mathematics

MDPI

Article

An Improved Mayfly Optimization Algorithm for Type-2 Multi-Objective Integrated Process Planning and Scheduling

Ke Yang [1] and Dazhi Pan [1,2,*]

[1] College of Mathematic and Information, China West Normal University, Nanchong 637009, China; kyang0819@hotmail.com
[2] Sichuan Colleges and Universities Key Laboratory of Optimization Theory and Applications, Nanchong 637009, China
* Correspondence: pdzzj@cwnu.edu.cn

Abstract: The type-2 multi-objective integrated process planning and scheduling problem, as an NP-hard problem, is required to deal with both process planning and job shop scheduling, and to generate optimal schedules while planning optimal machining paths for the workpieces. For the type-2 multi-objective integrated process planning and scheduling problem, a mathematical model with the minimization objectives of makespan, total machine load, and critical machine load is developed. A multi-objective mayfly optimization algorithm with decomposition and adaptive neighborhood search is designed to solve this problem. The algorithm uses two forms of encoding, a transformation scheme designed to allow the two codes to switch between each other during evolution, and a hybrid population initialization strategy designed to improve the quality of the initial solution while taking into account diversity. In addition, an adaptive neighborhood search cycle based on the average distance of the Pareto optimal set to the ideal point is designed to improve the algorithm's merit-seeking ability while maintaining the diversity of the population. The proposed encoding and decoding scheme can better transform the continuous optimization algorithm to apply to the combinatorial optimization problem. Finally, it is experimentally verified that the proposed algorithm achieves better experimental results and can effectively deal with type-2 MOIPPS.

Keywords: multi-objective optimization; process planning; shop scheduling; neighborhood structure

MSC: 49M37; 90-10; 90C30

check for updates

Citation: Yang, K.; Pan, D. An Improved Mayfly Optimization Algorithm for Type-2 Multi-Objective Integrated Process Planning and Scheduling. *Mathematics* **2023**, *11*, 4384. https://doi.org/10.3390/math11204384

Academic Editors: Xue-Ming Yuan and Hao Yu

Received: 11 September 2023
Revised: 16 October 2023
Accepted: 18 October 2023
Published: 21 October 2023

1. Introduction

Process planning and shop scheduling are two critical aspects of a manufacturing system, and traditional manufacturing systems consider these two aspects independently, but as manufacturing grows, this model often fails to meet production needs [1,2]. By combining the two into one, the integrated process planning and scheduling problem (IPPS) is able to improve responsiveness to meet production needs. According to the differences in the approach of process planning, IPPS can be divided into two categories [3]. The first category (type-1 IPPS) deals with process planning and shop scheduling in turn, i.e., it first optimizes the processing path for each job and then carries out scheduling optimization; the second category (type-2 IPPS) optimizes both parts at the same time, taking into account the intrinsic connection between process planning and shop scheduling, which is more suitable for actual production and increases the difficulty of problem solving. Real-life optimization problems often involve multiple optimization goals, such as people often hoping to obtain more benefits at a lower cost. The solution of a multi-objective optimization problem (MOP) differs significantly from that of a single-objective optimization problem due to the fact that it is difficult to reach a solution that is optimal for each objective value in an MOP, and often a compromise set of solutions is obtained. The type-2 multi-objective integrated process planning and scheduling problem (type-2 MOIPPS) is more practically relevant, and in addition, it is very important to make scheduling decisions for the Pareto front of type-2 MOIPPS in real production.

Research on IPPS began in the mid-1980s [4]; Jin et al. [3] divided IPPS problems into two categories based on differences in the treatment of the process planning, and proposed several hybrid 0-1 integer planning models for type-2 IPPS. The common benchmark instance for type-2 IPPS was proposed by Kim et al. [5], which contains 18 jobs and 15 machines, which can be divided into 24 cases of different sizes according to different degrees of process planning flexibility. For type-2 IPPS problems, there are mainly exact algorithms and intelligent algorithms. The research in Bahman et al. [6] first proposed a constraint planning model for type-2 IPPS and developed a logic-based Benders decomposition method for solving it with good results. The research in Ausaf et al. [7] proposed a priority-based heuristic algorithm to optimize the maximum completion time with good results. The research in Huang et al. [8] proposed a quadratic description method based on OR subgraphs and used the ant colony algorithm to solve them. The research in Zhang and Wong [9] constructed a multi-intelligence system based on the ant colony algorithm to solve them. The research in Wu and Li [10] proposed a hierarchical algorithm fusing harmony search and genetic algorithm to solve type-2 IPPS. For MOIPPS, Xuan et al. [11] proposed a hybrid algorithm for solving it by integrating the operation of clustering algorithm, differential evolution algorithm, and genetic algorithm, which maintains the diversity of feasible solutions and effectively optimizes the process planning and scheduling scheme, but the algorithm is designed for type-1 IPPS. The research in Li et al. [12] solves MOIPPS based on game theory. The research in Mohapatra et al. [13] treats MOIPPS as a multi-objective optimization problem in a reconfigurable manufacturing environment with NSGA-2 processing. The research in Shokouhi [14] solves MOIPPS with completion time, total machine load, and critical machine load as the objectives in a weighted manner. The research in Wen et al. [15] considers the robustness of the system as well as machine failures to optimize MOIPPS for rescheduling. In summary, there is relatively little research on type-2 MOIPPS. The above literature analysis shows that the solution of type-2 IPPS is mainly focused on intelligent algorithm, and the crossover and mutation operations of genetic algorithm are usually applied in the iterative process of intelligent algorithm, which is determined by its encoding and decoding scheme. How to further apply continuous optimization algorithms to discrete problems remains to be studied. For type-2 IPPS, Table 1 summarizes the proposed algorithms from the literature.

Table 1. The available literature on the type-2 IPPS.

Publication	Algorithms
[3]	Exact algorithm
[6]	Logic-based Benders decomposition method
[7]	Priority-based heuristic algorithm
[8]	Quadratic description method based on OR subgraphs
[9]	Multi-intelligence system based on the ant colony algorithm
[10]	Hierarchical algorithm fusing harmony search and genetic algorithm

The mayfly optimization algorithm (MA) [16], as a novel metaheuristic optimization algorithm, combines the main advantages of genetic algorithm, particle swarm optimization algorithm, and firefly algorithm, and is widely used in various optimization problems. There are some examples. The research in Awei et al. [17] proposed a fusion algorithm combining an improved mayfly optimization algorithm and dynamic windowing method to solve the robot path planning problem. The research in Dong et al. [18] proposed an optimal siting and capacity determination method for distribution networks, and designed an improved multi-objective mayfly algorithm to solve it. The research in Damin et al. [19] proposed a resource allocation algorithm with improved discrete mayfly algorithm for cognitive heterogeneous cellular networks for the optimization of the uplink resource allocation problem in cognitive heterogeneous cellular networks, but the algorithm has not been applied to the solution of type-2 MOIPPS.

Lastly, we conclude that solving type-2 MOIPPS has several key problems, as follows:

- One problem is how to deal with the process planning and shop scheduling aspects in IPPS at the same time.
- Solving MOP not only requires the solution set to be close to the true Pareto front, but also makes the solution set uniformly distributed on the Pareto front [20], so the convergence and distribution of the solution need to be considered at the same time. This raises the question of how to balance the convergence and distribution of the solution set.
- Lastly, we have the question of how to decide the solutions in the Pareto optimal set.

To address the above aspects, the main contributions of this paper are as follows:

(1) We develop a mathematical model for type-2 MOIPPS based on the AND/OR directed graph.
(2) We extend the mayfly optimization algorithm for the first time for solving this multi-objective optimization problem.
(3) In the coding section, we use two coding methods to better apply the sequential optimization algorithm to this combinatorial optimization problem.
(4) We improve the neighborhood structure of the moving process to adapt to the solution of this problem, forming a dual population structure of internal population and external archive, which effectively balances the convergence of the algorithm and the diversity of the populations, and encourages the Pareto solution set to uniformly approximate the true Pareto front.

The remainder of this paper is organized as follows. Section 2 describes how the problem is described and mathematically models it based on AND/OR directed graphs. Section 3 describes the fundamentals of the mayfly algorithm. Section 4 proposes an encoding scheme, a decoding scheme, and a population initialization strategy. Section 5 describes the use of the proposed algorithm for solving the type-2 MOIPPS. Finally, Section 6 presents an experimental validation of the proposed algorithm. We conclude the paper in Section 7.

2. Problem Description and Mathematical Modeling

2.1. Basic Definitions of MOP

An MOP often contains multiple conflicting optimization objectives, which can be mathematically defined as follows:

$$\begin{aligned} \min \quad & \boldsymbol{f}(\boldsymbol{x}) = \big(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_p(\boldsymbol{x})\big) \\ \text{s.t.} \quad & \begin{cases} g_i(\boldsymbol{x}) \le 0, & i = 1,2,\ldots,k \\ h_j(\boldsymbol{x}) = 0, & j = 1,2,\ldots,l \end{cases} \end{aligned} \tag{1}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{D}$ is a decision variable vector from the decision space $\mathbb{D}$, $\boldsymbol{f}(\cdot)$ denotes p objectives to be optimized, $g_i(\cdot)$ is the inequality constraint, and $h_i(\cdot)$ is the equation constraint.

Definition 1 (Pareto dominate)**.** *Given decision variable vectors $\boldsymbol{x} \in \mathbb{D}, \boldsymbol{y} \in \mathbb{D}$ if $\forall i = 1,2,\ldots,p, f_i(\boldsymbol{x}) \le f_i(\boldsymbol{y})$, and $\exists j \in \{1,2,\ldots,p\}, f_j(\boldsymbol{x}) < f_j(\boldsymbol{y})$, we say that $\boldsymbol{x}$ dominates $\boldsymbol{y}$, noted as $\boldsymbol{x} \prec \boldsymbol{y}$.*

Definition 2 (Absolute optimal solution)**.** *The decision variable vector $\boldsymbol{x}^* \in \mathbb{D}$, if $\forall \boldsymbol{x} \in \mathbb{D}$, $f_i(\boldsymbol{x}^*) \le f_i(\boldsymbol{x})$, $\forall i = 1,2,\ldots,p$, so $\boldsymbol{x}^*$ is said to be the absolute optimal solution.*

Definition 3 (Pareto optimal)**.** *The decision variable vector $\boldsymbol{x}^* \in \mathbb{D}$ is said to be a Pareto optimal solution (non-dominated solution) if $\nexists \boldsymbol{x} \in \mathbb{D}$ and $\boldsymbol{x} \prec \boldsymbol{x}^*$.*

Definition 4 (Pareto set)**.** $PS = \{\boldsymbol{x} \in \mathbb{D} \mid \nexists \boldsymbol{y} \in \mathbb{D}, \boldsymbol{y} \prec \boldsymbol{x}\}$.

Definition 5 (Pareto front)**.** $PF = \{\boldsymbol{f}(\boldsymbol{x}) \mid \boldsymbol{x} \in PS\}$.

2.2. Problem Description

Let $\mathbb{J} = \{J_1, J_2, \ldots, J_n\}$ be a set of jobs; $\forall J_i, i = 1, 2, \ldots n$ requires a set of operations $\mathbb{J}_i = \{O_{i,1}, O_{i,2}, \ldots O_{i,n_i}\}$ for completion. Let $\mathbb{M} = \{M_1, M_2, \ldots, M_m\}$ be a set of machines; $\forall O_{i,j}, i = 1, 2, \ldots, n, j = 1, 2, \ldots n_i$ can be processed on several different machines $\mathbb{M}_{i,j} \subset \mathbb{M}$, with different processing times for different operations on different machines. The process planning of the job is represented by an AND/OR directed graph, as illustrated in Figure 1a. Solving the problem requires determining the processing path of each job, determining the processing machine for each operation, and determining the processing order of the operations on each machine to satisfy optimization objectives.

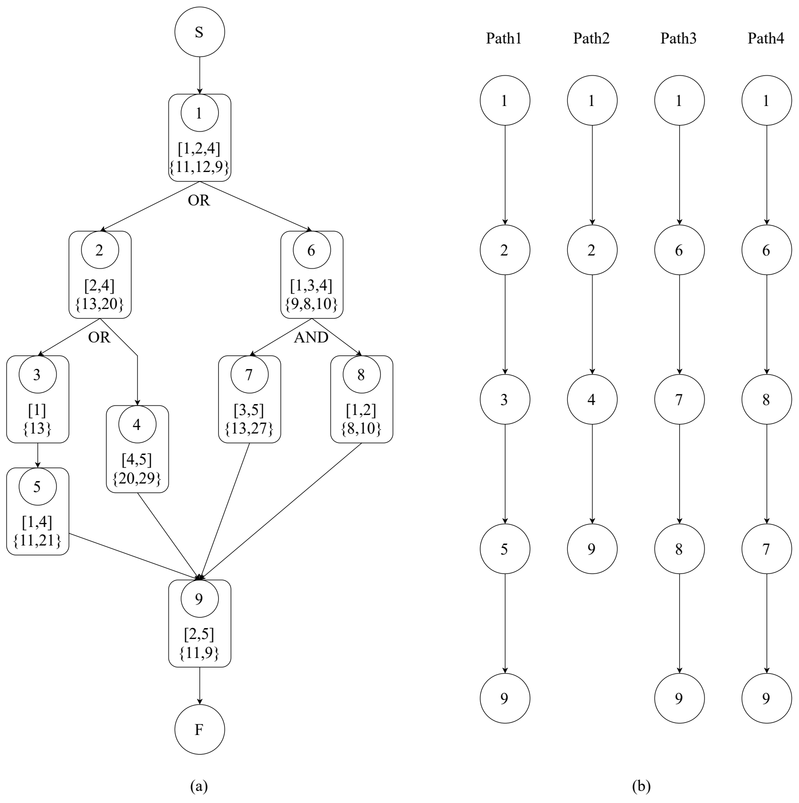


Figure 1. AND/OR directed graph.

In the AND/OR directed graph, each node represents one operation, the directed edges link two adjacent operations, and the direction of the directed edges indicates the processing order of the operations. The process flexibility of the job is represented by the AND node and the OR node. The AND node means that to complete the job, the operations under the AND node need to be processed, but there is no sequential constraint between these operations; the OR node means that to complete the job, only one path under the OR node needs to be selected. The number in the middle bracket indicates the number of the machine available for this operation, and the number in the curly brackets indicates the processing time of this operation on the corresponding machine. The AND/OR directed graph shown in Figure 1 has four machining paths (Figure1b).

The type-2 MPIPPS is also subject to the following assumptions:

1. All machines are available from moment 0.
2. All jobs are available to be machined from moment 0.
3. Each job cannot be stopped until machining is complete.

4. There are no priority constraints between jobs.
5. One job can only be machined in one process at the same moment.
6. Only one job can be processed by one machine at the same moment.
7. The transportation time of the jobs is ignored.

2.3. Process Planning Approach

Before building the mathematical model, the methodology used in this paper for treatment process planning is first introduced. As seen in the AND/OR directed graph, the process planning of a job consists of two parts—the selection of the operations to be processed and the ranking of the selected operations.

For the first problem, this paper uses the approach proposed by [6] to determine the operation of processing. For each job $J_i, i = 1, 2, \ldots, n$, a decision variable $s_{i,v}$ is assigned to each OR node $Q_{i,v}, v = 1, 2, \ldots, nq_i$ of the job (nq_i denotes the number of OR nodes of the job J_i), $s_{i,v} = 0$ if that OR node chooses the path on the left, and $s_{i,v} = 1$ otherwise. Then two sets, $F^0_{i,j}, F^1_{i,j}$, are assigned to each operation $O_{i,j}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, n_i$, where the elements in $F^0_{i,j}$ are the OR nodes that select the left-hand path making the operation required to be processed, and the elements in $F^1_{i,j}$ are the OR nodes that select the right-hand path, making the operation required to be processed, so that the sufficient conditions for the operation of the job to be selected can be obtained as follows:

Theorem 1 (A sufficient condition for the operation to be selected)**.** *The operation $O_{i,j}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, n_i$ is performed if and only if $\sum_{v \in F^0_{i,j}} s_{i,v} + \sum_{v \in F^1_{i,j}} (1 - s_{i,v}) = 0$.*

For the second problem, a priority constraint matrix $A_i = \left(a_{j,r}\right)_{n_i}$ is assigned to each job $J_i, i = 1, 2, \ldots, n$, with $a_{j,r} = 1$ when operation $O_{i,j}$ is processed before operation $O_{i,r}$ and $a_{j,r} = 0$ otherwise. The priority of the two operations will be determined by this matrix.

2.4. Mathematical Model

Objectives

- Makespan (MK): The makespan indicates the completion time of the last job to leave the machining system, reflects the efficiency of machine utilization, and is the most widely used evaluation indicator in scheduling problems.

$$f_1 = \min \quad \max_{i \in \mathbb{J}, j \in \mathbb{J}_i} \left(c_{i,j}\right) \tag{2}$$

- Total machine load (TL): The total machine load represents the sum of the time spent on all machine processes and reflects the total consumption cost of the machine.

$$f_2 = \min \quad \sum_{i \in \mathbb{J}} \sum_{j \in \mathbb{J}_i} \sum_{k \in \mathbb{M}_{i,j}} y_{i,j,k} p_{i,j,k} \tag{3}$$

- Key machine load (KL): Since the processing time of the same process varies on different machines, the selection of machines may lead to machine load imbalance, and the maximum machine load is used to measure the machine load balance.

$$f_3 = \min \quad \max_{k \in \mathbb{M}} \left(\sum_{i \in \mathbb{J}} \sum_{j \in \mathbb{J}_i} y_{i,j,k} p_{i,j,k} \right) \tag{4}$$

Constrains

$$\sum_{k \in \mathbb{M}_{i,j}} y_{i,j,k} \leq 1, \forall i \in \mathbb{J}, j \in \mathbb{J}_i \tag{5}$$

$$\sum_{k\in\mathbb{M}_{i,j}} y_{i,j,k} \geq 1 - \left(\sum_{v\in F^0_{i,j}} s_{i,v} + \sum_{v\in F^1_{i,j}} (1-s_{i,v}) \right), \forall i \in \mathbb{J}, j \in \mathbb{J}_i \tag{6}$$

$$c_{i,j} \geq 0, \forall i \in \mathbb{J}, j \in \mathbb{J}_i \tag{7}$$

$$c_{i,j} \leq L \cdot \left(1 - \left(\sum_{v\in F^0_{i,j}} s_{i,v} + \sum_{v\in F^1_{i,j}} (1-s_{i,v}) \right) \right), \forall i \in \mathbb{J}, j \in \mathbb{J}_i \tag{8}$$

$$\begin{gathered} c_{l,r} \geq c_{i,j} + y_{i,j,k} p_{i,j,k} - L \cdot (1 - z_{ij,lr,k}), \\ \forall i,l \in \mathbb{J}, j \in \mathbb{J}_i, r \in \mathbb{J}_l, k \in \mathbb{M}_{i,j} \cap \mathbb{M}_{l,r} \end{gathered} \tag{9}$$

$$a_{j,r} + a_{r,j} = 1, \forall i \in \mathbb{J}, j,r \in \mathbb{J}_i, a_{j,r} = 0 \wedge a_{r,j} = 0 \tag{10}$$

$$c_{i,r} \geq c_{i,j} + \sum_{k\in\mathbb{M}_{i,j}} y_{i,j,k} p_{i,j,k}, \forall i \in \mathbb{J}, j,r \in \mathbb{J}_i, a_{j,r} = 1 \tag{11}$$

Constraints (5) and (6) assign a unique machine to each selected operation. Constraint (7) ensures that the completion time of each operation is not less than 0. Constraint (8) sets the completion time for operations that are not processed to 0. Constraint (9) processes the scheduling of operations on the same machine. Constraints (10) and (11) process the scheduling of operations for the same job.

Indices and sets:	
i, l	Index for jobs
j, r	Index for operations
k, u	Index for machines
v	Index for OR nodes
$\mathbb{J}_i$	The set of all operations of job J_i
$\mathbb{M}_{i,j}$	The set of machines for $O_{i,j}$
Parameters:	
$p_{i,j,k}$	The processing time of $O_{i,j}$ on machine k
$c_{i,j}$	Completion time of $O_{i,j}$
L	A sufficiently large positive real number
Decision variables:	
$s_{i,v}$	$s_{i,v} = 1$ if OR node chooses right path, and 0 otherwise
$a_{j,r}$	$a_{j,r} = 1$ if $O_{i,j}$ precedes $O_{i,r}$, otherwise $a_{j,r} = 0$
$y_{i,j,k}$	$y_{i,j,k} = 1$ if $O_{i,j}$ is processed on M_k, and 0 otherwise
$z_{ij,lr,k}$	$z_{ij,lr,k} = 1$ if the $O_{i,j}$ precedes $O_{l,r}$, and 0 otherwise

3. Overview of Mayfly Optimization Algorithm

Mayfly optimization algorithm (MA) was proposed by [16], which can be seen as an improvement of particle swarm algorithm with the advantages of particle swarm algorithm, genetic algorithm, and firefly algorithm, with the process of finding the optimal male mayfly move, female mayfly move, and male and female mayfly cross.

1. Movement of male mayflies. Male mayflies will adjust their position based on their experience and their neighbors with the following formula for speed update and position update.

$$v_i^{t+1} = v_i^t + a_1 e^{-\beta r_p^2} (pbest_i - x_i^t) + a_2 e^{-\beta r_g^2} (gbest_i - x_i^t) \tag{12}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{13}$$

2. Movement of female mayflies. Female mayflies are attracted to male mayflies and the algorithm models this attraction process as a deterministic model where the better male individual attracts the better female individual, and the female individual velocity update and position update equations are as follows.

$$v_i^{t+1} = \begin{cases} v_i^t + a_2 e^{-\beta r_m^2}\left(x_i^t - y_i^t\right), & f(y_i) > f(x_i) \\ v_i^t + l_{fl} \cdot r, & \text{otherwise} \end{cases} \tag{14}$$

$$y_i^{t+1} = y_i^t + v_i^{t+1} \tag{15}$$

3. Mating of mayflies. Sires are selected in the same way as male and female mayflies are attracted, with the better males crossing over with the better females and males.

$$\begin{aligned} off_1 =& a_3 \cdot P_1 + (1 - a_3) \cdot P_2 \\ off_2 =& (1 - a_3) \cdot P_1 + a_3 \cdot P_2 \end{aligned} \tag{16}$$

where v_i^t denotes the velocity of the ith individual in the tth generation, x_i^t, y_i^t subtables denote the positions of male and female mayflies in the tth generation, $pbest_i$, $gbset_i$ denote the historical optimal position and global optimal position of male mayflies, r_p, r_g, r_m denote the distance of the individual from $pbest_i.gbest_i$ and the corresponding male individual, d represents the dance coefficient, $r \in [-1, 1]$ is the random number, β denotes the visibility coefficient, and l_{fl} is the random wandering coefficient. a_1, a_2 are the number of attraction for male mayflies to move and a_3 is the random number satisfying the Gaussian distribution.

4. Encoding, Decoding, and Population Initialization

4.1. Encoding

The type-2 IPPS contains two subproblems, process planning and shop scheduling, where the shop scheduling part can be regarded as a flexible job–shop scheduling problem (FJSP). For this subproblem, this paper uses the two-part coding scheme, which is commonly used by FJSP [21]; it encodes operation sequencing and machine selection with OS strings and MS strings, respectively. Unlike FJSP, the operation number cannot be represented by the number of occurrences of the job number in the OS string because of the uncertainty of the operation order in the type-2 IPPS problem, so two integers are used to represent an operation in OS, and they represent the job number and the operation number, respectively. For the process planning subproblem, this paper deals with the operation selection problem in the encoding process, and the encoding indicates the $s_{i,v}$ values of each OR node of the job $J_1, J_2, \ldots, J_n$ from left to right; for the operation ordering problem of the same job, it is dealt with in the decoding. In summary, the particle coding consists of three parts, namely operation ordering, machine selection, and OR node selection, and the coding length is $L = 2 \times \sum_{i=1}^{n} n_i + \sum_{i=1}^{n} nq_i$, where n_i, nq_i denote the number of operations and OR nodes of J_i, respectively, and Figure 2 is an example of coding.

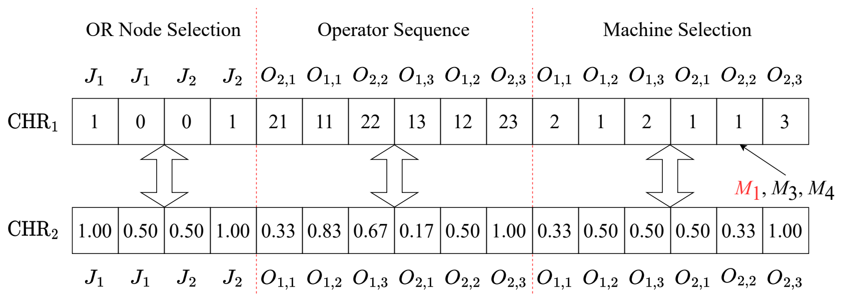

OR Node Selection
Operator Sequence
Machine Selection
CHR$_1$
CHR$_2$
J_1 J_1 J_2 J_2 $O_{2,1}$ $O_{1,1}$ $O_{2,2}$ $O_{1,3}$ $O_{1,2}$ $O_{2,3}$ $O_{1,1}$ $O_{1,2}$ $O_{1,3}$ $O_{2,1}$ $O_{2,2}$ $O_{2,3}$
1 0 0 1 21 11 22 13 12 23 2 1 2 1 1 3
M_1, M_3, M_4
1.00 0.50 0.50 1.00 0.33 0.83 0.67 0.17 0.50 1.00 0.33 0.50 0.50 0.50 0.33 1.00
J_1 J_1 J_2 J_2 $O_{1,1}$ $O_{1,2}$ $O_{1,3}$ $O_{2,1}$ $O_{2,2}$ $O_{2,3}$ $O_{1,1}$ $O_{1,2}$ $O_{1,3}$ $O_{2,1}$ $O_{2,2}$ $O_{2,3}$

Figure 2. Coding example.

The above three-part integer coding can better represent the solution of type-2 IPPS, but it is not suitable for the operation of MA. In this paper, different parts of the coding are mapped into real numbers of the interval $[0, 1]$ in different ways; for the operation ordering part, the purpose of this part of the coding is to determine the processing order of the operation, and the index of each position can indicate the priority of the process it represents, as shown in Figure 2. The operations are processed in the order $O_{2,1}, O_{1,1}, O_{2,2}, O_{1,3}, O_{1,2}, O_{2,3}$, and their indexes are normalized and reordered in the order $O_{1,1}, O_{1,2}, O_{1,3}, O_{2,1}, O_{2,2}$, and the new encoding can still represent the processing order of each operation; for the machine selection and OR node selection parts, the interval $[0, 1]$ is equipartitioned and the corresponding equipartition points can be removed, so that the trivial integer encoding is mapped to a real vector on $[0, 1]$, and these two encoding methods are noted as CHR$_1$, CHR$_2$. Since they have their own advantages, this paper will use both schemes and apply them to different parts of the algorithm.

4.2. Decoding

The decoding process in this paper is performed for CHR$_1$, and for CHR$_2$, it is transformed into CHR$_1$ and then decoded. The process planning part of CHR$_1$ only solves the determination problem of the operation to be processed. For the second problem, operation ordering of the same job, it is only necessary to generate a binary tree based on the priority constraint matrix of the job, and finally traverse this binary search tree in the middle order to obtain the correct process ordering.

The main steps of decoding are as follows:

1. Pre-processing. First determine the selection of each operation based on the OR node selection part, and then determine the correct operation processing order for each job based on the job priority constraint matrix.
2. Determine the processing machine for the operation. The machine selection part of the code is read in turn to determine the processing machine for each operation and to obtain the corresponding processing time for each operation.
3. Generate feasible scheduling. According to the operation sequencing part of the code, the processing order and start time of each operation on the corresponding machine are obtained. In this paper, we use the greedy interpolation method [21] to shorten the processing time of the job, i.e., we traverse all the free time windows of that machine during the decoding process and assign the process to the time window that can be processed earlier.

4.3. Population Initialization

The quality of the initial solution has an impact on the solving ability of the intelligent algorithm [22,23]. In order to improve the quality of the initial solution under the condition of ensuring the distributivity of the initial solution, this paper adopts a hybrid initialization strategy to generate the initial solution, and mainly adopts the following three initial strategies.

- Global minimum machine load strategy. The load of each machine is recorded in the process of coding, and the machine with the minimum current load in the set of optional machines is assigned to the process.
- Maximum remaining processing time strategy. Prioritize operations with longer processing times.
- Shortest machining path strategy. It is easy to see that the processing path length of a job is determined by the selection of OR nodes, and the processing path of the job is shorter when the OR nodes select the decision variables with fewer occurrences in F^0, F^1. In this paper, the shortest machining path strategy is proposed based on the above analysis.

The combination of the above three strategies and the random initialization strategy can be used to improve the quality of the initial solution in an effective way, and the selection probabilities of the above three strategies in this paper are 0.78, 0.6, and 0.2, respectively.

5. Improved Mayfly Optimization Algorithm for Type-2 MOIPPS

The mayfly algorithm has a strong optimization-seeking capability, but the global exploitation capability is poor and easily falls into a local optimum. Secondly, the mayfly algorithm is an optimization algorithm designed for single-objective optimization problems, which is not suitable for solving multi-objective optimization problems. First, this paper introduces the decomposition idea in the mayfly algorithm to deal with the convergence and distribution of balanced solutions for multi-objective problems, making the solution set uniformly close to the true Pareto front, and using an external archive to store non-dominated solutions. Then, the neighborhood structure is improved to adapt it to the multi-objective optimization problem. The adaptive neighborhood search cycle is designed based on the average distance from the non-dominated solution set to the ideal point, which further improves the algorithm's search capability on the one hand and considers the diversity of the equilibrium population on the other.

5.1. MA with Decomposition Idea

In order to balance the convergence and distributivity of the solution, this paper introduces the decomposition idea in MA to make the solution uniformly close to the real Pareto front. In this paper, the penalty-based boundary intersection (PBI) [24] is used to perform the decomposition, which is calculated as follows:

$$\begin{aligned} \min \quad & g^{(pbi)}(\boldsymbol{x} \mid \boldsymbol{\omega}, \boldsymbol{z}^*) = d_1 + \theta d_2 \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{D} \end{aligned} \tag{17}$$

where $\boldsymbol{\omega}$ is a weight vector, $\boldsymbol{z}^*$ is the ideal point, $d_1 = \frac{\|(f(x)-z^*)\cdot\omega\|}{\|\omega\|}$, $d_2 = \|\boldsymbol{f}(\boldsymbol{x}) - \left(\boldsymbol{z}^* + d_1\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}\right)\|$.

Figure 3 explains the principle of the approach as an example of the dual-objective minimization problem, which shows that d_1 controls the convergence of the solution, d_2 controls the distributivity of the solution, and $\theta \geq 0$ is a pre-set penalty parameter to balance d_1 and d_2.

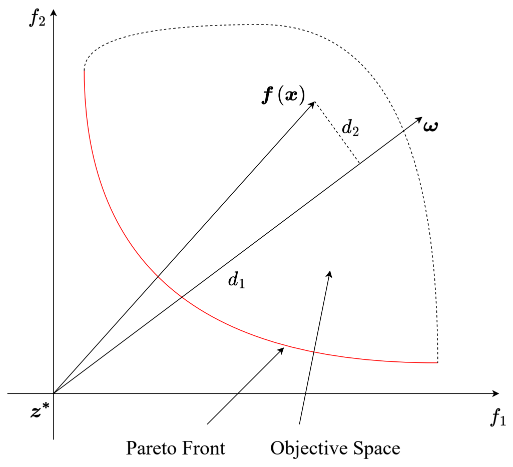


Figure 3. Principle of PBI approach.

The specific steps of MA with the decomposition idea are as follows:

1. Initialization. First generate the initial population of size N and initialize the ideal point as the current population ideal point z according to the hybrid initialization strategy in Section 4.3. Then generate N weight vectors uniformly dividing the target space and determine the neighborhood for each weight vector [25]. Next, associate a weight vector to each individual, and finally initialize the external archive EP $= \varnothing$.
2. Update the mayfly position. Update the male mayfly position according to Equations (12) and (13) and the female mayfly position according to Equations (14) and (15). And accept the new position according to Equation (17), and update the ideal point $\boldsymbol{z}$ whenever the position of a mayfly is updated.
3. Crossover of male and female mayflies. For each female mayfly, randomly select a weight vector from its associated neighbors of the weight vector, find its associated male mayfly as the crossover object for crossover, and accept offspring individuals according to Equation (17). The specific method of crossover is given in Section 5.3.
4. Update the external archive EP. For each individual generated in step 3, determine whether it is dominated by an individual ND_i in the EP; if $\mathrm{ND}_i \prec \mathrm{C}$, delete C, and otherwise, add C to the EP and delete the individual dominated by C in the EP.

5.2. Matching Male and Female Mayflies

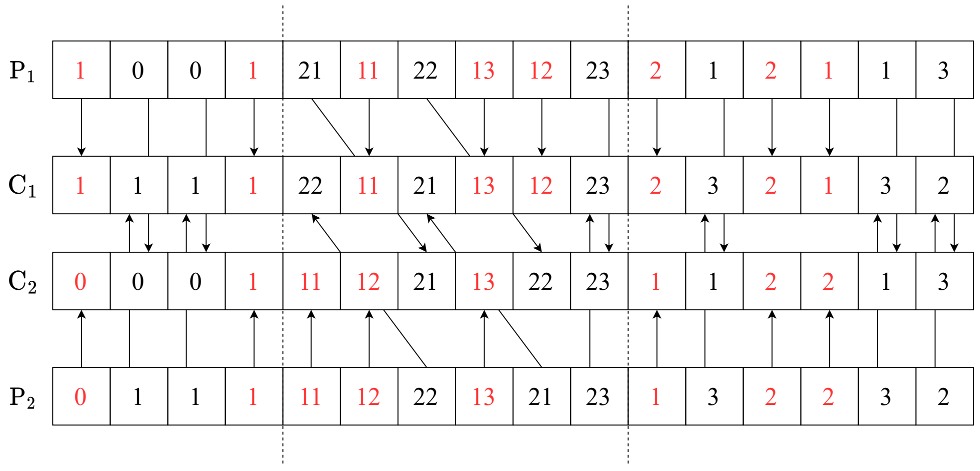
Matching operation is needed before the crossover operation of female and male individuals, and the standard MA algorithm models this process as a deterministic one. In this paper, matching will be performed based on the set of weight vectors, and the main idea is to view the neighborhood of the weight vector associated with the current female mayfly. A random selection from that domain picks a weight vector and uses the male individual associated with it as the match. The specific steps of the process are as follows:

- Associate a weight vector for each female and male.
  The population first needs to be normalized, and then a weight vector with the shortest vertical distance is associated with each individual. Here we do not associate an individual with each weight vector, which means that some weight vectors are not used, because the PF of the type-2 MOIPPS is irregular, and the weight vectors uniformly distributed in the target space cannot make the solution set uniformly close to the PF. This approach can to a certain extent discard the weight vectors that are sparsely distributed in the population, so that the solution set can be relatively uniformly distributed on the PF.
- For each female mayfly, a male mayfly is matched.
  First determine the weight vector $\boldsymbol{w}$ associated with this female mayfly, and then put the male individuals associated with the weight vector in the $\boldsymbol{w}$ neighborhood into

the candidate set $\mathbb{C}$. A male individual from $\mathbb{C}$ is randomly selected as a match for that female individual.

5.3. Mating of Mayflies

According to the characteristics of the type-2 MOIPPS, different crossover approaches are used for each part in this paper. For the OR node selection and machine selection parts, random-point preservation crossover (RPX) is used, and for the operation sequencing part, improved precedence preserving order-based crossover (IPOX) [26] is used. The specific processes of the two crossover approaches are as follows, and Figure 4 shows an example of the crossover process.



P1 | 1 | 0 | 0 | 1 | 21 | 11 | 22 | 13 | 12 | 23 | 2 | 1 | 2 | 1 | 1 | 3
C1 | 1 | 1 | 1 | 1 | 22 | 11 | 21 | 13 | 12 | 23 | 2 | 3 | 2 | 1 | 3 | 2
C2 | 0 | 0 | 0 | 1 | 11 | 12 | 21 | 13 | 22 | 23 | 1 | 1 | 2 | 2 | 1 | 3
P2 | 0 | 1 | 1 | 1 | 11 | 12 | 22 | 13 | 21 | 23 | 1 | 3 | 2 | 2 | 3 | 2

Figure 4. Example of crossover approach.

RPX: For parents P $_1$, P$_2$, first generate a set of random numbers $\mathbb{R}$ in $[0, L]$; then the encoded positions of parents P$_1$, P$_2$ in $\mathbb{R}$ are are kept in the children C$_1$, C$_2$, and the positions that are not in $\mathbb{R}$ are inserted into the empty positions of C$_2$, C$_1$ in turn.

IPOX: For the parents P$_1$, P$_2$, first divide the artifact set $\mathbb{J}$ into two non-empty complementary sets $\mathbb{J}_1, \mathbb{J}_2$ at random; then divide the parents P$_1$, P$_1$, P$_2$ in the set $\mathbb{J}_1$ into C$_1$, C$_2$ in their original position; then put the parent P$_1$, P$_2$ in the set $\mathbb{J}_1$ in the corresponding process order; then put the parent P$_1$, P$_2$ in the set $\mathbb{J}_2$ into the empty spaces of C$_2$ and C$_1$ in order of the corresponding process order of the artifacts.

5.4. Neighborhood Structure Based on Moving One Critical Operation

Since moving non-critical processes cannot reduce the makespan, the neighborhood structure in this paper is constructed based on moving the critical operation. There are two options for moving an operation: (1) moving an operation with the same machine, and (2) moving an operation across machines. The three optimization objectives of makespan, total machine load, and key machine load are considered simultaneously, and since the total machine load or critical machine load cannot be changed by moving the operation with the same machine, the neighborhood structure is constructed by moving the critical operation across machines. To better describe this neighborhood structure, some necessary definitions are first introduced. For operation w, r_w, t_w are the head time and tail time of w, which denote the length of the longest path from S to w and the length of the longest path from w to F in the parse diagram, respectively; the job precursors and job successors of w are $JP[w], JS[w]$, and the machine precursors and machine successors of w are $MP[w], MS[w]$, respectively; the earliest start time and the latest start time of w are $ES(w), EC(w)$, and the earliest completion time and the latest completion time of w are $LS(w), LC(w)$, respectively. The neighborhood structure of this paper is based on the following theorem [27]:

Theorem 2 (Conditions for generating an improved neighborhood solution)**.** *Move the critical operation w to process between operations i, j on the optional machine m. The generated* *neighborhood solution is still feasible and has a shorter makespan or a smaller number of critical paths if the following conditions are also satisfied:*

1. $r_i < r_{JS[w]} + p_{JS[w]}$;
2. $r_j + p_j > r_{JP[w]}$;
3. $\min\{LS(j), LS(JS[w])\} - \max\{EC(i), EC(JP[w])\} > p_{wm}$.

In the above, (1) nd (2) ensure that the neighborhood solution is feasible and (3) ensures that the neighborhood solution is improved.

The following neighborhood structure is designed based on Theorem 2:

1. For a feasible schedule, calculate its critical path and find all insertable positions that satisfy Theorem 2 for each critical operation and place them into set $\mathbb{P}$ [28].
2. For each critical operation, the positions in the set $\mathbb{P}$ are arranged in ascending order of processing time.
3. For each critical operation, insert it into the first position in the sorted set $\mathbb{P}$.

It can be seen that the above neighborhood structure can improve both the makespan and the total machine load, especially the makespan, which is the most difficult objective to optimize among the three objectives, and it has been shown that minimizing makespan helps to optimize key machine load [29].

5.5. Adaptive Neighborhood Search Period

The improved neighborhood structure based on moving a critical operation can effectively improve the three optimization objectives of makespan, total machine load, and critical machine load, and promote the convergence speed of the algorithm as well as the search capability. As the number of iterations of the algorithm increases, the focus of the algorithm will shift from fast convergence to solution diversity, and it is more appropriate to set the local search period to a larger value at this time. In this paper, the adaptive local search period is designed based on the average distance from the current external archive EP to the ideal point z:

$$\mathrm{T} = \mathrm{iter\_num} \cdot e^{-\frac{1}{|EP|}\sum_{x\in \mathrm{EP}} \|f(x)-z\|} \tag{18}$$

where iter_num denotes the number of iterations and $\|\cdot\|$ denotes the Euclidean distance.

5.6. Game-Theoretic-Based Scheduling Decision

The Pareto front PF is obtained based on the improved algorithm for the type-2 MOIPPS solution. In real-life production, it is also necessary to make a satisfactory scheduling solution from PF, and this paper will make a decision on PF based on game theory [30]. A complete game model consists of three parts: game party, game strategy, and benefit function. In this paper, the three minimization objectives of the type-2 MOIPPS are regarded as the game party, the variable space as the decision space, and the objective function as the benefit function to establish the non-cooperative game model. Nash equilibrium, as an important element of non-cooperative game theory, can make each objective value as close to optimal as possible without compromising the interests of other objectives. In this paper, the optimal Nash equilibrium solution is used as the decision result of the Pareto solution set, which is defined as follows:

$$\mathrm{Nash} = \min_{d_\in \mathrm{PF}} \left(\sum_{i=1}^{3} \frac{f_{id} - best_i}{best_i} \right) \tag{19}$$

where $best_i$ denotes the optimal value of the ith objective in the PF.

5.7. Algorithm Flow

The improved algorithm can be viewed as a two-population interactive evolution process, where the internal population POP is updated iteratively using a mayfly algorithm based on the decomposition idea, and the external population EP stores the non-dominated solutions in the POP, which is updated using adaptive neighborhood search and influences the internal population through the mayfly location update method, where the internal population POP has better diversity and the external population EP ensures faster convergence of the algorithm. The overall flow of the algorithm is shown in Figure 5.

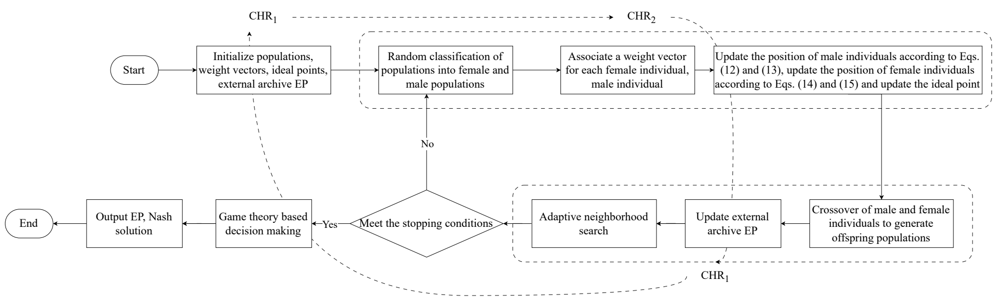

CHR$_1$
CHR$_2$
Start
Initialize populations, weight vectors, ideal points, external archive EP
Random classification of populations into female and male populations
Associate a weight vector for each female individual, male individual
Update the position of male individuals according to Eqs. (12) and (13), update the position of female individuals according to Eqs. (14) and (15) and update the ideal point
No
End
Output EP, Nash solution
Game theory based decision making
Yes
Meet the stopping conditions
Adaptive neighborhood search
Update external archive EP
Crossover of male and female individuals to generate offspring populations
CHR$_1$

Figure 5. Overall flow of the algorithm.

6. Experimental Analysis

6.1. Experimental Environment and Experimental Data

Experiments are programmed in Python 3.11.4 (Numpy 1.23.4, Matplotlib 3.6.1) and run on an Arch Linux operating system with a CPU of 12th Gen Intel i9-12900H 4.900 GHz and 16G of RAM. Experimental data 1 are from Shao et al. [31], which contains six jobs and eight machines; experimental data 2 are presented by Kim et al. [5], which contains eighteen jobs and fifteen machines and can be divided into 24 cases of different scales. In this paper, Kim1, Kim11, Kim18, and Kim24 are selected as cases of different sizes.

In order to verify the feasibility and effectiveness of the algorithm, this paper will compare both single-objective and multi-objective aspects. Since minimizing makespan is a common optimization metric for studying scheduling problems and is more difficult to optimize than other metrics, this objective is selected for the comparison of single-objective experimental results. For the comparison of multi-objective optimization results, this paper compares the scheduling decision results and the Pareto optimal solution ratio, which is described in Section 6.3.

6.2. Experimental Parameter Setting

After several experiments and comparisons, the parameters in this paper are set as follows: $\beta = 2, a_1 = 1.2, a_2 = 1.6, l_{fl} = 1, \theta = 5$.

The main parameter settings, which are the same as those of NSGA-2 and MOEA/D algorithms, are shown in Table 2.

Table 2. Experimental parameter setting.

Instance	Population Size	Maximum Number of Iterations	Number of Weight Vector Neighbors
6×8	100	100	10
Kim1	100	100	10
Kim11	200	200	20
Kim18	300	300	20
Kim24	500	500	20

6.3. Evaluation Indicator

In order to compare the goodness of the Pareto front computed by each algorithm, this paper uses the Pareto optimal solution ratio (AR) [32] as the evaluation index of MOP.

AR denotes the proportion of the new PFs produced by merging the PFs of two algorithms that are occupied by the non-dominated solutions of the original PF. In general, a larger value of AR indicates a better performance of the algorithm, which is calculated as follows:

$$\mathrm{AR}_{\mathrm{PF}_1} = \frac{|\mathrm{PF}_1 - \{x \in \mathrm{PF}_1 | \exists y \in \mathrm{PF}_2 : y \prec x\}|}{|\mathrm{PF}_1|} \tag{20}$$

6.4. Comparison of Single-Objective Experimental Results

Experiment 1 (6×8)

The scheduling decision solution of IMOMA is compared with the ACO-MPP algorithm proposed by Huang et al. [8], and the Modified GA proposed by Shao et al. [31]. The comparison of the experimental results is shown in Table 3.

Table 3. Comparison of experimental (6×8) results.

Algorithm	Modified GA	ACO-MPP	IMOMA
MK	162	145	131

It can be seen that the results of IMOMA are 19.1% lower than those of Modified GA and 9.7% lower than those of ACO-MPP. Figure 6 shows the optimization of 6 × 8 using the IMOMA algorithm. The scheduling Gantt chart is obtained from the experiment.

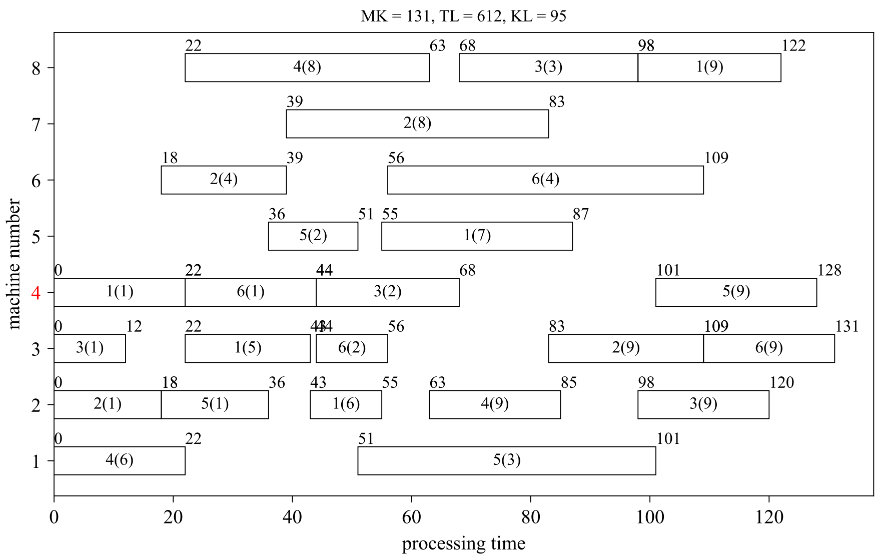


Figure 6. Experiment 6 × 8 scheduling Gantt chart.

Experiment 2 (18×15)

The Nash solution obtained by the algorithm IMOMA in this paper is compared with the THA algorithm proposed by Wen et al. [33], the ACO algorithm proposed by Zhang and Wong [34], and the ACO-MPP algorithm proposed by Huang et al. [8]. The experimental results are as follows (Table 4).

Table 4. Comparison of experimental (18×15) results.

Instance	Number of Jobs	ACO	THA	ACO-MPP	IMOMA
Kim1	6	427	427	N/A	427
Kim11	9	364	365	N/A	347
Kim18	12	378	353	N/A	342
Kim 24	18	525	511	522	492

From the experimental results, it can be seen that IMOMA can achieve better results in medium-scale cases compared to other algorithms, and also has a better finding ability in medium- and large-scale cases.

6.5. Comparison of Multi-Objective Experimental Results

To verify the effectiveness of this paper's algorithm, IMOMA, for optimizing the type-2 MOIPPS, IMOMA is compared with NSGA-2 [35], MOEA/D, and MOMA algorithms, where NSGA-2 and MOEA/D are representative algorithms for solving MOP, which represent two main ideas for solving MOP. Based on Pareto domination and based on the decomposition idea, the MOMA algorithm is formed by extending the MA algorithm to MOP, i.e., the IMOMA algorithm does not use the decomposition idea and improved neighborhood structure. The main parameters of each algorithm are set as the same.

The scheduling decision schemes for each case are given in Table 5. From the table, it can be seen that the IMOMA algorithm is able to generate relatively better solutions, although some solutions do not dominate each other. It can be seen that the results obtained by IMOMA are able to achieve better results in more than two objectives compared to other algorithms. As in the case Kim24, the scheduling solution obtained by IMOMA computation can dominate the results of the NSGA-2 algorithm, and for MOEA/D and MOMA algorithms, the results of the IMOMA algorithm are more advantageous in the optimization of MK.

Table 5. Comparison of decision results by algorithms.

Instance	**NSGA-2**			**MOEA/D**			**MOMA**			**IMOMA**		
	MK	**TL**	**KL**	**MK**	**TL**	**KL**	**MK**	**TL**	**KL**	**MK**	**TL**	**KL**
6×8	137	612	90	136	616	95	142	602	111	131	611	109
Kim1	428	1858	152	430	1837	154	428	1872	173	427	1864	163
Kim11	396	2507	221	403	2486	269	409	2507	201	347	2459	200
Kim18	386	3034	245	375	3089	229	378	3076	258	342	3034	221
Kim24	497	5224	397	520	5143	396	519	5199	372	492	5159	386

Table 6 shows the results of comparing the AR values of IMOMA and each algorithm. It can be seen that in small- and medium-scale cases, multi-objective optimization algorithms such as NSGA-2 and MOEA/D have certain advantages, and the IMOMA algorithm has better results for solving medium- and large-scale cases. To further illustrate the effectiveness of the IMOMA algorithm in this paper, Figures 7 and 8 show the Pareto front comparison of the IMOMA algorithm with the NSGA-2 algorithm, MOEA/D algorithm, and MOMA algorithm, and the scheduling Gantt chart obtained from solving the Kim24 case using the IMOMA algorithm, respectively. From Figure 7, it can be seen that the Pareto frontier obtained by the IMOMA algorithm is superior compared to the other algorithms. The machining path for each job of Kim24 is shown as follows (Table 7).

Table 6. Comparison of the AR value of each algorithm.

Instance	**NSGA-2**	**IMOMA**	**MOEA/D**	**IMOMA**	**MOMA**	**IMOMA**
6×8	0.46	0.83	0.50	0.76	0.35	1.00
Kim1	0.23	0.75	0.42	0.81	0.54	0.93
Kim11	0.18	0.91	0.21	1.00	0.19	1.00
Kim18	0.12	1.00	0.04	1.00	0.02	1.00
Kim24	0.06	0.92	0.03	1.00	0.02	1.00

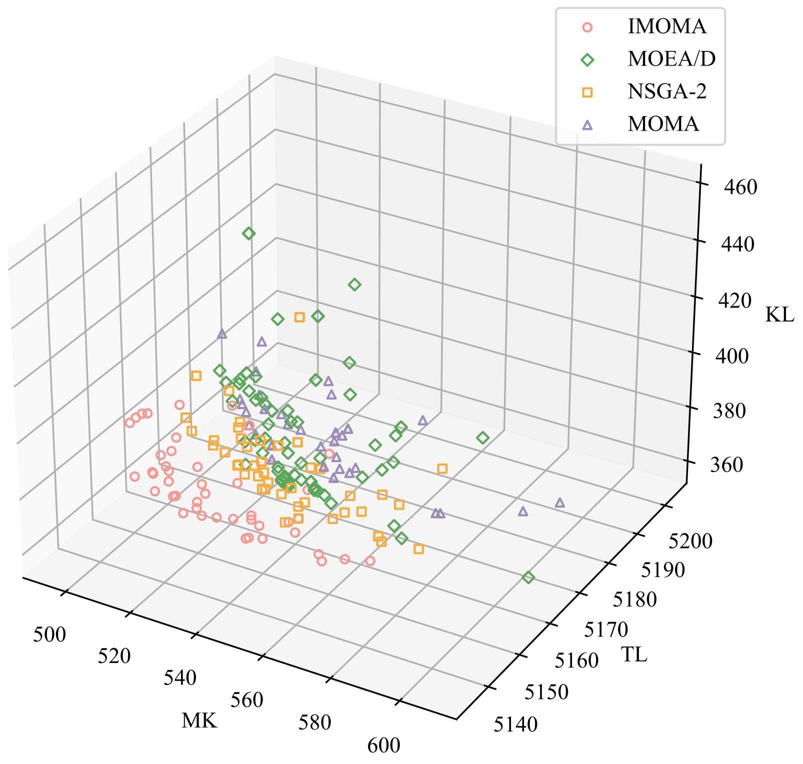


Figure 7. Comparison of different algorithms for the case Kim24 Pareto front.

Table 7. The machining path for each job of Kim24.

J_1	$O_{1,1} \rightarrow O_{1,2} \rightarrow O_{1,3} \rightarrow O_{1,4} \rightarrow O_{1,5} \rightarrow O_{1,6} \rightarrow O_{1,7} \rightarrow O_{1,8}$
J_2	$O_{2,1} \rightarrow O_{2,4} \rightarrow O_{2,2} \rightarrow O_{2,3} \rightarrow O_{2,5} \rightarrow O_{2,7} \rightarrow O_{2,8} \rightarrow O_{2,12} \rightarrow O_{2,13} \rightarrow O_{2,14} \rightarrow O_{2,6}$
J_3	$O_{3,16} \rightarrow O_{3,7} \rightarrow O_{3,13} \rightarrow O_{3,14} \rightarrow O_{3,15} \rightarrow O_{3,17} \rightarrow O_{3,18} \rightarrow O_{3,19} \rightarrow O_{3,1} \rightarrow O_{3,5} \rightarrow O_{3,6} \rightarrow O_{3,8} \rightarrow O_{3,9} \rightarrow O_{3,10} \rightarrow O_{3,11} \rightarrow O_{3,12} \rightarrow O_{3,2} \rightarrow O_{3,3} \rightarrow O_{3,4}$
J_4	$O_{4,13} \rightarrow O_{4,16} \rightarrow O_{4,1} \rightarrow O_{4,5} \rightarrow O_{4,6} \rightarrow O_{4,7} \rightarrow O_{4,8} \rightarrow O_{4,9} \rightarrow O_{4,10} \rightarrow O_{4,11} \rightarrow O_{4,12}$
J_5	$O_{5,14} \rightarrow O_{5,15} \rightarrow O_{5,16} \rightarrow O_{5,1} \rightarrow O_{5,10} \rightarrow O_{5,11} \rightarrow O_{5,12} \rightarrow O_{5,13} \rightarrow O_{5,17} \rightarrow O_{5,18}$
J_6	$O_{6,5} \rightarrow O_{6,6} \rightarrow O_{6,7} \rightarrow O_{6,15} \rightarrow O_{6,17} \rightarrow O_{6,18} \rightarrow O_{6,16} \rightarrow O_{6,19} \rightarrow O_{6,20} \rightarrow O_{6,1} \rightarrow O_{6,2} \rightarrow O_{6,3} \rightarrow O_{6,4} \rightarrow O_{6,10} \rightarrow O_{6,9} \rightarrow O_{6,14}$
J_7	$O_{7,1} \rightarrow O_{7,7} \rightarrow O_{7,19} \rightarrow O_{7,20} \rightarrow O_{7,21} \rightarrow O_{7,8} \rightarrow O_{7,9} \rightarrow O_{7,10} \rightarrow O_{7,11} \rightarrow O_{7,12} \rightarrow O_{7,13} \rightarrow O_{7,18}$
J_8	$O_{8,1} \rightarrow O_{8,2} \rightarrow O_{8,3} \rightarrow O_{8,4} \rightarrow O_{8,5} \rightarrow O_{8,7} \rightarrow O_{8,8} \rightarrow O_{8,10} \rightarrow O_{8,11} \rightarrow O_{8,17} \rightarrow O_{8,19} \rightarrow O_{8,20}$
J_9	$O_{9,1} \rightarrow O_{9,2} \rightarrow O_{9,13} \rightarrow O_{9,4} \rightarrow O_{9,5} \rightarrow O_{9,6} \rightarrow O_{9,14} \rightarrow O_{9,15} \rightarrow O_{9,16} \rightarrow O_{9,18} \rightarrow O_{9,19} \rightarrow O_{9,20} \rightarrow O_{9,7} \rightarrow O_{9,10} \rightarrow O_{9,11} \rightarrow O_{9,12}$
J_{10}	$O_{10,1} \rightarrow O_{10,2} \rightarrow O_{10,3} \rightarrow O_{10,4} \rightarrow O_{10,5} \rightarrow O_{10,6} \rightarrow O_{10,9} \rightarrow O_{10,10} \rightarrow O_{10,11}$
J_{11}	$O_{11,8} \rightarrow O_{11,9} \rightarrow O_{11,6} \rightarrow O_{11,7} \rightarrow O_{11,1} \rightarrow O_{11,2} \rightarrow O_{11,3} \rightarrow O_{11,4} \rightarrow O_{11,5}$
J_{12}	$O_{12,1} \rightarrow O_{12,5} \rightarrow O_{12,2} \rightarrow O_{12,3} \rightarrow O_{12,4} \rightarrow O_{12,8} \rightarrow O_{12,9} \rightarrow O_{12,10} \rightarrow O_{12,11} \rightarrow O_{12,12} \rightarrow O_{12,6} \rightarrow O_{12,7} \rightarrow O_{12,13} \rightarrow O_{12,14} \rightarrow O_{12,15} \rightarrow O_{12,18}$
J_{13}	$O_{13,17} \rightarrow O_{13,18} \rightarrow O_{13,1} \rightarrow O_{13,12} \rightarrow O_{13,13} \rightarrow O_{13,14} \rightarrow O_{13,15} \rightarrow O_{13,16}$
J_{14}	$O_{14,9} \rightarrow O_{14,10} \rightarrow O_{14,11} \rightarrow O_{14,12} \rightarrow O_{14,13} \rightarrow O_{14,1} \rightarrow O_{14,2} \rightarrow O_{14,3} \rightarrow O_{14,5} \rightarrow O_{14,6} \rightarrow O_{14,8}$
J_{15}	$O_{15,1} \rightarrow O_{15,3} \rightarrow O_{15,12} \rightarrow O_{15,14} \rightarrow O_{15,13} \rightarrow O_{15,7} \rightarrow O_{15,15} \rightarrow O_{15,8} \rightarrow O_{15,9} \rightarrow O_{15,4} \rightarrow O_{15,5} \rightarrow O_{15,6} \rightarrow O_{15,11}$
J_{16}	$O_{16,1} \rightarrow O_{16,6} \rightarrow O_{16,7} \rightarrow O_{16,8} \rightarrow O_{16,9} \rightarrow O_{16,10} \rightarrow O_{16,11} \rightarrow O_{16,15} \rightarrow O_{16,16} \rightarrow O_{16,17} \rightarrow O_{16,18} \rightarrow O_{16,20} \rightarrow O_{16,21}$
J_{17}	$O_{17,18} \rightarrow O_{17,19} \rightarrow O_{17,20} \rightarrow O_{17,22} \rightarrow O_{17,1} \rightarrow O_{17,2} \rightarrow O_{17,3} \rightarrow O_{17,4} \rightarrow O_{17,5} \rightarrow O_{17,7} \rightarrow O_{17,12} \rightarrow O_{17,13} \rightarrow O_{17,15} \rightarrow O_{17,16} \rightarrow O_{17,17}$
J_{18}	$O_{18,12} \rightarrow O_{18,1} \rightarrow O_{18,4} \rightarrow O_{18,7} \rightarrow O_{18,5} \rightarrow O_{18,6} \rightarrow O_{18,8} \rightarrow O_{18,10} \rightarrow O_{18,11} \rightarrow O_{18,14} \rightarrow O_{18,15} \rightarrow O_{18,16} \rightarrow O_{18,17}$

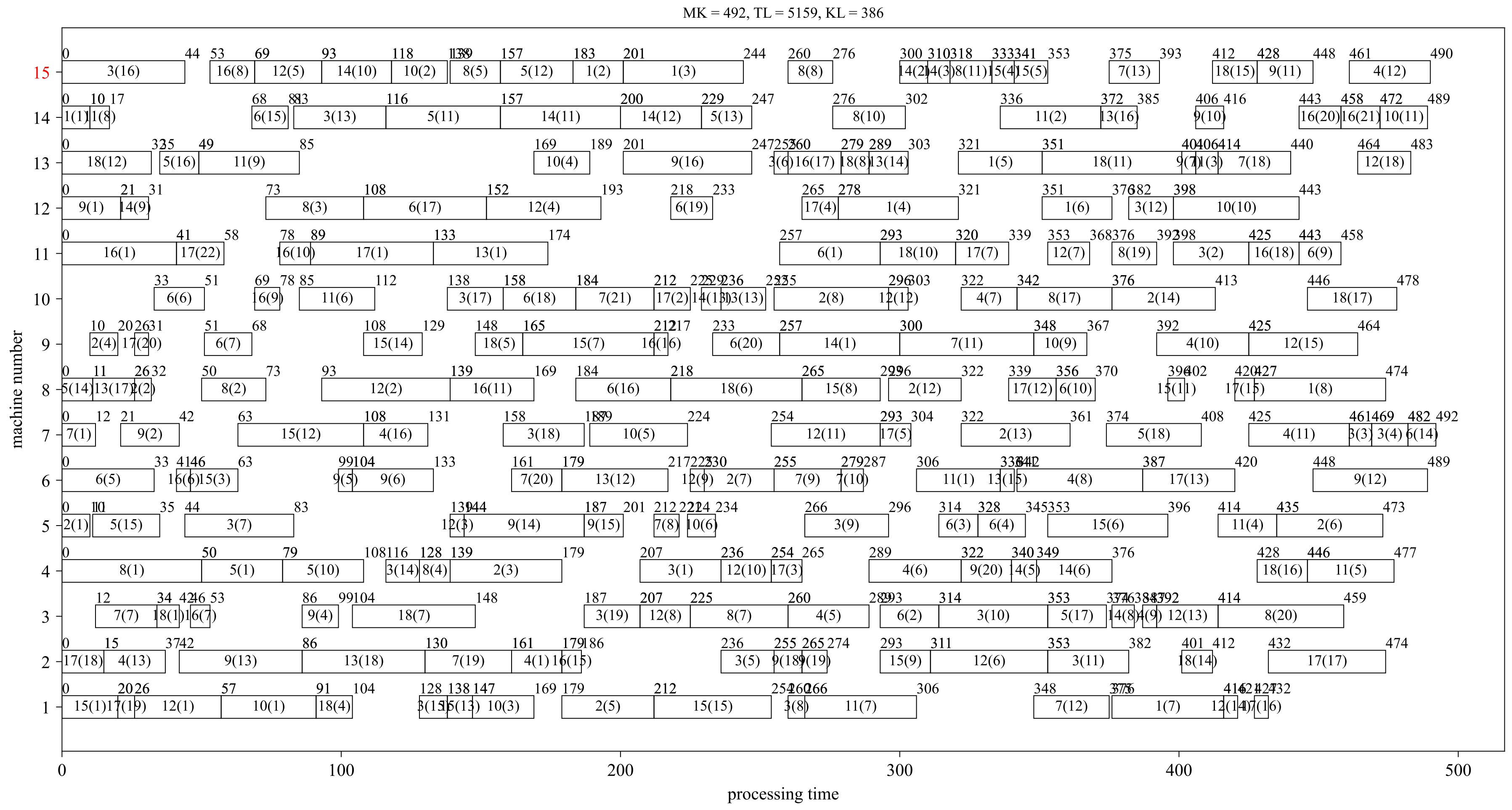


Figure 8. Kim24 scheduling Gantt chart.

7. Conclusions

In this paper, a mixed-integer programming model is developed for the type-2 MOIPPS, and a conversion strategy for two coding schemes is designed so that the two schemes can be transformed into each other during the evolution of the algorithm, and a mixed population initial strategy is designed to improve the initial solution quality considering the diversity of initial solutions. The mayfly algorithm is extended to solve the MOP problem, and the decomposition idea is introduced to balance the distribution and convergence of the non-dominated solution set. The neighborhood structure based on the mobile key process is improved to fit the three optimization objectives selected in this paper, and the adaptive local search cycle is designed based on the average distance between the non-dominated solution set and the ideal point to improve the algorithm's ability to find the best while considering the diversity of solutions.

The type-2 MOIPPS studied in this paper is a deterministic scheduling problem, which is often accompanied by many uncertainties in the actual production process, such as machine failure, emergency order insertion, etc. In addition, workers have a large impact on the actual production. The authors will further explore the type-2 dynamic multi-objective integrated process planning and scheduling problem under dual resource constraints based on the research in this paper and analyze the convergence and complexity of the proposed algorithm.

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

Funding: This research was funded by National Natural Science Foundation of China grant number 11871059 and Natural Science Foundation of Sichuan Education Department grant number 18ZA0469.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data that can reproduce the results in this study can be requested from the corresponding author.

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

References

1. Chryssolouris, G.; Chan, S.; Cobb, W. An integrated apporach to process planning and scheduling. *CIRP Ann.* **1985**, *1*, 315–319.
2. Tan, W.; Khoshnevis, B. Integration of process planning and scheduling—A review. *J. Intell. Manuf.* **2000**, *11*, 51–63. [CrossRef]
3. Jin, L.; Tang, Q.; Zhang, C.; Shao, X.; Tian, G. More MILP models for integrated process planning and scheduling. *Int. J. Prod. Res.* **2016**, *54*, 4387–4402.
4. Chryssolouris, G.; Chan, S.; Cobb, W. Decision making on the factory floor: An integrated approach to process planning and scheduling. *Robot. Comput.-Integr. Manuf.* **1984**, *1*, 315–319.
5. Kim, Y.K.; Park, K.; Ko, J. A symbiotic evolutionary algorithm for the integration of process planning and scheduling. *Comput. Oper. Res.* **2003**, *30*, 1151–1171.
6. Naderi, B.; Begen, M.A.; Zaric, G.S. Type-2 integrated process-planning and scheduling problem: Reformulation and solution algorithms. *Comput. Oper. Res.* **2021**, *142*, 105728.
7. Ausaf, M.F.; Gao, L.; Li, X.; Al Aqel, G. A priority-based heuristic algorithm (PBHA) for optimizing integrated process planning and scheduling problem. *Cogent Eng.* **2015**, *2*, 1070494.
8. Huang, X.; Zhang, X.; Ai, Y. ACO integrated approach for solving flexible job-shop scheduling with mulitple process plans. *Comput. Integr. Manuf. Syst.* **2018**, *24*, 558–569.
9. Zhang, S.; Wong, N.T. Flexible job-shop scheduling/rescheduling in dynamic environment: A hybrid MAS/ACO approach. *Int. J. Prod. Res.* **2016**, *55*, 3173–3196. [CrossRef]
10. Wu, X.; Li, J. Two layered approaches integrating harmony search with genetic algorithm for the integrated process planning and scheduling problem. *Comput. Ind. Eng.* **2021**, *155*, 107194.
11. Xuan, D.U.; Wang, A.; Pan, Z. Clustering and differential evolution algorithm for solving multi-objectives IPPS problem. *Comput. Integr. Manuf. Syst.* **2019**, *25*, 1729–1738.

12. Li, X.; Gao, L.; Li, W. Application of game theory based hybrid algorithm for multi-objective integrated process planning and scheduling. *Expert Syst. Appl.* **2012**, *39*, 288–297. [CrossRef]
13. Mohapatra, P.; Benyoucef, L.; Tiwari, M.K. Integration of process planning and scheduling through adaptive setup planning: A multi-objective approach. *Int. J. Prod. Res.* **2012**, *51*, 7190–7208. [CrossRef]
14. Shokouhi, E. Integrated multi-objective process planning and flexible job shop scheduling considering precedence constraints. *Prod. Manuf. Res.-Open Access J.* **2018**, *6*, 61–89. [CrossRef]
15. Wen, X.; Lian, X.; Qian, Y.; Zhang, Y.; Wang, H.; Li, H. Dynamic scheduling method for integrated process planning and scheduling problem with machine fault. *Robot. Comput.-Integr. Manuf.* **2022**, *77*, 102334. [CrossRef]
16. Zervoudakis, K.; Tsafarakis, S. A mayfly optimization algorithm. *Comput. Ind. Eng.* **2020**, *145*, 106559. [CrossRef]
17. Zou, A.; Wang, L.; Li, W.; Cai, J.; Wang, H.; Tan, T. Mobile robot path planning using improved mayfly optimization algorithm and dynamic window approach. *J. Supercomput.* **2022**, *79*, 8340–8367. [CrossRef]
18. An D.; Yang, D.Y.; Wu, W.L.; Cai, W.; Li, H.; Yang, B.; Han, Y. Optimal location and sizing of battery energy storage systems in a distribution network based on a modified multiobjective mayfly algorithm. *Power Syst. Prot. Control* **2022**, *50*, 31–39.
19. Zhang, D.; Wang, Y.; Zou, C.; Zhao, P.; Zhang, L. Resource allocation strategies for improved mayfly algorithm in cognitive heterogeneous cellular network. *J. Commun.* **2022**, *43*, 156–167.
20. Yuan, Y.; Xu, H.; Wang, B.; Zhang, B.; Yao, X. Balancing Conver- gence and Diversity in Decomposition-Based Many-Objective Optimizers. *IEEE Trans. Evol. Comput.* **2016**, *20*, 180–198. [CrossRef]
21. Lou, H.; Wang, X.; Dong, Z.; Yang, Y. Memetic algorithm based on learning and decomposition for multiobjective flexible job shop scheduling considering human factors. *Swarm Evol. Comput.* **2022**, *75*, 101204. [CrossRef]
22. Guo, Y.W.; Li, W.D.; Mileham, A.R.; Owen, G.W. Applications of particle swarm optimisation in integrated process planning and scheduling. *Robot. Comput.-Integr. Manuf.* **2009**, *25*, 280–288. [CrossRef]
23. Hosseinzadeh, M.R.; Heydari, M.; Mazdeh, M.M. Mathematical modeling and two metaheuristic algorithms for integrated process planning and group scheduling with sequence-dependent setup time. *Oper. Res.* **2022**, *22*, 5055–5105. [CrossRef]
24. Zhang, Q.; Li, H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]
25. Tian, Y.; Xiang, X.; Zhang, X.; Cheng, R.; Jin, Y. Sampling Reference Points on the Pareto Fronts of Benchmark Multi-Objective Optimization Problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018.
26. Zhang, G.; Gao, L.; Shi, Y. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Syst. Appl.* **2011**, *38*, 3563–3573. [CrossRef]
27. Huang, X.W.; Chen, S.F.; Zhou, T.Y.; Sun, Y. A new neighborhood structure for solving the flexible job-shop scheduling problem. *Syst. Eng.-Theory Pract.* **2021**, *41*, 2367–2378.
28. Taillard, E.D. Parallel Taboo Search Techniques for the Job Shop Scheduling Problem. *INFORMS J. Comput.* **1994**, *6*, 108–117. [CrossRef]
29. Chiang, T.; Lin, H. Flexible job shop scheduling using a multiobjective memetic algorithm. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence: Proceedings of the 7th International Conference, ICIC 2011, Zhengzhou, China, 11–14 August 2011*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 1, pp. 49–56.
30. Zhang, Y.; Wang, J.; Liu, Y. Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact. *J. Clean. Prod.* **2017**, *167*, 665–679. [CrossRef]
31. Shao, X.; Li, X.; Gao, L.; Zhang, C. Integration of process planning and scheduling—A modified genetic algorithm-based approach. *Comput. Oper. Res.* **2009**, *36*, 2082–2096. [CrossRef]
32. Chutima, P.; Chimklai, P. Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge. *Comput. Ind. Eng.* **2011**, *62*, 39–55. [CrossRef]
33. Wen, X.; Luo, G.; Li, H.; Xiao, Y.; Qiao, D. Two-stage Hybrid Algorithm for Integrated Process Planning and Scheduling Problems. *China Mech. Eng.* **2018**, *29*, 2716–2724.
34. Zhang, L.; Wong, T.N. Solving integrated process planning and scheduling problem with constructive meta-heuristics. *Inf. Sci.* **2016**, *340*, 1–16. [CrossRef]
35. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T.A. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]