Hindawi
Complexity
Volume 2022, Article ID 3551508, 13 pages
https://doi.org/10.1155/2022/3551508

Research Article

Deep Reinforcement Learning for UAV Intelligent Mission Planning

Longfei Yue,[1] **Rennong Yang**,[1] **Ying Zhang**,[1] **Lixin Yu**,[1] **and Zhuangzhuang Wang**[2]

[1]*Air Traffic Control and Navigation College, Air Force Engineering University, Xi'an 710051, China*
[2]*Aviation Maintenance NCO School, Air Force Engineering University, Xinyang 464000, China*

Correspondence should be addressed to Ying Zhang; trustedagent@163.com and Lixin Yu; corresylf2@163.com

Received 8 November 2021; Accepted 16 March 2022; Published 31 March 2022

Academic Editor: Wen-Long Shang

Rapid and precise air operation mission planning is a key technology in unmanned aerial vehicles (UAVs) autonomous combat in battles. In this paper, an end-to-end UAV intelligent mission planning method based on deep reinforcement learning (DRL) is proposed to solve the shortcomings of the traditional intelligent optimization algorithm, such as relying on simple, static, low-dimensional scenarios, and poor scalability. Specifically, the suppression of enemy air defense (SEAD) mission planning is described as a sequential decision-making problem and formalized as a Markov decision process (MDP). Then, the SEAD intelligent planning model based on the proximal policy optimization (PPO) algorithm is established and a general intelligent planning architecture is proposed. Furthermore, three policy training tricks, i.e., domain randomization, maximizing policy entropy, and underlying network parameter sharing, are introduced to improve the learning performance and generalizability of PPO. Experiments results show that the model in this work is efficient and stable, and can be adapted to the unknown continuous high-dimensional environment. It can be concluded that the UAV intelligent mission planning model based on DRL has powerful intelligent planning performance, and provides a new idea for researching UAV autonomy.

1. Introduction

Mission planning is the process of making an operational plan, including a route plan, weapon plan, and avionics plan [1, 2]. Intelligent planning capability is an important symbol of unmanned aerial vehicle (UAV) autonomy. With the development of UAV technology, UAVs can fly independently and complete simple missions, such as reconnaissance and strike, which greatly improves efficiency and reduces the labor cost. However, for complex cooperative missions, UAV intelligent planning is still a key research issue.

Mission planning is a decision optimization problem that solves the optimal solution of mission objective function under certain constraints, such as the shortest route [3], minimum threat [4], and maximum efficiency [5, 6]. Because the mission planning problem considers many mutually coupled factors, large decision space, and nonlinear constraints, traditional mission planning is mostly solved by intelligent optimization algorithms. Xin et al. [3] modeled the route planning problem by solving the shortest-path problem from the starting point to the target point, established an optimization model based on an ant colony optimization (ACO) algorithm, and searched for the shortest route. Zhang et al. [4] studied the tactical maneuver planning problem. Taking the minimum threat of a fighter and the maximum damage effectiveness of ground targets as the optimization objectives, and considering the capability constraints of weapons and equipment, the optimal flight route and weapon delivery time were solved using the multiobjective evolutionary algorithm based on decomposition (MOEA/*D*) [5]. Aiming at the task allocation problem of UAV identification, attack, and evaluation targets, a genetic algorithm (GA) was used to solve the optimal allocation result [6]. Zhang et al. [7] studied electronic warfare mission planning. By taking the route safety width and electronic jamming effect of a jammer as the objective function, the multiobjective particle swarm optimization

(PSO) algorithm was used to solve it, and the optimal jamming array model was obtained. A search-based intelligent optimization algorithm can find the global optimal solution or suboptimal solution of the complex objective function through the parallel optimization mechanism, but its essence is still random search. Each solution can only be researched in the static and known environment (explicit objective function and constraints) and cannot be generalized to the dynamic and unknown environment. The computational complexity increases exponentially with the growth in problem scale. Therefore, its application has certain limitations for rapid planning and dynamic scenarios of future large-scale operations.

In recent years, due to the expansion of computing power, the emergence of big data [8], and the development of artificial intelligence (AI) algorithms, learning-based methods such as neural networks [9, 10] and reinforcement learning (RL) [11] have promoted the second wave of AI. From AlphaGo [12] to AlphaGo Zero [13], AlphaZero [14], and AlphaStar [15], DRL has achieved a series of breakthroughs in a range of challenging domains. Among such methods, deep learning (DL) has been used to solve high-dimensional mapping problems, RL has been used to solve sequential decision-making problems, and DRL has been successfully applied to a series of robotics [16, 17], autonomous driving [18], real-time strategy (RTS) games [19], and optimization and scheduling [20] problems. A learning-based method, also known as the data-driven method, refers to feeding data to improve the prediction or decision-making performance of a model. Such method uses a neural network to learn or fit the complex and high-dimensional nonlinear relationship between input and output, so as to achieve the minimal mean square error or optimal prediction and decision results, and saves the mapping network parameters to realize offline training and online inferencing. It also has certain robustness and generalization for new input data, which is suitable for fast and dynamic mission planning. The comparison of the two methods is shown in Table 1.

Therefore, by taking a high-risk typical suppression of enemy air defense (SEAD) mission planning [21, 22] as an example, we propose an end-to-end UAV intelligent mission planning method based on DRL. First, the mission planning problem of SEAD operation is formalized, then the basic principles of a DRL algorithm are introduced, and a DRL-based intelligent planning model is established. Finally, the superiority and potential value of this method are analyzed and verified by simulation experiments.

2. SEAD Mission Planning Problem Formulation

A SEAD mission is an operational style of offensive counter-air (OCA) combat carried out by an air force. Its mission goal is to break through the enemy's surface-to-air missile (SAM) threat and strike the enemy's radar or target through a cooperative combat between attacking UAV, named fighter, and jamming UAV, named jammer. A schematic of a SEAD mission is presented in Figure 1.

In Figure 1, the mission is to use a fighter to safely destroy an enemy SAM. However, since the detection range of the enemy SAM is longer than the attack range of the fighter, the fighter faces the threat of being detected and attacked by the SAM. Therefore, a jammer is required to jam the SAM to reduce its detection range. Only then the fighter can take the opportunity to attack and destroy the SAM. Therefore, the jammer must jam at the right position and time, and the fighter must attack at the right position and time simultaneously. The two cooperate to complete the mission.

To summarize, mission planning is essentially a sequential decision-making problem. Under different space-time sequence states, a combat unit adopts the optimal decision-making sequence to transfer from the initial situation to the termination situation to achieve the mission objectives. Therefore, SEAD mission planning can be modeled as an end-to-end sequence optimization problem from the state (position and situation) to the decision (maneuver, attack, and jamming). The optimization goal is to solve an optimal state decision sequence to meet the needs of fighters and jammer to destroy an enemy SAM and ensure their own safety through tactical cooperation, as shown in Figure 2.

3. Deep Reinforcement Learning

3.1. Principles of Reinforcement Learning. RL is a machine learning approach for teaching agents how to solve tasks by trial and error. The main characters of RL are the agent [23] and the environment. The agent perceives an initial state of the environment and then decides on an action to take. The environment changes when the agent acts on it and gives the agent an instant reward signal, while the agent transfers to the next state and continues to choose new actions until reaching a termination state. The goal of the agent is to maximize the sum of rewards in the entire decision-making process, i.e., to find the optimal policy. Therefore, RL is a decision optimization method. The reinforcement learning framework is shown in Figure 3.

RL is usually modeled as a finite Markov decision process, which is represented by five tuples $\langle S, A, R, P, \gamma \rangle$, in which S is a finite state set, A is a finite action set, R is a reward function, P is a state transition function, and γ is a discount factor, which is used to calculate the long-term discount rewards. Assuming that the state $s \in S$ of the agent at time t is to take action $a \in A$ according to policy π: $S \longrightarrow A$, the environment will feed back an instant reward $r \in R$ to the agent, and the agent will transfer to a new state $s' \in S$. 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" problem, which spurred the development of DRL [23–26].

3.2. Deep Reinforcement Learning. DRL refers to the combination of RL with DL. DRL uses a neural network to approximate policy and value functions to solve the high-

Table 1: Traditional and intelligent planning.

Category	Traditional planning	Intelligent planning
Algorithm	Intelligent optimization algorithm	Deep reinforcement learning
Method	Search-based	Learning-based
Advantage	Parallel optimization	Offline training and online inferencing, high-dimension, scalable
Limitations	Not scalable, static	Computational power

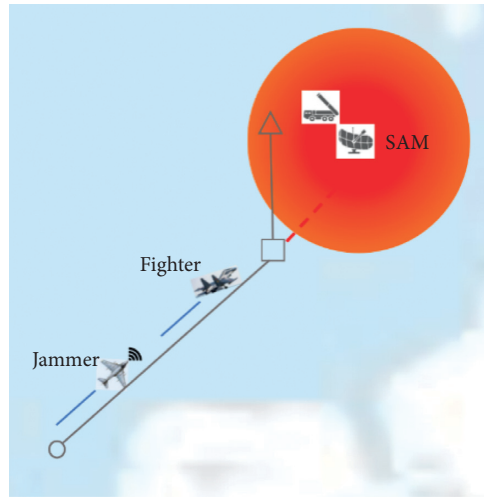

Figure 1: Schematic of a SEAD mission.

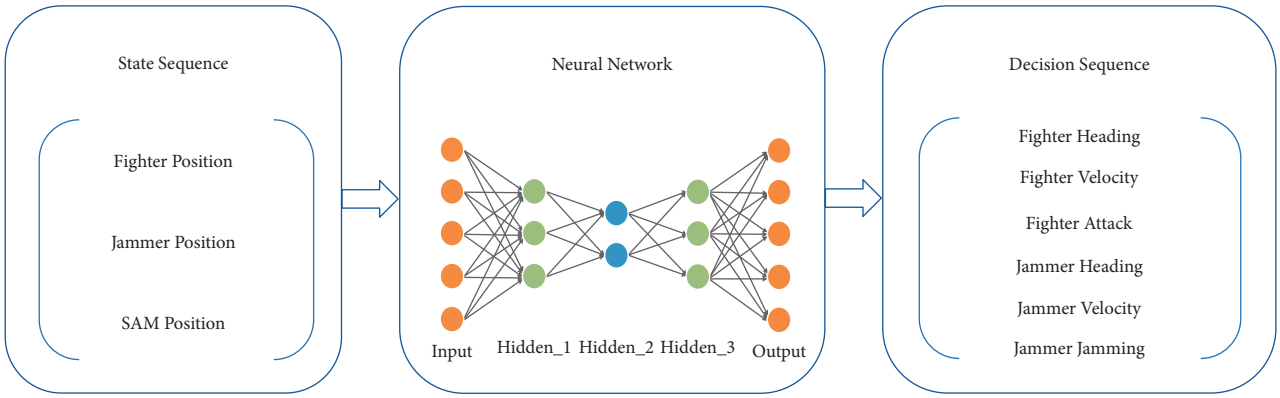

Figure 2: Intelligent mission planning SEAD model.

dimensional mapping problem. The goal of the agent is to find the parameterized policy π_θ with the maximum expected return $J(\pi_\theta) = \mathrm{E}_{\tau\sim\pi_\theta}[R(\tau)]$, which refers to the discounted cumulative rewards $R(\tau) = \sum_{t=0}^{\infty}\gamma^t r_t$ on the trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots)$. θ is a policy parameter. The optimal policy is as follows:

$$\pi_\theta^* = \arg\max_{\pi_\theta} \mathrm{E}_{\tau\sim\pi_\theta}\left(\sum_{t=0}^{\infty}\gamma^t r_t\right). \tag{1}$$

A DRL algorithm is divided into three learning paradigms: value-based, policy gradient, and actor-critic. The actor-critic reinforcement learning algorithm integrates the value function and policy gradient, and uses the value function error to guide the policy gradient update to accelerate the learning speed. The policy is updated through the gradient of expected return, which can be written as follows:

$$\nabla_\theta J(\pi_\theta) = \mathrm{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log\, \pi_\theta(a|s)R(\tau)\right], \tag{2}$$

where $\pi_\theta(a|s)$ is an actor and $R(\tau)$ a critic, and can also take other forms, such as a state-action value function $Q^\pi(s_t, a_t)$, advantage function $A^\pi(s_t, a_t)$, or temporal difference (TD) residual $\mathrm{G}_t = r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$. When the critic takes the TD residual and the value function is approximated by the neural network with parameter ω, the derivative of equation (2) is obtained, the critic is updated according to equations (3) and (4), and the actor is updated according to equation (5).

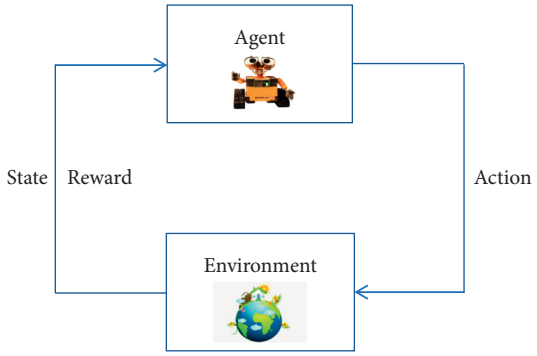

Figure 3: Schematic of reinforcement learning framework.

$$\delta \leftarrow G_t - \hat{v}(S_t, \omega), \tag{3}$$

$$\omega \leftarrow \omega + \beta\delta\nabla_w \hat{v}(s_t, \omega), \tag{4}$$

$$\theta \leftarrow \theta + \alpha\delta\nabla_\theta \log \pi_\theta(a_t|s_t). \tag{5}$$

3.3. Proximal Policy Optimization Algorithm. Because proximal policy optimization (PPO) [27] algorithm is a simple, stable, and easy-to-implement actor-critic algorithm. Both Dota2 AI (OpenAI Five) [28] and Honor of Kings AI Juewu (Tencent) [29] are implemented by PPO. The PPO algorithm addresses the computationally expensive problem in the trust region policy optimization (TRPO) [30] algorithm, which needs enormous calculations to ensure monotonous policy performance improvement. Through the first-order approximation, the surrogate loss function is optimized. The new policy is calculated in each iteration, and it is close to the old policy. The policy is optimized in the direction of minimizing the loss function (maximizing the expected return). The PPO algorithm achieves a balance between sampling efficiency, algorithm performance, and engineering implementation complexity.

In a PPO algorithm, a critic uses the advantage function to measure the quality of the action, and (2) becomes the following:

$$\nabla_\theta J(\pi_\theta) = \mathrm{E}_{a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A^\pi(s,a)]. \tag{6}$$

Because PPO is an on-policy method, importance sampling is introduced to improve sample utilization, and the old $\pi_{\theta'}$ is used the sample to obtain the following:

$$\nabla_\theta J(\pi_\theta) = \mathrm{E}_{a\sim\pi_{\theta'}}\left[\frac{\pi_\theta}{\pi_{\theta'}} A^{\pi_{\theta'}}(s,a)\nabla_\theta \log \pi_\theta\right]. \tag{7}$$

Because $\pi_\theta \nabla_\theta \log \pi_\theta = \nabla\pi_\theta$, Eq. (7) becomes the following:

$$\nabla_\theta J(\pi_\theta) = \mathrm{E}_{a\sim\pi_{\theta'}}\left[\frac{\nabla\pi_\theta}{\pi_{\theta'}} A^{\pi_{\theta'}}(s,a)\right]. \tag{8}$$

The optimization objective function corresponding to the gradient is as follows:

$$J(\pi_\theta) = \mathrm{E}_{a\sim\pi_{\theta'}}[\pi_\theta/\pi_{\theta'} A^{\pi_{\theta'}}(s,a)]. \tag{9}$$

In practical application, based on the sampling estimation expectation, the optimization objective of PPO, i.e., the surrogate loss function, is simplified as shown in equation (10). The policy update amplitude is limited by the truncation operation, i.e., CLIP, to ensure the training stability.

$$J^{CLIP}(\theta) = \sum_{(s,a)} \min(r(\theta)\widehat{A}, \mathrm{clip}(r(\theta), 1-\varepsilon, 1+\varepsilon)\widehat{A}), \tag{10}$$

where $r(\theta) = \pi_\theta(a|s)/\pi_{\theta'}(a|s)$ is the ratio of new and old policies, and ε is a hyperparameter. The generalized advantage estimation [31] is used to calculate the advantage and keep the variance and deviation estimated by the value function small, as shown in the following:.

$$\widehat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=1}^{\infty} (\gamma\lambda)^l (r_t + \gamma V(s_{t+l+1}) - V(s_{t+l})). \tag{11}$$

The PPO algorithm is executed as Table 2:

4. Modeling of SEAD Intelligent Planning Based on the PPO Algorithm

4.1. UAV Kinematics Equation. In this paper, we aim to study the feasibility and future value of an end-to-end DRL method in intelligent mission planning. Therefore, we construct a simple two-dimensional (2-D) environment, where the fighter and jammer adopt a three-degree-of-freedom (3-DOF) model, and its kinematic equation is as follows:

$$\begin{cases} \dot{x}_f = v_f \cos\varphi_f, \\ \dot{y}_f = v_f \sin\varphi_f, \\ \dot{x}_j = v_j \cos\varphi_j, \\ \dot{y}_j = v_j \sin\varphi_j, \end{cases} \tag{12}$$

where $\dot{x}_f$, $\dot{y}_f$, v_f, and φ_f represent the differentiation of coordinate X and Y, speed, and heading of the fighter, respectively, and $\dot{x}_j$, $\dot{y}_j$, and v_j, and φ_j represent the differentiation of coordinate X and Y, speed, and heading of the jammer, respectively. The heading is any continuous value in $(-\pi, \pi)$ and the speed is a continuous value in $(0, 1]$.

4.2. MDP Modeling

4.2.1. State Space. The state of the fighter, jammer, and SAM is defined as the coordinate in 2-D space, which are continuous values and represented by (x_f, y_f), (x_j, y_j), and (x_s, y_s), as shown in Table 3.

4.2.2. Action Space. The actions of the fighter and jammer are their respective heading and speed, as shown in Table 4. The UAV can change the position in 2-D space by controlling the heading, and reaching the attack position by controlling the speed. When the fighter fires the missile and

Table 2: Algorithm: PPO (CLIP).

Proximal policy optimization algorithm (PPO)
1. For $i = 1$ to N do
2. Run policy $\pi_{\theta'}$ for T timesteps, collecting (s, a, r, s', d);
3. Estimate return $R(\tau)$ and advantage $\widehat{A}_t^{GAE(\gamma,\lambda)}$;
4. For $k = 1$ to K do
5. Sample minibatch from the trajectory, calculate policy loss and value loss;
6. Optimize surrogate loss function $J(\theta)$;
7. Update policy parameter $\theta' \leftarrow \theta$;
8. End for
9. End for

Table 3: State-space description.

State space	Position coordinate
Dimension	6
Description	Coordinates X, Y of fighter, jammer, and SAM

Table 4: Action-space description.

Action space	Position coordinate
Dimension	4
Description	Heading and velocity of fighter and jammer

the jammer turns on the jamming, it is automatically completed by default by setting the distance conditions. In an actual mission, it is necessary to calculate the fire time, position, and parameters in detail.

4.2.3. Reward Function. The design of the reward function follows the principle of Occam's razor, i.e., it should be simple and effective. If the jammer suppresses the SAM without entering its missile range, and the fighter destroys the enemy's SAM radar without entering its missile range, it will get a reward +1. If the jammer or fighter enters the range of SAM or flies out of the environmental boundary, it will get a reward −1. For other circumstances, reward shaping is adopted according to experience knowledge, and a continuous reward of the relative distance is given to guide the agent to learn. The reward function formula is expressed as follows:

$$r = \begin{cases} 1, d_s < d_{fs} \le d_f \text{ and } d_s < d_{js} \le d_j, \\ -1, d_{fs} \le d_s \text{ or } d_{js} \le d_s \text{ or}, \\ x_{f,j}, y_{f,j} \le 0 \text{ or } x_{f,j}, y_{f,j} \ge x_{\max}, \\ x_f + y_f + x_j + y_j, \\ -x_s - y_s - 0.1, \text{other}, \end{cases} \tag{13}$$

where d_{fs} and d_{js} represent the distance between fighter and SAM and between jammer and SAM, respectively, d_s represents the attack range of the SAM, d_f represents the attack range of the fighter, and d_j represents the jamming range of the jammer.

4.2.4. Environment Class Development. The environment class mainly includes two parts: the step() and reset() functions. The step() function realizes the deterministic state transition according to the UAV kinematics equation, and returns the judgment results of the new state, reward, and termination. The reset() function is designed to reset the fighter and jammer to the initial position.

4.3. Policy Training Tricks

4.3.1. Domain Randomization. To improve the robustness of the agent policy and adapt to diversified inputs, add disturbances to the inputs in the training stage [32], as shown in Eq. (14); that is, train the agent in different environments with parameter disturbances on each random seed, so that the agent can abstract higher-level policy features, avoid overfitting the one environment and policy, and that the final learned policy is the more robust and better generalization to new environments.

$$\begin{cases} \tilde{x} = x + \text{rand}, \\ \tilde{y} = y + \text{rand}. \end{cases} \tag{14}$$

4.3.2. Maximization Policy Entropy. Entropy is used to measure the randomness of random variables. The greater the entropy, the more random it is. 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 fully explore the state space to avoid the policy falling into local minima, and can explore multiple feasible schemes to complete the mission, which improves the exploration performance, robustness, and generalization performance of the policy. The calculation formula is shown as follows:

$$H\left(\pi\left(\cdot|s_t\right)\right) = -\sum_t \pi\left(\cdot|s_t\right) \log\, \pi\left(\cdot|s_t\right). \tag{15}$$

4.3.3. Parameter Sharing. Actor and critic networks share network parameters. The loss function is the sum of policy loss, value function loss, and policy entropy, and the loss function gradient is backpropagated. The training difficulty is reduced by sharing the underlying network characteristics through parameter sharing. The loss function is expressed as follows:

$$\text{Total\_loss} = \text{loss\_}a + 0.5 \times \text{loss\_}v + 0.01 \times \text{loss\_}e. \tag{16}$$

4.3.4. Construction of Intelligent Planning Model. To summarize, an end-to-end intelligent mission planning model for SEAD mission is established. The input is the position of fighter, jammer, and SAM, which is normalized by zero mean (Z-score) according to equation (17), and then put into the two-layer fully connected neural network. Finally, the heading, speed of the fighter, and jammer are output. The optimization goal is to maximize the cumulative rewards. The solution is the optimal or suboptimal policy, and the policy network architecture of the intelligent planning model is shown in Figure 4.

$$x^* = \frac{x - \mu}{\sigma}. \tag{17}$$

Therefore, this paper adopts the idea of offline training and online inferencing. First, the agent is trained in the environment. After the training is completed, the inference module is carried out to test the planning performance of the agent. The research framework is shown in Figure 5.

A general intelligent planning architecture is proposed, including environment, planner (agent), and controller. First, the planner inputs the initial situation and interacts with the environment, that is, offline training. The trained planner can be directly applied and input the new initial situation for online inferencing. The inference decision sequence, i.e., planning results, is put into the controller for execution. The architecture realizes the entire process of intelligent planning, as shown in Figure 6.

5. Evaluation in Simulation Experiments

5.1. Experimental Setup and Training Tricks. In this paper, the SEAD environment is a square area of $100 \times 100\,\text{km}^2$. The fighter position is (40, 40), the attack range is 30 km, the jammer position is (30, 30), the attack range is 50 km, the SAM position is (80, 80), and the attack range is 40 km. After jamming, the SAM attack range is reduced to 25 km, as shown in Figure 7. In the experiment, the above range is reduced 100 times for normalization, which is easy for neural network training and can prevent gradient vanishing. The simulation environment uses *Python* 3.6 and PyCharm. The intelligent planning experiments of three different scenarios are completed separately. Compared with other classical DRL algorithms, robustness and ablation studies are carried out to verify the algorithm performance.

The hyperparameter settings in this work are shown in Table 5. The neural network adopts orthogonal initialization, the optimizer is Adam [33], and other training parameters are described in Section 5.1.1.

5.1.1. State Space. The status of the fighter, jammer, and SAM is defined as the coordinate position in 2-D space, which are continuous values and represented by (x_f, y_f), (x_j, y_j), and (x_s, y_s).

(1) Advantage function normalization. The advantage function value is normalized to improve the training stability and policy learning skills. The formula is as follows:

$$A^* = \frac{A - \mu_A}{\sigma_A}. \tag{18}$$

(2) Value function normalization. Similarly, the value function loss is also normalized. The policy training skills are studied and compared in the subsequent simulation. The formula is as follows:

$$V^*_{\text{loss}} = \sum_i^N \frac{(V_i - R_i)^2}{6T\sigma_{R_i}}. \tag{19}$$

5.1.2. Adaptive Adjustment Parameters

(1) Adaptive learning rate. In the early training stage, a larger learning rate is adopted to accelerate convergence, and in the later training stage, a smaller learning rate is adopted to find the optimal value. The formula is as follows:

$$lr = LR^*\left(1 - \frac{i\_EP}{MAX\_EP}\right). \tag{20}$$

(2) Adaptive clip value. The clip adaptive change is consistent with the change of learning rate. A larger clip value is allowed to accelerate the policy update in the early training stage, and a smaller clip value is used in the later stage to ensure the policy update is stable. The formula is as follows:

$$\text{clip} = \text{CLIP} \times \left(1 - \frac{i\_EP}{\text{MAX}\_EP}\right). \tag{21}$$

5.2. Intelligent Planning for Fighter-Jammer Scenario. In Experiment 1, we set up a fighter-jammer scenario, as depicted in Section 5.1. Therefore, the fighter has to be able to safely destroy the SAM under jamming to complete the mission.

Therefore, in this work, the intelligent planning model is used to simulate the 300 episodes to train the model, and use the trained model for online inferencing to test the intelligent planning performance and cooperative attack performance.

5.2.1. Offline Training. Three different random seeds are used to train the PPO policy and value function network. We record the average timesteps reward in the training process, and compare it with the classical advantage actor-critic (A2C) and TRPO algorithms to obtain the cumulative rewards learning curve, as shown in Figure 8.

As it can be seen from the above figure, the A2C model has a large variance and the TRPO model has poor convergence performance. However, the PPO model used in this work has higher episode rewards, more stable training, smaller episode rewards variance, and good robustness, so its performance is better.

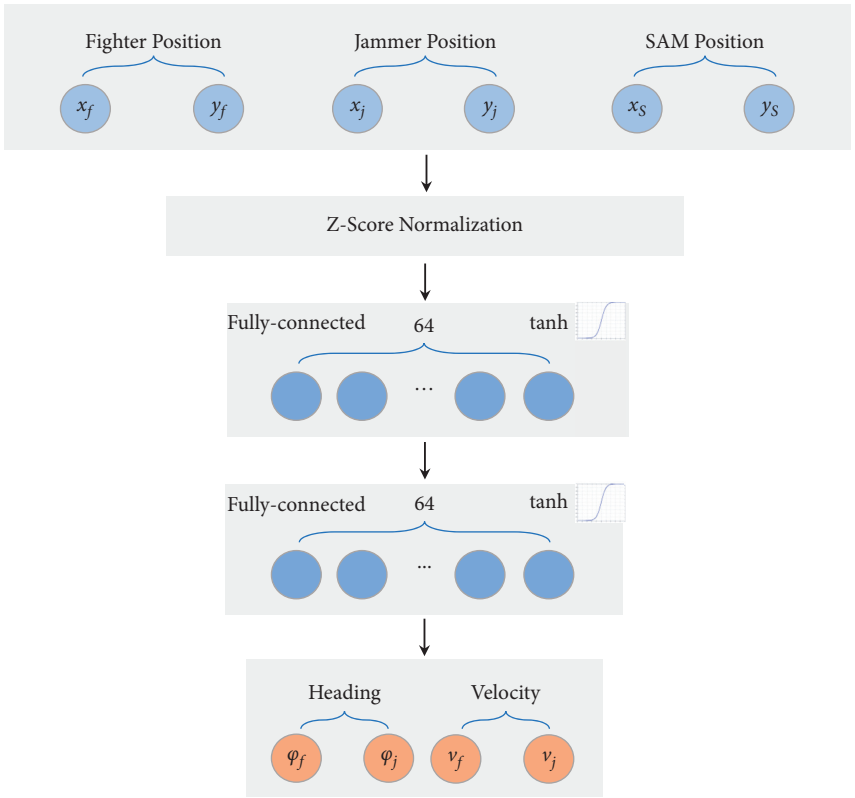

Figure 4: Illustration of observation vector and policy network architecture used in the present work.

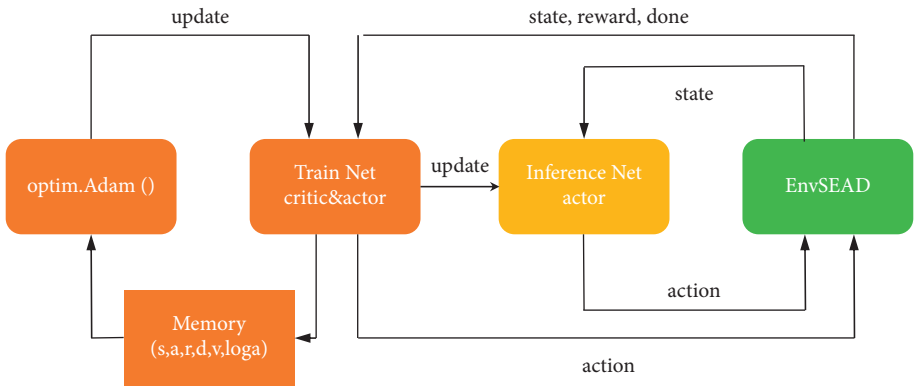

Figure 5: Research framework used in the present work.

5.2.2. Online Planning. The trained model is used to input the initial situation information of the environment, test the online inferencing performance of the model, and obtain the cooperative attack process, as shown in Figure 9.

It can be seen from Figure 9(a) that the fighter and jammer cleverly complete a cooperative attack. Before the jammer successfully jams, the fighter does not enter the detection range of the SAM or start an attack for its own safety. As can be seen from Figure 9(b), when the jammer jams the SAM and degrades the detection range of the SAM, the fighter attack quickly and successfully destroys the SAM, which reflects strong intelligent cooperative planning performance.

5.3. Intelligent Planning for Fighter-Decoy Scenario. To further test the intelligent planning performance of the model proposed in this paper, a cooperative attack for fighter-decoy scenario is designed in which the SAM attack range is 20 km and that of the fighter is also 20 km, which could not directly and safely destroy the SAM and target. The decoy is a low-cost expendable UAV, which could be sacrificed. Therefore, the fighter has to attack with the decoy cooperatively, sacrificing the decoy first and utilizing the interval between the SAM attacks on the decoy to destroy the SAM and target to complete the mission. Therefore, the reward function is modified as follows:

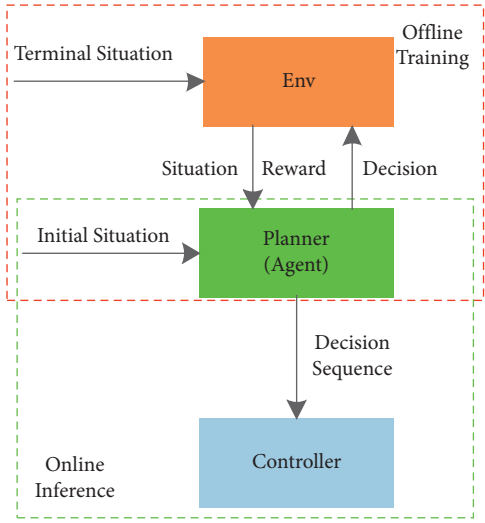

Figure 6: General architecture of intelligent planning.

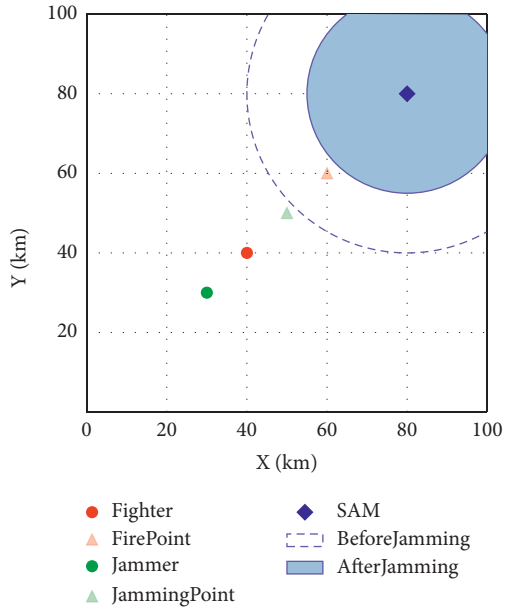

Figure 7: SEAD environment in present work.

$$r = \begin{cases} 1, d_{fs} \le d_f \text{ and } d_{ds} \le d_s \\ -1, d_{ds} > d_s \text{ or } x_{f,d}, y_{f,d} \le 0 \\ \text{or } x_{f,d}, y_{f,d} \ge x_{\max} \\ x_f + y_f - x_d - y_d + 0.8, \text{other}, \end{cases} \tag{22}$$

where d_{fs} and d_{ds} represent the distance between fighter and SAM and between decoy and SAM, respectively, d_s represents the attack range of the SAM, d_f represents the attack range of the fighter, x_d and y_d represent the coordinate X and Y of the decoy.

5.3.1. Offline Training. Similarly, the policy and value function network of PPO are trained on three different random seeds, and the cumulative rewards of each episode in the training process are recorded and compared with the intelligent planning model based on A2C and TRPO to obtain the episode rewards learning curve, as shown in Figure 10.

It can be seen from the figure that the A2C and TRPO models have large variance, low training stability, and poor convergence, while the PPO model proposed in this paper obtains higher cumulative episode rewards, a smooth and stable learning curve, and excellent convergence performance.

Table 5: Hyperparameter settings.

Parameter	Value
max_episode	300
learning_rate	0.0003
Clip	0.2
Gamma	0.995
Lambda	0.95
max_step_per_round	1000
coeff_loss_policy	1
coeff_loss_value	0.5
coeff_loss_entropy	0.01
minibatch_size	64
num_seed	3

5.3.2. Online Planning. The trained PPO model is tested in the environment to verify the planning performance of the model. The results are as follows:

In Figure 11(a), the decoy flies directly to the SAM to attract SAM radar tracking and attacking, while the fighter waits for an opportunity to fly around. In Figure 11(b), the fighter utilizes the time interval between the SAM tracking, locking on to the decoy, and completing the attack quickly, successfully destroying the SAM and the targets. The fighter sacrifices the decoy, but completes the mission, which shows that the PPO-based intelligent planning model has certain tactical cooperative planning performance.

5.4. Robustness Studies. The robustness of the intelligent planning model is tested next, and certain randomness is added to the training environment. As shown in Figure 12, the starting positions of fighter, decoy, and SAM can change randomly in a certain surrounding area to test the generalization performance and robustness of the trained model in the unknown environment.

The test results are shown in Figure 13. In Figure 13, the initial position of the fighter is (20, 20), that of the decoy (85, 95), and that of the SAM (55, 60). Upon being input into the trained model, it is found that the fighter will still intelligently wait for the decoy to enter the SAM range first, and then take the opportunity to attack quickly and successfully destroy the SAM and target, as shown in Figure 13. Therefore, this shows that the model has certain robustness and generalization performance to unknown situations, can adapt to an uncertain environment, and has strong practical application value.

5.5. Ablation Studies. Finally, the effectiveness of different training tricks are compared; that is, the performance of models using all tricks, and the advantage function normalization, layer

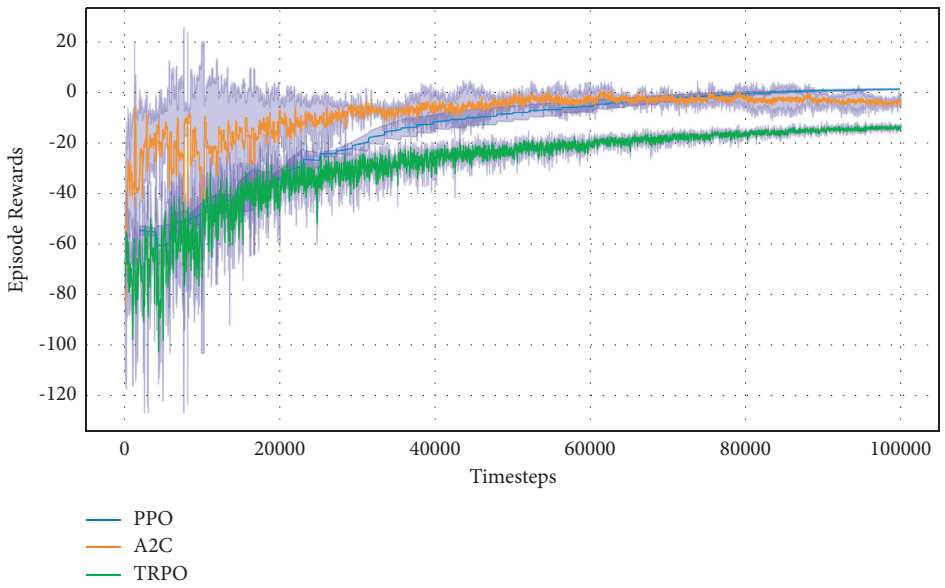

Figure 8: Learning curves for the fighter-jammer scenario. The shaded region represents the standard deviation of average evaluation over three trails.

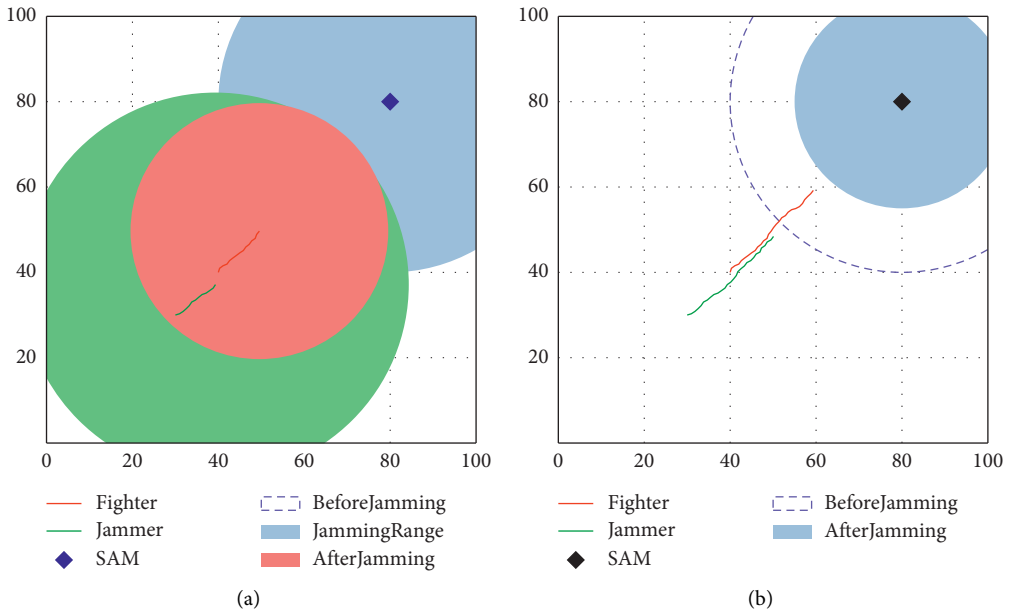

Figure 9: (a) Fighter and jammer flight cooperatively; (b) jammer jamming the Sam to reduce the weapon range, and the fighter takes the opportunity to attack the Sam. When the Sam is destroyed, it turns black.

orthogonal initialization, value function normalization, adaptive learning rate, and adaptive clip. The ablation studies are completed. The results are shown in Figure 14.

As it can be seen from Figure 14, the episode rewards using all training tricks are higher. The advantage function normalization will greatly improve the early convergence speed of the model, layer orthogonal initialization can improve the final performance of the model, and other tricks have relatively little impact on model performance.

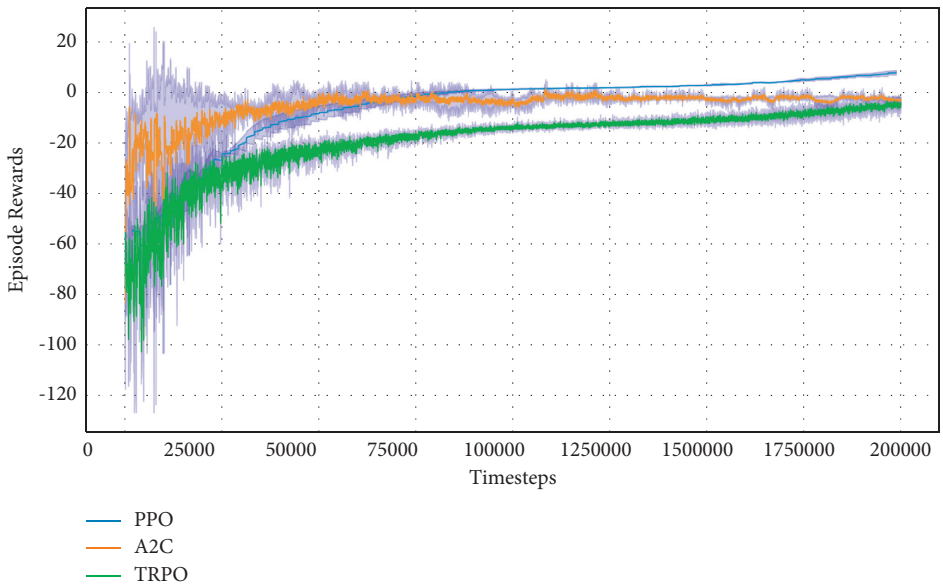

Figure 10: Learning curves for the fighter-decoy scenario. The shaded region represents a standard deviation of average evaluation over three trails.

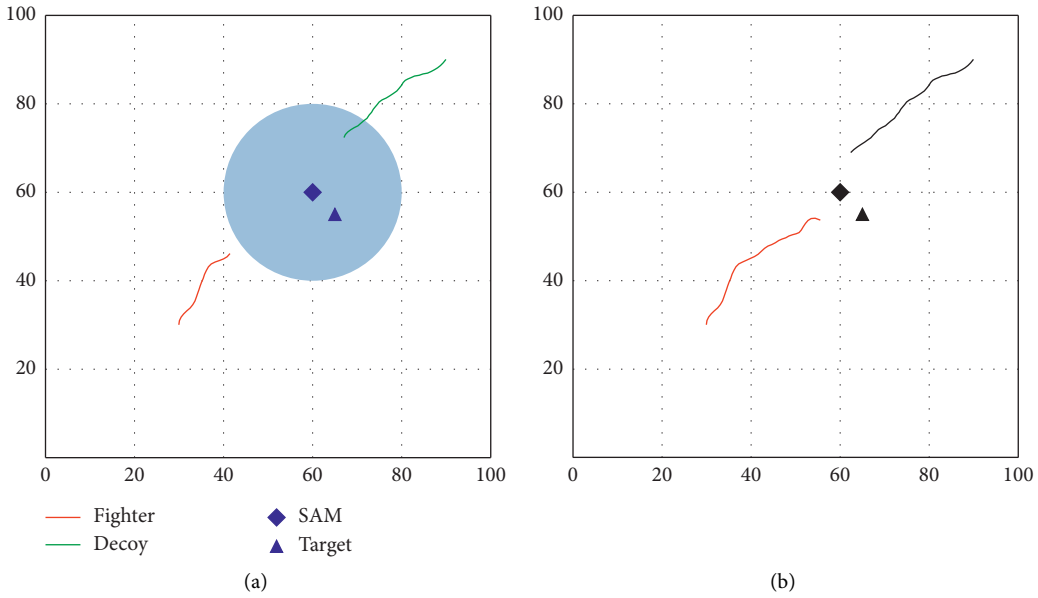

Figure 11: (a) Decoy entry early; (b) Sam tracks decoy, and fighter takes the opportunity to attack the Sam and target. When the Sam and target are destroyed, they turn black.

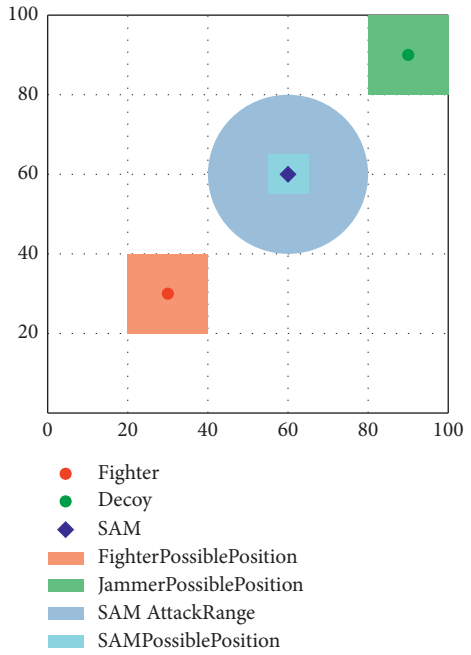

Figure 12: SEAD environment with different initial positions.

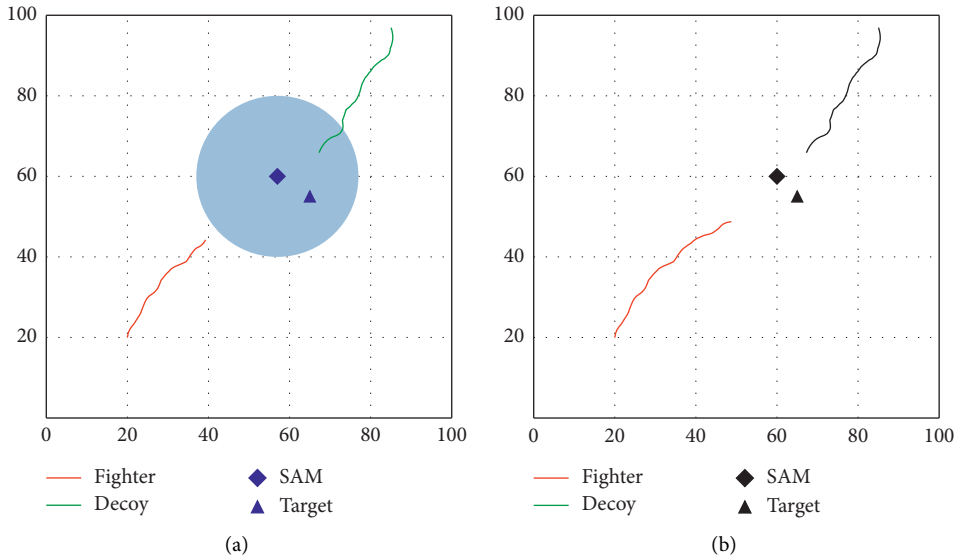

Figure 13: (a) Decoy enters early; (b) Sam tracks and attacks decoy, and fighter takes the opportunity to attack the Sam and target. When the decoy, Sam, and target are destroyed, they turn black.

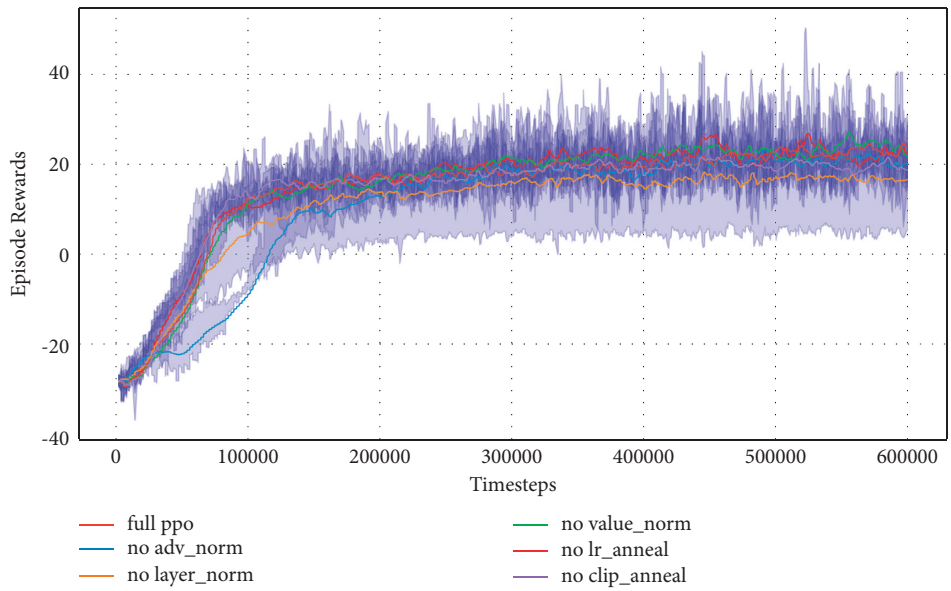

Figure 14: Learning-curve comparison of different tricks.

6. Conclusions

In this paper, we propose an end-to-end DRL-based UAV intelligent mission planning method. First, a SEAD mission is selected as the research object, and mission planning is described as a sequential decision-making problem. Then, the SEAD mission intelligent planning model based on the PPO algorithm is established, including the design of UAV state space, action space, and reward function. Three policy training tricks, namely domain randomization, maximizing policy entropy, and parameter sharing, are introduced, and the intelligent planning general architecture is constructed. Finally, two experiments analyses, robustness studies, and an ablation experiment are completed. We conclude that the intelligent planning model based on deep reinforcement learning, which adopts an end-to-end architecture and offline training and online inferencing, can adapt to the dynamic situation, and it is advanced and valuable. In future work, this end-to-end method will be extended to large-scale complex combat scenarios, and the problem of multi-agent cooperative planning will be studied in depth.

Data Availability

The data used to support the findings of this paper are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of the paper.

Authors' Contributions

Longfei Yue planned the work, completed the simulation experiment, and drafted the main part of the paper. Rennong Yang, Ying Zhang, and Lixin Yu contributed error analysis. Zhuangzhuang Wang contributed to setup type.

Acknowledgments

The work described in this paper is partially supported by the Nature Science Foundation of Shaanxi Provincem of China under Grant no. 2021JQ-370 and the National Natural Science Foundation of China under Grant no. 62106284.

References

[1] L. Shen, J. Chen, and N. Wang, "Overview of air vehicle mission planning techniques," *Acta Aeronautica et Astronautica Sinica*, vol. 35, no. 3, pp. 593–606, 2014.

[2] Joint mission planning system, "Joint mission planning system," 2020, https://www.globalsecurity.org/military/systems/aircraft/jmps.htm.

[3] C. Xin, Q. Luo, C. Wang, Z. Yan, and H. Wang, "Research on route planning based on improved ant colony algorithm," *Journal of Physics: Conference Series*, vol. 1820, no. 1, Article ID 012180, 2021.

[4] Y. Zhang, R. Yang, and J. L. Zuo, "Tactic maneuver planning of loft delivery of laser-guided bomb with no offset," *Systems Engineering and Electronics*, vol. 38, no. 5, pp. 1074–1080, 2016.

[5] Q. Qingfu Zhang and H. Hui Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[6] M. Darrah and W. Niland, "UAV cooperative task assignments for a SEAD mission using genetic algorithms," in *Proceedings of the AIAA Guidance, navigation & control conference & exhibit*, pp. 72–78, Keystone, America, August, 2006.

[7] H. Zhang, R. Yang, and J. Wu, "Research on multi-aircraft cooperative suppressing jamming embattling in electronic warfare planning," *Systems Engineering and Electronics*, vol. 39, pp. 542–548, 2017.

[8] W.-L. Shang, J. Chen, H. Bi, Y. Sui, Y. Chen, and H. Yu, "Impacts of COVID-19 pandemic on user behaviors and environmental benefits of bike sharing: a big-data analysis," *Applied Energy*, vol. 285, Article ID 116429, 2021.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[10] H. Bi, W.-L. Shang, Y. Chen, K. Wang, Q. Yu, and Y. Sui, "GIS aided sustainable urban road management with a unifying queueing and neural network model," *Applied Energy*, vol. 291, Article ID 116818, 2021.
[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, pp. 58-59, MIT Press, Cambridge, MA, USA, 2018.
[12] D. Silver, A. Huang, C. J. Maddison et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[13] D. Silver, J. Schrittwieser, K. Simonyan et al., "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[14] D. Silver, T. Hubert, and J. Schrittwieser, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," 2017, https://arxiv.org/abs/1712.01815.
[15] O. Vinyals, I. Babuschkin, and W. Czarnecki, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 1, pp. 1–5, 2019.
[16] J. Hwangbo, J. Lee, A. Dosovitskiy et al., "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, pp. 1–13, 2019.
[17] Y. Song, M. Steinweg, and E. Kaufmann, "Autonomous drone racing with deep reinforcement learning," 2021, https://arxiv.org/abs/2103.08624.
[18] A. Kendall, J. Hawke, and D. Janz, "Learning to drive in a day," in *Proceedings of the 2019 International Conference on Robotics and Automation*, pp. 8248–8254, Montreal, Canada, May, 2019.
[19] D. H. Ye, Z. Liu, and M. Sun, "Mastering complex control in MOBA games with deep reinforcement learning," 2019, https://arxiv.org/abs/1912.09729.
[20] H. Hu, X. Zhang, and X. Yan, "Solving a new 3d bin packing problem with deep reinforcement learning method," 2017, https://arxiv.org/abs/1708.05930.
[21] 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," *Naval Research Logistics*, vol. 49, no. 8, pp. 723–742, 2002.
[22] M. Haque, M. Egerstedt, and A. Rahmani, "Multilevel coalition formation strategy for suppression of enemy air defenses missions," *Journal of Aerospace Information Systems*, vol. 10, no. 6, pp. 287–296, 2013.
[23] W.-L. Shang, Y. Chen, and W. Y. Ochieng, "Resilience analysis of transport networks by combining variable message signs with agent-based day-to-day dynamic learning," *IEEE Access*, vol. 8, Article ID 104458, 2020.
[24] V. Mnih, K. Kavukcuoglu, and D. Silver, "Playing atari with deep reinforcement learning," 2013, https://arxiv.org/abs/1312.5602.
[25] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[26] T. Lillicrap, J. Hunt, and A. Pritzel, "Continuous control with deep reinforcement learning," 2015, https://arxiv.org/abs/1509.02971.
[27] J. Schulman, F. Wolski, and P. Dhariwal, "Proximal policy optimization algorithm," 2017, https://arxiv.org/abs/1707.06347.
[28] C. Berner, G. Brockman, and B. Chan, "Dota 2 with large scale deep reinforcement learning," 2019, https://arxiv.org/abs/1912.06680.
[29] D. H. Ye, G. Chen, and W. Zhang, "Towards playing full MOBA games with deep reinforcement learning," 2020, https://arxiv.org/abs/2011.12692.
[30] J. Schulman, S. Levine, and P. Moritz, "Trust region policy optimization," 2015, https://arxiv.org/abs/1502.05477.
[31] J. Schulman, P. Moritz, and S. Levine, "High-dimensional continuous control using generalized advantage estimation," 2015, https://arxiv.org/abs/1506.02438.
[32] J. Tobin, R. Fong, and A. Ray, "Domain randomization for transferring deep neural networks from simulation to real," in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 23–30, Vancouver, BC, Canada, September, 2017.
[33] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.