuous mapping problem. The agent’s goal is to find the policy \( \pi_\theta \) with the maximum expected return \( J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] \). \( \theta \) is the weight parameters of policy network. \( R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \) refers to the discounted cumulative rewards on the trajectory \( \tau = (s_0, a_0, s_1, a_1, \ldots) \). The optimal policy is

\[
\pi_\theta^* = \arg \max_{\pi_\theta} \mathbb{E}_{\tau \sim \pi_\theta}(\sum_{t=0}^{\infty} \gamma^t r_t)
\]  

(1)

A DRL algorithm is divided into three learning paradigms: value-based; policy gradient; and actor-critic [24].

### B. DEEP Q NETWORK

Deep Q network (DQN) is the first off-policy value-based DRL algorithm [25], and can be used to solve the high dimensional decision-making problem. A neural network approximates the value of discrete state-action pair in Q-learning to evaluate the value of state-action pairs, which addresses the continuous state and discrete action decision-making problem.

Therefore, we refer to a neural network function approximator as a Q-network. A Q-network can be trained by minimizing a loss function [22]:

\[
J(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)}[y_i - Q(s, a; \theta_i)]^2
\]  

(2)

where \( y_i = \mathbb{E}_{s' \sim \gamma}[r + \gamma \max_{a'} Q'(s', a'; \theta_i^-)|s, a] \) is a target for iteration \( i \) and \( \rho(s, a) \) is a probability distribution over sequences \( s \) and actions \( a \).

### C. ASYNCHRONOUS PROXIMAL POLICY OPTIMIZATION

The actor-critic RL algorithm integrates the value function and policy gradient methods and uses the value function to guide the policy update to accelerate the learning speed. The policy is updated through the gradient of expected return, which can be written as

\[
\nabla_{\theta} J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}(\sum_{t=0}^{T} \nabla_{\theta} \log \pi_\theta(a|s)R(\tau))
\]  

(3)

where \( \pi_\theta(a|s) \) is an actor and \( R(\tau) \) a critic, and \( R(\tau) \) can also take other forms, such as a state-action value function \( Q'(s, a) \), advantage function \( A'(s, a) \), or temporal difference (TD) residual error \( G_t = r_t + \gamma V'(s_{t+1}) - V'(s_t) \).

Proximal policy optimization (PPO) [26] aims to take the most significant improvement step on a policy without stepping so far away from the old policy. PPO proposes a surrogate loss function to control the policy update step size and uses importance sampling to improve sample efficiency. The algorithm balances between sample efficiency, algorithm performance, and engineering implementation complexity. The optimization objective of PPO, i.e., the surrogate loss function, is simplified to

\[
J_{\text{CLIP}}(\pi_\theta) = \mathbb{E}_{(s,a) \sim \pi}(\min(\rho(\theta)A^\pi(s, a), \ \text{clip}(\rho(\theta), 1 - \epsilon, 1 + \epsilon)A^\pi(s, a)))
\]  

(4)

where \( \rho(\theta) = \frac{\pi(a|s)}{\pi^*(a|s)} \) is the ratio of new and old policies, and \( \epsilon \) is a hyper-parameter. The truncation operation, i.e., CLIP operate, limits the policy update amplitude to ensure the training stability. The generalized advantage estimation (GAE) [27] is used to calculate the advantage \( A^\pi(s, a) \) and keep the variance and deviation estimated by the value function small, as shown in (5).

\[
A_t^{\text{GAE}}(\gamma, \lambda) = \sum_{i=0}^{\infty} (\gamma \lambda)^i (r_{t+i} + \gamma V(s_{t+i+1}) - V(s_{t+i}))
\]  

(5)

Asynchronous proximal policy optimization (APPO) is an asynchronous variant of PPO algorithm [28]. APPO uses a surrogate policy loss with clip operate. Compared to a synchronous PPO, APPO is usually more efficient due to its asynchronous sampling. Figure 3 shows the architecture of APPO.

---

In each episode of updates, the algorithm runs \( L \) actors in parallel, and each actor runs \( T \) steps, for a total of \( LT \) items of step data, and calculates advantages \( A_1, \ldots, A_T \). The policy parameters are updated after sampling is completed, where the cumulative rewards of the loss function are \( J(\pi_\theta) \). We randomly sample \( M \) in each episode of updates, where \( M \leq LT \), and learn \( K \) times to improve sample efficiency. The asynchronous training mode is used for efficient training, whereby sampling and learning do not need to wait for each other.

### D. HRL

HRL is also an important technique that can be used to solve complex decision-making problems. The idea of “divide and conquer” decomposes the original problem into several sub-problems. The simple sub-problems are solved one by one and then are integrated to get the solution of the original problem. Temporal abstraction of state sequence is used to treat the problem as a semi-Markov decision process (SMDP) [29]. Basically, the idea is to define macro-actions, composed of primitive actions, which allow for modeling the agent at
different levels of abstraction. This approach is known as HRL [30], which is similar to the hierarchical structure of human learning [31], and has generated essential advancements in RL such as option-learning [32], option critic [33], FeUdal networks [34], data-efficient HRL known as HIRO [35], etc.

The main advantages of using HRL are scalability and generation ability. Scalability decomposes the large problems into smaller ones, avoiding the curse of dimensionality. Generalization ability is acquired due to the combination of smaller sub-tasks that allows for generating new skills and avoiding super-specialization [16].

E. MARL

MARL studies how multiple agents interact with the environment to achieve human-like collaboration in cooperative tasks. MARL has three learning frameworks, including centralized learning (CL), independent learning (IL), and centralized learning with decentralized execution (CLDE) [36]. CL framework treats the multi-agent system as one and uses a single-agent RL algorithm to address the non-stationarity of the environment. The input state space grows linearly with the number of agents, which will be computationally expensive, and requires global communication from a “God’s eye view.” Therefore, it cannot be used in non-communication and large-scale scenarios. IL framework allows each agent to train its policy separately, which brings scalability. IQL [37] and IPPO [38] have achieved good results, but the relationship between multi-agents is ignored, leading to instability in the learning process. CLDE framework can improve learning efficiency with a “God’s eye view” during training and make a decision independently during execution [39]. The joint Q-value function in CLDE converges slowly when the number of agents increases.

We consider the scalability of large-scale UAV swarm cooperative decision-making and the particularity of UAV swarm that allocates targets first and then strikes independently, without experience and information sharing between two phases. Therefore, a hierarchical IL framework is chosen to solve the multi-agent cooperation problem.

IV. METHOD

In this section, we build an end-to-end HMARL method for UAV swarm cooperative decision-making in SEAD mission. First, a UAV kinematic model is developed. Second, SEAD mission is divided into two hierarchical sub-problems, i.e., target allocation sub-problem and cooperative attacking sub-problem, which are modeled as a cooperative attacking MDP model and a target allocation MDP model, respectively. Metropolis criterion and inter-agent information exchange techniques are introduced to improve the sample efficiency and learning speed. Third, we design the HMARL agent network structure and describe the proposed learning framework in this paper. Finally, six policy training tricks are described for ablation study.

A. KINEMATICS OF UAV

In practical SEAD mission, UAV cannot be operated at high angle of attack or high angular rates due to weapon and sensor limitations. Therefore, we assume that the altitude of the UAV remains constant, and a four-degree-of-freedom (4-DOF) kinematics model is used

\[
\begin{align*}
\dot{x}_f &= v_f \cos \varphi_f \\
\dot{y}_f &= v_f \sin \varphi_f \\
\dot{x}_j &= v_j \cos \varphi_j \\
\dot{y}_j &= v_j \sin \varphi_j
\end{align*}
\]  

(6)

where \( \dot{x}_f, \dot{y}_f, v_f, \) and \( \varphi_f \) represent the differentiation of coordinate \( X \) and \( Y \), speed, and heading of the fighter, respectively; \( \dot{x}_j, \dot{y}_j, v_j, \) and \( \varphi_j \) represent the differentiation of coordinates \( X \) and \( Y \), speed, and heading of the jammer, respectively. In addition, state input and action output constraints are modeled as lower and upper bounds:

\[
\begin{align*}
0 \leq x &\leq x_{max} \quad v_{min} \leq v \leq v_{max} \\
0 \leq y &\leq y_{max} \quad \varphi_{min} \leq \varphi \leq \varphi_{max}
\end{align*}
\]  

(7)

Therefore, in this MDP, state is defined as the coordinate vector and action is the velocity and heading of the fighter and jammer.

B. MDP

1) COOPERATIVE ATTACKING MDP

To solve the scalability problem of large-scale combat scenario, an independent learning framework is adopted to establish the same cooperative attacking MDP model for each formation. Detail designs of the MDP model are listed as follows.

a: STATE SPACE

The state space is defined as the coordinates vector \((x_f, y_f), (x_j, y_j)\), and \((x_s, y_s)\) of the fighter, jammer, and IADS, which are continuous values. In the actual battlefield environment, the IADS coordinate can be transmitted to the UAV by airborne warning and control system (AWACS) or space-based satellites.

b: ACTION SPACE

The action space is defined as the fighter’s and jammer’s heading \( \varphi_f, \varphi_j \) and speed \( v_f, v_j \). By controlling the heading to change the direction of movement of a UAV in a 2-D environment, the UAV can coordinate in time to reach the desired position by controlling the speed. The jammer adopts a simple jamming model for analysis simplicity. By setting the jamming distance condition, when the IADS enters the jamming range, we assume that the jammer will automatically turn on jamming, and IADS radar detection range is degraded.

c: REWARD FUNCTION

In this paper, a non-sparse reward is designed to guide UAV accomplish SEAD mission. Four types of reward are
TABLE 1. Reward function.

| Description | Reward Function |
|-------------|-----------------|
| Success reward | \( r_1 = 1 \) |
| Death or collision or out of the boundary | \( r_2 = -1 \) |
| Distance reward | \( r_3 = D_{t-1} - D_t \) |
| Step penalty | \( r_4 = -0.1 \) |

considered in the reward function. If the jammer can suppress the IADS without entering its attack range, and at the same time the fighter can destroy the IADS without entering its range, it will be rewarded with a success reward \( r_1 = 1 \). If the jammer or fighter enters the IADS range or flies out of the environment boundary, it will get a reward \( r_2 = -1 \). If the jammer and fighter collide, it will also get a reward \( r_2 = -1 \). In other circumstances, reward shaping is adopted and a continuous distance reward \( r_3 = D_{t-1} - D_t \) is given to guide the UAV to approach the IADS, where \( D_{t-1} \) and \( D_t \) denote the distance between UAV and IADS at the last moment \( t-1 \) and the current moment \( t \). We set a step penalty reward \( r_4 = -0.1 \) to encourage the agent to complete the task as soon as possible. The settings of reward function are listed as Table 1.

The final reward \( R_{CA} \) is defined as the sum of four types of reward:

\[
R_{CA} = r_1 + r_2 + r_3 + r_4 \tag{8}
\]

Cooperative attacking MDP model is implemented by IAPPO algorithm as shown in Algorithm 1. Each formation adopts the same IAPPO algorithm.

Algorithm 1 Independent Asynchronous Proximal Policy Optimization - Cooperative Attacking (IAPPO-CA).

- Initialize actor network parameter, critic network parameter, and replay buffer \( D \).
- For episode \( = 1 : M \) do
  - For \( n = 1 : N \) do
    - Run policy \( \pi_{\theta'} \) for \( T \) timesteps asynchronously, collecting experience \((s, a, r, s')\).
    - Estimate return \( R(\tau) \) and advantage \( A^GAE(\gamma, \lambda) \).
  - For \( k = 1 : K \) do
    - Sample minibatch from the trajectory, calculate policy loss, value loss and entropy loss.
    - Optimize surrogate loss function \( J(\theta) \).
  - Update policy parameter \( \theta' \leftarrow \theta \).
- End for
- End for
- End for

2) TARGET ALLOCATION MDP
It is challenging to consider the dynamic changing factors for the traditional target allocation method, which establishes a static optimization model based on intelligent optimization algorithms. In this paper, we adopt a learning-based method to establish a target allocation agent model. Through dynamic interaction with environment and trial-and-error, agent can find an optimal or sub-optimal target allocation scheme and quickly allocate targets online with a low amount of computation.

Target allocation is a continuous state and discrete action decision-making problem. As the number of targets increases, the number of target allocation schemes increases exponentially, and the action space dimension of the agent will explode. The MARL is used to prevent the “curse of dimensionality” problem. Therefore, the target allocation problem is modeled as a cooperative MARL model. Each formation can be seen as a DQN agent, an off-policy framework and sample efficient. There is inter communications between multi-agents, so it is a MADQN algorithm.

Therefore, target allocation MDP is modeled as follows:

\( a: STATE SPACE \)
Target allocation considers the position states of the fighter, jammer, and IADS, which are continuous values \((x_f, y_f), (x_j, y_j), \) and \((x_i, y_i)\), respectively. If a IADS is chose, the IADS coordinate will be the input vector of DQN agent. Therefore, the state space is defined as the coordinate concatenated vector of each formation of fighter, jammer, and IADS in a 2-D environment, as the Table 1 shown.

\( b: ACTION SPACE \)
Each DQN agent outputs \( N \) dimensions and selects a target IADS ID to attack. So the action space dimension of the DQN agent is the same as the number of IADS, as shown in Table 2.

\( c: REWARD FUNCTION \)
We apply the fully cooperative MADQN algorithm that all agents jointly maximize the global team reward. The reward of the MADQN-TA agent depends on the effect of the final cooperative attacking. Therefore, the reward of the MADQN-TA agent is the sum of the rewards obtained by all lower-level IAPPO-CA agents,

\[
R_TA = \sum_{i=1}^{N} R_{CA}^{i} \tag{9}
\]

where \( R_{CA}^{i} \) is the reward obtained by each converged IAPPO-CA agent. To maximize the reward \( R_{CA}^{i} \), the MADQN-TA agent has to allocate the nearest target to each corresponding IAPPO-CA agent because of the step penalty.

TABLE 2. Action-space description of DQN agent.

| State space | Value |
|-------------|-------|
| Dimension Description | \( N \) | Number of IADS |

TABLE 1.

| Reward Function |
|-----------------|
| Success reward | \( r_1 = 1 \) |
| Death or collision or out of the boundary | \( r_2 = -1 \) |
| Distance reward | \( r_3 = D_{t-1} - D_t \) |
| Step penalty | \( r_4 = -0.1 \) |
d: METROPOLIS CRITERION

Moreover, action is chosen according to the Metropolis criterion as (10) to encourage exploration and prevent the policy from falling into a local minimum. After adopting the $\epsilon$-greedy policy ($\epsilon$ decays with increasing iterations) to select the action, we use the Metropolis criterion of simulated annealing algorithm to promote the agent’s exploration efficiency. The action exploration equation is defined as:

$$ a_t = \begin{cases} 
\text{a}_\text{ε-greedy}, & r_t \geq \max(r) \\
\text{a}_\text{ε-greedy}, & r_t < \max(r), \ \eta \geq \eta_0 \\
\text{a}_\text{random}, & r_t < \max(r), \ \eta < \eta_0
\end{cases} \quad (10) $$

Suppose the reward for a DQN agent’s action is less than the history maximum reward. In that case, an action is randomly selected with a certain probability to increase the probability of jumping out of a local minimum, and otherwise, the $\epsilon$-greedy policy is used to select the action.

e: INTER-AGENT INFORMATION EXCHANGE

The targets are allocated in turn according to the position state of both sides. DQN agent chooses the IADS target in turn. The inter-agent information exchange is adopted to ensure no repeated strike and promote learning efficiency, as depicted in Figure 4. That is, agent 2 knows the action of agent 1. If agent 2 choose an action the same as agent 1, it will change again. That means the IADS ID already assigned by the previous agent is no longer allocated by the next agent. Multi-agent system is trained in parallel at the same time, which addresses the problem of action space explosion [40].

![Figure 4. Inter-agent information exchange.](image)

Target allocation MDP model is implemented by MADQN algorithm. Algorithm 2 shows the details of MADQN-TA algorithm within one episode:

C. HMARL AGENT NETWORK DESIGN

The design of an agent network must consider the coupling relationship between target allocation and cooperative attacking, and their precedence relationship. Hence, we propose a hierarchical structure. The higher level is $N$ target allocation agents, which is implemented by the MADQN algorithm. The MADQN-TA agents considers the situations of both parties, outputs the optimal or suboptimal target list. The lower level is $N$ cooperative attacking agents, which is implemented by the IAPPO algorithm. The IAPPO-CA agent completes the attack process according to the assigned IADS ID, and thereby the mission is completed. The higher and lower-level agents are cascaded to obtain a whole HMARL agent. The network structure of the end-to-end HMARL agent is shown in Figure 5.

The input of DQN consists of the coordinates of each group of the fighter, jammer, and IADS, then normalizes by Z-Score and input to the two-layer fully connected neural network, output the Q value of each allocation target ID. The ID with the max Q value is the optimal target and its coordinate inputs to the lower-level IAPPO agent, respectively. The lower-level IAPPO agent includes critic network and actor/policy network. The IAPPO policy network inputs the coordinates of the fighter, jammer, and IADS ID allocated by the higher-level MADQN-TA agent and then inputs to the two-layer fully connected neural network, outputs the heading and speed of the fighter and jammer.

Algorithm 2 Multi-Agent Deep Q Network – Target Allocation (MADQN-TA).

1. Initialize replay buffer $D$ to capacity $K$ for each formation.
2. Initialize action-value function $Q$ with random weights $\theta$.
3. Initialize target action-value function $Q'$ with weights $\theta^- = \theta$.
4. For episode $e = 1 : M$ do
   1. Initialize state $s_0$ for each agent.
   2. For $t = 1 : T$ do
      1. Observe state $s^n_t$ and select action $a^n_t$ for agent $n$ using $\epsilon$-greedy.
      2. Execute action $a^n_t$ based converged IAPPO and observe next state $s^n_{t+1}$, reward $r^n_t$, and done signal $d^n_t$ to indicate whether $s^n_t$ is terminal.
      3. Action exploration according to (10) and do state transfer.
      4. Store the transition $(s^n_t, a^n_t, r^n_t, s^n_{t+1}, d^n_t)$ in $D$.
   3. Round robin the order of agents in inter-agent information exchange.
5. End for
6. Calculate the team reward $r_t$ according to (9).
7. Randomly sample a batch of transitions from $D$.
8. Calculate target $Q$ value: $y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-)$ for each formation.
9. Update $Q$ function by gradient descent step on $\theta$ to minimize $[y_t - Q(s_t, a_t; \theta))^2$ for each formation.
10. Every $C$ steps reset $Q' = Q$ for each formation.
11. End for
12. End for
In summary, the higher-level and lower-level agents are designed to be “offline training” first, then “online inference.” The target ID is first assigned according to the relative distance between target and formation, and the IAPPO-CA agent is trained. After the IAPPO-CA agent has converged, the higher-level MADQN-TA agent is concatenated to optimize jointly. Finally, the two converged agent models are cascaded to obtain the complete HMARL model, which is directly tested online to verify its intelligent decision-making performance, as depicted in Figure 1.

D. POLICY TRAINING TRICKS
To improve the training efficiency of the policy, six policy training tricks are studied and compared in the experiments.

1) DATA NORMALIZATION
   a: ADVANTAGE FUNCTION NORMALIZATION
   The advantage function is normalized to improve training stability and policy learning efficiency,
   \[ A^* = \frac{A - \mu_A}{\sigma_A} \] (11)

   b: VALUE FUNCTION NORMALIZATION
   The value function loss is normalized as
   \[ V^*_\text{loss} = \frac{1}{N} \sum_{i=1}^{N} \frac{(V_i - R_i)^2}{\sigma R_i} \] (12)

2) ADAPTIVE ADJUSTMENT PARAMETERS
   a: ADAPTIVE LEARNING RATE
   The experiment adopts learning rate \( \eta \) annealing as (13),
   \[ \eta = \eta_0 \times \left(1 - \frac{i_{\text{episode}}}{i_{\text{max}_\text{episode}}} \right) \] (13)

   In the early stage of training, a larger learning rate is adopted to accelerate learning, and a lower learning rate is adopted later to prevent the policy from prematurely converging to a bad local optimum.

   b: ADAPTIVE CLIP VALUE
   The principle of the adaptive clip is the same as that of the adaptive learning rate. A larger clip value is used in the early stage of training to speed up policy learning, and a smaller value in the later stage ensures policy stability,
   \[ \text{clip} = \text{CLIP} \times \left(1 - \frac{i_{\text{episode}}}{i_{\text{max}_\text{episode}}} \right) \] (14)

3) DOMAIN RANDOMIZATION
   To improve the robustness of the agent policy and adapt to a diversified environment, random perturbations are added to the scenario in the training process [41], as shown in (15).
   \[ \begin{cases} \tilde{x} = x + \delta \\ \tilde{y} = y + \delta \end{cases} \] (15)

   where \( x \) and \( y \) are the fighter, jammer, and IADS coordinates, respectively. We run a different random perturbation environment on each random seed to train the agent, enabling it to abstract higher-level complex policy features and avoid over-fitting to one specific environment or policy. Consequently, the learned policy is more robust and better generalized to a new environment.

4) MAXIMIZATION THE ENTROPY OF THE POLICY
   Greater entropy means more randomness and encourages exploration. Therefore, while maximizing the cumulative rewards, the entropy of the policy is maximized and the policy is made as random as possible. The agent can adequately explore the state space to complete the mission, which enhances the robustness and generalization. Entropy is calculated:
   \[ H(\pi(\cdot|s_t)) = -\sum_{t=1}^{T} \pi(\cdot|s_t) \log \pi(\cdot|s_t) \] (16)

5) PARAMETER SHARING
   The critic and actor can share lower layer parameters of the network [37]. The loss function is the sum of policy loss \( J(\theta) \),
value function loss $J(\phi)$, and policy entropy $\mathcal{H}(\pi)$, and its gradient is backpropagated. The training difficulty is reduced by sharing the underlying network features through parameter sharing. The loss function is

$$J(\theta, \phi) = J(\theta) + \lambda_{\text{critic}} J(\phi) + \lambda_{\text{entropy}} \mathcal{H}(\pi)$$  \hspace{1cm} (17)

6) LAYER NORMALIZATION

Each layer of the policy network is normalized, and the weights are initialized by orthogonal initialization, namely, torch.nn.init.orthogonal_(), and the standard deviation is 1. The deviation is initialized by constants, namely, torch.nn.init.constant_(), and the standard deviation is 0, which can greatly improve the algorithm performance.

V. EXPERIMENTS

In this section, we describe the experimental setup and conduct three experiments to test the effectiveness, robustness, and scalability of HMARL model. Moreover, an ablation study is performed to find the most effective tricks.

A. EXPERIMENTAL SETUP

A light UAV swarm cooperative combat environment for SEAD mission is developed in this paper. The scenario is a square of area 100 km $\times$ 100 km. The attack range of the fighter is 15 km, and the jammers range is 25 km. The detection range of the IADS radar is 20 km, the attack distance is 15 km, and the detection range is reduced to 10 km after jamming from a jammer [22], as shown in Figure 6. To complete the mission, the fighter needs to fly to the fire point, and the jammer to the jamming point. The above distance is normalized during training, which facilitates neural network training and prevents the gradient vanishing.

Simulation experiments are completed in different scenario, from static known to dynamic uncertain, and from small- to large-scale. Furthermore, compared with those classic DRL algorithms, we complete the effectiveness (convergence), robustness, scalability testing, and ablation study to verify the performance of the HMARL method and intelligent decision-making capabilities. All experiments are conducted on a computer with a 3.6 GHz Intel i7 CPU, 32 GB of DDR4 RAM, and RTX 3060 GPU, using PyTorch 1.70 and Python 3.6.

| Parameters                      | Values |
|---------------------------------|--------|
| MADQN-IA Parameters            |        |
| Number of layers for DQN        | 3      |
| Number of nodes for each layer  | $(32, 32, N)$ |
| Active function                 | (Relu, Relu, Linear) |
| Reward decay                    | 0.95   |
| Learning rate                   | 0.01   |
| Replay buffer size, batch size  | 64,8   |
| Maximal episode                 | 500    |
| Action exploration, $\eta_0$ decay | 0.2  |
| IAPPO-CA Parameters            |        |
| Number of layers for actor network | 3      |
| Number of layers for critic network | 3      |
| Number of nodes for actor layers | $(64, 64, 4)$ |
| Number of nodes for critic layers | $(64, 64, 1)$ |
| Actor network active function   | (tanh, tanh, tanh) |
| Critic network active function  | (tanh, tanh, tanh) |
| Learning rate for actor, critic network | 0.0001 |
| Clip linear decay               | 0.2    |
| Reward decay gamma              | 0.995  |
| Maximal step per episode        | 1000   |
| Coefficient for policy loss     | 1      |
| Coefficient for value loss      | 0.5    |
| Coefficient for entropy loss    | 0.01   |
| GAE lambda                      | 0.95   |
| Minibatch size                  | 64     |
| Number of random seeds          | 3      |

TABLE 3. Simulation parameters.
The hyperparameter settings are listed in Table 3. The Adam optimizer is employed to optimize all the network parameters [42], and other training tricks are introduced in Subsection IV.D.

**B. EXPERIMENT I: EFFECTIVENESS IN 6V3 KNOWN SCENARIO**

In experiment I, we set up a 6V3 small-scale SEAD scenario. Three fighters and three jammers form three fighter-jammer...
formations (formation 1-3), and the enemy consists of three IADSs. The combat capabilities of the fighter, jammer, and IADS are as described in Subsection 5.A. Therefore, the fighter has to be able to safely destroy the IADS under the jamming suppression of the jammer to complete the mission.

The HMARL based intelligent decision-making model is trained offline for 300 episodes, and the converged HMARL model is used to test the online intelligent decision-making performance through online inference.

1) OFFLINE TRAINING OF THE IAPPO-CA AGENT
First, the IAPPO-CA agent is trained to convergence. We record the average cumulative rewards during the training stage and compare them with asynchronous advantage actor-critic (A3C) and deep deterministic policy gradient (DDPG) algorithms. The effectiveness can be evaluated from the final performance and learning speed. An algorithm’s final performance is better if it achieves the higher final cumulative rewards with fewer training steps. The learning curves of the three IAPPO-CA agents are similar. Therefore, we take the learning curve of formation 1 as an example and show the convergence performance in Figure 7. The shaded region corresponds to the minimum and maximum episode rewards across three random seeds.

As can be seen from Figure 7, A3C algorithm shows very high variance across seeds, indicating substantially worse stability. DDPG algorithm has poor final convergence, while IAPPO algorithm is more stable, has a higher episode rewards, and smaller variance. Therefore, IAPPO-CA agent converges and the performance of IAPPO algorithm is better.

2) OFFLINE TRAINING OF THE MADQN-TA AGENT
We train the MADQN-TA agent embedded the converged IAPPO-CA agent. Since the target allocation policy is relatively simple, it is trained for 200 episodes, and a converged Q-value network is obtained for target allocation.

We record the average episode rewards of three DQN agents and target allocation result and compare the convergence performance with the MADQN without Metropolis technique. The results are shown in Figure 8.

Figure 8(a) shows the comparison of episode rewards learning curve. MADQN with Metropolis technique outperforms MADQN without Metropolis, both in terms of learning speed and the final performance. Figure 8(b) shows target allocation result based on MADQN algorithm with Metropolis technique. Moreover, target allocation is stable after training 80 episodes, and formations 1-3 attack IADS 1-3, respectively.

3) ONLINE INFERENCE OF THE HMARL MODEL
Then we test the intelligent decision-making capabilities online of the HMARL model, consisted of the converged IAPPO-CA and MADQN-TA agents. We input the initial state information to the trained HMARL model and get a cooperative attacking process, as shown in Figure 9.

In Figure 9(a), the HMARL agent first completes the target allocation. The result is that formations 1–3 attack IADS 1–3, respectively. In Figure 9(b), the formation 1 launches a cooperative attacking. The jammer of formation 1 has reached jamming conditions, the detection range of IADS 1 has been degraded, and formations 2 and 3 have not yet reached the jamming conditions. The fighter adopts a specific deceleration tactic of flying around to ensure its safety and waiting for jamming. In Figure 9(c), formation 1 is still attacking, formations 2 and 3 have completed jamming and cooperative attacking, and IADS 2 and 3 have been destroyed. In Figure 9(d), formation 1 has also completed a cooperative attacking. The three formations have completed the 6V3 cooperative decision-making mission, which reflects the strong cooperative formation and intelligent decision-making capability.

C. EXPERIMENT II: ROBUSTNESS IN 6V3 UNCERTAIN SCENARIO
To improve the robustness and generalization of the HMARL model to adapt to the dynamic uncertain scenario, domain randomization is used during training. The model is trained under different initial positions. The formation and IADS starting positions are randomly initialized in a certain square area for each episode. The uncertain scenario of experiment II is shown in Figure 10.

![SEAD environment with different initial positions.](image)

1) OFFLINE TRAINING
First, we train the IAPPO-CA and MADQN-TA agents to converge in the same way as in Subsection 5.B.

2) ONLINE INFERENCE OF THE HMARL MODEL
To verify the robustness of the model in a dynamic and uncertain environment, we load the HMARL model and perform
online inference and robustness tests in a new random scenario. The initial positions of each fighter, jammer, and IADS are randomly generated, and it is different from Figure 9. The results are shown in Figure 11.

Figure 11 shows that the HMARL model can still complete target allocation and cooperative attacking. Therefore, we conclude that the proposed model has certain robustness and generalization performance to the dynamic changing scenarios, can adapt to the uncertain battlefield environment, and has strong practical application value.

**D. EXPERIMENT III: SCALABILITY IN 20V10 LARGE-SCALE SCENARIO**

To further test the scalability of the HMARL model in the large-scale scenario, we set up a 20V10 SEAD scenario in experiment III. Our force consists of ten fighters and ten jammers, forming ten fighter-jammer formations, and the enemy force is ten IADSs and ten targets. The combat capabilities of fighters, jammers, and IADS are the same as in Subsection 5.A. After training 500 episodes, the trained model is used for online inference to test the scalability.

1) OFFLINE TRAINING OF THE IAPPO-CA AGENT

Similarly, we train the IAPPO agent across three different random seeds, record the cumulative rewards during the training
process, and compare it with A3C and DDPG algorithms to obtain the average reward learning curve, as shown in Figure 12.

Figure 12 shows that A3C and DDPG have higher variances and poor convergence, while IAPPO has a higher episode rewards and excellent convergence performance. Moreover, the IAPPO-CA agent has converged and can be used to train the higher level MADQN-TA agent.

2) OFFLINE TRAINING OF THE MADQN-TA AGENT

After training, the average convergence performance and target allocation result of MADQN-TA agent are shown in Figure 13.

As depicted in Figure 13(a), MADQN with Metropolis technique converges faster and the episode rewards are higher than MADQN without Metropolis technique. Therefore, the Metropolis technique improves convergence performance. Figure 13(b) shows the final target allocation result of MADQN with Metropolis technique, which is formations 1-10 attack IADS 1-10, respectively. Finally, we can test the online inference capability of the HMARL model.

3) ONLINE INFERENCE OF THE HMARL MODEL

The converged IAPPO-CA and MADQN-TA agents are cascaded and tested in a 20V10 large-scale scenario. Results are shown in Figure 14.

In Figure 14(a), the formation is attacking cooperatively, and the jammer is looking for a suitable position. In Figure 14(b), after jamming, the IADSs and targets are destroyed successfully by fighters and the cooperative SEAD mission is completed. We conclude that the proposed HMARL model in this paper can scale to a large-scale scenario.

E. ABLATION STUDY

Finally, the effectiveness of different training tricks is compared; that is, the performance of models using all tricks and the advantage function normalization, the value function normalization, the layer normalization, the adaptive learning rate, and adaptive clip annealing, and an ablation study is completed. The impact of these tricks on the model performance provides an empirical reference for subsequent research. The result is shown in Figure 15.
FIGURE 15. Convergence comparison of different tricks.

As we can see from Figure 15, the episode rewards of the model using all training tricks is higher, the advantage function normalization greatly improves the early convergence speed of the model, layer normalization can improve the final performance of the model, and other tricks have relatively little impact on model performance.

VI. CONCLUSION

This paper proposes an end-to-end HMARL method to solve the UAV swarm cooperative decision-making problem. A HMARL agent model is established, where the higher level is the target allocation agent based on MADQN algorithm, and the lower level is the cooperative attacking agent based on IAPPO algorithm. Metropolis criterion and inter-agent information exchange techniques improve the action exploration and learning efficiency of MADQN-TA agent. IAPPO-CA agent can be distributed deployed online and flexibly scaled to large-scale scenarios based on an independent learning paradigm. Numerical experiments demonstrate that the proposed HMARL method is effective, robust, and scalable, and can adapt to uncertain and large-scale scenarios.

In the future, we will further study this joint optimization problem. The higher-level agent can schedule the lower-level agent more flexibly to attack the dynamic changing target. Moreover, a more precise jamming model and weapon model will be a focus of future research, as well as more complicated multi-wave strikes and multi-target scenarios, such as 6V4 and 22V10.

REFERENCES

[1] P. Smith, E. Mayo, J. O’Hara, and D. Griffith, “Combat UAV real-time SEAD mission simulation,” in Proc. Model. Simul. Technol. Conf. Exhibit, Oregon, Portland, Aug. 1999, pp. 4185–4190.
[2] T. C. Barkdoll, D. P. Gaver, K. D. Glazebrook, P. A. Jacobs, and S. Posadas, “Suppression of enemy air defenses (SEAD) as an information duel,” Nav. Res. Log., vol. 49, pp. 723–742, Dec. 2002.
[3] M. Durrah, W. Niland, B. Stolarik, and L. Walp, “UAV cooperative task assignments for a SEAD mission using genetic algorithms,” in Proc. AIAA Guid., Navigat., Control Conf. Exhibit, Aug. 2006, pp. 3257–3266.
[4] M. Haque, M. Egerstedt, and A. Rahmani, “Multilevel coalition formation strategy for suppression of enemy air defenses missions,” J. Aereosp. Inf. Syst., vol. 10, no. 6, pp. 287–296, Jun. 2013.
[5] W. Seo, H. M. Lee, J.-H. Kim, K. Choi, and C.-K. Lee, “A methodology for evaluating mission suitability of manned-unmanned aircraft teaming for SEAD missions,” J. Korean Soc. Aeronaut. Space Sci., vol. 48, no. 11, pp. 935–943, Nov. 2020.
[6] X. H. Wang, J. L. Cheng, and Q. Guo, “Research on intelligent cooperative combat system in air defense suppression mission,” Unmanned Syst. Technol., vol. 3, no. 4, pp. 10–21, Jul. 2020.
[7] G. H. Burgin, L. J. Fogel, and J. P. Phelps, “An adaptive maneuvering logic computer program for the simulation of one-on-one air-to-air combat,” NASA, Washington, DC, USA, Contractor Rep. 2582, 1975.
[8] K. Goodrich and J. Mcmanus, “Development of a tactical guidance research and evaluation system (TGRES),” in Proc. Flight Simul. Technol. Conf. Exhibit, Boston, MA, USA, Aug. 1989, p. 3312.
[9] K. Virtanen, T. Raivio, and R. P. Hausamäki, “An influence diagram approach for one-on-one air combat,” in Proc. 10th Int. Symp. Differ. Games Appl., vol. 2, 2002, pp. 859–864.
[10] K. Virtanen, J. Karelaiti, and T. Raivio, “Modeling air combat by a moving horizon influence diagram game,” J. Guid. Control Dyn., vol. 29, no. 5, pp. 1080–1091, Sep. 2006.
[11] N. Ernest and D. Carroll, “Genetic fuzzy based artificial intelligence for unmanned combat aerial vehicle control in simulated air combat missions,” J. Defense Manag., vol. 6, no. 1, pp. 0374–2167, 2016.
[12] H. Zhang, R. Yang, C. Sun, and H. Han, “Research on multi-aircraft cooperative suppression interference array based on an improved multiobjective particle swarm optimization algorithm,” Math. Problems Eng., vol. 2017, pp. 1–12, Mar. 2017.
[13] T. Guo, N. Jiang, B. Li, X. Zhu, Y. Wang, and W. Du, “UAV navigation in high dynamic environments: A deep reinforcement learning approach,” Chin. J. Aeronaut., vol. 34, no. 2, pp. 479–489, Feb. 2021.
[14] Z. Hu, X. Gao, K. Wan, Y. Zhai, and Q. Wang, “Relevant experience learning: A deep reinforcement learning method for UAV autonomous motion planning in complex unknown environments,” Chin. J. Aeronaut., vol. 34, no. 12, pp. 187–204, Dec. 2021.
[15] Y. Li, W. Han, and Y. Wang, “Deep reinforcement learning with applica- tion to air combat intelligent decision-making of manned/unmanned aerial vehicle cooperative system,” IEEE Access, vol. 8, pp. 67887–67898, 2020.
[16] A. P. Pope, J. S. Ide, D. Micovic, H. Diaz, D. Rosenbluth, L. Ritholtz, J. C. Twedt, T. T. Walker, K. Alcedo, and D. Javorek, “Hierarchical reinfor- cement learning for air-to-air combat,” in Proc. Int. Conf. Unmanned Aircr. Syst. (ICUSAS), Jun. 2021, pp. 275–284.
[17] Z. Sun, H. Piao, Z. Yang, Y. Zhao, G. Zhan, D. Zhou, G. Meng, H. Chen, X. Chen, B. Qu, and Y. Lu, “Multi-agent hierarchical policy gradient for air combat tactics emergence via self-play,” Eng. Appl. Artif. Intell., vol. 96, pp. 104–112, Dec. 2021.
[18] G. Zhang, Y. Li, X. Xu, and H. Dai, “Multiagent reinforcement learning for swarm confrontation environments,” in Intelligent Robotics and Applications (ICIRA) (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2019, pp. 533–543.
[19] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, and J. Oh, “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” Nature, vol. 575, no. 7782, pp. 350–354, 2019.
[20] C. Berner, G. Brockman, B. Chen, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Heise, and R. Józefowicz, “Dota 2 with large scale deep reinforcement learning,” 2019, arXiv:1912.06680.
[21] Z. Yu, C. Jing, and S. Lincheng, “Real-time trajectory planning for UAV air-to-surface attack using inverse dynamics optimization method and receding horizon control,” Chin. J. Aeronaut., vol. 26, no. 6, pp. 1038–1056, 2013.
[22] L. A. Zhang, X. Jia, G. Dara, H. Jeff, K. K. J. Ajay, J. L. Andrew, and A. O. Osondo. (2020). Air Dominance Through Machine Learning: A Preliminary Exploration of Artificial Intelligence-Assisted Mission Planning. Santa Monica, CA, USA. RAND Corporation. [Online]. Available: https://www.rand.org/pubs/research_reports/RR4311.html.
[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antoniouglou, D. Wierstra, and M. Riedmiller, “Playing Atari with deep reinforcement learning,” 2013, arXiv:1312.5660.
[24] R. Sutton and A. Barto, Reinforcement Learning: An Introduction. Cambridge, MA, USA: MIT Press, 1998.
[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and S. Petersen, “Human level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347.

[27] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” 2015, arXiv:1506.02438.

[28] Q. Fu, C.-L. Fan, Y. Song, and X.-K. Guo, “Alpha C2—An intelligent air defense commander independent of human decision-making,” IEEE Access, vol. 8, pp. 87504–87516, 2020.

[29] R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” Artif. Intell., vol. 112, nos. 1–2, pp. 181–211, 1999.

[30] A. G. Barto and S. Mahadevan, “Recent advances in hierarchical reinforcement learning,” Discrete Event Dyn. Syst., vol. 13, nos. 1–2, pp. 41–77, 2003.

[31] M. M. Botvinick, “Hierarchical reinforcement learning and decision making,” Current Opinion Neurobiol., vol. 22, no. 6, pp. 956–962, 2012.

[32] G. Comanici and D. Precup, “Optimal policy switching algorithms for reinforcement learning,” in Proc. 9th Int. Conf. Auto. Agents Multiagent Syst. (AAMAS), 2010, pp. 709–714.

[33] L. Bacon, J. Harb, and D. Precup, “The option-critic architecture,” in Proc.31th AAAI Conf. Artif. Intell., 2017, pp. 1726–1734.

[34] S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, “Feudal networks for hierarchical reinforcement learning,” in Proc. 34th Int. Conf. Mach. Learn., 2017, pp. 3540–3549.

[35] Q. Fu, C.-L. Fan, Y. Song, and X.-K. Guo, “Alpha C2—An intelligent air defense commander independent of human decision-making,” IEEE Access, vol. 8, pp. 87504–87516, 2020.

[36] L. Bacon, J. Harb, and D. Precup, “The option-critic architecture,” in Proc. 9th Int. Conf. Auto. Agents Multiagent Syst. (AAMAS), 2010, pp. 709–714.

[37] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. S. Torr, M. Sun, and S. Whiteson, “Is independent learning all you need in the StarCraft multi-agent challenge?” 2020, arXiv:2011.09533.

[38] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, “QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning,” 2018, arXiv:1803.11485.

[39] W. Shi, J. Li, H. Wu, C. Zhou, N. Cheng, and X. Shen, “Drone-cell trajectory planning and resource allocation for highly mobile networks: A hierarchical DRL approach,” IEEE Internet Things J., vol. 8, no. 12, pp. 9800–9813, Jun. 2021.

[40] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Sep. 2017, pp. 23–30.

[41] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.

LONGFEI YUE was born in Shaanxi, China, in 1995. He received the M.S. degree from Air Force Engineering University, where he is currently pursuing the Ph.D. degree. His research interests include deep reinforcement learning-based intelligent mission planning and UAV swarm cooperative planning.

RENNONG YANG was born in Sichuan, in 1969. He is currently a Professor and a Ph.D. Tutor with Air Force Engineering University. His research interests include intelligent decision, expert systems, and autonomous planning.

JIALIANG ZUO was born in Nanchang, in 1986. He received the Ph.D. degree from Air Force Engineering University. He is currently an Associate Professor and a Researcher. His research interests include intelligent decision and intelligent air combat and deep learning.

YING ZHANG was born in 1988. She is currently an Assistant Professor with Air Force Engineering University. Her research interests include multi-objective optimization and intelligent mission planning.

QIUNI LI was born in 1985. She is currently an Assistant Professor with Air Force Engineering University. Her research interests include mission planning and intelligent air combat.

RENNONG YANG was born in Sichuan, in 1969. He is currently a Professor and a Ph.D. Tutor with Air Force Engineering University. His research interests include intelligent decision, expert systems, and autonomous planning.

VOLUME 10, 2022

**...**