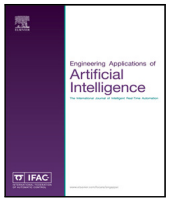
Engineering Applications of Artificial Intelligence 132 (2024) 107893

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



# Dynamic scheduling for multi-level air defense with contingency situations based on Human-Intelligence collaboration

Rugang Tang [a,b], Xin Ning [a,b,*], Zheng Wang [c], Jiaqi Fan [b], Shichao Ma [d]

[a] National Key Laboratory of Aerospace Flight Dynamics, Northwestern Polytechnical University, Xi'an, China
[b] School of Astronautics, Northwestern Polytechnical University, Xi'an 710072, China
[c] Research Center for Unmanned System Strategy Development, Northwestern Polytechnical University, Xi'an, China
[d] School of Aeronautics and Astronautics, Sun Yat-sen University, Shenzhen, 518038, China

ARTICLE INFO

*Keywords:*
Key-point air defense
Dynamic scheduling
Deep reinforcement learning
Human-Intelligence collaboration

ABSTRACT

Resource scheduling is an important part of military operation, especially in key-point air defense under saturation attack. Many achievements have been made in the field of radar resource scheduling and multi-aircraft scheduling. However, there is little research on the integrated scheduling of detection, tracking and attack, which will greatly increase the resource utilization to deal with the resource shortage problem caused by saturation attacks on key places. In this paper, we propose to autonomously accomplish real-time resource dispatching through end-to-end deep reinforcement learning (DRL), while allowing the commander's intervention to accomplish a variety of complex tactics. First, an integrated scheduling model of detection, tracking and interception is proposed and transformed into a sequential decision problem by introducing a disjunctive graph and a graph neural network (GNN) to extract node features. Subsequently, the Proximal Policy Optimization (PPO) algorithm is applied to learn the air defense environment (ADE), which is modeled as an Markov decision process (MDP). Benefitting from the powerful generalization capability of the policy network, our algorithm can adapt to scheduling missions of different sizes. Moreover, we propose a novel Human-Intelligence collaborative dynamic scheduling framework for emergency response. Simulation results indicate that our algorithm generates high-quality scheduling policies for defense resources, exhibiting superior performance than existing methods. In addition, the dynamic scheduling performance of the human and intelligence collaboration approach in response to multiple contingencies is proven.

## 1. Introduction

Key-point defense is an important part of military operations, the goal is to prevent damage to military facilities in and around a certain range of the defense center, with short-range and medium-range interception as the main weapon. As the last line of defense, key-point defense requires the system to complete the mission of detection, tracking, and interception in a very short time. The scheduling of such resources is the first step of the task, and reasonable scheduling wins a lot of time for the defense, especially under saturation attacks. Key-point defense is so important that many scholars have conducted rich research on various aspects of defense. For example, researches (Wang and Pan, 2010; Jia et al., 2021; Liu et al., 2013) provide a detailed study of the optimization of the deployment of defensive weapons around the stronghold. In Nie et al. (2021), air defense environment (ADE) is fully modeled and a method to quantify the threat level of the defense target is proposed. More research has focused on the assignment of targets and trajectory control (Summers et al., 2020; Zhang et al., 2021; Deng et al., 2013; Zhang and Jiahao, 2020), and significant progress has been made in recent years in fast assignment algorithms (Zhao et al., 2019; Hocaoğlu, 2019; Liu et al., 2015) and game-theory-based decision-making (Li et al., 2022; Ma et al., 2019).

In the field of defense resource scheduling, the scheduling of phased radar has always been the focus of research, which has had a lot of excellent results and has been put into engineering application (Miranda et al., 2007; Sgambato et al., 2016; Shan et al., 2020; Li et al., 2020; Winter and Baptiste, 2007). In particular, in the dynamic scheduling of radar systems for multiple-input multiple-output (MIMO), effective methods have been proposed and extensive simulation experiments have been done in Kim and Chang (2020), Yi et al. (2020) and Zhang et al. (2019). Zhang et al. (2017) and Tuncer and Cirpan (2022) propose some target prioritization methods and apply them as the basis for node ordering in scheduling. In addition, considerable progress has been made in the multi-constrained instant scheduling algorithm for multi-functional radars (Orman et al., 1996; Tian et al., 2018). Although these

* Corresponding author at: National Key Laboratory of Aerospace Flight Dynamics, Northwestern Polytechnical University, Xi'an, China.
*E-mail address:* ningxin@mail.nwpu.edu.cn (X. Ning).

https://doi.org/10.1016/j.engappai.2024.107893
Received 8 October 2022; Received in revised form 28 November 2023; Accepted 9 January 2024
Available online 20 January 2024

studies are only for radar scheduling, their application scenarios are similar to this study, which gives us a lot of inspiration.

In spite of many special constraints on defense resources, the scheduling of defense resources is still a typical scheduling problem, which is considered to be NP-hard, requiring an exhaustive search of all possible combinations to find the best solution, and the complexity of this search increases exponentially with the number of tasks. Therefore, it is impossible to solve the scheduling problem by polynomial time algorithm, and only approximate algorithm or heuristic method can be used to obtain the suboptimal solution. Job shop problem (JSP) is one of the earliest widely studied scheduling problems, and it is easy to be described standardized, so many advanced algorithms have been produced in the study of JSP (Han and Yang, 2021; Zhang et al., 2020b; Park et al., 2021; Han and Yang, 2020). Although the constraints and objectives of JSP are very different compared to military resource scheduling, these algorithms are still of great interest to us. Flexible job shop problem (FJSP) involves the problem of machine selection, which is closer to our scenario. Lei et al. (2022) and Luo (2020) propose some algorithms that have good results in their models, which is worthy of our reference.

It is very encouraging that such significant results have been achieved, for example, many efficient algorithms of phased radar scheduling have been tested in theory and practice. However, integrated scheduling of multi-level resources in the defense of key-point is rarely mentioned, and existing research has mainly focused on quantifying the priority of targets, which is less effective in dealing with situations such as system overload scheduling under saturation attacks. In addition, the studies ignore the commander's influence on the battlefield and use a single algorithm result as the basis for scheduling. In this paper, considering the above gaps in the current study, we have done some meaningful work. Compared with previous studies, the main features of this paper can be summarized as:

- A novel multi-level joint scheduling framework for defense systems is proposed, which models the scheduling problem of detection, tracking and interception as Markov decision process (MDP), filling the gap in the field of multi-level defense resource scheduling.
- In this paper, an end-to-end algorithm based on deep reinforcement learning is proposed, which can obtain a better multi-level defense resource scheduling scheme than the existing algorithms under saturation attacks. At the same time, well-trained policy networks have good generalization ability and can cope with various changes of ADEs. A large number of simulation experiments prove the superiority of the algorithm.
- A novel framework of dynamic scheduling of Human-Intelligence collaboration for emergency response is proposed, which we call DSHI, and a corresponding emergency event triggering mechanism is developed, making the scheduling algorithm more flexible for ADE applications and providing an excellent example for subsequent research as well as other scheduling scenarios.

## 2. Problem formulation and preliminaries

### 2.1. Scenario description

Swarm warfare has become an important mode of combat, attributed to the improved capability of aircraft to operate in collaboration, with the attacking side $\mathscr{B}$ using a large number of UAVs or missile swarms as combat units to attack strategic locations. The defending side $\mathscr{R}$ uses a fan-shaped deployment to set up multiple lines of defense around the stronghold and conducts multi-method defense through detection, tracking, and interception, which is shown in Fig. 1. In the real battlefield environment, to adapt to the complexity and variability of the battlefield environment, both sides are equipped with a diversity of weapon resources, making the defensive weapon resources as well as combat missions heterogeneous. Under saturation attacks such as swarm combat mode, it is urgent to explore a resource planning and scheduling mechanism that integrates detection and warning, tracking and identification, coordinated interception to meet the requirement of online response to complex heterogeneous tasks. The integrated detection, tracking and interception scheduling framework proposed in this paper is oriented to the autonomous defense system of key points under saturation attack. This framework can use the existing resources as much as possible, save the scheduling time, and optimize the utilization of defense resources.

### 2.2. Parameter definitions and assumptions

(1). Target: $\mathbb{T}_t = \left\{\text{target}_1^t, \text{target}_2^t, \ldots, \text{target}_{N_t}^t\right\}$ denotes the set of suspicious targets in this defense area at time $t$. In which $N_t$ denotes the number of suspicious targets at time $t$, $\text{target}_i^t = \{Type_i^t, Threat_i^t, Position_i^t, Time_i^t\}(i = 1, 2, \ldots, N_t)$ represents the state information of target $i$ at time $t$, including the type tags $Type_i^t$, the threat level $Threat_i^t$, the node position prediction vector $Position_i^t$, and the node time prediction vector $Time_i^t$ of the $i$th target which is obtained from the higher-level warning system and a priori information.

(2). Mission: $\mathbb{M}_t = \left\{Mission_1^t, Mission_2^t, \ldots, Mission_{N_t}^t\right\}$ represents the set of defense missions for team $\mathscr{R}$ at time $t$. The number of tasks and objects are determined by the number and type of task targets. Where $Mission_i^t = \{Mission_i^D, Mission_i^T, Misson_i^I\}$ denotes the task $i$ corresponding to target $i$ at time $t$, including three submissions: detection $Mission_i^D$, tracking $Mission_i^T$, and interception $Mission_i^I$. Taking the detection submission as an example, the information of each mission includes the index of its matching defense resources $res_i^D$, the estimated consumption time $T_i^D{}_{consum}$, the earliest start time $T_i^D{}_{start}$, and the latest end time $T_i^D{}_{end}$.

(3). Resource: $\mathbb{R} = \left\{\text{Resource}_1, \text{Resource}_2, \ldots, \text{Resource}_{N_R}\right\}$ denotes the set of defense resources. In which $N_R = N_R^D + N_R^T + N_R^I$ represents the number of all defense resources, including the number of detection resources $N_R^D$, the number of tracking resources $N_R^T$, and the number of interception resources $N_R^I$. $Resource_i = \{Type_i^R, Position_i^R, Capability_i^R\}$ includes the $Type_i^R$, $Position_i^R$, and $Capability_i^R$ of the $i$th defense resource.

To simplify non-essential conditions, the following assumptions are made to clarify the environment in which the model applies:

**Assumption 1.** Different targets have varying levels of $Threat_i^t$, which is obtained from the system state, leading to discrepancies in priority between missions.

**Assumption 2.** The time consumption $T_i^D{}_{consum}$ of each submission for the corresponding defense resource can be obtained from the target information $\text{target}_i^t = \{Type_i^t, Threat_i^t, Position_i^t, Time_i^t\}(i = 1, 2, \ldots, N_t)$ and resource status $Resource_i = \{Type_i^R, Position_i^R, Capability_i^R\}$.

**Assumption 3.** Under saturation attack, the consumption time is the minimum time for resources to complete the basic task. Consumption time of interception resource is the total duration of the launch system from preparation to launch completion, excluding the flight time of the missile.

### 2.3. Mathematical formulations

The objective of this study is to construct an integrated joint scheduling framework for detection, tracking, and interception in the case of heterogeneous defense resources and targets, and to achieve fast solutions in the case of battlefield emergencies. let $S_i^{D,T,I}(i = 1, 2, \ldots, N_t)$ and $C_i^{D,T,I}(i = 1, 2, \ldots, N_t)$ denote the start and the completion time of the $i$th task respectively, then the algorithm needs to quickly find a set of task sequences $\mathbb{S}$ with all constraints satisfied so that the maximum task completion time is as small as possible. The

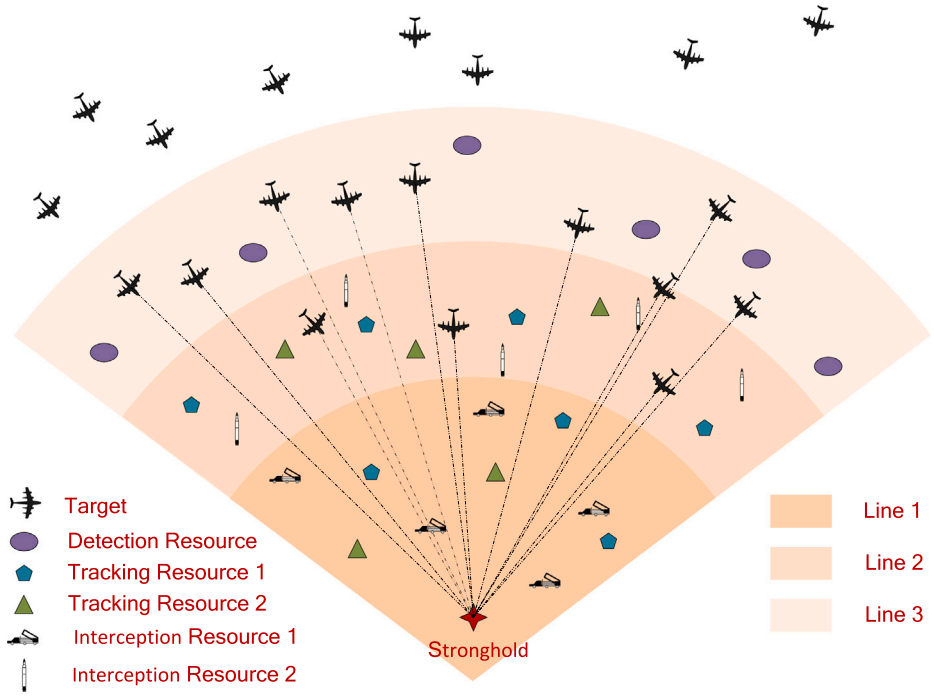



**Fig. 1.** Key position air-defense based on multi-weapon platforms.

mathematical form is expressed as follows:

$$\min_{\mathbb{S}_{ij}} \max_{i} \mathrm{C}_i \quad (\mathrm{i} = 1, 2, \ldots, \mathrm{N}_t; j = 1, 2, 3) \tag{1}$$

For $\forall\ \mathrm{i} = 1, 2, \ldots, \mathrm{N}_t$, we have

$$\begin{cases} T^D_{i\ start}(res^D_i) < S^D_i < C^D_i < T^D_{i\ end}(res^D_i) \\ T^T_{i\ start}(res^T_i) < S^T_i < C^T_i < T^T_{i\ end}(res^T_i) \\ T^I_{i\ start}(res^I_i) < S^I_i < C^I_i < T^I_{i\ end}(res^I_i) \end{cases} \tag{2a}$$

$$Type^t_i = Type^R_{res^D_i} \tag{2b}$$

$$S^D_i < S^T_i < S^I_i \tag{2c}$$

where Eq. (2a) indicates that each submission is completed within the specified time window, which ensures the time effectiveness; Eq. (2b) indicates that the target and the defensive weapon are type-matched, ensuring the efficient use of resources; Eq. (2c) represents that submissions are guaranteed to be performed sequentially as Detection → Tracking → Interception.

## 3. GNN-based mission description

### 3.1. Disjunctive graph

The disjunctive graph is widely used in the job shop scheduling problem (JSSP) and has remarkable effects in some scenarios. However, limited by the complex constraints and the specificity of the defense system, the traditional dynamic program (DP) cannot adequately reflect the state information of the scheduling problem in our case. Therefore, we propose a new Disjunctive graph to meet the model requirements in this section.

Let $\mathcal{O} = \{O_{ij} \mid \forall i \in N_t, j \in N_R\} \cup \{S, T\}$ be the set of all operations, where $S$ and $T$ represent the initial and terminal operation, respectively, for receiving/sending commands to the end defense command system. For each node $O_{ij} \in \mathcal{O}$, meaning that $i$th target is assigned to $j$th resource, we assign four features to reflect the type of $Resource_j$ as:

$$O_{ij}(C_{LB}, I, T^u_{i\ start}(Resource_j), T^u_{i\ end}(Resource_j), u, Pri) \tag{3}$$

where $C_{LB}$ represents the lower bound of the estimated completion time; $I$ is 0 or 1 for the completion status of the operation; $T^u_{i\ start}(Resource_j)$ and $T^u_{i\ end}(Resource_j)$ denote the earliest start time and the latest end time of the node respectively, $u \in \{D, T, I\}$ is the type of $Resource_j$; and $Pri$ is the level of priority. Then the disjunctive graph of scheduling is $G = (\mathcal{O}, C, D, \mathcal{E})$, in which $C$ is the set of directed arcs (conjunctions) connecting resources of $Mission^D_i, Mission^T_i, Misson^I_i$ that pass through the same target; $D$ is the set of dashed undirected arcs (disjunctions), which connects resources under the same mission; $\mathcal{E}$ denotes a set of dashed directed arcs connecting different operations on the same resource.

**Remark 1.** Node features shown in Eq. (3) can be adjusted to system characteristics and mission requirements, such as adding resource utilization, thus adapting to different battlefield environments.

An example of three targets at time $t$ is shown in Fig. 2, where $N^D_R = N^T_R = N^I_R = 3$. For brevity, the node features in the figure is only labeled with estimated consumption time $T^I_{i\ consum}$. The final state of the example indicates that $target^t_1$ is assigned to $Resource_3$, $Resource_5$, $Resource_9$ in $Mission^D_i$, $Mission^T_i$ and $Mission^I_i$, respectively; and $Resource_3$ processes $target^t_1$ and $target^t_3$ in turn.

### 3.2. Graph embedding with GNN

The Graph Isomorphism Network (GIN) framework (Xu et al., 2018), which has been shown to outperform structures such as GraphSAGE (Hamilton et al., 2017) and Graph Convolutional Network (GCN) (Kipf and Welling, 2016) and achieve state of the art (SOTA) accuracy on multiple tasks, is used to extract node features from the disjunctive graph in this study.

$$h^{(k)}_O = \mathrm{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \cdot h^{(k-1)}_O + \sum_{u \in \mathcal{N}(O)} h^{(k-1)}_u\right) \tag{4}$$

where $h^{(k)}_O$ and $h^{(k-1)}_O$ is the feature vector of $O$ at the $k$th and $(k-1)$th iteration, respectively; $\epsilon$ is a learnable parameter or a fixed scalar; $h^{(k-1)}_u$ is the feature vector of $u \in \mathcal{N}(O)$, which is a set of nodes adjacent to $O$ at the $(k-1)$th iteration.

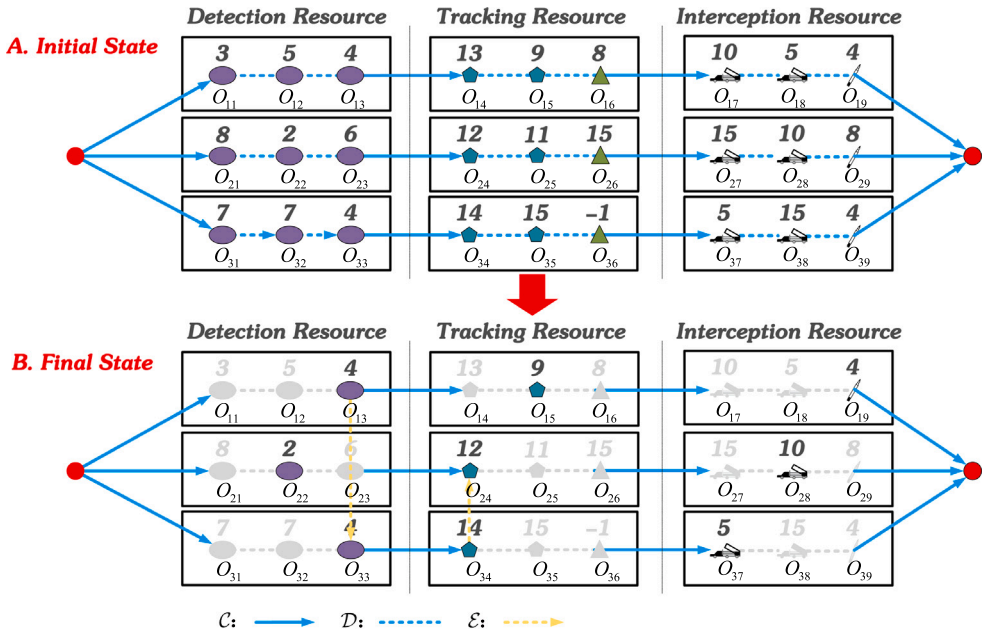



**Fig. 2.** An example of disjunctive gragh.

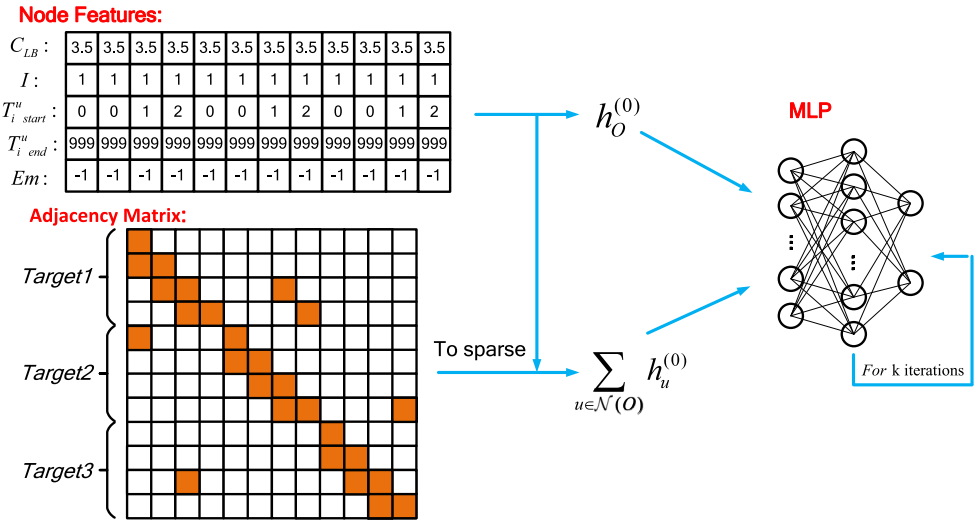



**Fig. 3.** Embedding with GIN.

In the resource scheduling model, each task is a node, and the logical sequence of tasks is reflected by the connections between nodes. As shown in Fig. 3, input node data $\mathcal{O}$ which contains information of $C_{LB}$, $I$, $T^u_{i\ start}(Resource_j)$, $T^u_{i\ end}(Resource_j)$, $u$, and $Pri$, as well as data about the relationship between nodes such as $\mathcal{C}, \mathcal{D}, \mathcal{E}$ into Multi-Layer Perceptron (MLP). MLP uses multi-layer stacking of graph neural network, which has strong feature extraction ability and generalization performance. In this case, scheduling problems of any size can be converted into fixed dimension embedding, which is the basis of dynamic scheduling.

## 4. Scheduling with DRL

In this section, the multi-weapon platform joint scheduling problem is modeled as an MDP, based on which the environment for the scheduling problem is built and learned by deep reinforcement learning (DRL).

### 4.1. MDP modeling

The Markov decision process is defined by the tuple $(S, A, T, R, \gamma)$, where $S$ is the state space, $A$ is the set of actions, $T : S \times A \times S \to [0, 1]$ is the state transfer probability function, $R$ is the reward function, and $\gamma$ is the discount factor. $T$ represents the probability distribution of transitioning to the next state given the current state and behavior:

$$\sum_{s' \in S} T\left(s, a, s'\right) = 1 \quad \forall s \in S, \forall a \in A \tag{5}$$

in which, $s'$ represents the possible state at the next moment. $R$ represents the reward in the next state given the current behavior and state. MDP has the following Markov properties: the player's next state and reward depend only on the player's current state and behavior. Player strategy $\pi : s \to A$ is defined as the probability distribution of player behavior in a given state. The player's strategy $\pi(s, a)$ should satisfy:

$$\sum_{a \in A} \pi(s, a) = 1 \quad \forall s \in S \tag{6}$$

In any MDP, there exists a certain optimal policy, where $\pi^*(s,a) \in [0,1]$. The goal of the player in the MDP is to maximize the expected long-term return. To evaluate a player's strategy, one needs to have the following state-value function: When the player starts from state s and then executes policy $\pi$, the value of state $s$ (or state-value function) under that policy is defined as the expected return. Thus, the state value function $V^{\pi}(s)$ is

$$V^{\pi}(s) = E_{\pi}\left\{\sum_{k=0}^{T} \gamma^k r_{k+t+1} \mid s_k = s\right\} \tag{7}$$

where $T$ is the final time; $t$ is the current time; $r_{k+t+1}$ is the direct reward obtained at the time $k+t+1$; $\gamma \in [0,1]$ is the discount factor.

### 4.2. Air defense environment

To meet the characteristics of MDP, the following settings are made for the air defense environment (ADE).

**State.** The state of the scheduling problem should contain node features and topological information between nodes, which are characterized in this paper using $O_{ij}(C_{LB}, I, Type, Em)$ and adjacency matrix, respectively. Details are shown in Section 3.

**Action.** Use the set of nodes $\mathcal{O} = \{O_{ij} \mid \forall i,j\} \cup \{S,T\}$ as the Action Space. Some of these nodes are not optional in state $s$, e.g. (a).Nodes that have already been planned; (b).Nodes that do not belong to the current process; (c).Nodes that do not match the target type. let the set of unselectable nodes be $\widetilde{A}$, then $\forall\ a$ in $\widetilde{A}$, $\pi(s,a) = 0$.

**Priority.** Prioritization is based on the threat level of the target and the task urgency, which is $Pri = Threat_i^t + Em$. If the node with the highest priority is not selected during the current node selection, it will be punished accordingly, thus guiding the policy network to preferentially select the node with the highest priority during the training process. It is worth noting that in our model, the priority of nodes does not represent the priority of node execution, and those tasks scheduled later may also be executed earlier than those scheduled first. The Priority mechanism can be seen as a sort of Priority Dispatching Rule (PDR) method that complements the DRL method in our framework.

**Reward.** The reward is divided into three parts:

1. The change in total makespan after completing the current action $max(C_{LB}^{current}) - max(C_{LB}^{former})$.
2. When the priority of the selected action is not the highest in $\left\{A - \widetilde{A}\right\}$, the penalty $Reward = Reward - P_{Pri} \times (Pri_{max} - Pri_a)$ will be executed, where $P_{Pri} = 20$ is the penalty factor of priority.
3. When action $O_{ij}$ does not fall within the time window $[T^u_{i\ start}(Resource_j), T^u_{i\ end}(Resource_j)]$, the penalty $Reward = Reward - P_{TW}$ will be executed, where $P_{TW} = 500$ is the penalty factor of the time window.

**Reset & Update.** The algorithm of reset and update are shown in Algorithms 1 and 2, respectively.

**Algorithm 1** ADE Reset

**Require:**
Target: $\mathbb{T}_t = \left\{\text{target}_1^t, \text{target}_2^t, \cdots, \text{target}_{N_t}^t\right\}$
Mission: $\mathbb{M}_t = \left\{Mission_1^t, Mission_2^t, \cdots, Mission_{N_t}^t\right\}$
Resource: $\mathbb{R} = \left\{\text{Resource}_1, \text{Resource}_2, \cdots, \text{Resource}_{N_R}\right\}$
**Ensure:**
Initial state & System info;
Calculate the duration time of Missions in Resources:
**for** i in number of Missions and j in number of Resources **do**
  **if** $Type_i$ != $Type_j^R$ **then**
    $duration_{ij} = -1$
  **else**
    Combine relative positions and $Capability_j^R$ to determine $duration_{ij}$
  **end if**
**end for**
Initialize Adjacency Matrix
Initialize node features: $C_{LB}, I, Type, Em$:
**for** i in number of Missions and j in number of Resources **do**
  Calculate the minimum duration time in submissions ($Mission_i^D$, $Mission_i^T$, $Mission_i^I$);
  Calculate $C_{LBij}$ based on the minimum time of the last submission and the current duration time
**end for**
Initialize candidate set
**return** Adjacency Matrix, Node features, Candidate;

**Remark 2.** In practice, in order to achieve uniformity in the candidate set dimension, we equate it to $\mathcal{O} = \{O_{ij} \mid \forall i,j\} \cup \{S,T\}$ and update the optional set by setting the probability of unselectable nodes to 0, which is $\forall\ a$ in $\widetilde{A}$, $\pi(s,a) = 0$.

**Algorithm 2** ADE Update

**Require:**
action: $\mathcal{O} = \{O_{ij} \mid \forall i \in N_t, j \in N_R\} \cup \{S,T\}$
**Ensure:**
Next state, Reward, System info;
**if** action available **then**
  Update Finished mark, Current process...
  **if** $Mission_i$ can operate in the gap time of $resource_j$ **then**
    Set $InsertFlag = 1$ and Calculate the insert time
  **else**
    Set $InsertFlag = 0$ and Arrange $Mission_i$ at the end of $resource_j$
  **end if**
  Update Adjacency Matrix ($adj$):
  Get the previous operation as $Action_{precd}$ and the current ($InsertFlag = 0$) or next ($InsertFlag = 1$) operation as $Action_{succd}$
  Connect Adjacent Operations:
  $adj[action, Action_{precd}] = 1; adj[Action_{succd}, precd] = 1$
  **if** $InsertFlag$ and $Action_{precd}$ != $action$ and $Action_{succd}$ != $action$ **then**
    $adj[succd, precd] = 0$
  **end if**
  Update $C_{LB}$, Node features, Candidate, Reward
**end if**
**return** Adjacency Matrix, Node features, Reward, Candidate;

**Remark 3.** The same target executed by different resources is considered a different tasks. When $Target_i$ is assigned to $Resource_j$, which is $action = O_{ij}$, nodes of other resources of the same level (Detection/Tracking/Interception) are added to $\widetilde{A}$ (fail).

### 4.3. Learning via proximal policy optimization

Developed from TRPO (Schulman et al., 2015), PPO is a well-established reinforcement learning algorithm with many successful applications (Schulman et al., 2017). TRPO is difficult to handle because it treats the KL scatter constraint as an additional constraint that is not placed inside the objective. PPO, on the other hand, treats the constraint directly as part of the objective function, making the algorithm much more feasible.

First, state of air defense environment (ADE) extracted by GIN in Section 3 is fed into the actor network, the mean and variance are calculated, and a normal distribution is constructed, which is then used to sample an action. The environment calculates the reward and the next state based on the current state and action. Keep cycling through this step until a sufficient amount of state–action-reward and so on

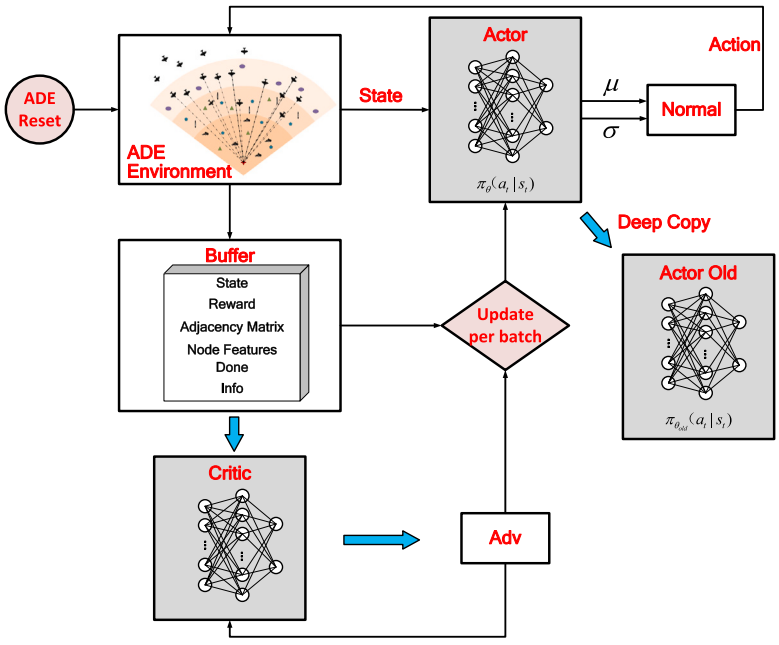



**Fig. 4.** Proximal policy optimization algorithm for ADE.

data is stored. During this process, the actor network was not updated. Second, input the stored state–action-reward and other data into the Critic network to get the evaluation of all states. Third, Calculate loss and back propagate update critic network. The Actor network deep copy forms an Actor Old network, which is updated after a certain number of cycles. The important weight is calculated by Actor network and Actor Old network, which is a key parameter for actor network update (see Fig. 4).

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[ \min \left( r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{8}$$

## 5. Dynamic scheduling of Human-Intelligence collaboration for emergency response

In this section, a framework of dynamic scheduling of Human-Intelligence (DSHI) collaboration for emergency response is proposed, which is shown in Fig. 5, and based on which common contingencies on the battlefield as well as human operational events are listed; In addition, due to system errors and possible system unlisted events, we propose a hybrid scheduling strategy of cyclical reprogramming and emergency response.

### 5.1. Structure of DSHI

Although it consumes more computational resources during offline training, the proposed algorithm can effectively cope with the disruption of schedule by battlefield contingencies because it can perform online fast settlement after training and has good generalization performance. With the scheduling center as the information hub, the system contains ADE, Operators, Battlefield, Resources, and strategy network modules.

Before going into detail on each part, let us give an overview of the mechanism of action of DSHI. The first is the initial off-line training of the policy network. The PPO policy network has undergone a large amount of training in the ADE constructed above to form an initial policy network, which can complete decision-making in the changing ADE. The second step is the fast online decision-making of the policy network. The scheduling center obtains manual instructions, target information, battlefield situation information, etc., and inputs them into the encapsulated policy network. The policy network outputs scheduling results and returns them to the center, which sends corresponding instructions to the detecting, tracking and intercepting resources. At the same time, ADE will also update according to the battlefield information and the actual scheduling effect, and the updated strategy network will be fed back to the scheduling center.

**Scheduling Center.** The scheduling center is responsible for receiving data and system status information from modules such as targets, resources, and operators, and then inputting the status information into the strategy network to obtain a dispatch plan and distribute it to sub-dispatch stations and resources.

**Strategy Network & ADE.** The strategy network is obtained by offline interactive learning computation of the agent with the built ADE through the proposed algorithm, is able to solve instantaneously according to the system state. It is worth mentioning that the network has excellent generalization performance when the resources and targets change. To further enhance the system's scheduling capabilities, ADE regularly receives information from resources, targets, etc., and continuously improves model accuracy to closely match real battlefield conditions.

**Resources/Sub-Dispatch Stations.** The resource/sub-dispatch station is the actuator of the defense mission and performs detection, tracking, or interception missions against targets by decoding the dispatch results sent by the scheduling center. At the same time, each resource needs to monitor its status and send it to the dispatch center in time.

**Battlefield.** The battlefield is the physical environment where the scheduling system acts and contains massive amounts of data. In the scheduling algorithm, we mainly extract target information explained in Section 2.2 and send it to the dispatch center and operators. In addition, ADE performs model updates based on battlefield data.

**Human-in-the-loop.** The policy network-driven scheduling algorithm works well in routine situations but has natural limitations when executing some special tactics or accomplishing special operational requirements. To cope with these situations, the operator is allowed to intervene in the dispatch algorithm to customize the combat strategy through some combinations of commands while combining with the strategy network to achieve collaborative human-intelligence dispatch.

In order to address complex emergencies and normalize event trigger signals, the system needs to define basic emergency events from which more complex events can be combined. In this paper, we propose $E1 - E7$ emergency events, where $E7a$, $E7b$, and $E7c$ are manual

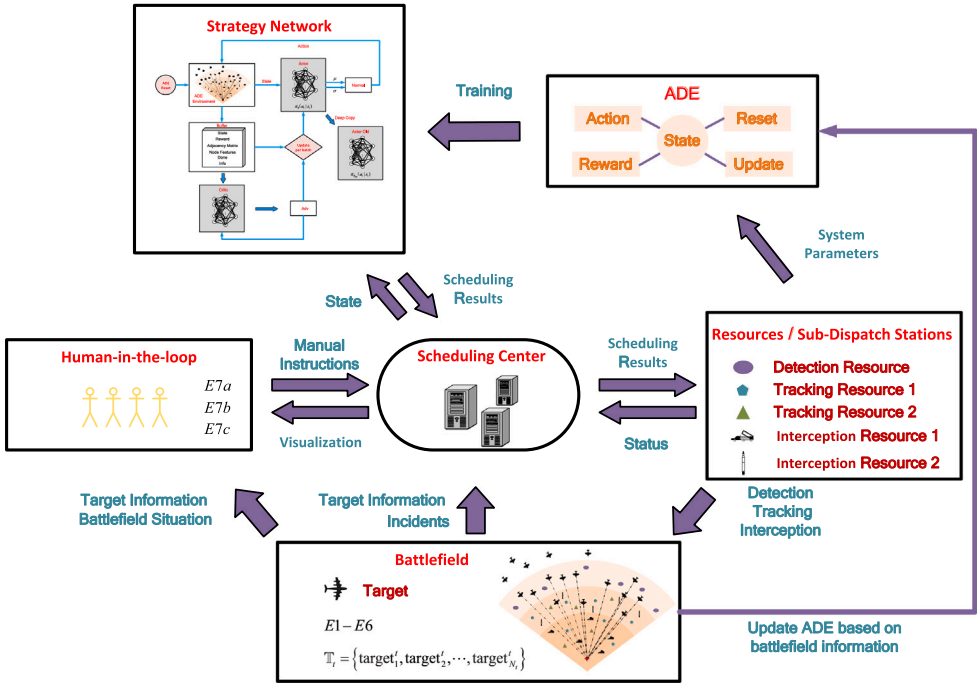



**Fig. 5.** Framework of dynamic scheduling.

**Table 1**
Emergency events.

| Index | Description | Response |
|---|---|---|
| $E1$ | Resource failure | Remove failed resources and reschedule related missions |
| $E2$ | Ineffective interception | Regenerate the intercept submissions and classify it as urgent |
| $E3$ | Target intrusion without warning | Generate tracking (high priority) and intercept submissions |
| $E4$ | Multiple interception requirements | Generate multiple parallel intercept tasks and limit them to the same time window |
| $E5$ | Priority changes | Update $Em$ of $O_{ij}$ |
| $E6$ | Early mission termination | Remove related missions (assigned but not completed/unassigned) |
| | Manual instructions | |
| $E7a$ | Specify the main interception direction | Increase the priority of missions in this direction according to the degree of urgency |
| $E7b$ | Change defense center | Change location information to completely reprogram missions not performed |
| $E7c$ | Prioritize the use of specific missions | Raise the priority of relevant missions according to the degree of urgency |

instructions. Details are shown in Table 1. These manual instructions are equivalent to emergencies events for policy networks, which will simplify the algorithmic flow from the bottom up.

### 5.2. Hybrid rescheduling

Since insufficient detection accuracy, data transmission delay, and hardware errors are unavoidable, event-triggered online replanning cannot fully meet the demand for scheduling accuracy. In this paper, an online rescheduling mechanism combining periodic reprogramming and emergency event-triggered rescheduling is used. The scheduling triggered by emergency events enables the proposed algorithm to respond in time in the changing environment, while the periodic scheduling can reevaluate the system state at one cycle interval and correct the system errors including the detection accuracy.

Considering that the system has delays in performing rescheduling tasks due to hardware switching and other factors, it is necessary to set the lead time for rescheduling. Set the replanning period to $T$ and the lead time to $\Delta T$, then the system starts the replanning mechanism at the time $nT (n = 1, 2, \ldots)$ and replans the tasks after $T^n = nT + \Delta T$. In the $\Delta T$ time, the resource and other executing agencies will complete the configuration of new tasks. Similarly, event-triggered replanning requires setting the lead time according to the urgency and complexity of the event.

An example is given in Fig. 6. Incidents $E3$, $E2$, $E6$, and $E7b$ occur at moments $T_{I1}$, $T_{I2}$, $T_{I3}$, and $T_{I4}$, respectively, and the lead times are $\Delta T_3$, $\Delta T_2$, $\Delta T_6$, and $\Delta T_{7b}$.

## 6. Numerical experiments

We show the results of the simulation experiments in this section. First, the experimental environment is described, including the algorithm parameters, the construction of the simulation scenario, and the generation of the datasets. Second, the superiority of the proposed algorithm in solving the static scheduling problem and its strong generalization performance are verified by comparative simulation. Finally, we test the performance of DSHI in dynamic scheduling and prove its robustness and speediness.

### 6.1. Experimental conditions

In order to simulate the complex environment of the battlefield, we have designed a data generator based on defense information with the module information shown in the figure. Flexible data classes were created to enable us to customize the data in different situations while facilitating peers to modify it according to their needs.

First, basic information, such as deployment radius and target radius, is used as input to create the base class of defense data. Then, a number of targets and resources are created and initialized by choosing random generation or custom settings. The initial defense data is

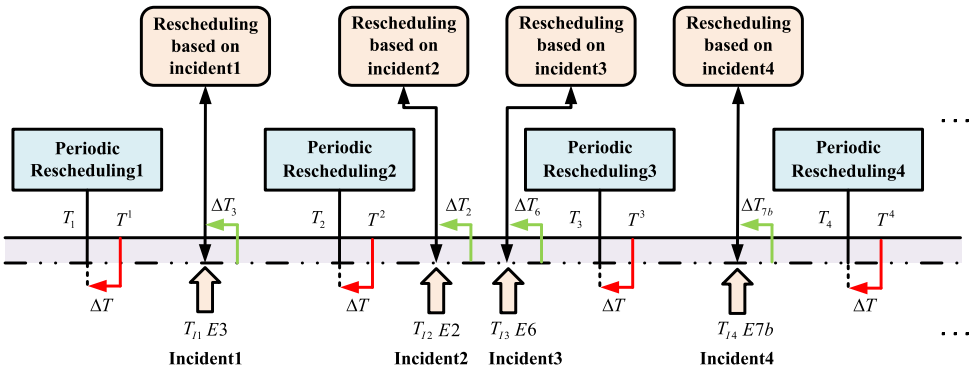



**Fig. 6.** Hybrid scheduling of periodic reprogramming and emergency response.

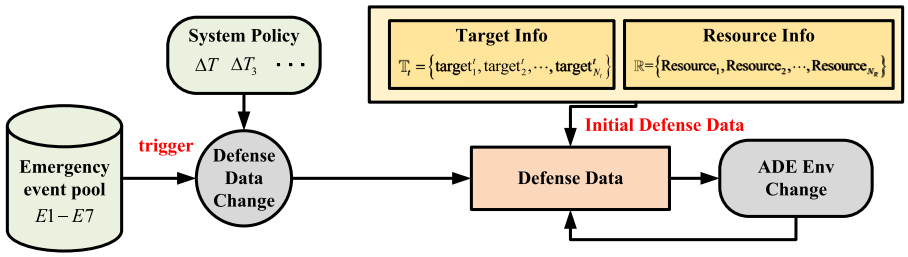



**Fig. 7.** Data generator.

**Table 2**
Example of generated defense data.

| Resource | | $Target_1$ | $Target_2$ | $Target_3$ | $Target_4$ | $Target_5$ | $Target_6$ |
|---|---|---|---|---|---|---|---|
| Detection | 1 | 78 | 51 | 91 | 52 | −1 | 78 |
| | 2 | −1 | −1 | 60 | 88 | −1 | 63 |
| | 3 | −1 | −1 | 93 | 94 | 55 | 95 |
| Tracking | 4 | −1 | −1 | 97 | 58 | −1 | 53 |
| | 5 | 76 | 70 | −1 | −1 | 71 | −1 |
| | 6 | −1 | −1 | 81 | 77 | −1 | 75 |
| Interception | 7 | −1 | −1 | 74 | 75 | −1 | 87 |
| | 8 | 65 | 55 | −1 | −1 | 55 | −1 |
| | 9 | −1 | −1 | 83 | 84 | −1 | 59 |
| Emergency events: 107-E6-M3, 249-E1-R4, 532-E7c-M8 | | | | | | | |

[1] Duration is a number between 50-99 time units in this case.
[2] −1 means the target is not executable by this resource.
[3] According to Table 1, emergency event 107-E6-M3 indicates that Mission 3 ($Target_3$) terminates early at moment 107; emergency event 249-E1-R4 indicates that at moment 249, $Resource_4$ fails; and emergency event 532-E7c-R8 indicates that a manual operation causes Mission 8 ($Target_8$) to be elevated in priority at moment 532.

obtained by creating tasks with the status of targets and resources and pre-set information. The defense information correction is triggered based on the timing information of randomly generated events in the emergency event set defined in Table 1, and the ADE will be changed as a result. Fig. 7 shows the data transfer process. An example with 6 targets and 9 resources is given in Table 2. All work is done using python = 3.6 and torch = 1.7.1 with Intel i9-12900k and a simple Nvidia 3070 GPU. Parameters of the algorithm are set as in Appendix.

### 6.2. Results of static scheduling

The solution to the scheduling problem can be divided into two types of methods: rule-based methods and heuristic algorithm-based methods. Among them, the accuracy of the solution of the heuristic algorithm has been shown to be better than single rule planning in some scenarios. However, in our scenario, ADE strongly requires the speed of the algorithm, compared to industrial production or other operations research scenarios. Even though genetic algorithms, for example, might achieve better solutions, their huge computational time consumption is unacceptable for military applications. Based on this consideration, in this paper, we mainly use various rule-based scheduling methods, such as First In Fist Out (FIFO), as a comparison to check the performance of the proposed algorithm. In addition, inspired by the research of Holland (1992), Defersha and Rooyani (2020) and Zhang et al. (2020a), we use improved genetic algorithms (GA) as a typical method of meta-heuristic algorithms for comparison.

**Random.** Select action randomly from $\mathcal{O} = \{O_{ij} \mid \forall i \in N_t, j \in N_R\} \cup \{S, T\} - \widetilde{A}$.

**First In Fist Out (FIFO).** Select the task that can be executed first.

**Most Processing Time Remaining (MPTR).** Select the mission with the least total operation time left.

**Shortest Operation Time (SOT).** Select the action that takes the least time for the optional operation.

**Earliest End Time (EET).** Select the earliest resource to end the assignment.

To verify the superiority of the proposed algorithm over the traditional rules in different scenarios, we simulated the battlefield data at different scales, e.g., $6 \times (3+3+3)$, $20 \times (3+3+3)$, and randomly generated multiple cases. Then the cases are solved separately using the above five dispatching rules and the proposed deep reinforcement learning based algorithm to generate the scheduling scheme and get the completion time as shown in Table 3. For a more visual comparison, we have made Gantt charts of the scheduling results for the 15th case. The target allocation results for our proposed algorithm and the rule-based MPTR method that performed best in this case are shown in Figs. 8 and 9, respectively.

We can see that compared to the five rules, our algorithm obtains optimal results under different scales of data, which is about a twenty percent improvement, fully demonstrating the superiority of the algorithm. On the other hand, compared with the meta-heuristic algorithm GA, the proposed algorithm not only has obvious advantages in the quality of the solution, but also has essential differences in efficiency. The time required for GA to make a better decision is more than 60 min, and it increases exponentially with the increase of problem dimension. Especially when the defense system is subject to saturation attacks, the solution of scheduling plays a decisive role in the success

**Table 3**
Static scheduling results for proposed algorithms and common rules.

| Data size | Index | *Random* | *FIFO* | *MPTR* | *SOT* | *EET* | *GA* | *Proposed* |
|---|---|---|---|---|---|---|---|---|
| $6\times(3+3+3)$ | 1 | 419 | 496 | 397 | 393 | 397 | 341 | **341** |
| | 2 | 482 | 519 | 494 | 339 | 365 | 310 | **310** |
| | 3 | 408 | 449 | 494 | 501 | 434 | 387 | **387** |
| | 4 | 462 | 375 | 448 | 437 | 375 | 337 | **325** |
| | 5 | 539 | 534 | 507 | 460 | 487 | 409 | **409** |
| | 6 | 435 | 456 | 458 | 422 | 416 | 376 | **376** |
| | Average | 457.5 | 471.5 | 466.3 | 425.3 | 412.3 | 360 | **358** |
| $20\times(3+3+3)$ | 7 | 1105 | 1071 | 1091 | 936 | 1092 | 845 | **753** |
| | 8 | 1038 | 1079 | 1084 | 1054 | 1022 | 903 | **869** |
| | 9 | 1003 | 946 | 1088 | 952 | 839 | 631 | **631** |
| | 10 | 1008 | 915 | 901 | 887 | 848 | 778 | **627** |
| | 11 | 1093 | 1081 | 1207 | 1044 | 1092 | 796 | **745** |
| | 12 | 1074 | 1145 | 1071 | 1098 | 1092 | 951 | **790** |
| | Average | 1053.5 | 1039.5 | 1073.7 | 1092 | 995.2 | 817.3 | **735.8** |
| $40\times(6+6+6)$ | 13 | 2126 | 2069 | 1973 | 2044 | 1974 | 1588 | **1359** |
| | 14 | 1748 | 1640 | 1761 | 1775 | 1724 | 1540 | **1260** |
| | 15 | 1891 | 1793 | 1680 | 2035 | 1857 | 1209 | **1144** |
| | 16 | 1508 | 1540 | 1505 | 1421 | 1520 | 1341 | **1083** |
| | 17 | 1632 | 1643 | 1631 | 1567 | 1563 | 1459 | **1038** |
| | 18 | 1205 | 1289 | 1491 | 1048 | 1027 | 1137 | **841** |
| | Average | 1685.0 | 1662.3 | 1673.5 | 1648.3 | 1610.8 | 1379 | **1120.8** |
| $80\times(10+10+10)$ | 19 | 1310 | 1348 | 1771 | 1095 | 1142 | 1020 | **811** |
| | 20 | 1470 | 1520 | 1759 | 1245 | 1216 | 1158 | **954** |
| | 21 | 1516 | 1372 | 1721 | 1056 | 1177 | 973 | **826** |
| | 22 | 1470 | 1337 | 1492 | 1197 | 1098 | 892 | **819** |
| | 23 | 1278 | 1303 | 1733 | 1175 | 1064 | 1064 | **793** |
| | 24 | 1292 | 1480 | 1957 | 1298 | 1174 | 1053 | **901** |
| | Average | 1389.3 | 1393.3 | 1738.8 | 1177.7 | 1145.2 | 1026.7 | **850.7** |

[1] $6\times(3+3+3)$ indicates 6 targets and 9 resources, including 3 each for detection, tracking, and interception.
[2] We use defense data of $40\times(5+5+5)$ to train the strategy network.
[3] For the Random method, we took the average of 100 tests.

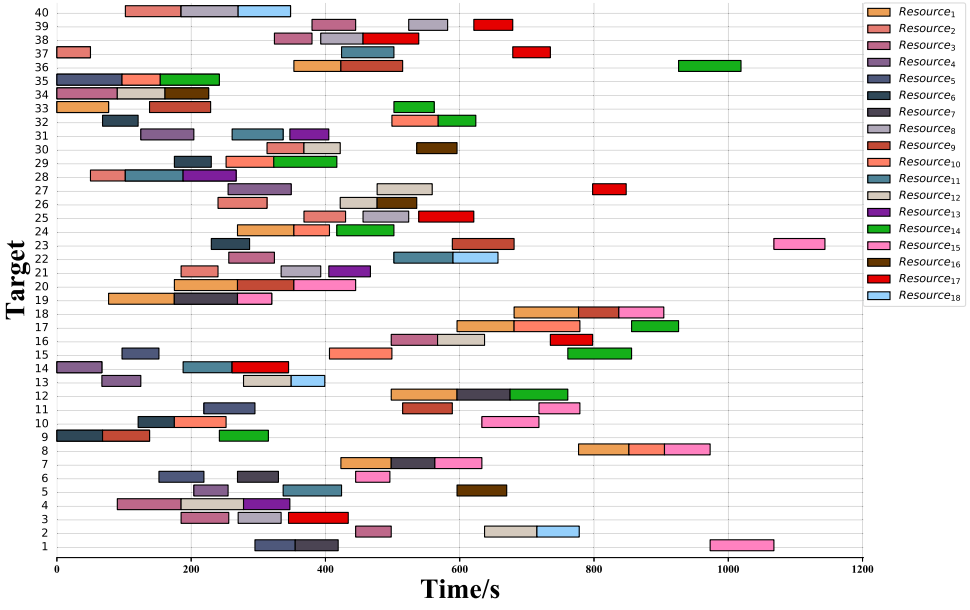



**Fig. 8.** Gantt chart of our algorithm in a $40\times(6+6+6)$ case.

of the defense under the same defense conditions. Using our method can significantly reduce the total scheduling time, improve the defense capability, and fully utilize the system resources. Furthermore, since we generate the policy network under only a single scale of data and use the same network to schedule defense data of different scales, it reflects the strong generalization performance of the algorithm. On the one hand, we need fast planning of defense data at different scales. On the other hand, in dynamic scheduling, changes in the defense environment not only cause changes in resource and target state parameters, but in most cases, the number of resources and tasks also change. Traditional optimization algorithms and general network training take an unacceptably large amount of time and cannot accommodate static scheduling for multiple environments and dynamic scheduling for a single environment. This GNN-based node feature extraction algorithm in our framework effectively solves this problem. It is worth mentioning that our framework is still real-time even in large-scale data scenarios, capable of generating scheduling policies in an average time of 1–2 s. The computational elapsed time is reflected in the system latency described in Section 5.2. Shorter computational elapsed time means less latency, which improves the ability to handle unexpected events.

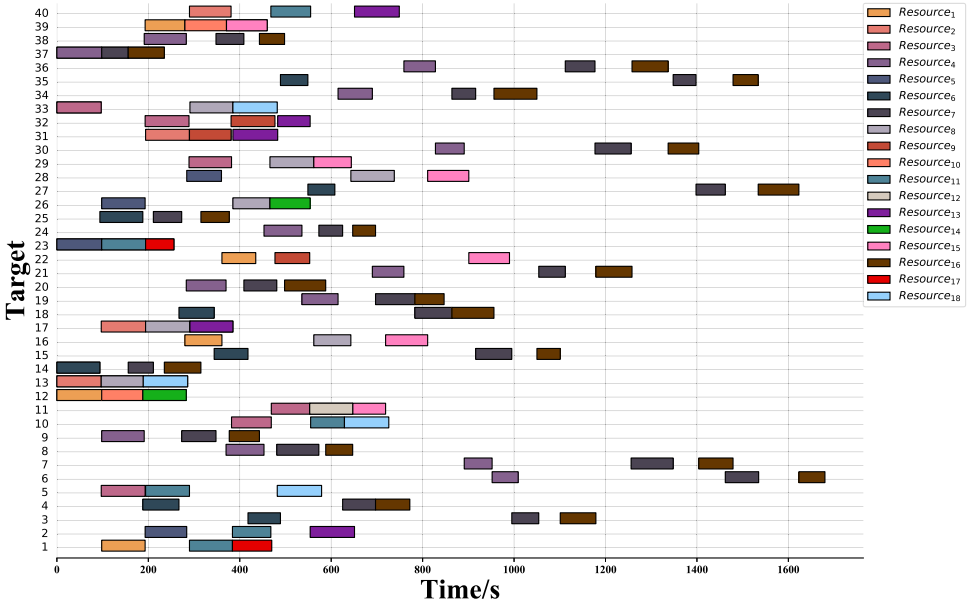



**Fig. 9.** Gantt chart of rule-based method in a $40 \times (6 + 6 + 6)$ case.

**Table 4**
Dynamic scheduling results for proposed algorithms and common rules.

| Data size | Index | *Random* | *FIFO* | *MPTR* | *SOT* | *EET* | *Proposed* |
|---|---|---|---|---|---|---|---|
| $6 \times (3 + 3 + 3)$ | 1 | 410 | 457 | 397 | 397 | 397 | **356** |
| | 2 | 551 | 540 | 540 | 535 | 500 | **448** |
| | 3 | 491 | 439 | 434 | 419 | 444 | **381** |
| | 4 | 569 | 637 | 634 | 447 | 424 | **392** |
| | 5 | 487 | 481 | 482 | 481 | 484 | **435** |
| | 6 | 575 | 447 | 633 | 441 | 468 | **402** |
| | Average | 513.8 | 500.2 | 520 | 453.3 | 452.8 | **402.3** |
| $20 \times (3 + 3 + 3)$ | 7 | 1113 | 1134 | 948 | 1049 | 1039 | **715** |
| | 8 | 1031 | 908 | 1068 | 868 | 974 | **693** |
| | 9 | 1128 | 1074 | 1251 | 1155 | 1161 | **791** |
| | 10 | 976 | 1030 | 987 | 960 | 1038 | **748** |
| | 11 | 1100 | 1048 | 1081 | 995 | 1053 | **753** |
| | 12 | 1532 | 1360 | 1395 | 1400 | 1446 | **1013** |
| | Average | 1146 | 1092.3 | 1121.7 | 1071.2 | 1118.5 | **785.5** |
| $40 \times (6 + 6 + 6)$ | 13 | 2046 | 2153 | 2076 | 2047 | 2018 | **1452** |
| | 14 | 1925 | 1955 | 2006 | 1917 | 1921 | **1433** |
| | 15 | 882 | 1037 | 1141 | 982 | 867 | **608** |
| | 16 | 1350 | 1297 | 1490 | 970 | 1018 | **644** |
| | 17 | 1816 | 1778 | 1861 | 1718 | 1764 | **1182** |
| | 18 | 1448 | 1383 | 1495 | 1262 | 1230 | **818** |
| | Average | 1577.8 | 1597.5 | 1678.2 | 1482.7 | 1469.7 | **1022.8** |
| $80 \times (10 + 10 + 10)$ | 19 | 1332 | 1478 | 1779 | 1103 | 1147 | **906** |
| | 20 | 3214 | 3204 | 3256 | 3136 | 3142 | **2397** |
| | 21 | 2050 | 2273 | 2323 | 1721 | 1820 | **1419** |
| | 22 | 1989 | 1867 | 1733 | 1250 | 1318 | **965** |
| | 23 | 1305 | 1416 | 1574 | 1038 | 966 | **741** |
| | 24 | 1272 | 1465 | 1768 | 1061 | 1114 | **829** |
| | Average | 1860.3 | 1950.5 | 2072.2 | 1551.5 | 1584.5 | **1029.5** |

[1] For the Random method, we took the average of 100 tests.
[2] The system reprogramming period is set to 800 time units.

### 6.3. Results of dynamic scheduling

The previous section analyzed the advantages of our algorithm for static scheduling in detail, demonstrating the superiority of our solution over rule-based approaches at multiple scales, and the ability of the same network to generalize to different environments shows us the potential of the proposed framework for dynamic scheduling. In Section 6.3, we will simulate dynamic scheduling by adding emergency events as well as manual operations (Table 1) to the test data. In addition, to increase the realism of the data, we have refined the setting of the time window compared to static scheduling. As you can see from the Gantt chart, the starting position of the scheduling is not necessarily 0.

Similar to static scheduling, we generate dynamic scheduling data at multiple scales and use rule-based approaches and our algorithm to compute the data to obtain the scheduling results, the total duration of the scheduling is shown in Table 4. As the dynamic scheduling problem can be considered as a combination of multiple static schedules, our algorithm unsurprisingly achieves a large advantage, obtaining the best results in multiple cases at different scales.

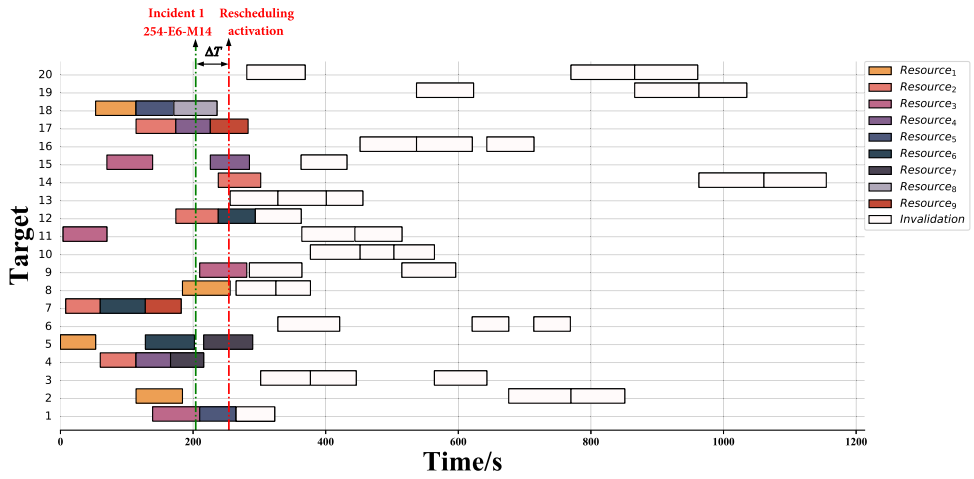



**Fig. 10.** Example of dynamic scheduling (the initial program).

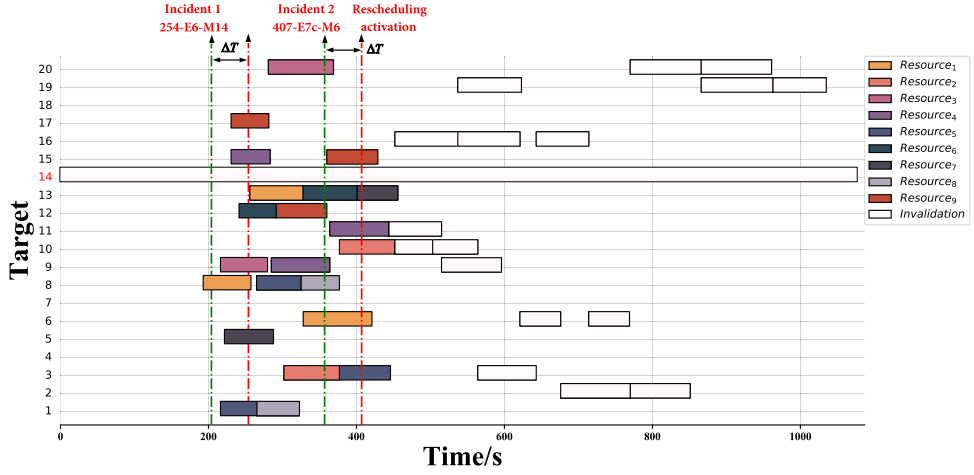



**Fig. 11.** Example of dynamic scheduling (after incident 254-E6-M14).

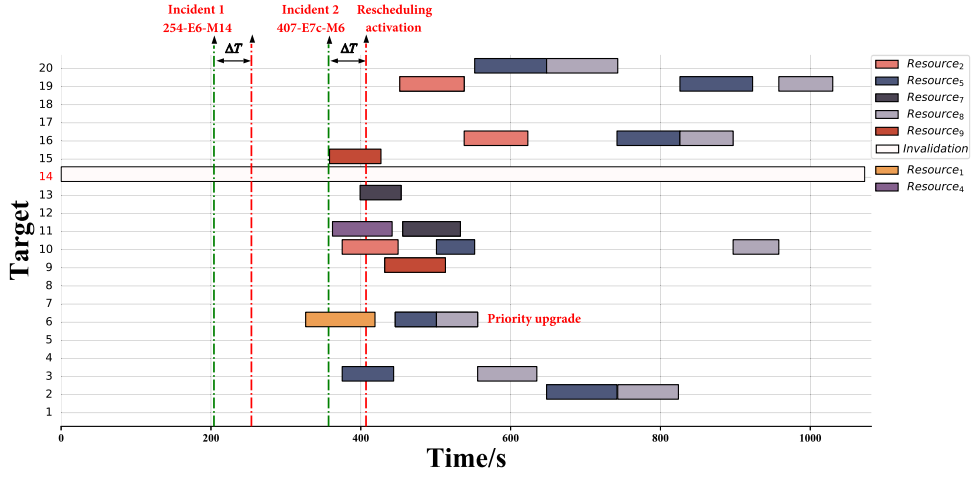



**Fig. 12.** Example of dynamic scheduling (after incident 407-E7c-M6).

The simulations illustrate that our algorithm generates better solutions quickly compared to rule-based algorithms in dynamic scheduling scenarios; on the other hand, the algorithm is robust to the random nature of scheduling data and event-driven data generation.

To visualize the flow of dynamic scheduling, we visualize a simple case in Figs. 10–12 with an initial data scale of 20×(3+3+3). Within the time frame of this dispatch, we set up two random emergency events with system delay time $\Delta T = 50$:

1. 204-E6-M14: Mission 14 ($Target_{14}$) terminates early at moment 254.
2. 357-E7c-M6: Human intervention makes Mission 6 ($Target_6$) a higher priority at moment 357.

As can be seen in Fig. 10, all tasks are assigned to complete in the initial scheduling and the total scheduling time is about 1150. Event 204-E6-M14 occurs at moment 204, and given the delay time, the scheduling center reschedules all tasks with a task start time later than $\Delta T + 204 = 254$. Due to the early termination of Mission 14, the system load was reduced, resulting in a reduction of the total dispatch time to 1080 as shown in Fig. 11. Similarly, the priority of $Target_6$ is raised in the scheduling after moment $\Delta T + 357 = 407$ due to the intervention of the manual operation at moment 357. Thus, in the second rescheduling result in Fig. 12, the two submissions of Mission 6 are given priority compared to the first rescheduling.

### 6.4. Discussion

Key-point defense is the last barrier of a military stronghold, with short response times and dramatic environmental changes, requiring rapid scheduling of multiple levels of defense resources simultaneously. At the same time, constraints such as time windows for resources and targets and the allocation of resources greatly increase the difficulty of the solution compared to general scheduling models.

Compared with the heuristic algorithms widely used for combinatorial optimization problems, our proposed method is almost instantaneous in solving scheduling tasks. Especially in the battlefield environment, the complex and changing combat missions make the general heuristic algorithms completely ineffective, while our method can cope with various emergency events and complete the initial scheduling and rescheduling in a very short time, gaining sufficient time for defense.

Rule-based scheduling methods have been widely used in industrial production. When the scheduling environment is fixed, a specific rule or a combination of rules can achieve a better solution. At the same time, unlike heuristic algorithms, rule-based approaches generate solutions in real time. However, such methods are only a single ranking of a particular feature of the nodes. In contrast, this paper extracts the interrelationships between nodes and complex features of nodes by disjunctive graph and GNN. A rational reward design in DRL allows the agent to rank the optional actions by means of probability functions based on the goal of minimizing the total scheduling time. The comprehensive extraction of node features and direct target-driven training allow our algorithm to achieve superiority over rule-based methods in different environments. Furthermore, our algorithm has good generalization performance and is insensitive to the scale of the data and the ratio of targets to resources, which allows the policy network obtained from one offline training to be applied online to different environments. We demonstrate the above advantages in numerical experiments using a large number of random arithmetic examples of different sizes.

The outcome of a policy-based network's scheduling decisions is driven by the setting of the reward, such as in our model, the network will always choose the scheduling scheme that makes the total makespan as short as possible. But in the actual battlefield environment, there are some complex tactics, in these cases, the commander is willing to sacrifice some scheduling time to complete a certain mission. The human-intelligent collaborative scheduling framework we constructed makes it possible to combine the commander's will with the intelligent policy network. Under this framework, more complex operations can be developed to apply to more environments that are not limited to key-point defense.

Although the proposed method has obvious advantages over heuristic algorithm, rule-based algorithm and simple reinforcement learning algorithm, it still has some limitations. First, this research focuses on building a new human-intelligent collaboration dynamic scheduling framework for emergency response. Although some advanced algorithms, including GNN and DRL, are embedded into the framework after improvement, and better simulation results are obtained than existing methods, more algorithms need to be studied in the future for comparison. Second, compared with the rule-based algorithm, the proposed algorithm has complex structure and long off-line training time, so it needs to further simplify the network and improve the training convergence efficiency. In addition, although the algorithm proposed in the simulation has good generalization performance and can adapt to dynamically changing environments and tasks, it is still necessary to further test the adaptability of the algorithm in more complex environments.

## 7. Conclusions

In this paper, an integrated scheduling model of detection, tracking and interception is proposed and transformed into a sequential decision problem by introducing a disjunctive graph and a graph neural network to extract node features. The PPO-based reinforcement learning algorithm is then applied to the Markov modeled scheduling model. Benefiting from our rational design of rewards, action spaces, etc., the policy network can efficiently allocate resources and quickly decide the scheduling order after acceptable training iterations. Taking into account the complexity and tactical diversity of the key-point defense, we develop a framework of DSHI and propose a hybrid dispatch algorithm for emergency response. Extensive simulation results have shown that our framework has significant advantages over traditional methods in dealing with static and dynamic problems, and is capable of handling a wide range of contingencies to suit complex battlefield environments.

Based on the research in this paper, in future work we can introduce multiple objectives other than the total scheduling time, such as the average resource load, total system working time, etc. In addition, the research on the refinement of DSHI framework and the way of organic integration of human and intelligence is also a valuable research direction. DRL has many successful cases in solving game theory problems, and the commander's intervention in the scheduling strategy is the result of a game between two parties. If the agent can effectively learn and replace the role of humans in the DSHI framework, the scheduling capability of the network will be further enhanced.

## CRediT authorship contribution statement

**Rugang Tang:** Conceptualization, Methodology, Investigation, Formal analysis, Writing – original draft. **Xin Ning:** Conceptualization, Funding acquisition, Resources, Supervision, Writing – review & editing. **Zheng Wang:** Data Curation, Writing – original draft. **Jiaqi Fan:** Visualization, Investigation, Manuscript revision. **Shichao Ma:** Resources, Supervision.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants no. 11772256, Science and Technology on Electromechanical Dynamic Control Laboratory, China, No. 6142601190210, the Open Research Fund of CAS Key Laboratory of Space Precision Measurement Technology, No. SPMT-2022-06, and Doctor Dissertation of Northwestern Polytechnical University, China, No. CX2023037.

**Table A.5**
Algorithm settings.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Layers of feature extraction | 3 | Number of dimension of raw node features | 5 |
| Hidden dim of MLP in fea extract GNN | 64 | Layers of MLP in fea extract GNN | 2 |
| Layers in actor MLP | 2 | Hidden dim of MLP in actor | 32 |
| Layers in critic MLP | 2 | Hidden dim of MLP in critic | 32 |
| Learning rate | 0.02 | Decay ratio of learning rate | 0.9 |
| Clip parameter for PPO | 0.2 | Critic loss coefficient | 1 |
| Policy loss coefficient | 2 | Entropy loss coefficient | 0.01 |

## Appendix. Algorithm settings

See Table A.5.

## References


Defersha, F.M., Rooyani, D., 2020. An efficient two-stage genetic algorithm for a flexible job-shop scheduling problem with sequence dependent attached/detached setup, machine release date and lag-time. Comput. Ind. Eng. 147, 106605.

Deng, Q., Yu, J., Wang, N., 2013. Cooperative task assignment of multiple heterogeneous unmanned aerial vehicles using a modified genetic algorithm with multi-type genes. Chin. J. Aeronaut. 26 (5), 1238–1250.

Hamilton, W., Ying, Z., Leskovec, J., 2017. Inductive representation learning on large graphs. Adv. Neural Inf. Process. Syst. 30.

Han, B.-A., Yang, J.-J., 2020. Research on adaptive job shop scheduling problems based on dueling double DQN. Ieee Access 8, 186474–186495.

Han, B., Yang, J., 2021. A deep reinforcement learning based solution for flexible job shop scheduling problem. Int. J. Simul. Model. 20 (2), 375–386.

Hocaoğlu, M.F., 2019. Weapon target assignment optimization for land based multi-air defense systems: A goal programming approach. Comput. Ind. Eng. 128, 681–689.

Holland, J.H., 1992. Genetic algorithms. Sci. Am. 267 (1), 66–73.

Jia, Y., Bian, W., Huang, K., Liu, C., Lv, X., Yuan, W., 2021. Heterogeneous interception equipment deployment method based on heuristic optimization algorithm. In: 2021 IEEE International Conference on Unmanned Systems. ICUS, IEEE, pp. 449–454.

Kim, H., Chang, Y.-K., 2020. Optimal mission scheduling for hybrid synthetic aperture radar satellite constellation based on weighting factors. Aerosp. Sci. Technol. 107, 106287.

Kipf, T.N., Welling, M., 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Lei, K., Guo, P., Zhao, W., Wang, Y., Qian, L., Meng, X., Tang, L., 2022. A multi-action deep reinforcement learning framework for flexible job-shop scheduling problem. Expert Syst. Appl. 205, 117796.

Li, S.-y., Chen, M., Wang, Y.-h., Wu, Q.-x., 2022. Air combat decision-making of multiple UCAVs based on constraint strategy games. Def. Technol. 18 (3), 368–383.

Li, Z., Xie, J., Zhang, H., Xiang, H., Zhang, Z., 2020. Adaptive sensor scheduling and resource allocation in netted collocated MIMO radar system for multi-target tracking. Ieee Access 8, 109976–109988.

Liu, J.-s., Jiang, W.-z., Dai, J.-j., Liu, T., 2015. Weapon target assignment of key points air-defense under Dynamic Fire Alliance. In: 2015 IEEE International Conference on Mechatronics and Automation. ICMA, IEEE, pp. 415–419.

Liu, L., Li, X., Yan, J., 2013. Key-point air defense fan-shaped deployment with large-dimensional multi-objective multi-constraint group divided optimization. Syst. Eng. Electron. 35 (12), 2513–2520.

Luo, S., 2020. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. Appl. Soft Comput. 91, 106208.

Ma, Y., Wang, G., Hu, X., Luo, H., Lei, X., 2019. Cooperative occupancy decision making of multi-UAV in beyond-visual-range air combat: A game theory approach. Ieee Access 8, 11624–11634.

Miranda, S., Baker, C., Woodbridge, K., Griffiths, H., 2007. Comparison of scheduling algorithms for multifunction radar. IET Radar Sonar Navig. 1 (6), 414–424.

Nie, X., Wang, B., Yan, P., Wu, B., Gao, C., 2021. Key point defense and target threat analysis based on data fusion. In: The 2nd International Conference on Computing and Data Science. pp. 1–3.

Orman, A., Potts, C.N., Shahani, A., Moore, A., 1996. Scheduling for a multifunction phased array radar system. Eur. J. Oper. Res. 90 (1), 13–25.

Park, J., Chun, J., Kim, S.H., Kim, Y., Park, J., 2021. Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning. Int. J. Prod. Res. 59 (11), 3360–3377.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P., 2015. Trust region policy optimization. In: International Conference on Machine Learning. PMLR, pp. 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Sgambato, P., Celentano, S., Di Dio, C., Petrillo, C., 2016. A flexible on-line scheduling algorithm for multifunctional radar. In: 2016 IEEE Radar Conference (RadarConf). IEEE, pp. 1–5.

Shan, G.-l., Xu, G.-g., Qiao, C.-l., 2020. A non-myopic scheduling method of radar sensors for maneuvering target tracking and radiation control. Def. Technol. 16 (1), 242–250.

Summers, D.S., Robbins, M.J., Lunday, B.J., 2020. An approximate dynamic programming approach for comparing firing policies in a networked air defense environment. Comput. Oper. Res. 117, 104890.

Tian, T., Zhang, T., Kong, L., 2018. Timeliness constrained task scheduling for multifunction radar network. IEEE Sens. J. 19 (2), 525–534.

Tuncer, O., Cirpan, H.A., 2022. Target priority based optimisation of radar resources for networked air defence systems. IET Radar Sonar Navig. 16 (7), 1212–1224.

Wang, Y., Pan, L.-p., 2010. Study of mean-entropy models for key point air defense disposition. In: Fuzzy Information and Engineering 2010: Volume I. Springer, pp. 647–656.

Winter, E., Baptiste, P., 2007. On scheduling a multifunction radar. Aerosp. Sci. Technol. 11 (4), 289–294.

Xu, K., Hu, W., Leskovec, J., Jegelka, S., 2018. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826.

Yi, W., Yuan, Y., Hoseinnezhad, R., Kong, L., 2020. Resource scheduling for distributed multi-target tracking in netted colocated MIMO radar systems. IEEE Trans. Signal Process. 68, 1602–1617.

Zhang, G., Hu, Y., Sun, J., Zhang, W., 2020a. An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints. Swarm Evol. Comput. 54, 100664.

Zhang, J., Jiahao, X., 2020. Cooperative task assignment of multi-UAV system. Chin. J. Aeronaut. 33 (11), 2825–2827.

Zhang, T., Li, C., Ma, D., Wang, X., Li, C., 2021. An optimal task management and control scheme for military operations with dynamic game strategy. Aerosp. Sci. Technol. 115, 106815.

Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, P.S., Chi, X., 2020b. Learning to dispatch for job shop scheduling via deep reinforcement learning. Adv. Neural Inf. Process. Syst. 33, 1621–1632.

Zhang, H., Xie, J., Shi, J., Zhang, Z., Fu, X., 2019. Sensor scheduling and resource allocation in distributed MIMO radar for joint target tracking and detection. IEEE Access 7, 62387–62400.

Zhang, H., Xie, J., Zong, B., Lu, W., Sheng, C., 2017. Dynamic priority scheduling method for the air-defence phased array radar. IET Radar Sonar Navig. 11 (7), 1140–1146.

Zhao, X., Zong, Q., Tian, B., Zhang, B., You, M., 2019. Fast task allocation for heterogeneous unmanned aerial vehicles through reinforcement learning. Aerosp. Sci. Technol. 92, 588–594.