scientific reports

Check for updates

OPEN

Intelligent decision for joint operations based on improved proximal policy optimization

Chen Li, Wenhan Dong✉, Lei He, Ming Cai & Dafei Wang

To tackle challenges such as convergence difficulties and suboptimal performance in the application of reinforcement learning to intelligent decision-making for joint operations, this study introduces an enhanced decision-making approach for joint operations utilizing an improved Proximal Policy Optimization (PPO) algorithm. We propose a structured intelligent decision-making model designed to execute decision-making functions effectively. The strategy loss mechanism is improved by constraining the upper limit of the strategy loss function. Furthermore, a priority sampling mechanism, is developed to assess sample values, thereby enhancing the efficiency of sampling training. Additionally, a network structure facilitating distributed interaction and centralized learning is designed to expedite the training process. The proposed method is then applied to a joint operations simulation platform for intelligent decision-making. Simulation results demonstrate that our algorithm successfully addresses the aforementioned issues, enabling autonomous decisions based on battlefield dynamics, and ultimately leading to victory.

Keywords Joint operations, Military simulation, Intelligent decision-making, Reward function, Policy loss

In the contemporary domain of military research, joint operations have become a pivotal focus for military strategists globally. The inherent complexity within the decision-making landscape of joint operations presents significant challenges to their simulation and decision-making processes. Deep reinforcement learning (DRL) has exhibited exceptional decision-making capabilities in large-scale gaming and complex task environments. This study aims to explore the application of deep reinforcement learning in intelligent decision-making for joint combat. Considering the intricacy of real-world battlefield environments, direct application in such scenarios is currently impractical. Instead, we propose the utilization of a joint operations simulation platform as a surrogate for the actual joint operation battlefield. Initially, this research endeavors to employ a joint operations simulation platform. Should the outcomes prove sufficiently mature and concurrent advancements in battlefield modeling techniques occur, the findings could be translated for application in actual combat scenarios. The ultimate theoretical contributions of this study are anticipated to foster the evolution of joint warfare concepts and potentially give rise to new operational doctrines. These advancements could facilitate the broader application of deep reinforcement learning across various facets of warfare. The practical application of this research lies in aiding commanders in validating and innovating new tactical approaches. Additionally, it can be employed for preliminary research and simulated battlefield validation of weaponry and equipment. Hence, the objective of this research is to develop a generalized intelligent decision-making method for joint operations using reinforcement learning techniques, and to address the primary challenges associated with its implementation.

DRL is increasingly being leveraged to tackle a variety of military decision-making challenges. Hu et al.[1] introduced a novel algorithm, the dynamic quality replay method, designed to address maneuvering decision problems in Unmanned Combat Air Vehicle (UCAV) combat confrontations using DRL. This method was initially validated for its feasibility and effectiveness. Zhao et al.[2] proposed an end-to-end intelligent reconnaissance mission planning method based on DRL to assist reconnaissance UAVs in achieving efficient path planning in high-threat and high-density environments. Yuan et al.[3] developed a hybrid approach that integrates hierarchical target-guided learning with reinforcement learning to tackle the issue of fixed-wing UAVs' inability to generate autonomous evasive maneuvers in dynamic hostile environments, while also enabling them to intelligently evade air-to-air missiles. Li et al.[4] proposed an intelligent weapon target assignment model, RL4WTA, based on DRL to overcome challenges in modeling traditional weapon target assignment and limitations in search efficiency. This model has shown strong adaptability and computational efficiency in both small-scale and large-scale scenarios. Furthermore, numerous scholars have employed DRL technology to address core technical difficulties in close air combat[5–10], yielding satisfactory results. Similarly, DRL techniques have been applied

Air Force Engineering University, Xi'an 710038, China. ✉email: dongwenhan@sina.com

to naval warfare-related problems[11,12] and large-scale strategy game StarCraft II[13–15], which is very similar to battlefield simulation games, also with positive outcomes.

In addition, we explored the applicability of optimization algorithms, such as reinforcement learning algorithms, in complex problems. Numerous scholars have successfully employed optimization methods like Particle Swarm Optimization, CatBoost, and Meta-learning[16] to address decision-making issues in complex systems, including Multi-Agent Systems[17], vehicle scheduling[18], industrial load identification[19], and refined oil distribution[20]. In the realm of NFV (Network Function Virtualization), Sun et al.[21] proposed a BFS (Breadth-First Search)-based SFCDO (Service Function Chain Deployment Optimization) algorithm, which optimizes end-to-end delay and bandwidth resource consumption. In the field of microgrid scheduling, Zhang et al.[22] introduced a multi-objective optimization method based on PBI, utilizing a deep reinforcement elite learning strategy to manage frequency dynamics in microgrid scheduling. Zhang et al.[23] proposed a novel hierarchical RL approach, which effectively alleviates the challenges in searching for sub-goals within large target spaces by constraining the action space of higher layers within the k-step neighborhood of the current state. Experimental results demonstrate that incorporating adjacency constraints enhances the performance of existing HRL methods in both deterministic and stochastic environments. These studies illustrate the potential of optimization algorithms, such as RL algorithms, in various domains, offering new perspectives for resolving practical issues.

Although DRL has achieved successful applications in military decision-making and gaming, its integration into joint operation command and decision-making encounters several challenges. These challenges encompass: (1) Sparse rewards, a pervasive issue within combat environments in DRL applications; (2) the policy update step of the Proximal Policy Optimization (PPO) algorithm, employed in this study, is deemed unreasonable, resulting in instability during the learning process and protracted convergence; (3) the original sampling method of the algorithm, which relies on random sampling, lacks the adaptability to adjust the sampling strategy in response to the value of samples, leading to inefficient learning; and (4) the current structure of the training network, consisting solely of an actor and a single critic, fails to fully harness the multi-core and multi-threaded computing resources available in this research, thereby excessively prolonging training durations.

This study contributes to the field of joint operation command and decision-making utilizing DRL in the following ways: (1) We construct a joint operation network model based on reinforcement learning theory and propose a design principle for the reward system. This principle facilitates the creation of guiding rewards tailored to specific environments, thereby addressing the issue of sparse rewards. (2) We introduce an improved policy loss mechanism that restricts the upper limit of the strategy loss function, contributing to a stable learning process and accelerated convergence. (3) We design an enhanced sampling mechanism that ranks empirical data according to Generalized Advantage Estimation (GAE)[24] values and increases sampling weights for high-value samples, thereby improving sampling and training efficiency. (4) We develop a network architecture featuring distributed interaction and centralized learning, achieved by separating the training process into interaction and learning phases. By leveraging the server's multi-threading capabilities, we accelerate the interaction process, significantly reducing training time. (5) The feasibility of our model and the effectiveness of our algorithmic improvements are validated through experiments conducted in a reliable joint operation simulation environment. In the second section, we provide an overview of the fundamental algorithms used in our study. The third section details the construction of an intelligent decision model for joint operations and presents our improved algorithms. In the fourth section, we conduct ablation experiments and simulation experiments to assess the performance of our model and algorithms. The fifth and sixth section discusses the experimental results and concludes the study.

Background of RL and PPO

Reinforcement learning

Reinforcement learning is an approach that empowers an agent to learn and make progress through iterative trial-and-error interactions, aiming to maximize cumulative rewards. A Markov Decision Process (MDP) is commonly utilized to model the problem, represented by a quintuple (S, A, P, R, γ). Here, S denotes a finite set of states, encompassing the states of all combat units. A signifies a set of feasible actions. P is the state transition probability matrix, describing the likelihood of transitioning between states within the battlefield environment. R represents the reward function, quantifying the rewards derived from battle victories. The symbol γ denotes the discount factor, which accounts for the future rewards in the decision-making process.

The Q-Learning algorithm, a classical off-policy reinforcement learning method, has found extensive application in diverse military decision-making scenarios[25–27]. The iterative formula of the algorithm is expressed as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{1}$$

The Poilcy Gradient (PG) algorithm is a classical on-policy reinforcement learning algorithm. It updates the parameters of the policy by gradient ascent:$\theta \leftarrow \theta + \alpha * PG$. The gradient is computed as follows:

$$PG = \hat{E}_t \left[\nabla_\theta \log \pi_\theta(s_t, a_t) Q_t(s_t, a_t) \right] \tag{2}$$

The majority of other algorithms are innovative variations that have been derived from these two foundational methods.

PPO algorithm

The PPO algorithm is a DRL method rooted in the Poilcy Gradient. It represents a simplified yet enhanced version of the Trust Region Policy Optimization (TRPO) algorithm. PPO has been adopted as OpenAI's default

reinforcement learning algorithm, owing to its superior robustness and stability, while its performance is comparable to that of the TRPO algorithm. Therefore, this study employs PPO algorithm as the fundamental method to address the problem.

The TRPO algorithm employs the Kullback-Leibler (*KL*) divergence between the old and new policies to regulate the step size of the updating strategy within a trust region, addressing two critical issues: first, the risk of premature convergence to a local optimum due to excessively small step sizes during parameter updates along the policy gradient in the A-C framework; second, the challenge of achieving convergence during training when the step size is too large, potentially degrading the policy. Theoretical proofs demonstrate that this method can monotonically increase cumulative rewards and stabilize the policy improvement process. The objective function of its penalized form is expressed as follows:

$$\max_{\theta} \hat{E}_t \left[\frac{\pi_\theta(s_t, a_t)}{\pi_{\theta_{old}}(s_t, a_t)} \hat{A}_t - \beta KL\left[\pi_{\theta_{old}}(\cdot\,|s_t), \pi_\theta(\cdot\,|s_t)\right] \right] \tag{3}$$

Where β is the penalty coefficient against *KL* dispersion; $\pi_\theta(s_t,a_t)$ and $\pi_{\theta_{old}}(s_t,a_t)$ are the probabilities that the new and the old strategies will take an action at time *t* and state *s* respectively; and $KL\left[\pi_{\theta_{old}}(\cdot\,|s_t), \pi_\theta(\cdot\,|s_t)\right]$is the *KL* dispersion between the new and old strategies.

However, TRPO computes the divergence between the old and new policies using the second-order Hessian matrix, which can be computationally intensive. Subsequently, a more computationally efficient variant of TRPO was proposed, designated as the PPO algorithm, which utilizes first-order derivatives. This first-order approach is more practical, computationally efficient, and easier to implement. Furthermore, it maintains performance comparable to that of TRPO. The penalized form of the PPO's objective function is expressed as follows:

$$L(\theta)^{CLIP} = \hat{E}_t \left[\min\left(r_t(\theta)\hat{A}_t, clip\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t \right) \right] \tag{4}$$

Where $r_t(\theta)$ denotes the ratio of the probability of selecting a specific action between the previously utilized strategy and the newly implemented one. The calculation is as follows:

$$r_t(\theta) = \frac{\pi_\theta\left(s_t, a_t\right)}{\pi_{\theta_{old}}\left(s_t, a_t\right)} \tag{5}$$

Where ε is a hyperparameter that typically assumes a value of 0.2; the $clip\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)$ function serves to clip the ratio of the probabilities of the old and new strategies to the range $[1-\varepsilon, 1+\varepsilon]$, which is employed to limit the magnitude of the updating strategy.

It is widely recognized that the PPO algorithm excels in terms of parameter tuning simplicity and performance. However, the majority of research has focused on its application across various domains, with relatively fewer studies dedicated to the algorithm itself and its enhancements. Such research can be broadly categorized into two areas. One area of research is presented in Table 1, while the other area of research is presented in Table 2. These studies collectively demonstrate that while the clip mechanism is critical for PPO computations, the optimization techniques in code implementation are equally essential. Furthermore, improvements in PPO's network architecture represent key breakthroughs, enabling its application in diverse environments and enhancing training efficiency.

Method

This section is organized around four primary components: the observation space (inputs to the model), the action space (outputs of the model), the reward function, and the decision-making model.

Observation space

In military engagements, the information accessible to both parties can be classified into two types: The first type comprises the status information of our own combat units, including details such as the battle timestamp, remaining fuel, longitude, latitude, altitude, and weapon specifics. The second type involves the status

Year	Research content	Conclusions
2020[28]	Is the clip mechanism in the PPO algorithm important for its performance?	Experiments have proven that arbitrarily amplifying the clip range or removing the clip mechanism leads to learning failure, indicating that the clip mechanism plays a crucial role in model learning
2020[29]	In the implementation of the Proximal Policy Optimization (PPO) algorithm, the optimization techniques employed may appear to be of secondary importance, but are they indeed crucial for the algorithm's performance?	The experimental results indicate that the superior performance of PPO over TRPO is attributable, to a significant extent, to these code-level optimization techniques
2021[30]	By adaptively tuning the hyperparameters of the clip mechanism, is it possible to control the distance between the new and old policies as needed, thereby enhancing the algorithm's performance?	Experimental findings suggest that this modification is capable of overcoming the issue of local optima in specific domains, thereby enhancing the efficiency of the algorithm
2024[31]	Does the PPO-clip algorithm possess global optimality?	This marks the first instance of obtaining a globally convergent solution for the PPO-clip algorithm, thereby providing theoretical support for the PPO-clip approach

Table 1. Research on the clip mechanism and optimization techniques for PPO.

Year	Research content	Conclusions
2022[32]	Addressing the Issue of Inaccurate Value Function Estimation in the Classic Actor-Critic Architecture	By incorporating the policy network into the value update process, authors propose PPO-PF (PPO with Policy Feedback). Experimental results indicate that PPO-PF outperforms PPO in terms of convergence speed and reward variance
2023[33]	To address the issue of low computational efficiency in the model, a novel PPO training framework has been designed, leveraging the multi-threaded distributed computing capabilities of computers, and applied to the energy management strategy for fuel cell systems	The integration of distributed computation into network architectures is a widespread strategy in the application of RL algorithms, aligning with the last innovative objective of this paper, albeit employing distinct technical methodologies. Experimental results demonstrate that this approach effectively enhances training efficiency and contributes to energy conservation and prolonged lifespan of fuel cell systems
2024[34]	To enhance the performance of the PPO algorithm in robotic autonomous navigation, the authors have introduced an enhanced neural network architecture for the PPO algorithm	The optimization of this network architecture constitutes an adaptive enhancement within the domain of Reinforcement Learning (RL) applications targeting obstacle avoidance. Experimental results indicate that this improvement exhibits commendable performance in both obstacle-populated and obstacle-free environments, making a significant contribution to the advancement of robotic autonomous navigation in complex settings
2024[35]	To address the mission planning issue of Unmanned Aerial Vehicles (UAVs) within the Space-Air-Ground integrated network, the authors developed a task scheduling algorithm founded on Proportional Fairness-Aware Auction with Proximal Policy Optimization (PFAPPO), employing a distributed Proximal Policy Optimization (PPO) algorithm for intelligent offloading decisions	The optimization of this network architecture represents an adaptive improvement in the application of Reinforcement Learning (RL) to task planning under specific conditions. Experiments demonstrate that this intelligent decision-making method, which decouples the task scheduling process into resource allocation and task offloading decisions, outperforms existing algorithms in terms of system profitability, load balancing, and fairness

Table 2. Research on the architecture improvements for PPO.

information of enemy combat units, acquired through intelligence sources, and encompasses details such as unit type, military classification (whether enemy, friendly, or civilian), longitude, latitude, altitude, and weapon specifics. All state information is normalized to ensure a consistent range of variability, thereby enhancing the neural network's sensitivity to the input data. This normalization is critical due to the inherent variations in scale and range among the different types of information.

Action space

In the context of the Markov Decision Process (MDP) framework for intelligent decision-making in joint military operations, the selection of an appropriate action space is essential to enable the agent to explore a wide range of tactical options. The action space can be defined using various structures, such as gym.spaces.box, gym.spaces.discrete, or gym.spaces.multiDiscrete. The dimensionality of the action space is determined by the number of combat unit types involved, where each dimension represents the specific tasks that can be executed by the corresponding unit type. For example, combat units like fighter aircraft may be capable of performing multiple tasks, such as reinforcement, area patrol, and bomber escort. Each of these tasks is represented as a distinct dimension within the action space. However, the agent is constrained to selecting only one specific task for each combat unit type during each decision-making epoch, which allows for the issuance of diverse tactical commands across different types of combat units. Furthermore, within this framework, each combat unit retains the flexibility to either opt for a new action or to continue with its previous action.

Reward function

The application of DRL to complex problem-solving frequently encounters the significant challenge of sparse rewards. The core issue resides in the reward function's inability to effectively guide the parameter updates of the agent during the learning and training phases of the reinforcement learning model. The application of DRL in this paper underscores the considerable negative impact of sparse rewards. In the environment described in Sect. 4, the red agent's objective is to attack and destroy the blue agent's two command posts, while the blue agent aims to defend and protect its own command posts. Consequently, the joint combat environment does not provide intermediate rewards during the simulation. That is, there are no rewards for each step of the deduction during the learning process. Instead, rewards are only granted to the red agent upon the destruction of the blue agent's two command posts, or to the blue agent upon successfully defending its two command posts. This issue hinders the agent's ability to improve its decision-making through learning from rewards. Furthermore, the observation dimension of the intelligent decision-making problem in joint operations simulation typically comprises several hundred dimensions, and the state transition relationship of the environment are complex. This complexity makes it challenging for the agent to acquire a successful strategy within a limited time and number of steps, relying on sparse rewards[36].

To address the aforementioned challenges, this paper propose a method for designing the reward function tailored to the task environment. The reward function is bifurcated into two components: result reward and process reward. The result reward represents the sparse reward determined by the task's outcome, while the process reward is a continuous reward obtained during model training, facilitating short-term bootstrapping for the intelligent agent's training. The specific design of the result and process rewards can be adapted flexibly to the task environment, and while the details are not elaborated here, a design method that effectively balances the result and process reward functions is emphasized. The principle of this balanced design is to ensure that the process rewards do not surpass the discounted cumulative result rewards, thereby maintaining the agent's motivation to pursue the ultimate outcome rather than solely focusing on immediate process rewards. In other words, it is necessary to ensure that the return received by an agent that only receives result reward are higher than those received by an agent that only receives process rewards. This principle is expressed as follows:

$$\sum {}_{t=0}^{T}\gamma^{t}P(t) < \gamma^{T}R \tag{6}$$

Where $P(t)$ represents the process reward obtained by the agent at $step = t$, R denotes the result reward received by the agent, and T represents the number of steps until the environment terminates.

Our objective is to maximize the process reward. Here, assuming $P(t)$ is the maximum process-based reward at any given moment, denoted as P_{max}, Eq. (6) becomes:

$$P_{max}\sum {}_{t=0}^{T}\gamma^{t} < \gamma^{T}R \tag{7}$$

According to the formula for the sum of a geometric series, the following expression can be derived:

$$P_{max}\frac{1-\gamma^{T+1}}{1-\gamma} < \gamma^{T}R \tag{8}$$

Further calculations yield:

$$P_{max} < \frac{1-\gamma}{1-\gamma^{T+1}}\gamma^{T}R \tag{9}$$

Taking the experiment in Sect. 4 as an example, the principle for allocating process rewards is as follows: With a maximum decision step count of 1200 and a discount factor γ of 0.99, the process reward for each step, P_{max}, is less than 5.78×10^(-8) times the result reward. This ensures compliance with the principle of balanced reward design.

Intelligent decision model for joint operations

This subsection delineates the intelligent decision model for joint operations, which comprises several integral components: the state space, action space, reward function, network architecture, and training methodologies. Collectively, these elements form the comprehensive decision model. Figure 1 depicts the training process of this model.

The agent generates training data by interacting with the wargaming environment, which provides the current state observations, denoted as s_t. Upon receiving these observations, the agent makes strategic decisions and generates an action a_t. The environment then executes this action a_t, and produces the subsequent state value, s_{t+1}, which serves as the foundation for the next iteration of the loop. This process describes the interaction mechanism. Subsequently, the training process to develop a superior strategy is outlined. The environment generates a corresponding reward r_t, based on the outcome of the current action's execution. The PPO algorithm gathers tuples of (s_t,a_t,r_t,s_{t+1}) during interactions to train the policy network, which is then updated after the training process. PPO is designed to continuously learn and refine the optimal policy network until a policy network and decision model that fulfill the requirements are achieved.

PPO is based on the Actor-Critic framework. In this paper, the structures of the actor and critic networks are fundamentally similar. The enhancements to the PPO algorithm can be broadly categorized into three types.

Improved policy loss mechanism

The policy loss function of the PPO algorithm is designed as follows:

$$L(\theta) = \hat{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, clip\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right)\right] \tag{10}$$

Where $r_t(\theta)$ denotes the ratio of the probability of selecting a specific action between the previous and current strategies. The significance of the *clip()* function can be expressed as follows:

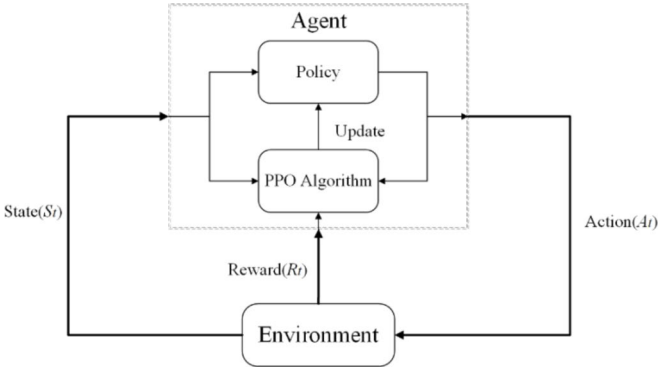


Fig. 1. Training process of intelligent decision models.

$$\mathrm{clip}(x,m,n)=\begin{cases} m & x<m \\ n & x>n \\ x & m<x<n \end{cases} \tag{11}$$

The advantage function, denoted as $\widehat{A}_t$, represents the advantage of a state-action pair (s_t,a_t). Utilizing the advantage function offers two primary benefits: first, it enhances learning efficiency and stabilizes the learning process; second, empirical evidence suggests that the advantage function aids in reducing policy variance and alleviating overfitting. When $A>0$, it signifies that the state-action pair (s_t,a_t) is more advantageous compared to the average, warranting an increase in its selection probability. Conversely, when $A<0$, it indicates that the state-action pair (s_t,a_t) is less advantageous compared to the average, and thus its selection probability should be decreased. Figure 2a and b depict the policy loss function $L(\theta)^{CLIP}$ for the cases when $A>0$ and $A<0$, respectively.

As depicted in Fig. 2b, when $A<0$ and $r_t(\theta)\gg 1$, the policy loss function $L(\theta)^{CLIP}$ increases without bound as $r_t(\theta)$ increases. This leads to the policy loss function lacking an upper bound, causing the variance of the strategy loss to escalate and resulting in instability during the learning process.

To address the instability in such issues, this paper proposes modifying the original policy loss function by defining it as a segmented function. The function remains unaltered for $A>0$. However, for $A<0$, it is modified as shown in Fig. 2c. This modification ensures that the policy loss function is bounded even when $A<0$, thereby enhancing the stability of the policy learning process. The modified policy loss function is expressed as follows:

$$L(\theta)^{CLIP}=\begin{cases} \hat{E}_t\left[\min\left(r_t(\theta)\hat{A}_t,\mathrm{clip}\left(r_t(\theta),1-\varepsilon,1+\varepsilon\right)\hat{A}_t\right)\right] & \hat{A}_t\geqslant 0 \\ \hat{E}_t\left[\min\left(\mathrm{clip}\left(r_t(\theta),1-\varepsilon,2\right)\hat{A}_t,\mathrm{clip}\left(r_t(\theta),1-\varepsilon,1+\varepsilon\right)\hat{A}_t\right)\right] & \hat{A}_t<0 \end{cases} \tag{12}$$

For the sake of simplicity, the algorithm's name is altered in this paper to facilitate its representation. The algorithm is named $l^{org1}(r_t(\theta))$ when $A\geq 0$, and $l^{org2}(r_t(\theta))$ when $A<0$ before modification. After the modification, the algorithm remains unchanged when $A\geq 0$, and the algorithm is named $l^{clip}(r_t(\theta))$ when $A<0$.

$$\begin{cases} l^{org1}\left(r_t(\theta)\right)=\min\left(r_t(\theta)\hat{A}_t,\mathrm{clip}\left(r_t(\theta),1-\varepsilon,1+\varepsilon\right)\hat{A}_t\right) & \hat{A}_t\geqslant 0 \\ l^{org2}\left(r_t(\theta)\right)=\min\left(r_t(\theta)\hat{A}_t,\mathrm{clip}\left(r_t(\theta),1-\varepsilon,1+\varepsilon\right)\hat{A}_t\right) & \hat{A}_t<0 \\ l^{clip}\left(r_t(\theta)\right)=\min\left(\mathrm{clip}\left(r_t(\theta),1-\varepsilon,2\right)\hat{A}_t,\mathrm{clip}\left(r_t(\theta),1-\varepsilon,1+\varepsilon\right)\hat{A}_t\right) & \hat{A}_t<0 \end{cases} \tag{13}$$

Improved sampling mechanism

The conventional PPO algorithm employs random sampling from the experience pool; however, this method may not optimally leverage samples with high values, which can hinder the convergence process. In contrast, off-policy algorithms often incorporate the Prioritized Experience Replay (PER) technique to augment the efficiency of experience learning. This method utilizes Temporal Difference (TD) error to evaluate the significance of samples for training. Samples with a larger *TD-error* indicate a greater divergence between the model and the sample, thereby assigning higher significance for training. Conversely, samples with a smaller *TD-error* suggest a closer alignment between the model and the sample, resulting in lower training significance. The *TD-error* is computed using the following equation:

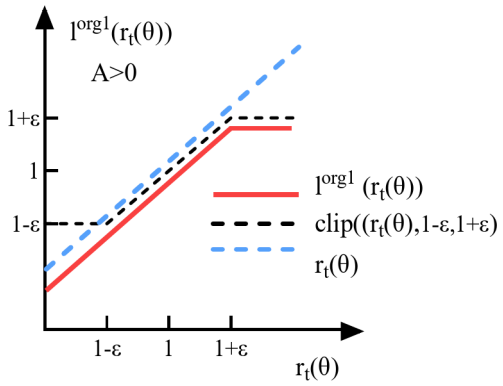
$$TD-error=r+\gamma Q_{target}\left(s',\arg\max Q\left(s',a'\right)\right)-Q(s,a) \tag{14}$$

The PPO algorithm employs the importance sampling technique, which renders the process of PPO algorithm sampling experience from the experience pool analogous to that of an off-policy algorithm. As a result, the PPO algorithm can adopt the experience sampling methods used by off-policy algorithms, such as Deep Q-Network (DQN). However, using *TD-error* to evaluate the significance of experiences has its limitations. *TD-error* reflects the importance of experience primarily from the perspective of network data, rather than directly indicating the advantages of the experience for the environment's operation. Moreover, *TD-error* can be noisy due to the randomness inherent in environmental rewards. This noise can introduce bias into the estimation of the experience's merit.

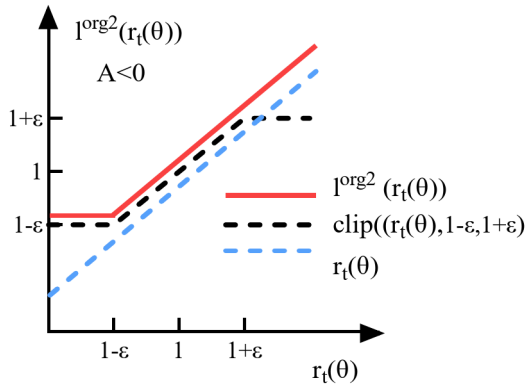
Current advancements and applications of PER primarily revolve around the prioritization criteria for ranking experiences. The original PER utilizes *TD-error* as its prioritization criterion. Sun et al.[37] employed the similarity between past experience states and the current state as the prioritization criterion, proposing the Attentive Experience Replay (AER) algorithm. Hu et al.[38] also used state similarity as a criterion but implemented a distinct function for calculating this similarity. Dong et al.[39] introduced curiosity into PER to more efficiently utilize experience samples.

While these methods may not universally surpass the original PER across all environments, their specialized prioritization criteria undoubtedly confer advantages compared to the original PER in certain specialized domains. Achieving victory in joint operations environments, which entails maintaining an optimal battlefield posture, aligns with the PPO algorithm's intrinsic objective of improving current strategies over previous ones. Consequently, selecting the Generalized Advantage Estimation (GAE)[24] dominance value as a prioritization measure is judicious within the context of these environments.

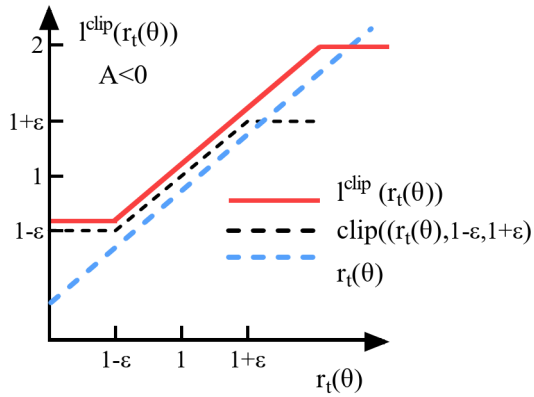
To address these issues, this paper proposes a prioritized sampling mechanism. This mechanism utilizes the GAE of each experience to assess its utilization merit, providing a more accurate estimation than TD-error. The GAE advantage value of the experience is calculated as follows:



(a) Policy Loss when(or at) $A>0$



(b) Policy Loss when(or at) $A<0$



(c) improved Policy Loss when(or at) $A<0$

Fig. 2. Policy Loss Function Before and After Modification. The dashed black line represents the value of $L(\theta)^{CLIP}$ after being processed by the clip function. The dashed blue line represents the value of $r_t(\theta)*A$. The solid red line represents the value of the modified policy loss function.

$$\hat{A}_t = TD - error_t + \gamma TD - error_{t+1} + \cdots + \gamma^{T-t+1} TD - error_{T-1} \tag{15}$$

The variable T represents the time at the end of the round.

The prioritized sampling mechanism introduced in this paper modifies the sampling process in two ways to achieve the intended objectives. Firstly, the sequence of experiences is reordered, sorting the outcomes of the interactions from largest to smallest based on the absolute value of the GAE. Secondly, the sampling weights are adjusted to increase the probability of sampling both positive and negative experiences with a high GAE value. The sampling weights for the experiences are computed using the following formula:

$$P(m) = \frac{P_m^\alpha}{\sum_k P_k^\alpha} \tag{16}$$

In the equation, m represents the ordinal number of the sequence after adjustment; P_m denotes the sampling probability of the experience sample with ordering m; α is a parameter that adjusts the strength of the prioritized sampling. When $\alpha = 0$, the sampling probability for each experience is equal, leading to uniform sampling. The prioritized sampling mechanism becomes more pronounced as α increases. P_m is calculated as follows:

$$P_m = \frac{1}{m} \tag{17}$$

The prioritized sampling mechanism introduced in this study confers two principal benefits. Firstly, it increases the likelihood of sampling instances with high GAE values, allowing the learning model to effectively utilize samples with extreme reward values, both positive and negative. This strategy accelerates convergence during training. Secondly, the mechanism demonstrates improved performance robustness. The proposed approach prevents the overuse of samples with large GAE values.

Efficient network architecture

We observed that the agent initially engages in a standard round of interaction with the environment, subsequently, the algorithm utilizes the experiences generated from this interaction for learning, training, and updating the network parameters, following which, the agent continues to interact with the environment for the subsequent round. It is evident that the processes of interaction and learning, training are executed separately and independently. Due to the considerable time required for the agent to interact with the environment to collect experiences, the training process in this paper is prolonged. Consequently, this study isolates the most time-consuming process (intelligent body-environment interaction) for parallel processing, thereby maintaining the algorithmic structure while accelerating the learning and training process.

This paper builds upon the insights from Mnih et al.[40] to design a network structure that enables distributed interaction, centralized learning, and training. The proposed structure is depicted in Fig. 3. Players are tasked with interacting with the environment and gathering experiences. The learner is responsible for leveraging the experience data collected by the players to derive a high-performing policy network through learning and training. Subsequently, the network parameters of all players are updated with the latest parameters of the policy network. Players then re-engage with the environment, accumulating experience data under the new policy network parameters. This cycle continues until the training process reaches a convergence point.

It is of paramount importance that the specific implementation be mindful of the following points:

1. A switch control mechanism for managing interaction and learning events is essential. When players interact with the environment to gather data, the learning event state is set to inactive. Conversely, during the learner's learning phase from accumulated experiences, the interaction event state is deactivated. This cycle is repeated to alternate between interaction and learning phases.
2. While multiple players share the same learner, there is no direct communication among the players. Each player independently transmits the data they have collected to the learner.
3. To ensure that the learner utilizes experiences generated by a consistent strategy during each training session, all players must interact with the same exploration noise. Additionally, the Stochastic Differential Equation (SDE) exploration mechanism should distribute the noise uniformly to all players following each interaction event.

Details of the algorithm implementation

The implementation nuances of the PPO algorithm significantly affect its performance, despite some details being mentioned merely as auxiliary techniques in the literature, and others not being explicitly stated. To

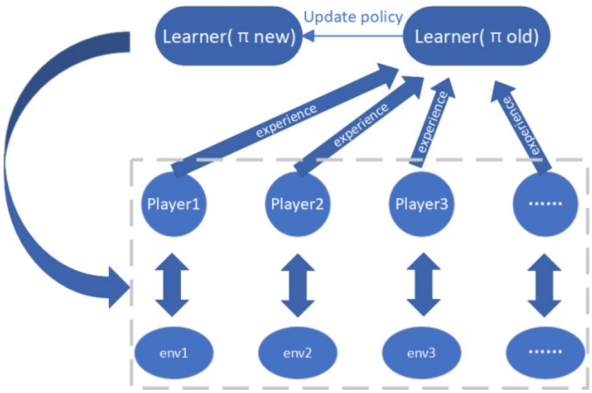


Fig. 3. The network architecture for distributed interaction, centralized learning and training.

Hyper parameter	Value
γ	0.99
GAE_λ	0.95
n_epochs	10
n_steps	512
batch size	128
clip_range	0.2
epoch	10
optimizer	Adam
actor learning rate	0.001
critic learning rate	0.001
activate function	tanh
actor architecture	(540,540,6)
critic architecture	(540,540,1)

Table 3. Setting of hyper parameters.

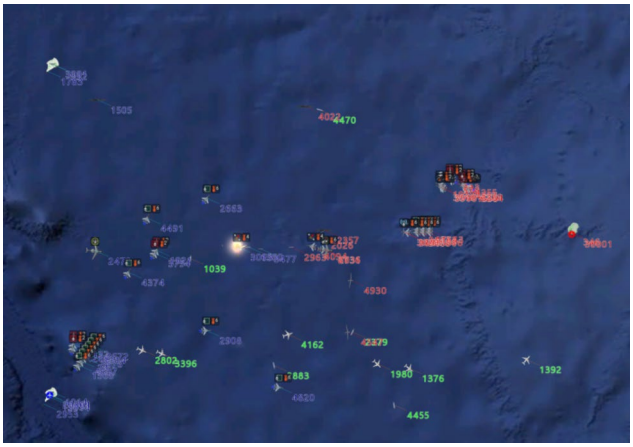


Fig. 4. Battlefield background schematic.

elucidate the impact of the methods proposed in this paper, we clarify the implementation details of the original PPO algorithm employed:

1. GAE: Although the original PPO algorithm does not elaborate on the specifics of advantage estimation, this paper utilizes GAE to compute the advantage values in its implementation of the PPO algorithm.
2. Mini-batch Updates: When the PPO algorithm processes empirical data, to ensure training stability and prevent performance degradation due to excessive policy updates, fixed-size data are sampled, then divided into smaller batches for gradient computation.
3. Reward Clipping: The reward function design incorporates reward clipping within a fixed range. This control mechanism regulates the guidance provided by various kinds of rewards to the intelligent agent.
4. Observation Normalization: Given the varying scale proportions of the original state input data, preprocessing is conducted to normalize the state input data. This adjustment ensures that the data range facilitates easier learning for the model.

Experiments

This section reports on the simulation outcomes, which were conducted to evaluate the feasibility and practicality of the intelligent decision-making method proposed in this study. The simulation took place within the environment of a joint operations simulation platform.

The experimental hardware is a Dell tower server equipped with a Nvidia 3090Ti graphics card. Each round of the battlefield simulation commenced from an identical initial scenario. The hyper-parameter configurations are detailed in Table 3. As shown in Table 1, this paper employs a single-hidden-layer MLPs, considering computational costs. Five independent experiments were conducted, employing the four methods described in Sect. 4.1.The state space is defined as gym.spaces.box with a latitude of 540. The action space is specified as gym.spaces.multiDiscrete, with the configuration [6,4,4,4,4,4,3], denoting the number of executable actions for fighters, bombers, jammers, AWACS, UAVs, and frigates, respectively. The maximum duration for each simulation is set at 2.5 h. The battlefield background information is provided as Fig. 4: In the simulated scenario,

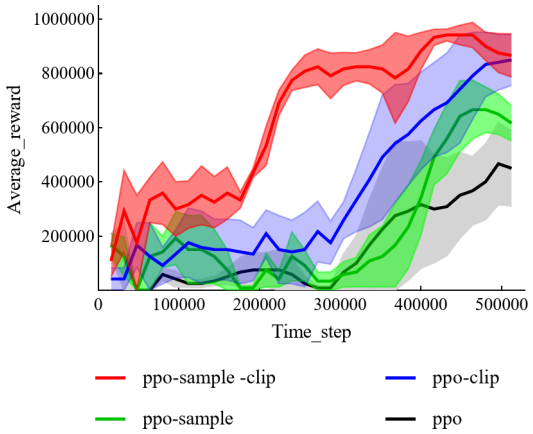


Fig. 5. Comparison of mean rewards in ablation study.

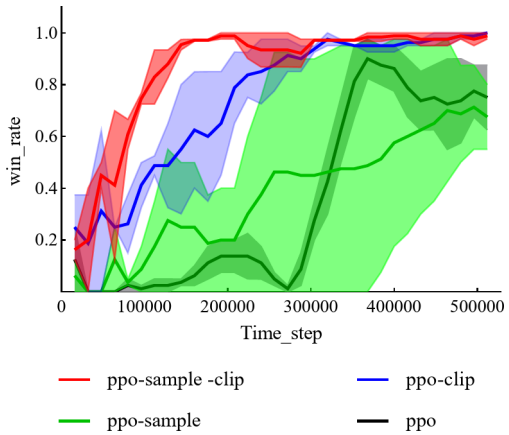


Fig. 6. Comparison of winning percentage for ablation experiments.

the Blue side has maintained control over the island previously held by the Red side for an extended duration. To reassert sovereignty and regain possession of the island, the Red side mobilizes its air and naval forces to launch strikes against key Blue side targets on the island. The objective of the Blue forces is to defend the island and two command posts, leveraging the combined firepower of their ground, sea, and air forces. Conversely, the Red forces aim to penetrate the Blue side's defensive network and destroy the enemy's command posts through a coordinated sea and air assault, supported by additional forces.The Blue side has deployed seven types of forces, including bombers, AWACS, fighters, destroyers, ground radar, ground guidance battalion, and airports, totaling 30 combat units. Similarly, the Red side has mustered seven types of forces, comprising bombers, AWACS, unmanned reconnaissance aircraft, electronic warfare aircraft, fighters, destroyers, ground radar, and airports, amounting to 45 combat units. The complex force formations is designed to execute a variety of tasks, such as reconnaissance, early warning, assault, jamming, escorting, air defense, and ground defense.

Ablation study

This section details the experimental setup, which involves the application of four distinct methodologies. The label **ppo** refers to the use of the standard PPO algorithm. The term **ppo-sample** denotes the application of the PPO algorithm with an enhanced sampling mechanism. **ppo-clip** signifies the use of the PPO algorithm with an improved policy loss mechanism. Lastly, **ppo-sample-clip** represents the application of the PPO algorithm with both an enhanced sampling mechanism and an improved policy loss mechanism. The experiment includes an evaluation that records the outcomes of policy evaluation at predetermined intervals, encompassing metrics such as average reward and winning rate. These metrics are derived statistically from the results of training conducted using each method over five iterations. Figures 5 and 6 illustrate the average reward and winning rate, respectively, depicted by solid lines, with the shaded areas indicating the standard deviation. The program captures real-time changes in key parameters, which are subsequently utilized to evaluate the algorithm's performance in relation to these parameters.

Average reward

Figure 5 illustrates the evolution of the average reward for each method throughout the training process. Initially, all methods exhibit low reward values during the pre-training phase. This is attributed to the agents' initial stage of trial-and-error learning, where they are accumulating experience without having yet acquired effective strategies. Subsequently, the reward values for all methods show significant improvement, suggesting that the reward design effectively guides agents in learning decision-making strategies. In the later stages, the reward values for all methods tend to stabilize or slightly decline, indicating that the agents have reached a point where they are unable to discover further strategies to enhance the reward value. This suggests that the training has converged.

Upon analyzing the speed of reward enhancement and the magnitude of the final reward, it is evident that the **ppo-sample-clip** method outperforms the other three methods in terms of both reward boosting speed and final reward value. The performance of **ppo-clip** is the closest to **ppo-sample-clip**, yet its relatively weaker exploration capability results in a slower reward enhancement rate. The **ppo-sample** method exhibits the most frequent fluctuations in reward value, indicating a strong ability to explore rewards. Although **ppo** demonstrates a slower initial reward enhancement rate, it maintains a robust rate in the later stages. Overall, **ppo-sample-clip** and **ppo-clip** are superior in terms of reward enhancement speed, while **ppo-sample** is the least stable in this regard.

Average winning percentage

It is important to note that a high reward value does not inherently correlate with a high probability of winning, and vice versa. A high reward value may indicate that the agent has learned a strategy that optimizes reward maximization. To accurately evaluate the performance of the different methods, it is essential to consider the winning percentages. To further validate the effectiveness of the enhancements proposed for the PPO algorithm in this study, Fig. 6 depicts the progression of the mean winning rate for each method as the step size varies.

Overall, during the pre-training phase, the success rate of the agent trained using each method shows an upward trend with the increase in the number of training steps. This suggests that the agents are becoming more adept. However, all methods tend to stabilize or slightly decline in the post-training period, indicating that the agents have reached a learning plateau, implying that further training is ineffective.

The performance of the four methods is assessed based on learning efficiency, final win rate, and standard deviation. In the case of **ppo**, the results indicate that the original PPO algorithm demonstrates the slowest learning speed, with an uptick in the winning rate observed at 270,000 steps. Its final winning rate is slightly higher than that of the **ppo-sample**. The smaller variance suggests a stable learning process. For **ppo-sample**, the learning efficiency exceeds that of **ppo** in the initial stages, but it progresses erratically, leading to a lower final result for **ppo-sample** compared to **ppo**. The large standard deviation implies that while **ppo** may enhance the agent's short-term exploration efficiency, it results in a less stable learning process over time. Regarding **ppo-clip**, both the learning efficiency and the final winning percentage surpass those of both **ppo** and **ppo-sample**. The stable growth and reduced standard deviation indicate that even using **ppo-clip** independently can yield favorable outcomes. In the case of **ppo-sample-clip**, the winning percentage of **ppo-sample-clip** increases at a significantly faster rate than the other three methods. The final win rate is the highest, and the standard deviation is small. This combination enhances the agent's exploration efficiency through **ppo-sample** and improves the stability of the learning process through **ppo-clip**, resulting in a more effective and efficient learning process and superior final outcomes compared to using **ppo-sample** or **ppo-clip** alone.

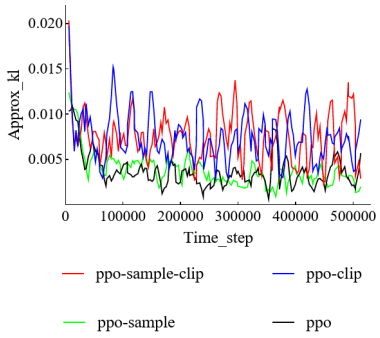
Parameters in training

The complexity of the operational environment gives rise to a multitude of factors that influence the experimental outcomes. Consequently, evaluating the algorithms based solely on evaluation results is inadequate. To address this, the experiments capture significant parameters of the four algorithms at regular intervals. This section delves into the analysis of three key parameters: *approx_kl*, *clip_fraction*, and *entropy_loss*, aiming to gain a more profound understanding of the performance of the proposed algorithms.

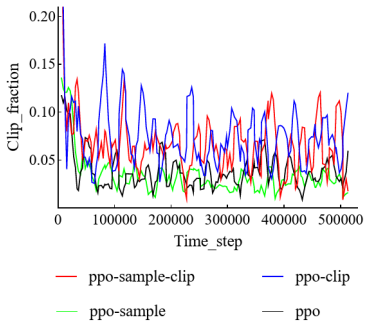
Approx_kl is utilized to quantify the divergence between the previous and new policies. The clip fraction represents the proportion of the surrogate loss that is truncated. A higher *entropy_loss* signifies a more uniform probability distribution of the actor's policy choices, implying a greater degree of stochasticity in the strategy. As depicted in Fig. 7a and b, the **ppo-sample-clip** and **ppo-clip** methods exhibit higher values for the *approx_kl* and *clip_fraction* parameters compared to the other two methods. This indicates that the **ppo-sample-clip** and **ppo-clip** methods adopt a larger learning step size, enabling them to implement more substantial strategy updates. A comparison of the rewards and win rate data reveals that the learning speed of the former two methods significantly surpasses that of the latter two. Conversely, the slower learning speed of the latter two methods can be attributed to their smaller learning and strategy update step sizes. Figure 7c shows that the former two methods exhibit lower *entropy loss*, suggesting that their strategies are trained to be more deterministic. This indicates a higher level of confidence in decision-making compared to random action selection. A comparison of the rewards and win rate data reveals that the latter two methods demonstrated lower performance in the early stages. Initially, there was a lack of effective decision-making strategies, resulting in random actions. Consequently, the initial stages were characterized by trial-and-error, with the agents failing to secure rewards or victories in the stage.

Comparison of time consuming

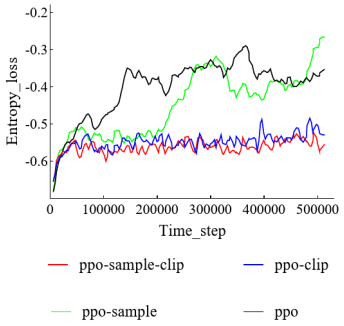
This paper provides a comparison of the time required to train a neural network across 10,000, 100,000, and 1,000,000 timesteps within the same environment, utilizing two distinct network structures: the original network



(a) *Approx_kl*



(b) *Clip_fraction*



(c) *Entropy_loss*

Fig. 7. The variations in the three parameters including approx_kl, clip_fraction, and entropy_lose during the training process of four algorithms.

	Network structure	
Timesteps	**New network structure**	**Old network structure**
Timesteps = 10,000	206.16s	8359.2s
Timesteps = 100,000	2593.8s	97891.2s
Timesteps = 1,000,000	46620.0s	1000957.2s

Table 4. Comparison of time consumption between the old and new network structures.

structure and the distributed interaction, centralized learning network structure. Table 4 presents a comparison of the time consumed, which fully illustrates the efficiency advantages of the proposed network structure.

In conclusion, the comprehensive analysis of average reward, winning percentage, training parameters, and time consumption underscores the efficacy of the proposed method in enhancing intelligent decision-making within joint operations simulations. The introduction of an enhanced sampling mechanism, an optimized policy loss mechanism, and an enhanced network structure, as detailed in this paper, markedly enhance the capability

and efficiency of the PPO algorithm in facilitating intelligent decision-making for joint operations simulation environments.

Validation in public environment

To assess the generalization performance of the proposed enhanced algorithm, it was applied to the public environment Pendulum-v1. The hyperparameters were maintained consistent, with the exception of the state space and action space. The improved network structure outlined in **Sect. 3.4** was employed, and training was conducted with eight players simultaneously.

Figure 8 illustrates the average reward observed during the training process. The findings indicate that, during the strategy enhancement phase, **ppo-clip** and **ppo-sample** marginally outperform **ppo**, while **ppo-sample-clip** significantly surpasses the other three methods; In the strategy stabilization phase, the final scores of all four methods are largely comparable, although **ppo** exhibits the highest degree of fluctuation, suggesting its performance is the most unstable; **ppo-sample** demonstrates enhanced exploration capabilities, and **ppo-clip's** performance is closely aligned with **ppo-sample-clip**. The results from the public environment experiments align largely with the conclusions drawn from the aforementioned experiments, indicating that the algorithms proposed in this paper possess a degree of generalization performance. Additionally, the improvement effect of the proposed algorithm in the public environment is not as pronounced as in the previously described environment. This is because the enhancements introduced in this paper are primarily targeted at complex joint operations environments, and simpler environments like Pendulum-v1 are less sensitive to these improvements.

Simulation experiment

As depicted in Fig. 9, following adequate training, the Red bomber initiated its mission to bomb the Blue command center at 11 min and 6 s, in stark contrast to the untrained bomber, which was sent to bomb the Blue command at the beginning of the mission. As illustrated in Fig. 9, the bomb mission unfolded through several stages: take-off, formation assembly, arrival at the target area, commencement of the bombing run, and finally, the return to base upon mission completion.

Figure 10 provides an overhead view of the tactical engagement process. At the center of the figure, the Red side's destroyer is depicted. The blue circle represents the destroyer's air detection range, the brown circle indicates its air strike range, and the green line signifies the destroyer's line of attack. It is evident that the destroyer's air detection range just encompasses the two Blue command posts. The destroyer is engaging the nearest Blue air force units and the combat aircraft in proximity. In this scenario, the Red air combat units commence take-off to provide reinforcement to the Red destroyer and to establish a surrounding formation. This formation, centered around the destroyer, then defends against the Blue Forces' incursion, initiates a counterattack, and subsequently, the bomber strikes the Blue Commands following the establishment of air superiority.

Discussion

In Fig. 9, the Red bombers exhibit a delayed onset of their bombing missions compared to the period when they were untrained. This delay is attributed to the agent's learning process, which reveals that initiating bomber missions at the outset results in the majority of bombers being neutralized by Blue fighters. As a result, the remaining bombers are insufficient to fulfill the attack mission. However, through iterative trial and error, the agent discovers that deploying early warning aircraft (EWA), fighters, and electronic warfare aircraft to first establish air superiority over the battlefield significantly enhances the survival rate of the bombers during subsequent bombing operations. Consequently, the agent has acquired strategies to safeguard crucial combat units and has learned a variety of tactics, including reconnaissance, early warning, assault, and jamming.

During the untrained period, the Red destroyer and combat aircraft operate independently, lacking coordination. However, as depicted in Fig. 10, following adequate training, the combat aircraft engage in operations centered around the destroyer. The Red Navy and Air Force have established a cooperative operational

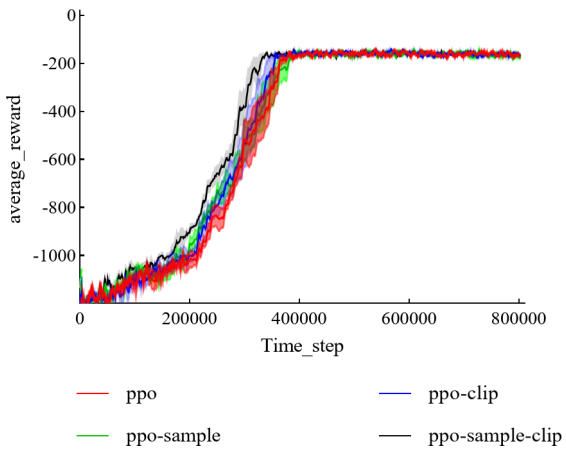


Fig. 8. Comparison of average rewards in pendulum-v1.

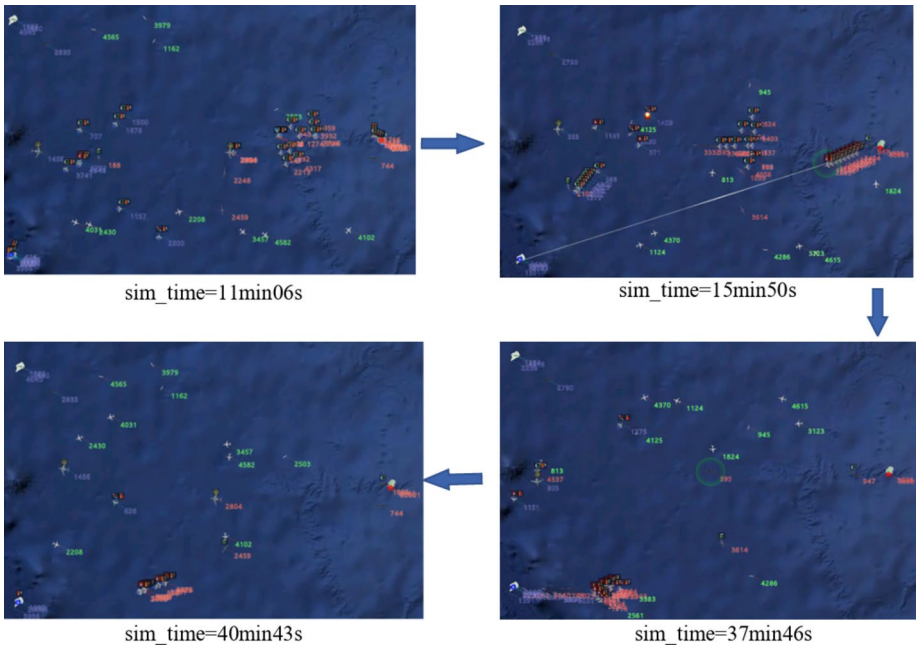


Fig. 9. Bombing tactical process executed by the red force. The red force's bomber receives the order to bomb the blue force's command post and begins the action process.

dynamic. Upon the destroyer's detection of enemy targets, various Air Force combat units, including early warning aircraft (EWA), fighters, electronic warfare aircraft, and bombers, are deployed in formations focused on the naval destroyer. The Navy and Air Force collaborate in defending against and counterattacking the Blue combat units. Subsequently, as the battlefront advances to secure air superiority, the bombers are deployed to strike the Blue Command. This scenario illustrates the agent's capability to coordinate across different military branches and combat units, command joint defensive and offensive operations, and demonstrates its acquisition of tactics such as escort and air defense.

In conclusion, the simulation outcomes indicate that after training, the decision-making agent can effectively utilize reward guidance to identify warfare targets, direct combat units to promptly and successfully enter the task area, cooperate to maintain their combat effectiveness, and efficiently engage and neutralize enemy combat units. In-depth analysis reveals that the agent can accumulate experience through continuous trial and error in complex battlefield situations. It can discern strengths and weaknesses on both sides, coordinate the forces and formations of various military services, and fundamentally fulfill the roles of joint operational commanders. In a more realistic environment, with a greater number of combat units and extended training durations, the agent could potentially employ learned strategies to assist, or even replace, human commanders.

Summary and outlook

In this study, we address the challenges associated with command and decision-making in joint combat environments by employing the PPO algorithm to develop an intelligent decision-making model for joint operations. We propose a novel reward function design methodology to effectively mitigate the issue of sparse rewards. Additionally, we introduce three improvements to the PPO algorithm, and experimental results demonstrate that enhancements to the strategy loss mechanism and the introduction of a prioritized sampling mechanism offer significant advantages over the original PPO algorithm. A comparison of time consumed reveals that the proposed network structure exponentially enhances the algorithm's efficiency. A comprehensive summary of the research findings is presented in Table 5. Ultimately, simulation experiments indicate that the agent constructed using deep reinforcement learning technology can adeptly and decisively deploy forces, achieving desired outcomes in response to complex battlefield dynamics, akin to a real battlefield commander. Furthermore, it has the capability to self-learn and generate novel tactics and warfare methodologies, potentially redefining battlefield rules. These findings underscore the profound application value of DRL technology in the domain of joint operations.

Future research directions will focus on increasing the diversity and number of actions available, as well as enhancing training efficiency, as these represent significant breakthrough opportunities because expanding the range of actions in the decision space implies a more comprehensive weapon system, bringing academic research closer to real-world scenarios. Although the sparse reward issue has been partially addressed, the learning speed of intelligent agents remains less than optimal. To further mitigate the sparse reward problem, we can explore the adoption of hierarchical reinforcement learning network architectures. Additionally, the implementation of multi-agent reinforcement learning methods, where each combat unit operates as an independent decision-making agent, may help alleviate the learning burden on individual agents and yield superior outcomes.

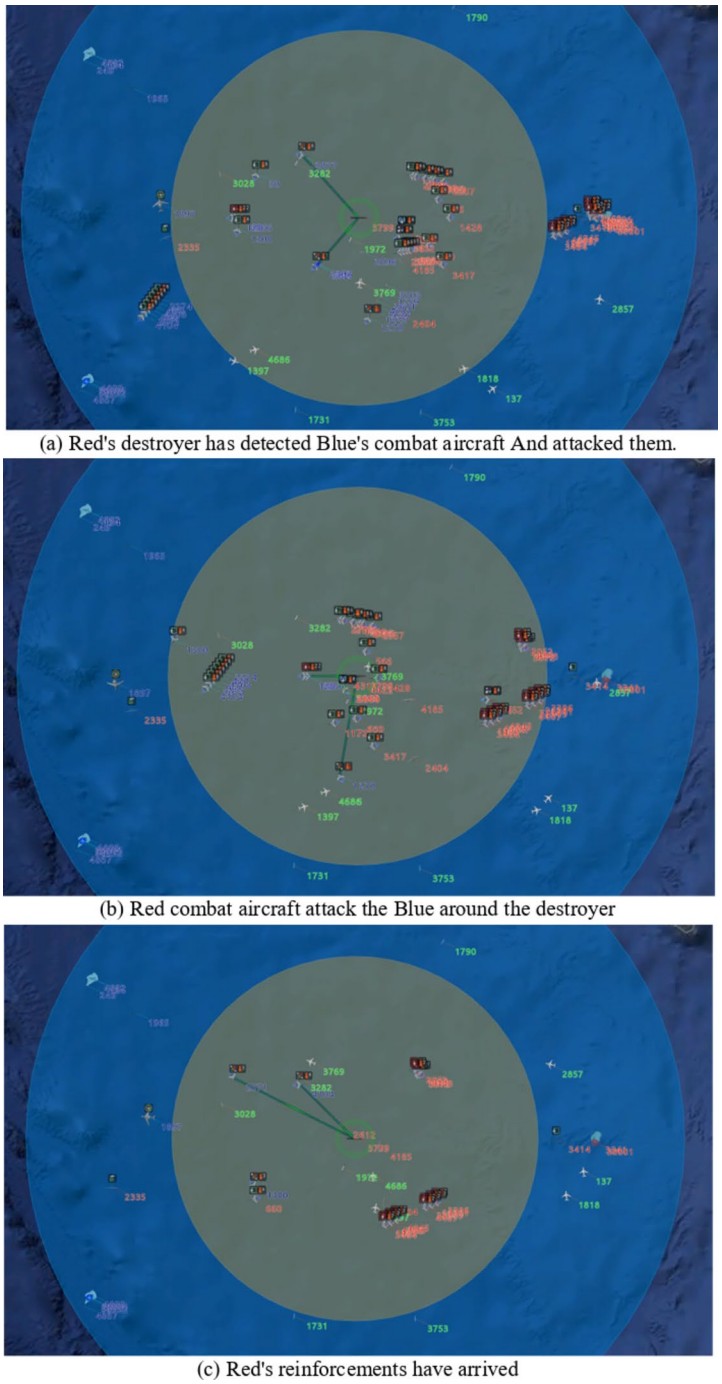


Fig. 10. Tactical Process of formation Executed by The Red Force. After learning, the red force combat units implement tactics around the blue force target and deploy a joint combat formation.

Contributions	Achieved outcomes	Effect
Establishment of a Joint Warfare Intelligent Decision-Making Model	This study initially proposes a universal framework for implementing joint operations intelligent decision-making utilizing RL methodologies, and experimentally verifies the feasibility of the proposed framework	This constitutes the foundation for all research content presented in this paper. The introduction of this universal framework offers a novel perspective for research in the domain of joint operations intelligent decision-making
A methodology for the design of reward functions	A method for balancing process rewards with result rewards is theoretically derived	This reward design approach can mitigate the adverse effects of sparse rewards on model training to a certain extent, and is applicable to nearly all RL training scenarios
An enhanced policy loss mechanism for PPO	By constraining the upper bound of the policy loss function, the variance of the policy loss is reduced, leading to a more stable learning process. The efficacy of this algorithmic improvement has been validated through ablation experiments	This modification targets the PPO loss function and yields performance enhancements in stabilizing the learning process. However, in the context of simple tasks, the performance improvement might be less pronounced
A prioritized sampling mechanism	This paper modifies the original random sampling process by introducing a prioritized sampling mechanism, which enhances the efficiency of sample learning. Ablation experiments demonstrate that the synergistic effect of this method, when combined with the previously mentioned improvement, yields superior results	This represents an enhancement to the PPO sampling learning process, which can accelerate learning speed, albeit potentially introducing instability in the learning process
A network architecture featuring distributed sampling and centralized training	This paper analyzes the learning process of Reinforcement Learning (RL) and employs a method that separates the sampling from the learning process, which effectively enhances the sampling speed. Experimental results validate the improvement in training speed achieved by this network architecture	By fully leveraging the hardware advantages of multi-core and multi-threaded computers, the training speed is doubled. This network architecture is universally applicable for RL training

Table 5. Summary of research findings.

Data availability

The datasets generated and/or analysed during the current study are not publicly available due that the experimental platform in this paper is an internal software, and the organization that owns the software requires that the experimental data be kept confidential. But they are available from the corresponding author on reasonable request.

Received: 19 August 2024; Accepted: 9 January 2025
Published online: 19 March 2025

References

1. Hu, D. et al. Aerial combat maneuvering policy learning based on confrontation demonstrations and dynamic quality replay. *Eng. Appl. Artif. Intell.* **111**, 104767 (2022).
2. Zhao, X., Yang, R., Zhang, Y., Yan, M. & Yue, L. Deep reinforcement learning for intelligent dual-UAV reconnaissance mission planning. *Electronics* undefined, undefined (2022).
3. Yuan, Y. et al. Hierarchical goal-guided learning for the evasive maneuver of fixed-wing UAVs based on deep reinforcement learning. *J. Intell. Rob. Syst.* **109**, undefined (2023).
4. Li, S. et al. Weapon-target assignment strategy in joint combat decision-making based on multi-head deep reinforcement learning. *IEEE Access.* (2023).
5. Pope, A. P. et al. Hierarchical reinforcement learning for air combat at DARPA's AlphaDogfight trials. *IEEE Trans. Artif. Intell.* (2022).
6. Fan, Z., Xu, Y., Kang, Y. & Luo, D. Air combat Maneuver decision method based on A3C deep reinforcement learning. *Machines* **10**, 1033 (2022).
7. Kong, W., Zhou, D., Du, Y., Zhou, Y. & Zhao, Y. Reinforcement learning for multi-aircraft autonomous air combat in multi-sensor UCAV platform. *IEEE Sens. J.* (2022).
8. Li, Y., Shi, J., Jiang, W., Zhang, W. & Lyu Y.-X. Autonomous maneuver decision-making for a UCAV in short-range aerial combat based on an MS-DDQN algorithm. *Def. Technol.* **18**, 1697–1714 (2022).
9. Chai, J. J., Chen, W. Z., Zhu, Y. H., Yao, Z. X. & Zhao, D. B. A hierarchical deep reinforcement learning framework for 6-DOF UCAV air-to-air combat. *IEEE Trans. Syst. Man. Cybern -Syst.* **53**, 5417–5429. https://doi.org/10.1109/tsmc.2023.3270444 (2023).
10. Cao, Y., Kou, Y. X., Li, Z. W. & Xu, A. Autonomous maneuver decision of UCAV air combat based on double deep Q network algorithm and stochastic game theory. *Int. J. Aerosp. Eng.* **2023**, 20 https://doi.org/10.1155/2023/3657814 (2023).
11. Rao, J. J. et al. A modified random network distillation algorithm and its application in USVs naval battle simulation. *Ocean. Eng.* **261**, 15. https://doi.org/10.1016/j.oceaneng.2022.112147 (2022).
12. Wang, P., Yu, C. P., Lv, M. L. & Cao, J. D. Adaptive fixed-time optimal formation control for uncertain nonlinear multiagent systems using reinforcement learning. *IEEE Trans. Netw. Sci. Eng.* **11**, 1729–1743. https://doi.org/10.1109/tnse.2023.3330266 (2024).
13. Liu, R. Z. et al. On efficient reinforcement learning for full-length game of StarCraft II. *J. Artif. Intell. Res.* **75**, 213–260 (2022).
14. Sun, W. F., Lee, C. K., See, S. & Lee, C. Y. A unified framework for factorizing distributional value functions for multi-agent reinforcement learning. *J. Mach. Learn. Res.* **24**, 32 (2023).
15. Liu, Q. et al. Data efficient deep reinforcement learning with action-ranked temporal difference learning. *IEEE Trans. Emerg. Top. Comput. Intell.* **13** https://doi.org/10.1109/tetci.2024.3369641 (2024).
16. Xia, J. Y. et al. Metalearning-based alternating minimization algorithm for nonconvex optimization. *IEEE Trans. Neural Netw. Learn. Syst.* **34**, 5366–5380 (2022).
17. Long, S. et al. A fixed-time consensus control with prescribed performance for multi-agent systems under full-state constraints. *IEEE Trans. Autom. Sci. Eng.* (2024).
18. Sun, G., Zhang, Y., Yu, H., Du, X. & Guizani, M. Intersection fog-based distributed routing for V2V communication in urban vehicular ad hoc networks. *IEEE Trans. Intell. Transp. Syst.* **21**, 2409–2426 (2019).
19. Lin, L., Ma, X., Chen, C., Xu, J. & Huang, N. Imbalanced Industrial load identification based on optimized CatBoost with Entropy features. *J. Electr. Eng. Technol.*, 1–16 (2024).
20. Xu, X., Lin, Z., Li, X., Shang, C. & Shen, Q. Multi-objective robust optimisation model for MDVRPLS in refined oil distribution. *Int. J. Prod. Res.* **60**, 6772–6792 (2022).
21. Sun, G. et al. Low-latency and resource-efficient service function chaining orchestration in network function virtualization. *IEEE Internet Things J.* **7**, 5760–5772 (2019).
22. Zhang, H., Yue, D., Dou, C. & Hancke, G. P. PBI based multi-objective optimization via deep reinforcement elite learning strategy for micro-grid dispatch with frequency dynamics. *IEEE Trans. Power Syst.* **38**, 488–498 (2022).

23. Zhang, T., Guo, S., Tan, T., Hu, X. & Chen, F. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *Adv. Neural. Inf. Process. Syst.* **33**, 21579–21590 (2020).
24. Schulman, J., Moritz, P., Levine, S., Jordan, M. & Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint* arXiv:1506.02438 (2015).
25. Wei, Y. Y., Jiang, N., Zhang, Z., Zeng, M. X. & Yang, Z. K. Research on combat simulation agent modelling methods combined with reinforcement learning. *J. Intell. Fuzzy Syst.* **44**, 1625–1636. https://doi.org/10.3233/jifs-213222 (2023).
26. Alpdemir, M. N. & Sezgin, M. A reinforcement learning (RL)-based hybrid method for ground penetrating radar (GPR)-driven buried object detection. *Neural Comput. Appl.* **21** https://doi.org/10.1007/s00521-024-09466-8 (2024).
27. Ma, C. D., Liu, J. N., He, S. C., Hong, W. J. & Shi, J. Confrontation and obstacle-avoidance of Unmanned vehicles based on progressive reinforcement learning. *Ieee Access.* **11**, 50398–50411. https://doi.org/10.1109/access.2023.3278597 (2023).
28. Wang, Y. H., He, H. & Tan, X. Y. In *35th Uncertainty in Artificial Intelligence (UAI) Conference.* 113–122 (Jmlr-Journal Machine Learning Research (2020).
29. Engstrom, L. et al. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint* arXiv:2005.12729 (2020).
30. Yang, J. C., Zhang, J. P. & Wang, H. H. Urban traffic control in software defined internet of things via a multi-agent deep reinforcement learning approach. *IEEE Trans. Intell. Transp. Syst.* **22**, 3742–3754. https://doi.org/10.1109/tits.2020.3023788 (2021).
31. Huang, N. C., Hsieh, P. C., Ho, K. H. & Wu, I. C. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 12600–12607.
32. Gu, Y., Cheng, Y. H., Chen, C. L. P. & Wang, X. S. Proximal policy optimization with policy feedback. *IEEE Trans. Syst. Man. Cybern -Syst.* **52**, 4600–4610. https://doi.org/10.1109/tsmc.2021.3098451 (2022).
33. Huang, R. C., He, H. W., Zhao, X. Y. & Gao, M. J. Longevity-aware energy management for fuel cell hybrid electric bus based on a novel proximal policy optimization deep reinforcement learning framework. *J. Power Sources.* **561**, 13. https://doi.org/10.1016/j.jpowsour.2023.232717 (2023).
34. Taheri, H., Hosseini, S. R. & Nekoui, M. A. Deep reinforcement learning with enhanced ppo for safe mobile robot navigation. *arXiv preprint* arXiv:2405.16266 (2024).
35. Sun, G., Wang, Y., Yu, H. & Guizani, M. Proportional fairness-aware task scheduling in space-air-ground integrated networks. *IEEE Trans. Serv. Comput.* (2024).
36. Ng, A. Y., Harada, D. & Russell, S. *Icml* 278–287 (Citeseer).
37. Sun, P., Zhou, W. & Li, H. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 5900–5907.
38. Zijian, H., Xiaoguang, G., Kaifang, W., Yiwei, Z. & Qianglong, W. Relevant experience learning: a deep reinforcement learning method for UAV autonomous motion planning in complex unknown environments. *Chin. J. Aeronaut.* **34**, 187–204 (2021).
39. Dong, L. W., Li, N. & Gong, G. H. Curiosity-tuned experience replay for wargaming decision modeling without reward-engineering. *Simul. Model. Pract. Theory.* **129** https://doi.org/10.1016/j.simpat.2023.102842 (2023).
40. Mnih, V. Asynchronous Methods for Deep Reinforcement Learning. *arXiv preprint* arXiv:1602.01783 (2016).

Author contributions

CL and WD carried out the concepts, design, literature search, data acquisition, data analysis and manuscript preparation.LH and MC provided assistance for data acquisition and analysis. DW collected important background information. All authors reviewed the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to W.D.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.