symmetry

MDPI

Article

A Q-Learning Evolutionary Algorithm for Solving the Distributed Mixed No-Idle Permutation Flowshop Scheduling Problem

Fangchi Zeng * and Junjia Cui

School of Mechanical and Transportation Engineering, Hunan University, Changsha, Hunan 410082, China; cuijunjia@hnu.edu.cn

* Correspondence: zengfangchi@hnu.edu.cn

Abstract: In this paper, a Distributed Mixed No-Idle Permutation Flowshop Scheduling Problem with Sequence-Dependent Setup Times (DMNIPFSP/SDST) is studied. Firstly, a multi-objective optimization model with completion time (makespan), Total Energy Consumption (TEC), and Total Tardiness (TT) as objectives is established. Based on problem characteristics and multi-objective characteristics, a Q-Learning Evolutionary Algorithm (QLEA) is proposed. Secondly, in order to improve the quality and diversity of the initial solution, two improved initialization strategies are proposed. Based on the characteristics of the problem solved (In the distributed system, symmetry design is adopted to ensure that the load of each workstation is relatively balanced in different time periods, avoid resource waste or bottleneck, and achieve the goal of no idle.), a novel population updating mechanism is designed to balance the ability of global exploration and local development of the algorithm. At the same time, a variable neighborhood local search based on Q-Learning is used to refine the non-dominated solution, thus guiding the population evolution. Finally, the simulation results show that this method has good performance in solving the multi-objective DMNIPFSP/SDST and can provide good economic benefits for enterprises.

Keywords: distributed manufacturing; sequence-dependent setup times; Q-Learning; variable neighborhood local search

Academic Editors: Deming Lei, Jingcao Cai and Angelo Freni

Received: 25 December 2024
Revised: 20 January 2025
Accepted: 28 January 2025
Published: 11 February 2025

Citation: Zeng, F.; Cui, J. A Q-Learning Evolutionary Algorithm for Solving the Distributed Mixed No-Idle Permutation Flowshop Scheduling Problem. *Symmetry* **2025**, *17*, 276. https://doi.org/10.3390/sym17020276

1. Introduction

In the process of smart manufacturing, it is crucial to allocate production resources efficiently to achieve higher economic benefits. Consequently, the job shop scheduling problem, which is closely related to production scheduling, has garnered widespread attention [1]. In job shop scheduling, multi-factory processing has become a mainstream trend [2]. To adapt to a multi-factory production environment, NADERI [3] proposed the definition of the distributed job shop scheduling problem. However, the No-Idle Permutation Flowshop Scheduling Problem (NIPFSP) is an extension of the permutation flowshop scheduling problem and represents a very important category within it. This problem is widely present in combinatorial optimization problems in modern industries such as glass processing, foundries, and steelmaking.

However, in actual production processes, two common situations arise: First, it is unrealistic for all machines to operate in a no-idle state. Most machines allow idle time, while some high-cost machines are required to operate under no-idle constraints. This scenario is usually referred to as the Mixed No-Idle Permutation Flowshop Scheduling Problem (MNIPFSP). Second, the machine may experience sudden malfunctions during

processing [4], which necessitates considering sequence-dependent setup times (SDSTs). From the existing literature, ALMEIDA et al. [5] proposed four algorithms to solve the no-idle flowshop scheduling problem with SDST, where the objective function is to minimize the completion time (makespan). The results showed that the iterative greedy algorithm performed well. ABREU and NAGANO [6] designed an adaptive large neighborhood search algorithm for the open shop scheduling problem with SDST. GUO et al. [7] introduced a fruit fly optimization algorithm, improved by a differential flight strategy, to solve the distributed flowshop scheduling problem with SDST. MIYATA and NAGANO [8] used a variable neighborhood iterative greedy algorithm to solve the distributed reverse job shop scheduling problem with SDST. KARABULUT et al. [9] employed an adaptive evolutionary algorithm to minimize the makespan for the distributed flowshop scheduling problem with SDST. From the aforementioned literature, three points can be noted: Most studies focus on economic indicators from the perspective of production efficiency, with relatively few considering energy, environmental factors, and customer satisfaction. Search strategies are often limited, with most algorithms using random methods to generate initial solutions. A good initialization strategy can significantly improve the quality and diversity of the population. Most proposed algorithms are suitable only for solving single-objective scheduling optimization problems. However, the solution space of multi-objective optimization problems is more complex, and further development is needed for algorithms that can effectively solve multi-objective optimization problems.

In recent years, carbon emissions and energy consumption have garnered widespread attention. As the manufacturing industry is the largest energy-consuming sector, energy consumption is often a critical factor that cannot be overlooked. If energy consumption is not controlled, excessive use will ultimately have negative effects on the natural environment and human society. Therefore, energy-efficient scheduling plays a crucial role in green manufacturing systems, mainly because energy savings can be achieved through production scheduling without requiring excessive capital investment, and it can easily be applied to existing manufacturing and production environments [10]. Moreover, in today's customer-oriented competitive market, minimizing Total Tardiness (TT) is also vital for the survival and development of an enterprise [11]. Reducing TT allows products to be delivered as quickly as possible, thereby increasing customer satisfaction and reducing company losses. In summary, in this paper, our optimization objectives for the Distributed Mixed No-Idle Permutation Flowshop Scheduling Problem with Sequence-Dependent Setup Times (DMNIPFSP/SDST) consider not only the minimization of makespan under SDST constraints but also the minimization of Total Energy Consumption (TEC) and TT. Since the single-objective Distributed Mixed No-Idle Permutation Flowshop Scheduling Problem (DMNIPFSP) has already been proven to be an NP-hard problem [12], the multi-objective DMNIPFSP/SDST with objectives of minimizing the makespan, TEC, and TT is also an NP-hard problem, presenting even greater complexity in terms of solution difficulty. Naturally, as the problem size increases, it becomes less suitable to solve using exact algorithms. Therefore, it is of great significance to propose effective meta-heuristic algorithms to solve the multi-objective DMNIPFSP/SDST. At the same time, distributed no-idle flowshop scheduling is related to symmetry, as symmetry helps to equalize task and resource allocation and reduce idle time. If the machines are symmetrical in function and performance, the scheduler can distribute tasks evenly, preventing some workstations from being overloaded or idle, thus achieving no-idle scheduling. Therefore, symmetry helps to improve system efficiency and reduce resource waste.

From the above studies, it is not difficult to see the limitations of the current research. (1) The use of meta-heuristic algorithm alone shows advantages in solving small-scale examples, but with the increase in problem scale, disadvantages will be shown; in particular,

the complex and changeable actual environment will lead to long computing time, slow convergence, and other problems, and it is difficult to obtain a satisfactory solution in a short time. (2) Although meta-heuristic algorithms are universal, their generality also leads to certain limitations. The unique structure of a particular problem is often ignored when solving the actual problem, which will negatively affect the search efficiency and the quality of the solution.

In recent years, reinforcement learning (RL) has been used to solve combinatorial optimization problems because of its simple and fast decision-making ability [13]. For example, KOOL et al. and BELLO et al. [14,15] utilized a combination of RL and neural networks to solve the traveling salesman problem. At the same time, RL has also been applied to production scheduling problems. CHEN et al. [16] proposed a genetic algorithm that integrates reinforcement learning for the flexible job shop scheduling problem. WANG [17], considering the uncertainty in shop-floor production environments, proposed a Q-Learning algorithm based on clustering and dynamic search to solve dynamic scheduling problems. For the dynamic flexible job shop scheduling problem under new job insertions, LUO [18] modeled the problem as a Markov decision process and solved it using reinforcement learning algorithms. LI et al. [19], based on a decomposition-based multi-objective evolutionary algorithm, proposed a Q-Learning-based adaptive parameter strategy and designed an RL-based variable neighborhood search strategy to guide the algorithm in selecting appropriate local search methods. To solve the distributed three-stage assembly scheduling problem, WANG et al. [20] integrated Q-Learning into the artificial bee colony algorithm, designing appropriate states and actions, enabling the algorithm to dynamically select the right search operator. Extensive experiments confirmed the effectiveness of the algorithm in solving distributed assembly shop scheduling problems considering equipment maintenance.

Therefore, this paper uses the advantages of a reinforcement learning algorithm to embed it into the structure of the meta-heuristic algorithm stage and proposes a Q-Learning Evolutionary Algorithm (QLEA) to solve the DMNIPFSP/SDST problem. According to the characteristics of the problem, a multi-perturbation operator is designed by using the advantage of a reinforcement learning algorithm for fast decision-making. It can dynamically adjust and automatically select suitable search strategies in the process of evolution, which solves the limitation of slow convergence and poor searchability of the classical algorithm framework. The main contributions are as follows:

First, a mathematical model is constructed with the objectives of minimizing makespan, TEC, and TT. To improve the quality and diversity of the initial solutions, two enhanced initialization strategies are proposed, combined with a random generation initialization method.

Second, a discretized population update mechanism is proposed to enhance the global search capability of the algorithm.

Third, to further enhance solution quality and precision, four neighborhood strategies are designed for local search, tailored to different optimization objectives.

Finally, a large number of instances are used to verify that the QLEA has certain advantages in solving the DMNIPFSP/SDST.

The remaining arrangement of this paper is as follows: Section 2 describes the DMNIPFSP/SDST. In Section 3, the QLEA solves the DMNIPFSP/SDST. Section 4 is QLEA performance verification. Section 5 is the conclusion.

2. Problem Description and Mathematical Model

2.1. Problem Description

This paper explores a DMNIPFSP/SDST, with the objectives of minimizing three objectives: makespan, TEC, and TT. This problem extends the DMNIPFSP by address-

ing two subproblems: ① assigning each job to the appropriate factory; ② determining the job processing sequence within each factory. Specifically, there are n jobs processed in F factories, with each factory having the same m machines. The processing speeds of the machines differ across factories, and each job can be processed in any factory [21]. Additionally, the DMNIPFSP/SDST with sequence-dependent setup times must satisfy the following basic assumptions: ① The job processing order on different machines follows the same sequence. ② Each job must be processed in the designated factory and cannot be transferred to another factory during processing. ③ At any given time, only one job can be processed on a machine, and once a job starts processing, it cannot be interrupted. These assumptions form the basis for the mathematical model of the DMNIPFSP/SDST, setting the stage for optimization across multiple conflicting objectives.

2.2. Mathematical Model

The relevant symbols used in this paper are defined in Table 1.

Table 1. Meaning of symbols.

	Meaning of Symbols
n	Number of jobs.
F	Number of factories.
m	Number of machines.
k	Job position index on the machine.
J_j	Set of jobs $J = \{J_1, \ldots, J_n\}$.
M_i	Set of machines $M = \{M_1, \ldots, M_m\}$.
F_f	Set of factories $F = \{F_1, \ldots, F_F\}$.
M'	No-idle machine set.
v_i	Processing speed on machine i.
$p_{i,j}$	Processing time of job J_j on machine M_i.
$C_{f,i,j}$	Completion time of job J_j on machine M_i in factory f.
$C_{\max}$	Maximum completion time across all factories.
$s^i_{j,l}$	Setup time between job J_j and J_l on machine M_i.
PP_i	Energy consumption per unit time for processing on machine M_i.
SP_i	Energy consumption per unit time for sequence-dependent setup on machine M_i.
IP_i	Energy consumption per unit time for idling on machine M_i.

The DMNIPFSP can be described as follows: n jobs $J = \{J_1, J_2, \ldots, J_n\}$ need to be assigned to F factories $F = \{F_1, F_2, \ldots, F_F\}$, with each factory containing m machines $M = \{M_1, M_2, \ldots, M_m\}$. Each job follows the same route across the machines, and each job consists of m operations. The i-th operation can only be executed on machine Mi. Notably, no idle time is allowed between consecutive jobs on the same machine. Once an operation begins, it cannot be interrupted, and each machine can process only one job at a time. However, in actual production processes, it is unrealistic for all machines to operate under a no-idle condition. Most machines allow idle time, while certain high-cost machines must adhere to no-idle constraints. Moreover, machines often require additional operations between consecutive jobs. Therefore, this paper develops a mathematical model for the DMNIPFSP/SDST, with the multi-objective function aiming to minimize makespan, TEC, and TT, as shown below.

Objective:

$$F_1 = \min\{C_{\max}\} \tag{1}$$

$$F_2 = TEC = \min\{PEC + SEC + IEC\} \tag{2}$$

$$F_3 = TT = \min\{\sum_{f=1}^{F}\sum_{j=1}^{n}\max(C_{m,j} - d_j, 0)\} \tag{3}$$

Constraint:

$$C_{\max} \geq C_{f,i,j} \forall f \in \{1,\ldots,F\}, \forall i \in \{1,\ldots,m\}, \forall j \in \{1,\ldots,n\} \tag{4}$$

$$\sum_{k=1}^{n} \sum_{f=1}^{F} X_{f,j,k} = 1, \forall j \in \{1,\ldots,n\} \tag{5}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{n} X_{f,j,k} = 1, \forall f \in \{1,\ldots,F\} \tag{6}$$

$$X_{f,j,k} = \sum_{l=1}^{n} Y_{f,l,k,j} \forall f \in \{1,\ldots,F\}, \forall j \in \{1,\ldots,n\}, \forall k \in \{2,\ldots,n\} \tag{7}$$

$$X_{f,j,k} = \sum_{l=1}^{n} Y_{f,l,k+1,j} \forall f \in \{1,\ldots,F\}, \forall j \in \{1,\ldots,n\}, \forall k \in \{1,\ldots,n-1\} \tag{8}$$

$$Y_{f,l,1,j} = 0 \forall f \in \{1,\ldots,F\}, \forall j,l \in \{1,\ldots,n\} \tag{9}$$

$$C_{f,1,k} \geq \sum_{j=1}^{n} X_{f,j,k} \cdot \frac{p_{i,j}}{v_1} \forall f \in \{1,\ldots,F\}, \forall k \in \{1,\ldots,n\} \tag{10}$$

$$C_{f,i,k} \geq C_{f,i-1,k} + \sum_{j=1}^{n} X_{f,j,k} \cdot \frac{p_{i,j}}{v_i} \forall f \in \{1,\ldots,F\}, \forall i \in \{2,\ldots,m\}, \forall k \in \{1,\ldots,n\} \tag{11}$$

$$C_{f,i,k} \geq C_{f,i,k-1} + \sum_{j=1}^{n} X_{f,j,k} \cdot \frac{p_{i,j}}{v_i} \forall f \in \{1,\ldots,F\}, \forall i \in M', \forall k \in \{1,\ldots,n\} \tag{12}$$

$$\begin{aligned} &C_{f,i,k} \geq C_{f,i,k-1} + \\ &\sum_{j=1}^{n} X_{f,j,k} \cdot \frac{p_{i,j}}{v_i} + \sum_{j=1}^{n} \sum_{l=1}^{n} Y_{f,l,k,j} \cdot s_{j,l}^{i} \forall f \in \{1,\ldots,F\}, \forall i \in M', \forall k \in \{2,\ldots,n\} \end{aligned} \tag{13}$$

$$PEC = \sum_{f=1}^{F} \sum_{j=1}^{n} \sum_{i=1}^{m} PP_i \cdot X_{f,j,k} \cdot \frac{p_{i,j}}{v_i} \forall k \in \{1,\ldots,n\} \tag{14}$$

$$SEC = \sum_{f=1}^{F} \sum_{j=1}^{n} \sum_{l=1}^{n} \sum_{i=1}^{m} SP_i \cdot X_{f,j,k} \cdot X_{f,l,k+1} \cdot s_{j,l}^{i} \forall k \in \{1,\ldots,n\} \tag{15}$$

$$IEC = \sum_{f=1}^{F} \sum_{j=1}^{n} \sum_{i=1}^{m} IP_i \cdot X_{f,j,k+1} \cdot (C_{f,i,k+1} - C_{f,i,k} - s_{k,k+1}^{i} - \frac{p_{i,j}}{v_i}) \forall k \in \{1,\ldots,n-1\} \tag{16}$$

$$C_{f,i,k} \geq 0 \forall f \in \{1,\ldots,F\}, \forall i \in \{1,\ldots,m\}, \forall k \in \{1,\ldots,n\} \tag{17}$$

Decision Variable:

$$X_{f,i,k} \in \{0,1\} \forall f \in \{1,\ldots,F\}, \forall i \in \{1,\ldots,m\}, \forall k \in \{1,\ldots,n\} \tag{18}$$

$$Y_{f,l,k,j} \in \{0,1\} \forall f \in \{1,\ldots,F\}, \forall j,k,l \in \{1,\ldots,n\} \tag{19}$$

Equations (1)–(3) represent the makespan, TEC, and TT; Constraint (4) ensures that $C_{\max}$ is the makespan across all factories; Constraint (5) indicates that each job can only be assigned to one factory; Constraint (6) specifies that each position can hold at most one job; Constraint (7) ensures that in factory f, only one job precedes job J_j in position k; Constraint (8) ensures that in factory f, only one job follows job J_j in position k; Constraint (9) states that the setup time for the first job in each factory is 0; Constraints (10) and (11) ensure that a job can only be processed on the next machine after the previous machine has completed processing; Constraint (12) represents the completion time for machines under

the no-idle constraint; Constraint (13) represents the completion time for conventional machines; Constraint (14) denotes the energy consumption during processing; Constraint (15) denotes the energy consumption during sequence-dependent setup times; Constraint (16) denotes the energy consumption during idle times; Constraint (17) states that the completion time of each job is non-negative; and Constraints (18) and (19) define the values of the decision variables.

3. QLEA for Solving DMNIPFSP/SDST

3.1. Encoding and Decoding

The paper adopts the job encoding method proposed by PAN [22], where each individual represents a solution in the feasible domain. Each feasible solution π is represented by a two-dimensional array, consisting of F dimensions, where each dimension represents the job processing sequence of a factory, indicated by the job index numbers for that factory. Based on the description of the encoding, a feasible solution π can directly represent a collection of job sequences for F factories, denoted as $\pi = \{\pi_1, \pi_2, \ldots, \pi_F\}$, where $\pi_l = \{\pi_{l,1}, \pi_{l,2}, \ldots, \pi_{l,nl}\}$ indicates the processing sequence of jobs within that factory. Additionally, $\sum_{l=1}^{F} n_l = n$; the total length of the encoding sequence equals the number of all jobs. Since a feasible solution π contains both the jobs assigned to each factory and their processing sequences within that factory, this encoding method can be easily decoded into a scheduling solution for DMNIPFSP/SDST.

To better understand the encoding and decoding process, this paper presents an example with 9 jobs (n = 9) and 2 factories (F = 2), assuming that a feasible solution to the problem is $\pi = \{\pi_1, \pi_2\}$, where $\pi_1 = \{J_7, J_8, J_2, J_6, J_3\}$,$\pi_2 = \{J_5, J_4, J_1, J_9\}$, as shown in Figure 1.

F_1	7	8	2	6	3
F_2	5	4	1	9	

Figure 1. Example of encoding.

Accordingly, the decoding is as follows: In factory 1, assign jobs J_2, J_3, J_6, J_7, and J_8, totaling five jobs, and process them in the given order. In factory 2, assign jobs J_1, J_4, J_5, and J_9, totaling four jobs, and process them in the order of $J_5 \rightarrow J_4 \rightarrow J_1 \rightarrow J_9$.

3.2. Population Initialization

Since the initialization method directly impacts the solution quality of the algorithm, employing various strategies for initialization can effectively improve the initial solution quality and increase the diversity of the population. It is well known that the LR (Liu–Reeves) heuristic algorithm exhibits excellent performance when solving permutation flowshop scheduling problems [23]. Based on this, and in accordance with the characteristics of DMNIPFSP/SDST, this paper extends the related index functions and proposes two improved LR rules for generating the initial population, denoted as LR1 and LR2. The specific computational steps are as follows:

The detailed description of the improved LR1 rule is as follows:

Step 1: Calculate the average, standard deviation, and skewness of the completion times of each job on all machines and sum these values.

Step 2: For jobs j = 1, 2, …, n, sort them in descending order based on the results computed in Step 1.

Step 3: Insert the first F jobs from the sorted order in Step 2 into F different factories sequentially.

Step 4: Compute the index values for the $F + 1$ job and subsequent jobs using the index function, as detailed in Equations (17)–(19).

Step 5: Identify the job with the smallest index value obtained in Step 4.

Step 6: Insert the job identified in Step 5 into the factory that currently has the minimum completion time.

Step 7: Repeat Steps 4 to 6 until all jobs have been assigned.

Suppose we take job J_j as an example and find the factory g^* with the minimum current completion time, which is calculated as follows:

$$IF_{j,n_{g^*}} = (c - n_{g^*}) \times IT_{j,n_{g^*}} + C_{m,j} \tag{20}$$

$$IT_{j,n_{g^*}} = c \times m \times \sum_{i=2}^{m} \frac{t_{ij}}{c \times i + n_{g^*} \times (m - i)} \tag{21}$$

$$t_{ij} = \max\{(C_{i-1,j} - C_{i,[n_{g^*}]}), 0\} \tag{22}$$

where c is a constant ($c = n/F - 2$, where n represents the number of jobs, and F represents the number of factories); m denotes the number of machines; n_{g^*} signifies the current number of jobs in factory g^*; $[n_{g^*}]$ represents the job corresponding to position n_{g^*}; $C_{m,j}$ denotes the completion time of job j on the m-th machine.

Detailed Description of the Improved LR2 Rule:

Step 1: Calculate the completion time of each job on all machines.

Step 2: Sort the jobs j = 1, 2, . . ., n in descending order based on the results from Step 1.

Step 3: Insert the first F jobs from the sorted order obtained in Step 2 sequentially into F different factories.

Step 4: Calculate the index values for the job $F + 1$ and subsequent jobs using the index function, as detailed in Equations (17)–(19).

Step 5: Identify the job with the smallest index value obtained in Step 4.

Step 6: Insert the job identified in Step 5 into the factory that currently has the minimum completion time.

Step 7: Repeat Steps 4 to 6 until all jobs are assigned.

In QLEA, an external archive set (AS) is established to store the non-dominated solutions found during the evolutionary process of the population. By continuously updating AS, the solution approaches the Pareto Frontier (PF). The initial population is generated using the above two heuristic mixed strategies, with the population size set to n, initializing 50% of individuals with LR1 and 50% with LR2. Simultaneously, the non-dominated solutions obtained after initialization are added to AS.

3.3. Population Evolution Mechanism

This paper proposes a QLEA to solve the DMNIPFSP/SDST. The specific update steps are as follows:

Step 1: Randomly select three individuals from the AS, each denoted by $\varepsilon, \beta, \delta$. Generate a random number r, where r is in the range [0,1]. If $r \leq 1/3$, then remove the jobs that have the same job code at the same position from the current individual and the ε individual. If $1/3 < r \leq 2/3$, then remove the jobs that have the same job code at the same position from the current individual and the β individual; otherwise, remove the jobs that have the same job code at the same position from the current individual and the δ individual, thus obtaining the reference individual job sequence ω_1.

Step 2: Introduce the reference value $K = |(L \cdot C)/2|$, where L is the convergence factor and C is the oscillation factor. Both L and C are random numbers within the range [0, 2].

Step 3: Cycle through the job sequence obtained from Step 1. For each job, generate a corresponding random number in the interval [0,1]. Then, determine if this random number is less than the K value. If it is less than K, retain the job at its current position; otherwise, remove it to obtain the optimized sequence ω_2.

Step 4: Process the optimized sequence ω_2 obtained from Step 3 using the reference individual job sequence ω_1 obtained from Step 1. In this sequence, each job has a 50% probability of being swapped (for example, if a job a in the optimized sequence ω_2 is located at position j, find the job b at position j in job sequence ω_1, and swap job a in sequence ω_1 with job b). There is also a 50% probability of performing an insertion (for example, if a job a in the optimized sequence ω_2 is located at position j, find the job b at position j in job sequence ω_1, and insert job b from sequence ω_1 before job a). This results in the next generation of individuals.

To facilitate a clearer understanding of the aforementioned process, the specific operations are illustrated in Figures 2–5.

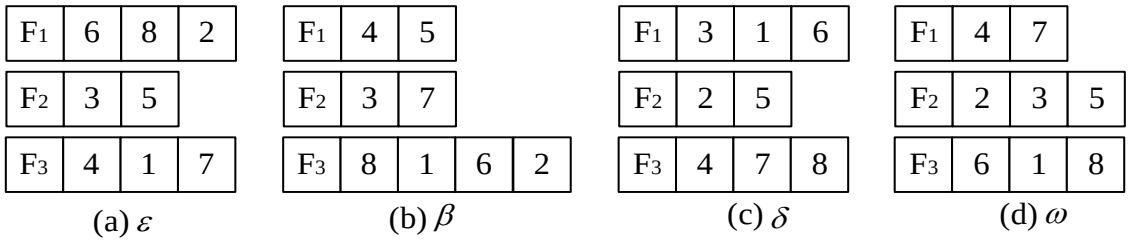

Figure 2. Sequence of jobs for different individuals.

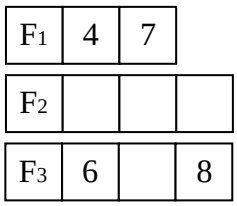

Figure 3. Results of Step 1.

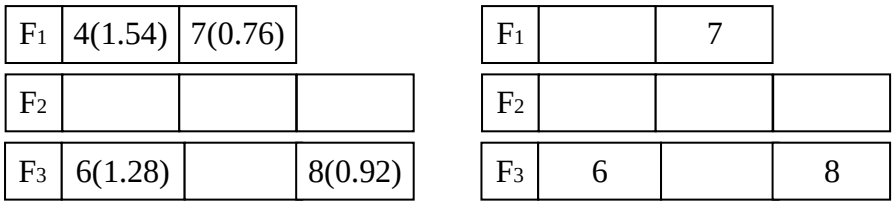

Figure 4. Results of Step 3.

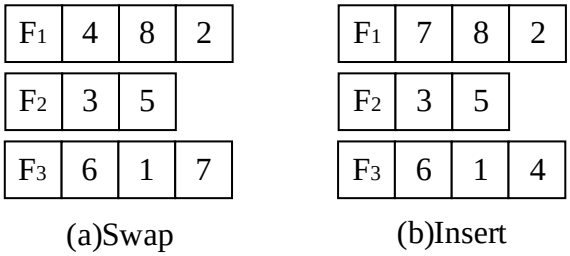

Figure 5. Schematic diagram of insertion and swapping.

Randomly select three individuals from the set AS, each denoted as ε, β, δ. See Figure 2 for details. Execute Step 1, assume $r < 1/3$, and then remove the parts that have the same job code from the current individual and the α individual as shown in Figure 3.

Execute Step (2), assuming K = 1.3.

Execute Step 4; if all the jobs that meet the requirements are subject to the swapping probability, then refer specifically to Figure 5a. If job numbers 7 and 6 meet the swapping probability, while job number 8 meets the insertion probability, then refer specifically to Figure 5b.

3.4. Variable Neighborhood Local Search Based on Q-Learning

The core idea of Q-Learning is to map actions to states to accumulate the maximum reward from the environment. It evaluates an agent's actions based on environmental signals rather than training the agent to choose the correct actions. The agent continuously interacts with the environment, selecting appropriate actions based on feedback, ultimately achieving optimal decision-making. The model is illustrated in Figure 6. Initially, the agent acquires the current state S_t and performs an action at time step t. Subsequently, the environment transitions to state S_{t+1}, and the agent receives a reward R. In the next time step $t + 1$, the agent performs an action at $t + 1$ and transitions to the next state. Appropriate actions enable the agent to easily find an optimal strategy to achieve the greatest long-term reward. The Q-Learning algorithm records feedback values between states and actions in a Q-table, updated by the following Equation (23):

$$Q(S_t, a_t) = (1 - \alpha)Q(S_t, a_t) + \alpha(R + \gamma \max(Q(S_{t+1}, a_{t+1}))) \quad (23)$$

where St represents the state at time t; at represents the action taken at time t; S_{t+1} represents the state at time $t + 1$; a_{t+1} represents the action taken at time $t + 1$; and the learning rate and discount factor are represented.

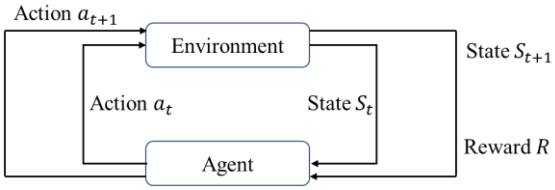

Figure 6. Q-Learning framework.

The archive set AS stores multiple Pareto solutions with superior advantages, allowing for further refinement of these solutions through increased effort. In the QLEA, a local search strategy is adopted to further refine the non-dominated solutions. Additionally, to reduce computational resources, the appropriate local search operator is selected each time a local search is conducted. This paper proposes a Q-Learning-based variable neighborhood local search to refine non-dominated solutions, thereby guiding the population evolution.

Since the search process is based on the continuous updating of the non-dominated solution set, this paper uses the Inverse Generational Distance (IGD) [24] as a metric to evaluate changes in the solution set. It calculates the average distance from each reference point on the Pareto approximation front to the closest solution in the set, assessing both diversity and convergence. Hence, the reward values use changes in the IGD value to update the state-action table. According to the existing literature, no neighborhood structure is currently suitable for solving all problems. Considering problem characteristics, three populations are defined based on three optimization objectives: makespan, TEC, and TT. Additionally, three key factories are defined: the one with the maximum completion time (F_c), the one with maximum energy consumption (F_e), and the one with maximum

tardiness (F_t). Based on this, the Q-Learning-based variable neighborhood local search mechanism is designed as follows:

State Set: State = {0,1}. State s = 0 indicates that after the most recent round, the reward value is less than 0; State s = 1 indicates that the reward value is greater than 0.

Action Set: Action = {LS1, LS2, LS3}. Based on different objectives leading to different key factories, this paper designs three types of local search actions, as follows:

(a) LS1: Focusing on the factory with the maximum completion time as the key factory, perform the following four neighborhood operations:

LS1(1): Randomly select a job from key factory F_c and swap it with a subset of jobs (half of all jobs in the same factory) to save feasible solutions.

LS1(2): Randomly select a position p_1 within key factory F_c, and then randomly select another position p_2 from a different factory outside the key factory. Swap the jobs corresponding to these two positions and save the feasible solution.

LS1(3): Randomly select a position p_1 within key factory F_c, and then randomly select another position p_2 from a different factory outside the key factory. Insert the job corresponding to position p_2 into position $p_1 + 1$, insert the job corresponding to position p_1 into position $p_2 + 1$, and save the feasible solution.

LS1(4): Randomly select a position p_1 within key factory F_c, and then randomly select another position p_2 from a different factory outside the key factory. Insert the job corresponding to position p_2 into position $p_1 - 1$, insert the job corresponding to position p_1 into position $p_2 - 1$, and save the feasible solution.

(b) LS2 and LS3 are similar to LS1, also performing four types of neighborhood operations. The difference lies in replacing the key factory with factories.

(c) Reward value: Each time an action is performed, it may cause a change in the non-dominant solution set. The change in IGD value was used to measure the change in AS, as shown in Equation (24):

$$R = \exp(IGD(AS', AS)) - 1.0 \tag{24}$$

where AS indicates the unsupported solution set before the execution; $R < 0$ indicates that the non-dominant solution has not improved; R > 0 indicates that the non-dominated solution set has been improved. The calculation method of IGD is shown in Equation (25):

$$IGD(A, PF^*) = \frac{\sum_{x \in PF^*} \min_{y \in A} d_e(x, y)}{|PF^*|} \tag{25}$$

where A represents an approximate non-dominant solution set; PF * represents the optimal solution set or the reference solution set; $d_e(x, y)$ represents the Euclidean distance between the solution x and y.

In QLEA, variable neighborhood local search based on Q-Learning is used to refine the non-dominated solution, thus guiding the population evolution. In order to select the appropriate local search operator each time the local search is performed, the computational resources are reduced.

3.5. Elite Strategy

This paper designs a new elitism strategy based on the non-dominated sorting and crowding distance methods used in the NSGA-II algorithm [25]. The criteria for evaluation are as follows: solutions with smaller non-dominated ranking values are better than those with larger ranking values. To preserve population diversity, solutions with greater crowding distances are preferred over those with smaller distances. Thus, a high-quality Pareto optimal solution set typically consists of solutions with smaller non-dominated

ranking values and larger crowding distances. In the population update mechanism of the QLEA, first, the current population individuals and the archive set AS are merged, all individuals are subject to non-dominated sorting, and their respective crowding distances are calculated. Second, individuals with smaller non-dominated ranking values are prioritized; when ranking values are the same, individuals with larger crowding distance values are chosen. Finally, the next generation of the population is selected based on the aforementioned elitism strategy. Figure 7 shows the selection process.

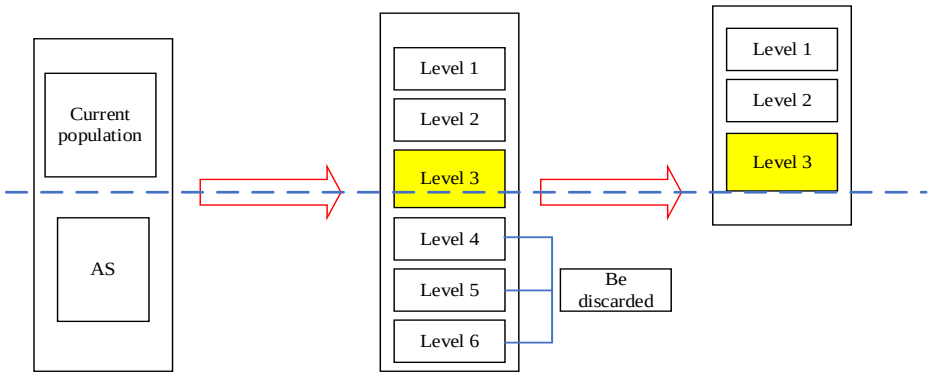

Figure 7. Schematic diagram of elite selection strategy.

3.6. QLEA Framework

Based on the above description, the complete framework of the QLEA is given. First, the parameters required for the algorithm are set up, and the initial population and AS are established. Next, the cycle is executed: the population renewal mechanism refines the non-dominated solution in the local search phase, and the elite strategy builds the next generation population. When the termination condition is met, the loop exits. Finally, the AS is output as a Pareto optimal solution. The detailed pseudocode of the QLEA is shown in Algorithm 1.

Algorithm 1: QLEA

Input: Population size n; Learning rate α; Discount factor γ.
Output: AS
1. **Begin**
2. According to different optimization objectives, it can be divided into three sub-populations.
3. The mixed initialization strategy was adopted to start the population.
4. **while** *The termination condition is not met* **do**
5. Perform Section 3.3 population update mechanism;
6. Perform the local search mechanism in Section 3.4;
7. Perform the Elite Policy in Section 3.5;
8. **end while**
9. The population is merged and the non-dominated solution set AS is output.
10. **end**

4. Analysis of Experimental Results

4.1. Experimental Settings

To evaluate the performance of the proposed QLEA, this study refers to the datasets from Ruiz et al. [26] and Ruiz and Stutzle [27], including problem instances with two different sequence-dependent setup times (SDST-50, SDST-100). The combinations of job quantities and machines are denoted as $n \in \{20, 50, 100, 200, 500\}$ and $m \in \{5, 10, 20\}$, respectively. Experiments are conducted using the last instance in each size category, with the number of factories represented by $F = \{2, 4, 6\}$, differing machine processing speeds

by $v_i = \{1.0, 2.0, 3.0\}$, per unit time processing energy consumption by $PP_i = 4 \times v_i^2 kw$, per unit time sequence-dependent setup energy consumption by $SP_i = [1kw, 2kw]$, and per unit time idle energy consumption by $IP_i = 0.25 \times PP_i kw$. Additionally, since this paper addresses the MNIPFSP, three different mixed no-idle constraint scenarios are considered: ① randomly selecting 25% of the machines to be no-idle; ② randomly selecting 50% of the machines to be no-idle; ③ randomly selecting 75% of the machines to be no-idle. Thus, the total number of instances in the benchmark tests is denoted by $2 \times 5 \times 3 \times 3 \times 3 = 270$.

4.2. Evaluation Metrics

To assess the performance of the QLEA and other comparative algorithms in solving the mentioned instances, this study employs two performance metrics: inverted generational distance (IGD) and coverage (C-metric) [28]. IGD is a comprehensive performance measure that considers both diversity and convergence, while the C-metric is used to measure the dominance relationship between the non-dominated solution sets of two algorithms; e.g., C (A, B) indicates the proportion of solutions in the non-dominated set of algorithm B that are dominated by the non-dominated set of algorithm A.

4.3. Parameter Settings

Because the value of control parameters has an important effect on the performance of the algorithm, selecting the best control parameter value is helpful to improve the performance of the algorithm. Therefore, the control parameters of the QLEA proposed in this paper are population size (n), learning factor (α), and discounting factor (γ). In order to determine the best combination of these three parameters, the Taguchi method [29] is designed, and four different level factors are designed for each parameter. The parameter levels of each influencing factor are shown in Table 2. An example of SDST-100 with $20 \times 20 \times 2$ mixed no-idle constraint condition type ① is conducted by using an $L_{16}(4^3)$ orthogonal table. Each parameter combination was run independently 10 times, and the running time under each parameter group was $20 \times 20 \times 2 \times 5$ milliseconds. IGD was selected as the evaluation index. The specific parameter combination is shown in Table 3 and Figure 8.

Table 2. Parameter levels.

Level	n	α	γ
1	20	0.1	0.3
2	30	0.2	0.4
3	40	0.3	0.5
4	50	0.4	0.6

Table 3. Orthogonal table and experimental results.

Level	n	α	γ	IGD
1	1	1	1	5043.33
2	1	2	2	6684.34
3	1	3	3	4255.75
4	1	4	4	4101.73
5	2	1	2	5621.10
6	2	2	1	2056.07
7	2	3	4	5549.18
8	2	4	3	5164.86
9	3	1	3	4143.86
10	3	2	4	3097.66

Table 3. *Cont.*

Level	n	α	γ	IGD
11	3	3	1	2623.52
12	3	4	2	4031.79
13	4	1	4	3394.69
14	4	2	3	7694.51
15	4	3	2	4221.34
16	4	4	1	5229.61

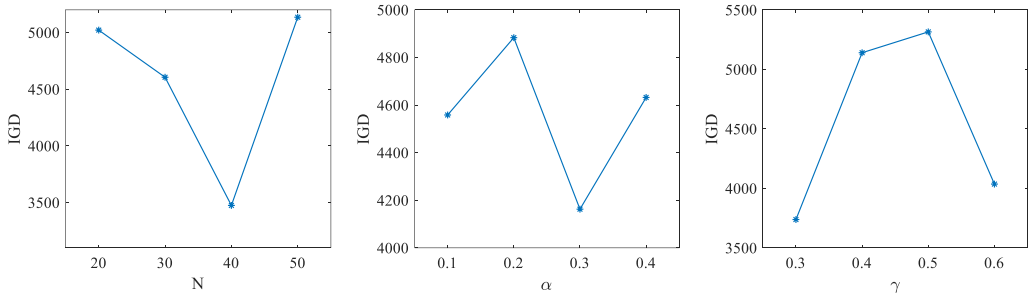

Figure 8. Main effect diagram of parameters.

Table 4 and Figure 8 present the response values for different levels of factors, with the optimal values highlighted in bold. The range for population size n is the largest, indicating that variations in population size have the most significant impact on the outcomes of the problem being solved. This is because the population size determines the range of the search space; if set too low, it can lead to an overly restricted search area, inhibiting effective exploration of the entire space. Conversely, if set too high, it may consume excessive time for solving, reducing the algorithm's computational efficiency and convergence. The discount factor γ follows in significance, and the learning rate α has the smallest range, suggesting that variations in α have the least impact on the results of the problem. However, setting α appropriately can enhance the algorithm's effectiveness in solving the problem. Consequently, the optimal combination of these three parameters is $n = 40$, $\gamma = 0.3$, $\alpha = 0.3$.

Table 4. Parameter response table.

Level	n	α	γ
1	5021.2880	4558.2457	3737.8825
2	4605.3024	4883.1463	5139.6425
3	3474.2068	4162.4464	5314.7450
4	5135.0401	4631.9988	4035.8150
Range	1660.8334	720.6999	1576.8605
Rank	1	3	2

4.4. Effectiveness of Neighborhood Search Strategy

To verify the effectiveness of the proposed neighborhood search strategy, a comparison was conducted between the QLEA incorporating a Q-Learning-based neighborhood search and a variant without this strategy (named QLEA-RE). Both algorithms share all other components. The comparison utilized two evaluation metrics: IGD and C-metric, to analyze and contrast the performance of these two algorithms across different problem instances. The optimal results are highlighted in bold, as detailed in Tables 5–8.

Table 5. IGD values of QLEA and QLEA-RE under SDST-50 constraint.

Type	①		②		③	
	QLEA	**QLEA-RE**	**QLEA**	**QLEA-RE**	**QLEA**	**QLEA-RE**
$F = 2$	$\mathbf{1.47 \times 10^4}$	8.32×10^5	$\mathbf{1.93 \times 10^4}$	9.06×10^5	$\mathbf{9.98 \times 10^4}$	7.88×10^5
$F = 4$	$\mathbf{5.68 \times 10^4}$	8.13×10^5	$\mathbf{210 \times 10^4}$	9.74×10^5	$\mathbf{1.33 \times 10^5}$	7.53×10^5
$F = 6$	$\mathbf{4.35 \times 10^4}$	8.21×10^5	$\mathbf{1.30 \times 10^5}$	8.31×10^5	$\mathbf{1.18 \times 10^5}$	8.32×10^5
Mean	$\mathbf{3.83 \times 10^4}$	8.22×10^5	$\mathbf{5.68 \times 10^4}$	9.04×10^5	$\mathbf{1.17 \times 10^5}$	7.91×10^5
$n = 20$	$\mathbf{3.48 \times 10^3}$	2.16×10^5	$\mathbf{3.85 \times 10^4}$	2.05×10^5	$\mathbf{5.70 \times 10^4}$	1.80×10^5
$n = 50$	$\mathbf{1.11 \times 10^2}$	3.53×10^5	$\mathbf{3.45 \times 10^2}$	3.78×10^5	$\mathbf{6.47 \times 10^4}$	3.27×10^5
$n = 100$	$\mathbf{1.37 \times 10^5}$	7.30×10^5	$\mathbf{5.35 \times 10^4}$	8.42×10^5	$\mathbf{7.39 \times 10^4}$	7.52×10^5
$n = 200$	$\mathbf{0.00 \times 10^0}$	2.57×10^6	$\mathbf{1.74 \times 10^5}$	2.83×10^6	$\mathbf{3.49 \times 10^5}$	2.46×10^6
$n = 500$	$\mathbf{0.00 \times 10^0}$	2.68×10^6	$\mathbf{1.98 \times 10^5}$	2.93×10^6	$\mathbf{3.66 \times 10^5}$	2.72×10^6
Mean	$\mathbf{2.81 \times 10^4}$	1.31×10^6	$\mathbf{9.29 \times 10^4}$	1.44×10^6	$\mathbf{1.82 \times 10^5}$	1.29×10^6
$m = 5$	$\mathbf{7.89 \times 10^3}$	1.16×10^5	$\mathbf{1.70 \times 10^4}$	9.59×10^4	$\mathbf{1.51 \times 10^4}$	1.03×10^5
$m = 10$	$\mathbf{1.28 \times 10^4}$	5.82×10^5	$\mathbf{3.22 \times 10^4}$	6.01×10^5	$\mathbf{6.73 \times 10^4}$	5.14×10^5
$m = 20$	$\mathbf{8.67 \times 10^4}$	1.59×10^6	$\mathbf{1.11 \times 10^5}$	1.81×10^6	$\mathbf{2.43 \times 10^5}$	1.58×10^6
Mean	$\mathbf{3.58 \times 10^4}$	7.63×10^5	$\mathbf{5.34 \times 10^4}$	8.36×10^5	$\mathbf{1.08 \times 10^5}$	7.32×10^5

Table 6. IGD values of QLEA and QLEA-RE under SDST-100 constraint.

Type	①		②		③	
	QLEA	**QLEA-RE**	**QLEA**	**QLEA-RE**	**QLEA**	**QLEA-RE**
$F = 2$	$\mathbf{0.00 \times 10^0}$	9.08×10^5	$\mathbf{8.27 \times 10^4}$	7.37×10^5	$\mathbf{2.11 \times 10^4}$	8.79×10^5
$F = 4$	$\mathbf{2.05 \times 10^4}$	8.70×10^5	$\mathbf{3.86 \times 10^4}$	8.53×10^5	$\mathbf{9.59 \times 10^4}$	7.38×10^5
$F = 6$	$\mathbf{1.41 \times 10^5}$	7.26×10^5	$\mathbf{8.25 \times 10^3}$	8.37×10^5	$\mathbf{7.49 \times 10^4}$	8.09×10^5
Mean	$\mathbf{5.38 \times 10^4}$	8.35×10^5	$\mathbf{4.32 \times 10^4}$	8.09×10^5	$\mathbf{6.40 \times 10^4}$	8.09×10^5
$n = 20$	$\mathbf{0.00 \times 10^0}$	2.20×10^5	$\mathbf{7.83 \times 10^3}$	2.29×10^5	$\mathbf{4.73 \times 10^3}$	2.51×10^5
$n = 50$	$\mathbf{0.00 \times 10^0}$	4.03×10^5	$\mathbf{2.36 \times 10^4}$	3.82×10^5	$\mathbf{5.59 \times 10^3}$	3.62×10^5
$n = 100$	$\mathbf{9.65 \times 10^4}$	8.48×10^5	$\mathbf{1.15 \times 10^5}$	7.14×10^5	$\mathbf{1.46 \times 10^5}$	6.78×10^5
$n = 200$	$\mathbf{1.51 \times 10^5}$	2.39×10^6	$\mathbf{1.79 \times 10^4}$	2.46×10^6	$\mathbf{1.18 \times 10^5}$	2.51×10^6
$n = 500$	$\mathbf{1.62 \times 10^5}$	2.58×10^6	$\mathbf{1.89 \times 10^4}$	3.58×10^6	$\mathbf{2.46 \times 10^5}$	3.62×10^6
Mean	$\mathbf{8.19 \times 10^4}$	1.29×10^6	$\mathbf{3.66 \times 10^4}$	1.47×10^6	$\mathbf{1.04 \times 10^5}$	1.48×10^6
$m = 5$	$\mathbf{4.77 \times 10^3}$	1.33×10^5	$\mathbf{8.29 \times 10^3}$	1.33×10^5	$\mathbf{1.99 \times 10^4}$	1.09×10^5
$m = 10$	$\mathbf{1.52 \times 10^4}$	4.40×10^5	$\mathbf{1.50 \times 10^4}$	4.38×10^5	$\mathbf{5.91 \times 10^4}$	3.92×10^5
$m = 20$	$\mathbf{1.29 \times 10^5}$	1.76×10^6	$\mathbf{9.75 \times 10^4}$	1.69×10^6	$\mathbf{1.02 \times 10^5}$	1.75×10^6
Mean	$\mathbf{4.97 \times 10^4}$	7.78×10^5	$\mathbf{4.03 \times 10^4}$	7.54×10^5	$\mathbf{6.03 \times 10^4}$	7.50×10^5

Table 7. C-metric values of QLEA and QLEA-RE under SDST-50 constraint.

Type	①		②		③	
	QLEA	**QLEA-RE**	**QLEA**	**QLEA-RE**	**QLEA**	**QLEA-RE**
$F = 2$	**0.86**	0.00	**0.83**	0.00	**0.55**	0.00
$F = 4$	**0.74**	0.00	**0.61**	0.00	**0.53**	0.00
$F = 6$	**0.89**	0.00	**0.70**	0.00	**0.58**	0.00
Mean	**0.83**	0.00	**0.71**	0.00	**0.55**	0.00
$n = 20$	**0.89**	0.00	**0.73**	0.00	**0.50**	0.00
$n = 50$	**0.93**	0.00	**0.89**	0.00	**0.41**	0.00
$n = 100$	**0.56**	0.00	**0.44**	0.00	**0.72**	0.00
$n = 200$	**1.00**	0.00	**0.83**	0.00	**0.58**	0.00
$n = 500$	**1.00**	0.00	**0.95**	0.00	**0.83**	0.00
Mean	**0.87**	0.00	**0.77**	0.00	**0.61**	0.00
$m = 5$	**0.78**	0.00	**0.58**	0.00	**0.78**	0.00
$m = 10$	**0.86**	0.00	**0.67**	0.00	**0.54**	0.00
$m = 20$	**0.83**	0.00	**0.86**	0.00	**0.39**	0.00
Mean	**0.82**	0.00	**0.70**	0.00	**0.57**	0.00

Table 8. C-metric values of QLEA and QLEA-RE under SDST-100 constraint.

Type	①		②		③	
	QLEA	**QLEA-RE**	**QLEA**	**QLEA-RE**	**QLEA**	**QLEA-RE**
$F = 2$	**1.00**	0.00	**0.50**	0.00	**0.73**	0.00
$F = 4$	**0.86**	0.00	**0.91**	0.00	**0.65**	0.00
$F = 6$	**0.82**	0.00	**0.82**	0.00	**0.61**	0.00
Mean	**0.89**	0.00	**0.74**	0.00	**0.66**	0.00
$n = 20$	**1.00**	0.00	**0.78**	0.00	**0.85**	0.00
$n = 50$	**1.00**	0.00	**0.72**	0.00	**0.85**	0.00
$n = 100$	**0.72**	0.00	**0.67**	0.00	**0.28**	0.00
$n = 200$	**0.83**	0.00	**0.83**	0.00	**0.67**	0.00
$n = 500$	**0.90**	0.00	**0.92**	0.00	**0.81**	0.00
Mean	**0.89**	0.00	**0.78**	0.00	**0.69**	0.00
$m = 5$	**0.89**	0.00	**0.83**	0.00	**0.70**	0.00
$m = 10$	**0.96**	0.00	**0.67**	0.00	**0.54**	0.00
$m = 20$	**0.83**	0.00	**0.75**	0.00	**0.75**	0.00
Mean	**0.89**	0.00	**0.75**	0.00	**0.66**	0.00

The results from Tables 5–8 indicate that across all instances under various sequence-dependent setup times and no-idle constraint scenarios, the IGD values obtained by the QLEA are consistently lower than those obtained by the QLEA-RE. This demonstrates that the QLEA, which includes the Q-Learning-based neighborhood search strategy, achieves better convergence and distribution of solutions. Additionally, for all instances under different no-idle constraint scenarios, the C(QLEA, QLEA-RE) values are consistently higher than the C(QLEA-RE, QLEA) values, indicating that the majority of solutions from the QLEA-RE are dominated by the non-dominated solutions of the QLEA. This highlights the superior performance of the QLEA's solutions in terms of dominance and overall effectiveness in the problem space.

To further illustrate the performance superiority of the QLEA that includes the Q-Learning-based neighborhood search strategy, a variance analysis was conducted on the test results of the QLEA and QLEA-RE across different instances. This analysis provides a more intuitive representation of the performance differences between the two algorithms. Figure 9 displays the mean changes and the confidence intervals under the 95% confidence level using Turkey's HSD test. The figure shows that the introduction of the Q-Learning neighborhood search strategy has a significant positive impact on the performance of the QLEA, thereby further validating that the Q-Learning neighborhood search strategy contributes to enhancing the algorithm's performance.

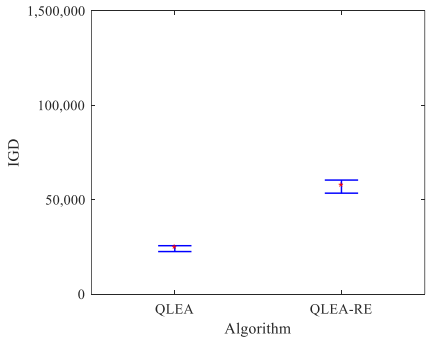

Figure 9. ANOVA of the QLEA and QLEA-RE.

4.5. Comparison with Related Algorithms

To validate the performance of the QLEA compared to other algorithms, a series of comparative experiments were conducted against the QLEA, an Improved Non-Dominated Sorting Genetic Algorithm II (INSGA-II) [30], a Multi-Objective Hybrid Iterated Greedy (MOHIG) algorithm [31], and a Competitive Memetic Algorithm (CMA) [32]. The ter-

mination time for all comparative algorithms was set to $n \times m \times f \times C$ milliseconds, with a value of C set to 5. The delivery deadlines for workpieces were designated as $d_j = P_j \times (1 + rand(0,1))$, where P_j represents the total processing time of job j across all machines. Each algorithm was run independently ten times on test instances. Tables 9 and 10 display the IGD results for QLEA and the three comparative algorithms for all instances grouped by factory, job, and machine settings under SDST-50 and SDST-100, respectively. This comparison helps to highlight the effectiveness of the QLEA in managing complex scheduling tasks and its ability to efficiently navigate and optimize the search space compared to other established algorithms in the field. In addition, Table 11 shows that the preparation time of the QLEA and comparison algorithm in scenario ① is the running time of SDST-50 for solving each instance.

As can be seen from Tables 9 and 10, regardless of different no-idle constraint scenarios under any of the SDSTs, the results obtained by the QLEA proposed in this paper are superior to other comparison algorithms in solving most; especially with the increase in the number of jobs, QLEA shows more obvious advantages. Specifically, when the sequence-dependent setup time is SDST-50, the IGD means of the QLEA in all the three no-idle scenarios are 1.25×10^4, 7.92×10^3, and 5.16×10^3, which are much smaller than those of INSGA-II, MOHIG and CMA. When the sequence-dependent setup time is SDST-100, the IGD means of the QLEA in all three no-idle scenarios are 9.12×10^3, 6.92×10^3, and 6.62×10^3, which are much smaller than those of INSGA-II, MOHIG, and CMA, which shows that the QLEA proposed in this paper has certain advantages, which may be due to the improved initialization strategy and the initialization method combined with random generation so that the initial solutions are evenly distributed in different regions, which is conducive to improving the quality and diversity of solutions. Secondly, the discrete population updating mechanism is adopted to enable the algorithm to search directly in the discrete scheduling space, and the variable neighborhood local search based on Q-Learning is designed, which greatly enhances the local development ability. In addition, according to different optimization objectives, different neighborhood search strategies are proposed to avoid the local optimal problem caused by a single search strategy and further improve the quality of understanding and solution accuracy. From the running time of the QLEA and the comparison algorithm in solving each instance in Table 11, it can be directly seen that the QLEA is significantly faster than INSGA-II, MOHIG, and CMA, further demonstrating the superiority of the QLEA. Meanwhile, the proposed QLEA can also improve the competitiveness of enterprises and increase economic benefits.

Figure 10 illustrates the non-dominated solution results of the QLEA compared to the three other algorithms (INSGA-II, MOHIG, and CMA) for a scenario involving 100 jobs and 20 machines under the sequence-dependent setup time (SDST-50) with the third type of no-idle constraint. From the diagram, it is evident that the QLEA's results are notably superior. Furthermore, the distribution of the QLEA's population is significantly better than that of the three comparative algorithms.

A variance analysis of the test results of the QLEA, INSGA-II, MOHIG, and CMA across different instances was conducted to further verify the differences in performance among these algorithms. Figure 11 displays the mean change lines and the confidence intervals under a 95% confidence level using Turkey's HSD test. It is observable from Figure 11 that the IGD values achieved by the QLEA for instances with different sequence-dependent setup times are consistently lower than those achieved by INSGA-II, MOHIG, and CMA. This further underscores that the QLEA produces higher-quality results and possesses distinct advantages, validating its effectiveness and efficiency in tackling complex scheduling problems with various constraints and requirements.

Table 9. IGD values of different algorithms under SDST-50 constraints.

Type	①				②				③			
	INSGA-II	**MOHIG**	**CMA**	**QLEA**	**INSGA-II**	**MOHIG**	**CMA**	**QLEA**	**INSGA-II**	**MOHIG**	**CMA**	**QLEA**
$F = 2$	3.87×10^4	2.23×10^4	1.14×10^5	$\mathbf{1.44 \times 10^4}$	1.67×10^4	1.77×10^4	1.14×10^5	$\mathbf{8.93 \times 10^3}$	2.74×10^4	1.43×10^4	9.96×10^4	$\mathbf{5.77 \times 10^3}$
$F = 4$	2.48×10^4	1.47×10^4	6.93×10^4	$\mathbf{9.71 \times 10^3}$	2.19×10^4	1.26×10^4	8.54×10^4	$\mathbf{4.83 \times 10^3}$	1.28×10^4	8.60×10^3	6.59×10^4	$\mathbf{2.91 \times 10^3}$
$F = 6$	1.74×10^4	1.49×10^4	6.26×10^4	$\mathbf{5.09 \times 10^3}$	1.33×10^4	9.76×10^3	6.99×10^4	$\mathbf{4.98 \times 10^3}$	1.45×10^4	8.05×10^3	6.37×10^4	$\mathbf{3.35 \times 10^3}$
Mean	2.70×10^4	1.73×10^4	8.20×10^4	$\mathbf{9.73 \times 10^3}$	1.73×10^4	1.34×10^4	8.98×10^4	$\mathbf{6.25 \times 10^3}$	1.82×10^4	1.03×10^4	7.64×10^4	$\mathbf{4.01 \times 10^3}$
$n = 20$	3.29×10^3	2.45×10^3	1.71×10^3	$\mathbf{8.82 \times 10^2}$	2.90×10^3	1.96×10^3	2.46×10^3	$\mathbf{6.57 \times 10^2}$	3.65×10^3	3.34×10^3	2.31×10^3	$\mathbf{8.71 \times 10^2}$
$n = 50$	5.58×10^3	5.39×10^3	8.97×10^3	$\mathbf{1.57 \times 10^3}$	3.87×10^3	5.14×10^3	7.66×10^3	$\mathbf{1.80 \times 10^3}$	6.21×10^3	5.65×10^3	1.25×10^4	$\mathbf{6.90 \times 10^2}$
$n = 100$	3.12×10^4	8.39×10^3	8.10×10^4	$\mathbf{4.88 \times 10^3}$	1.98×10^4	9.60×10^3	8.36×10^4	$\mathbf{2.94 \times 10^3}$	1.41×10^4	1.06×10^4	6.49×10^4	$\mathbf{1.37 \times 10^3}$
$n = 200$	8.83×10^4	7.09×10^4	3.14×10^5	$\mathbf{4.26 \times 10^4}$	5.52×10^4	4.84×10^4	3.52×10^5	$\mathbf{2.63 \times 10^4}$	6.43×10^4	2.74×10^4	3.01×10^5	$\mathbf{1.77 \times 10^4}$
$n = 500$	9.96×10^4	8.76×10^4	4.54×10^5	$\mathbf{4.26 \times 10^4}$	5.53×10^4	4.46×10^4	3.24×10^5	$\mathbf{2.62 \times 10^4}$	6.35×10^4	2.78×10^4	3.23×10^5	$\mathbf{1.75 \times 10^4}$
Mean	4.56×10^4	3.49×10^4	1.72×10^5	$\mathbf{1.85 \times 10^4}$	2.74×10^4	2.19×10^4	1.54×10^5	$\mathbf{6.98 \times 10^3}$	3.04×10^4	1.50×10^4	1.41×10^3	$\mathbf{7.63 \times 10^3}$
$m = 5$	7.65×10^3	2.83×10^3	8.81×10^3	$\mathbf{2.03 \times 10^3}$	2.60×10^3	2.89×10^3	4.20×10^3	$\mathbf{1.08 \times 10^3}$	2.21×10^3	2.04×10^3	3.39×10^3	$\mathbf{7.78 \times 10^2}$
$m = 10$	2.67×10^4	1.79×10^4	6.48×10^4	$\mathbf{8.03 \times 10^3}$	1.32×10^4	1.41×10^4	6.47×10^4	$\mathbf{4.68 \times 10^3}$	2.13×10^4	9.63×10^3	5.25×10^4	$\mathbf{6.16 \times 10^3}$
$m = 20$	4.18×10^4	2.76×10^4	1.54×10^5	$\mathbf{1.73 \times 10^4}$	3.23×10^4	1.96×10^4	1.79×10^5	$\mathbf{1.17 \times 10^4}$	2.71×10^4	1.72×10^4	1.55×10^5	$\mathbf{4.29 \times 10^3}$
Mean	2.54×10^4	1.61×10^4	7.59×10^4	$\mathbf{9.12 \times 10^3}$	1.60×10^4	1.25×10^4	8.26×10^4	$\mathbf{5.82 \times 10^3}$	1.69×10^4	9.62×10^3	7.03×10^4	$\mathbf{3.74 \times 10^3}$

Table 10. IGD values of different algorithms under SDST-100 constraints.

Type	①				②				③			
	INSGA-II	**MOHIG**	**CMA**	**QLEA**	**INSGA-II**	**MOHIG**	**CMA**	**QLEA**	**INSGA-II**	**MOHIG**	**CMA**	**QLEA**
$F = 2$	2.98×10^4	3.41×10^4	2.03×10^5	$\mathbf{1.03 \times 10^4}$	3.56×10^4	1.66×10^4	1.74×10^5	$\mathbf{9.56 \times 10^3}$	4.16×10^4	1.58×10^4	1.91×10^5	$\mathbf{4.20 \times 10^3}$
$F = 4$	3.37×10^4	2.06×10^4	1.36×10^5	$\mathbf{6.03 \times 10^3}$	3.25×10^4	1.57×10^4	1.28×10^5	$\mathbf{3.83 \times 10^3}$	1.70×10^4	1.35×10^4	9.80×10^4	$\mathbf{8.17 \times 10^3}$
$F = 6$	2.24×10^4	1.42×10^4	9.83×10^4	$\mathbf{5.16 \times 10^3}$	2.47×10^4	1.60×10^4	9.88×10^4	$\mathbf{4.31 \times 10^3}$	2.82×10^4	1.30×10^4	1.02×10^5	$\mathbf{3.52 \times 10^3}$
Mean	2.86×10^4	2.30×10^4	1.46×10^5	$\mathbf{7.16 \times 10^3}$	3.09×10^4	1.61×10^4	1.34×10^5	$\mathbf{5.90 \times 10^3}$	2.89×10^4	1.41×10^4	1.30×10^5	$\mathbf{5.30 \times 10^3}$
$n = 20$	5.02×10^3	5.05×10^3	5.86×10^3	$\mathbf{9.59 \times 10^2}$	4.28×10^3	3.22×10^3	2.23×10^3	$\mathbf{8.41 \times 10^2}$	5.92×10^3	5.22×10^3	4.48×10^3	$\mathbf{1.15 \times 10^3}$
$n = 50$	1.20×10^4	1.11×10^4	1.57×10^4	$\mathbf{1.85 \times 10^3}$	7.18×10^3	6.63×10^3	1.41×10^4	$\mathbf{2.05 \times 10^3}$	7.09×10^3	7.44×10^3	1.44×10^4	$\mathbf{1.17 \times 10^3}$
$n = 100$	1.92×10^4	6.80×10^3	1.01×10^5	$\mathbf{2.98 \times 10^3}$	3.37×10^4	2.06×10^4	1.00×10^5	$\mathbf{6.58 \times 10^3}$	3.26×10^4	1.65×10^4	8.66×10^4	$\mathbf{3.06 \times 10^3}$
$n = 200$	1.03×10^5	9.20×10^4	6.18×10^5	$\mathbf{3.07 \times 10^4}$	1.03×10^5	4.29×10^4	5.59×10^5	$\mathbf{1.82 \times 10^4}$	9.06×10^4	3.39×10^4	5.59×10^5	$\mathbf{2.11 \times 10^4}$
$n = 500$	1.26×10^5	9.24×10^4	6.36×10^5	$\mathbf{3.08 \times 10^4}$	1.23×10^5	4.46×10^4	5.96×10^5	$\mathbf{1.84 \times 10^4}$	9.43×10^4	3.34×10^4	5.35×10^5	$\mathbf{2.45 \times 10^4}$
Mean	5.30×10^4	4.15×10^4	2.75×10^5	$\mathbf{1.35 \times 10^4}$	5.42×10^4	2.36×10^4	2.54×10^5	$\mathbf{9.21 \times 10^3}$	4.61×10^4	1.93×10^4	2.40×10^5	$\mathbf{1.02 \times 10^4}$
$m = 5$	5.08×10^3	2.47×10^3	1.05×10^4	$\mathbf{9.18 \times 10^2}$	5.02×10^3	4.78×10^3	9.10×10^3	$\mathbf{1.74 \times 10^3}$	3.57×10^3	1.69×10^3	8.04×10^3	$\mathbf{3.80 \times 10^2}$
$m = 10$	2.36×10^4	2.11×10^4	9.12×10^4	$\mathbf{6.90 \times 10^3}$	2.09×10^4	1.83×10^4	9.82×10^4	$\mathbf{1.11 \times 10^4}$	2.92×10^4	1.11×10^4	1.00×10^5	$\mathbf{6.68 \times 10^3}$
$m = 20$	5.14×10^4	4.03×10^4	$3{,}02 \times 10^5$	$\mathbf{1.21 \times 10^4}$	6.04×10^4	2.24×10^4	2.62×10^5	$\mathbf{3.83 \times 10^3}$	4.77×10^4	2.65×10^4	2.52×10^5	$\mathbf{7.60 \times 10^3}$
Mean	2.67×10^4	2.13×10^4	1.35×10^5	$\mathbf{6.64 \times 10^3}$	2.88×10^4	1.52×10^4	1.23×10^5	$\mathbf{5.56 \times 10^3}$	2.68×10^4	1.31×10^4	1.20×10^5	$\mathbf{4.89 \times 10^3}$

Table 11. QLEA and the comparison algorithm solve the running time of each instance.

$N \times m \times F$	① Time(s)			
	QLEA	**CMA**	**MOHIG**	**INSGA-II**
20 × 5 × 2	**6.73**	8.56	10.13	13.32
20 × 10 × 2	**30.12**	33.32	45.31	52.13
20 × 20 × 2	**61.34**	78.56	80.13	98.13
50 × 5 × 2	**120.32**	134.54	156.43	165.43
50 × 10 × 2	**300.13**	312.44	321.45	395.46
50 × 20 × 2	**600.45**	623.45	650.46	664.24
100 × 5 × 2	**612.26**	623.13	643.12	654.16
100 × 10 × 2	**1203.13**	1300.24	1401.16	1403.61
100 × 20 × 2	**2403.42**	2503.13	2589.16	2601.87
200 × 5 × 2	**1826.16**	1903.42	1964.17	1980.16
200 × 10 × 2	**3603.11**	3645.16	3700.16	3761.18
200 × 20 × 2	**7236.72**	7346.79	7463.98	7501.46
500 × 5 × 2	**8603.46**	8800.47	8934.23	9103.1
500 × 10 × 2	**14,403.36**	15,900.16	16,901.00	17031.45
500 × 20 × 2	**28,803.47**	29,900.13	29,323.45	30,000.48
20 × 5 × 4	**8.63**	9.42	11.36	13.58
20 × 10 × 4	**62.31**	66.56	72.42	80.12
20 × 20 × 4	**121.13**	134.56	144.56	158.96
50 × 5 × 4	**302.46**	313.46	345.41	364.15
50 × 10 × 4	**600.16**	621.13	643.12	645.19
50 × 20 × 4	**1264.89**	1299.46	1346.49	1365.54
100 × 5 × 4	**956.65**	970.36	980.35	990.62
100 × 10 × 4	**1859.25**	1899.35	1963.56	1998.22
100 × 20 × 4	**3645.65**	3752.63	3856.21	3956.56
200 × 5 × 4	**2563.52**	2630.26	2703.52	2865.60
200 × 10 × 4	**5463.56**	5632.32	5783.26	5896.62
200 × 20 × 4	**10,882.23**	10,956.26	11,231.23	11,323.23
500 × 5 × 4	**11,230.45**	11,563.89	12,032.26	12,122.45
500 × 10 × 4	**21,648.56**	22,659.56	22,989.64	23,023.78
500 × 20 × 4	**43,256.78**	44,332.41	45,623.13	46,563.56
20 × 5 × 6	**13.72**	14.38	15.86	16.79
20 × 10 × 6	**63.45**	67.56	68.89	70.23
20 × 20 × 6	**184.56**	189.56	190.45	191.48
50 × 5 × 6	**612.35**	622.45	664.36	674.12
50 × 10 × 6	**1235.56**	1239.45	1326.56	1378.93
50 × 20 × 6	**2456.89**	2536.45	2645.78	2899.56
100 × 5 × 6	**1879.78**	1953.78	1996.46	1999.47
100 × 10 × 6	**3648.89**	3789.47	3812.13	3978.56
100 × 20 × 6	**7245.63**	7356.13	7456.17	7569.68
200 × 5 × 6	**3648.78**	3885.69	3956.54	4125.68
200 × 10 × 6	**7245.32**	7412.36	7546.89	7784.59
200 × 20 × 6	**14,562.23**	15,462.79	14,896.47	15,368.98
500 × 5 × 6	**19,835.65**	20,315.58	21,653.48	22,345.65
500 × 10 × 6	**28,456.89**	29,784.56	31,214.15	32,146.89
500 × 20 × 6	**57,689.49**	58,963.45	64,523.19	65,894.46

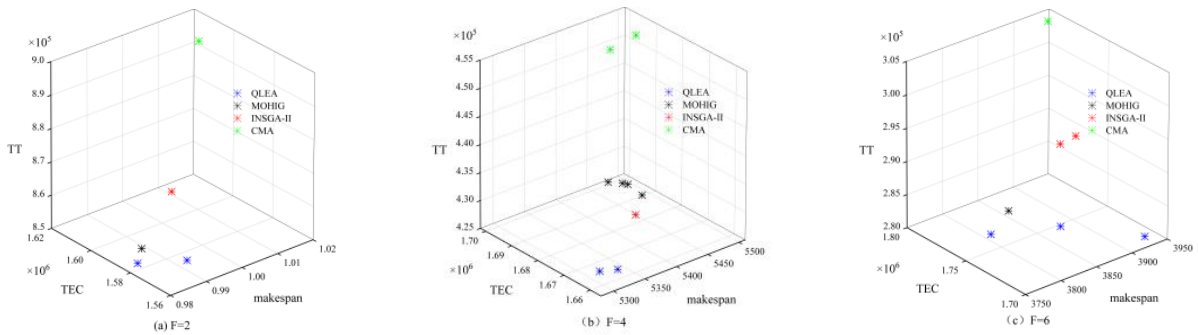

Figure 10. Comparison of non-dominated solution algorithms for 100 jobs and 20 machines in three different factories.

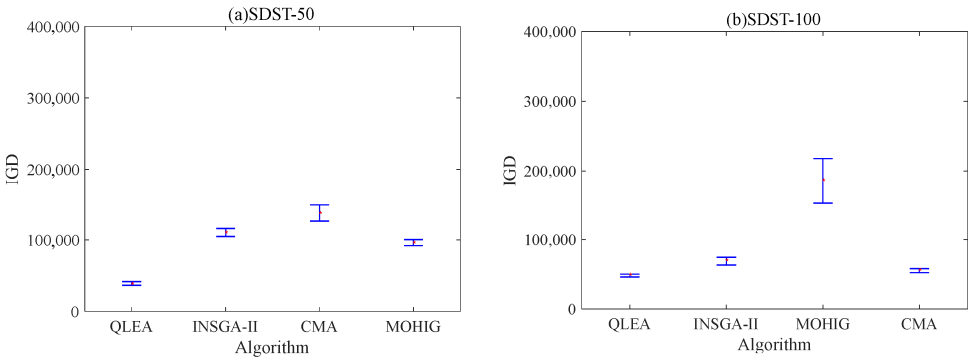

Figure 11. Variance analysis of the QLEA and comparison algorithms under different SDSTs.

4.6. Non-Parametric Statistical Test

To make the comparison results more convincing and substantiated, a statistical analysis was performed on the results of different algorithms across various sequence-dependent setup times as detailed in Tables 9 and 10. To test the significant differences between these algorithms, a non-parametric statistical method, the Wilcoxon signed-rank test [33], was utilized. The significance level for the tests was set at 5%. The outcomes of these tests are meticulously recorded in Table 12.

Table 12. Wilcoxon test results of the QLEA comparison algorithm.

		SDST-50				SDST-100	
F	**QLEA VS.**	***p*-Value**	***R***	***F***	**QLEA VS.**	***p*-Value**	***R***
2	INSGA-II	1.60×10^{-9}	+	2	INSGA-II	8.31×10^{-7}	+
	MOHIG	1.85×10^{-6}	+		MOHIG	1.24×10^{-6}	+
	CMA	1.18×10^{-9}	+		CMA	2.36×10^{-6}	+
4	INSGA-II	1.61×10^{-9}	+	4	INSGA-II	7.64×10^{-9}	+
	MOHIG	4.83×10^{-5}	+		MOHIG	8.44×10^{-6}	+
	CMA	2.37×10^{-10}	+		CMA	8.33×10^{-7}	+
6	INSGA-II	2.34×10^{-9}	+	6	INSGA-II	2.34×10^{-10}	+
	MOHIG	1.55×10^{-5}	+		MOHIG	4.47×10^{-6}	+
	CMA	4.65×10^{-10}	+		CMA	2.35×10^{-10}	+

Table 12 shows the *p*-values calculated by INSGA-II, MOHIG, and CMA in the Wilcoxon signed-rank test of the QLEA. It can be seen from the results of Table 12 that the *p*-value of all test examples is much less than 5%, indicating that there are significant differences between the QLEA and the other three comparison algorithms, and the QLEA is significantly better.

5. Conclusions

For the characteristics of the DMNIPFSP/SDST, a QLEA is proposed. First, to enhance the quality and diversity of initial solutions, two improved initialization strategies are introduced. Second, a novel population updating mechanism is designed based on the characteristics of the problem addressed, balancing the global exploration and local exploitation capabilities of the algorithm. Additionally, a Q-Learning-based variable neighborhood local search is utilized to refine non-dominated solutions, thereby guiding the population evolution. Finally, simulation experiments are conducted on instances under two different sequence-dependent setup time scenarios and three types of no-idle constraint conditions, comparing the results with those from INSGA-II, MOHIG, and CMA. The results show that the QLEA has significant advantages in solving the multi-objective

DMNIPFSP/SDST, and this method can also improve the competitiveness of enterprises and increase economic benefits.

For future work, we can develop higher-performance algorithms based on the QLEA and the characteristics of the problem to address other complex engineering optimization challenges, such as vehicle scheduling [34,35], logistics scheduling [36–38], distributed scheduling [39–41], and flexible job scheduling [42–44]. These are the directions we will strive towards in the future.

Author Contributions: Conceptualization, F.Z.; methodology, F.Z.; software, F.Z.; validation, J.C., F.Z. and F.Z.; formal analysis, J.C. investigation, F.Z.; resources, F.Z.; data curation, F.Z.; writing—original draft preparation, J.C.; writing—review and editing, F.Z.; visualization, F.Z.; supervision, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to the need for further analysis in future studies. The researchers plan to conduct additional analyses, and thus, the data are temporarily withheld to preserve the novelty of subsequent research findings.

Conflicts of Interest: All authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Zhu, N.; Zhao, F.; Wang, L. A discrete learning fruit fly algorithm based on knowledge for the distributed no-wait flow shop scheduling with due windows. *Expert Syst. Appl.* **2022**, *198*, 116921. [CrossRef]
2. Zhang, X.; Yuan, J.; Chen, X. Development of an improved water cycle algorithm for solving an energy-efficient disassembly-line balancing problem. *Processes* **2022**, *10*, 1908. [CrossRef]
3. Naderi, B.; Ruiz, R. The distributed permutation flowshop scheduling problem. *Comput. Oper. Res.* **2010**, *37*, 754–768. [CrossRef]
4. Huang, J.P.; Pan, Q.K.; Gao, L. An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. *Swarm Evol. Comput.* **2020**, *59*, 100742. [CrossRef]
5. Almeida, F.S.D.; Nagano, M.S. Heuristics to optimize total completion time subject to makespan in no-wait flow shops with sequence-dependent setup times. *J. Oper. Res. Soc.* **2023**, *74*, 362–373. [CrossRef]
6. Abreu, L.R.; Nagano, M.S. A new hybridization of adaptive large neighborhood search with constraint programming for open shop scheduling with sequence-dependent setup times. *Comput. Ind. Eng.* **2022**, *168*, 108128. [CrossRef]
7. Guo, H.-W.; Sang, H.-Y.; Zhang, B.; Meng, L.-L.; Liu, L.-L. An effective metaheuristic with a differential flight strategy for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. *Knowl. Based Syst.* **2022**, *242*, 108328. [CrossRef]
8. Miyata, H.H.; Nagano, M.S. An iterated greedy algorithm for distributed blocking flow shop with setup times and maintenance operations to minimize makespan. *Comput. Ind. Eng.* **2022**, *171*, 108366. [CrossRef]
9. Karabulut, K.; Öztop, H.; Kizilay, D.; Tasgetiren, M.F.; Kandiller, L. An evolution strategy approach for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. *Comput. Oper. Res.* **2022**, *142*, 105733. [CrossRef]
10. Lu, C.; Zhang, B.; Gao, L.; Yi, J.; Mou, J. A knowledge-based multiobjective memetic algorithm for green job shop scheduling with variable machining speeds. *IEEE Syst. J.* **2021**, *16*, 844–855. [CrossRef]
11. Vallada, E.; Ruiz, R.; Minella, G. Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Comput. Oper. Res.* **2008**, *35*, 1350–1373. [CrossRef]
12. Cheng, C.-Y.; Ying, K.-C.; Chen, H.-H.; Lu, H.-S. Minimising makespan in distributed mixed no-idle flowshops. *Int. J. Prod. Res.* **2019**, *57*, 48–60. [CrossRef]
13. Wang, L.; Pan, Z.; Wang, J. A review of reinforcement learning based intelligent optimization for manufacturing scheduling. *Complex Syst. Model. Simul.* **2021**, *1*, 257–270. [CrossRef]
14. Kool, W.; van Hoof, H.; Welling, M. Attention, Learn to Solve Routing Problems. *arXiv* **2018**, arXiv:1803.08475. [CrossRef]
15. Bello, I.; Pham, H.; Le, Q.V.; Norouzi, M.; Bengio, S. Neural combinatorial optimization with reinforcement learning. *arXiv* **2017**, arXiv:1611.09940. [CrossRef]

16. Chen, R.; Yang, B.; Li, S.; Wang, S. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. *Comput. Ind. Eng.* **2020**, *149*, 106778. [CrossRef]
17. Wang, Y.F. Adaptive job shop scheduling strategy based on weighted Q-learning algorithm. *J. Intell. Manuf.* **2020**, *31*, 417–432. [CrossRef]
18. Luo, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Appl. Soft Comput.* **2020**, *91*, 106208. [CrossRef]
19. 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]
20. Wang, J.; Lei, D.; Cai, J. An adaptive artificial bee colony with reinforcement learning for distributed threestage assembly scheduling with maintenance. *Appl. Soft Comput.* **2022**, *117*, 108371. [CrossRef]
21. Zhao, C.; Wu, L.H.; Zuo, C.L.; Zhang, H.Q. An improved fruit fly optimization algorithm with Q-learning for solving distributed permutation flow shop scheduling problems. *Complex Intell. Syst.* **2024**, *10*, 5965–5988. [CrossRef]
22. Pan, Q.-K.; Gao, L.; Xin-Yu, L.; Jose, F.M. Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem. *Appl. Soft Comput.* **2019**, *81*, 105492. [CrossRef]
23. Liu, J.; Reeves, C.R. Constructive and composite heuristic solutions to the P//$\sum$ Ci scheduling problem. *Eur. J. Oper. Res.* **2001**, *132*, 439–452. [CrossRef]
24. Zitzler, E.; Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.* **1999**, *3*, 257–271. [CrossRef]
25. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
26. Ruiz, R.; Maroto, C.; Alcaraz, J. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *Eur. J. Oper. Res.* **2005**, *165*, 34–54. [CrossRef]
27. Ruiz, R.; Stutzle, T. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *Eur. J. Oper. Res.* **2008**, *187*, 1143–1159. [CrossRef]
28. 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]
29. Montgomery, D.C. *Design and Analysis of Experiments*; John Wiley & Sons: Hoboken, NJ, USA, 2017.
30. Zeng, Q.-Q.; Li, J.-Q.; Li, R.-H.; Huang, T.-H.; Han, Y.-Y.; Sang, H.-Y. Improved NSGA-II for energy-efficient distributed no-wait flow-shop with sequence-dependent setup time. *Complex Intell. Syst.* **2023**, *9*, 825–849. [CrossRef]
31. Lu, C.; Liu, Q.; Zhang, B.; Yin, L. A pareto-based hybrid iterated greedy algorithm for energy-efficient scheduling of distributed hybrid flowshop. *Expert Syst. Appl.* **2022**, *204*, 117555. [CrossRef]
32. Deng, J.; Wang, L. A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem. *Swarm Evol. Comput.* **2017**, *32*, 121–131. [CrossRef]
33. Wilcoxon, F. *Individual Comparisons by Ranking Methods*; Springer: New York, NY, USA, 1992; pp. 196–202.
34. Yu, X.Y.; Zhu, N.; Ma, Y.M. Research on the scheduling problem of public transportation vehicles considering abnormal train numbers. *Syst. Eng. Theory Pract.* **2023**, *43*, 910–928.
35. Tanng, J.Q.; Qi, C.; Wang, H.W. Multi warehouse order splitting with time windows and joint optimization method for heterogeneous vehicle paths. *Syst. Eng. Theory Pract.* **2023**, *43*, 1446–1464.
36. Liu, Z.S.; Cheng, Y.A.; Zuo, X.Q. Research on Optimization of Multi tank Depot and Multi trip Gas Station Distribution Considering Tank Capacity Limitation. *Syst. Eng. Theory Pract.* **2023**, *43*, 1232–1250.
37. Li, J.B.; Song, X.R.; Guo, J.S. Research on Multi warehouse Collaborative Product Selection Strategy in Manufacturing Production Logistics. *Syst. Eng. Theory Pract.* **2023**, *43*, 1736–1749.
38. Xiangi, C.K.; Wu, Z.B.; Xu, J.P. A Crowdsourcing Logistics Task Allocation Model for Dual Agent Collaborative Learning. *Syst. Eng. Theory Pract.* **2023**, *43*, 1978–1995.
39. Fathollahi-Fard, A.M.; Lyne, W.; Ouassima, A. A distributed permutation flow-shop considering sustainability criteria and real-time scheduling. *J. Ind. Inf. Integr.* **2024**, *39*, 100598. [CrossRef]
40. Luo, C.; Gong, W.Y.; Li, R. Problem-specific knowledge MOEA/D for energy-efficient scheduling of distributed permutation flow shop in heterogeneous factories. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106454. [CrossRef]
41. Song, H.B.; Yang, Y.H.; Lin, J. An effective hyper heuristic-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem. *Appl. Soft Comput.* **2023**, *135*, 110022. [CrossRef]
42. Guo, X.H.; Liu, J.H.; Wang, Y. An improved genetic programming hyper-heuristic for the dynamic flexible job shop scheduling problem with reconfigurable manufacturing cells. *J. Manuf. Syst.* **2024**, *74*, 252–263. [CrossRef]

43. Fan, C.S.; Wang, W.T.; Tian, J. Flexible job shop scheduling with stochastic machine breakdowns by an improved tuna swarm optimization algorithm. *J. Manuf. Syst.* **2024**, *74*, 180–197. [CrossRef]
44. Wei, S.P.; Tang, H.T.; Li, X.X. An improved memetic algorithm for multi-objective resource-constrained flexible job shop inverse scheduling problem: An application for machining workshop. *J. Manuf. Syst.* **2024**, *74*, 264–290. [CrossRef]