algorithms

MDPI

Article

Multi-Objective Order Scheduling via Reinforcement Learning

Sirui Chen [1], Yuming Tian [2,3,*] and Lingling An [2]

[1] Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China; 21181214083@stu.xidian.edu.cn
[2] School of Computer Science and Technology, Xidian University, Xi'an 710071, China; an.lingling@gmail.com
[3] Key Laboratory of Smart Human-Computer Interaction and Wearable Technology of Shaanxi Province, Xi'an 710071, China
* Correspondence: ymtian@mail.xidian.edu.cn

Abstract: Order scheduling is of a great significance in the internet and communication industries. With the rapid development of the communication industry and the increasing variety of user demands, the number of work orders for communication operators has grown exponentially. Most of the research that tries to solve the order scheduling problem has focused on improving assignment rules based on real-time performance. However, these traditional methods face challenges such as poor real-time performance, high human resource consumption, and low efficiency. Therefore, it is crucial to solve multi-objective problems in order to obtain a robust order scheduling policy to meet the multiple requirements of order scheduling in real problems. The priority dispatching rule (PDR) is a heuristic method that is widely used in real-world scheduling systems In this paper, we propose an approach to automatically optimize the Priority Dispatching Rule (PDR) using a deep multiple-objective reinforcement learning agent and to optimize the weighted vector with a convex hull to obtain the most objective and efficient weights. The convex hull method is employed to calculate the maximal linearly scalarized value, enabling us to determine the optimal weight vector objectively and achieve a balanced optimization of each objective rather than relying on subjective weight settings based on personal experience. Experimental results on multiple datasets demonstrate that our proposed algorithm achieves competitive performance compared to existing state-of-the-art order scheduling algorithms.

Keywords: priority dispatching rule; order dispatching; multi-objective reinforcement learning

Citation: Chen, S.; Tian, Y.; An, L. Multi-Objective Order Scheduling via Reinforcement Learning. *Algorithms* **2023**, *16*, 495. https://doi.org/10.3390/a16110495

Academic Editor: Gianlorenzo D'Angelo

Received: 11 September 2023
Revised: 7 October 2023
Accepted: 13 October 2023
Published: 24 October 2023

1. Introduction

Order scheduling is a crucial decision-making problem in supply chain management and the manufacturing industry; it plays an important role in rational resource allocation and utilization, making companies more competitive in the global marketplace. Before an order can be initiated for manufacturing, there are several preliminary steps, termed as 'pre-manufacturing activities', that must be undertaken. Moreover, in light of the various uncertainties in the actual production cycle, the daily output for any order may not always align with initial expectations. In reality, before a new order can be put into production, a series of activities known as preproduction events need to be completed. In addition, in real production process, the daily production quantity of each order is not always as expected owing to various uncertainties [1]. For example, the rapid development of technology and the exponential growth of mobile internet services in China have led to a significant increase in the volume of mobile communications and the complexity of work processes. In response, dispatching system providers are actively seeking ways to improve the efficiency and cost effectiveness of work order scheduling. While current order scheduling partially meet operators' requirements, they face challenges such as inefficient utilization of human resources and inability to fully automate work order dispatching.

Previous research on automatic work order scheduling optimization has primarily focused on refining dispatching rules based on the Priority Dispatching Rule (PDR) [2], a widely used method in scheduling problems. Compared to complex optimization techniques such as mathematical programming, PDR offers computational efficiency, intuitiveness, ease of implementation, and inherent adaptability to uncertainties commonly encountered in practice [3]. Current studies for order scheduling can be divided into two categories, namely, heuristic scheduling and deep learning scheduling. The core idea of heuristic scheduling is to find an approximately optimal solution through heuristic rules or methods. Heuristic rules are estimated values or rules based on experience which guide the decision-making process of the selected strategy. However, heuristic scheduling design requires a great deal of specialized knowledge and often delivers limited performance in different situations. Moreover, most order scheduling models rely on heuristic methods [4], lacking a guarantee of global optimality and exhibiting performance variability based on specific problem instances and the designer's experience.

For deep learning scheduling, deep neural network can be used to automatically extract the scheduling based on the data. However, most dispatching algorithms prioritize minimizing the operating time as their sole objective [5]. Despite this, practical scenarios often demand the consideration of multiple metrics. For example, in manufacturing systems, a scheduling algorithm must handle dynamic demand, machine breakdowns, and uncertain processing times in real time. Similarly, in online ride-hailing services, a scheduling algorithm should aim to reduce both the average waiting time for customers and the pick-up cost for drivers [6]. Taking the customer service of a telecom company as an example, the work order assignment process involves the customer, the telecom company, and the operator. When a customer generates a repair work order through a call, the telecom company needs to efficiently assign the nearest repairer to the customer's location while minimizing costs [7]. Therefore, a good scheduling algorithm needs to simultaneously guarantee time, cost, and distance. Figure 1 illustrates the difference between traditional reinforcement learning and multiple-objective reinforcement learning. The latter allows for the simultaneous optimization of three objectives, enabling greater flexibility to meet diverse needs [8].

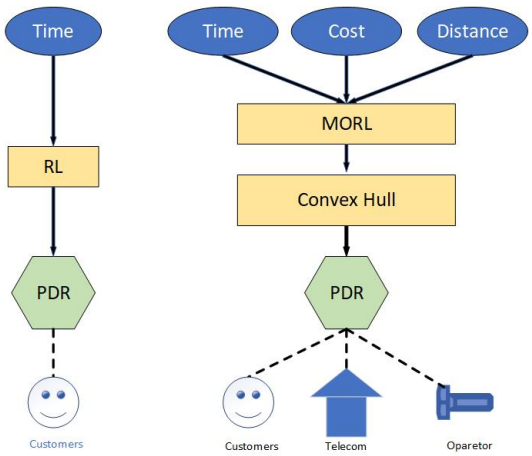


Figure 1. The (**left**) picture presents traditional reinforcement learning with a single objective, while the (**right**) picture presents multiple-objective reinforcement learning.

To address the challenges of multi-objective order scheduling, we need to answer the following three questions: (1) how to formulate the multi-objective order scheduling problems? (2) how to construct policy for perform order scheduling? and (3) how to obtain a Pareto-optimal policy of order scheduling to meet the multiple requirements of order scheduling? For the first question, we formulate the problem as a Multiple-Objectives Markov Decision Process (MOMDP), in which we represent the states using a disjunctive graph representation. This discrete graph structure effectively captures the characteristics

of the order scheduling problem. For the second question, we introduce a Graph Neural Network (GNN) to construct a robust policy network. The GNN utilizes fixed-dimensional embeddings to represent nodes in the disjunctive graph, enabling effective decision-making for order scheduling. By learning the graph structure data, extracting relevant features, and exploring patterns within the graph, the GNN provides valuable insights. For the last question, we employ policy gradient methods to train the network and overcome the limitations associated with traditional approaches i order to obtain high quality Priority Dispatching Rules (PDRs). However, in multi-objective reinforcement learning, designing the reward function with subjective weight settings for each index often leads to imbalanced objective optimization. To address this issue, our algorithm employs a linear embedding technique that achieves a more balanced solution. Furthermore, to ensure the best optimization of all objectives, we introduce the convex hull algorithm to guarantee that the weight vector yields a maximized linearly scalarized value function.

To the best of our knowledge, no previous work has specifically addressed the same problem as we do here. In this paper, we focus on the unique requirements of communication work order dispatching systems while considering the needs of both operators and customers. By developing a multi-objective deep reinforcement learning approach, it is possible to objectively optimize each crucial objective based on the available data.

2. Related Work

This research focuses on addressing a problem with multiple objectives. In this section, we discuss the problem used in the experiment and the model building process. One relevant problem that we consider is the Job Shop Scheduling Problem (JSSP), which is a well-known optimization problem in operations research.

Traditional approaches to solving the JSSP can be categorized into four categories: analytical techniques, meta-heuristic [9] algorithms, rule-based approaches, and simulation approaches. However, as the environments become more complex, traditional analytical techniques and simple mathematical models may not be capable of effectively analyzing them. While meta-heuristic algorithms have shown good performance in certain environments [9], they cannot guarantee optimal solutions and are often unstable, yielding different results for different instances of the same problem. Rule-based algorithms, although effective, can be costly and difficult to transfer to new environments. In contrast, our proposed method is fully trained and can be directly applied to solve problems of different sizes without the need for transfer learning. Previous research has utilized the Genetic Algorithm (GA) [10] and Shift Bottleneck (SB) [11] methods to address the JSSP. However, these approaches require rebuilding when the environment changes, resulting in high computational costs and inspiring researchers to develop new methods for solving the problem.

In 2020, Reinforcement Learning (RL) was proposed to develop a Priority Dispatching Rule (PDR) for solving the JSSP [12]. This approach utilized Graph Neural Networks (GNNs) to perform embeddings on the disjunctive graph, which captures the processing order and is size-agnostic in terms of both jobs and machines [12]. However, this algorithm solely focuses on time optimization, while many scheduling goals involve multiple objectives, such as minimizing makespan and processing costs. Traditional RL methods cannot effectively handle these multi-objective scenarios, and may face challenges when the environment changes.

In another study proposed by Yang et al. [13], the convex hull method was applied to implement ethical embedding in multiple-objective reinforcement learning [14]. This novel algorithm aligns with current developments in the Multi-Objective Reinforcement Learning (MORL) literature to create an ethical environment as a single-objective Markov Decision Process (MDP) derived from the multi-objective MDP resulting from the reward specification process [13].

Multi-objective optimization has already been applied to solve many different problems. In 2013, considering multiple production departments and multiple production processes,

an NSGA-II-based Pareto optimization model was developed to handle this problem [15]. In a study by Debiao Li, the multi-objective optimization problem (MOP) for minimizing collation delays and makespan was presented for the order scheduling problem in a mail-order pharmacy automation systems with a min–max Pareto objective function [16].

To summarize, while previous research has made progress in addressing the JSSP, there are limitations in handling multiple objectives and adapting to changing environments. Our research aims to address these challenges and to develop an effective and robust solution using multi-objective deep reinforcement learning.

3. Problem Description

In the communication work order scheduling scenario, there is a set of work orders J such as complaint sheets and staff orders M. Because of process or permission, each order $J_i \in J$ always has to be reviewed or processed by different workers, where each step O_{ij} is called the operation of j_i and must be processed by a number n_i workers in an order $O_{i_1} \rightarrow \ldots O_{in_1}$. In addition, each worker can only work on one order at a time. To solve this problem, we need to find a dispatching approach that takes the time as target while simultaneously optimizing multiple objectives. In this paper, we set the extra targets as the finish rate and cost.

3.1. Disjunctive Graph

It is well known that disjunction can represent the JSSP. A disjunctive graph represents the scheduling problem as a directed graph $G = (V, C \cup D)$, which is the set of all the vertices of the directed graph and each vertex represents an operation (with two empty operations to start and end). In particular, C is a set of directed arcs (conjunctions), which are the arcs connecting two adjacent operations of the same job in the directed graph G and D is the set of all the disjunctive arcs, which are the arcs connecting two operations corresponding to an operator in the directed graph G. Consequently, finding a solution to a job-dispatching instance is equivalent to fixing the direction of each disjunction such that the resulting graph is a DAG [2].

3.2. Markov Decision Process Formulation

The continuous decision method based on PDR uses a series of steps to solve JSSP instances. In each step, a set of qualified operations (that is, operations for which previous operations have been scheduled) is first identified. The specific PDR is then applied to calculate the priority index of each qualified operation, and the one with the highest priority is selected for scheduling. Traditional PDRs design the priority by rules, such as selecting the shortest processing time from a set of operations. As mentioned above, solving a job dispatching instance can be viewed as the task of determining the direction of each disjunction. Therefore, we consider the dispatching decisions made by PDRs as actions of changing the disjunctive graph, and formulate the underlying MDP model as follows.

State. The state s_t of the disjunctive graph $G(t) = (O, C \cup D_u(t), D(t))$ presents the solution up to time t, where $D_u(t) \subseteq D$ contains all the disjunctive arcs that have been assigned a direction up to t and $D(t) \subseteq D$ includes arcs which are not dispatched. For each node O in the graph, we record the recursively calculated value for each target and a binary record $I(O, s_t)$, the value of which is be 1 if the node is scheduled in s_t:

$$T_{LB}(O_{ij}, s_t) = T_{LB}(O_{i,j-1}, s_t) + t_{ij}, \quad (1)$$

$$F_{LB}(O_{ij}, s_t) = F_{LB}(O_{i,j-1}, s_t) + f_{ij}, \quad (2)$$

$$C_{LB}(O_{ij}, s_t) = C_{LB}(O_{i,j-1}, s_t) + c_{ij}, \quad (3)$$

where t_{ij} is the process time for this operation, f_{ij} is the degree of completion, and c_{ij} is the cost of this operation. We calculate the target by only considering the precedence constraints from its predecessor.

Action. An action a_t is one executable operation in a decision step t.

Rewards. The reward function is vital for reinforcement learning. At each juncture, the agent selects an action based on the present state according to its policy and the environment generates a reward signal that is the goal the agent wants to maximize during the action selection process.

4. Method

4.1. Weighted Multiple-Objective Markov Decision Process Formulation

In this problem, our aim is to find a way to dispatch the work order step-by-step to make sure all objectives can be optimized. We use a weighted reward to build our reward function. First, we design the reward function for each of the objectives that we intend to optimize. As the quality difference between the partial solutions corresponds to s_t and s_{t+1}, the respective reward functions for the time, cost, and distance are

$$R_t(a_t, s_t) = H_t(s_t) - H_t(s_{t+1}), \tag{4}$$

$$R_c(a_t, s_t) = H_c(s_t) - H_c(s_{t+1}), \tag{5}$$

$$R_d(a_t, s_t) = H_d(s_t) - H_d(s_{t+1}), \tag{6}$$

where $H(\cdot)$ is the quality measure, which we define as the lower bound of the objectives, computed as $\max_{i,j}\{\cdot_{LB}(O_{ij}, S_t)\}$.

Then, we must design an embedding function that scalarises the rewards received by the agent in a way that ensures that ethical–optimal policies are optimal for the agent. This embedding function takes the form of a linear combination of all objectives:

$$f(\vec{R}) = \vec{w} \cdot \vec{R} = w_t \cdot R_t + w_c \cdot R_c + w_f \cdot R_f, \tag{7}$$

where $\vec{w}$ is a weight vector and all weights $w_t, w_c, w_f > 0$ in order to guarantee all of them are optimized. Without of loss of generality, we fix the weights at 1.

4.2. Parameterizing the Policy

For state s_t, our policy outputs a set of probability $pi(a_t \mid s_t)$ over the actions in A_t, and the action with highest probability has the highest priority. The disjunctive graph provides a view of the dispatching process; it records the information of each state in the process, including the processing order, processing time, processing cost. and precedence constraints. Using this information to dispatch jobs, we can then use a graph neural network to parameterize the policy $\pi(a_t \mid s_t)$, allowing the priority dispatching rule to be learned.

4.2.1. Graph Embedding

A graph embedding neural network is a kind of deep learning method that can learn from graph structures. Graph neural networks have been widely used in knowledge graphs [17] and recommendation systems [18]. In this paper, we use a Graph Isomorphism Network (GIN) [19]. Considering a graph $g = (V, E)$, After k iterations, the representation of the nodes implicitly contains information about the structure of the graph. The k-th level of the GIN is calculated as follows:

$$h_v^{(k)} = MLP_{\theta_k}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right), \tag{8}$$

where $h_v^{(k)}$ is the embedding presentation of node v after k iterations, $MLP_{\theta_k}^{(k)}$ is Multiple-Layer Perceptron followed by batch normalization [20], its parameter is $\theta_k^{(k)}$ in k iterations, ϵ is an arbitrary number, and $N(v)$ means the neighbourhoods of node v. After K iterations, we use average pooling to obtain a global representation h_g for this graph g $h_g = 1/|V| \sum_{v \in V} h_v^K$.

There is a problem in that the GIN is proposed to solve undirected graph problems. However, because our application scenario is communication order dispatching, we should extend this to handle the directed graph. The traditional method is to replace one undirected arc with two opposite directed graphs constructed as a fully directed graph $G_D(t)$ [13]. However, this makes the graph too dense to be efficiently processed via GIN. To reduce redundant calculations, we propose adding an arc strategy during the dispatching process. To be more specific, we use $\overline{G}_d(t) = (O, C \cup D_u(t))$ as an approximation of $G_D(t)$. As the process runs and more arcs are added in, the size of the directed arcs $D_u(t)$ becomes larger. Figure 2 provides a view of the adding strategy. Obviously, the graph built using the adding strategy is sparser than the traditional method. Finally, we define the raw feature for each node O at s_t as $h_O^{(0)}(s_t) = (I(O, s_t), (T_{LB}(O_{ij}, s_t), F_{LB}(O_{ij}, s_t), C_{LB}(O_{ij}, s_t))$ and obtain the node and graph embeddings after K iterations as $h_O^{(K)}(s_t)$ and $h_g^{(s_t)}$, respectively.

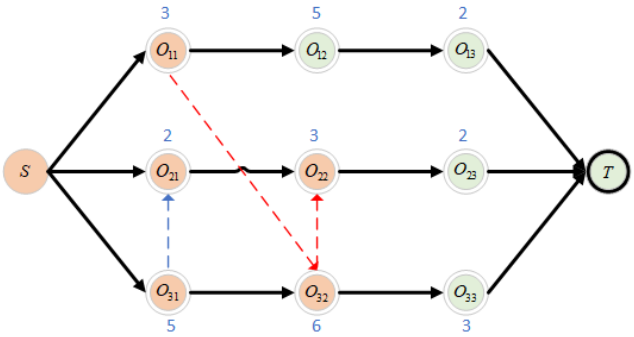


Figure 2. The adding-arc strategy. The directed arc following the current decision is added to the graph, i.e., arcs (O_{11}, O_{32}) and (O_{32}, O_{22}).

4.2.2. Action Selection

In order to select an action, we propose a selection network to process the graph embedding $h_O^{(K)}(st)$. To do this, we need to output a probability distribution and select the action with the maximum probability. Specifically, during the processing of graph embedding, we adopt the MLP to find a scalar score with graph embedding as input $scr(a_t) = MLP_{\theta_\pi}(h_O^{(K)}(st), h_g^{(s_t)})$. Then, we use softmax functions to output the probability distribution and pick a_t with the maximum probability.

4.3. Convex Hull

In our algorithm, we use the weight vector $w = (w_t, w_c, w_f)$ to form a linear combination of all objectives. During the training stage, we set all of them as 1 in order to maintain generality. To build a subjective optimal weight vector, we resort to the multi-objective RL concept of a convex hull [21].

Definition 1. *(Convex hull). For a given MOMDP, its convex hull CH is the subset of policies π^M for which there exists a weight vector $\vec{w}$ for which the linear value function is maximal:*

$$CH(\mathcal{M}) = \left\{ \pi_* \in \Pi^{\mathcal{M}} \mid \exists \vec{w} : \pi_* \in \arg\max_{\pi} \vec{w} \cdot \vec{V}^{\pi} \right\}. \tag{9}$$

Optimized through a convex hull function, $\vec{w}$ can create a subjective environment for our agent to learn a policy. To be specific, we need find a weight vector $\vec{w}$ which can guarantee that our policy is optimal in the partial convex hull P. Formally, we need find a value such as

$$\vec{w} \cdot V^{\pi_*}(s) > \max_{\pi \in P \setminus \Pi_*} \vec{w} \cdot V^{\pi}(s), \tag{10}$$

where s means every state and π_* is the optimal policy extracted from convex hull. Taking two MOMDP objectives as example, where each point represents the multiple value for

two objectives, the convex hull is shown as a set of filled circles connected by lines that form a convex surface. According to the slope of the convex surface, we can calculate the ratio of the two weights using Equation (10) and obtain the weight vector. During training of the policy, we record the optimal policy in each batch according the weighted value with all weights as 1 to form a convex hull.

4.4. Algorithm

In this paper, we train the policy network using Proximal Policy Optimization (PPO) [22], which uses two neural networks, an actor network and a critic network. Both of them share the same GIN described above. The actor network has the policy π_θ and the critic network uses MLP_{θ_v}, which takes $h_g^{(s_t)}$ as input to find the estimated cumulative rewards at s_t. Algorithm 1 shows the complete process.

Algorithm 1 MOMDP Building

1: Graph embedding(MOMDP M);
2: Compute $P \in CH(M)$ the partical convex hull for weight vector $\vec{w} = (1, 1, 1)$;
3: Find Π^* the set of optimal policies within P by solving

$$\Pi^* = \arg\max_{\pi \in P} (V_N^\pi(s) + V_E^\pi(s)) \text{ for every state } s$$

4: Given the $w_t = 1$, get values for w_c, w_f by solving the Equation (10)
5: Return MOMDP with new weight vector

The components of the algorithm and the flow can be seen in the following Figure 3.

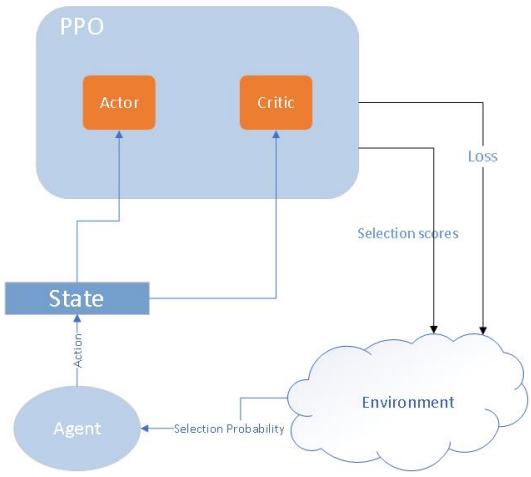


Figure 3. Illustration of the algorithm showing the structure.

5. Experiments

We evaluated and compared the training and test performance of our multiple-objective PDR and single-objective PDR. Both of the experiments were performed on generated instances provided by Zhang [12]. According to the data from China Telecom Guangdong Company, work orders can be classified into four categories: consulting, complaints, business processing, and repair. When scheduling business processing and repair orders, the addresses are always different and distance is an important objective which greatly affects operations. For the same reason, the cost for repair should be taken into account. For application to specific work orders for telecoms, we used maintenance as an example to design three objectives, namely, the estimated time t, cost provided by worker c, and distance to the destination d.

5.1. Models and Configurations

Our model was trained and validated on 100 instances, and we train our policy network for 10,000 iterations with predefined hyperparameters. For the graph embedding we used GIN composed by MLP_{θ_v} which had four hidden layers with a size dimension of 64. For the layers in GIN, we set the number of iterations K to 4. For PPO, the actor network MLP_{θ_v} and critic network MLP_{θ_v} shared the same structure, which had four hidden layers with a dimension of 32. We setthe training epochs of the network as 1, the clip parameter for PPO as 0.2, the critic loss coefficient as 1, the policy loss coefficient as 2, and the entropy loss coefficient as 0.01. During the training process, we set the discount of the reward γ to 1 and used the Adam function with a learning rate of $lr = 2 \times 10^{-5}$ as the optimizer. All other parameters followed the default settings in PyTorch.

5.2. Baselines

There are a number of methods which adopt reinforcement learning on job shop scheduling; however, all of them have only focused on time optimization. In order to prove our method's effect for each objective, we used the algorithm proposed by Cong Zhang [13], which is based on Python. In addition, there have been a number of PDRs proposed for JSSP; thus, we selected two traditional PDRs based on their performance as reported in [23], including Shortest Processing Time (SPT) and Most Work Remaining (MWKR). SPT is one of the most widely used PDRs in research and industry, while MWKR is a newly developed model which has demonstrated excellent performance [23]. We compared our time objective with traditional algorithms as well.

5.3. Results on Generated Instances

Figure 4 illustrates the process of a customer complaint handling system developed by Guangzhou Telecom. Customers call the customer service line to communicate their requirements. These requirements can be divided into four categories: 31% Consulting, 24% Malfunction, 23% Complaint, and 22% Business Process. After generating orders, 21% of orders about business expenses are handled by the intensive process. The remaining work orders are distributed to the ICS (Intensive Customer Service). These order are handled by district branches. Considering the area and maintenance costs, the distance and costs need to be considered to ensure that the single target approach will no longer work for this scenario. To simulate the dispatch data in real applications, we generate a dataset with three objectives: time, cost and distance.

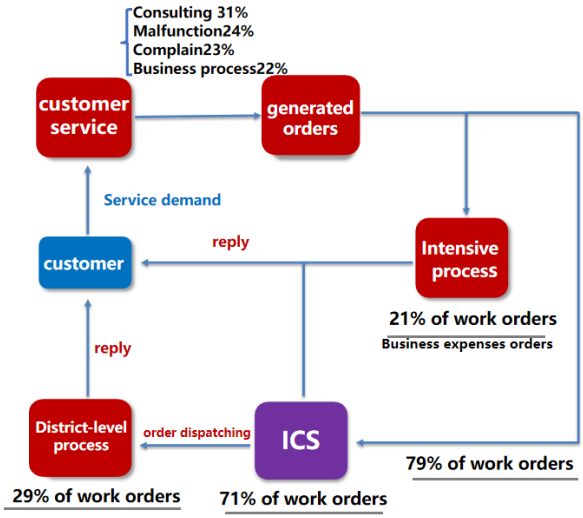


Figure 4. The figure shows the process of generating and distributing work orders.

First, we trained our method on generated data with a size of $15 \times 15, 20 \times 20$. The three objective training curves are shown in Figure 5. We generated 100 instances of each

size (10×10, 15×15, 20×20) randomly for ten times, and report the average of three objectives to suggest the effect of optimization. The learning curves of the three objectives are reported below. All of them show the effective optimization process of our algorithm.

In order to highlight the significance of our multiple-objective algorithm, We performed testing on different sizes of generated datasets with three objectives, as mentioned above. The results of these three objectives are recorded in the table. We compared our dispatching policy with the deep reinforcement policy proposed by Zhang [12] and compared the time objective with a traditional baseline.

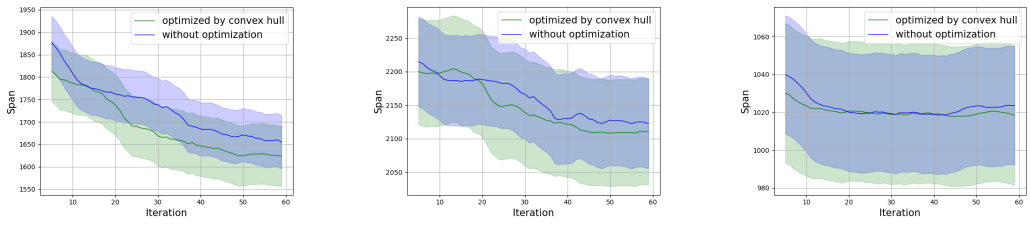
We tested and verified our method on generated instances of sizes (10×10, $15 \times 15, 20 \times 15, 20 \times 20, 30 \times 15, 30 \times 20$). The results in Table 1 show an obvious improvement with our method. As Table 1 shows, our algorithm has better behavior on solving multiple-objective problems, while our MORL algorithm improves performance on the cost objective and distance objective. In particular, for datasets of size $15 \times 15, 20 \times 15, 20 \times 20$ we reached an approximately 20% reduction in the cost objective, while for datasets of size $10 \times 10, 15 \times 15, 30 \times 15, 30 \times 20$ we reached an approximately 20% reduction in the distance objective, which can ensure the nearest service point to take the job. Furthermore, through comparison with two widely used PDRs, namely, SPT and MWKR, we verified the optimization on the time objective.

Table 1. Results on generated instances compared with Zhang [12].

Size	Zhang [12]			Ours		
	Time	Cost	Distance	Time	Cost	Distance
10×10	983.57	441.41	456.09	1138.67	464.38	**396.67**
15×15	1489.91	800.79	789.92	1633.5	**655.72**	**643.99**
20×15	1735.11	909.56	937.34	1971.35	**727.28**	**879.7**
20×20	2005.05	1031.73	1072.31	2369.32	**873.75**	**1009.42**
30×15	2216.51	1199.61	1183.06	2542.12	**1097.57**	**969.33**
30×20	2471.58	1325.24	1311.9	2856.38	**1200.59**	**1053.54**

Above all, even though our method had slightly poorer performance on the time objective compared with single-objective reinforcement learning algorithms, the results show better performance on multiple-objective optimization, which is more suitable for practical applications. In addition, we realized optimization on the time objective.

In order to demonstrate the superiority of our algorithm, we compared a model optimized using the convex hull algorithm with a model that was not optimized using the convex hull algorithm; the results are shown below in Table 2. The convex hull algorithm was used to optimize the model, aiming to ensure that the weight vector used in the linearly scalarized value function was maximized, thereby achieving optimal optimization across all objectives. The training curves for the three objectives are shown in Figure 5a–c. Based on our experimental results, we observed that the training curve significantly outperforms the unoptimized algorithm (see the Table 3). This indicates that our optimization strategies effectively improved the performance of the algorithm, increasing the stability and accuracy of model training. In the future, we intend to further investigate and optimize the details of the algorithm in order to achieve even better results.



(**a**) Time Objectve (**b**) Cost Objective (**c**) Distance Objective

Figure 5. (**a**–**c**) The respective training curves for time, cost, and distance.

Table 2. Results on generated instances compared to the algorithm without convex hull.

Size	Without Convexhull [12]			Ours		
	Time	Cost	Distance	Time	Cost	Distance
10 × 10	1466.24	548.67	390.14	1138.67	464.38	**396.67**
15 × 15	2012.84	811.77	813.63	1633.5	**655.72**	**643.99**
20 × 15	2375.15	909.56	917.10	1971.35	**727.28**	**879.7**
20 × 20	2417.89	893.47	1131.72	2369.32	**873.75**	**1009.42**

Table 3. Comparison with traditional algorithms.

Size	SPT	MWKR	FDD/MWKR	Ours
10 × 10	1210.98	1151.41	1184.5	**1138.67**
15 × 15	1890.91	1812.13	1782.3	**1633.5**
20 × 15	2253.6	2190.7	2015.8	**1971.35**
20 × 20	2519.8	2469.19	2387.2	**2369.32**
30 × 15	2888.4	2728	2601.3	**2542.12**
30 × 20	3208.69	3080.11	2888.1	**2856.38**

5.4. Results on Public Benchmarks

In order to ensure the validity of our experimental results, we carefully selected the DMU dataset and divided it into eight groups based on dataset size. Our team then generated policies for three of these groups, while the remaining five groups were used as the test set. The purpose of this experimental setup was to evaluate the performance of our policies against a baseline in a variety of dataset scenarios. In this way, we were able to determine the effectiveness of our policies in improving the performance of the DMU dataset. The results of our experiments, shown in Table 4, clearly demonstrate that our policies outperformed the baseline in all of the tested scenarios. This indicates that our policies have the potential to significantly improve the performance of similar datasets in the future.

Table 4. Results on DMU benchmark.

Size	Instance	SPT	MWKR	FDD/WKR	Ours
30 × 20	dmu01	6241	**5837**	5948	5917
	dmu02	6487	6610	6035	**5271**
	dmu03	6978	6363	5863	**5553**
	dmu04	5767	6385	5424	**6360**
	dmu05	6910	6472	6444	**4791**
	dmu41	7698	7930	8248	**6887**
	dmu42	7746	7063	7694	**6913**
	dmu43	7269	7708	7601	**7063**
	dmu44	7114	7335	7490	**6489**
	dmu45	**6212**	6844	7101	7085
	Time(s)	0.721	0.731	0.731	1.804
50 × 15	dmu31	8869	8147	7899	**7433**
	dmu32	7814	8004	7316	**7305**
	dmu33	8114	7710	**7262**	7774
	dmu34	7625	7709	7725	**7385**
	dmu35	8626	7617	**7099**	7240
	dmu71	**9594**	9978	10,889	10,828
	dmu72	**9882**	10,135	11,602	10,144
	dmu73	9953	9721	10,212	**9625**
	dmu74	9866	10,086	10,659	**9473**
	dmu75	9411	9953	10,839	**9270**
	Time(s)	2.403	2.323	2.524	4.972
50 × 20	dmu36	9911	8090	**8084**	8433
	dmu37	8917	9685	9433	**8774**
	dmu38	9384	8414	8428	**8385**
	dmu39	9221	9266	8177	**7801**
	dmu40	9406	8261	**7773**	8256
	dmu76	11,677	10,571	11,576	**10,114**
	dmu77	10,401	11,148	11,910	11,564
	dmu78	10,585	**10540**	11,464	11,345
	dmu79	11,115	11,201	11,035	**10,828**
	dmu80	10,711	10,894	11,116	**11,033**
	Time(s)	2.521	2.694	2.723	5.084

6. Conclusions and Future Work

In this paper, we have proposed a multiple-objective deep reinforcement learning algorithm to solve the work order dispatching problem. First, we suggest an MDP and use a disjunctive graph to represent its state based on the particularity of dispatching. Then, we use the weight vector to compute the reward for multiple objectives. In order determine the weight vector, we use the convex hull approach to search for the optimal vector. Then, we propose a policy network based on PPO and GIN, which ensures that our model contains all of the information of the problem. Based on a comparison of our approach with single-objective DRL approaches, our results suggest that the proposed method has superior behavior on larger instances. In the future, we intend to focus on optimization of the time to inference and extend it in order to support more complex environments.

Author Contributions: Conceptualization, Y.T. and L.A.; Methodology, S.C.; Software, S.C.; Validation, S.C.; Formal analysis, S.C.; Writing—original draft, S.C.; Writing—review & editing, Y.T. and L.A.; Supervision, Y.T. and L.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the National Natural Science Foundation of China (Grant No. 62072355), the Key Research and Development Program of Shaanxi Province of China (Grant No. 2022KWZ-10), the Natural Science Foundation of Guangdong Province of China (Grant No. 2022A1515011424), and the Science and Technology Planning Project of Guangdong Province of China (Grant No. 2023A0505050126).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Du, W.; Tang, Y.; Leung, S.Y.S.; Tong, L.; Vasilakos, A.V.; Qian, F. Robust Order Scheduling in the Discrete Manufacturing Industry: A Multiobjective Optimization Approach. *IEEE Trans. Ind. Inform.* **2018**, *14*, 253–264. [CrossRef]
2. Haupt, R. A survey of priority rule-based scheduling. *Oper.-Res.-Spektrum* **1989**, *11*, 3–16. [CrossRef]
3. Song, W.; Kang, D.; Zhang, J.; Cao, Z.; Xi, H. A sampling approach for proactive project scheduling under generalized time-dependent workability uncertainty. *J. Artif. Intell. Res.* **2019**, *64*, 385–427. [CrossRef]
4. Kenny, G. An introduction to Moustakas's heuristic method. *Nurse Res.* **2012**, *19*, 6–11. [CrossRef] [PubMed]
5. Zhang, T.; Guo, S.; Tan, T.; Hu, X.; Chen, F. Adjacency constraint for efficient hierarchical reinforcement learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 4152–4166. [CrossRef]
6. Luo, S.; Zhang, L.; Fan, Y. Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning. *Comput. Ind. Eng.* **2021**, *159*, 107489. [CrossRef]
7. Zhou, F.; Lu, C.; Tang, X.; Zhang, F.; Qin, Z.; Ye, J.; Zhu, H. Multi-objective distributional reinforcement learning for large-scale order dispatching. In Proceedings of the IEEE International Conference on Data Mining (ICDM), Auckland, New Zealand, 7–10 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1541–1546.
8. Guo, S.; Yan, Q.; Su, X.; Hu, X.; Chen, F. State-temporal compression in reinforcement learning with the reward-restricted geodesic metric. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 5572–5589. [CrossRef] [PubMed]
9. Meloni, C.; Pacciarelli, D.; Pranzo, M. A rollout metaheuristic for job shop scheduling problems. *Ann. Oper. Res.* **2004**, *131*, 215–235. [CrossRef]
10. Liu, T.K.; Chen, Y.P.; Chou, J.H. Solving distributed and flexible job-shop scheduling problems for a real-world fastener manufacturer. *IEEE Access* **2014**, *2*, 1598–1606.
11. Balo, H.T. Optimization of Job Shop Scheduling Problem. *J. Mech. Civ. Ind. Eng.* **2020**, *1*, 14–20.
12. Zhang, C.; Song, W.; Cao, Z.; Zhang, J.; Tan, P.S.; Chi, X. Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1621–1632.
13. Yang, L.; Liu, Z.; Dou, Y.; Ma, J.; Yu, P.S. Consisrec: Enhancing gnn for social recommendation via consistent neighbor aggregation. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, 11–15 July 2021; pp. 2141–2145.
14. Zhang, T.; Guo, S.; Tan, T.; Hu, X.; Chen, F. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21579–21590.
15. Guo, Z.; Wong, W.; Li, Z.; Ren, P. Modeling and Pareto optimization of multi-objective order scheduling problems in production planning. *Comput. Ind. Eng.* **2013**, *64*, 972–986. [CrossRef]
16. Dauod, H.; Li, D.; Yoon, S.W.; Srihari, K. Multi-objective optimization of the order scheduling problem in mail-order pharmacy automation systems. *Int. J. Adv. Manuf. Technol.* **2018**, *99*, 73–83. [CrossRef]

17. Chen, X.; Jia, S.; Xiang, Y. A review: Knowledge reasoning over knowledge graph. *Expert Syst. Appl.* **2020**, *141*, 112948. [CrossRef]
18. Isinkaye, F.O.; Folajimi, Y.O.; Ojokoh, B.A. Recommendation systems: Principles, methods and evaluation. *Egypt. Inform. J.* **2015**, *16*, 261–273. [CrossRef]
19. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How powerful are graph neural networks? *arXiv* **2018**, arXiv:1810.00826.
20. Santurkar, S.; Tsipras, D.; Ilyas, A.; Madry, A. How does batch normalization help optimization? *arXiv* **2018**, arXiv:1805.11604
21. Roijers, D.M.; Vamplew, P.; Whiteson, S.; Dazeley, R. A survey of multi-objective sequential decision-making. *J. Artif. Intell. Res.* **2013**, *48*, 67–113. [CrossRef]
22. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
23. Sels, V.; Gheysen, N.; Vanhoucke, M. A comparison of priority rules for the job shop scheduling problem under different flow time-and tardiness-related objective functions. *Int. J. Prod. Res.* **2012**, *50*, 4255–4270. [CrossRef]