electronics

MDPI

Article

A Method for the Predictive Maintenance Resource Scheduling of Aircraft Based on Heterogeneous Hypergraphs

Long Kang [1], Muhua He [2], Jiahui Zhou [2], Yiran Hou [2], Bo Xu [3] and Haifeng Liu [4,5,*]

[1] Information Engineering Department, AECC Harbin Dongan Engine Co., Ltd., Harbin 150066, China; kanglong1120@163.com
[2] School of Software, Dalian University of Technology, Dalian 116023, China; 2440776168@mail.dlut.edu.cn (M.H.); zjhjixiang@mail.dlut.edu.cn (J.Z.); 215363@stu.hebut.edu.cn (Y.H.)
[3] School of Computer Science and Technology, Dalian University of Technology, Dalian 116023, China; xubo@dlut.edu.cn
[4] School of Computer and Electronic Information/School of Artificial Intelligence, Nanjing Normal University, Nanjing 210046, China
[5] Adolescent Education and Intelligence Support Lab, Nanjing Normal University, Nanjing 210046, China
* Correspondence: liuhaifeng@nnu.edu.cn

Abstract: The resource scheduling optimization problem in predictive maintenance is a complex operational research challenge involving reasoning about stochastic failure models and the dynamic allocation of repair resources. In recent years, resource scheduling methods based on deep learning have been increasingly applied in this field, demonstrating promising performances. Among these, resource scheduling algorithms based on heterogeneous graphs have shown exceptional results in multi-objective optimization tasks. However, conventional graph neural networks primarily operate on binary relational graphs, which struggle to effectively utilize data in multi-relational settings, thereby limiting the scheduler's performance. To address this limitation, this paper proposes a heterogeneous hypergraph-based resource scheduling algorithm for aircraft maintenance tasks to tackle the challenges of higher-order and many-to-many relationship processing inherent in traditional graph neural networks. Specifically, the proposed algorithm defines aircraft nodes and maintenance personnel nodes while introducing decision nodes and state nodes to construct hyperedges. It employs hypergraph convolution with a multi-head attention mechanism to learn the long-term value of decisions, followed by policy selection based on a Markov decision process. This method offers a lightweight, non-parametric dynamic scheduling solution capable of robust learning in highly stochastic environments. Comparative experiments conducted on three datasets of varying scales demonstrate that the proposed method outperforms both heuristic algorithms and existing deep learning methods in terms of its optimization performance on M1 and M2 metrics. Furthermore, it surpasses resource scheduling algorithms based on heterogeneous graph neural networks across multiple metrics.

Keywords: hypergraph; GAT; resource optimization; reinforcement learning; predictive maintenance scheduling

check for updates

Academic Editor: Javid Taheri

Received: 6 January 2025
Revised: 7 February 2025
Accepted: 12 February 2025
Published: 17 February 2025

Citation: Kang, L.; He, M.; Zhou, J.; Hou, Y.; Xu, B.; Liu, H. A Method for the Predictive Maintenance Resource Scheduling of Aircraft Based on Heterogeneous Hypergraphs. *Electronics* **2025**, *14*, 782. https://doi.org/10.3390/electronics14040782

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Resource optimization plays an important role in domains such as healthcare, manufacturing, and services [1]. In these real-world scenarios, task allocation and resource scheduling face multiple challenges related to uncertainty and stochasticity, including the effects of equipment failures, priority changes, and uncertain processing times [2]. Aircraft maintenance

scheduling is one such domain. In this scenario, the scheduling process needs to consider the random failure time and maintenance time of aircraft components, which are random variables with inherent uncertainties. The stochastic nature and multiple constraints of maintenance scheduling make it complex and unpredictable. Real-time responses to failures are crucial for prioritizing maintenance on aircraft subsystems. Effective predictive maintenance reduces unplanned work and significantly lowers operational and maintenance (O&M) costs [3]. For instance, it can cut unplanned work by about 33%, saving the aviation industry approximately USD 21.7 billion annually. Thus, predictive maintenance is an increasingly effective way to ensure aircraft safety and reduce O&M costs.

The resource optimization problem in predictive maintenance involves reasoning with complex stochastic failure models and dynamically allocating repair resources, making it a typical computational challenge in operations research [4]. Currently, traditional fault diagnosis and scheduling methods primarily rely on heuristic algorithms, supplemented by manual adjustments from domain experts [5–7]. Although this approach can partially meet scheduling requirements, it relies on manual intervention. As a result, the method is difficult to scale to large-scale scenarios and shows limited performance when dealing with complex systems and uncertainty.

In recent years, deep learning has rapidly developed in academia and industry. Its powerful capabilities in terms of feature extraction and pattern recognition have led to its remarkable performance across various fields. Deep learning, in particular, has shown an exceptional ability to handle complex, high-dimensional data, which has attracted a large number of scholars to its study and exploration. Currently, experts from various fields are attempting to use deep learning to solve diverse scheduling and optimization problems [8,9]. Among the deep learning frameworks available, graph neural networks (GNNs) are a class of models suitable for handling structured data. They can directly manipulate graph structures and capture complex relationships between nodes [3]. Graph neural networks (GNNs) have proven to be highly effective in modeling graph-structured data [9]. The emergence of GNNs offers a novel approach to tackling complex tasks in resource scheduling. In particular, when applied to heterogeneous graph structures, GNNs can effectively model the relationships between various resources and tasks, offering significant advantages for multi-objective optimization.

The current best-performing method is HetGPO, which employs a resource scheduling algorithm based on heterogeneous graph neural networks. Additionally, it leverages attention mechanisms to enhance model performance, surpassing previous methods based on machine learning, deep learning, and heuristic algorithms. However, traditional graph neural network models are typically based on binary relation graphs, which are primarily used to represent simple point-to-point relationships, such as user associations in social networks and atomic bonds in molecular graphs [10]. Binary relation graphs are insufficient for modeling complex many-to-many relationships. For example, in the aircraft scheduling problem, there are complex higher-order correlations between different aircraft and multiple maintenance personnel, and it is difficult to fully capture their interactions using only binary relations. This issue is particularly evident in predictive maintenance scenarios, where scheduling decisions involve multi-dimensional and multi-level relationships. Traditional graph neural networks have a limited ability to deal with higher-order relationships. As a result, these models cannot fully capture deeper information from the data, which in turn reduces their scheduling performance [11–13].

To address this issue, we introduce a resource scheduling algorithm using a heterogeneous hypergraph. This approach captures complex many-to-many relationships between aircraft and maintenance resources more effectively than traditional graphs. By incorporating decision and state nodes, our method leverages Markov decision processes to enhance

model flexibility and scalability. This architecture fully utilizes multivariate information, leading to more efficient predictive maintenance scheduling.

2. Related Work

We present previous work that is related to this paper in terms of two main areas: classical resource scheduling algorithms and the core technologies of our model.

2.1. Resource Scheduling Algorithms

Classical resource scheduling algorithms can be mainly divided into two categories: heuristic algorithms and deep learning-based algorithms. We introduce these representative algorithms below.

2.1.1. Heuristic Algorithms

Heuristic algorithms are widely used to solve resource scheduling problems. The Random Scheduler determines the execution order of tasks by random selection. While this approach is simple and easy to implement, it fails to consider task demands and resource status, making it poorly suited for complex dynamic environments. Although intelligent strategies, such as the Corrective Scheduler [14] and Condition-based Scheduler [5], can dynamically adjust scheduling plans and consider system states, they rely heavily on manually set rules. These strategies are limited to low-dimensional, linear application scenarios and cannot handle complex interdependencies between tasks and resources. Similarly, the Periodic Scheduler [6] is designed for fixed periodic scheduling tasks and lacks the ability to adapt to changing conditions. Therefore, despite their applicability and ease of implementation, heuristic algorithms often struggle to address high-dimensional, heterogeneous interdependencies among tasks when faced with complex resource scheduling optimization problems [15,16].

2.1.2. Deep Learning-Based Algorithms

Among the deep learning-based scheduling algorithms, deep reinforcement learning models, such as DeepRM [7] and Decima [8], have been applied to resource scheduling optimization. DeepRM intelligently allocates resource demands through deep reinforcement learning. Decima, on the other hand, optimizes workload-specific scheduling strategies through reinforcement learning. It embeds state and decision nodes into the graph structure, but its simple graph representation struggles to capture the complex correlations and heterogeneity among tasks and large amounts of training data are required to ensure the model's ability to generalize. In contrast, resource scheduling algorithms based on heterogeneous graph neural networks (GNNs) have made breakthroughs in modeling task–resource relationships. One such model is HetGPO, proposed by Wang et al. [3], which achieves resource optimization through heterogeneous graph-based modeling. However, the graph structure of HetGPO can only handle limited associations between nodes and struggles to model complex high-order task relationships, which limits its effectiveness in more complex scenarios.

2.2. Core Technologies of Our Model

The core technologies of our model include hypergraph neural networks and Markov decision process (MDP) modeling. The following subsections will introduce the progress that has been made in these two core technologies.

2.2.1. Hypergraph Neural Networks

In the real world, many complex network relationships cannot be fully represented by ordinary graph edges, such as the various types of connections in social networks. The

HGNN model proposed by Feng et al. [17–19] and the HyperGCN model proposed by Yadati et al. [19] are among the first convolutional methods for hypergraphs. The core advantage of the HGNN lies in its ability to express and process high-order data correlations, which is particularly important in multimodal or multi-type data correlation problems. Compared to traditional graph neural networks (GNNs), HGNN naturally models higher-order correlations through the flexible hyperedge structure of hypergraphs, making it more suitable for applications in areas such as social networks, biological networks, and transportation networks. Furthermore, the HGNN framework is highly versatile and can learn implicit representations of higher-order data structures, with Graph Convolutional Networks (GCNs) being considered a special case of HGNN, where the edges in simple graphs can be viewed as second-order hyperedges connecting two vertices. In practical applications, HGNN has been shown to outperform traditional Graph Convolutional Networks (GCNs) in citation-based network classification and visual object recognition tasks. The aforementioned methods actually fail to focus on the heterogeneity of aircraft data, overlooking the differences among data points. This work constructs a hypergraph to model multivariate relationships, revealing higher-order structures and patterns in the data that may be ignored in traditional graphs. By analyzing the hyperedge connections in the hypergraph, more complex associations between nodes can be discovered.

2.2.2. Markov Decision Process Modeling

Markov decision process (MDP) models involving decision-making algorithms have been shown to be useful in various sequential planning applications [20], where accounting for the uncertainty in the process is crucial. The success of MDP applications can be attributed to efficient algorithms that find optimal solutions to the MDP model.

Partially Observable Markov Decision Processes (POMDPs) are based on Markov chains and introduce a process for decision-making and goal optimization. Partially Observable Markov Decision Processes (POMDPs) model sequential decision problems under conditions of uncertainty and partial observability [20]. In a POMDP, an agent makes decisions over a series of discrete time steps, with the goal of maximizing some cumulative reward [21]. Unlike a pure Markov chain, a Markov decision chain emphasizes the effect of the actions in a given state that change the probability of state transitions, thereby affecting the long-term payoff. A complete POMDP consists off a reward process and a decision process and aims at balancing long-term and short-term values to select an optimal strategy. This allows certain decisions to be made in a quantifiable manner, which can be processed by computers [3].

The computational complexity of solving the optimal POMDP increases exponentially with respect to the number of channels. If the number of channels available to secondary users is limited, the optimal policy can be computed offline, and the implementation cost remains reasonable [22]. Regardless of the strategy algorithm used, the objective is to identify the actions that maximize the reward and increase their selection probabilities as much as possible, so that they are highly likely to be chosen in the strategy scheduling problem. The maximum value objective function $J(\theta)$ is defined as follows:

$$\max_{\theta} J(\theta) = \max_{\theta} \mathbb{E}_{\tau \sim \pi_t} R(\tau) = \max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau) \tag{1}$$

In MDPs, the reward function is crucial as it directly influences the agent's objective of maximizing cumulative rewards. It shapes the learning process and determines the optimal policy. Sparse rewards can slow down learning, making exploration challenging. Reward shaping techniques, which add auxiliary rewards based on potential functions, can accelerate learning while maintaining policy optimality. However, over-reliance on these auxiliary rewards may lead to policy deviations. The reward function's design also impacts

sample complexity, with optimized rewards reducing the number of samples needed for effective learning.

Here, the state–action trajectory $\tau = (s_1, a_1, \ldots, s_t, a_t)$ generated by the interaction between the agent and its environment represents all the possible situations that occur during the interaction, as dictated by the strategy. The probability of trajectory τ occurring under $\pi_t(a \mid s)$ is given by the following expression:

$$P(\tau|\theta) = \left[\prod_{t=0}^{T} P(s_{t+1}|s_t, a_t) \cdot \pi_\theta(a_t|s_t)\right] \quad (2)$$

where $P(\tau|\theta)$ represents the probability of transitioning from state s_0 at time $t = 0$ to state s_t by taking actions $a_0 \ldots a_t$ under the parameters θ.

The MDP offers significant advantages over random and corrective schedulers by enabling optimized decision-making that maximizes long-term rewards while handling uncertainties and adapting dynamically to changing conditions

In existing works, several notable research gaps have emerged. First, many studies have overlooked the complexity of high-order relationships in data, especially in scenarios where multiple entities interact in intricate ways. For example, in resource allocation problems, the relationships between different resources and tasks have often been simplified, leading to sub-optimal solutions. Second, the handling of heterogeneous data remains a challenge. Existing methods struggle to effectively integrate data from diverse sources with varying characteristics, which limits their performance in real-world applications.

Our proposed approach aims to fill these gaps. We utilize hypergraph neural networks to capture the complex high-order relationships present in the data. Hypergraphs are well suited to representing these intricate connections, allowing our model to make more accurate predictions and decisions. Regarding heterogeneous data, we have developed a novel data integration mechanism that can effectively combine different data types, enabling our approach to handle real-world data more comprehensively. By doing so, we believe our proposed approach can overcome the limitations of existing works and provide more effective solutions.

3. Heterogeneous Hypergraph-Based Aircraft Maintenance Scheduling Algorithm

3.1. Problem Description

Within one time frame, there may be several damaged aircraft and a number of maintenance personnel available. Given that different aircraft require varying repair times and generate different hourly revenues post-repair, and considering the scarcity of maintenance personnel, different repair plans can lead to different total profits. Generally, profit is considered as the revenue generated by the aircraft minus the cost of repairing the aircraft and the cost incurred by paying the maintenance personnel for the repair work.

Each aircraft *pi* comprises *Ki* components/parts, contingent on its type (e.g., airliner or helicopter). Per the MSG-3 document, failure is characterized by a component's inability to operate within predefined limits. The failure probability for each component is modeled via the Weibull distribution, correlating with aircraft usage metrics like flight hours or landing/takeoff counts. When failure occurs for at least one of its components, the plane is grounded and marked as broken. Consequently, the plane failure model becomes a hybrid probabilistic model that jointly considers different plane parameters. The hyper-parameters of the failure model are not accessible for scheduling policies and must be inferred. This inference will serve as the basis for selecting the damaged aircraft nodes to construct the hypergraph in the present work.

Based on a virtual aircraft maintenance scheduling environment, we define two scheduling categories—aircraft and maintenance personnel—and a number of virtual states, including time, flight schedules, and landings, which simulate real-world scenarios. We analyze the gains after thirty days of operation in the virtual aircraft maintenance environment, under different optimization objectives, and explore the decision-making process used when an aircraft is damaged, with the goal of maximizing the gains.

Figure 1 illustrates the aircraft maintenance scheduling environment and the scheduling policy network. In this paper, the aircraft maintenance scheduling environment is created based on Wang et al.'s predictive maintenance environment [3], which includes the type of aircraft, its operational status, its components, and the status of the maintenance personnel. This environment simulates the real-life state and use of aircraft and maintenance personnel by modeling their states and operations through functions. At each time step, the scheduling strategy network initializes the node features, places the obtained observations into the feature dictionary of the nodes, monitors the state nodes, and trains them end-to-end by constructing a heterogeneous hypergraph to form the decision d_i with the largest reward value. Simultaneously, the requested decision is fed back into the virtual maintenance environment, and the resulting rewards, observations, overheads, and the KL divergence loss function are recorded during that training batch. The learner updates the gradients of the hypergraph learning method and the policy scheduling method upon receiving environmental feedback rewards. Meanwhile, weight files are saved to generate the next time step's heterogeneous hypergraph. The system makes scheduling decisions iteratively based on time steps. Therefore, the hypergraph network adjusts dynamically while making decisions. When the next time step arrives, data from the aircraft maintenance environment, including information about the aircraft and maintenance personnel, are used as features for the aircraft and maintenance personnel nodes. A neural network is then generated, and a classifier is used to obtain the decision d_{i+1}. After several time steps, a series of decisions is obtained, along with the total reward for each time step.

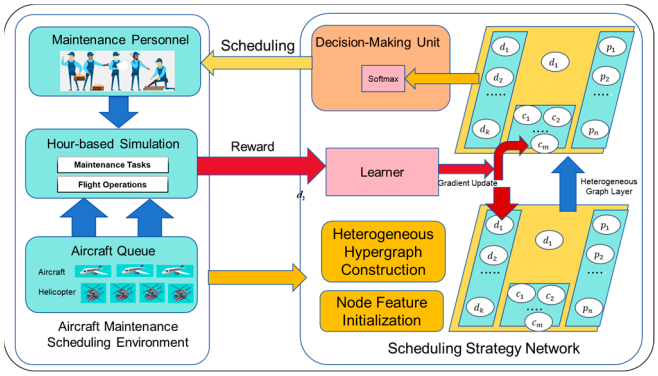


Figure 1. Scheduling process diagram.

Specifically, in this paper, the aircraft cluster N_p is denoted as $\{p_i\}$ and the maintenance personnel cluster N_c is denoted as $\{c_j\}$. Aircraft are categorized into passenger planes and helicopters. Each aircraft $\{p_i\}$ contains a set of observable parameters $\{o_k^i\}$ including its operation time, total number of takeoffs, and engine status. Each aircraft p_i is associated with a repeatable maintenance task m_i, a probabilistic failure model $P_i(break|usage)$, and a repeatable aircraft operating state f_i. Each maintenance personnel member c_j can be assigned a maintenance task m_i in the simulated aircraft maintenance scheduling environment and cannot take on any other task until their current maintenance task is completed. Each scheduling decision d_i in this paper is represented as a binary tuple $< p_i, c_j >$. d denotes that maintenance personnel c_j is assigned to perform a specific maintenance operation

(whether prioritized or not) on the faulty aircraft p_i starting at the current time step. The main notations used in this paper and their corresponding definitions are illustrated as follows (Table 1).

Table 1. Notations and descriptions.

Notations	Descriptions
$W_{edgename} \in R^{D \times S}$	Hyperedge trainable weight matrix
$\overrightarrow{h}_p, \overrightarrow{h}_s, \overrightarrow{h}_c, \overrightarrow{h}_d$	Feature vectors of the different nodes
$\alpha_{s,i}^{edgename}$	Attention score of the hyperedge
$R(s,a)$	Immediate reward for taking action a in state s
G_t	Total discounted reward
$V_\pi(s)$	Long-term value of state s
$Q_\pi(s,a)$	Action value function
γ	Discount factor

3.2. Selection Based on Markov Decision Process

3.2.1. The Definition of POMDP

In this paper, fault prediction and scheduling in the aircraft maintenance environment are formulated as a Partially Observable Markov Decision Process (POMDP), represented by the seven-tuple $< S, A, T, R, \Omega, O, \gamma >$:

1. S: States. The state S consists of the joint states of all aircraft and maintenance personnel, as well as the system hyperparameters that are used to model the aircraft's operational tasks and maintenance assignments.
2. A: Actions. At any given state, the agent can choose from a set of actions, each of which affects the probability of transition to the next state. In this paper, at time step t, the action set is defined as the set of maintenance decisions $U_t = \{d_1, d_2, \cdots, d_n\}$.
3. T: Transition. The transition function T defines the state transition probabilities, which represent the probability of an agent moving from their current state to the next state after taking an action. T corresponds to executing a particular action within the aircraft maintenance scheduling environment and transitioning to the next time step.
4. R: Rewards. The reward function R measures whether a decision d is optimal for achieving the desired outcome. It is an important parameter that needs to be learned within the aircraft maintenance scheduling environment, and it is set to match the scheduling objective that the user aims to maximize.
5. Ω: Observation. The observation Ω includes the state space $\{o_k^i\}$ for all aircraft, which encompasses parameters such as operational time, total takeoff count, and engine status. In addition, it also includes the observable states of all maintenance personnel, the current operational status of the aircraft, and the progress of maintenance tasks.
6. O: Observation Functions. The observation function O is generated by the aircraft maintenance scheduling environment, which updates the observation results after an action is taken.
7. γ: Discount Factor. The discount factor γ is primarily used to balance the relationship between immediate rewards and future rewards, helping to decide which type of reward the agent should prioritize when making decisions [23,24].

This paper optimizes the following three objectives:

Objective 1: Optimize for overall profit, which includes hourly revenue and maintenance costs, by calculating the total reward after training, factoring in penalty terms from the probabilistic fault model.

Objective 2: Consider revenue alone, i.e., calculate the total reward without accounting for costs.

Objective 3: Consider the availability of the aircraft team, i.e., calculate the average number of aircraft available per hour.

3.2.2. Reward Function Design

The reward function serves as a fundamental element guiding the decision-making process within our algorithm. In the context of decision-making, a well-designed reward function can effectively steer the system towards making optimal choices. For example, if the reward is defined as the net profit from aircraft operations minus maintenance costs, the algorithm will be incentivized to allocate maintenance resources in a way that maximizes this net profit. This means that it will prioritize maintaining high-revenue-generating aircraft and minimize unnecessary maintenance on less-profitable ones. Regarding learning efficiency, the reward function plays a crucial role in how quickly and accurately the algorithm converges to an optimal solution. A simple and straightforward reward function can lead to faster learning as the agent can more easily understand the relationship between its actions and the resulting rewards. However, in more complex scenarios, a more sophisticated reward function might be necessary. For instance, if we incorporate long-term consequences into the reward function, such as the impact of current maintenance decisions on future aircraft availability and profitability, the algorithm can learn to make more far-sighted decisions. Although this might initially slow down the learning process due to the increased complexity it entails, it can lead to more sustainable and optimal long-term performance.

To this end, we designed our reward function based on an existing scheduling strategy network [2] to align with the designed heterogeneous hypergraph. First, the policy learned through training the scheduling strategy network is denoted as $\pi_\theta(u|o)$, where θ represents the parameters of the heterogeneous hypergraph neural network. In the aircraft maintenance scheduling environment, an action corresponds to a set of maintenance decisions, i.e., $u_t = \{d_1, d_2, \cdots, d_n\}$, where n ranges from 0 (no scheduled maintenance) to N_{avail} (the number of aircraft assigned to each available maintenance staff member). To handle this flexible space, u_t is restructured as an ordered decision-making strategy, where each successive decision (e.g., d_i) depends on the previous one (e.g., d_{i-1}). This strategy can then be decomposed into the following formula:

$$p_\theta(u_t|o_t) = \prod_{i=1}^{n} p_\theta(d_i|o_t, d_{1:i-1}) \tag{3}$$

The scheduling strategy network recursively computes the conditional probability $p_\theta(d_i|o_t, d_{1:i-1})$, which is then used to sample each decision. After each maintenance decision, the heterogeneous graph is updated before making the next decision. Finally, the network collects all the decisions and sends them to the aircraft maintenance scheduling environment for execution. The expected value of each decision is implicitly learned through the reward function.

$$R_t^i = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}^i \tag{4}$$

The pseudocode for the resource scheduling strategy algorithm based on heterogeneous hypergraphs proposed in this paper is detailed below (Algorithm 1). The construction and learning process of the heterogeneous hypergraph will be introduced in subsequent sections.

Algorithm 1 Resource scheduling strategy based on heterogeneous hypergraph.

Input: Training epochs K; Number of batches per epoch N; Learning rate η; Number of gradient update iterations per epoch N_{iter}; Epoch length parameters T_{min}, T_{step} and T_{range}.
Output: The trained policy π_θ
1. Initialize the policy network parameters θ_0
2. **for** t = 1 to K
3. Average the sample length T (T_{min}, T_{min} + T_{range})
4. The extraction of a random aircraft scheduling environment instance E_k
5. **for** i = 1 to N
6. Reset the aircraft scheduling environment instance to t = 0
7. **for** t = 0 to $T - 1$
8. Obtain the observation o_t^i from the aircraft scheduling environment instance E_k
9. Construct the heterogeneous hypergraph and input the node features
10. Use a multi-head attention mechanism-based hypergraph convolutional network to update the node features
11. Calculate the decision probability values $m_{pi,cj}$ based on the node features to obtain policy π_θ
12. Sample $u_t = \{d_1, d_2, \ldots d_n\}$ from the policy π_θ
13. Step into the aircraft scheduling environment E_k and obtain the intermediate reward r_t^i
14. Temporarily store $\{(o_t^i, u_t^i, r_t^i)\}$
15. **if** the number of damaged aircraft > 80% **then**
16. Terminate the current epoch and set future rewards to 0
17. **end**
18. **end**
19. **end**
20. Compute the reward: $R_t^i = \Sigma_{t'=t}^{T} \gamma^{t'-t} r_{t'}^i$
21. Compute the advantage estimate: $A_t^i = R_t^i - \frac{1}{N}\Sigma_{i'=1}^{N} R_t^{i'}$
22. **for** j = 1 to N_{iter}
23. Compute the clipped surrogate objective $L(\theta)$ using Equation (12)
24. Use the Adam optimizer with learning rate η to perform stochastic gradient ascent on the derivative of $L(\theta)$
25. **end**
26. **if** $T_{min} \leq T_{min} + T_{step}$
27. $T_{min} = T_{min} + T_{step}$
28. **end**
29. **end**

3.3. Heterogeneous Hypergraph Network

The overall process of the aircraft maintenance resource scheduling algorithm was introduced in the previous section. This section provides a detailed explanation of the heterogeneous hypergraph component within the algorithm.

3.3.1. Node Initialization

When developing a heterogeneous graph representation for a resource optimization problem, the computational objective is to model entities in the environment (i.e., aircraft, maintenance personnel) and RL components (i.e., states, decisions) within the same graph to enable joint learning of the problem and strategy. Our graph consists of four types of nodes:

1. Aircraft Node $\vec{h}_p$: The initial features of this type of node are represented by the observable parameters of the aircraft.
2. Maintenance Personnel Node $\vec{h}_c$: The initial features of this type of node are represented by the observable parameters of the maintenance personnel.
3. State Node $\vec{h}_s$: The initial features of this type of node consist of metadata related to the problem being solved (e.g., the types and quantities of aircraft). The inclusion of state nodes enables the policy network to explicitly learn high-level global embeddings to estimate the problem state, regardless of the problem's scale.
4. Decision Value Node $\vec{h}_d$: The initial features of this type of node are represented by the value of each decision. By introducing decision value nodes, the heterogeneous graph is enhanced to allow the policy to handle varying numbers of schedules. The initial features of the decision nodes are set as zero vectors.

3.3.2. Hyperedge Initialization

We model the interactions between entities as directed edges to construct a heterogeneous hypergraph. For example, if a maintenance worker is performing maintenance on an aircraft, an edge is established between them. Relationships between multiple nodes are described using either triplets or pairs. The types of edges and their initial weights are initialized as follows:

Triplets: <Aircraft, Helicopter, State> with weight $W_{planein}$, <Aircraft, Helicopter, Decision> with weight $W_{planeto}$, and <Helicopter, Aircraft, Maintenance Personnel> with weight W_{repair}.

Pairs: <Maintenance Personnel, State> with weight W_{crewin}, <State, Decision> with weight $W_{stateto}$, <Aircraft, Aircraft> with weight W_{plane}, <Helicopter, Helicopter> with weight W_{plane}, <Maintenance Personnel, Maintenance Personnel> with weight W_{crew}, <State, State> with weight $W_{stateto}$, and <Decision, Decision> with weight $W_{decision}$. The initial weights are randomly generated.

3.3.3. Heterogeneous Hypergraph Learning

After initializing the nodes and hyperedges, a heterogeneous hypergraph is constructed for the aircraft maintenance scheduling environment, as shown in Figure 2. It encompasses all types of nodes and edges. Self-loops are added to all nodes to account for their own features from the previous layer during the current layer's computation. The input consists of data obtained from the virtual environment, including aircraft and maintenance personnel information, while the output consists of a tuple $< p_i, c_j >$, where each decision is made through a graph neural network. At each time step, multiple decisions are made, implying that the weight adjustments from one decision will influence subsequent decision-making processes, reflecting the Markov property inherent in reinforcement learning (RL). The optimization objective is to maximize the value function, implicitly inferring the long-term value of each decision.

Since each decision influences subsequent ones, choosing the decision with the highest immediate reward may not necessarily lead to the optimal outcome. It is therefore crucial to consider the long-term value of decisions. This paper utilizes a heterogeneous hypergraph to model and learn the long-term value of decisions, thereby selecting the most optimal decision. The following are the feature update equations for the nodes in the heterogeneous hypergraph. Here, $\sigma(\,)$ denotes the ReLU activation function.

1. Feature update equation for aircraft nodes:

$$\vec{h}'_p = \sigma\left(W_{plane}\vec{h}_p + W_{repair}\vec{h}_c\right) \tag{5}$$

2. Feature update equation for maintenance crew nodes:

$$\vec{h}'_c = \sigma\left(W_{repair-by}\vec{h}_p + W_{crew}\vec{h}_c\right) \tag{6}$$

3. Feature update equation for state nodes:

$$\vec{h}'_s = \sigma\left(\sum_{p\in N_{planein(s)}} \alpha_{s,p}^{plainin} W_{planein}\vec{h}_p + \sum_{c\in N_{crewin(s)}} \alpha_{s,c}^{crewin}\vec{h}_c + W_{state}\vec{h}_s\right) \tag{7}$$

4. Feature update equation for decision nodes:

$$\vec{h}'_d = \sigma\left(W_{planeto}\vec{h}_p + W_{stateto}\vec{h}_s + W_{decision}\vec{h}_d\right) \tag{8}$$

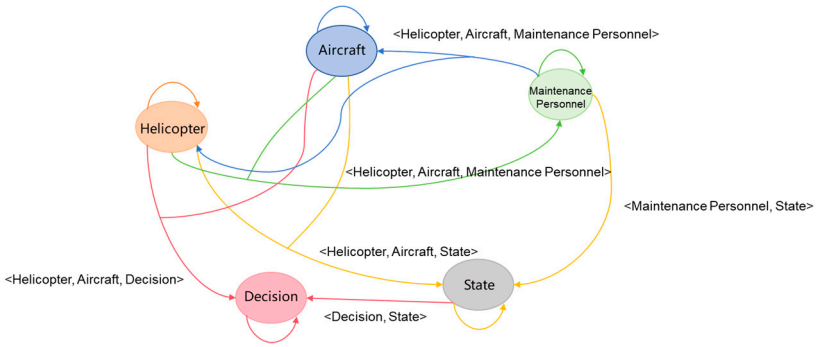


Figure 2. Construction of heterogeneous hypergraph network.

When using the feature update equation for a state node, this paper adopts a multi-head attention mechanism for the graph, which evaluates the input of each edge without a predefined structure, based on feature vectors. The attention coefficient $\alpha_{s,i}^{edgename}$ for each edge is computed from the features of the source and target nodes, as shown in Equation (9). Here, $\vec{a}^{T}_{edgename}$ represents the learnable weights, $||$ denotes the concatenation operation, and $\sigma'(\,)$ is the LeakyReLU function.

$$\alpha_{s,i}^{edgename} = softmax_i\left(\sigma'\left(\vec{a}^{T}_{edgename}\left[W_{state}\vec{h}_s \middle|\middle| W_{edgename}\vec{h}_i\right]\right)\right) \tag{9}$$

We use K independent heterogeneous hypergraph layers to compute the node features in parallel. The results are then combined into multi-head attention outputs, m_{p_i,c_j}, as shown in Equation (10). These node features are subsequently used to compute m_{p_i,c_j}, which represents the probability of selecting the pair $< p_i, c_j >$ as the decision d, as shown in Equation (11).

$$H'_{i\in N} = \frac{1}{K}\sum_{k=1}^{K} h_i^k \tag{10}$$

$$m_{p_i,c_j} = \frac{\exp\left(h_d^{p_i,c_j}\right)}{\sum_{\substack{0\le i\le p\\ 0<j<c}} \exp\left(H_d^{i,j}\right)} \tag{11}$$

We employ the Clip method, which is based on the widely used Proximal Policy Optimization (PPO) approach, for updating the policy gradient, with the main purpose

of restricting the magnitude of policy updates, thereby allowing for a distinct method of updating the neural network parameters. This method is shown in Equation (12).

$$L^{CLIP}(\theta) = \hat{E}\left[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right] \quad (12)$$

Here, the function $r_t(\theta)$ represents the probability ratio between the current and the old policy on the collected rollout data, A_t is the advantage term, and ε is the clipping parameter.

To ensure more efficient resource utilization, we adopt the same strategy as HetGPO-skip. Specifically, during node initialization, a special aircraft node with an ID of 0 is added. If the selected decision d_i contains aircraft ID 0, the decision will not be adopted. This means that the policy network allows for the possibility of abstaining from making a decision in certain situations, thereby equipping the network with the ability to prevent future scheduling issues.

4. Experiments and Analysis

4.1. Experimental Setup

We investigate the performance of our heterogeneous hypergraph-based resource scheduling algorithm with respect to multiple optimization objectives and compare it to other algorithms to highlight its advantages. Environmental situations containing varying problem sizes and random initializations were generated, which were saved as test datasets. Three different environment scales were considered (Table 2):

1. Small scale: The number of passenger aircraft, helicopters, and maintenance personnel are selected from the ranges [16, 24], [8, 12], and [6, 8], respectively, using a uniform distribution.
2. Medium scale: The ranges for the number of passenger aircraft, helicopters, and maintenance personnel are [32, 48], [16, 24], and [12, 16], respectively.
3. Large scale: The ranges for the number passenger aircraft, helicopters, and maintenance personnel are [64, 96], [32, 48], and [24, 32], respectively.

Table 2. Statistics of the datasets.

Dataset	Number of Aircraft	Number of Helicopters	Number of Maintenance Personnel
Large scale	[64, 96]	[32, 48]	[24, 32]
Medium scale	[32, 48]	[16, 24]	[12, 16]
Small scale	[16, 24]	[8, 12]	[6, 8]

For each test environment scenario, the algorithm was evaluated across eight episodes. In each episode, the resource scheduling algorithm under evaluation ran in a simulated predictive maintenance environment. The simulation time for each algorithm was fixed (set to a virtual period of thirty days in this paper). The results were averaged from the system data, and performance metrics were computed to derive the final experimental results. The models were trained and evaluated on a server equipped with an NVIDIA RTX 4090/24GB GPU and an Intel Xeon(R) Platinum 8325V/16-core processor.

4.2. Comparison of Algorithms

To validate the effectiveness of the heterogeneous hypergraph-based aircraft maintenance resource scheduling method, we compared it with five advanced heuristic-based resource scheduling methods and three state-of-the-art deep learning-based resource scheduling algorithms.

4.2.1. Heuristic Resource Scheduling Algorithms

1. Random Scheduler [25]: At time t, the Random Scheduler algorithm assigns each available maintenance personnel to a randomly selected aircraft for maintenance, and selects a set of aircraft u_t from all the aircraft that have not yet undergone maintenance.

2. Corrective Scheduler [3]: This scheduler plans corrective maintenance tasks only for components that have already failed. When multiple aircraft experience failures at time t, these aircraft are queued in a priority list based on their hourly revenue, and maintenance is performed according to this list.

3. Condition-Based Scheduler [4]: Condition-Based Maintenance (CBM) has been shown to improve system efficiency by reducing the number of corrective maintenance tasks required. In addition to addressing the failures of all aircraft components, the Condition-Based Scheduler also places non-faulty aircraft in another priority queue, assigning remaining maintenance personnel to perform CBM. A threshold β_c is set for aircraft that do not qualify for the priority queue, serving as a measure to balance aircraft availability and future risk of failure. The choice of β_c plays a crucial role in determining the scheduler's overall performance.

4. Periodic Scheduler [14]: This scheduler arranges regular maintenance tasks at predetermined time intervals. After a period of β_p, the Periodic Scheduler assigns each available maintenance personnel to an aircraft for maintenance. Aircraft are prioritized first by their fault status and then by their flight hours. The performance of the Periodic Scheduler is closely related to the choice of β_p.

5. Model-Based Planning: To construct a model-based scheduler, the Condition-Based Scheduler is enhanced to incorporate the aircraft failure model within its system, planning based on the actual probability of failure. In this case, non-faulty aircraft are sorted according to the likelihood of their failure in the next time step, with a failure threshold of β_u.

4.2.2. Resource Scheduling Algorithms Based on Deep Learning

1. DeepRM [7]: DeepRM represents resource allocation as a fixed-size tensor and uses a feedforward neural network to learn a strategy with a fixed output size. The input to the strategy network is constructed by concatenating the flattened features of the aircraft, maintenance personnel, and their states. To handle variable problem sizes, DeepRM pads the feature tensors of the aircraft and maintenance personnel with zeros to fit the maximum number of aircraft and maintenance personnel. The output size of the strategy network is also set to the maximum number of aircraft. When generating the probability distribution of its decisions, this method masks invalid aircraft (i.e., aircraft already under maintenance) from the network output.

2. Decima [8]: Decima employs a scalable architecture that combines a graph neural network (GNN) with a separate policy network controlled by scheduling events. In the aircraft maintenance scheduling environment, the GNN used in Decima is a bipartite graph where the maintenance task nodes are the child nodes and the global state summary node serves as the parent node. Considering the homogeneity of personnel, the construction of maintenance task nodes follows a similar approach to the aircraft nodes used in this study.

3. HetGPO [3]: HetGPO utilizes a dynamic scheduler that makes decisions based on a heterogeneous graph approach. The method includes three distinct decision strategies: (1) HetGPO-Full—For each available maintenance personnel, the policy assigns an aircraft for maintenance. A placeholder aircraft with ID 0 is included, which, when selected, indicates a "no operation" action for the specified maintenance personnel. (2) HetGPO-Skip—Although the policy still recursively assigns aircraft to available maintenance personnel, the aircraft with ID 0 now acts as a "skip" token. Choosing this aircraft indicates that the policy does not intend to schedule any further maintenance tasks and

aims to proceed to time step t + 1 Aircraft. (3) HetGPO-Single—During training, the policy is restricted to issuing only one scheduling decision at any given time step. We provide the parameter settings of the models in Table 3.

Table 3. The parameter settings of the models for all the test datasets.

Parameter	O1	O2	O3
K	2000	2000	2000
N	8	8	8
η	10^{-3}	10^{-3}	10^{-3}
N_{iter}	3	3	3
T_{min}	50	50	50
T_{step}	0.8	0.8	0.8
γ	0.99	0.99	0.99
r	e^{-3}	e^{-3}	e^{-3}
ε	0.2	0.2	0.2

4.3. Evaluation Metrics

To comprehensively evaluate the effectiveness of the algorithms, two metrics are used in this experiment.

M1: For each scheduler, an identical environment is simulated and different reward functions are applied based on the optimized objectives (O1, O2, O3). The average reward per hour is calculated according to the selected optimization goal. For example, if O1 is chosen, the returned average reward per hour represents the profit after subtracting costs, or the number of available aircraft per hour. This value is then normalized by comparing it with the average reward or total number of aircraft generated in an ideal environment where no aircraft are damaged. The normalized value is then used.

M2: Using the Corrective Scheduler as the baseline strategy, the results of each algorithm are evaluated by running them for the same time span in the predictive maintenance environment. The percentage of the improvement in the target strategy relative to the baseline strategy is computed. We let the reward of the target strategy be R_1 and the reward of the baseline Corrective Scheduler be R_0; then, the improvement percentage M2 is calculated as shown in Equation (13).

$$M2 = \frac{R_1 - R_0}{R_0} \times 100\% \tag{13}$$

4.4. Analysis of Experimental Results

Tables 4–6 present the results of the evaluation of various resource scheduling algorithms across multiple optimization objectives (O1, O2, O3). The labels 'small', 'medium', and 'large' correspond to the small-, medium-, and large-scale datasets, respectively. The results demonstrate that the proposed heterogeneous hypergraph-based aircraft maintenance resource scheduling algorithm outperforms all baseline methods across all dataset sizes. Moreover, the proposed method does not rely on domain-specific knowledge and can automatically determine strategies. The results from Tables 4–6 are summarized as follows:

(1) In comparison with heuristic resource scheduling algorithms, the proposed algorithm demonstrates better high-order relationship modeling capabilities, achieving the best results across all three optimization objectives without relying on manual guidance, and it also exhibits better robustness. The Random method, compared to the baseline Condition-based Scheduler, lacks domain-specific knowledge and does not consider the impact of previous decisions on subsequent ones. As a result, its M2 metric for all optimization objectives is negative. The Condition-based Scheduler, which estimates aircraft failure probabilities, leverages a priority queue. It can select the aircraft that most urgently need maintenance and assign the most suitable maintenance personnel. However, both the

Condition-based Scheduler and Periodic Scheduler depend on manually selected thresholds. If the thresholds β_c or β_p deviate from the optimal values, their performance rapidly deteriorates. Additionally, selecting these thresholds requires repetitive experiments and input from experts in operations research to define the valid range for these thresholds. Thus, the Condition-based Scheduler and Periodic Scheduler cannot be used without expert intervention. While the Model-Based Planning approach outperforms the Condition-based Scheduler, its optimization results still do not reach those of the proposed method.

(2) In comparison to deep learning-based resource scheduling methods, the proposed algorithm leverages a hypergraph structure, which offers superior multivariate relationship modeling capabilities and achieves better results across all evaluation metrics. DeepRM, however, fails to learn useful scheduling strategies, and its performance is close to that of the Random Scheduler. This indicates that the fixed-size tensor representation used in DeepRM is ineffective in modeling heterogeneous scheduling environments and cannot support effective end-to-end learning strategies in a predictive maintenance context. In contrast, Decima uses a graph neural network to learn structural information via global aggregate nodes, enabling it to adopt scheduling strategies comparable to those of the Condition-based Scheduler under O1 and O2. However, the graph structure in Decima only supports a single round of message passing between the nodes, and its global node does not utilize attention mechanisms. Although HetGPO demonstrated a superior performance compared to previous methods, it overlooks the heterogeneity of aircraft data. By employing graph neural networks (GNNs) for modeling, it fails to accurately capture the complex relationships between aircraft and maintenance personnel. Due to the differences in the aircraft types modeled, the data associated with the two types of aircraft are heterogeneous. This heterogeneity means that the relationship between aircraft and maintenance personnel is not a simple binary relationship but rather a many-to-many relationship. Therefore, employing hypergraphs for modeling can significantly enhance the efficiency of the network. Hypergraphs allow for the representation of complex, high-order interactions between multiple entities, which is particularly useful for capturing the intricate relationships and dependencies in this sort of heterogeneous data. The limitations in model expressiveness cause Decima to perform worse than the proposed method on evaluation metrics M1 and M2. In addition to its performance advantages, the proposed method demonstrates greater robustness and consistency. The standard deviation of its M1 metric is five times smaller than that of the heuristic methods, twice as small as that of Decima, and superior to HetGPO, the best-performing baseline algorithm.

Table 4. Comparison of experimental results of resource scheduling algorithms when optimizing O1.

Method	M1 (Small)	M2 (%) (Small)	M1 (Medium)	M2 (%) (Medium)	M1 (Large)	M2 (%) (Large)
Random	0.522	−2.87	0.532	−2.65	0.533	−2.23
Corrective	0.539	0.0	0.547	0.0	0.546	0.0
Condition-based	0.656	21.7	0.661	20.8	0.648	18.6
Periodic	0.599	11.1	0.598	9.38	0.587	7.52
Model-based	0.669	24.0	0.671	22.8	0.658	20.5
DeepRM	0.533	−0.88	0.538	−1.47	0.539	−1.11
Decima	0.651	21.1	0.660	20.9	0.663	21.6
HetGPO-Single	0.680	26.4	0.676	23.7	0.666	22.3
HetGPO-Skip	0.695	29.1	0.697	27.5	0.695	27.5
HetGPO-Full	0.693	28.8	0.694	27.1	0.693	27.1
Ours	0.695	29.2	0.698	27.6	0.696	27.6

Table 5. Comparison of experimental results of resource scheduling algorithms when optimizing O2.

Method	M1 (Small)	M2 (%) (Small)	M1 (Medium)	M2 (%) (Medium)	M1 (Large)	M2 (%) (Large)
Random	0.611	−5.63	0.618	−5.74	0.618	−5.68
Corrective	0.648	0.0	0.547	0.0	0.655	0.0
Condition-based	0.724	11.6	0.727	10.7	0.718	9.45
Periodic	0.675	4.06	0.677	3.05	0.587	7.52
Model-based	0.728	12.3	0.732	11.5	0.658	20.5
DeepRM	0.625	−3.54	0.626	−4.66	0.539	−1.11
Decima	0.725	11.9	0.730	11.3	0.732	11.8
HetGPO-Single	0.736	13.7	0.735	12.0	0.730	11.5
HetGPO-Skip	0.747	15.3	0.748	14.0	0.747	14.0
HetGPO-Full	0.747	15.4	0.749	14.1	0.747	14.1
Ours	0.747	15.4	0.750	14.3	0.748	14.2

Table 6. Comparison of experimental results of resource scheduling algorithms when optimizing O3.

Method	M1 (Small)	M2 (%) (Small)	M1 (Medium)	M2 (%) (Medium)	M1 (Large)	M2 (%) (Large)
Random	0.699	−2.97	0.706	−3.26	0.705	−3.00
Corrective	0.722	0.0	0.730	0.0	0.728	0.0
Condition-based	0.810	12.0	0.812	11.1	0.796	9.16
Periodic	0.747	3.19	0.743	1.48	0.725	−0.67
Model-based	0.814	12.6	0.817	11.8	0.803	10.0
DeepRM	0.708	−1.82	0.709	−2.88	0.706	−2.90
Decima	0.748	3.98	0.755	3.69	0.760	4.76
HetGPO-Single	0.842	13.7	0.840	15.2	0.837	15.1
HetGPO-Skip	0.847	13.7	0.848	15.2	0.847	15.1
HetGPO-Full	0.849	13.7	0.849	15.2	0.849	15.1
Ours	0.850	13.9	0.849	15.3	0.850	15.4

In summary, the hypergraph-based structure used in our method contributes to improved scheduling in several ways. Firstly, as mentioned before, it can capture the complex relationships among the entities in the scheduling problem accurately. For example, it can model the relationships between multiple relevant factors precisely. This accurate representation leads to more informed decision-making in the scheduling process. Secondly, its hypergraph-based structure enables more efficient information propagation among different elements in the scheduling network. It can quickly update the status and relationships of entities as the scheduling process evolves. As for the trade-offs the hypergraph introduces, one possible trade-off is increased computational complexity. Since hypergraphs can represent complex relationships, the operations within hypergraphs, such as message passing in hypergraph neural networks, can be more computationally demanding compared to those of simpler graph-based models. However, we believe that the benefits in terms of more accurate scheduling and the better handling of complex relationships outweigh this cost in our specific application scenario.

In practical maintenance operations, the improvement our model enables has a direct and significant impact on the financial aspect of maintenance. When considering the cumulative profit seen over multiple time steps, our new approach, which shows a significant improvement, results in a total profit that is higher compared to that of the original algorithm. This increase in profit has several far-reaching benefits. Firstly, from a cost-efficiency perspective, the higher profit margin allows maintenance teams to invest more in high-quality maintenance equipment and training for their personnel. This, in turn, could lead to more effective and efficient maintenance procedures, reducing the likelihood of equipment failures and costly breakdowns in the long run. Secondly, the additional profit can be used to expand the scope of maintenance services. For example, maintenance providers could afford to take on more complex maintenance tasks or serve a larger customer base, enhancing their competitiveness in the market. Overall, this improvement could help bring

substantial and practical advantages to real-world maintenance operations, contributing to both the economic viability and the quality of maintenance services.

There are several challenges to be addressed in the future. Heterogeneous hypergraphs and POMDPs are both computationally intensive algorithms, imposing high demands on hardware resources. Additionally, whether the same approach would be effective for addressing other resource scheduling problems and whether there exist more superior modeling methods are also challenges that this paper needs to address.

5. Conclusions

This paper proposes a resource scheduling algorithm based on heterogeneous hypergraphs, which uses those hypergraphs to represent the higher-order relationships between multiple nodes. This approach better captures complex many-to-many dependencies and enables the automatic learning of maintenance scheduling decisions. Our experimental results demonstrate that the proposed method outperforms heuristic resource scheduling algorithms and existing deep learning-based methods across two evaluation metrics (M1, M2), three data scales, and three optimization objectives (O1, O2, O3). Furthermore, the model's training process no longer requires domain-specific expert knowledge. Compared to graph neural network (GNN)-based methods, the proposed approach achieves a similar overall performance while delivering certain improvements in specific metrics. These findings fully validate the effectiveness and adaptability of our heterogeneous hypergraph-based resource scheduling algorithm in dynamic and complex environments, offering a novel and efficient solution to the maintenance scheduling problem. However, this method still has limitations. In the network constructed using hypergraphs, simple hypergraphs can express certain relationships but may not fully leverage the advantages of hypergraphs when dealing with complex relationships, which could even lead to information loss. Meanwhile, adopting POMDP involves solving the problem within a fixed-environment model. In cases where the environment's structure and parameters undergo dynamic changes, the adaptability of this model is relatively poor. When significant changes occur in the environment, it will have a certain impact on the model's performance. In the future, it is hoped that a dynamic scheduling strategy that is more adaptive can be identified. Moreover, efforts should be directed towards exploring modeling methods that are more effective than hypergraphs to achieve further enhancements in model performance.

Author Contributions: Investigation, M.H. and J.Z.; methodology, J.Z. and Y.H.; validation, L.K.; writing—original draft preparation, L.K. and M.H.; writing—review and editing, J.Z. and Y.H.; supervision, B.X. and H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 62406146) supported by Adolescent Education and Intelligence Support Lab.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in this paper are publicly available.

Conflicts of Interest: Author Long Kang was employed by the company AECC Harbin Dongan Engine Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Zhou, B.; Bao, J.; Li, J.; Lu, Y.; Liu, T.; Zhang, Q. A novel knowledge graph-based optimization approach for resource allocation in discrete manufacturing workshops. *Robot. Comput.-Integr. Manuf.* **2021**, *71*, 102160. [CrossRef]
2. Shah, J.A.; Williams, B.C. Fast dynamic scheduling of disjunctive temporal constraint networks through incremental compilation. In Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, 14–18 September 2008; pp. 322–329.
3. Wang, Z.; Gombolay, M. Stochastic resource optimization over heterogeneous graph neural networks for failure-predictive maintenance scheduling. *Proc. Int. Conf. Autom. Plan. Sched.* **2022**, *77*, 102268. [CrossRef]
4. Ren, H.; Qu, J.; Chai, Y. Research status and challenges of deep learning in the field of fault diagnosis. *Control Decis.-Mak.* **2017**, *32*, 14.
5. Yam, R.C.M.; Tse, P.; Li, L.; Tu, P. Intelligent predictive decision support system for condition-based maintenance. *Int. J. Adv. Manuf. Technol.* **2001**, *17*, 383–391. [CrossRef]
6. Ahmad, R.; Kamaruddin, S. An overview of time-based and condition-based maintenance in industrial application. *Comput. Ind. Eng.* **2012**, *63*, 135–149. [CrossRef]
7. Mao, H.; Alizadeh, M.; Menache, I.; Kandula, S. Resource management with deep reinforcement learning. In Proceedings of the 15th ACM Workshop on Hot Topics in Networks, Atlanta, GA, USA, 29–30 November 2016; pp. 50–56.
8. Mao, H.; Schwarzkopf, M.; Venkatakrishnan, S.B.; Meng, Z.; Alizadeh, M. Learning scheduling algorithms for data processing clusters. In Proceedings of the SIGCOMM'19: ACM SIGCOMM 2019 Conference, Beijing, China, 19–23 August 2019; pp. 270–288.
9. Hu, Z.; Dong, Y.; Wang, K.; Chang, K.W.; Sun, Y. GPT-GNN: Generative pre-training of graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 6–10 July 2020; pp. 1857–1867.
10. Ma, S.; Liu, J.; Zuo, X. Overview of graph neural networks. *Comput. Res. Dev.* **2022**, *59*, 47–80.
11. Zhan, Z.-H.; Liu, X.-F.; Gong, Y.-J.; Zhang, J.; Chung, H.S.-H.; Li, Y. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv.* **2015**, *47*, 1–33. [CrossRef]
12. Khallouli, W.; Huang, J. Cluster resource scheduling in cloud computing: Literature review and research challenges. *J. Supercomput.* **2022**, *78*, 6898–6943. [CrossRef]
13. Achamrah, F.E.; Attajer, A. Multi-objective reinforcement learning-based framework for solving selective maintenance problems in reconfigurable cyber-physical manufacturing systems. *Int. J. Prod. Res.* **2024**, *62*, 3460–3482. [CrossRef]
14. Stenström, C.; Norrbin, P.; Parida, A.; Kumar, U. Preventive and corrective maintenance–cost comparison and cost–benefit analysis. *Struct. Infrastruct. Eng.* **2016**, *12*, 603–617. [CrossRef]
15. Zhou, N. Research on Gesture Recognition Algorithm Based on Convolutional Neural Network. Master's Thesis, Guilin University of Technology, Guangxi, China, 2022.
16. Wang, C.; Zang, X.; Song, C.; Liu, Z.; Zhao, J.; Ang, M.H. Virtual tactile POMDP-based path planning for object localization and grasping. *Measurement* **2024**, *230*, 11–44. [CrossRef]
17. Feng, Y.; You, H.; Zhang, Z.; Ji, R.; Gao, Y. Hypergraph neural networks. In Proceedings of the 33rd AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 3558–3565.
18. Gao, Y.; Feng, Y.; Ji, S.; Ji, R. HGNN+: General hypergraph neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 3181–3199. [CrossRef] [PubMed]
19. Yadati, N.; Nimishakavi, M.; Yadav, P.; Nitin, V.; Louis, A.; Talukdar, P. HyperGCN: A new method for training graph convolutional networks on hypergraphs. In Proceedings of the Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 1509–1520.
20. Young, S.; Gasic, M.; Thomson, B.; Williams, J.D. POMDP-based statistical spoken dialog systems: A review. *Proc. IEEE* **2013**, *101*, 1160–1179. [CrossRef]
21. Araya, M.; Buffet, O.; Thomas, V.; Charpillet, F. A POMDP extension with belief-dependent rewards. *Adv. Neural Inf. Process. Syst.* **2010**, *23*. Available online: https://proceedings.neurips.cc/paper/2010/hash/68053af2923e00204c3ca7c6a3150cf7-Abstract.html (accessed on 11 February 2025).
22. Zhao, Q.; Tong, L.; Swami, A.; Chen, Y. Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework. *IEEE J. Sel. Areas Commun.* **2007**, *25*, 589–600. [CrossRef]
23. Elsborg, J.; Bhowmik, A. ArtiSAN: Navigating the complexity of material structures with deep reinforcement learning. *ChemRxiv* **2024**, *23*, 50–72. [CrossRef]

24. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef] [PubMed]
25. Burckhardt, S.; Kothari, P.; Musuvathi, M.; Nagarakatte, S. A randomized scheduler with probabilistic guarantees of finding bugs. *ACM SIGARCH Comput. Archit. News* **2010**, *38*, 167–178. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.