Multi-DAT: Dynamic Job Task Scheduling Method Based on Multi-agent Reinforcement Learning

Linwei Yao
Dongguan University of Technology
Dongguan, China
ylw105501@foxmail.com

Kuan Li*
Dongguan University of Technology
Dongguan, China
likuan@dgut.edu.cn

Huiying Xu
Zhejiang Normal University
Jinhua, China
xhy@zjnu.edu.cn

Xinzhong Zhu*
Zhejiang Normal University
Jinhua, China
Beijing Geekplus Technology Co., Ltd.
Beijing, China
zxz@zjnu.edu.cn

Hongbo Li
Beijing Geekplus Technology Co., Ltd.
Beijing, China
Jason.li@geekplus.com

***Abstract*—The scheduling of aircraft support tasks requires efficient planning based on the available resources in each support position. The many-to-many characteristics of tasks and the dynamic nature of scheduling environments place high demands on the real-time responsiveness of algorithms. Additionally, the complexity inherent in task scheduling and the need for flexible sequential processing further complicate decision-making. Existing methods often struggle to achieve both fast response times and effective task message capture. To address these challenges, we propose a novel scheduling method called Multi-DAT, which is designed to optimize dynamic task allocation using a multi-agent reinforcement learning algorithm. We improve on the traditional QMIX algorithm to select the shortest duration tasks based on priority, combined with a newly designed long-term reward function, which integrates long-term historical actions into the scheduling algorithm. Experimental results demonstrate that our proposed method outperforms the traditional rule-based method and seven other multi-agent reinforcement learning-based scheduling algorithms in terms of scheduling performance.**

***Index Terms*—Multi-agent reinforcement learning, Dynamic scheduling, Task allocation**

I. Introduction

As modern technology increasingly relies on aircraft operations, efficient scheduling of support tasks has become essential. These tasks involve complex operations such as inspections, refueling, power checks, hydraulic tests, and equipment loading. Given their complexity and the constant, constantly changing task requirements, scheduling becomes a challenging endeavor. The scheduling problem in this context is particularly complex due to the need to adhere to multiple constraints. These include sequential and parallel operation restrictions driven by safety considerations—for instance, equipment loading cannot occur simultaneously with refueling—and the rationality of resource allocation. Consequently, the scheduling process faces challenges related to multi-resource distribution, task sequence management, and other constraints that are both temporally and spatially limited. Addressing these challenges is crucial to ensure timely and efficient preparation of the aircraft for operations.

Efficiently completing tasks with limited resources and time has become a critical challenge in modern aircraft operations. Traditional scheduling methods relying on manual rules face significant limitations in complex, dynamic task environments. This issue is particularly acute in dynamic scheduling scenarios, where fluctuations in task priority, resource availability, and processing order make real-time responses extremely challenging. Reinforcement learning (RL) enables agents to learn optimal actions through trial-and-error feedback while interacting with their environment. This characteristic makes RL particularly suitable for addressing dynamic and uncertain problems, unlike traditional rule-based methods. Deep reinforcement learning (DRL) combines RL with deep neural networks, enhancing its ability to learn from complex, evolving environments. This capability enables effective task assignment and decision-making under various conditions. However, both RL and DRL algorithms typically employ single-agent architectures. These architectures integrate all possible actions for every task into one framework, creating a massive action space that requires highly complex deep neural networks [1].

Recently developed multi-agent reinforcement learning (MARL) enables agents to handle high-dimensional state spaces by transforming action space complexity into cooperative challenges among agents [2]. This paper proposes a MARL-based dynamic task scheduling method for aircraft support operations, with the following main contributions:

1) We model the scheduling process as a decentralized, partially observable Markov decision process (Dec-POMDP). By integrating invalid action masking [3] with QMIX [4], we develop task allocation strategies that adhere to the positional constraints of task execution.
2) To achieve dynamic task assignment, we design a strategy that prioritizes the shortest time-consuming tasks.
3) We present Multi-DAT, a novel approach enhancing

solution quality through QMIX-based long-term reward functions with historical action integration.

4) Our experimental results show the framework's superior performance and adaptability across diverse scenarios, outperforming both rule-based methods and seven alternative reinforcement learning algorithms.

II. Related Work

Limited research has addressed aircraft support task scheduling in resource allocation contexts. Yuan et al. [5] proposed a dynamic scheduling method using a rolling horizon strategy and a dual-population genetic algorithm for aircraft scheduling in dynamic environments. Cui et al. [6] introduced a multi-objective optimization framework with a super-heuristic algorithm to optimize flight deck operations and resource allocation, improving sortie rates and combat capabilities. Liu et al. [7] developed an enhanced variable neighborhood search algorithm to optimize fixed support resource configurations on carrier decks; however, its computational complexity increases exponentially when optimizing both mobile resource vehicles and fixed positions simultaneously.

In recent years, reinforcement learning has emerged as a promising approach for addressing aircraft support scheduling challenges. Li et al. [8] developed a real-time scheduling framework utilizing Deep Q-Network (DQN) to optimize aircraft support operations in highly dynamic combat environments. By formulating the problem as a partially observable Markov decision process (POMDP) and implementing an offline-to-online learning paradigm, they achieved significant improvements in scheduling efficiency. Wei et al. [9] introduced a Deep Reinforcement Learning-based Multi-Aircraft Cooperative Decision Framework (DRL-MACDF) for intelligent decision-making in multi-aircraft combat scenarios. Through the implementation of four novel algorithmic enhancement mechanisms, they demonstrated substantial improvements in inter-agent cooperation efficiency. Feng et al. [10] presented a deep reinforcement learning approach for aircraft carrier support operation scheduling, focusing on minimizing total operation duration under resource constraints. However, their methodology exhibits limitations in model generalization, fails to consider simultaneous resource and location constraints, and lacks dynamic task execution adjustment capabilities. Yao et al. [11] developed Multi-NPDQ, a multi-agent DRL framework for operation scheduling in high-dimensional state spaces. While their innovative Non-Stop strategy enhances performance in dynamic environments, the framework lacks explicit task execution ordering capabilities.

III. Problem Statement and Modeling

A. Problem Statement

In the dynamic scheduling of support tasks, we face the challenge of efficiently executing complex logistics tasks to ensure operational readiness. Each aircraft requires thorough inspection, maintenance, and supply loading, necessitating effective management and allocation of limited support resources. These aircraft must move between multiple positions across the carrier, each equipped with various resources to provide different services. Additionally, a group of aircraft must simultaneously perform various safety tasks, which calls for reasonable resource allocation and scheduling across multiple positions. Therefore, this paper's objective is to minimize the total completion time for a wave of aircraft scheduling tasks and to determine the optimal task sequence for each aircraft.

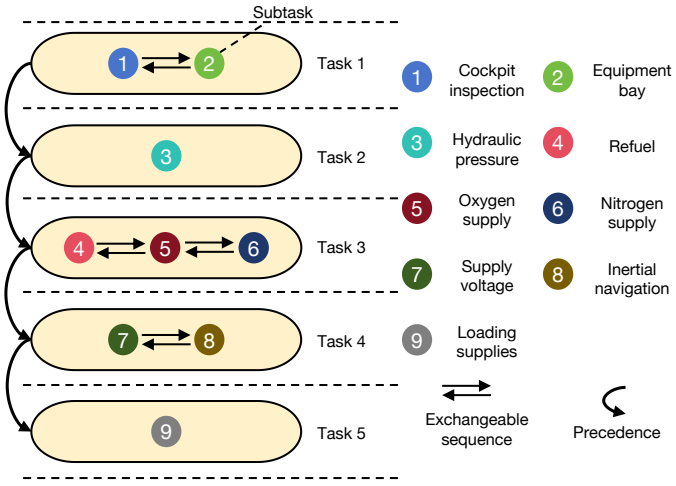

Fig. 1: Support Task Scheduling for Each Aircraft: The right section lists subtasks 1-9 with their corresponding resource services, while the left displays the task processing sequence.

In the scheduling process, as shown in Figure 1, the support task for each aircraft is considered a multi-stage task composed of several subtasks, such as cabin body inspection, hydraulic pressure checks, and power supply setups. These subtasks can be arranged in a flexible order within each stage. Still, they must adhere to priority constraints between stages—meaning that all subtasks in a previous stage must be completed before moving on to the next stage [12]. Each subtask requires different resources, and there is a deterministic matching relationship between these resources and the corresponding subtasks. The completion of all subtasks associated with a task signifies the completion of that task.

B. Scheduling model based on Dec-POMDP

The dynamic scheduling environment for aircraft support differs significantly from traditional job-shop scheduling in task representation and resource allocation. Each aircraft is modeled as an agent $a \in A$, where $A = \{a_1, a_2, ..., a_n\}$ represents all aircraft agents, each with k sequential task stages $T = \{T_1, T_2, ..., T_k\}$. For each stage $T_i \in T$, there are m tasks, each comprising mi subtasks $\{st_{i,1}, st_{i,2}, ..., st_{i,mi}\}$ requiring dedicated service resources. The environment includes s support positions $P = \{P_1, P_2, ..., P_s\}$, each providing specialized resources. The process terminates when all aircraft agents complete their tasks, modeled as a Multi-Agent System (MAS) where each agent selects actions based on real-time environmental feedback and its state.

We formulate the scheduling process as a decentralized partially observable Markov decision process (Dec-POMDP) using the tuple $\langle N, S, U, O, P, R, \gamma \rangle$, following Ref. [2]. Here, N denotes the set of agents, S describes the global state of the environment, U represents the joint action space, O defines the joint observation space, P captures the state transition probabilities, R indicates the shared reward among all agents, and γ is the discount factor. In our framework, the model components are specified as follows:

*a) **State Space**:* In a Dec-POMDP, the state space is defined as $S = \{S_1, S_2, ..., S_n\}$, where all agents share a global state that encapsulates completed tasks, historical position data, pending tasks for subsequent stages, and the associated position information (including position IDs and locations) for each task.

*b) **Observation Space**:* The observed value represents the environmental information perceived by each agent, defined as the joint observation $O_t = \{O_1, O_2, ..., O_n\}$, where O_i denotes the observation of the i-th agent at step t, including its current position, task progress, next stage tasks, compatible support position indices, and processing time (comprising both movement and operation durations).

*c) **Action Space**:* An action $u^a \in U^a$ is an action that the job agent is ready to take at the next step. The set of actions $U^a = \{1, 2, ..., s, s+1, s+2, s+3\}$, where s denote the number of support positions. There are four types of actions:

Action Type	Range of $u^a \in U^a$	Meanings
Moving action	$u^a \in \{1, 2, ..., s\}$	The agent selects the ID of the next position.
Waiting action	$u^a \in \{s+1\}$	There are no free position.
Busy action	$u^a \in \{s+2\}$	The agent is processing a task
Stop action	$u^a \in \{s+3\}$	The agent completes all tasks

*d) **Reward Function**:* In contrast to the reward function in Ref. [11], which relies on the number of agent moves to determine rewards and penalties, our approach comprehensively incorporates the objective of minimizing scheduling time. We design a reward function that not only considers the number of agent moves but also introduces a time factor to guide agents in learning strategies that effectively reduce scheduling time. Within the MARL framework, all agents share the same reward function, defined as follows:

$$R_t = \begin{cases} N_t^1 - \frac{K}{\sum_t T^t * N_t^2} & , N_t^2 \neq 0 \ and \ t \neq t_{end} \\ N_t^1 - \frac{K}{\sum_t T^t} & , N_t^2 = 0 \ and \ t \neq t_{end} \\ \frac{K}{T} & , t = t_{end} \end{cases} \quad (1)$$

Where K is parameters in the range of (0, $+\infty$). T^t represents the time on step t and $\sum_t T^t$ represents the cumulative completion time to step t. N_t^1, N_t^2 indicates the number of moving and waiting actions chosen by the agent in MAS.

*e) **Discount Factor**:* $\gamma \in [0, 1]$ controls the agent's focus on future rewards. A lower discount factor indicates that the agent prioritizes immediate returns over future expectations, while a higher discount factor signifies greater emphasis on long-term cumulative rewards. Our study prioritizes long-term rewards and sets the same parameter as in Ref. [2], $\gamma = 0.95$.

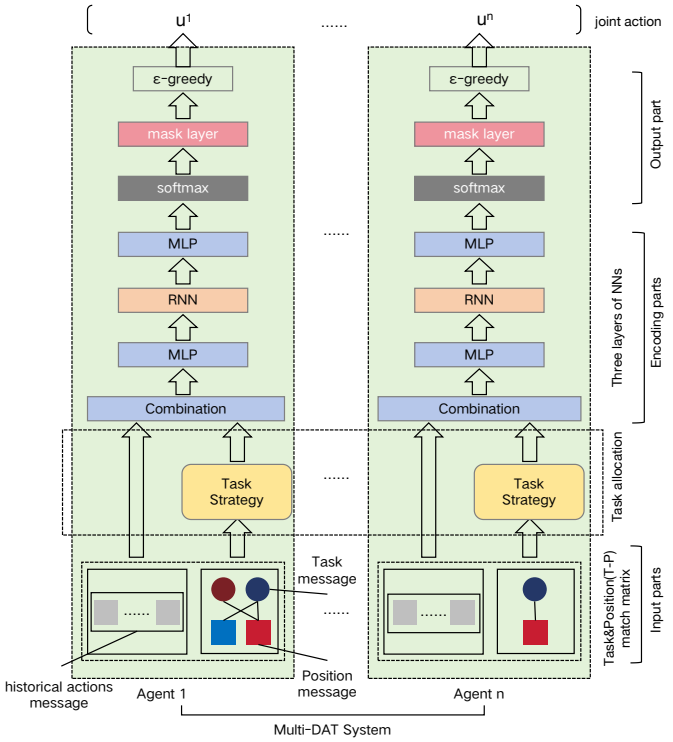

Fig. 2: The Multi-DAT method comprises three core components: input, encoding, and output. Agents receive task messages(type, priority, resources), position messages(current and support locations), and historical action messages in the input section, where a task strategy optimizes task-position matching. The encoding section employs RNNs for temporal dependencies and MLPs for feature extraction. In the output section, Softmax selects actions, a mask layer ensures validity, and ϵ-greedy balances exploration and exploitation.

We integrate guidance signals into the reward function to steer the algorithm toward strategies that reduce completion time. Training reveals a positive correlation between the number of moving actions and reward, indicating more reasonable strategies. Conversely, we minimize waiting actions to avoid service assignment conflicts. Additionally, we ensure the current completion time T is minimized at each step, as the total completion time critically affects the final reward.

IV. Scheduling Frame

Unlike other scheduling methods that depend on a single-agent RL model, this paper models each aircraft as an independent agent, representing a group of such aircraft as a MAS. As illustrated in Figure 2, each agent i receives input details corresponding to the current subtask message and compatible position. It determines task information through a task-position strategy, concatenates this information with the historical action set, and ultimately outputs the selected operation u_i to indicate the next position to be processed.

The Multi-DAT approach dynamically adjusts the task order, enhancing the prioritization of tasks. The pseudocode for the training algorithms is detailed in Algorithm 1.

Algorithm 1 Algorithm of Multi-DAT

Input: learning rate α; mini-batch size b.

Initialize RNN-eval Q, RNN-target Q^* for each agent and parameters θ, θ^*;
Initialize Mixing-net-eval Q_{tot}, Mixing-net-target Q^*_{tot} and parameters ϕ, ϕ^*;
Initialize replay memory D with capacity N;
Initialize scheduling environment;
for each episode **do**
  Reset environment;
  Initialize each agent's historical actions H_a with empty;
  for each time step **do**
    for each agent **do**
      // Get Task ID
      Into Task-Allocation Algorithm
      Execute Action Mask;
      With probability ϵ select a random action a_t;
      Otherwise select $a_t = \text{argmax}_a Q(g_t, T, H_a, a, \theta)$;
      Append a_t to the historical action H_a;
    end for
    Joint actions U_t;
    Execute U_t, observation O_t reward R_t and next state g_{t+1}, $s_t = g_{t+1}$;
    Store $(g_t, T, H_a, U_t, R_t, g_{t+1})$ in D;
    Update H_a;
    Update parameters θ, θ^*, ϕ, ϕ^* sampling from D;
  end for
  Continue to iterate until the convergence condition is satisfied;
end for

Output: Reward R.

A. Task-Allocation Strategy

To implement MARL training in dynamic task scheduling for each agent, we use a greedy algorithm to compute the Euclidean distance between aircraft positions and compatible subtask positions. The algorithm estimates the processing time for each task-position pair, selecting the pair with the shortest time for the agent's task sequence. The next target position is also determined, ensuring efficient task assignment. This minimizes latency and adapts to dynamic changes like new tasks or priority shifts, as detailed in Algorithm 2.

B. Long-Term Historical Actions

In MARL for scheduling, agents typically make independent decisions based on immediate environmental information. We propose a novel approach enabling collaborative decision-making and addressing long-term task dependencies. Our method requires agents to integrate both their long-term historical actions and current observations when making decisions. This combination allows agents to view their actions within the broader context of the entire task sequence, improving coordination among agents and optimizing scheduling outcomes over time. By considering the larger context, our approach better adapts to shifting priorities and evolving conditions.

Algorithm 2 The Task-Allocation Algorithm

Input: Subtask set S, location set P, current location PC, speed S, compatibility function $C()$.

$selected\_task\_ID \leftarrow$ None;
$min\_time \leftarrow \infty$;
// Greedy algorithm
for each subtask s_i in S **do**
  for each position p_j in P **do**
    // Check if s_i and p_j are compatible
    if $C(s_i, p_j) == 1$ **then**
      $d(p_j) \leftarrow$ Euclidean distance(PC, p_j);
      $t(s_i) \leftarrow$ processing time of s_i;
      $total\_time \leftarrow t(s_i) + \frac{d(p_j)}{S}$;
      if $total\_time < min\_time$ **then**
        $min\_time \leftarrow total\_time$;
        $selected\_task\_ID \leftarrow s_i$;
      end if
    end if
  end for
end for
return $selected\_task\_ID$

C. Long-Term Reward Function

Based on Equation 1, we enhanced the reward function to address MARL-based task scheduling in dynamic environments. While reducing agent moves is important, it alone cannot ensure timely task completion. To improve this, we introduced a factor to incentivize time efficiency and minimize unnecessary actions. Experiments show that this enhancement overcomes the limitations of Ref. [11], which focused solely on agent movements and ignored time efficiency. By integrating both movement and time considerations, agents develop more effective strategies for dynamic environments with tight deadlines, improving task completion times and balancing resource utilization with time management.

Additionally, we fine-tuned the parameter K through experiments, enhancing flexibility and adaptability. This adjustment helps agents minimize waiting times and time consumption while maintaining task quality, resulting in more efficient and intelligent scheduling strategies.

V. Experiment

A. preparation

a) Software and Hardware: A server computer with the configuration of Intel(R) Xeon(R) Gold 6148 CPU @ 2.40G Hz, 283 GB RAM, and NVIDIA RTX A6000 GPU is applied to train models. The deep learning platform is PyTorch 1.8.1 with CUDA 11.1 and Python3.7.16.

b) Data setting: We abstract the tasks as Task1-1, Task1-2, Task2, Task3-1, Task3-2, Task3-3, Task4-1, Task4-2, Task5, respectively. The consumption time of the corresponding task is consistent with that of Ref. [11], As shown in Table I. To ascertain the task dynamics of our proposed model, we have meticulously crafted three distinct scenarios, each featuring unique task constraints, as illustrated in Table II.

TABLE I: Task Processing Duration (Hours) for Each Operation.

Task Name	Task1-1	Task1-2	Task2	Task3-1	Task3-2	Task3-3	Task4-1	Task4-2	Task5
Duration Time	2	6	15	4	15	2	10	10	10

TABLE II: Different Task Constraint Scenarios and Configurations.

Scenarios ID	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Scene1	Task1-1, Task1-2	Task2	Task3-1, Task3-2, Task3-3	Task4-1, Task4-2	Task5
Scene2	Task1-1	Task2	Task3-1, Task3-2	Task4-1	Task5
Scene3	Task1-1, Task1-2	/	Task3-1, Task3-3	Task4-2	/

Seven MARL-based scheduling algorithms were executed for comparison: QMIX, VDN [13], COMA [14], QTRAN_base [15], QTRAN_alt [15], REINFORCE [16], and Central_V [17]. All algorithms were assessed using the same parameter settings. We ensured that the definitions of observation, reward, global state, and action were harmonized and aligned across these seven methods.

B. Generalization of different scene models

We utilize the task duration data presented in Table I and the environmental scenes in Table 2 to perform a series of comparative experiments on various reinforcement learning algorithms. These experiments evaluate the total completion time achieved by single-agent, distributed-agent, and multi-agent algorithms across diverse scenario models. To ensure consistency, each algorithm is subjected to multiple experimental trials, and the shortest total completion time from these trials is selected for comparative analysis. The results of these experiments are graphically presented in Figure 3.

The experiments show that the QMIX algorithm is better at reducing total time in different scene models than other algorithms. Consequently, in the subsequent sections of our analysis, the QMIX algorithm is employed as the default for completing the training, leveraging its proven efficiency in optimizing task completion times.

C. Parameter determination of reward function

We set the tasks in Table I to forced serial priority processing, constituting the environment scenario of Ref. [11]. We conducted several experiments for the proposed reward function and trained the same number of iterations according to different parameter settings $K = 1$, $K = 50$, $K = 100$, $K = 150$. The experimental results in Figure 4 show that with the parameter $K = 1$, we can obtain an optimal solution.

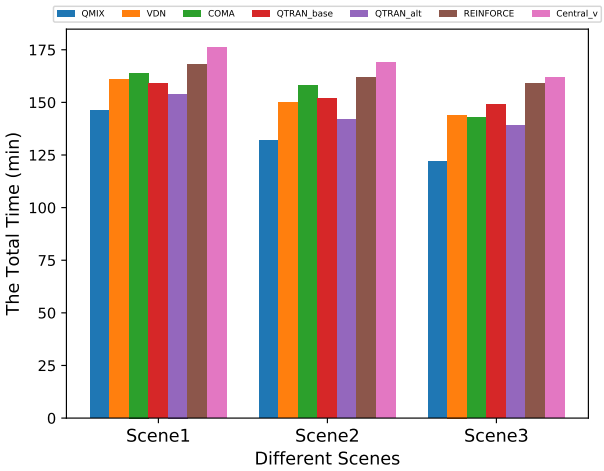

Fig. 3: Total Execution Time for Seven MARL Algorithms Across Different Scenarios.

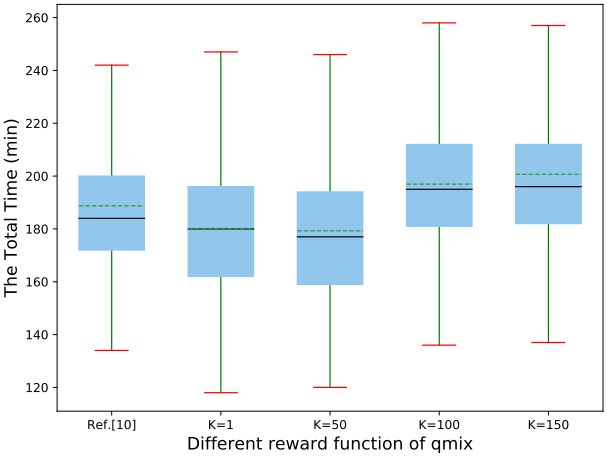

Fig. 4: Total Execution Time for Various Reward Functions

D. Ablation experiment

At the start of training, QMIX exhibited significant fluctuations in scheduling times with limited long-term improvement. However, with the new reward function or long-term historical actions, it quickly learns time-efficient scheduling solutions.

Figure 5 compares the time consumption of QMIX, Multi-DAT (without long-term historical actions), and Multi-DAT (without the new reward function) under identical scenarios. The results show that combining the new reward function with the long-term historical actions strategy outperforms using either strategy alone. This combination enhances scheduling efficiency and demonstrates the potential for improved task management in complex, dynamic environments.

E. Result analysis

Our learned scheduling strategy is compared with three practical priority dispatching rules (PDRs): FIFO, most operations remaining (MOR), and most work remaining (MWKR) [18]. The baseline PDRs are implemented in the

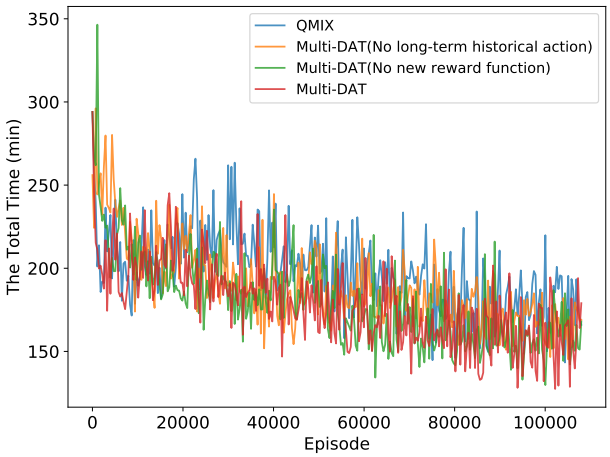

Fig. 5: Ablation experiments tested the new reward function and the long-term historical actions separately, assessing their impact on algorithm effectiveness in consuming time.

same environment as our approach for fair comparison, demonstrating its superiority over manual strategies. Additionally, we compare our method with Ref. [11], proving its enhanced effectiveness. Experimental results are presented in the following table, with each data group recorded twice to minimize random errors.

TABLE III: Comparison of Total Time: PDRs vs. MARL

Scene Models	PDRs			MARL	
	FIFO	MOR	MWKR	Ref. [11]	Multi-DAT
Ref. [11]	188	195	206	146	**120**
	182	195	201	148	**123**
Scene1	175	185	192	138	**112**
	176	180	192	132	**117**
Scene2	168	178	186	122	**107**
	164	176	183	122	**102**

Table III shows that our proposed approach, enhanced by a customized reward function and long-term action training, significantly outperforms both PDRs and the RL methods described in Ref. [11] across diverse scheduling scenarios.

VI. Conclution

Building on Ref. [11], our proposed Multi-DAT framework introduces a task-allocation strategy tailored to real-world constraints, along with a new reward function and the integration of long-term historical actions in training. Experimental results demonstrate that Multi-DAT outperforms PDRs and RL methods in task scheduling, demonstrating superior adaptability and efficiency by leveraging historical context. Nevertheless, challenges persist in managing tasks with large or continuous state spaces, which will be the focus of our future research.

Acknowledgment

This work was supported by the Dongguan Science and Technology of Social Development Program under Grants 20221800905182, 20231800940522 and 20231800936132; the National Natural Science Foundation of China (62376252); the Key Project of Natural Science Foundation of Zhejiang Province (LZ22F030003); and the Zhejiang Province Leading Geese Plan (2025C02025, 2025C01056).

References

[1] S. Baer, J. Bakakeu, R. Meyes, and T. Meisen, "Multi-agent reinforcement learning for job shop scheduling in flexible manufacturing systems," in *2019 Second International Conference on Artificial Intelligence for Industries (AI4I)*. IEEE, 2019, pp. 22–25.

[2] X. Wang, L. Zhang, T. Lin, C. Zhao, K. Wang, and Z. Chen, "Solving job scheduling problems in a resource preemption environment with multi-agent reinforcement learning," *Robotics and Computer Integrated Manufacturing: An International Journal of Manufacturing and Product and Process Development*, no. 77-, p. 77, 2022.

[3] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *arXiv preprint arXiv:2006.14171*, 2020.

[4] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," 2018.

[5] P. Yuan, W. Han, X. Su, J. Liu, and J. Song, "A dynamic scheduling method for carrier aircraft support operation under uncertain conditions based on rolling horizon strategy," *Applied Sciences*, vol. 8, no. 9, p. 1546, 2018.

[6] R. Cui, W. Han, X. Su, Y. Zhang, and F. Guo, "A multi-objective hyper heuristic framework for integrated optimization of carrier-based aircraft flight deck operations scheduling and resource configuration," *Aerospace Science and Technology*, vol. 107, p. 106346, 2020.

[7] L. Yujie, H. Wei, S. Xichao, and C. Rongwei, "Optimization of fixed aviation support resource station configuration for aircraft carrier based on aircraft dispatch mission scheduling," *Chinese Journal of Aeronautics*, vol. 36, no. 2, pp. 127–138, 2023.

[8] W. Han, F. Guo, and X. Su, "A reinforcement learning method for a hybrid flow-shop scheduling problem," *Algorithms*, vol. 12, no. 11, p. 222, 2019.

[9] S. Wei, F. Yang-He, C. Guang-Quan, H. Hong-Lan, H. Jin-Cai, L. Zhong, and H. Wei, "Research on multi-aircraft cooperative air combat method based on deep reinforcement learning," *Acta Automatica Sinica*, vol. 47, no. 7, pp. 1610–1623, 2021.

[10] H. Feng and W. Zeng, "Deep reinforcement learning for carrier-borne aircraft support operation scheduling," in *2021 International Conference on Intelligent Computing, Automation and Applications (ICAA)*. IEEE, 2021, pp. 929–935.

[11] L. Yao, Q. Chen, L. Gong, and K. Li, "Multi-npdq: A multi-agent approach through deep reinforcement learning for operation scheduling," in *Advanced Intelligent Computing Technology and Applications - 20th International Conference, ICIC 2024, Tianjin, China, August 5-8, 2024, Proceedings, Part II*, vol. 14863. Springer, 2024, pp. 467–479.

[12] X. Wang, L. Zhang, Y. Liu, F. Li, Z. Chen, C. Zhao, and T. Bai, "Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning," *Journal of Manufacturing Systems*, 2022.

[13] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, and K. Tuyls, "Value-decomposition networks for cooperative multi-agent learning," 2017.

[14] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," 2017.

[15] D. Kim, S. Moon, D. Hostallero, W. J. Kang, and Y. Yi, "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," *International Conference on Representation Learning*, 2019.

[16] J. Zhang, J. Kim, B. O'Donoghue, and S. P. Boyd, "Sample efficient reinforcement learning with reinforce," 2021.

[17] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multiagent policy gradients," in *National Conference on Artificial Intelligence*, 2018.

[18] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Annals of Operations Research*, vol. 41, no. 3, pp. 157–183, 1993.