Dynamic Scheduling Algorithm for Unmanned Delivery Resources Based on Hierarchical Attention

Jingjing Miao*, Xinghua Li, and Yaoxi Kang

The 15th Research Institute, China Electronics Technology Group Corporation, Beijing, China
Email: mjjblcu@126.com (J.J.M.); 419061969@qq.com (X.H.L.); 17310556670@163.com (Y.X.K.)
*Corresponding author

Manuscript received March 9, 2026; revised March 30, 2026; accepted April 9, 2026; published May 20, 2026

***Abstract*—The vigorous development of the low-altitude economy has led to a core contradiction in the field of unmanned logistics delivery: dynamically changing delivery demands versus limited resource allocation. To address this issue, a dynamic scheduling algorithm for unmanned delivery resources based on hierarchical attention is proposed, aiming to solve the problems of agile cooperative scheduling and dynamic adaptive expansion of unmanned delivery resources in an environment with variable delivery demands and achieve the overall optimization of delivery efficiency. The unmanned delivery resource scheduling task is modeled as a Markov Decision Process (MDP) with the goal of global delivery optimization. A multi-agent reinforcement learning framework is adopted to realize cross-domain cooperative decision-making of unmanned delivery resources, and a hierarchical attention mechanism is designed to support the unified representation of variable-scale cross-domain heterogeneous unmanned clusters. The prioritized experience replay technology is applied and a multi-scale reward function is established to solve the problem of reward sparsity in the multi-agent cooperative environment, which effectively improves the stability of the algorithm training process and the accuracy of the results. Experimental results show that the hierarchical attention mechanism has good multi-dimensional cooperative perception ability of environmental information; in the scenario where unmanned delivery resources can be newly added and expanded, the final task completion rate reaches 0.81, which is 12.5% higher than that of the strategy without expansion. The proposed algorithm provides an efficient solution for dynamic resource scheduling in unmanned logistics while laying a scalable and adaptable technical foundation for broader applications in intelligent transportation systems and smart city logistics.**

***Keywords*—unmanned delivery resource scheduling, hierarchical attention mechanism, multi-agent reinforcement learning, adaptive expansion, dynamic scheduling**

I. Introduction

The low-altitude economy has been designated as a national strategic emerging industry and incorporated into government work reports. This policy endorsement creates unprecedented growth opportunities [1]. Against this background, cross-domain heterogeneous unmanned delivery platforms such as Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) have shown enormous application potential in scenarios such as the "last mile" of logistics delivery and emergency material delivery due to their high flexibility and efficiency [2]. However, the demand for unmanned logistics delivery is highly dynamic and uncertain, while the available unmanned delivery resources have significant constraints in terms of quantity, type and endurance capacity, forming the core contradiction between dynamically changing delivery demands and limited allocation of unmanned delivery resources. Therefore, how to efficiently and flexibly schedule limited and dynamically adjustable unmanned delivery resources under the real-time changing delivery situation has become a core challenge in constructing an intelligent unmanned logistics system.

Traditional scheduling methods rely on static or quasi-static optimization models. Consequently, they struggle to adapt to real-time task dynamics and elastic resource allocation requirements [3]. Although some studies adopt meta-heuristic or heuristic rules such as genetic algorithm and A* algorithm for optimization, they still have limitations such as low solution efficiency and insufficient adaptability when dealing with large-scale and high-complexity resource dynamic scheduling problems. Reinforcement learning [4], especially Multi-Agent Reinforcement Learning (MARL) [5], provides a new solution to such complex sequential decision-making problems by maximizing the long-term reward through continuous interaction between agents and the environment. MARL has made remarkable progress in fields such as game theory and autonomous driving, and its framework of "centralized training and decentralized execution" is particularly suitable for cooperative decision-making scenarios.

However, existing MARL methods often lack scalability due to fixed network structures when facing scenarios with dynamically changing numbers of agents [6]. Most algorithms require a fixed number of agents and a definite input dimension, which leads to severe restrictions on scheduling flexibility in logistics systems with elastic resource scaling. The "UAV-UGV cooperative scheduling" still has many challenges in terms of dynamic environment adaptability, large-scale cluster scalability and real-time scheduling efficiency [7]. In addition, how to realize the unified representation of variable-scale agent clusters and carry out effective cooperative decision-making on this basis is an urgent technical bottleneck to be broken through in the current field of multi-agent scheduling.

To address the above challenges, this paper proposes a dynamic scheduling algorithm for unmanned delivery resources based on a hierarchical attention mechanism. The unmanned delivery resource scheduling task is modeled as a Markov Decision Process (MDP), and multi-agent reinforcement learning technology is used to solve the cross-domain cooperative scheduling problem under variable demands. The core innovation lies in the construction of a policy-value network integrated with hierarchical attention. Through lower-layer self-attention encoding and upper-layer global attention aggregation, the network realizes the dynamic and unified representation of variable-scale unmanned clusters, enabling the algorithm to adapt to the

dynamic changes in the number of agents without changing the core parameters.

II. Dynamic Scheduling Algorithm Based on Hierarchical Attention Mechanism

A. Problem Modeling and Algorithm Framework

The dynamic scheduling problem of cross-domain heterogeneous unmanned delivery resources follows the Markov property, that is, the system's transition to the next state only depends on the current state and has nothing to do with the historical state. Therefore, it is modeled as a Markov Decision Process. At each decision moment, the system needs to assign appropriate delivery tasks or action instructions to unmanned delivery resources according to the current global situation information (including pending delivery tasks, resource status, environmental parameters, etc.).

Unmanned logistics delivery scheduling involves sequential decision-making. At each time step, the system: (1) observes the environmental state, (2) executes scheduling actions, and (3) influences subsequent states and long-term performance. The Markov Decision Process provides a rigorous mathematical modeling framework for such problems, and its core assumption is that the future state only depends on the current state and action, i.e., satisfying the Markov property. In the unmanned logistics delivery scenario, the resource distribution, task queue, power status and other information at the current moment completely determine the evolution trend of the system at the next moment, thus conforming to the basic assumptions of MDP. By modeling the scheduling problem as an MDP, it can be transformed into a sequential decision-making problem of seeking the optimal strategy in an uncertain environment, which is convenient for solving by using methods such as reinforcement learning.

Traditional scheduling optimization methods such as integer programming and heuristic rules often face challenges such as complex modeling, low solution efficiency and poor adaptability when dealing with high-dimensional, dynamic and highly random unmanned logistics scheduling problems. Reinforcement learning adaptively learns scheduling strategies with the goal of optimizing cumulative rewards through interactive trial and error between agents and the environment, which is particularly suitable for dynamic and uncertain environments [8]. As a classic algorithm in reinforcement learning tasks, the multi-agent reinforcement learning method based on policy gradient has the advantages of being applicable to continuous action spaces, optimizing policy gradients and value functions [9], and adapting to dynamic-scale agent scheduling, which can realize efficient, scalable and adaptive cooperative strategy learning in dynamic environments.

To effectively solve the problem of dynamic scalable scheduling of cross-domain heterogeneous multi-agent resources, a Hierarchical Attention Multi-Agent Reinforcement Learning (HAMARL) algorithm is proposed on the basis of the deterministic policy gradient algorithm, and its overall architecture is shown in Fig. 1. In the HAMARL framework, each unmanned delivery resource is modeled as an unmanned agent, and all agents share the same set of policy-value network parameters. These agents conduct cooperative learning with the goal of maximizing the long-term cumulative reward through continuous interaction with the simulated unmanned delivery environment until a certain level of intelligence is achieved.

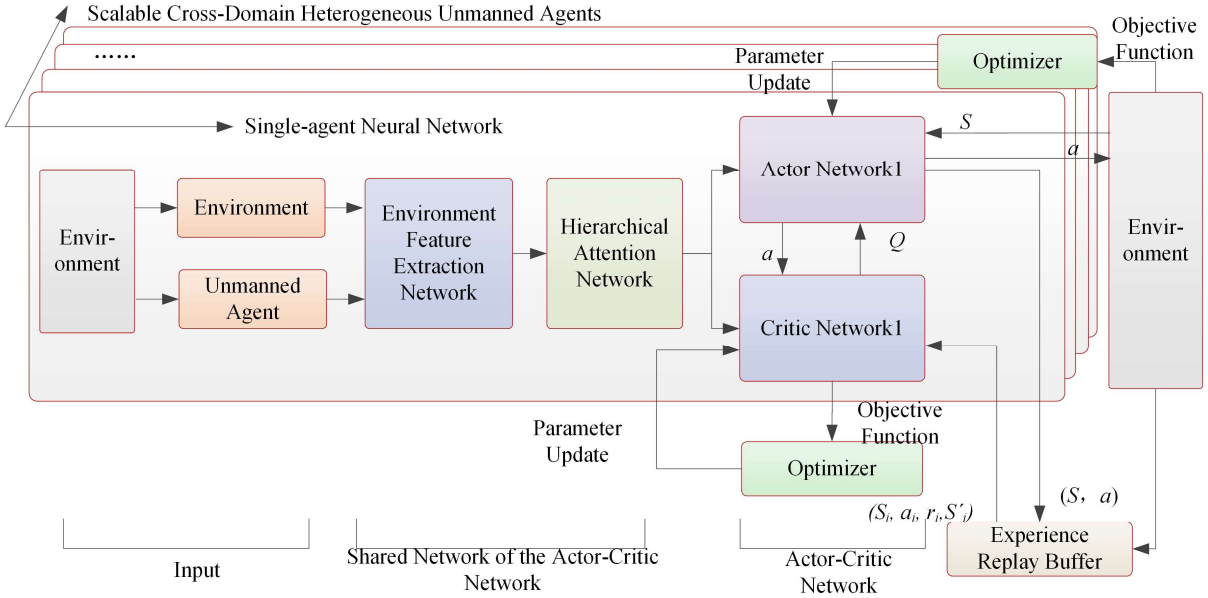

Fig. 1. Framework diagram of resource dynamic scheduling algorithm.

The core design of HAMARL adopts a parameter sharing mechanism. Specifically, each agent instantiates an identical network copy. In each copy, the environmental feature extraction network realizes the fusion and sharing of scheduling environmental information for multiple unmanned intelligent delivery resources, the hierarchical attention mechanism realizes global information integration through layer-by-layer encoding, and the policy-value network realizes cooperative material delivery between heterogeneous agents. The input of the model is real-time scheduling state information, and the output is the action policy and state value estimation of each agent. This design not only significantly reduces the model complexity and training difficulty, but more importantly, by dynamically

adjusting the number of network instances, the elastic expansion and contraction of scheduling resources can be directly realized, thus natively supporting the extended access and system reconstruction of unmanned delivery resources at the architecture level.

B. Hierarchical Attention Mechanism

Aiming at the problem of fixed resource scale representation under variable unmanned delivery demands, HAMARL designs a cross-domain heterogeneous multi-agent delivery environment representation network based on a hierarchical attention mechanism (Fig. 2), which provides a bottom-up environmental state modeling path and dynamically and agilely adapts to the complex and variable delivery capacity demands. The attention mechanism is used to distinguish the degree of attention to information [10], and the cross-domain heterogeneous multi-agent representation based on the hierarchical attention mechanism can further realize the extraction from the dynamic interaction of local entities to regional cooperation and then to global decision-making information [11, 12], enabling the algorithm to have both detailed expression ability and semantic compression and global situation understanding ability, and realizing the dynamic adaptive scalability of cross-domain heterogeneous multi-agent scale at the representation coding layer [13]. The hierarchical attention network structure includes 2 key layers: the lower-layer self-attention coding layer and the upper-layer global attention aggregation layer.

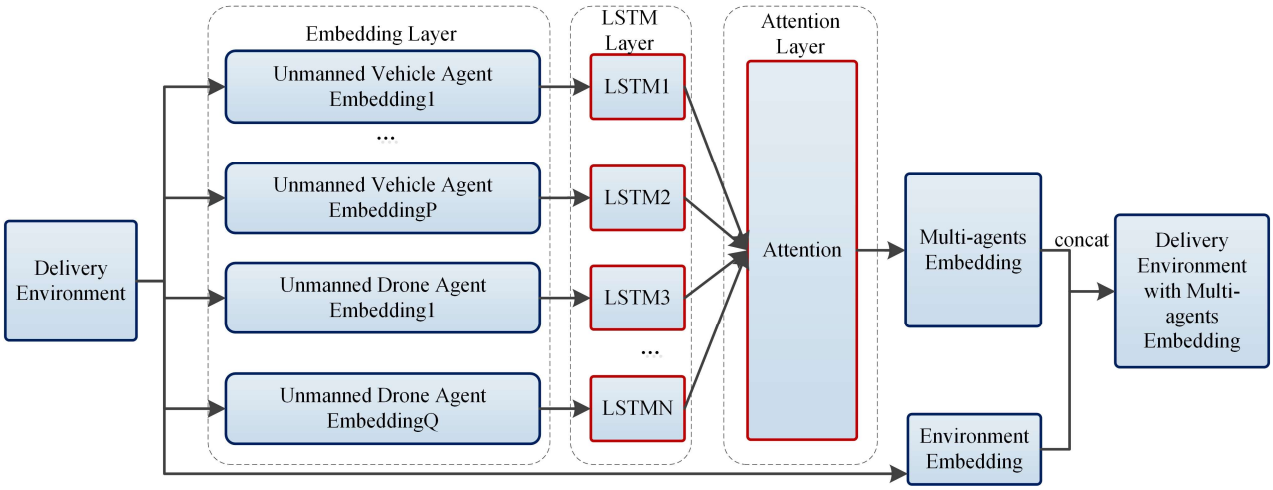

Fig. 2. Hierarchical attention network diagram.

The upper-layer global attention aggregation layer receives the embedded vectors of all agents, calculates the correlation weights between different agents by using the cross-attention mechanism, and then aggregates the high-dimensional feature information of cross-domain heterogeneous agents with emphasis to realize the deep fusion of global situation and cooperative perception between agents. This design ensures that the decision-making of each agent can be carried out under the guidance of global information, realizing real cross-domain cooperative decision-making.

Through this hierarchical coding structure, the network can output a fixed-dimensional global representation independent of the number of entities. Within a certain range, no matter how the number of unmanned agents participating in scheduling changes, the core parameter scale of the network remains unchanged, thus fundamentally ensuring the strong adaptability of the algorithm to dynamic resource changes.

1) Lower-layer attention mechanism based on unidirectional long short-term memory network

In the single-agent input layer, N unmanned delivery platforms are represented as N embedded vectors of unmanned agent entities, mainly including 11-dimensional features such as position coordinates (2D), remaining battery (1D), load status (1D), local task queue (5D), current speed (1D) and device type (1D), which are used as the input information of the lower attention layer (Fig. 3).

Each single-agent hidden expression module mainly contains a unidirectional Long Short-Term Memory (LSTM) network. Assume that the single-agent information at each moment is p_{it}, where p_{it} represents the information of the i-th agent at the t-th moment, and T is the total number of time steps. The vector of each unmanned single agent is taken as the input of the LSTM at each moment, and then the forward hidden state $\overrightarrow{h_{it}}$ of the LSTM network at the t-th moment is obtained. The specific calculation formula is as follows:

$$h_{it} = \overrightarrow{h_{it}} = \overrightarrow{LSTM}\left(p_{it}\right), t \in [1, T] \tag{1}$$

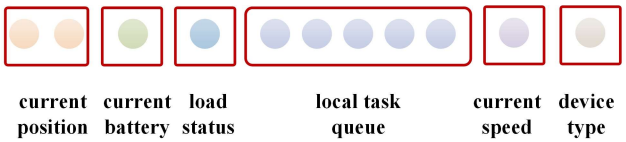

Fig. 3. Unmanned delivery agent embedding.

2) Upper-layer attention mechanism based on attention network

Each agent obtains its fixed-length vector representation by using the LSTM self-attention mechanism. The simple splicing of the vectors of all agents can represent the global agent information, but it cannot distinguish the differences and effective information between cross-domain heterogeneous agents, and the length of the total vector information after splicing is limited, which cannot allow the effective expansion of the agent scale. Therefore, the attention mechanism is introduced to extract the information of great significance in the upper layer, so as to make use of

the information of each agent with emphasis. The specific calculation formulas for the upper-layer attention mechanism are presented in Eqs. (2–4):

$$u_{it} = \tanh\left(W_w h_{it} + b_w\right) \tag{2}$$

$$a_{it} = \frac{\exp\left(u_{it}^T u_w\right)}{\sum_t \exp\left(u_{it}^T u_w\right)} \tag{3}$$

$$s_i = \sum_i a_{it} h_{it} \tag{4}$$

Specifically, a fully connected neural network is used to take u_{it} as the hidden representation of h_{it}, and then the softmax function is used to normalize the importance weight a_{it}. Among them, W_w, b_w and u_w are hyperparameters, which are randomly initialized during the training process. After that, the representation vector s_i of cross-domain heterogeneous multi-agents is calculated according to the weighted sum.

By applying the attention mechanism at different levels, the resource scheduling algorithm can better focus on the important parts of the unmanned single agent, which helps to reduce the interference of noise information, make the algorithm more focused on key effective information, and thus improve the accuracy and efficiency of the algorithm. The attention mechanism is used to fuse the information of an indefinite number of unmanned agents, realizing the dynamic scalability of cross-domain heterogeneous multi-agent scale at the representation coding layer.

C. Prioritized Experience Replay Technology

In the multi-agent training process, the quality distribution of samples in the experience replay buffer is uneven, and uniform sampling is likely to lead to low training efficiency and unstable convergence. For this reason, the prioritized experience replay technology is introduced. First, after the agent interacts with the environment to generate experience samples, the Temporal Difference error (TD-error) is calculated. The absolute value of TD-error reflects the prediction deviation of the current strategy for the sample, i.e., its learning value, which is used as a metric for the importance of the sample. Second, a priority-based sampling strategy is adopted to make samples with larger TD-error obtain a higher sampling probability. This method prompts the model to give priority to learning experiences with rich information or large prediction deviations, thus effectively accelerating the policy optimization process and improving the training stability and the performance of the final policy.

1) Priority design strategy

HAMARL adopts a proportional prioritization scheme, using the absolute value of TD-error as the basic metric of sample priority. For each experience sample i, its priority calculation formula is: $p_i = |\delta_i| + \varepsilon$. Among them, δ_i is the TD-error of sample i, and $\varepsilon = 10^{-6}$ is a very small positive number to prevent the priority from being zero and the sample from never being sampled. The sampling probability of sample i is proportional to its priority, as given in Eq. (5):

$$P(i) = \frac{p_i^\lambda}{\sum_k p_k^\lambda} \tag{5}$$

where $P(i)$ represents the sampling probability of sample i, and λ is the priority adjustment factor, which controls the influence degree of priority.

2) Importance sampling weight design

To correct the deviation introduced by priority sampling, an Importance Sampling (IS) weight correction mechanism is adopted, and the importance sampling weight formula is as follows:

$$w_i = \left(\frac{1}{K} \cdot \frac{1}{P(i)}\right)^\beta \tag{6}$$

where K is the capacity of the experience replay buffer, $P(i)$ is the sampling probability of sample i, and β is the importance sampling adjustment factor. β increases linearly from the initial value $\beta_0 = 0.4$ to the end value $\beta_T = 1.0$ during the training process, as expressed in Eq. (7):

$$\beta(i) = \beta_0 + (1 - \beta_0) \cdot \frac{t}{T} \tag{7}$$

where T is the total number of time steps.

This design makes the early stage of training focus on deviation correction (small β), and the later stage of training gradually transitions to unbiased estimation ($\beta \to 1.0$). In actual training, the importance sampling weights are normalized to ensure the stability of the gradient update amplitude and prevent training divergence, as shown in Eq. (8):

$$w_{i'} = \frac{w_i}{\max_j w_j} \tag{8}$$

3) Complete algorithm process

The complete implementation of prioritized experience replay mainly includes four stages:

In the experience storage stage, each experience sample is stored in the form of a tuple (s_t, a_t, r_t, s_{t+1}), and the initial priority of the sample is recorded at the same time:

$$p_i = |\delta_i|_{\text{initial}} \tag{9}$$

In the priority update stage, the priority of the corresponding sample is updated after each sampling and TD-error calculation:

$$p_i \leftarrow |\delta_i|_{\text{new}} + \varepsilon \tag{10}$$

In the sampling stage, a small batch of samples is extracted from the replay buffer according to the probability distribution $P(i)$, and the importance sampling weight $w_{i'}$ of each sample is calculated at the same time.

In the loss calculation stage, the importance sampling weight is introduced into the value function loss:

$$L(\theta) = \frac{1}{B}\sum_{i=1}^{B} w_{i'} \cdot \delta_i^2 \tag{11}$$

where $w_{i'}$ is the importance sampling weight, B is the batch size.

D. Multi-Scale Reward Function Design

A multi-scale reward function design is adopted in the training process. The multi-scale reward function is a reward calculation function that the environment feeds back to the model value network after the algorithm model gives and executes the actions of the unmanned agents. Reward design is an important part of reinforcement learning problems, and a good reward function can make unmanned agents learn better cooperative behaviors. The total reward of each agent is the weighted sum of local reward and global reward:

$$R_{total} = \alpha R_{local} + \beta R_{global} \tag{12}$$

where R_{total} represents the total reward for each agent; R_{local} denotes the local reward, R_{global} represents the global reward. the reward function weights are optimized through experiments and determined by traversing in the interval [0, 1], with the local weight $\alpha = 0.6$ and the global weight $\beta = 0.4$, which are used to balance individual and global goals, both promoting agents to optimize their own behaviors and contributing to the overall efficiency.

1) Local reward function

The local reward function focuses on the performance indicators of a single agent and is used to optimize individual behaviors. By designing the local reward function, the corresponding agent is guided to complete tasks as soon as possible. The specific description of the components of the local reward function is shown in Table 1.

The local reward function formula is as follows:

$$R_{local} = \omega_1 R_{time} + \omega_2 R_{failure} + \omega_3 R_{deliver} + \omega_4 R_{remain} \tag{13}$$

where the weight coefficients are $\omega_1 = -0.5$, $\omega_2 = -2.0$, $\omega_3 = +1.0$, $\omega_4 = -0.2$. The weight ratio of individual time spent to the number of failures is 1:2, which indicates that the algorithm pays more attention to avoiding failures rather than simply pursuing speed, reflecting the principle of reliability first. The weight of the quantity of delivered supplies is high (+1.0), which directly motivates agents to complete more tasks; the weight of the remaining quantity of supplies is low (-0.2), which is used to slightly penalize unfinished tasks, with the global delivery as the overall goal to prevent agents from ignoring long-term tasks excessively.

Table 1. Local reward function items

Reward Item	Symbol	Unit	Direction	Explanation
Individual time spent	R_{time}	1/minute	negative	Encourages the agent to complete delivery tasks quickly, with shorter task durations yielding higher rewards.
Number of failures	$R_{failure}$	1/time	negative	Penalizes the agent for failures (e.g., low battery, collisions), emphasizing algorithm reliability.
Quantity of delivered supplies	$R_{deliver}$	1/unit volume	positive	Rewards the agent for successfully delivering supplies.
Remaining quantity of supplies to deliver	R_{remain}	1/piece	negative	Penalizes incomplete delivery tasks to prevent task backlog.

2) Global reward function

The global reward function synthesizes the rewards obtained from the actions of all agents, focuses on the overall delivery efficiency of the algorithm, and is used to promote cross-domain cooperative delivery of heterogeneous agents. The specific description of the components of the global reward function is shown in Table 2.

Table 2. Global reward function items

Reward Item	Symbol	Unit	Direction	Explanation
Delivery cost	R_{total_cost}	1/cost unit	negative	Penalizes total system costs (overall energy consumption, route length) to encourage economic efficiency.
Delivery supply target	R_{total_task}	1/goal achieved	positive	Rewards the accomplishment of algorithm delivery targets to enhance overall throughput.

The global reward function is set as follows:

$$R_{global} = \omega_5 R_{total\_cost} + \omega_6 R_{total\_task} \tag{14}$$

where $\omega_5 = -0.8$, $\omega_6 = +1.5$. The ratio of delivery cost to delivery supply target is $|\omega_5| : |\omega_6| = 0.8 : 1.5 \approx 1 : 1.875$, which indicates that when balancing economy and efficiency, the algorithm is more inclined to improve the overall task completion volume, but cost control still occupies an important position to avoid resource waste.

3) Reward training

In the experiment, 5000 steps of data collection were carried out, and the reward curve during the experimental training process is shown in Fig. 4 (for the convenience of evaluation and comparison, the reward is scaled to the interval [0, 100]).

It can be seen from Fig. 4 that the reward value obtained

by the model is very low in the early stage of training because the random strategy is difficult to meet the delivery demand. With the increase of the number of model training times, the model gradually learns to choose the appropriate strategy. The simulation reward basically shows an upward trend during the whole training period of the model. In the later stage of training, the change of the reward value is not obvious and tends to be stable. On the one hand, the strategy tends to be stable in the later stage of training and reaches an approximate Nash equilibrium, which is difficult to achieve more improvement; on the other hand, to obtain a more robust strategy, the learning rate of the Adam optimizer is linearly and automatically adjusted during the training process in this paper. By dynamically adjusting the learning rate, the amplitude of parameter update is adjusted to gradually decrease with the number of policy training steps, which makes it easier to converge to the global optimum in the later stage of training.

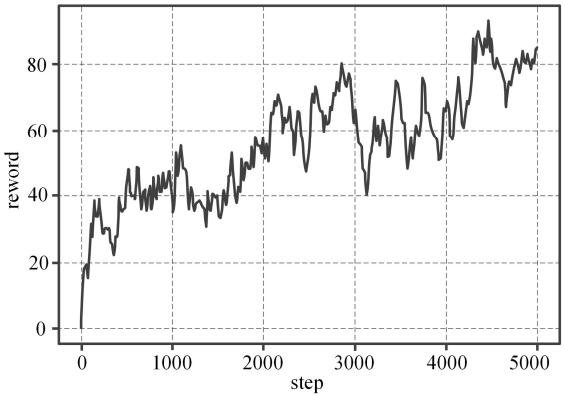

Fig. 4. Training reward information.

III. Experiments and Result Analysis

A. Experimental Environment

To comprehensively evaluate the algorithm performance, an unmanned logistics delivery simulation environment based on the OpenAI Gym standard is developed, which simulates a typical closed community logistics delivery network with realistic factors such as dynamic task generation and environmental disturbance.

In the task generation rules, the generation of each task follows a Poisson process, with an average of 2.5 tasks generated per unit time (minute). The task types are divided into "urgent" and "ordinary", of which urgent tasks account for 20% and must be completed within 30 min, otherwise they are regarded as failed. The starting and ending points of tasks are randomly selected among all delivery nodes, and the task distance follows a uniform distribution.

The dynamic changes of the environment are mainly reflected in 3 aspects: traffic congestion, weather changes and random events. Each road section has a 5% probability of entering a congested state in each simulation step, with a duration of 10~20 min, during which the speed of agents passing through the road section is reduced by 50%. The system switches the weather state according to the preset weather sequence (sunny/rainy/snowy) every 60 min. Rainy and snowy days introduce an influence coefficient of 0.2 respectively to adjust the endurance and speed of agents. Random events mainly include road construction and temporary blockage, which are triggered every hour with a 5% probability, affecting the traffic capacity of local paths.

The hardware and software environment of this experiment is as follows:

- Central Processing Unit (CPU): Intel® Xeon® Gold 5218R;
- Random Access Memory (RAM): 128 G;
- Graphics Processing Unit (GPU): NVIDIA RTX 3090*2;
- Software development language and framework: Python3.7 + tensorflow1.15.5;

B. Ablation and Comparison Experiments

1) Hierarchical attention ablation experiment

To verify the effectiveness of each component in the hierarchical attention architecture, an ablation experiment for the attention layer is designed under the same simulation environment and delivery demand. Based on the ablation experiment results (Table 3), the following conclusions can be drawn:

Table 3. Comparison of ablation experiment results for hierarchical attention

Model Variants	Task Completion Rate	Training Stability	Convergence Steps	Performance Degradation
Complete model	0.81	high	4530	-
Remove upper attention layer	0.74	medium	4640	8.6%
Remove lower attention layer	0.76	low	4750	6.2%
Remove dual attention layers	0.68	medium	4910	16.0%

(1) Core role of the upper attention layer: After removing the upper attention layer, the performance degrades by 8.6%, indicating that global information integration is crucial for cooperative decision-making. The lack of global attention leads agents to fall into local optimization, resulting in task allocation conflicts and resource competition.

(2) Basic function of the lower attention layer: After removing the lower attention layer, the performance degrades by 6.2%, indicating that local feature extraction is the premise of effective representation. In the alternative experiment, replacing LSTM with a fully connected network leads to a 13.6% performance degradation, indicating that LSTM can effectively capture the temporal dependence of state sequences.

(3) Optimization effect of hierarchical attention: When both attention layers are removed, the task completion rate drops to 0.68 with a performance degradation of 16.0%. It can be seen that the performance degradation of removing both attention layers is significantly higher than the superposition effect of removing them separately (the performance degradation of removing the upper layer alone is 8.6% and that of removing the lower layer alone is 6.2%, with a sum of 14.8%), which proves that the hierarchical attention mechanism has good cooperative perception ability of environmental information, and the interaction between the upper and lower attention layers can further improve the

efficiency of feature fusion, so the overall performance degradation range is larger than the superposition effect of removing them separately.

2) Analysis of dynamic resource scalability

To evaluate the algorithm's ability to support dynamic adaptive expansion of resources, 2 resource expansion strategies are set for comparison: no new resource expansion (the initial resource upper limit remains unchanged throughout the simulation) and new resource expansion (the HAMARL algorithm dynamically introduces new unmanned delivery resources during the simulation).

The comparison of real-time task completion rates in the two scenarios is shown in Fig. 5. Experimental results show that after adding new unmanned delivery resources, the real-time task completion rate with new resource expansion gradually rises, and the final task completion rate reaches 0.81, which is 12.5% higher than that of the strategy without resource expansion.

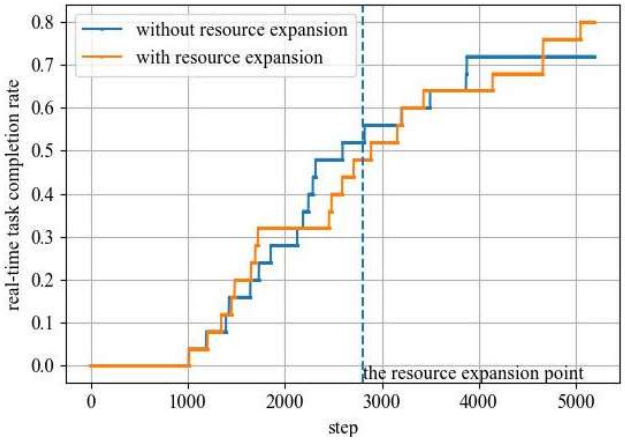

Fig. 5. Real-time task completion rate with/without resource expansion.

This result fully proves that the algorithm in this paper can not only agilely schedule existing resources, but also effectively expand and integrate new resources, significantly alleviate the performance bottleneck caused by limited initial resources, and greatly enhance the adaptability of the unmanned delivery system in complex and variable task environments.

3) Comparative analysis with mainstream attention mechanisms

To further clarify the theoretical advantages of the hierarchical attention mechanism proposed in this paper in the dynamic resource scheduling scenario, this section compares it with two types of mainstream attention methods: Graph Attention Networks (GAT) and Actor Attention Critic (AAC), focusing on 2 dimensions: dynamic adaptability and computational complexity.

a) Dynamic adaptability comparison

Graph Attention Networks usually model agent relationships based on static graph structures, and their adjacency matrix needs to be pre-defined or dynamically constructed. In scenarios with frequent changes in the number of agents, the dynamic reconstruction of graph structures will introduce additional computational overhead and training instability. Although the Actor Attention Critic introduces the attention mechanism to realize the interaction between agents, the network input dimension is usually fixed, which is difficult to directly support the dynamic increase or decrease of the number of agents, and needs to be processed by filling or truncation, affecting the consistency and efficiency of representation.

This paper realizes a fixed-dimensional global representation independent of the number of agents through the double-layer attention structure of "lower-layer self-attention encoding + upper-layer global attention aggregation". The lower-layer self-attention processes each agent independently, and the upper-layer attention dynamically fuses the feature information of different numbers of agents without reconstructing the network structure or adjusting the input dimension, thus natively supporting the elastic change of the number of agents at the architecture level.

b) Complexity calculation and performance analysis

Under the same delivery background, the number of unmanned delivery agents is set as $n = 25$, and the feature dimension of each agent is set as $d = 12$, with specific information shown in Fig. 3.

Table 4. Comparison of task completion rate and computational complexity across different attention networks

Method	Task Completion Rate	Computational Complexity (per layer)	Explanation
GAT	0.79	$O(n^2 d)$	Compute attention weights between each pair of agents, designing fully connected interactions.
AAC	0.77	$O(n^2 d)$	Similar to GAT, global attention calculates correlations among all agents.
HAMARL (lower layer)	-	$O(nd^2)$	Self-attention operates on the features within a single agent and does not increase significantly with the number of agents.
HAMARL (upper layer)	-	$O(nd)$	Global attention aggregates encoded agent vectors and computes weights based on a fixed-dimensional context.
HAMARL (dual-layer)	0.81	$O(nd^2)$	$\max(O(nd^2), O(nd))$

The comparison of dynamic scheduling algorithms integrated with different attention networks is shown in Table 4. Under the same simulation environment, the task completion rate of HAMARL exceeds that of GAT and AAC, indicating that HAMARL is superior in local single-agent representation and global multi-agent aggregation. To ensure the real-time computing capability of HAMARL, a comparison is made with the of GAT and AAC in terms of computational complexity. It can be seen from the table that HAMARL has obvious computational efficiency advantages when the number of agents is large, and is especially suitable for real-time high-concurrency scheduling scenarios of large-

scale unmanned clusters.

IV. Conclusion

Aiming at the problem of dynamic resource scheduling in unmanned logistics delivery, a dynamically scalable scheduling algorithm for unmanned delivery resources integrated with a hierarchical attention mechanism is proposed, and its effectiveness is verified through theoretical and experimental research. The main research results are as follows:

The reinforcement learning algorithm based on the hierarchical attention mechanism is applied to the unmanned delivery scenario, which successfully solves the problems of unified representation and agile decision-making of variable-scale unmanned clusters. In the resource expansion scenario, the algorithm increases the task completion rate to 0.81, which is 12.5% higher than that of the strategy without expansion. Compared with other dynamic representation methods, it not only realizes the unified representation of dynamic-scale agent clusters through the double-layer structure, but also has significant computational efficiency advantages, providing a scalable and efficient algorithm foundation for the real-time high-concurrency unmanned delivery resource scheduling system. Ablation experiments show that the hierarchical attention mechanism structure contributes significantly to the improvement of system performance: removing the upper attention layer leads to an 8.6% drop-in task completion rate, removing the lower attention layer leads to a 6.2% performance degradation, and removing both attention layers leads to a 16.0% performance degradation, which proves the key role of double-layer attention in cooperative decision-making.

The adopted prioritized experience replay technology greatly optimizes the training process. This technology reduces the algorithm convergence steps to 12800 steps, which is more than 43.9% faster than the benchmark model, and the training stability is significantly improved, providing a reliable guarantee for the efficient learning of complex multi-agent systems.

The research in this paper provides a feasible solution for the dynamic resource scheduling problem, and its technical framework can be further extended to fields such as military logistics, intelligent command and emergency rescue, with broad application prospects. Future work will focus on the lightweight design of the algorithm and its deployment and application in more diverse scenarios.

In practical deployment, the algorithm demonstrates strong feasibility: the hierarchical attention mechanism supports dynamic resource scaling without retraining; it has low computational complexity to meet real-time scheduling requirements; and centralized training with decentralized execution reduces communication overhead. However, challenges remain: partial observability (e.g., sensor noise, communication delays) may degrade performance; and the simulation-to-reality gap requires domain randomization or fine-tuning. Future work will focus on sim-to-real transfer learning, lightweight hardware acceleration, and robustness enhancement.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Jingjing Miao proposed the research idea, designed the algorithm and wrote the main body of the paper; Xinghua Li built the experimental simulation environment and completed the ablation experiments; Yaoxi Kang carried out the comparison experiments and analyzed the experimental data; all authors revised and approved the final version of the paper.

Acknowledgment

The authors would like to thank the technical support team of the 15th Research Institute of China Electronics Technology Group Corporation for their valuable suggestions and help in the research process.

References

[1] W. Yang, S. Li, G. Luo *et al.*, "A real-time human–machine–logistics collaborative scheduling method considering workers' learning and forgetting effects," *Applied System Innovation*, vol. 8, no. 2, pp. 1–25, 2025.

[2] H. C. Wei, J. M. Shi, Z. Liu *et al.*, "Survey on drone delivery mode and routing for last-mile delivery," *Computer Systems & Applications*, vol. 32, no. 09, pp. 1–18, 2023. (in Chinese)

[3] Y. F. Rao, Z. H. Chen, W. K. Fang *et al.*, "Research on simulation based dynamic scheduling strategy of the logistics distribution system for assembly lines in engineering machinery," *Journal of Physics: Conference Serie*s, vol. 2101, no. 1, pp. 1–8, 2021.

[4] Y. R. Chen, K. L. Liu, M. L. Ran, "Real-time optimization of instant meal delivery based on deep reinforcement learning," *Computer Engineering*, vol. 51, no. 09, pp. 328–339, 2025. (in Chinese)

[5] A. Arishi, P. Ahuja, "Multi-agent reinforcement learning for truck–drone routing in smart logistics: A comprehensive review," *Computers and Electrical Engineering*, vol. 127, no. Part A, p. 110529, 2025.

[6] C. Ding, "Scalable multi-agent reinforcement learning under dynamic agent scale: Research and implementation," M.S. thesis, School Inf. Softw. Eng., Univ. Electron. Sci. Technol. China, Chengdu, China, 2025. (in Chinese)

[7] Y. F. Liu, C. Li, Z. Wang *et al.*, "Research progress on multi-agent deep reinforcement learning and scalability," *Computer Engineering and Applications*, vol. 61, no. 04, pp. 1–24, 2025. (in Chinese)

[8] J. Song, Z. L. Wang, "Multi-objective multi-agent deep reinforcement learning method based on value decomposition," *Computer Engineering*, vol. 49, no. 01, pp. 31–40, 2023. (in Chinese)

[9] S. F. Ding, W. Du, J. Zhang *et al.*, "Research progress of multi-agent deep reinforcement learning," *Chinese Journal of Computers*, vol. 47, no. 07, pp. 1547–1567, 2024. (in Chinese)

[10] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5998–6008.

[11] W. H. Wang, Y. Q. Zhang, Y. L. Sui *et al.*, "Reinforcement-learning-guided source code summarization via hierarchical attention," *IEEE Trans. on Software Engineering*, vol. 48, no. 1, pp. 102–119, 2020.

[12] X. H. Nian, M. M. Li, H. B. Wang *et al.*, "Large-scale UAV swarm confrontation based on hierarchical attention actor-critic algorithm," *Applied Intelligence*, vol. 54, no. 4, pp. 3279–3294, 2024.

[13] X. Teng, X. Zhang, and Z. G. Luo, "Multi-scale local cues and hierarchical attention-based LSTM for stock price trend prediction," *Neurocomputing*, vol. 505, no. 9, pp. 92–100, 2022.