processes

Article

Reinforcement Learning-Based Multi-Objective of Two-Stage Blocking Hybrid Flow Shop Scheduling Problem

Ke Xu [1,2], Caixia Ye [2,3], Hua Gong [1,2,*] and Wenjuan Sun [1,2]

[1] School of Science, Shenyang Ligong University, Shenyang 110159, China; xuke@sylu.edu.cn (K.X.); sunwenjuan@sylu.edu.cn (W.S.)
[2] Liaoning Key Laboratory of Intelligent Optimization and Control for Ordnance Industry, Shenyang 110159, China; yecaixia1102@163.com
[3] School of Automation and Electrical Engineering, Shenyang Ligong University, Shenyang 110159, China
* Correspondence: gonghua@sylu.edu.cn

Abstract: Consideration of upstream congestion caused by busy downstream machinery, as well as transportation time between different production stages, is critical for improving production efficiency and reducing energy consumption in process industries. A two-stage hybrid flow shop scheduling problem is studied with the objective of the makespan and the total energy consumption while taking into consideration blocking and transportation restrictions. An adaptive objective selection-based Q-learning algorithm is designed to solve the problem. Nine state characteristics are extracted from real-time information about jobs, machines, and waiting processing queues. As scheduling actions, eight heuristic rules are used, including SPT, FCFS, Johnson, and others. To address the multi-objective optimization problem, an adaptive objective selection strategy based on t-tests is designed for making action decisions. This strategy can determine the optimization objective based on the confidence of the objective function under the current job and machine state, achieving coordinated optimization for multiple objectives. The experimental results indicate that the proposed algorithm, in comparison to Q-learning and the non-dominated sorting genetic algorithm, has shown an average improvement of 4.19% and 22.7% in the makespan, as well as 5.03% and 9.8% in the total energy consumption, respectively. The generated scheduling solutions provide theoretical guidance for production scheduling in process industries such as steel manufacturing. This contributes to helping enterprises reduce blocking and transportation energy consumption between upstream and downstream.

Keywords: blocking; hybrid flow shop scheduling problem; transportation time; multi-objective reinforcement learning; adaptive objective selection

check for updates

Citation: Xu, K.; Ye, C.; Gong, H.; Sun, W. Reinforcement Learning-Based Multi-Objective of Two-Stage Blocking Hybrid Flow Shop Scheduling Problem. *Processes* **2024**, *12*, 51. https://doi.org/10.3390/pr12010051

Academic Editors: Raul D. S. G. Campilho, Jie Zhang and Olympia Roeva

Received: 19 October 2023
Revised: 15 December 2023
Accepted: 22 December 2023
Published: 25 December 2023

1. Introduction

The hybrid flow shop scheduling problem (HFSP), which combines the features of traditional flow shop scheduling and parallel machine scheduling, is widely employed in the auto industry, food processing, steel forging [1], and other industries. The HFSP buffer is always intended to be infinite; however, owing to product processes and technological restrictions, the buffer is sometimes non-existent or confined. As a result, when all the machines in the following stage are in the processing state, the jobs processed in the previous stage will be blocked on the present machine until an idle machine in the next stage becomes available [2]. This is referred to as the blocking hybrid flow shop scheduling problem (BHFSP). Blocking increases the waiting time for jobs, resulting in a longer makespan and an increase in energy consumption, both of which have an influence on production efficiency. With increased worldwide environmental consciousness and the implementation of China's carbon peak and carbon neutrality goals, energy consumption is increasingly being emphasized as a critical green production metric for enterprises. At the same time, the transportation time of materials between different stages in the process industry cannot be ignored. As a result, in the production of hybrid flow shops, considering the impact of blocking, coordinating production, transportation times, and energy consumption, drawing up production plans can efficiently utilize resources, reduce production costs, and enhance enterprise competitiveness.

We take the heating–rolling stage in a steel enterprise as an example. When there is a need for processing, the slab is first heated in the heating furnace and then transported by trolley to the rolling stage. Steel rolling includes hot rolling, tube rolling, structural steel rolling, and wire rolling, etc. Semi-finished slabs will be blocked in the heating furnace for insulation when the slab is heated in the heating furnace and all the machines in the rolling stage are in processing mode to prevent material deterioration. The job blocking on machines can lead to energy waste and delay the delivery time. Therefore, a multi-objective scheduling problem for a hybrid flow shop considering transportation time was refined by jointly considering economic and green indicators. Figure 1 depicts the process flowchart for the heating–rolling stage.

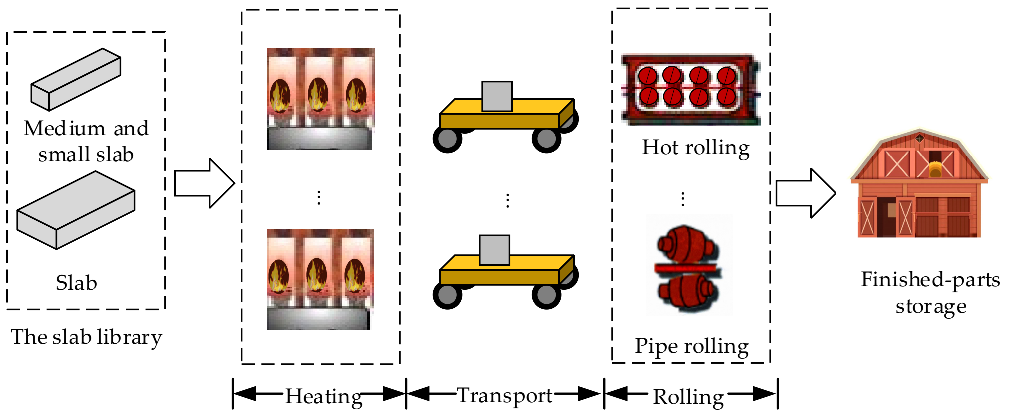

Figure 1. Process flowchart for the heating–rolling stage.

From the past to the present, the majority of scholars have focused on researching the blocking flow shop scheduling problem (BFSP). Du et al. [3] investigated a distributed BFSP with an assembly machine and optimized it for total assembly completion time. They proposed an effective discrete monarch butterfly optimization algorithm. Miyata et al. [4] aimed to minimize the total completion time subject to total maintenance costs in BFSP and introduced a mixed-integer linear programming method to solve the problem. Cheng et al. [5] aimed to minimize the total completion time and proposed an effective metaheuristic algorithm to solve BFSP with sequence-dependent setup times. Zhao et al. [6] studied the distributed assembly BFSP with the total tardiness criterion and employed a mixed-integer linear programming approach for problem modeling. They introduced a constructive heuristic algorithm and a water wave optimization algorithm based on problem-specific knowledge. Niu et al. [7] addressed the distributed group BFSP with carryover sequence-dependent setup time constraints. They proposed a two-stage cooperative coevolutionary algorithm aiming to minimize the makespan and total energy consumption. Zhao et al. [8] investigated the distributed BFSP with sequence-dependence, taking into account makespan, total tardiness, and total energy consumption. They introduced a cooperative whale optimization algorithm for solving this problem. Bao et al. [9] focused on the sequence-dependent BFSP with energy-aware considerations and constructed a mixed-integer linear programming model to minimize makespan and total energy consumption. They proposed a cooperative iterated greedy algorithm based on Q-learning. Nagano et al. [10] addressed the permutation flow shop problem with process blocking and setup times and presented an improved branch-and-bound algorithm with the objective of minimizing total flow time and tardiness. However, traditional flow shop scheduling lacks flexibility, and production lines are often singular. In contrast, the BHFSP allows for one or more parallel machines at each operation, providing adaptability to various production tasks. This not only reduces costs for enterprises but also enhances production efficiency.

Many researchers have conducted studies on HFSP with blocking constraints in recent years. Wang et al. [11] proposed a hybrid decode-assisted mutation iterative greedy algorithm for BHFSP with the objective of minimizing the makespan. Qin et al. [12] proposed a mathematical model of BHFSP based on energy-saving criteria and an improved iterative greedy algorithm based on an exchange strategy to minimize total energy consumption. Shao et al. [13] studied the distributed heterogeneous BHFSP, where the objective function is to minimize the makespan, and proposed a learning-based selection hyper-heuristic framework. Missaoui et al. [14] studied BHFSP where the objective function is to minimize the sum of weighted earliness and tardiness and proposed an efficient iterated greedy approach. Aqil et al. [15] studied BHFSP under the constraint of sequence-dependent setup time where the objective function is to minimize the total tardiness and earliness and proposed six algorithms based on the migratory bird optimization and water wave optimization. Qin et al. [16] established a mathematical model of BHFSP, where the objective is to minimize the makespan, and designed an iterative greedy algorithm with a double-level mutation strategy. Zhao et al. [17] proposed a cooperative monarch butterfly optimization algorithm to solve the distributed assembly blocking flow shop scheduling problem, where the optimization objective is to minimize the assembly completion time. Wang et al. [18] investigated the BHFSP on batch processing machines. Their objective was to minimize the total energy consumption of machines and the makespan. They designed a hybrid meta-heuristic algorithm based on ant colony optimization and genetic algorithms to solve this problem. It can be observed that most research on BHFSP primarily focuses on single-objective optimization, where the main optimization objectives are makespan, tardiness, or energy consumption. In light of the increasingly severe environmental challenges, the consideration of coordinated optimization among multiple objectives, such as completion time and energy consumption, is not only crucial for enhancing economic benefits for enterprises but also contributes to achieving sustainable development goals and alleviating environmental burdens.

In the research on multi-objective HFSP, Feng et al. [19] studied HFSP based on the parallel sequential movement mode, where the optimization objective is to maximize both the makespan and handling events, and proposed an improved non-dominated sorting genetic algorithm (NSGA-II) to find Pareto solutions. Lei et al. [20] focused on the optimization objectives of minimizing the makespan and maximizing the tardiness and designed an optimization algorithm based on multi-class teaching to solve the distributed HFSP with sequence-dependent setup times. Geng et al. [21] aimed to minimize the makespan and maximize the average agreement index and designed a hybrid NSGA-II algorithm to solve the fuzzy re-entrant HFSP. Wu et al. [22] studied the re-entrant HFSP with continuous batch processing machines and proposed an improved multi-objective evolutionary algorithm based on decomposition to reduce the production cycle and energy consumption in the production of cold-drawn seamless steel pipes. Wang et al. [23] aimed to minimize the makespan, the total energy consumption, and the processing cost of the machine and proposed an improved decomposition-based multi-objective evolutionary algorithm to solve the HFSP. Song et al. [24] aimed to minimize both the energy consumption and the makespan and proposed an improved fast NSGA-II to solve the HFSP. Lei et al. [25] solved the distributed two-stage HFSP considering sequence-dependent setup times and proposed an improved shuffled frog leaping algorithm to minimize the number of tardy jobs and the makespan simultaneously. Song et al. [26] aimed to minimize completion time and energy consumption and proposed a hybrid multi-objective teaching–learning-based optimization algorithm based on decomposition to solve the HFSP with an unrelated parallel machine. Li et al. [27] investigated energy-efficient HFSP with uniform machines and formulated a new multi-objective mixed-integer nonlinear programming model to minimize total tardiness, total energy cost, and carbon trading cost. They introduced the NSGA-II based on Q-learning and general variable neighborhood search. Wang et al. [28] explored HFSP with dynamic reconfiguration processes and the dual objectives of minimizing the makespan and the whole device's energy consumption. They obtained a Pareto-based optimal solution set using an improved multi-objective whale optimization algorithm. Cui et al. [29] studied a multi-objective HFSP with unrelated parallel machines, considering minimum makespan and total tardiness, and designed an enhanced multi-population genetic algorithm for solution optimization. In summary, it is essential to consider the impact of transportation time on scheduling results in the context of multi-objective HFSP.

Traditional HFSP solving methods often employ intelligent optimization algorithms and heuristic algorithms. For complex shop scheduling problems that are difficult to solve, reinforcement learning can learn the optimal strategy through interaction with the environment, and its application in the field of scheduling is becoming increasingly widespread. Currently, reinforcement learning has been researched in various settings, including single machines [30], parallel machines [31], flow shops [32,33], job shops [34,35], and flexible job shops [36]. Particularly in the context of reinforcement learning for solving multi-objective problems, Zhang et al. [37] conducted research on the distributed HFSP with a certain degree of symmetry. The objective is to minimize both the makespan and the number of tardy jobs and propose a dual-population genetic algorithm based on Q-learning. Cheng et al. [38] designed a multi-objective Q-learning hyper-heuristic algorithm based on Bi-criteria selection, where the objective is to optimize both production efficiency and energy consumption simultaneously. Chang et al. [39] studied the multi-objective dynamic flexible job shop scheduling problem (MODFJSP) and proposed a hierarchical reinforcement learning approach to solve the MODFJSP considering the arrival of random jobs. Li et al. [40] conducted research on the multi-objective flexible job shop scheduling problem with fuzzy processing times, where the optimization objectives are makespan and total machine workload, and proposed a reinforcement learning-based multi-objective optimization algorithm. Yuan et al. [41] studied the multi-objective optimization scheduling problem in heterogeneous cloud environments and proposed a multi-objective reinforcement learning job scheduling method with AHP-based weighting. Wu et al. [42] studied the green dynamic multi-objective scheduling problem in a re-entrant hybrid flow shop and proposed an improved Q-learning algorithm. To sum up, when dealing with multi-objective problems, reinforcement learning algorithms typically employ a weighted summation of multiple objectives to transform them into a single objective. Objective weights are typically determined according to expert experience or experimentation. However, fixed weights are challenging to adapt in real time to changes in the state of the problem, thereby affecting the quality of the solutions.

As mentioned above, previous research in BHFSP has predominantly focused on single-objective optimization, with limited consideration for the coordinated transportation between upstream and downstream. Since the machines are at different geographical locations, transportation times have an impact on scheduling systems. Moreover, optimizing a single objective has inherent limitations when dealing with complex and diverse problems. Therefore, this paper investigates a multi-objective scheduling problem in a two-stage blocking hybrid flow shop with transportation constraints. In multi-objective optimization, determining objective weights often relies on expert experience or experiments. However, fixed weights are challenging to adapt in real time to changes in problem states, affecting the quality of solutions. This paper introduces a Q-learning algorithm based on adaptive objective selection. The algorithm better adapts to dynamic problem changes, enhancing solution flexibility and robustness. The detailed contributions of this paper are as follows:

(1) For the problem of modern industrial process manufacturing, due to production process requirements, downstream machine congestion can result in upstream blocking, and the transportation time between upstream and downstream cannot be ignored. This paper formulates the HFSP with both transportation and blocking constraints. With the optimization objectives of minimizing the makespan and the total energy consumption, a two-stage BHFSP model incorporating transportation is established.

(2) We have designed an improved multi-objective Q-learning algorithm to address this model. Additionally, an adaptive object selection strategy based on t-tests has been developed for handling multi-objective optimization problems. This strategy coordinates the selection of different objectives by evaluating the confidence of the objective functions under the current job and machine state, thus optimizing both completion time and energy consumption indicators effectively.

The rest of this paper Is organized as follows: Section 2 establishes the mathematical model of the two-stage BHFSP with transportation times. Section 3 describes the implementation details of the Q-learning algorithm based on adaptive object selection. In Section 4, numerical experiments are conducted to demonstrate the effectiveness of the proposed algorithm. Finally, in Section 5, conclusions are drawn, and future research directions are proposed.

2. Problem Formulation

The two-stage BHFSP with transportation times can be described as follows: There are n jobs that need to go through s (s = 1, 2) processing stages, each of which has multiple identical parallel machines, and each machine is located at a different geographical location. The processing sequence for all jobs is the same, and each job can be processed on any machine at each stage. The jobs processed in the first stage are transported to the production machines in the next stage by the transport vehicles. There is no buffer between stages, meaning that once a job completes its processing in the previous stage, it can only leave the machine when the next stage has available machines. The waiting time of the job is referred to as the blocking time. The objective function is to minimize both the makespan and the total energy consumption.

We assume that:

(1) All jobs have arrived at time zero and can begin processing.
(2) There is no limit to the number of transport vehicles that can be used after the job leaves the first-stage machine.
(3) Once the job begins processing or transporting, it cannot be interrupted.

The parameters and decision variables are defined as follows:

$\boldsymbol{J}$: set of jobs, $\boldsymbol{J} = \{1, 2, \ldots, n\}$;
$\boldsymbol{M}_s$: set of machines, $\boldsymbol{M}_s = \{1, 2, \ldots, m_s\}$;
j: index of a job, $j = 1, 2, \ldots, n$;
i: index of the first-stage machine, $i = 1, 2, \ldots, m_1$;
k: index of the second-stage machine, $k = 1, 2, \ldots, m_2$;
p_{sj}: the processing time of job j at stage s;
t_{ik}: the transportation time of the job from machine i to machine k;
SP_i: the blocking power of a job on machine i in the first stage per unit of time;
TP_{ik}: the transportation power of a job from machine i to machine k per unit of time;
M: the large positive number;
A_j: the arrival time of job j;
B_{sj}: the start time of job j at stage s;
C_{sj}: the completion time of job j at stage s;
L_{1j}: the leave time of job j in the first stage;
π: the feasible overall scheduling solution;
$t_j(\pi)$: the transportation time of job j under the scheduling solution π;
$w_j(\pi)$: the waiting time of job j before processing in the first stage under the scheduling solution π;
$b_j(\pi)$: the blocking time of job j on the first-stage machine under the scheduling solution π;
C_{max}: the makespan of job j;
TEC: the total energy consumption;
X_{ij}: it is equal to 1 if job j is processed on machine i; otherwise, it is equal to 0;
Y_{jk}: it is equal to 1 if job j is processed on machine k; otherwise, it is equal to 0;

1. Makespan: The factors affecting the completion time of the job include processing time, transportation time, waiting processing time, and blocking time. The formula is defined as follows:

$$\min f_1 = C_{\max} = \max(C_1, \ldots, C_j, \ldots C_n) \tag{1}$$

$$C_j = p_{1j} + p_{2j} + t_j(\pi) + w_j(\pi) + b_j(\pi) \tag{2}$$

$$t_j(\pi) = \sum_{i=1}^{m_1} \sum_{k=1}^{m_2} X_{ij} Y_{jk} t_{ik} \tag{3}$$

$$w_j(\pi) = B_{1j} - A_j \tag{4}$$

$$b_j(\pi) = L_{1j} - C_{1j} \tag{5}$$

where Equation (1) represents the objective function to minimize the makespan. Equation (2) defines the completion time of job *j* as the sum of processing time, transportation time, waiting processing time, and blocking time. Equation (3) represents the transportation time of job *j*. Equation (4) defines the waiting processing time of job *j* before the first stage as the difference between its start processing time in the first stage and its arrival time. Equation (5) defines the blocking time of job *j* on the first-stage machine as the difference between its leave time on the first-stage machine and its completion time.

2. Total energy consumption: *TEC* includes blocking energy consumption (EC_1), transportation energy consumption (EC_2), and processing energy consumption (EC_3). Notably, EC_3 for each job is solely dependent on its processing time. Since each stage is equipped with identical parallel machines, EC_3 is not affected by different processing sequences and remains constant. Therefore, Equation (6) shows that minimizing TEC requires minimizing EC_1 and EC_2. The second objective function is as follows:

$$\min f_2 = TEC = EC_1 + EC_2 \tag{6}$$

$$EC_1 = \sum_{j=1}^{n} \sum_{i=1}^{m_1} (X_{ij} L_{1j} - X_{ij} C_{1j}) SP_i \tag{7}$$

$$EC_2 = \sum_{j=1}^{n} \sum_{i=1}^{m_1} \sum_{k=1}^{m_2} X_{ij} Y_{jk} t_{ik} TP_{ik} \tag{8}$$

where Equation (7) defines EC_1 as the energy consumed when a job is blocked on a machine. It is equal to the product sum of the blocking time of the job and the corresponding blocking power of the machine. Equation (8) defines EC_2 as the energy consumed when a vehicle transports a job. It is equal to the product of the transportation time of the job and the transportation power.

The following mathematical model is established based on the above problems:

$$\min\{C_{\max}, TEC\} \tag{9}$$

s.t

$$\sum_{i=1}^{m_1} X_{ij} = 1, \sum_{k=1}^{m_2} Y_{jk} = 1 \ \forall j \in \boldsymbol{J} \tag{10}$$

$$C_{1j} = B_{1j} + \sum_{i=1}^{m_1} p_{1j} X_{ij} \ \forall i \in \boldsymbol{M}_1, j \in \boldsymbol{J} \tag{11}$$

$$C_j = B_{2j} + \sum_{k=1}^{m_2} p_{2j} Y_{jk} \ \forall k \in \boldsymbol{M}_2,\ j \in \boldsymbol{J} \tag{12}$$

$$L_{1j} + t_j(\pi) = B_{2j} \ \forall j \in \boldsymbol{J} \tag{13}$$

$$B_{1j'} \geq L_{1j} - M(2 - X_{ij} - X_{ij'}) \ \forall j, j' \in \boldsymbol{J} \tag{14}$$

$$C_{1j} \leq L_{1j} \ \forall j \in \boldsymbol{J} \tag{15}$$

$$B_{sj} \geq 0 \ \forall j \in \boldsymbol{J} \tag{16}$$

$$X_{ij} = \begin{cases} 1, \text{if job } j \text{ is processed on machine } i \\ 0, \text{otherwise} \end{cases} \quad \forall i \in \boldsymbol{M}_1, j \in \boldsymbol{J} \tag{17}$$

$$Y_{jk} = \begin{cases} 1, \text{if job } j \text{ is processed on machine } k, \\ 0, \text{otherwise} \end{cases} \quad \forall k \in \boldsymbol{M}_2, j \in \boldsymbol{J} \tag{18}$$

where Equation (9) is the objective function and Equations (10)–(16) are constraints. Equation (10) represents a job that can only be processed by one machine at each stage. Equations (11) and (12) define the completion time of a job as the sum of its start processing time and its processing time. Equations (13) and (14) represent blocking constraints. Equation (13) defines the start processing time of a job in the second stage as the sum of its transportation time and its leave time in the first stage. Equation (14) represents when the start time of the latter processing job j' on the same machine cannot be shorter than the leave time of the former processed job j. Equation (15) represents a job that can only leave after the operation is finished. Equation (16) represents when the start processing time of a job must be greater than or equal to 0. Equations (17) and (18) represent constraints on decision variables.

3. Adaptive Objective Selection Q-Learning Algorithm

The two-stage BHFSP model with transportation time established in Section 2 is formulated as a multi-objective mixed-integer programming model. HFSP has been proven to be an NP-hard problem [43], and due to the complexity of the problem studied in this paper, it is also NP-hard. Reinforcement learning enables autonomous learning through interaction between agents and the environment. It can adapt to diverse tasks and environments while achieving continuous improvement, giving it an advantage in intelligent decision-making and scheduling. In this section, an adaptive objective selection Q-learning algorithm (AQL) for solving multi-objective scheduling problems is designed. The confidence of the two objective functions is computed using a t-test, allowing for a focus on optimizing the object with the highest confidence.

3.1. Problem Transformation

3.1.1. State

The state feature mainly shows the environmental features of the blocking hybrid flow shop, including real-time information on machines, jobs, and the waiting processing queues before the two stages. $f_{j,1}$ represents the state of job j; $f_{i,2}$ represents the working state of machine i in the first stage; $f_{k,3}$ represents the working state of machine k in the second stage; and $f_{s,4}$–f_9 represents the environmental state features of the waiting processing queues. Therefore, this paper studies the two-stage BHFSP with transportation time, and there are $n + m_1 + m_2 + 11$ states in the whole environment. The definitions of various state features are shown as follows: Q_1 represents the waiting processing queue before the first stage and Q_2 represents the processing queue blocked on the machines.

State 1 The five states of the job j.

$$f_{j,1} = \begin{cases} 0, \text{wait for the first stage of machine processing} \\ 1, \text{ is in the first stage of machine processing} \\ -1, \text{ is blocking on the machine in the first stage} \\ 1/2, \text{ is in the second stage of machine processing} \\ -1/2, \text{ complete the second stage of processing} \end{cases}, j = 1, 2, \ldots, n \tag{19}$$

State 2 The working state of machine i in the first stage.

$$f_{i,2} = \begin{cases} 0, \text{the machine } i \text{ is idle} \\ 1, \text{the machine } i \text{ is busy} \end{cases}, i = 1, 2, \ldots, m_1 \tag{20}$$

State 3 The working state of machine k in the second stage.

$$f_{k,3} = \begin{cases} 0, \text{the machine } k \text{ is idle} \\ 1, \text{the machine } k \text{ is busy} \end{cases}, k = 1, 2, \ldots, m_2 \tag{21}$$

State 4 The ratio of the number of all jobs in queue Q_s to the total number of jobs.

$$f_{s,4} = \frac{\eta(Q_s)}{n}, s = 1, 2 \tag{22}$$

State 5 The ratio of the average processing time of all jobs in queue Q_s to the average processing time of the job on the machine at this stage.

$$f_{s,5} = \frac{\sum\limits_{j \in Q_s} p_{sj}}{\eta(Q_s)} \cdot \frac{n}{\sum\limits_{j=1}^{n} p_{sj}}, Q_1 \neq 0; s = 1, 2 \tag{23}$$

State 6 Whether the job with minimum processing time is in queue Q_s.

$$f_{s,6} = \begin{cases} 0, \text{the job with the minimum processing} \\ \quad \text{time is not in queue } Q_s \\ 1, \text{otherwise} \end{cases}, s = 1, 2 \tag{24}$$

State 7 The ratio of the maximum processing time of a job in queue Q_s to the maximum processing time of all jobs.

$$f_{s,7} = \frac{\max\limits_{j \in Q_s}(p_{sj})}{\max\limits_{j \in N}(p_{sj})}, s = 1, 2 \tag{25}$$

State 8 The ratio of the minimum processing time of a job in queue Q_s to the maximum processing time of all jobs.

$$f_{s,8} = \frac{\min\limits_{j \in Q_s}(p_{sj})}{\max\limits_{j \in N}(p_{sj})}, s = 1, 2 \tag{26}$$

State 9 The ratio of the number of jobs in queue Q_1, whose processing time in the first stage exceeds that in the second stage, to the number of jobs in queue Q_1.

$$f_9 = \frac{\eta(JQ_1)}{\eta(Q_1)}, JQ_1 = \{J_j | p_{1j} > p_{2j}, J_j \in Q_1\}; Q_1 \neq 0 \tag{27}$$

3.1.2. Action

The actions are designed based on scheduling rules such as SPT, FCFS, Johnson, etc. These scheduling rules are primarily adopted to allocate waiting jobs to machines. When a machine is idle, a job can select it for processing; when a machine is busy, a job cannot choose that machine. If there are no available machines at a certain moment, the job can only be blocked at the current stage. Based on this, six actions are set in the first stage and four actions are set in the second stage, for a total of twenty-four joint actions.

- The First Production Stage

Action 1 SPT: Process the jobs in queue Q_1 in p_{1j} ascending order, selecting the job with the shortest processing time.

Action 2 LPT: Process the jobs in queue Q_1 in p_{1j} descending order, selecting the job with the longest processing time.

Action 3 SPT + SSO: Process the jobs in queue Q_1 in p_{1j} + p_{2j} ascending order, selecting the job with the shortest total processing time.

Action 4 LPT + LSO: Process the jobs in queue Q_1 in $p_{1j} + p_{2j}$ descending order, selecting the job with the longest total processing time.

Action 5 Johnson–Bellman: Divide the set of jobs in queue Q_1 into two subsets, SJ_1 and SJ_2. SJ_1 contains the set of jobs where $p_{1j} < p_{2j}$, and SJ_2 contains the remaining jobs. Then, apply the SPT rule to select jobs from SJ_1 and the LPT rule to select jobs from SJ_2.

Action 6 Select no job: Select this action when there are no jobs in queue Q_1 or the machines in the first stage are busy.

- The Second Production Stage

Action 7 SPT: Process the jobs in queue Q_2 in p_{2j} ascending order, selecting the job with the shortest processing time.

Action 8 LPT: Process the jobs in queue Q_2 in p_{2j} descending order, selecting the job with the longest processing time.

Action 9 FCFS: Process the jobs in queue Q_2 in an ascending order of completion time, selecting the job that finishes first.

Action 10 Select no job: Select no job: Select this action when there are no jobs in queue Q_2 or the machines in the second stage are busy.

3.1.3. Reward

The reward function represents the immediate feedback received after performing an action in the current state and is usually related to the objective function. Therefore, rewards based on the makespan and the energy consumption are defined as follows: r_t^1 represents reward 1 obtained at decision moment t and r_t^2 represents reward 2 obtained at decision moment t. They are defined as follows:

$$r_t^1 = f_1(t-1) - f_1(t) \tag{28}$$

$$r_t^2 = f_2(t-1) - f_2(t) \tag{29}$$

where $f_1(t)$ represents the makespan of the currently processed job at decision moment t and $f_2(t)$ represents the energy consumption already generated at decision moment t. They are represented as follows:

$$f_1(t) = \max\left\{C_{s_j(t)j} \middle| j \in \boldsymbol{CJ}(t)\right\} \tag{30}$$

$$f_2(t) = \sum_{j\in\boldsymbol{CJ}(t)} \sum_{i\in\boldsymbol{M}_1} (X_{ij}L_{1j} - X_{ij}C_{1j})SP_i + \sum_{j\in\boldsymbol{CJ}(t)} \sum_{i\in\boldsymbol{M}_1} \sum_{k\in\boldsymbol{M}_2} X_{ij}Y_{jk}t_{ik}TP_{ik} \tag{31}$$

where $s_j(t)$ represents the number of operations completed on job j at decision moment t and $\boldsymbol{CJ}(t)$ represents the set of jobs processed at decision moment t.

Based on the rewards at each decision moment, the cumulative rewards obtained are as follows:

$$R_1 = \sum_{t=1}^{T} r_t^1 = \sum_{t=1}^{T} f_1(t-1) - f_1(t) = f_1(0) - f_1(T) = -C_{\max} \tag{32}$$

$$R_2 = \sum_{t=1}^{T} r_t^2 = \sum_{t=1}^{T} f_2(t-1) - f_2(t) = f_2(0) - f_2(T) = -TEC \tag{33}$$

where T represents the last decision moment when all the jobs have been processed and $f_1(T)$ and $f_2(T)$ represent $C_{\max}$ and TEC, respectively. Since no processing operations have been performed at the initial moment, $f_1(0) = f_2(0) = 0$.

3.2. Value Function Approximation

The basic idea of the Q-learning algorithm is to guide the agent to make decisions by learning a Q-value function to maximize long-term cumulative rewards. The Q-value function $Q(s, a)$ represents the expected cumulative reward achievable by taking action a in state s. To simplify the problem and reduce computational complexity, state discretization is employed. In this paper, a parameterized approximation approach is used to update the state-value function by updating the weight of the basis function. The update formula is as follows:

$$Q(s,a) = \sum_{z=1}^{m_1+m_2+n+11} w_z^a \varphi_z(s) \tag{34}$$

where φ_z(s) represents the vector of basis functions in the state space and w_z^a is the weight for selecting action a in the current state s_z. The normalization of the basis functions is shown in Equation (35).

$$\varphi_z(s) = \begin{cases} f_{z,1}, 1 \le z \le n \\ f_{z,2}, n+1 \le z \le n+m_1 \\ f_{z,3}, n+m_1+1 \le z \le n+m_1+m_2 \\ f_{z,4}, n+m_1+m_2+1 \le z \le n+m_1+m_2+2 \\ f_{z,5}, n+m_1+m_2+3 \le z \le n+m_1+m_2+4 \\ f_{z,6}, n+m_1+m_2+5 \le z \le n+m_1+m_2+6 \\ f_{z,7}, n+m_1+m_2+7 \le z \le n+m_1+m_2+8 \\ f_{z,8}, n+m_1+m_2+9 \le z \le n+m_1+m_2+10 \\ f_{z,9}, z = n+m_1+m_2+11 \end{cases} \tag{35}$$

3.3. T-Test-Based Adaptive Objective Selection

Multi-objective reinforcement learning often employs a linear scalarization approach to address multiple objectives, with the primary challenge lying in determining the objective weights. In reinforcement learning, objective weights are typically globally fixed and do not adapt to the dynamically changing problem state space. To address this issue, we integrate objective weights with the problem state, representing the weights as functions of the state. By combining t-tests with confidence, we propose an adaptive objective selection strategy.

The basic idea of adaptive objective selection is the parallel estimation of the Q-function for object o. When action selection is required, a t-test is employed to calculate the confidence of the objective function in the current state, determining the objective where the agent has the highest confidence. As a result, the Q-value of the current object is selected to make action decisions. By using a t-test to calculate confidence, it is possible to demonstrate the significant differences in distributions based on each sample. This allows for a more targeted object selection and weight allocation. The specific steps of the algorithm are as follows:

Step 1: Select x = 10 most recently observed $r^o + \max Q(s', a', o)$ and add them to the sample set $\boldsymbol{SA}_o$.

Step 2: Calculate the confidence levels of each objective function using a t-test. The calculation formula is as follows:

$$t_o = \frac{\overline{x}_o - Q(s, a_v, o)}{\frac{g_o}{\sqrt{x}}}, v \in [1, 24] \tag{36}$$

where $\overline{x}_o$ is the mean of samples in $\boldsymbol{SA}_o$ and g_o is the standard deviation of samples in $\boldsymbol{SA}_o$. Find p_o in the T-bound table with confidence level 1-p_o.

Step 3: Put the confidence level 1-p_o of all objective functions at state s into the set $\boldsymbol{c}_o$, and the expression is as follows:

$$\boldsymbol{c}_o = confidence((s, a_1, o), \ldots, (s, a_{24}, o)) \tag{37}$$

Step 4: Define $\mu_o(s)$ as the weight of the o-th objective function at state s. Select the objective function with the highest confidence level.

$$\mu_o(s) = \begin{cases} 1 & if\ o = \operatorname{argmax} c_o \\ 0 & else \end{cases} \tag{38}$$

Step 5: Select an action based on the objective function with the highest confidence level.

3.4. Algorithm Framework

The visualization in Figure 2 shows the specific implementation process of the algorithm. The scheduling system is in the initial state s_0 at the start of the processing time. At this point, all jobs are in the waiting processing queue Q_1, and all machines are idle. Then, an action is selected based on the *ε-greedy* strategy, which involves selecting a job from queue Q_1 and an idle machine from the first stage for processing until either the set of idle machines or the set of jobs in queue Q_1 becomes empty. Following this, the blocking queue Q_2 and the set of idle machines before the second stage are evaluated. If they exist, the machine and the job are selected based on the actions. The scheduling system reaches the termination state s_T, where all processing queues are empty and all jobs have been handled, resulting in a scheduling solution.

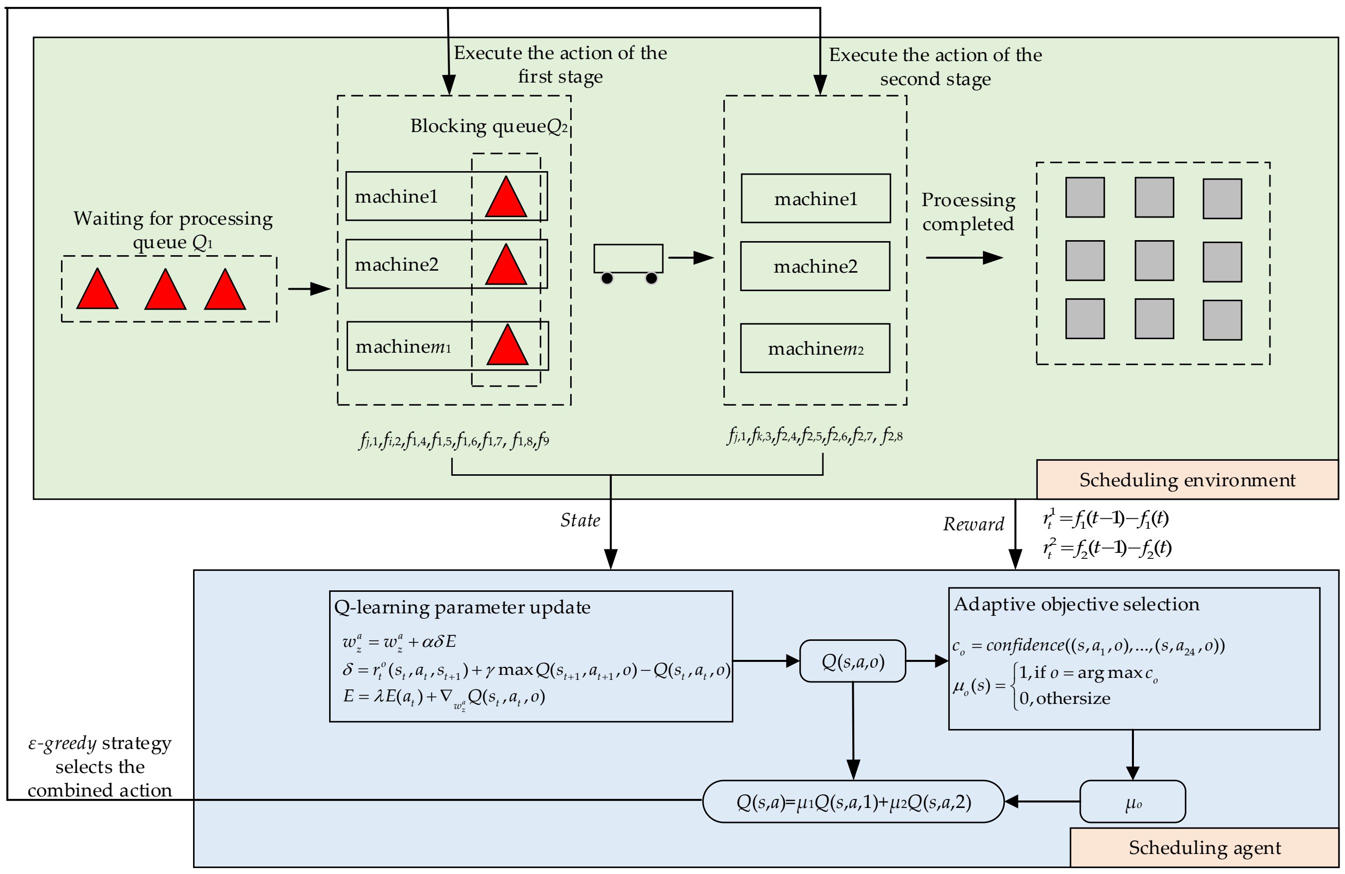
Execute the action of the first stage
Execute the action of the second stage
Blocking queueQ_2
Waiting for processing queue Q_1
machine1
machine2
machinem_1
machine1
machine2
machinem_2
Processing completed
$f_{j,1}, f_{i,2}, f_{1,4}, f_{1,5}, f_{1,6}, f_{1,7}, f_{1,8}, f_9$
$f_{j,1}, f_{k,3}, f_{2,4}, f_{2,5}, f_{2,6}, f_{2,7}, f_{2,8}$
Scheduling environment
State
Reward
$r_t^1 = f_1(t-1) - f_1(t)$
$r_t^2 = f_2(t-1) - f_2(t)$
Q-learning parameter update
$w_z^a = w_z^a + \alpha\delta E$
$\delta = r_t^o(s_t, a_t, s_{t+1}) + \gamma \max Q(s_{t+1}, a_{t+1}, o) - Q(s_t, a_t, o)$
$E = \lambda E(a_t) + \nabla_{w_z^a} Q(s_t, a_t, o)$
$Q(s,a,o)$
Adaptive objective selection
$c_o = confidence((s, a_1, o), \ldots, (s, a_{24}, o))$
$\mu_o(s) = \begin{cases} 1, \text{if } o = \arg\max c_o \\ 0, \text{othersize} \end{cases}$
ε-greedy strategy selects the combined action
$Q(s,a) = \mu_1 Q(s,a,1) + \mu_2 Q(s,a,2)$
μ_o
Scheduling agent

Figure 2. The specific implementation procedure of the algorithm.

The specific steps of the AQL algorithm are as follows:

Step 1: Initialize parameters.

Step 1.1: Input parameters of the scheduling problem: the number of jobs n, the number of machines in the first stage m_1, the number of machines in the second stage m_2, the processing time of each job in the two-stage machines p_{sj}, the transportation time between machines t_{ik}, the blocking power of the machine in the first stage SP_i, and the transportation power of the transporter TP_{ik}.

Step 1.2: Input parameters of the Q-learning algorithm: learning rate α, discount factor γ, greedy factor ε, decay rate λ, and two $m_1 + m_2 + n + 11$ dimensional vectors $E(a) = (0, 0, \ldots, 0)^T$, $w^a = (1, 1, \ldots, 1)^T$; *max_episode*, with the current iteration $g = 1$.

Step 2: Set the initial time t_0 and initial state s_0, and initialize two $Q(s, a)$ tables.

Step 3: Utilize a t-test to calculate the confidence of the objective function in the current state and determine the object o where the agent has the highest confidence.

Step 4: Use the *ε-greedy* strategy, where we obtain a probability of ε to randomly select an action and a probability of $1 - \varepsilon$ to select the action with the highest Q-value from the Q-table.

Step 5: Confirm the state transition time, calculate the reward, and update the Q-table. The reward $r(s_t, a_t, s_{t+1})$ is gained by taking action a_t from state s_t to s_{t+1}, then updating the basis function weights $w_z{}^a$, hence updating the Q-table. The update process is as follows:

$$w_z^a = w_z^a + \alpha\delta E \tag{39}$$

$$\delta = r_t^o(s_t, a_t, s_{t+1}) + \gamma \max Q(s_{t+1}, a_{t+1}, o) - Q(s_t, a_t, o) \tag{40}$$

$$E = \lambda E(a_t) + \nabla_{w_z^a} Q(s_t, a_t, o) \tag{41}$$

Step 6: If the number of jobs that the machine processed in the second stage < n, return to Step 3; otherwise, execute Step 7.

Step 7: If the current iteration number < *max_episode*, g = g + 1, return to Step 2; otherwise, the algorithm is terminated.

4. Numerical Experiments

4.1. Experimental Environment and Parameter Setting

To validate the effectiveness of the Q-learning algorithm, we designed the following instances for simulation analysis. The experiments are carried out on an Intel(R) Core(TM) i5-7200U CPU 2.50 GHz processor, 20 GB RAM, PyCharm2017.3.2 compiler, and python3.7 interpreter software.

The parameter settings are as follows: n = 15, m_1 = 2, m_2 = 3, p_{sj} and t_{ik} are generated at random between [1, 50], and SP_i and TP_{ik} are generated at random between [1, 10].

The initial parameter settings for the AQL algorithm, including α, γ, ε, and λ, are shown in Table 1. These NP values are obtained using an orthogonal experiment conducted according to the $L_9(3^4)$ rule.

Table 1. The initial parameter level table.

	α	γ	ε	λ
K1	0.001	0.1	0.01	0.1
K2	0.1	0.9	0.1	0.5
K3	0.9	0.99	0.2	0.9

Table 2 shows the values of the two objective functions for nine different parameter selections. The performance of the proposed model is evaluated using the Normalized Performance (NP). A smaller NP indicates better performance. The definition of NP is as follows:

$$NP = \frac{C_{\max} - \min \boldsymbol{MC}}{\max \boldsymbol{MC} - \min \boldsymbol{MC}} + \frac{TEC - \min \boldsymbol{MT}}{\max \boldsymbol{MT} - \min \boldsymbol{MT}} \tag{42}$$

Table 2. Orthogonal experimental results.

No.	α	γ	ε	λ	$C_{\max}$	TEC	NP
1	0.001	0.1	0.01	0.1	276	1548	1.84
2	0.001	0.9	0.1	0.5	268	1396	0.56
3	0.001	0.99	0.2	0.9	268	1396	0.56
4	0.1	0.1	0.1	0.9	266	1502	0.87
5	0.1	0.9	0.2	0.1	264	1380	0.18
6	0.1	0.99	0.01	0.5	271	1566	1.54
7	0.9	0.1	0.2	0.5	263	1538	0.80
8	0.9	0.9	0.01	0.9	276	1584	2.00
9	0.9	0.99	0.1	0.1	269	1356	0.46

In Equation (42), ***MC*** represents the set of all C_{max} values and ***MT*** represents the set of all *TEC* values.

Upon summarizing the results from Table 2, for $\alpha = 0.001$, the K1 result is the sum of the total *NP* when the parameter is set to 0.001. The summarized results are shown in Table 3.

Table 3. *NP* values of parameters.

	α	γ	ε	λ
K1	2.96	3.51	5.38	**2.49**
K2	**2.59**	2.74	1.89	2.89
K3	3.26	**2.56**	**1.54**	3.43
optimal	0.1	0.99	0.2	0.1

Note: The bolded values in the table represent the best results.

From Table 3, the minimum values are $\alpha = 2.59$, $\gamma = 2.56$, $\varepsilon = 1.54$, and $\lambda = 2.49$. The final parameter values obtained are $\alpha = 0.1$, $\gamma = 0.99$, $\varepsilon = 0.2$, and $\lambda = 0.1$.

Figure 3 shows the convergence of the objective value when the algorithm iterates up to 1000 generations under the above experimental parameters. It can be seen that the algorithm tends to converge around the 200th iteration. Hence, for this experiment, a maximum of 200 iterations is chosen.

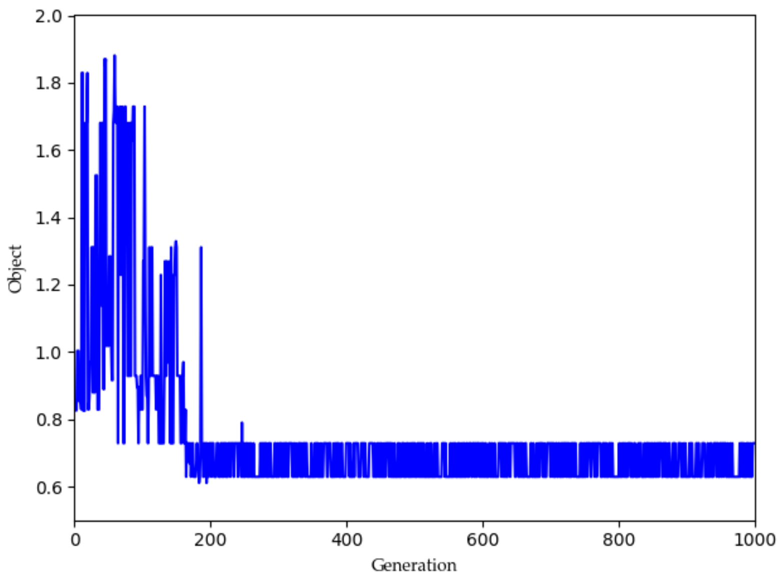

Figure 3. Graph of AQL algorithm convergence.

4.2. Experimental Results and Analysis

To validate the effectiveness of the model and the algorithm, experiments are conducted with the machine set to $m_1 = 2$ and $m_2 = 3$ and the job set to $n = 4$ and $n = 6$. As shown in Table 4, the two objective function values for this problem are obtained using the Gurobi solver and the AQL algorithm. Notably, a computation time limit of 1800 s is set for the Gurobi solver.

As we can see, when $n = 4$, Gurobi finds the optimal solution in less than 1 s. For $n = 7$, it consumes a significantly longer computation time, close to 1195 s. For $n = 8$, the computation time has already reached its limit, and an optimal solution cannot be achieved. The proposed AQL algorithm performs less effectively than the Gurobi solver in solving the first objective value. However, as the problem scale increases, the performance between AQL and Gurobi narrows. More importantly, AQL has a much lower computation time compared to the Gurobi solver. Therefore, as the instance scale increases, AQL can provide optimal solutions in a shorter time compared to the Gurobi solver.

Table 4. The calculation results of objective function values and CPU time for small-scale instances.

Job	Gurobi			AQL		
	$C_{\max}$	*TEC*	*T*/s	$C_{\max}$	*TEC*	*T*/s
$n = 4$	**97**	**354**	0.78	118	**354**	0.57
$n = 5$	**110**	540	5.22	122	**440**	1.74
$n = 6$	**122**	560	60	129	**474**	1.78
$n = 7$	**145**	644	1195	149	**542**	2.60
$n = 8$	—	—	1800	**170**	**696**	3.62

Note: The bolded values in the table represent the best results.

Experiments are conducted with $n = 30$ in three sets of machine: $m_1 = 3$, $m_2 = 5$; $m_1 = 5$, $m_2 = 5$; and $m_1 = 7$, $m_2 = 5$. Table 5 shows the scheduling solutions obtained by the AQL algorithm for these three sets of experimental scales. Figures 4–6 show the corresponding scheduling Gantt charts, where the black areas represent the blocked portions.

Table 5. Scheduling solution and object values.

Machine	Scheduling Solution	$C_{\max}$	*TEC*
$m_1 = 3$, $m_2 = 5$	[26, 10, 19, 17, 12, 30, 3, 22, 4, 8, 29], [25, 23, 5, 27, 21, 20, 7, 13, 9, 18], [2, 15, 16, 24, 11, 6, 28, 14, 1] [26, 27, 21, 20, 7, 22, 9, 18], [25, 15, 24, 30, 3, 4, 8, 29], [2, 17, 6, 28, 1], [23, 5, 19, 12, 13], [10, 16, 11, 14]	325	2218
$m_1 = 5$, $m_2 = 5$	[23, 5, 27, 24, 13, 29], [26, 6, 11, 17, 30, 14, 9], [25, 12, 21, 28, 8], [2, 15, 16, 7, 1], [10, 19, 22, 20, 3, 4, 18] [26, 15, 16, 7, 1, 18], [23, 5, 6, 22, 21, 3, 8], [25, 19, 20, 13], [10, 12, 27, 30, 28, 14, 9], [2, 11, 17, 24, 4, 29]	247	2838
$m_1 = 7$, $m_2 = 5$	[26, 23, 16, 7, 4], [25, 5, 29, 20, 8], [2, 3, 17, 28], [10, 19, 21, 13], [11, 9, 12, 18], [6, 27, 30, 1], [22, 15, 24, 14] [26, 23, 19, 12, 30], [25, 6, 3, 17, 7, 18, 1], [10, 27, 24, 14], [2, 22, 9, 15, 20, 28, 13], [11, 5, 29, 16, 21, 8, 4]	237	3515

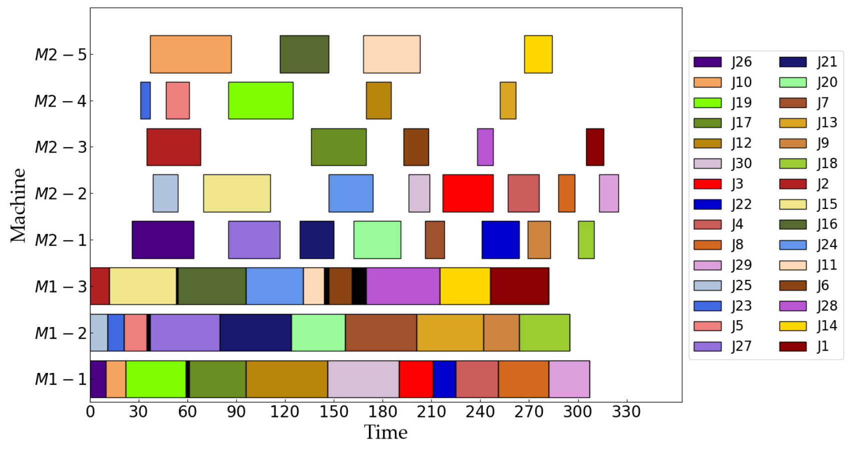

Figure 4. Gantt chart of optimal scheduling for $m_1 = 3$, $m_2 = 5$.

Figures 4–6 show that the performance of the AQL algorithm is influenced by different production configurations. As the number of machines increases in the first stage, the blocking time also increases, leading to higher *TEC*. Therefore, in real-world production environments, it is possible to reduce the risk of job blocking and improve the efficiency and stability of production by designing the layout of the production line.

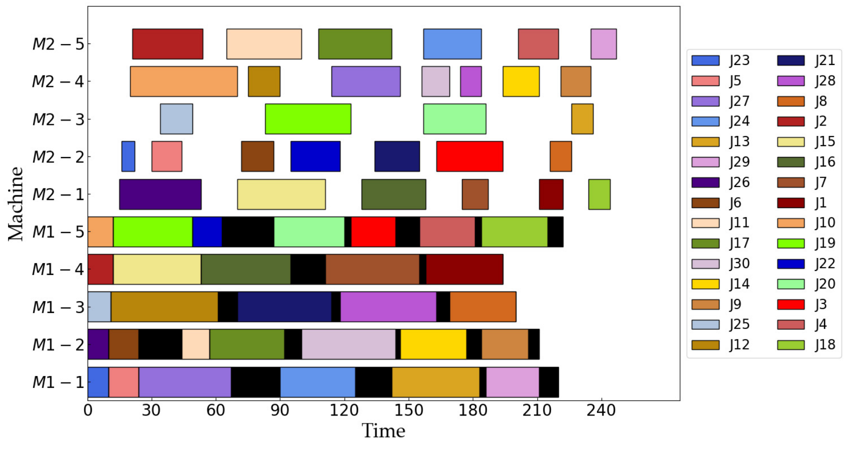

Figure 5. Gantt chart of optimal scheduling for $m_1 = 5$, $m_2 = 5$.

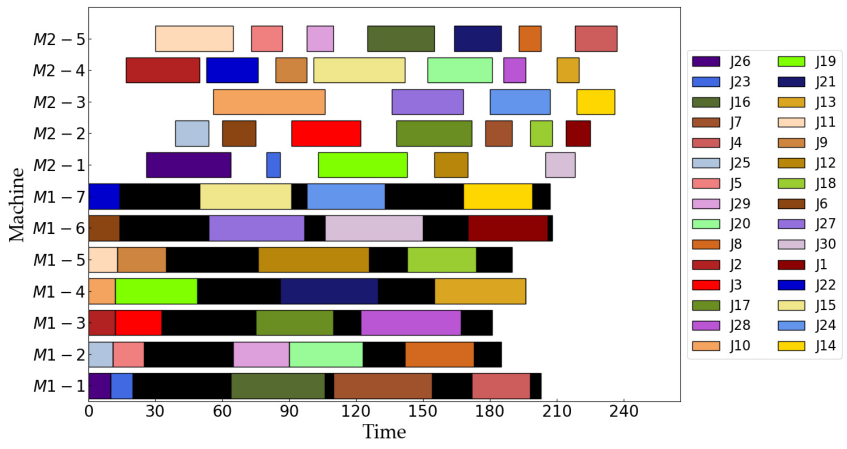

Figure 6. Gantt chart of optimal scheduling for $m_1 = 7$, $m_2 = 5$.

4.3. Experimental Comparison

To further validate the effectiveness of the algorithm, the performance of the AQL algorithm is compared with individual scheduling rules at different experimental scales. Furthermore, comparative analyses are conducted for AQL between Q-learning and NSGA-II, where the Q-learning algorithm linearly weights multiple objective functions as rewards for solving the problem.

Tables 6–8 show the comparative results of the two objective function values obtained by the AQL algorithm and individual scheduling rules at different machine scales, where n is set to 15, 30, 50, and 100, respectively. Figure 7 shows the comparison graph of the frequency of selecting different actions under different machine scales when AQL solves $n = 15$.

From Tables 6–8, it is evident that in 92% of the test instances, AQL consistently achieves lower makespan and TEC. Compared to individual heuristic rules, AQL shows an average improvement in $C_{\max}$ values ranging from a maximum of 21.2% to a minimum of 7.4%. Similarly, for *TEC* values, AQL demonstrates an average improvement ranging from a maximum of 37.4% to a minimum of 13.5%. This indicates that AQL can consistently find scheduling rules that result in better objective values at different scales. From the results with the superscript (*), it is apparent that the worst outcomes are evenly distributed across rules other than the SPT + SSO in the first stage. That is, none of the results obtained under the SPT + SSO rule are the worst. Combined with Figure 7, we can see that AQL selects the SPT + SSO rule significantly more often than other scheduling rules. It shows that AQL can find the scheduling rule that makes the objective value better at each decision point.

Table 6. Comparison of objective functions of different algorithms for $m_1 = 3$, $m_2 = 5$.

No.	Rule	*n* = 15		*n* = 30		*n* = 50		*n* = 100	
		$C_{\max}$	*TEC*	$C_{\max}$	*TEC*	$C_{\max}$	*TEC*	$C_{\max}$	*TEC*
R1	SPT-SPT	231 *	1843	388	3461	615	5551	1236	11,585
R2	SPT-LPT	231 *	1843	374	3584	629	5379	1196	11,402
R3	SPT-FCFS	231 *	1843	388	3461	629	5379	1207	11,789 *
R4	LPT-SPT	213	1791	390 *	3380	656 *	6205	1185	10,028
R5	LPT-LPT	214	1923	376	3289	650	5520	1235	11,638
R6	LPT-FCFS	213	1791	384	3617	651	6212	1250 *	11,557
R7	SPT + SSO-SPT	214	1696	363	2398	586	5962	1181	11,277
R8	SPT + SSO-LPT	214	1696	363	2398	577	5647	1178	10,478
R9	SPT + SSO-FCFS	214	1696	363	2398	586	5962	1178	10,466
R10	LPT + LSO-SPT	196	2097 *	356	2979	553	5611	1165	11,329
R11	LPT + LSO-LPT	196	2097 *	344	3402	574	5780	1141	10,670
R12	LPT + LSO-FCFS	196	2097 *	356	2979	557	5701	1132	11,029
R13	Johnson-SPT	194	1780	364	3653 *	575	5730	1186	10,041
R14	Johnson-LPT	194	1780	364	3653 *	591	6418	1183	11,526
R15	Johnson-FCFS	194	1780	364	3653 *	577	6492 *	1204	11,083
R16	AQL	**159**	**954**	**325**	**2218**	**511**	**4059**	**1098**	**8483**

Note: The bolded values in the table represent the best results, and the values marked with (*) represent the worst results.

Table 7. Comparison of objective functions of different algorithms for $m_1 = 5$, $m_2 = 5$.

No.	Rule	*n* = 15		*n* = 30		*n* = 50		*n* = 100	
		$C_{\max}$	*TEC*	$C_{\max}$	*TEC*	$C_{\max}$	*TEC*	$C_{\max}$	*TEC*
R1	SPT-SPT	222	2248	317	4367	506	8006	1014	16,707
R2	SPT-LPT	222	2248	286	4215	542	8552	1026	17,178
R3	SPT-FCFS	222	2248	317	4367	541	8332	996	15,915
R4	LPT-SPT	220	2837	323	4124	553	8987	1020	17,465
R5	LPT-LPT	216	2884	327 *	4767 *	547	8913	1031	17,606
R6	LPT-FCFS	224 *	3364 *	322	4648	567 *	9078	1042 *	17,655
R7	SPT + SSO-SPT	180	1930	309	3952	514	7996	1025	16,060
R8	SPT + SSO-LPT	180	1930	283	3291	525	8634	970	15,731
R9	SPT + SSO-FCFS	180	1930	309	3952	526	8392	1018	15,741
R10	LPT + LSO-SPT	186	2665	311	4208	513	8758	1018	17,777
R11	LPT + LSO-LPT	186	2665	303	4276	523	8900	1000	16,819
R12	LPT + LSO-FCFS	186	2665	287	3894	520	9389 *	994	17,155
R13	Johnson-SPT	168	2457	283	4043	470	8022	1026	18,701 *
R14	Johnson-LPT	168	2457	291	4496	491	9242	1026	18,555
R15	Johnson-FCFS	168	2457	291	3957	491	9242	1000	17,756
R16	AQL	**151**	**1653**	**247**	**2838**	**440**	**7327**	**936**	**14,899**

Note: The bolded values in the table represent the best results, and the values marked with (*) represent the worst results.

To validate the advantages of the AQL algorithm in solving multi-objective problems, the performance of AQL is compared with NSGA-II and the Q-learning algorithm, respectively. Table 9 shows the comparative results of the objective function values for each algorithm at different scales.

Table 9 presents the experimental results comparing AQL with NSGA-II and Q-learning algorithms. The experimental results indicate that, compared to the Q-learning algorithm, the AQL algorithm achieves lower $C_{\max}$ values in 66.7% of the test instances when n = 15 and n = 30. As the number of jobs increases, both objective function values under the AQL algorithm outperform the Q-learning algorithm. This suggests that linearly weighting multiple objective functions as rewards is subjective. Compared to the NSGA-II, the $C_{\max}$ values, on average, improved by 22.7%, and the *TEC* values increased by an average of 9.8%. In summary, the AQL algorithm significantly outperforms both Q-learning and NSGA in solving multi-objective problems, demonstrating its superiority.

Table 8. Comparison of objective functions of different algorithms for $m_1 = 7$, $m_2 = 5$.

No.	**Rule**	***n* = 15**		***n* = 30**		***n* = 50**		***n* = 100**	
		$C_{\max}$	*TEC*	$C_{\max}$	*TEC*	$C_{\max}$	*TEC*	$C_{\max}$	*TEC*
R1	SPT-SPT	218 *	3119	280	5624	496	10,688	1032	23,859
R2	SPT-LPT	218 *	3119	261	5464	486	10,571	1014	24,350
R3	SPT-FCFS	218 *	3119	280	5624	510	11,360	1005	22,438
R4	LPT-SPT	212	3501	307	5256	559 *	12,694 *	1077 *	25,128
R5	LPT-LPT	212	3458	307	5914	551	11,466	1048	23,694
R6	LPT-FCFS	212	3501	313	5737	551	11,466	1067	23,831
R7	SPT + SSO-SPT	179	2382	258	3945	475	9553	1061	24,597
R8	SPT + SSO-LPT	202	2689	296	5217	499	10,796	1048	25,348
R9	SPT + SSO-FCFS	202	2689	260	4155	457	**8329**	1025	23,356
R10	LPT + LSO-SPT	190	3729	312	6512	521	12,498	1049	26,380 *
R11	LPT + LSO-LPT	190	3729	327 *	7087	510	11,741	1075	25,628
R12	LPT + LSO-FCFS	190	3729	327 *	7601 *	510	11,741	1050	25,382
R13	Johnson-SPT	190	3803	271	5192	481	10,031	1008	25,046
R14	Johnson-LPT	201	3905 *	286	5707	492	11,569	994	22,842
R15	Johnson-FCFS	201	3905 *	271	5192	482	11,593	991	22,391
R16	AQL	**177**	**1937**	**237**	**3515**	**455**	9238	**872**	**19,876**

Note: The bolded values in the table represent the best results, and the values marked with (*) represent the worst results.

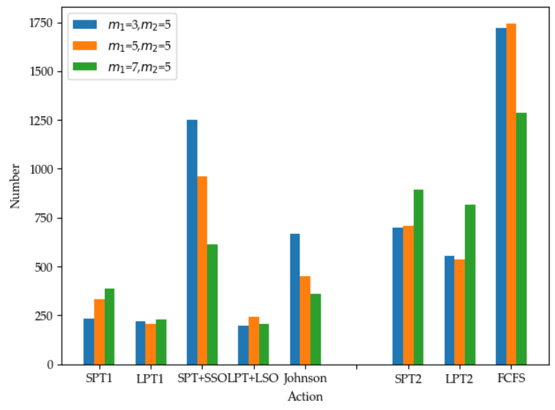

Figure 7. Comparison chart of action selection frequency.

Table 9. Comparison of objective functions for different multi-objective algorithms.

No.	**Job**	**Machine**	**NSGA-II**		**Q-Learning**		**AQL**	
			$C_{\max}$	*TEC*	$C_{\max}$	*TEC*	$C_{\max}$	*TEC*
R1	$n = 15$	$m_1 = 3$, $m_2 = 5$	221	1330	161	1221	**159**	**954**
R2	$n = 15$	$m_1 = 5$, $m_2 = 5$	165	2070	**151**	1661	**151**	**1653**
R3	$n = 15$	$m_1 = 7$, $m_2 = 5$	**149**	2000	168	2115	177	**1937**
R4	$n = 30$	$m_1 = 3$, $m_2 = 5$	410	2846	**321**	2359	325	**2218**
R5	$n = 30$	$m_1 = 5$, $m_2 = 5$	254	3117	271	3215	**247**	**2838**
R6	$n = 30$	$m_1 = 7$, $m_2 = 5$	239	3547	243	3895	**237**	**3515**
R7	$n = 50$	$m_1 = 3$, $m_2 = 5$	704	4503	547	4527	**511**	**4059**
R8	$n = 50$	$m_1 = 5$, $m_2 = 5$	445	7413	466	7403	**440**	**7327**
R9	$n = 50$	$m_1 = 7$, $m_2 = 5$	456	10,187	475	9806	**455**	**9238**
R10	$n = 100$	$m_1 = 3$, $m_2 = 5$	1356	9586	1132	9457	**1098**	**8483**
R11	$n = 100$	$m_1 = 5$, $m_2 = 5$	983	15,037	973	14,945	**936**	**14,899**
R12	$n = 100$	$m_1 = 7$, $m_2 = 5$	913	20,223	945	20,472	**872**	**19,876**

Note: The bolded values in the table represent the best results.

5. Conclusions

This paper, set against the backdrop of a steel manufacturing enterprise, focuses on the various production processes and material transportation. Due to the stringent temperature requirements of materials, we specifically investigated the two-stage BHFSP with transportation times. It formulates a multi-objective scheduling model with the objective of minimizing both the makespan and the total energy consumption. We designed the AQL algorithm to solve the model. Nine state features were designed based on real-time information about jobs, machines, and waiting processing queues in the blocked hybrid flow shop environment. Ten actions were formulated based on heuristic rules like SPT, FCFS, and Johnson. We proposed an adaptive objective selection strategy based on t-tests, wherein the algorithm calculates confidence to determine the most confident goal for the current action selection without relying on the fixed objective weights. Simulation analyses were conducted at different experimental scales, comparing single scheduling rules, the Q-learning algorithm, and NSGA-II. The experimental results demonstrate that the AQL algorithm can achieve optimal scheduling solutions in 92%, 83.3%, and 91.7% of the test instances, respectively. This research helps to optimize the production and transportation processes in process industries, reducing the impact of blocking and transportation time on completion time, and improving resource utilization. Additionally, this approach allows enterprises to consume less energy in terms of blocking and transportation. This is consistent with the research direction of green manufacturing mode in modern production.

The problem studied in this paper does not consider the number and capacity limitations of transportation vehicles. Future research can explore the coordination of production and transportation scheduling problems in multi-processing stage blocking hybrid flow shop environments when transportation resources are constrained.

Author Contributions: Conceptualization, K.X.; methodology, C.Y. and K.X.; software, C.Y. and W.S.; validation, K.X. and H.G.; formal analysis, H.G. and W.S.; resources, W.S.; data curation, C.Y.; writing—original draft preparation, C.Y.; writing—review and editing, K.X.; super-vision, K.X.; project administration, H.G.; funding acquisition, H.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Project of Liaoning BaiQianWan Talents Program under Grand No. 2021921089, the Science Research Foundation of Educational Department of Liaoning Province under Grand No. LJKQZ2021057 and LJKZ2060, and the Liaoning Province Xingliao Talents Plan project under Grant No. XLYC2006017.

Data Availability Statement: All data from the experiments are included in this article.

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

References

1. Cheng, Q.; Liu, C.; Chu, H.; Liu, Z.; Zhang, W.; Pan, J. A New Multi-Objective Hybrid Flow Shop Scheduling Method to Fully Utilize the Residual Forging Heat. *IEEE Access* **2020**, *8*, 151180–151194. [CrossRef]
2. Wardono, B.; Fathi, Y. A tabu search algorithm for the multi-stage parallel machine problem with limited buffer capacities. *Eur. J. Oper. Res.* **2004**, *155*, 380–401. [CrossRef]
3. Du, S.; Zhou, W.; Wu, D.; Fei, M. An effective discrete monarch butterfly optimization algorithm for distributed blocking flow shop scheduling with an assembly machine. *Expert Syst. Appl.* **2023**, *225*, 120113. [CrossRef]
4. Miyata, H.H.; Nagano, M.S.; Gupta, J.N.D. Solutions methods for m-machine blocking flow shop with setup times and preventive maintenance costs to minimise hierarchical objective-function. *Int. J. Prod. Res.* **2023**, *61*, 6308–6335. [CrossRef]
5. Cheng, C.-Y.; Pourhejazy, P.; Ying, K.-C.; Huang, S.-Y. New benchmark algorithm for minimizing total completion time in blocking flowshops with sequence-dependent setup times. *Appl. Soft Comput.* **2021**, *104*, 107229. [CrossRef]
6. Zhao, F.; Shao, D.; Wang, L.; Xu, T.; Zhu, N.; Jonrinaldi. An effective water wave optimization algorithm with problem-specific knowledge for the distributed assembly blocking flow-shop scheduling problem. *Knowl.-Based Syst.* **2022**, *243*, 108471. [CrossRef]
7. Niu, W.; Li, J. A two-stage cooperative evolutionary algorithm for energy-efficient distributed group blocking flow shop with setup carryover in precast systems. *Knowl.-Based Syst.* **2022**, *257*, 109890. [CrossRef]
8. Zhao, F.; Xu, Z.; Bao, H.; Xu, T.; Zhu, N.; Jonrinaldi. A cooperative whale optimization algorithm for energy-efficient scheduling of the distributed blocking flow-shop with sequence-dependent setup time. *Comput. Ind. Eng.* **2023**, *178*, 109082. [CrossRef]

9. Bao, H.; Pan, Q.; Ruiz, R.; Gao, L. A collaborative iterated greedy algorithm with reinforcement learning for energy-aware distributed blocking flow-shop scheduling. *Swarm Evol. Comput.* **2023**, *83*, 101399. [CrossRef]
10. Nagano, M.; Takano, M.; Robazzi, J. A branch and bound method in a permutation flow shop with blocking and setup times. *Int. J. Ind. Eng. Comput.* **2022**, *13*, 255–266. [CrossRef]
11. Wang, Y.; Wang, Y.; Han, Y. A Variant Iterated Greedy Algorithm Integrating Multiple Decoding Rules for Hybrid Blocking Flow Shop Scheduling Problem. *Mathematics* **2023**, *11*, 2453. [CrossRef]
12. Qin, H.-X.; Han, Y.-Y.; Zhang, B.; Meng, L.-L.; Liu, Y.-P.; Pan, Q.-K.; Gong, D.-W. An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem. *Swarm Evol. Comput.* **2022**, *69*, 100992. [CrossRef]
13. Shao, Z.; Shao, W.; Pi, D. LS-HH: A learning-based selection hyper-heuristic for distributed heterogeneous hybrid blocking flow-shop scheduling. *IEEE Trans. Emerg. Top. Comput. Intell.* **2023**, *7*, 111–127. [CrossRef]
14. Missaoui, A.; Boujelbene, Y. An effective iterated greedy algorithm for blocking hybrid flow shop problem with due date window. *RAIRO-Oper. Res.* **2021**, *55*, 1603–1616. [CrossRef]
15. Aqil, S.; Allali, K. Two efficient nature inspired meta-heuristics solving blocking hybrid flow shop manufacturing problem. *Eng. Appl. Artif. Intell.* **2021**, *100*, 104196. [CrossRef]
16. Qin, H.-X.; Han, Y.-Y.; Chen, Q.-D.; Li, J.-Q.; Sang, H.-Y. A double level mutation iterated greedy algorithm for blocking hybrid flow shop scheduling. *Control Decis.* **2022**, *37*, 2323–2332.
17. Zhao, F.-Q.; Du, S.-L.; Cao, J.; Tang, J.-X. Study on distributed assembly blocking flow shop scheduling algorithm. *J. Huazhong Univ. Sci. Technol. (Nat. Sci. Ed.)* **2022**, *50*, 138–142+148.
18. Wang, Y.; Jia, Z.; Zhang, X. A hybrid meta-heuristic for the flexible flow shop scheduling with blocking. *Swarm Evol. Comput.* **2022**, *75*, 101195. [CrossRef]
19. Feng, Y.; Kong, J. Multi-Objective Hybrid Flow-Shop Scheduling in Parallel Sequential Mode While Considering Handling Time and Setup Time. *Appl. Sci.* **2023**, *13*, 3563. [CrossRef]
20. Lei, D.; Su, B. A multi-class teaching–learning-based optimization for multi-objective distributed hybrid flow shop scheduling. *Knowl.-Based Syst.* **2023**, *263*, 110252. [CrossRef]
21. Geng, K.; Wu, S.; Liu, L. Multi-objective re-entrant hybrid flow shop scheduling problem considering fuzzy processing time and delivery time. *J. Intell. Fuzzy Syst.* **2022**, *43*, 7877–7890. [CrossRef]
22. Wu, X.; Cao, Z. An improved multi-objective evolutionary algorithm based on decomposition for solving re-entrant hybrid flow shop scheduling problem with batch processing machines. *Comput. Ind. Eng.* **2022**, *169*, 108236. [CrossRef]
23. Wang, J.; Wang, L.; Cai, J.; Li, J.; Su, X. Solution Algorithm of Multi-objective Hybrid Flow Shop Scheduling Problem. *J. Nanjing Univ. Aeronaut. Astronaut.* **2023**, *55*, 544–552.
24. Song, C. Improved NSGA-II algorithm for hybrid flow shop scheduling problem with multi-objective. *Comput. Integr. Manuf. Syst.* **2022**, *28*, 1777–17889.
25. Lei, D.-M.; Wang, T. An improved shuffled frog leaping algorithm for the distributed two-stage hybrid flow shop scheduling. *Control Decis.* **2021**, *36*, 241–248.
26. Song, C. A hybrid multi-objective teaching-learning based optimization for scheduling problem of hybrid flow shop with unrelated parallel machine. *IEEE Access* **2021**, *9*, 56822–56835. [CrossRef]
27. Li, P.; Xue, Q.; Zhang, Z.; Chen, Z.; Zhou, D. Multi-objective energy-efficient hybrid flow shop scheduling using Q-learning and GVNS driven NSGA-II. *Comput. Oper. Res.* **2023**, *159*, 106360. [CrossRef]
28. Wang, Y.; Wang, S.; Li, D.; Shen, C.; Yang, B. An improved multi-objective whale optimization algorithm for the hybrid flow shop scheduling problem considering device dynamic reconfiguration processes. *Expert Syst. Appl.* **2021**, *174*, 114793.
29. Cui, H.; Li, X.; Gao, L.; Zhang, C. Multi-population genetic algorithm with greedy job insertion inter-factory neighbourhoods for multi-objective distributed hybrid flow-shop scheduling with unrelated-parallel machines considering tardiness. *Int. J. Prod. Res.* **2023**, 1–19. [CrossRef]
30. Wang, J.; Li, X.; Zhu, X. Intelligent dynamic control of stochastic economic lot scheduling by agent-based reinforcement learning. *Int. J. Prod. Res.* **2012**, *50*, 4381–4395. [CrossRef]
31. Zhang, Z.; Zheng, L.; Li, N.; Wang, W.; Zhong, S.; Hu, K. Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning. *Comput. Oper. Res.* **2012**, *39*, 1315–1324. [CrossRef]
32. Lee, J.-H.; Kim, H.-J. Reinforcement learning for robotic flow shop scheduling with processing time variations. *Int. J. Prod. Res.* **2022**, *60*, 2346–2368. [CrossRef]
33. Zhao, F.; Zhang, L.; Cao, J.; Tang, J. A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. *Comput. Ind. Eng.* **2021**, *153*, 107082. [CrossRef]
34. 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.
35. Li, Z.; Wei, X.; Jiang, X.; Pang, Y. A kind of reinforcement learning to improve genetic algorithm for multiagent task scheduling. *Math. Probl. Eng.* **2021**, *2021*, 1796296. [CrossRef]
36. Luo, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Appl. Soft Comput.* **2020**, *91*, 106208. [CrossRef]
37. Zhang, J.; Cai, J. A Dual-Population Genetic Algorithm with Q-Learning for Multi-Objective Distributed Hybrid Flow Shop Scheduling Problem. *Symmetry* **2023**, *15*, 836. [CrossRef]

38. Cheng, L.; Tang, Q.; Zhang, L.; Zhang, Z. Multi-objective Q-learning-based hyper-heuristic with Bi-criteria selection for energy-aware mixed shop scheduling. *Swarm Evol. Comput.* **2022**, *69*, 100985. [CrossRef]
39. Chang, J.; Yu, D.; Zhou, Z.; He, W.; Zhang, L. Hierarchical Reinforcement Learning for Multi-Objective Real-Time Flexible Scheduling in a Smart Shop Floor. *Machines* **2022**, *10*, 1195. [CrossRef]
40. Li, R.; Gong, W.; Lu, C. A reinforcement learning based RMOEA/D for bi-objective fuzzy flexible job shop scheduling. *Expert Syst. Appl.* **2022**, *203*, 117380. [CrossRef]
41. Yuan, J.-L.; Chen, M.-C.; Jiang, T.; Li, C. Multi-objective reinforcement learning job scheduling method using AHP fixed weight in heterogeneous cloud environment. *Control Decis.* **2022**, *37*, 379–386.
42. Wu, X.; Yan, X. An Improved Q Learning Algorithm to Optimize Green Dynamic Scheduling Problem in a Reentrant Hybrid Flow Shop. *J. Mech. Eng.* **2022**, *58*, 246–259.
43. Wang, M.Y.; Sethi, S.P.; van de Velde, S.L. Minimizing makespan in a class of reentrant shops. *Oper. Res.* **1997**, *45*, 702–712. [CrossRef]