sustainability

MDPI

Article

Hybrid Flow-Shop Scheduling Problems with Missing and Re-Entrant Operations Considering Process Scheduling and Production of Energy Consumption

Hongtao Tang [1,*], Jiahao Zhou [1], Yiping Shao [1] and Zhixiong Yang [2]

[1] Institute of Industrial Engineering, College of Mechanical Engineering, Zhejiang University of Technology, Hangzhou 310023, China; 2112002049@zjut.edu.cn (J.Z.); syp123gh@zjut.edu.cn (Y.S.)
[2] Jiaxing Sudoku Bridge Technology Co., Ltd., Jiaxing 314599, China; fleetyang@gmx.com
* Correspondence: tanght@zjut.edu.cn

Abstract: A hybrid flow shop scheduling model with missing and re-entrant operations was designed to minimize the maximum completion time and the reduction in energy consumption. The proposed dual-population genetic algorithm was enhanced with a range of improvements, which include the design of a three-layer gene coding method, hierarchical crossover and mutation techniques, and the development of an adaptive operator that considered gene similarity and chromosome fitness values. The optimal and worst individuals were exchanged between the two subpopulations to improve the exploration ability of the algorithm. An orthogonal experiment was performed to obtain the optimal horizontal parameter set of the algorithm. Furthermore, an experiment was conducted to compare the proposed algorithm with a basic genetic algorithm, particle swarm optimization algorithm, and ant colony optimization, which were all performed on the same scale. The experimental results show that the fitness value of the proposed algorithm is above 15% stronger than the other 4 algorithms on a small scale, and was more than 10% stronger than the other 4 algorithms on a medium and large scale. Under the condition close to the actual scale, the results of ten repeated calculations showed that the proposed algorithm had higher robustness.

Keywords: hybrid flow-shop scheduling problem; missing operation; re-entry; genetic algorithm; energy consumption

check for updates

Citation: Tang, H.; Zhou, J.; Shao, Y.; Yang, Z. Hybrid Flow-Shop Scheduling Problems with Missing and Re-Entrant Operations Considering Process Scheduling and Production of Energy Consumption. *Sustainability* **2023**, *15*, 7982. https://doi.org/10.3390/su15107982

Academic Editor: Davide Settembre-Blundo

Received: 2 April 2023
Revised: 6 May 2023
Accepted: 11 May 2023
Published: 13 May 2023

1. Introduction

A hybrid flow-shop scheduling problem (HFSP) refers to a production shop with multiple processes and one or more parallel machines per process, arranged in an assembly line, and it is also referred to as a flexible flow-shop [1]. An HFSP is a combination of traditional flow shop scheduling and parallel-machine scheduling problems, and it has significant academic significance and application value. In other words, HFSP is a special kind of flow shop problem [2]. Furthermore, the HFSP can be extended to the re-entrant hybrid flow-shop problem (RHFS) [3] and hybrid flow shop scheduling problem with missing operations (HFSMO) [4], depending on the constraints of the specific process.

The "re-entrant shop" indicates that a workpiece may need to be processed several times on the same machine. In a study on semiconductor manufacturing processes, Kumar [3] proposed a third type of production system that differed from job shops and flow shops, wherein the workpiece needed to be repeatedly processed using certain machines or workstations at different stages of processing. This type of problem can be referred to as an RHFS problem. For the RHFS, most objective functions in the literature have focused on solving the makespan, with some studies examining objectives such as the weighted total completion time, cycle time, maximum delay time, total delay time, on-time completion rate, and early/delay penalties. For example, Chamnanlor et al. [5] aimed to maximize the system throughput by minimizing the maximum completion time based on the requirement of zero loss of work. Zhan et al. [6] discussed the production scheduling problem derived from a real rotor workshop and solved it with a minimum delay. Mousavis et al. [7] considered the setup time position-dependent learning effects and solved the maximum completion time and total delay as objective functions. Marichelvam et al. [8] used a hybrid monkey search algorithm to optimize the total flow time, considering the maximum completion time.

In recent years, with an increasing awareness of sustainability, more and more researchers have linked scheduling optimization to energy consumption and green costs. For example, Geng et al. [9] considered a right-shift operation and adjusted its start-up time to achieve the lowest possible energy cost by avoiding periods of high electricity prices. In addition to the objective function, a method for solving the RHFS is also the focus of this research. The HFSP has been proven to be NP-hard [10] and difficult to solve using traditional methods, whereas the RHFS, as an extension, is considered more difficult; therefore, researchers tend to adopt intelligent optimization algorithms to solve this problem. Geng et al. [11] developed an integrated scheduling model to minimize the maximum completion time, maximize the average agreement index, and design a hybrid NSGA-II based on the characteristics of the problem. Kun et al. [12] proposed a novel shuffle frog-jumping algorithm to handle the total delay and makespan. Wu et al. [13] discussed buffer sizes for batch machines and developed dosing methods, while proposing a greedy selection strategy to select machines for cold draw operations. Eskandadi et al. [14] generalized heuristics based on several basic scheduling rules and proposed a variable neighborhood search (VNS) to solve the problem of sequence-dependent setup times and unrelated parallel machines. Xu et al. [15] developed an improved moth–flame optimization algorithm to solve the green re-entrant hybrid flow shop scheduling problem. Qin et al. [16] proposed a rescheduling-based ant colony algorithm to solve the HFSP with uncertain processing time and introduced the concept of due date deviation to design a rolling-horizon-driven strategy that compressed the path of ant movement and reduced the cycle time in which to find a new solution. Nejad et al. [17] successfully traded off the relationship between process scheduling and production costs.

An HFSMO can be described as follows [18]: The sequence of processing machines in the production system is fixed, and the flow of workpieces to be processed is the same; however, some workpieces can skip process steps that do not need to be processed based on the characteristics of their own processes. This implies that not all workpieces have to undergo every process in a production system. Reading the literature [19] and analyzing it, we can find that "missing operations" are valuable for research. Although this type of problem is similar to traditional HFSP, it cannot simply be considered a unique case of a traditional HFSP due to the phenomenon of process omission. Setting the operation time of the skipped process stage to zero can result in the part being unable to skip the stage that does not require processing and being forced to wait for the process to complete. This not only leads to blockages but also delays in the start of the next process that the workpiece was originally intended for; in addition, this increases the idle time of the machine, which increases the completion time and affects the scheduling results. Therefore, research on HFSMO has important practical implications.

In 1985, Wittrock [4] first proposed an HFS containing missing operations in a practical production context based on an electronics processing plant and suggested that this scheduling model could be optimized using heuristic algorithms. Since then, several works of research have investigated this problem. Some researchers improved the existing intelligent algorithms. For example, Saravanan et al. [20] improved the simulated annealing (SA) algorithm and compared it with particle swarm optimization (PSO), demonstrating that the improved SA algorithm yielded better results and required a shorter computation time than with PSO. Li et al. [21] constructed a mathematical model of the HFSMO and then optimized it by improving the ABC algorithm to achieve certain results. Long et al. [22] proposed an improved GA which they applied to a steelmaking-continuous-casting production shop of a steel company. The final simulation experimental results indicated that the algorithm outperformed the other algorithms. Marichelvam et al. [23] proposed an improved hybrid genetic scatter search (IHGSS) algorithm by combining genetic operators and decentralized search algorithms, and they compared it with GA, a decentralized search algorithm, and the NEH heuristic algorithm to demonstrate the superior performance of the IHGSS algorithm. Saravanan et al. [24] fused GA with the SA algorithm and solved the problem model with an average delay as the scheduling objective using an improved GA.

In addition to the two aforementioned approaches, researchers have also designed new algorithms tailored to the characteristics of the problem. For instance, Lei et al. [25] devised a novel local search using a controlled deterioration algorithm to solve HFSMO. Dios et al. [26] constructed a problem model with completion time as the objective, performed a complexity analysis of HFSMO, and designed an effective heuristic algorithm. Siqueria et al. [27] proposed a new multi-objective VNS algorithm with which to solve problems with a maximum completion time and maximum total weighted delay as the optimization objectives.

After conducting an extensive review of the literature, it was found that research on RHFS is hot and has attracted many researchers. By contrast, the number of works in the literature on HFSMO is much less than the former. Both studies mainly focus on optimizing the maximum completion time, with a smaller number of researchers considering other objectives. Furthermore, the majority of existing studies do not consider important factors such as transport time, processing preparation time, and dynamic process jump constraints. Therefore, research on both needs to be further explored.

Our review of the literature revealed that the study of both the “re-entrant shop” and “workshops with missing operations” is a topic of great relevance and scientific value; however, to the best of the authors’ knowledge, no existing studies have considered them together. However, there are several hybrid flow shops in engineering that contain both re-entrant and missing operations. Therefore, this paper considers a combination of actual workshops, and then studies the new hybrid flow shop scheduling problem.

This study used the production organization of some products of an electric appliance manufacturing company as their research object and considered “re-entry” and “missing operations” together. This is referred to as the hybrid flow-shop scheduling problem with missing and re-entrant operations (HFS-MRO).

The main contributions to this paper are as follows:

- The integration of RHFS and HFSMO

To the best of the authors’ knowledge, this is the first study to consider “re-entry” and “missing operations” in a unified manner, bringing scheduling research closer to real production.

- The design of an improved dual-population genetic algorithm (IDPGA)

According to the processing characteristics, each chromosome of the proposed IDPGA adopts a three-layer gene coding mode and designs a “missing judgment vector” to integrate into the chromosome with which to determine the missing operations. To avoid the generation of non-feasible solutions, different crossover and mutation strategies are used for each coding layer, and adaptive operators based on gene similarity and chromosome fitness values are designed. The superiority and robustness of the algorithm are confirmed through the same-scale comparison experiment.

In this paper, “re-entry” and “missing operations” are considered together for the first time, and transportation and adjustment times are also taken into account, which could facilitate an in-depth study on hybrid flow shops. This could not only solve the actual factory problem but also provide a direction for follow-up researchers. Therefore, the research in this paper is of certain practical significance and scientific value.

The remainder of this paper is organized as follows: Section 2 describes the HFS-MRO and the mathematical model based on the problem characteristics. Section 3 presents the

IDPGA, and Section 4 presents the computational experiments. Finally, Section 5 concludes the paper and suggests future research directions.

2. Problem Description and Mathematical Model of HFS-MRO

2.1. Problem Description

The HFS-MRO problem was created for the first time and could be described as having *n* workpieces that must be processed on *s* different workstations (processing stages) (Figure 1). Each workstation *i* comprises a set of $m_i(m_i \geq 1)$ parallel machines with different machining capabilities. Each job has approximately the same processing flow, with minor differences in individual processes that can be skipped at one workstation or re-entered at another.

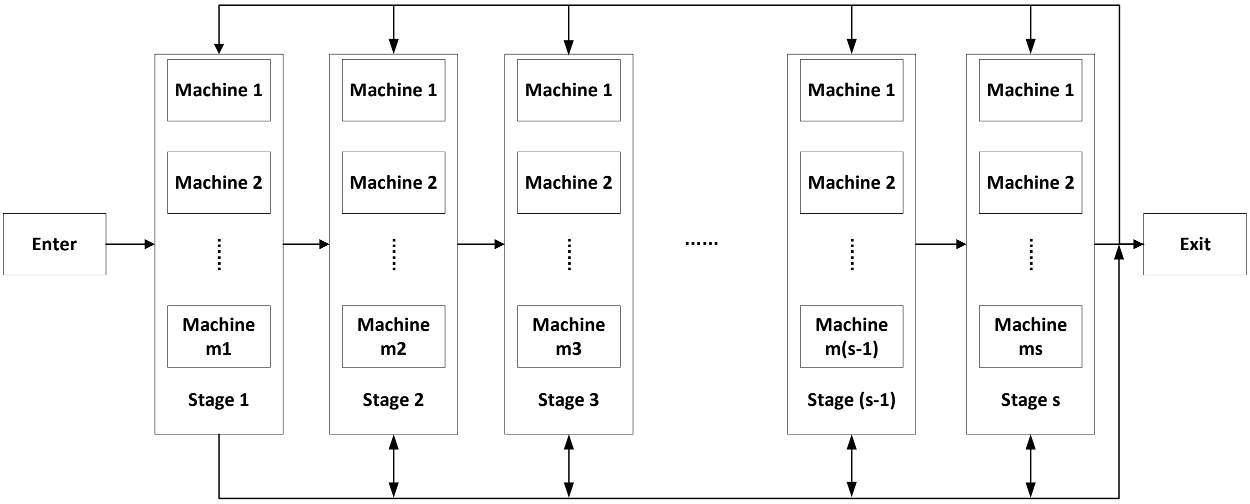


Figure 1. Schematic of HFS-MRO.

2.2. Model Assumptions

The following assumptions were considered based on the above problem description and the characteristics of an actual plant.

1. All workpieces and machines are ready at zero moments.
2. Operations can be ignored and re-entered.
3. At each processing stage, each process is exclusively performed on a single machine, and no machine is capable of processing more than one process simultaneously.
4. All processes for each workpiece are sequential and no misordering is allowed.
5. There is no priority requirement between the different workpiece processes. Once a workpiece has been processed, it cannot be interrupted, and there is no pre-emption.
6. The buffer capacity between two adjacent workstations is unlimited.
7. The workpiece is moved by workers between workstations, and the processing, preparation, and delivery times of each workpiece at each stage, as well as the processing energy consumption and idle energy consumption of each machine, are known and constant.
8. The case of machine breakdown is not considered.

2.3. Definition of Symbols

The parameter symbols and decision variables defined based on the model assumptions, are presented in Table 1.

Table 1. Definition of symbols.

Symbol	Definition
n	Number of jobs
j	Index for jobs, $j = 1, 2, \cdots, n$
S	Number of workstations (stages)
i	Index of workstations, $i = 1, 2, \cdots, s$
m_i	Number of identical parallel machines in workstation i
l	Index for machines in workstation i; $l = 1, 2, \cdots, m_i$
N_j	Number of operations for job j
k	Index of operations for job j; $k = 1, 2, \cdots, N_j$
O_{jk}	The k-th operation on job j
P_{jk}	Processing time of O_{jk}
S_{jk}	Starting time of O_{jk}
U_i	Collection of operations processed on the workstation i
C_j	Completion time of job j
PW_a	Machine a's energy consumption per unit of time during processing operations
PI_a	Energy consumption per unit of time when machine a is idle
M_{total}	Total number of machines at each workstation, $M_{total} = m_1 + m_2 + \cdots + m_s$
a	Index number of the machine, $a = 1, 2, \cdots, M_{total}$
R_a	Total number of operations on the machine a
B_{ar}	Start time of the r-th processing on machine a
D_{ar}	Completion time of the r-th processing on machine a
$T_{j(k-1)kau}$	Transport time of the two consecutive operations O_{jk-1} and O_{jk} of the job j from the machine a to the machine u
$ST_{jj'a}$	Adjustment time between two adjacent workpieces j and j' on machine a
λ	a sufficiently large positive number
Decision variables:	
r_{ijkl}	1, if O_{jk} processed on machine l in workstation i; 0, other
$Z_{jkj'k'}$	1, if O_{jk} prioritizes $O_{j'k'}$; 0, other
Y_{jkau}	1, if O_{jk} is processed on machine u while $O_{j(k-1)}$ is processed on machine a; 0, other
x_{jkar}	1, if O_{jk} corresponds to the r-th process of machine a; 0, other

2.4. Mathematical Planning Models

Objective function:

$$f_1 = \min C_{\max} \tag{1}$$

$$f_2 = EC_{\min} = \sum_{a=1}^{M_{\text{total}}} \sum_{r=1}^{R_a - 1} (D_{ar} - B_{ar}) \cdot PW_a + \sum_{a=1}^{M_{\text{total}}} \sum_{r=1}^{R_a - 1} (B_{ar+1} - D_{ar}) \cdot PI_a \tag{2}$$

Equations (1) and (2) represent the two objective functions of the optimization problem: minimizing the maximum completion time and minimizing the machine energy consumption, respectively.

For the bi-objective problem, this paper converted a multi-objective problem into a single-objective problem by the weighted sum method as:

$$f = \alpha_1 f_1^* + \alpha_2 f_2^* \tag{3}$$

where f_1^* and f_2^* represent the normalized values of the target functions f_1 and f_2,respectively, while α_1 and α_2 represent the weights of the target values, respectively. This then satisfies $\alpha_1 + \alpha_2 = 1$.

S.t.

$$\begin{gathered} \sum_{l=1}^{m_i} r_{ij(k-1)l} \cdot \left(S_{j(k-1)} + P_{j(k-1)} + T_{j(k-1)kau} \right) \leq \sum_{l=1}^{m_{i\prime}} r_{i\prime jkl'} \cdot S_{jk} + \lambda \left(1 - Y_{jkau} \right), \\ \forall i, i\prime, j; k = \{2, 3, \cdots, N_j\}; l = a, l' = u \end{gathered} \tag{4}$$

$$0 \leq \sum_{l=1}^{m_i} r_{ijkl} \leq 1, \forall i, j, O_{jk} \in U_i \tag{5}$$

$$\begin{gathered} \lambda \left(2 - r_{ijkl} - r_{ij'k'l} \right) + \lambda \left(1 - Z_{jkj'k'} \right) + \left(S_{j'k'} - S_{jk} \right) \geq P_{jk} + ST_{jj'a}, \\ \forall i, j < j', l = a, O_{jk} \in U_i, O_{j'k'} \in U_i \end{gathered} \tag{6}$$

$$\lambda\left(2 - r_{ijkl} - r_{ij'k'l}\right) + \lambda Z_{jkj'k'} + \left(S_{jk} - S_{j'k'}\right) \geq P_{j'k'} + ST_{j'ja}, \quad \forall i, j < j', l = a, O_{jk} \in U_i, O_{j'k'} \in U_i \tag{7}$$

$$\lambda\left(2 - r_{ijkl} - r_{ij'k'l}\right) + \left(S_{jk'} - S_{jk}\right) \geq P_{jk}, \quad \forall i, j < j', l, O_{jk} \in U_i, O_{j'k'} \in U_i \tag{8}$$

$$D_{ar} + ST_{jj'a} \cdot x_{j'k'ar} \leq D_{a(r+1)} - P_{jk} + \lambda\left(1 - x_{jka(r+1)}\right) \quad \forall j, j', a, k, k'; r = \{1, 2, \cdots, R_a - 1\} \tag{9}$$

Work pieces need to be manually moved between each workstation with transportation time taken into consideration. Therefore, constraint (4) ensures that the processing start time of operation O_{jk+1} is not earlier than the sum of the processing completion and transportation times of operation O_{jk}. When a certain operation in the workpiece can be missed, it means that the workpiece can skip the corresponding workstation. Therefore, the cumulative value of constraint (5) is zero. Conversely, it means that the workpiece must be processed on one machine in the workstation, and the cumulative value becomes 1. Constraints (6) and (7) ensure that each machine simultaneously processes one operation at most. Constraint (8) ensures that no wrong order occurs in the first processing and re-entry processing of the workpiece in the same process. There is a setting time before the machine starts processing; therefore, the machine's starting processing time must be later than the sum of the completion time of the previous process of the current workpiece and the adjustment time of the machine itself. Hence, constraint (9) ensures that the start time of the current operation on a machine is not less than the sum of the completion time of the previous operation on that machine and the set-up time between the two operations.

3. Improved Dual-Population Genetic Algorithm

HFS-MRO, as an extension of HFS, also belongs to the NP-hard family. It was discovered that, as the scale increased, the solution time increased exponentially, and the precise solution method could not solve the problem. Therefore, we used a meta-heuristic algorithm to solve the problem. The genetic algorithm (GA) [28], as an evolutionary algorithm that simulates the law of superiority and inferiority in nature, was a good choice. It is also a neighborhood search algorithm with the core idea of evolving continuously in a solution space, selecting the offspring with high fitness by the selection operator, performing genetic operations on the offspring with the highest fitness, and stopping the algorithm by iterating a certain number of times or when the individuals reach the required fitness value.

GA is commonly used to solve shop scheduling problems because of its simple operation and strong adaptability. However, the limitations of traditional GA, in parent selection, crossover, and variation, cause the algorithm to easily fall into the local optima. Therefore, the IDPGA was designed in this study based on the idea of a dual-population GA to avoid this mentioned problem, as shown in Figure 2. IDPGA used a three-layer gene coding method to better integrate the "re-entrant" and "ignore process". At the same time, the dual-population model was adopted alongside the hyperbolic tangent function to define the fitness similarity within the population, which successfully gave consideration to the "exploration and development" ability of the algorithm.

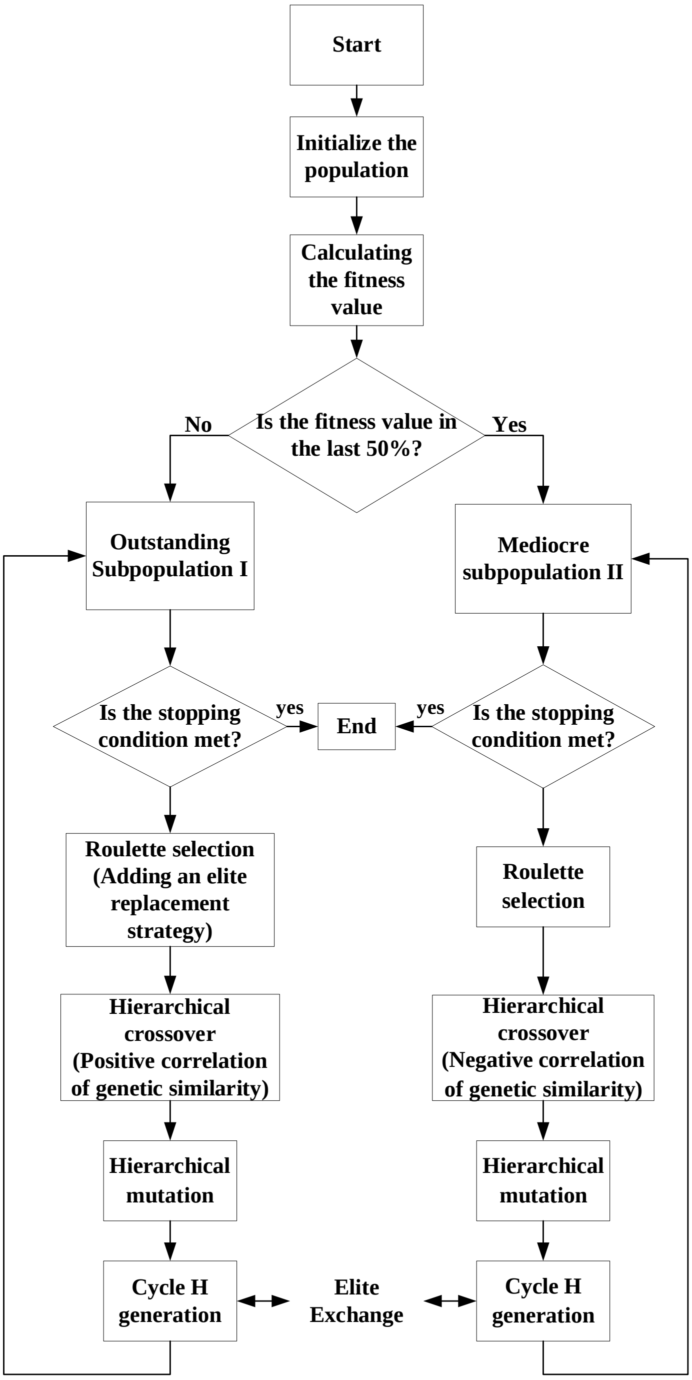

Start
Initialize the population
Calculating the fitness value
Is the fitness value in the last 50%?
No
Yes
Outstanding Subpopulation I
Mediocre subpopulation II
Is the stopping condition met?
yes
End
yes
Is the stopping condition met?
Roulette selection (Adding an elite replacement strategy)
Roulette selection
Hierarchical crossover (Positive correlation of genetic similarity)
Hierarchical crossover (Negative correlation of genetic similarity)
Hierarchical mutation
Hierarchical mutation
Cycle H generation
Elite Exchange
Cycle H generation

Figure 2. Flowchart of the IDPGA.

3.1. Encoding

There are two sub-problems to the general HFS problem: artifact ordering and machine assignment. The HFS-MRO problem requires "re-entry" and "missing" to be considered in a workpiece sequencing sub-problem. Therefore, this study used a three-level coding approach to represent individuals, including an operation vector, a machine vector, and a missing judgment vector. Each coding level was equal to the sum of the operations of all workpieces. Figure 3 shows a schematic of an individual chromosome with three workpieces, three workstations, and one round of re-entry. The number of machines in the workstation was [2 1 2].

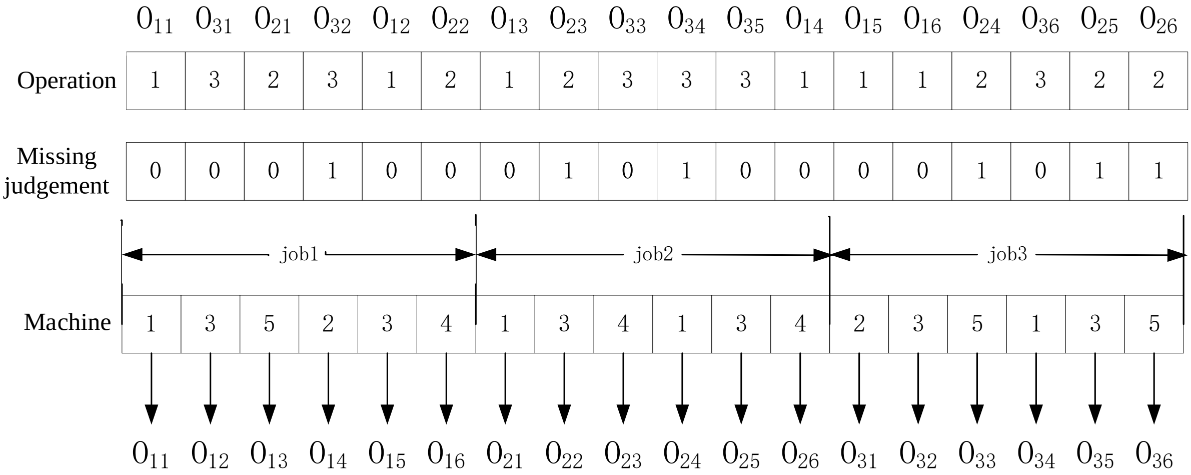


Figure 3. Chromosome encoding.

3.1.1. Operation Vector

An integer operation-based coding approach was used, wherein each gene was represented by a workpiece number. The order in which the same workpiece number appeared indicated the different processes of the workpiece. In this study, "re-entry" was considered before "ignore", as follows: first, a round of re-entry was considered with which to replicate the first three workstations, i.e., the number of operations in the workpiece increased from three to six, after which the ignore judgment vector in the second layer was used to determine which of the six operations should be ignored.

3.1.2. Missing Judgment Vector

Each part of the *n* vector was displayed as either "0" or "1" to determine if the corresponding operation was a "missing operation". If "0" was displayed, then the operation corresponding to that part of the workpiece could not be ignored; however, if "1" was displayed, the operation corresponding to that part was an ignorable process. This layer of vector was considered an attribute layer of the first vector and followed the process vector as it changed.

3.1.3. Machine Vector

The machine vector consisted of *n* parts, each of which indicated the machine corresponding to the operation of a given workpiece. Furthermore, all machines for each workstation were numbered together starting from 1. The total number of machines was the sum of the number of machines in all processing stages.

3.2. Decoding

The decoding steps were as follows.

Step 1: Based on three-level coding, the genes in the operation sequencing section are read from left to right to determine the sequences of all operations;

Step 2: The status of each operation is determined based on the missing judgment vector;

Step 3: The machine corresponding to each operation is determined based on the machine vector;

Step 4: The actual processing and preparation times for each operation are obtained;

Step 5: The start and finish times for each operation are calculated.

3.3. Calculation of Fitness Values

To convert a multi-objective into a single objective, objective functions f_1 and f_2 can be normalized to obtain f_1^* and f_2^* using Equation (10). Then, using the weighted sum method, the multi-objective problem is converted to a single objective according to Equation (3) to obtain the objective function f.

$$f_t^* = \frac{f_t - \min f_t}{\max f_t - \min f_t},\ t = 1,2 \tag{10}$$

where $t = 1,2$ denotes the objective function, and f_t^* represents the normalized value of the objective f_t. Maxf_t and minf_t represent the upper and lower bounds of the objective f_t, respectively. The smaller the objective function value of an individual in the population, the better the solution is. Therefore, the fitness function can be expressed as:

$$Fit = \frac{1}{f} = \frac{1}{\alpha_1 f_1^* + \alpha_2 f_2^*} \tag{11}$$

Makespan and energy-consumption minimization were the two objectives of this study. The primary objective of line optimization was to maximize productivity and efficiency, whereas the secondary objective for green production and reducing energy costs was to minimize energy consumption. Therefore, in this study, the objective weights were set as $\alpha_1 = 0.6$ and $\alpha_2 = 0.4$.

3.4. Initializing the Population

A simultaneous random first-tier permutation order and third-tier machine order were used to generate the initial population to increase population richness. Individual fitness values were calculated, with larger fitness values representing better solutions. Based on the fitness values, the initial population was divided into the top 50% of the outstanding sub-population I and the bottom 50% of the mediocre sub-population II.

3.5. Selection

Both populations were selected using the roulette wheel method.

Step1: Based on the obtained fitness values for each individual $Fit(1)$,$Fit(2)$,$\cdots$ $Fit(b)$,$\cdots$ $Fit(W)$, where W represents the size of each subpopulation, and b = 1,2,$\cdots$ W.

Step 2: The sum of all fitness values is used to obtain $\sum_{v=1}^{W} Fit(b)$ and to calculate the percentage of each individual fitness value in the sum as $p(b)$ and the cumulative probability as $q(b)$.

$$p(b) = \frac{Fit(b)}{\sum_{b=1}^{W} Fit(b)}, b = 1,2,\cdots W \tag{12}$$

$$q(b) = \sum_{b=1}^{W} p(b), b = 1,2,\cdots W \tag{13}$$

Step 3: A random number is generated $\gamma \in (0,1)$. If $\gamma \leq q(1)$, the first individual can be selected. If $q(b-1) \leq \gamma \leq q(b)$, the *b-th* individual can be selected.

Step 4: This step is then iterated W times.

An elite replacement strategy was used to accelerate the convergence of the outstanding subpopulation. In each iteration of the selection operation, the top 5% of individuals with the highest fitness values were selected to replace the bottom 5% of individuals with the lowest fitness values that completed the overall genetic operation for that round. This facilitated the rapid elimination of poorer individuals from the population.

3.6. Crossover

Considering the characteristics and various constraints of the HFS-MRO problem, a large number of non-feasible solutions were generated if the three vector layers were crossed directly. To avoid this scenario, this study designed different crossover methods for each layer of the vectors.

The first layer of the chromosome was crossed using a job-based crossover. The steps for this were as follows:

Step 1: All workpieces are randomly divided into two subsets job_1 and job_2.

Step 2: The operation genes belonging to subset job_1 from parents P_1 and P_2 are copied to children O_1 and O_2 at the same location.

Step 3: The operation genes belonging to subset job_2 from parents P_1 and P_2 are added to children O_1 and O_2, respectively, in the original order.

Figure 4 shows the operation vector crossover diagram, where $job_1 = [2]$ and $job_2 = [1,3]$.

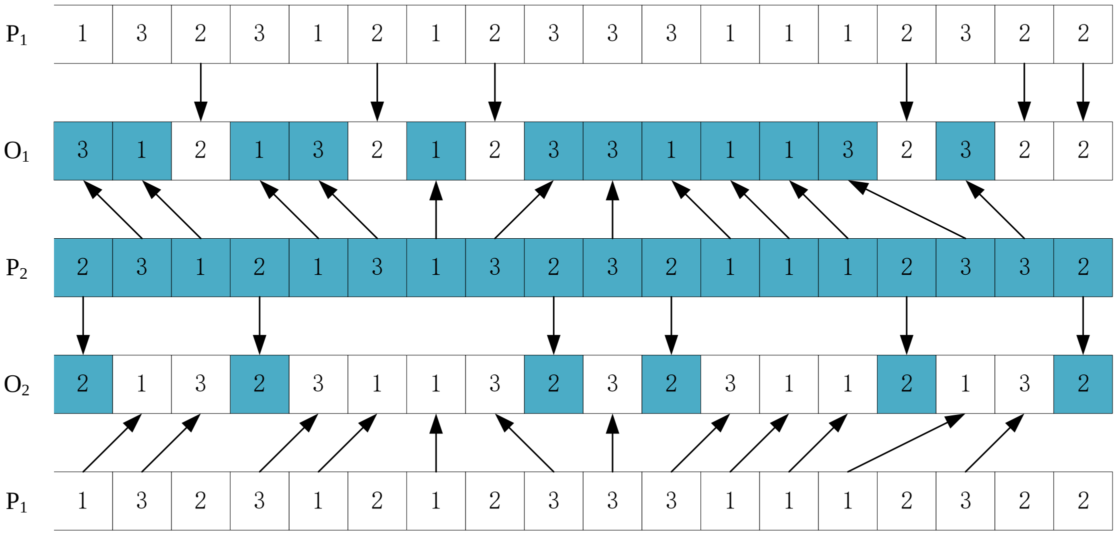


Figure 4. Schematic of the operation vector crossover.

The second layer of the chromosome can be considered the attribute layer of the first layer, and, therefore, it follows the changes made by the first layer.

The third layer of the chromosome is crossed using a method similar to that described previously. The steps for this are as follows:

Step 1: Divide all workstations into two subsets $station_1$ and $station_2$.

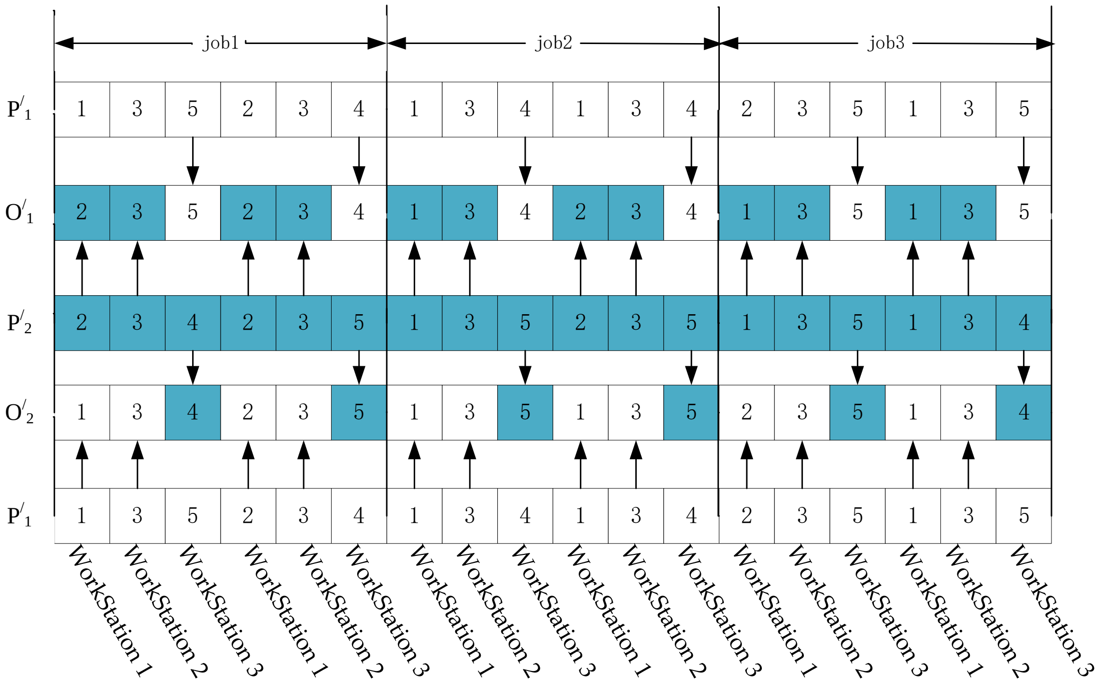
Step 2: Copy the genes from parents P'_1 and P'_2 belonging to subset $station_1$ onto children O'_1 and O'_2, respectively.

Step 3: Copy the genes from parents P'_1 and P'_2 belonging to subset $station_2$ onto children O'_1 and O'_2, respectively.

Crossover probabilities are dynamically adaptive based on the genetic similarity of individuals within a subpopulation. To improve the local search capability of each subpopulation, the crossover probabilities of the outstanding subpopulation were positively correlated with the similarity of individuals within the species when performing crossovers. This increased the efficiency of the subpopulation when searching for optimal solutions. The crossover probability of the mediocre subpopulation was negatively correlated with the similarity of individuals within the species, which expanded the search range to identify new search spaces. A solution for gene similarity has already been reported [29]. In this study, the hyperbolic tangent function was considered to define the relationship between crossover probability and individual similarity, and the value range was taken to be [−1,1] in the domain of its function definition. $\phi(\varphi_1, \varphi_2)$ denotes the similarity between the individuals of two parents. The similarity between the parental chromosomes must be positive, and, therefore, the first quadrant of the hyperbolic tangent function can be selected. p_{c1} denotes the crossover probability of the outstanding subpopulation, and p_{c2} denotes the crossover probability of the mediocre subpopulation:

$$p_{c1} = \frac{e^{\phi} - e^{-\phi}}{e^{\phi} + e^{-\phi}}, p_{c2} = 1 - \frac{e^{\phi} - e^{-\phi}}{e^{\phi} + e^{-\phi}}, \phi \in [0,1] \tag{14}$$

Figure 5 shows the strategy of the machine vectors for the crossing, where $station_1 = [3]$ and $station_2 = [1, 2]$.



	job1						job2						job3					
P'_1	1	3	5	2	3	4	1	3	4	1	3	4	2	3	5	1	3	5
O'_1	2	3	5	2	3	4	1	3	4	2	3	4	1	3	5	1	3	5
P'_2	2	3	4	2	3	5	1	3	5	2	3	5	1	3	5	1	3	4
O'_2	1	3	4	2	3	5	1	3	5	1	3	5	2	3	5	1	3	4
P'_1	1	3	5	2	3	4	1	3	4	1	3	4	2	3	5	1	3	5
	WorkStation 1	WorkStation 2	WorkStation 3	WorkStation 1	WorkStation 2	WorkStation 3	WorkStation 1	WorkStation 2	WorkStation 3	WorkStation 1	WorkStation 2	WorkStation 3	WorkStation 1	WorkStation 2	WorkStation 3	WorkStation 1	WorkStation 2	WorkStation 3

Figure 5. Schematic of the machine vector.

3.7. Mutation

The mutation operation involves introducing a random change in some genes of the chromosome with a certain probability of generating new chromosomal individuals. This operation can help the algorithm improve its local search ability by increasing the population diversity with a small perturbation. The mutation of each layer of the chromosomal vector remains different, considering the specificity of HFS-MRO and the requirement of avoiding non-feasible solutions while also mimicking the general strategy of the crossover operator.

The first layer of the chromosome uses a two-point swap variant approach, wherein two genes g_1 and g_2 are selected randomly on the operation vector, and their positions are swapped with each other.

The third layer of the chromosomal machines, the machine vector, uses a single-point mutation; that is, a gene g_3 is selected randomly on the machine vector and is changed to another machine in the same workstation.

The variation probability affects the quality of the chromosome. A higher mutation probability can destroy good individual genes, and, therefore, a lower mutation probability is selected when the chromosome fitness value is large. Conversely, the mutation probability can be increased appropriately to widen the search space.

$$p_m = \begin{cases} p_{m\min}, Fit' > Fit_{cvg} \\ p_{m\min} + \frac{(p_{m\max} - p_{m\min})(Fit_{cvg} - Fit')}{Fit_{cvg} - Fit_{\min}}, Fit' \le Fit_{avg} \end{cases} \tag{15}$$

where $p_{m\max}$ and $p_{m\min}$ denote the maximum and minimum mutation probabilities, respectively. Here, Fit_{avg} represents the average fitness value of the current subpopulation, and Fit' represents the fitness value of the selected chromosome.

3.8. Elite Exchange

The basic dual-population GA undergoes genetic manipulation to add the best individuals from each population to the other population, facilitating the other population's evolution and accelerating the convergence of the algorithm.

Based on this concept, the two subpopulations can be set to exchange the two best individuals in the outstanding subpopulation with two of the worst individuals in the mediocre subpopulation in terms of the fitness value after each cycle of the H generations. Simultaneously, the two best individuals in the mediocre subpopulation can be exchanged with the two worst individuals in the outstanding subpopulation to enhance the exploration capability of the algorithm.

4. Experiments and Result Analysis

4.1. Background of the Problem

Given the confidentiality constraints of the business, this section briefly describes the representative product process prototypes. The process flow diagrams for each of the five products are shown in Figure 6.

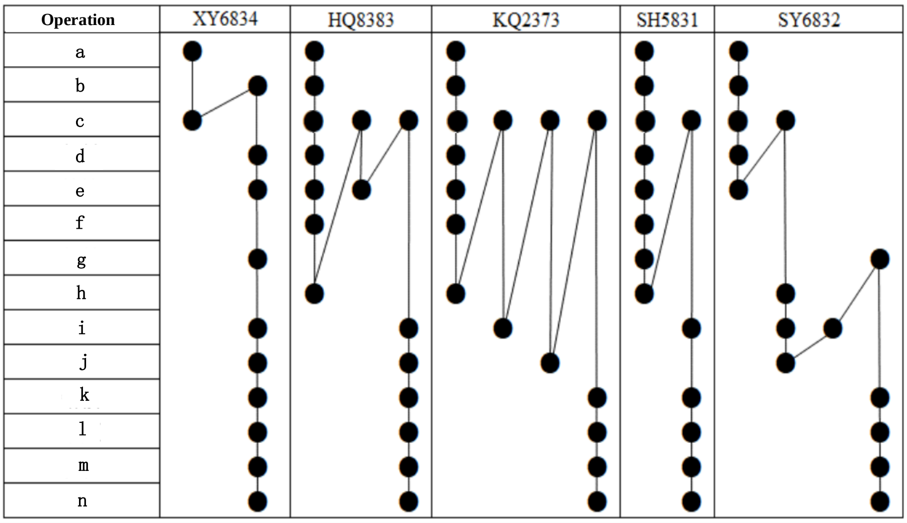


Figure 6. Process flow diagram.

The process flow diagrams shown in Figure 6 indicate that each product had a consistent flow direction, with "re-entry" and "missing operation". For example, product "KQ2373" was re-entered three times in operation "c" and never passed through operation "g", i.e., this operation was ignored. Product "SY6832" was re-entered once in operation "c", and operation "f" was ignored. A general HFS-MRO can be formulated based on this type of production line.

4.2. Experimental Data

This study used data from a company's electrical production and processing plants as a case study to verify the feasibility of the algorithm. The data are not disclosed in this study owing to the confidentiality requirements of the actual case data provided by the company. In this study, the features of the data are extracted based on the actual case data of the enterprise; the rules for generating arithmetic cases are listed in Table 2. As shown in Table 2, all quantities are in "pcs", the times in "min", and the powers in "kw". The process route includes a heat treatment process and occupies only one workstation.

Table 2. Table of arithmetic cases generation rules.

Parameters	Range of Values
Number of workpieces	{30,450,1200}
Number of workstations	U [5,10]
Number of re-entries	U [1,3]
Number of missing operations	U [1,4]
Number of machines per workstation	U [2,5]
Processing time of heat treatment	U [100,200]
Processing time of other stages	U [1,30]
Processing power of heat treatment machines	U [300,450]
Idle power of heat treatment machines	U [100,200]
Processing power of other machines	U [5,10]
Idle power of other machines	U [0.5,3]

This study assumed that the adjustment and transport times were approximately 20–60% and 30–50% of the standard processing times, considering the influence of worker handling. To facilitate the calculation, both processing and transport times were considered as integers with a minimum value of 1.

4.3. Parameter Setting

The parameter settings of the IDPGA algorithm affected the experimental results. The key factors that had to be set included the subpopulation size W, elite exchange H, and adaptive variation probability $P_{m\max}$ and $p_{m\min}$. A four-factor, three-level orthogonal test was designed to determine these four parameters. Table 3 lists the orthogonal factor.

Table 3. Orthogonal factor levels.

Levels Factor	W	H	$P_{m\max}$	$p_{m\min}$
1	50	5	0.001	0.10
2	100	10	0.005	0.15
3	150	15	0.010	0.20

To ensure a realistic experiment, a medium-sized workpiece volume was selected for the experiment, i.e., the number of workpieces was 450. Other data were generated randomly from Table 2. Each combination was run ten times, and the average of the optimal objective function values was selected as the response value. When the number of iterations reached 500, the algorithm terminated. Table 4 lists the designed L9 orthogonal test and its corresponding response values. The main effect diagram of the mean value after the analysis in Minitab 20 is presented in Figure 7. The mean responses are listed in Table 5, where "Delta" represents the maximum average minus the minimum average of the orthogonal factors. Figure 7 shows that the IDPGA algorithm performs optimally when the subpopulation size was $W = 150$, the elite exchange was $H = 5$, $p_{m\max} = 0.1$, and $p_{m\min} = 0.01$.

Variance analysis was performed for $p_{m\max}$, which ranked fourth in Table 5. By keeping the other three parameter levels constant, each level of $p_{m\max}$ was run 30 times, and the three columns of data were subjected to an ANOVA. The results are listed in Table 6. The test criterion for ANOVA was set at 0.05. The p-value in Table 6 is 0.02084, which is less than the set standard and indicates a significant difference among all levels. Therefore, $W = 150$, $H = 5$, $p_{m\max} = 0.1$, and $p_{m\min} = 0.01$ were selected as the initial parameter values.

Table 4. Orthogonal test table.

Serial Number	Factor				Response Value
	W	H	$P_{m\max}$	$p_{m\min}$	
1	50	5	0.10	0.001	0.123468
2	50	10	0.15	0.005	0.125570
3	50	15	0.20	0.010	0.124798
4	100	5	0.15	0.010	0.120744
5	100	10	0.20	0.001	0.126256
6	100	15	0.10	0.005	0.123383
7	150	5	0.20	0.005	0.121117
8	150	10	0.10	0.010	0.122076
9	150	15	0.15	0.001	0.123383

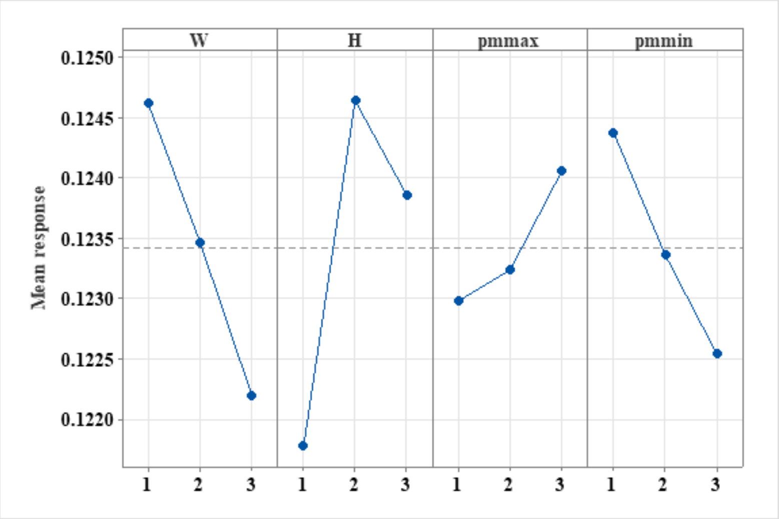


Figure 7. Mean main effects plot.

Table 5. Mean response table.

Levels	W	H	$p_{m\max}$	$p_{m\min}$
1	0.1246	0.1218	0.1230	0.1244
2	0.1235	0.1246	0.1232	0.1234
3	0.1222	0.1239	0.1241	0.1225
Delta	0.0024	0.0029	0.0011	0.0018
Ranking	2	1	4	3

Table 6. Analysis of variance.

Source of Difference	SS	df	MS	F	p-Value	F Crit
Intergroup difference	0.00014	2	6.81×10^{-5}	4.48305	0.02084	3.35413
Intra-group variation	0.00041	27	1.52×10^{-5}			
Total	0.00055	29				

4.4. Algorithm Test

The effectiveness of the IDPGA was verified by comparing it to the basic GA, ant colony optimization (ACO), and PSO. The convergence of various algorithms was tested by selecting a realistic medium-scale algorithm example. These algorithms were encoded in a similar manner. The population size of the basic GA was set to twice that of W, the crossover probability was 0.9, and the mutation probability was 0.1. For PSO [30], the inertia weight was $w = 0.8$ and factor was $c_1 = 2$ and $c_2 = 2$. For the basic ACO [31], the heuristic factor was $\alpha = 4$, the expectation factor was $\beta = 6$, and the information volatilization coefficient was $\rho = 0.5$. The parameters of the mayfly algorithm (MA) are derived from reference [32–34]. The algorithm termination condition was set to 500 iterations, MATLAB R2018b was adopted, and the test environment was the Windows 10 system, Intel® Core^{TM} i7-7700, 3.60 GHz processor with 16 GB of memory.

Figure 8 shows a comparison of the convergence of the four algorithms on a medium scale.

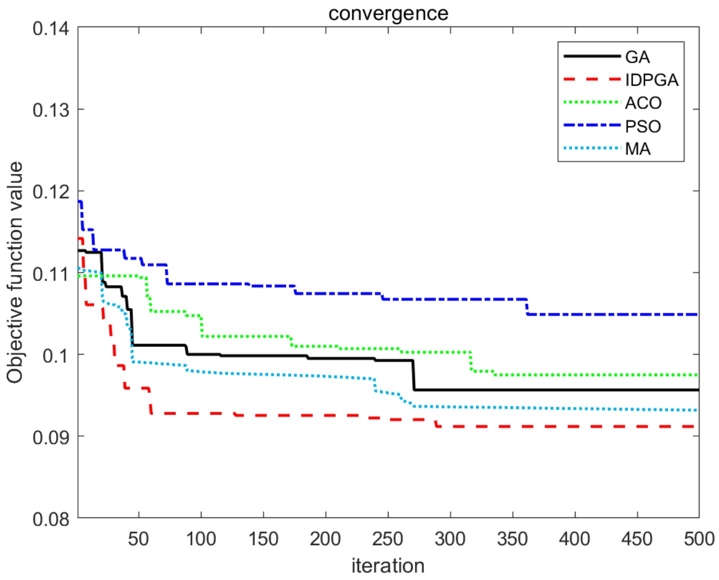


Figure 8. Comparison of convergence effect.

Figure 8 shows that the convergence effects of the five algorithms differed significantly. Although the initial solution of ACO was optimal, the convergence effect was insufficient and easily fell into the local optimal solution. Therefore, the results were not as positive as those of the other three algorithms. The initial solution of the PSO algorithm is insufficient, and the convergence rate was the slowest, even falling into local optima. The initial solution of the mayfly algorithm was similar to the ACO algorithm, but it was clearly seen that the convergence of the mayfly algorithm was stronger and faster, and the final result was only behind that of the IDPGA. The initial GA solution was acceptable; however, the convergence effect was unsatisfactory. In contrast, the proposed IDPGA used an adaptive genetic operator to accelerate the convergence speed and enhance the local search ability. Therefore, even if the initial solution was not satisfactory, it still achieved the highest solution accuracy, fastest solution speed, and optimum stability. Overall, this served to demonstrate the effectiveness of IDPGA.

To further illustrate the effectiveness of the algorithm, ten examples were randomly generated under various order sizes, according to Table 2, for a total of thirty examples. Four algorithms were solved for the thirty cases. To enhance accuracy, each case was run ten times, and the average of the optimal fitness values from the ten replicate trials was obtained. The optimization rates of the IDPGA relative to the various algorithms were also calculated, and the results are listed in Tables 7 and 8. For example, the optimization rate of IDPGA relative to GA is calculated as follows:

$$optimization\ rate = \frac{Ft_{IDPGA} - Fit_{GA}}{Fit_{GA}} \tag{16}$$

Table 7. Comparison of algorithmic results.

Serial Number	IDPGA	GA	Optimization Rate	PSO	Optimization Rate	ACO	Optimization Rate	MA	Optimization Rate
1	11.01	9.985	10.27%	9.845	11.83%	9.73	13.16%	10.21	7.84%
2	6.251	5.054	23.68%	5.055	23.66%	4.966	25.88%	5.19	20.44%
3	7.615	6.746	12.88%	6.491	17.32%	6.949	9.58%	6.63	14.86%
4	8.478	7.568	12.02%	7.052	20.22%	7.312	15.95%	7.47	13.49%
5	9.784	8.947	9.36%	8.082	21.06%	8.336	17.37%	8.09	20.94%
6	5.521	4.662	18.43%	4.605	19.89%	4.783	15.43%	4.83	14.31%
7	6.893	5.599	23.11%	5.364	28.50%	5.659	21.81%	5.137	34.18%
8	7.458	6.397	16.59%	6.457	15.50%	6.687	11.53%	6.38	16.90%
9	9.142	8.052	13.54%	7.668	19.22%	8.613	6.14%	8.212	11.32%
10	7.965	6.995	13.87%	6.828	16.65%	6.686	19.13%	6.7	18.88%
11	11.782	10.66	10.53%	10.66	10.53%	10.82	8.89%	10.33	14.06%
12	13.563	12.08	12.28%	12.42	9.20%	12.33	10.00%	12.01	12.93%
13	9.612	8.719	10.24%	8.771	9.59%	8.721	10.22%	8.73	10.10%
14	11.212	9.792	14.50%	10.17	10.25%	10.21	9.81%	10.22	9.71%
15	9.453	8.521	10.94%	8.392	12.64%	8.415	12.34%	8.44	12.00%
16	8.897	7.927	12.24%	7.876	12.96%	8.219	8.25%	7.88	12.91%
17	8.511	7.624	11.63%	7.827	8.74%	7.56	12.58%	7.55	12.73%
18	8.127	7.362	10.39%	7.179	13.21%	7.327	10.92%	7.2	12.88%
19	11.565	11.13	3.91%	11.13	3.91%	11.04	4.76%	11	5.14%
20	10.378	9.306	11.52%	9.271	11.94%	9.627	7.80%	9.33	11.23%
21	8.971	8.356	7.36%	8.328	7.72%	8.012	11.97%	8.28	8.35%
22	7.886	7.216	9.28%	7.1135	10.86%	7.178	9.86%	7.21	9.38%
23	13.201	11.71	12.73%	12.41	6.37%	12.17	8.47%	12.1	9.10%
24	7.89	6.751	16.87%	6.688	17.97%	6.959	13.38%	6.81	15.86%
25	12.134	10.4	16.67%	10.51	15.45%	11.27	7.67%	10.59	14.58%
26	11.156	10.01	11.45%	10.27	8.63%	9.789	13.96%	10.18	9.59%
27	10.571	9.107	16.08%	9.14	15.66%	9.466	11.67%	9.41	12.34%
28	9.45	8.41	12.37%	8.929	5.83%	8.807	7.30%	8.85	6.78%
29	7.241	6.336	14.28%	6.526	10.96%	6.27	15.49%	6.42	12.79%
30	6.251	5.054	23.68%	5.055	23.66%	4.966	25.88%	5.19	20.44%

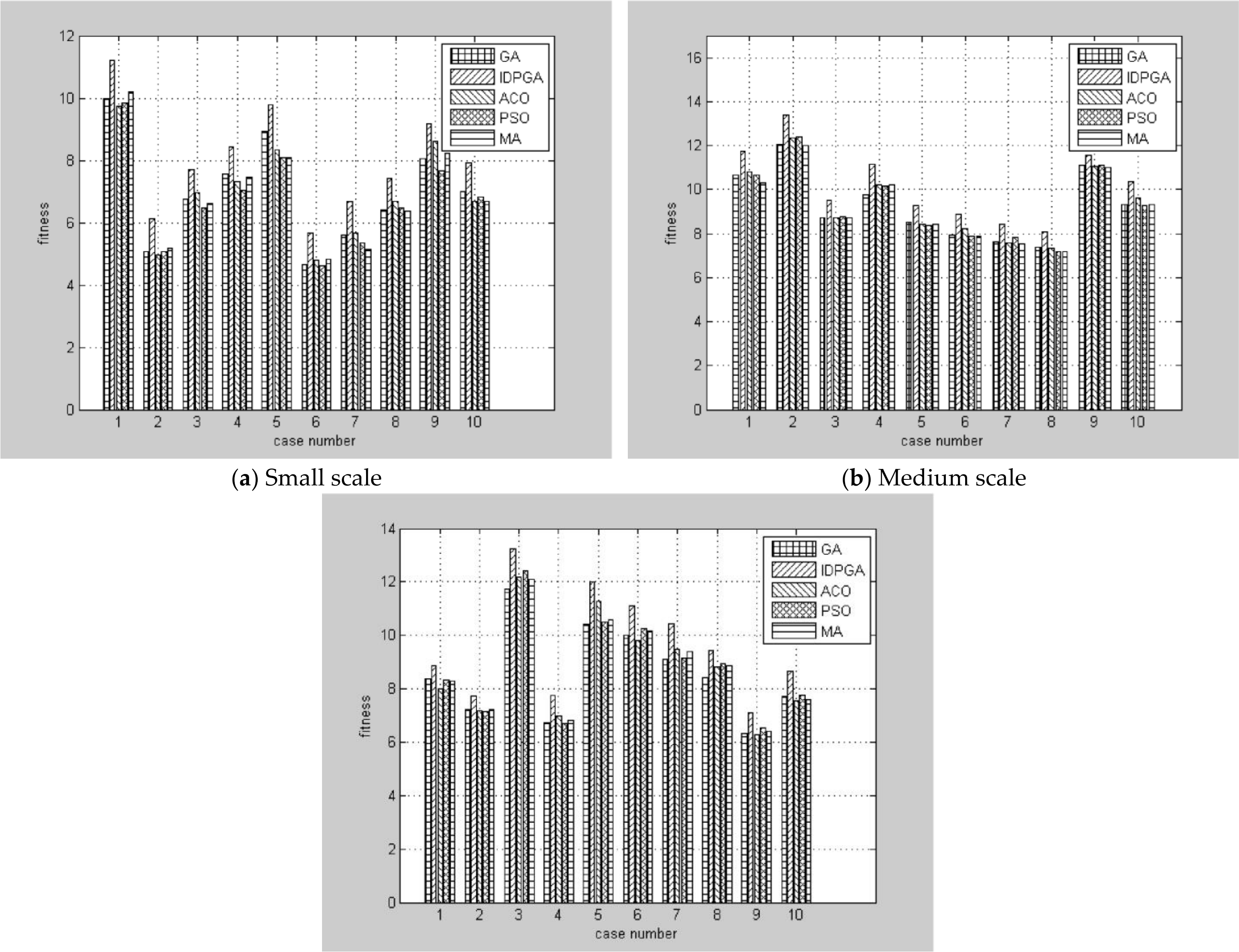
Table 8. Average relative optimization rate of IDPGA.

Algorithm Scale	Optimization Rate Relative to GA	Optimization Rate Relative to PSO	Optimization Rate Relative to ACO	Optimization Rate Relative to MA
Small	15.37%	19.39%	15.60%	18.36%
Medium	10.82%	10.30%	9.56%	11.37%
Large	12.90%	11.07%	11.43%	11.28%

Serial numbers 1–10, 11–20, and 21–30 represent small-scale, medium-scale, and large-scale cases, respectively. For a more intuitive representation, the fitness values of each algorithm listed in Table 7 are selected, and the results of the algorithm are compared as shown in Figure 9.

Tables 7 and 8 and Figure 9 show that IDPGA outperformed the other four algorithms on all three scales. The average optimal fitness value of the IDPGA was higher than those of the other four algorithms, and a relative optimization rate was the highest when solving small-scale examples. Therefore, compared with other algorithms, IDPGA had significant advantages in solving this problem.

The company aims to improve productivity through algorithmic scheduling, and, therefore, in addition to the superior performance of the algorithms, the robustness of the algorithms was also investigated to solve the problem efficiently and consistently each time. A randomly generated medium-scale example was run ten times for each algorithm, using the objective function value as the evaluation metric. The results obtained were tallied and plotted on a box-line graph, as shown in Figure 10.



(**a**) Small scale

(**b**) Medium scale

(**c**) Large scale

Figure 9. Comparison of algorithmic results: (**a**) Small scale; (**b**) Medium scale; (**c**) Large scale.

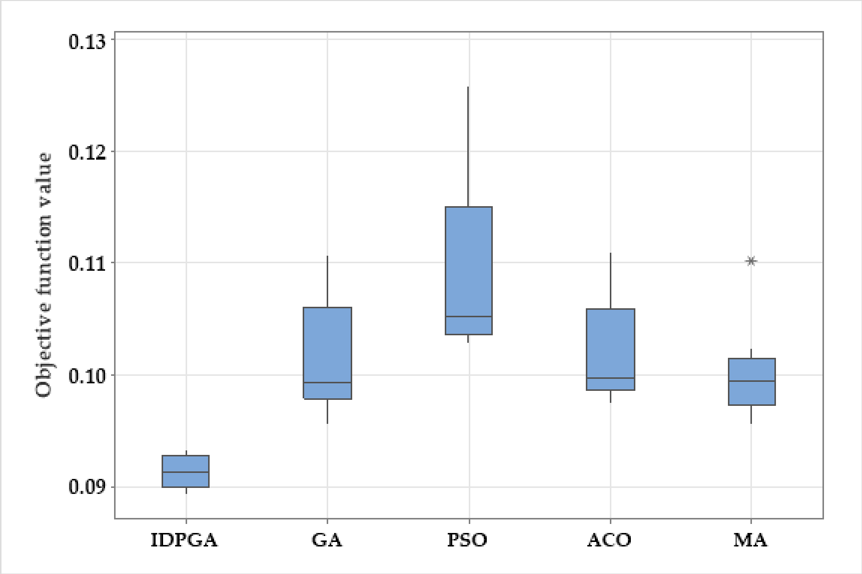


Figure 10. Robustness comparison.

The boxplot in Figure 10 shows that IDPGA was more stable and robust than the other four algorithms in solving the HFS-MRO scheduling problem; it could meet the actual production needs of enterprises.

5. Conclusions

An HFS-MRO was extracted and analyzed based on a real shop, and an integer programming model was established. In addition, IDPGA was proposed and applied to solve this problem. An example of a generating table was designed by analyzing and extending the actual data of the enterprise. The optimal parameter set was obtained by an orthogonal experiment, and the IDPGA was compared with four other algorithms on the same scale to confirm the significant advantages of the IDPGA in solving this problem and satisfying the actual production requirements of enterprises. IDPGA was used to solve the practical problems of the factory, including the low efficiency caused by manual scheduling plans and a reduction in production energy consumption. This could serve to enhance the competitiveness of enterprises and conform to the sustainable development policy advocated by the government.

However, there are some limitations to this study. Due to manual handling between each workstation, the transportation time experienced a large fluctuation. In the future, staff training could be considered or AGVs introduced instead of manual handling, aimed at standardizing transportation time. This would not only make the research more consistent with the actual situation but also play a greater role in the enterprise.

Author Contributions: Conceptualization, H.T.; methodology, H.T.; software, J.Z.; validation, J.Z.; formal analysis, H.T.; investigation, Y.S. and Z.Y.; resources, H.T. and Y.S.; data curation, J.Z. and Y.S.; writing—original draft preparation, H.T. and J.Z.; writing—review and editing, H.T., J.Z., and Z.Y.; visualization, J.Z. and Z.Y.; supervision, H.T. and Y.S.; project administration, H.T.; funding acquisition, H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China under Grant 2018YFB1308102.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Li, Y.; Li, X.; Gao, L. Review on Hybrid Flow Shop Scheduling Problems. *Chin. Mech. Eng.* **2020**, *31*, 2798.
2. Marichelvam, M.K.; Geetha, M. A memetic algorithm to solve uncertain energy-efficient flow shop scheduling problems. *Int. J. Adv. Manuf. Technol.* **2021**, *115*, 515–530. [CrossRef]
3. Kumar, P.R. Re-entrant lines. *Queueing Syst. Theory Appl.* **1993**, *13*, 87–110. [CrossRef]
4. Wittrock, R.J. Scheduling algorithms for flexible flow lines. *IBM J. Res. Dev.* **1985**, *29*, 401–412. [CrossRef]
5. Chamnanlor, C.; Sethanan, K.; Chien, C.-F.; Gen, M. Re-entrant flow shop scheduling problem with time windows using hybrid genetic algorithm based on auto-tuning strategy. *Int. J. Prod. Res.* **2014**, *52*, 2612–2629. [CrossRef]
6. Zhang, X.Y.; Chen, L. A re-entrant hybrid flow shop scheduling problem with machine eligibility constraints. *Int. J. Prod. Res.* **2018**, *56*, 5293–5305. [CrossRef]
7. Mousavi, S.M.; Mahdavi, I.; Rezaeian, J.; Zandieh, M. Bi-objective scheduling for the re-entrant hybrid flow shop with learning effect and setup times. *Sci. Iran.* **2018**, *25*, 2233–2253. [CrossRef]
8. Marichelvam, M.K.; Tosun, O.; Geetha, M. Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time. *Appl. Soft Comput.* **2017**, *55*, 82–92. [CrossRef]
9. Geng, K.; Ye, C.; Dai, Z.H.; Liu, L. Bi-Objective Re-Entrant Hybrid Flow Shop Scheduling considering Energy Consumption Cost under Time-of-Use Electricity Tariffs. *Complexity* **2020**, *2020*, 8565921. [CrossRef]
10. Gupta, J.N.D. Two-stage, hybrid flowshop scheduling problem. *J. Oper. Res. Soc.* **1988**, *38*, 359–364. [CrossRef]
11. 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]
12. Kun, W.; Lamei, X.; Cong, L. A novel SFLA for re-entrant hybrid flow shop scheduling. *J. Phys. Conf. Ser.* **2019**, *1314*, 012126.

13. 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]
14. Eskandari, H.; Hosseinzadeh, A. A variable neighbourhood search for hybrid flow-shop scheduling problem with rework and set-up times. *J. Oper. Res. Soc.* **2014**, *65*, 1221–1231. [CrossRef]
15. Xu, F.; Tang, H.T.; Xun, Q.N.; Lan, H.Y.; Liu, X.; Xing, W.F.; Zhu, T.Y.; Wang, L.; Pang, S.B. Research on Green Reentrant Hybrid Flow Shop Scheduling Problem Based on Improved Moth-Flame Optimization Algorithm. *Processes* **2022**, *10*, 2475. [CrossRef]
16. Qin, W.; Zhang, J.; Song, D. An improved ant colony algorithm for dynamic hybrid flow shop scheduling with uncertain processing time. *J. Intell. Manuf.* **2018**, *29*, 891–904. [CrossRef]
17. Nejad, M.G.; Shavarani, S.M.; Vizvari, B.; Barenji, R.V. Trade-off between process scheduling and production cost in cyclic flexible robotic cells. *Int. J. Adv. Manuf. Technol.* **2018**, *96*, 1081–1091. [CrossRef]
18. Geng, K.-F.; Ye, C.-M. Joint scheduling of machines and AGVs in green hybrid flow shop with missing operations AGV. *Kongzhi Juece Control Decis.* **2022**, *37*, 2723–2732.
19. Tseng, C.-T.; Liao, C.-J.; Liao, T.-X. A note on two-stage hybrid flowshop scheduling with missing operations. *Comput. Ind. Eng.* **2008**, *54*, 695–704. [CrossRef]
20. Saravanan, M.; Sridhar, S.; Harikannan, N. Optimization of Two-Stage Hybrid Flow Shop Scheduling Problems Using Genetic Algorithm. *Appl. Mech. Mater.* **2015**, *766–767*, 962–967.
21. Li, J.Q.; Pan, Q.K.; Duan, P.Y. An Improved Artificial Bee Colony Algorithm for Solving Hybrid Flexible Flowshop With Dynamic Operation Skipping. *IEEE Trans. Cybern.* **2016**, *46*, 1311–1324. [CrossRef]
22. Long, J.; Zheng, Z.; Gao, X.; Pardalos, P.M. Scheduling a realistic hybrid flow shop with stage skipping and adjustable processing time in steel plants. *Appl. Soft Comput.* **2018**, *64*, 536–549. [CrossRef]
23. Marichelvam, M.K.; Prabaharan, T. Performance evaluation of an improved hybrid genetic scatter search (IHGSS) algorithm for multistage hybrid flow shop scheduling problems with missing operations. *Int. J. Ind. Syst. Eng.* **2014**, *16*, 120–141. [CrossRef]
24. Saravanan, M.; Sridhar, S.; Harikannan, N. Minimization of mean tardiness in hybrid flow shop with missing operations using genetic algorithm. *J. Adv. Manuf. Syst.* **2016**, *15*, 43–55. [CrossRef]
25. Lei, D.; Guo, X. Hybrid flow shop scheduling with not-all-machines options via local search with controlled deterioration. *Comput. Oper. Res.* **2016**, *65*, 76–82. [CrossRef]
26. Dios, M.; Fernandez-Viagas, V.; Framinan, J.M. Efficient heuristics for the hybrid flow shop scheduling problem with missing operations. *Comput. Ind. Eng.* **2018**, *115*, 88–99. [CrossRef]
27. Siqueira, E.D.; Souza, M.; Souza, S.D. A Multi-objective Variable Neighborhood Search algorithm for solving the Hybrid Flow Shop Problem. *Electron. Notes Discret. Math.* **2018**, *66*, 87–94. [CrossRef]
28. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimedia Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef] [PubMed]
29. Chen, C.; Zhang, D.; Ren, P. Synthetic similarity measure for case retrieval in case-based reasoning diagnosis system. *Chin. J. Mech. Eng.* **2004**, *40*, 48–52. [CrossRef]
30. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings, Anchorage, AK, USA, 4–9 May 1998.
31. Alaykýran, K.; Engin, O.; Döyen, A. Using ant colony optimization to solve hybrid flow shop scheduling problems. *Int. J. Adv. Manuf. Technol.* **2007**, *35*, 541–550. [CrossRef]
32. Zervoudakis, K.; Tsafarakis, S. A mayfly optimization algorithm. *Comput. Ind. Eng.* **2020**, *145*, 106559. [CrossRef]
33. Zhao, C.; Lui, C.F.; Du, S.C.; Wang, D.; Shao, Y.P. An earth mover's distance based multivariate generalized likelihood ratio control chart for effective monitoring of 3D point cloud surface. *Comput. Ind. Eng.* **2023**, *175*, 108911. [CrossRef]
34. Shao, Y.P.; Xu, F.C.; Chen, J.; Lu, J.S.; Du, S.C. Engineering surface topography analysis using an extended discrete modal decomposition. *J. Manuf. Process.* **2023**, *90*, 367–390. [CrossRef]

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