Mathematical Biosciences and Engineering

MBE, 20(6): 9781–9817.
DOI: 10.3934/mbe.2023429
Received: 29 December 2022
Revised: 27 February 2023
Accepted: 13 March 2023
Published: 24 March 2023

http://www.aimspress.com/journal/MBE

Research article

Multi-flexible integrated scheduling algorithm for multi-flexible integrated scheduling problem with setup times

Dan Yang, Zhiqiang Xie* and Chunting Zhang

School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150000, China

* **Correspondence:** Email: xiezhiqianghrbust@163.com.

Abstract: To address the multi-flexible integrated scheduling problem with setup times, a multi-flexible integrated scheduling algorithm is put forward. First, the operation optimization allocation strategy, based on the principle of the relatively long subsequent path, is proposed to assign the operations to idle machines. Second, the parallel optimization strategy is proposed to adjust the scheduling of the planned operations and machines to make the processing as parallel as possible and reduce the no-load machines. Then, the flexible operation determination strategy is combined with the above two strategies to determine the dynamic selection of the flexible operations as the planned operations. Finally, a potential operation preemptive strategy is proposed to judge whether the planned operations will be interrupted by other operations during their processing. The results show that the proposed algorithm can effectively solve the multi-flexible integrated scheduling with setup times, and it can also better solve the flexible integrated scheduling problem.

Keywords: multi-flexible integrated scheduling; flexible planning path; flexible sequential operations; flexible manufacturing; setup times

1. Introduction

Scheduling is a critical factor for the efficiency of production manufacturing systems. The main studies of scheduling basically include two important stages: part processing and product assembly. The traditional scheduling approaches usually regard them as independent of each other, such as the flow shop scheduling for mass product production, the job shop scheduling (JSS) for various of types

of small-batch parts or components and the assembly sequence planning (ASP) for final assembly or subassemblies. With the pursuit of the individuation and customization of complex products, make-to-order production has become one of the essential production modes in enterprises. Especially for some complex, large and heavy products such as the heavy machine tool, medical equipment and large military equipment, they are generally customized with single or minimal orders, and JSS and ASP are essential to use to shorten the makespan, reduce the manufacturing cost and improve the production efficiency.

JSS is a scheduling method to generate an optimal and reasonable processing plan for components or parts, according to which the equipment resources are allocated to the processing operations for the efficient production of the enterprise. After the JSS is completed, all parts or components that need to be assembled are scheduled according to the optimal assembly plan generated by the ASP to achieve the final assembly of the product efficiently. As the critical optimization scheduling technologies, the approaches for JSS and ASP are widely studied and applied in production manufacturing.

Since the JSS problem (JSSP) has been proven to be an NP-hard combinatorial optimization problem, it has become one of the critical research subjects in the field of production manufacturing in recent decades, and plenty of approaches have been designed to solve it. Valenzuela-Alcaraz et al. [1] studied the JSS with no-wait operations and defined a timetabling rule by using binary chains to make it automatically optimized during the evolution of the proposed coevolutionary algorithm. Salido et al. [2] proposed a model to analyze the relationship among energy efficiency, robustness and makespan, and they used CPLex to solve the JSSP with machines with different working speeds. Dai et al. [3] proposed an enhanced EDA to solve the JSSP in consideration of transportation constraints, with the aim of reducing energy consumption. Hao et al. [4] proposed multi-objective EDA for the bi-criteria stochastic JSSP. Wang and Wang [5] studied the two-objective JSS model with a hybrid genetic algorithm. Nguyen et al. [6] adopted cooperative coevolution genetic programming to solve the dynamic multi-objective JSSP. Meanwhile, the approaches for ASP (ASPP) have been studied and applied. Wang et al. [7] proposed a dual Q-learning-based algorithm to solve the ASPP with uncertain assembly times. Ying et al. [8] studied ASP in consideration of the physical characteristics of robotic arms in a cyber-physical assembly system. Giorgio et al. [9] proposed online reinforced learning for ASP with interactive guidance systems.

With the improvement of manufacturing technology and manufacturing environments, such as the flexible machining center and the CNC, flexible manufacturing is becoming more and more common, and the research on flexible product scheduling is becoming more critical. As respective extensions of JSS and ASP, flexible JSS (FJSS) and flexible ASP (FASP) are applied to flexible manufacturing, in which an operation has at least one available machine for processing and is allowed to choose any machine among them. The approaches for solving the FASP problems and the FJSS problems are also widely investigated.

Baykasoglu and Madenoglu [10] proposed a greedy randomized adaptive search for dynamic FJSS. Furthermore, Baykasoglu et al. [11] applied the greedy randomized adaptive search procedure for the dynamic FJSS problem by considering preventive maintenance activities. Vital-Soto et al. [12] built a mathematical model and proposed a hybridized bacterial foraging optimization algorithm for the FJSS problem with sequencing flexibility. Gong et al. [13] proposed a hybrid artificial bee colony (ABC) algorithm for the FJSS problem with worker flexibility. Li et al. [14,15] proposed a reinforcement learning-based RMOEA/D and a self-adaptive multi-objective evolutionary algorithm for the FJSS problem with fuzzy processing time. Liu et al. [16] proposed a hierarchical and distributed

architecture to achieve real-time control of the dynamic FJSS and solve the problem by using a double deep Q-network algorithm-based approach. Lei et al. [17] proposed an end-to-end deep reinforcement framework to automatically learn a policy for solving FJSS problems and embedded the local state encountered by using a graph neural network. Du et al. [18] proposed a hybrid multi-objective optimization algorithm for the estimation of distribution and a deep Q-network to solve the FJSS problem; it considers both the maximum completion time and total electricity price. For FASP, Hottenrott et al. [19] studied the flexible assembly layouts in smart manufacturing and provided a framework based on analytical insight and chance-constrained problem formulation. Li et al. [20] proposed an iterative widen heuristic beam search algorithm to minimize the makespan. Finetto et al. [21] proposed a mixed-model sequencing optimization algorithm for fully FASP.

The above studies on FJSS and FASP problems only focus on pure processing and pure assembly, which may lead to a large gap generated by inventory time between the processing stage and the assembly stage during the entire production process. To avoid this problem, flexible assembly JSS (FAJSS) has been put forward and studied to consider the processing operations with some assembly operations simultaneously in flexible manufacturing to further eliminate the gap and shorten the makespan.

Ren et al. [22] proposed an energy-aware hybrid algorithm based on particle swarm optimization (PSO) and a genetic algorithm (GA) to address the multi-objective FAJSS problem and improve production efficiency and minimize energy consumption. Lin et al. [23] studied the FAJSS problem in consideration of tight job constraints and designed a GA-based algorithm to minimize the makespan. Fattahi et al. [24] established a mathematical model and proposed a hybrid algorithm based on PSO and a parallel variable neighborhood search algorithm for the FAJSS problem with the purpose of minimizing the makespan. Wu et al. [25] improved the differential evolution algorithm and applied it to solve the FAJSS problem in a distributed environment. Zhang and Wang [26] designed a constraint programming model and solved the FAJSS problem considering both sequence-dependent setup times and part sharing in a dynamic environment, with the objective of minimizing the maximum completion time of all jobs. Li et al. [27] improved the ABC algorithm and applied it to solve the FAJSS problem with batch-scheduling to reduce switching costs in the job processing stage.

Although the above investigations tend to study assembly operations during the processing stage, the processing operations and the assembly operations of these studies are usually assigned to different machines. Meanwhile, current research rarely considers all assembly operations and never takes the reprocessing operations of subassembly into account. Therefore, to further make up for the above deficiencies, integrated scheduling (IS) has been proposed and studied; the interaction between assembly operations and processing operations is fully considered simultaneously.

Xie et al. [28] investigated an IS problem with no-wait operations and solved it with their proposed hybrid algorithm. Xie et al. [29] proposed a dispatching rule-based algorithm to address the IS problem in consideration of the maintenance of machines. Xie et al. [30] investigated an IS problem in two-workshop and designed an IS algorithm by considering the prescheduling of the root-subtree processes. Xie et al. [31] studied an IS problem with a pre-start device and proposed an idle machine-driven-based algorithm with the objective of minimizing the completion time of products. Zhan et al. [32] proposed an IS algorithm based on the end time driven and processing area priority to solve a two-workshop collaboration IS problem. Gao et al. [33] fully considered the characters of the tree structure and proposed a GA-based hybrid algorithm to solve the IS problem. Wang et al. [34] applied a non-dominated sorting genetic algorithm-II with a hybrid chromosome coding mechanism to solve

the IS problem, aiming to minimize the total production completion time and the total inventory time.

Besides the above investigations, the flexible IS problem (FISP) has been investigated to optimize the scheduling sequence of all operations simultaneously in flexible manufacturing, regardless of whether the operation belongs to the part processing operations, the assembly operations or the assembly reprocessing operations. Some approaches for FISPs are as follows.

Gao et al. [35] presented a GA-based algorithm with remaining work probability selection coding to solve the FISP. Xie et al. [36] proposed a device-driven and essential path-based integrated algorithm to solve the FISP. Xie et al. [37] proposed a device-driven based conflict mediation algorithm to solve the FISP. Lu et al. [38] investigated the order review/release and dispatching rules to solve the FISP by using a simulation approach. Xie et al. [39] proposed a shorten-time-based algorithm to simplify the FISP into an IS problem and solved it with the allied critical path method. Xie et al. [40] adopted the layer priority-based integrated algorithm to solve the FISP in the reverse order.

Actually, the high flexibility characteristic of flexible manufacturing is not only specific to the equipment. There can be more than 15 flexibility types in a flexible manufacturing system [41], such as a flexible variable disturbance [42], a flexible sequence [12], flexible workers [13] and so on. Furthermore, flexible manufacturing with multiple types of concurrent flexibility is called multi-flexible manufacturing. However, most of the current research on FISPs only focuses on machine flexibility. For some complex products, especially those with a treelike structure, due to the complex product structure, the complex constraint relationship between operations and the processing characteristics of materials, alternative planning routes can be chosen to achieve the same processing effect; consequently, the tightness between some processes is enhanced. To take advantage of the flexibility of complex products and reduce the inventory time resulting from separate part processing and assembly, it is necessary to further investigate multi-flexible features for the FISP, which is much closer to the actual enterprise production activities.

Therefore, it is necessary to propose an effective algorithm to optimize the scheduling sequence for the multi-flexible integrated problem (MFISP), to further improve the efficiency and reduce the cost. Thus, we investigate the multi-flexible IS problem with setup times (MFISP), which considers flexible machines, flexible sequential operations, flexible planning paths and operation-dependent setup times. To reduce the complexity of the problem and facilitate the dynamic scheduling of the machines and operations, the machine idle time is treated as the driven time to schedule the processing operation tree in the reverse order.

The remainder of this paper is as follows. In Section 2, the problem description and mathematical model for the MFISP are presented. In Section 3, the design and implementation of four scheduling strategies are given. In Section 4, the implementation of the multi-flexible integrated scheduling algorithm (MFISA) is given, and the complexity analysis of the algorithm is in Section 5. In Section 6, experiments are carried out to test the performance and the efficiency of the proposed algorithm for the MFISP and FISP. Finally, the conclusions and future research work are given.

2. Problem description and mathematical models

2.1. Problem description

To make IS problems more general, the assembly machines and the machining machines are not distinguished according to their functions, but are uniformly called machines. By default, each machine

can have capacities of both assembly and machining. The machining operations and the assembly operations are collectively known as operations. To better describe the MFISP, we introduce the following concepts.

Definition 1. (Flexible planning path group) If different planning paths can be selected from one operation to another, then the set of all of these paths is called the flexible planning path group. The paths are called the flexible planning paths.

Definition 2. (Flexible sequential operation group) If there are non-strict precedence constraints between the operations involved in a path from one operation to another, then the set of the involved operations is called the flexible sequential operation group. The involved operations are called the flexible sequential operations.

Definition 3. (Flexible operation) The operations that belong to the flexible planning path group or the flexible sequential operation group are collectively referred to as the flexible operations. The first operation of a flexible planning path is called the first-position flexible operation.

Definition 4. (Setup time) The waiting time required for the machine on which different types of operations are processed successively is called the setup time.

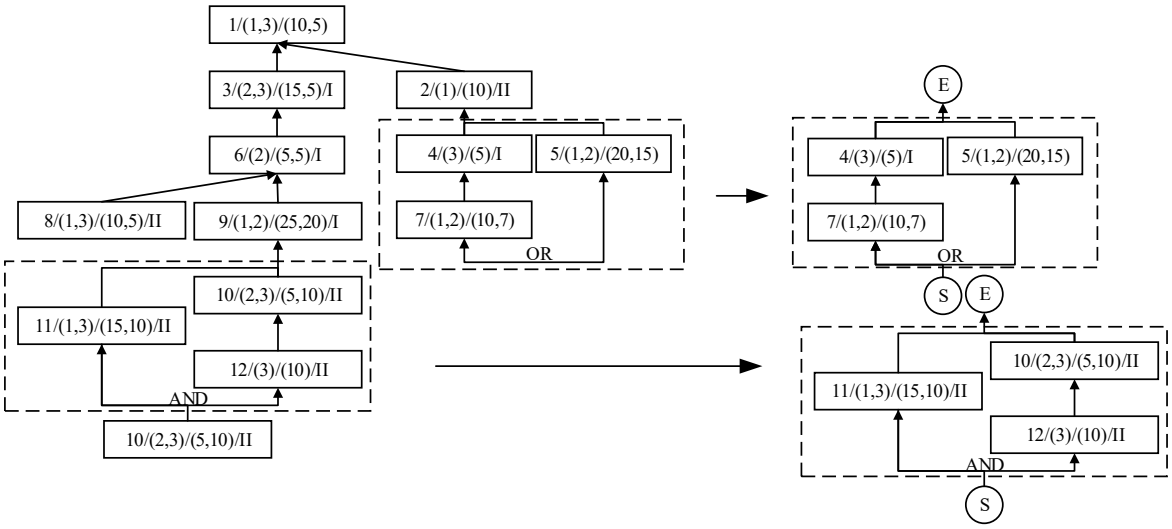


Figure 1. Extensive processing operation tree for the MFISP, AND sub-graph and OR sub-graph.

In the multi-flexible IS system, at least one machine is available for each operation and the processing time required is not necessarily the same for each machine. Each operation can only choose one machine to be processed. The waiting time between different types of operations on the same machine should not be less than the corresponding setup times. And, the flexible planning path groups and the flexible sequential operation groups can be expressed by the OR sub-graphs and the AND sub-graphs on a processing operation tree, respectively, as shown in Figure 1. The node information of a special-type operation becomes four components (i.e., operation number/machine number/processing time/operation type). The operations in the flexible sequential operation groups are for the same workpiece, so there is no overlapping processing time. As different planning paths in a flexible planning path group can achieve the same processing effect, only one planning path can be selected from the same flexible planning path group.

2.2. Mathematical model

Table 1. Symbols and indices.

Type	Symbol	Instruction
Index	i, i', i''	Indices of operations
	k, k'	Indices of machines
	u, u'	Indices of flexible planning paths
	j, j'	Indices of flexible sequential operation groups
	v, v'	Indices of flexible planning path groups
Set	O	Set of operations
	OF	Set of operations on non-flexible planning paths
	R	Set of operation pairs with precedence constraint, $R = \{(i, i')\}$, where operation i is the predecessor of operation i'
	$O_{v,u}$	Set of operations on the u^{th} path of flexible planning path group v
	O_j	Set of operations in flexible sequential operation group j
	F	Set of flexible planning path groups
	F_v	Set of planning paths of flexible planning path group v
	M_i	Set of available machines for operation i
Constant	$P_{i,k}$	Processing time of operation i on machine k
	W	A large enough integer
	$s_{i,k}$	Starting time of operation i on machine k
	$ST_{i,i',k}$	Setup time of machine k to process operation i' after operation i
	$c_{i,k}$	Completion time of operation i on machine k
	P_i	Processing time of operation i
	C_i	Completion time of operation i
Decision variable	S_i	Starting time of operation i
	$a_{i,k}$	$a_{i,k} = 1$ if operation i is selected by machine k. Otherwise, $a_{i,k} = 0$.
	$b_{i,i',k}$	$b_{i,i',k} = 1$if operation i starts earlier than operation i' on machine k. Otherwise, $b_{i,i',k} = 0$.
	$d_{i,i'}$	$d_{i,i'} = 1$ if operation i starts earlier than operation i'. Otherwise, $d_{i,i'} = 0$.
	$r_{v,u}$	$r_{v,u} = 1$ if the u^{th} planning path in planning path group v is selected as the final path. Otherwise, $r_{v,u} = 0$.

Based on the description of the MFISP, the assumptions are as follows.

1) The information on the machines, the processing time, the operations types, the setup times and the precedence constraints between operations is all given.

2) The processing of each operation cannot be interrupted. If it is interrupted, the operation must be reprocessed.

3) The breakdown of machines will not happen, and the migration time of each workpiece between different machines is negligible.

Then, the constraints of the MFISP are as follows.

1) The starting time of each operation must be no earlier than the ending time of its predecessors.

2) The starting time of each operation cannot be earlier than the maximum completion time of the previous operations on the same machine.

3) In each flexible planning path group, only one flexible planning path can be chosen.

4) There must be no overlapping processing times between operations in each flexible sequential operation group.

According to the assumptions and constraints, the established mathematical model and symbols of the MFISP are described as follows, where the symbols and indices are shown in Table 1.

Constraint conditions:

$$\sum_{k\in M_i} a_{i,k}=1 \quad ,\forall i \in OF \tag{1}$$

$$\sum_{k\in M_i} a_{i,k} \le 1 \quad ,\forall i \in O \tag{2}$$

$$P_i = \sum_{k\in M_i} a_{i,k} p_{i,k} \quad ,\forall i \in O \tag{3}$$

$$S_i = \sum_{k\in M_i} a_{i,k} s_{i,k} \quad ,\forall i \in O \tag{4}$$

$$C_i = S_i + P_i \quad ,\forall i \in O \tag{5}$$

$$\sum_{k'\in M_{i'}} a_{i',k'} = \sum_{k\in M_i} a_{i,k} = r_{v,u} \quad ,\forall i,i' \in O_{v,u} \quad ,v \in F \quad ,u \in F_v \tag{6}$$

$$\sum\nolimits_{u\in F_v} r_{v,u}=1 \quad ,\forall v \in F \tag{7}$$

$$d_{i,i'} \ge \sum_{k\in M_i} a_{i,k} + \sum_{k'\in M_{i'}} a_{i',k'} - 1 \quad ,\forall (i,i') \in R \quad ,\forall i,i' \in O \tag{8}$$

$$c_{i,k} \le s_{i',k'} + W(1-d_{i,i'}), k \in M_i, k' \in M_{i'} \quad ,\forall (i,i') \in R \quad ,\forall i,i' \in O \tag{9}$$

$$c_{i,k} + ST_{i,i',k} \le s_{i',k} + W(1-b_{i,i',k}) \quad ,\forall i,i' \in O \tag{10}$$

$$c_{i,k} + ST_{i,i',k} \le s_{i',k} + W(1-d_{i,i'}) \quad ,\forall i,i' \in O_j \tag{11}$$

$$c_{i,k} \ge 0 \quad ,\forall i \in O, k \in M_i \tag{12}$$

$$s_{i,k} \ge 0 \quad ,\forall i \in O, k \in M_i \tag{13}$$

Equation (1) ensures that all non-flexible operations will inevitably be performed on one machine. Equation (2) further illustrates that some operations may not be performed. Equations (1) and (2) reflect the characteristics of route flexibility in the MFISP. Equations (3) and (4) ensure that the processing time and the starting time of each operation are only related to the selected machine. They reflect the characteristics of machine flexibility. Equation (5) illustrates the relationship between the

processing time, the starting time and the completion time of each operation. It reflects the time characteristics of operations. Equation (6) ensures that the operations on any flexible planning path are either all or not processed. Equation (7) ensures that only one flexible planning path can be selected in each flexible planning path group. Equations (6) and (7) reflect the constraint (3). Equations (8) and (9) ensure that each operation must comply with the precedence constraint between operations, that is, each operation can only start after the completion of its immediate predecessor operations. They reflect the constraint (1). Equation (10) prevents the overlapping of operations on each machine with consideration of setup times. It reflects the characteristics of setup times and the constraint (2). Equation (11) ensures that the processing times of the operations in the same flexible sequential operation group cannot overlap, that is, the operations are serial processing. It reflects the constraint (4). Equations (12) and (13) ensure that the completion time and the starting time have practical significance.

This work was aimed to reduce the makespan of the product as the ultimate goal to solve the MFISP. The objective function is shown in Eq (14).

$$\text{Min } C = \{\max\{ C_i \}\},\ i \in O \tag{14}$$

where C indicates the completion time of the product, which is equal to the maximum completion time of all operations.

3. Design and analysis of scheduling strategies

In order to keep the makespans of the complex products in the MFISP as short as possible, it is necessary to select the appropriate machine for each operation on the premise of satisfying the precedence constraints. It is also necessary to consider the choice of the flexible planning paths and the serial processing relationship between the operations in each flexible sequential operation group. Therefore, the algorithm proposed in this paper should consider four requirements: a. Uncouple the flexible operations from the flexible sequential operation groups and the flexible planning paths; b. Reduce the backsliding and the stacking of serial operations on the critical paths; c. Reasonable arrangement of the parallel operations; d. Preemption of busy machines by certain operations. Hence, the algorithm in this paper reduces the complexity of the problem and makes the scheduling procedure dynamic by adopting reverse-order scheduling of the processing operation tree and driving scheduling events at each machine's idle time. The idle moment for a machine is known as the driven time, at which a new scheduling event occurs. Meanwhile, four corresponding scheduling strategies have been designed and adopted: a. Flexible operation determination strategy (FODS); b. Operation optimization allocation strategy (OOAS); c. Parallel optimization strategy (POS); d. Potential operation preemptive strategy (POPS). Among these strategies, the FODS is a supplementary strategy to the dispatching strategies OOAS and POS. Since flexible operations are involved in each round of dispatching, it is necessary to determine the category of the flexible operation so as to determine whether the operation can be set as a planned operation for an optimal planning path or an optimal order. In addition, for the flexible operation of the flexible planning path group, the optimal path and the optimal machine are dynamic, so it is necessary to use the FODS to dynamically judge the operation to match the corresponding machine in order to achieve the current optimal scheduling effect.

3.1. Relevant concepts and definitions

Definition 5. (Competitive operation) At a driven time, there is a schedulable operation for an idle machine. If the earliest completion time of the operation on this machine is earlier than the earliest completion time on any other available machine, then the operation is known as a competitive operation for the idle machine.

Definition 6. (Planned operation) At a driven time, an operation that the idle machine plans to process is known as the planned operation of the idle machine.

Definition 7. (Quasi-schedulable operation) At a driven time, an operation whose immediate predecessors are being processed or starting to be processed is called the quasi-schedulable operation.

Definition 8. (Potential operation) For a planned operation, if the static starting time of a quasi-schedulable operation on the same machine is less than its completion time, then the quasi-schedulable operation is known as the potential operation for the planned operation. The static starting time of an operation is equal to the maximum completion time of its immediate predecessors.

Definition 9. (Relative subsequent path) The path with the longest average length involved from all immediate successors of operation i to the leaf nodes is called the relative subsequent path (RSP) of operation i. The length formula of the relative subsequent path is shown in Eq (15).

$$l_i = \max_h (1/G_{i,h} \sum_g \sum_{i' \in O_{i,g,h} \& i' \in OF_i} (1/|M_{i'}| \sum_{k \in M_{i'}} p_{i',k})) \tag{15}$$

where l_i indicates the length of the relative subsequent path of operation i; $O_{i,g,h}$ indicates the set of operations on the g^{th} path from an immediate successor of operation i to the h^{th} reachable leaf node; OF_i is the set of successors of operation i; $G_{i,h}$ indicates the total number of paths from the immediate successor of operation i to the h^{th} reachable leaf node; $M_{i'}$ is the set of the available machines of operation i'; $|M_{i'}|$ is the number of available machines of operation i'.

Definition 10. (Competition indicator of potential operation) The indicator to measure the competitiveness of potential operations i' compared to the corresponding planned operation i is called the competition indicator of potential operation i'. It can be computed by using Eq (16).

$$\theta_{i',i,k_0} = \frac{\max(c_{i',k_0} + l_{i'}, \min_{k \in M_i}(c_{i,k}) + l_i)}{\max(c_{i,k_0} + l_i, \min_{k' \in M_{i'}}(c_{i',k'}) + l_{i'})} \tag{16}$$

c_{i,k_0} is the completion time of planned operation i on machine k_0 at a driven time; l_i indicates the length of the relative subsequent path of operation i; $\min_{k' \in M_{i'}}(c_{i',k'})$ indicates the earliest completion time of operation i' among all of its available machines; $l_{i'}$ indicates the length of the relative subsequent path of operation i'; c_{i',k_0} is the completion time of operation i' on machine k_0 if planned operation i will not be processed on machine k_0 at the driven time; $\min_{k \in M_{i'}}(c_{i,k})$ indicates the earliest completion time of operation i among all of its available machines.

If $0 < \theta_{i',i,k_0} < \alpha$ (α is suggested to be close to 1), then it indicates that, for machine k_0, potential

operation *i*' is more competitive than planned operation *i*, and it has the advantage of preempting it on machine k_0; operation *i* is known as the interrupted operation for machine k_0 at the driven time. If $\theta_{i',i,k_0} > \alpha$, then it means that planned operation *i* is more competitive and will not be preempted.

3.2. Flexible operation determination strategy

3.2.1. Design ideas

At a driven time, when an operation selected by an idle machine is a flexible operation, there are two situations: 1) If it is the first-position flexible operation, then it indicates that the planning path in the flexible planning path group is not determined. One should look for other flexible planning paths in the flexible planning path group and calculate the earliest completion time for each flexible planning path. If the flexible planning path with the earliest completion time is the path to which the operation belongs and the idle machine is the selected machine, then the operation is determined to be the planned operation for the idle machine. Otherwise, the operation will not be selected by the idle machine. 2) If the operation is a flexible sequential operation, then one should calculate the sum of the completion time of the operation and the length of its relative subsequent path. One should look for other flexible operations in the flexible sequential operation group that have no precedence constraint with it. If there is at least one of the above operations, for which the sum of the earliest completion time and the length of the subsequent path is less than its own, then it will not be selected by the idle machine. Otherwise, it is determined as the planned operation for the idle machine. This strategy realizes the purpose of determining whether the flexible operation selected at the moment is the optimal solution and shortening the completion time by dynamically selecting the flexible operation with the least serial processing at each driven time.

3.2.2. Implementation of the strategy

The specific implementation steps of the FODS are as follows, and its flowchart is shown in Figure 2.

1) Schedule initialization and input the information on the operations and machines, the flexible sequential operation groups and the flexible planning path groups.

2) At the driven time, flexible operation *i* is selected by the idle machine *k*.

3) If the flexible operation *i* is the first-position flexible operation, then go to Step 5. Otherwise, go to Step 14.

4) If the flexible operation *i* is a flexible sequential operation, then go to Step 9. Otherwise, go to Step 14.

5) Find the flexible planning path group to which operation *i* belongs; calculate the earliest completion time C_u of each flexible planning path in the group.

The specific methods to compute C_u are as follows. At the driven time, according to the completion time of the determined operations on machines, and without considering the other

operations, if the u^{th} flexible planning path is scheduled in the group, then the earliest completion time $C_u = \max_{i' \in O_u}(\min_{k' \in M_{i'}}(s_{i',k'} + p_{i',k'}))$.

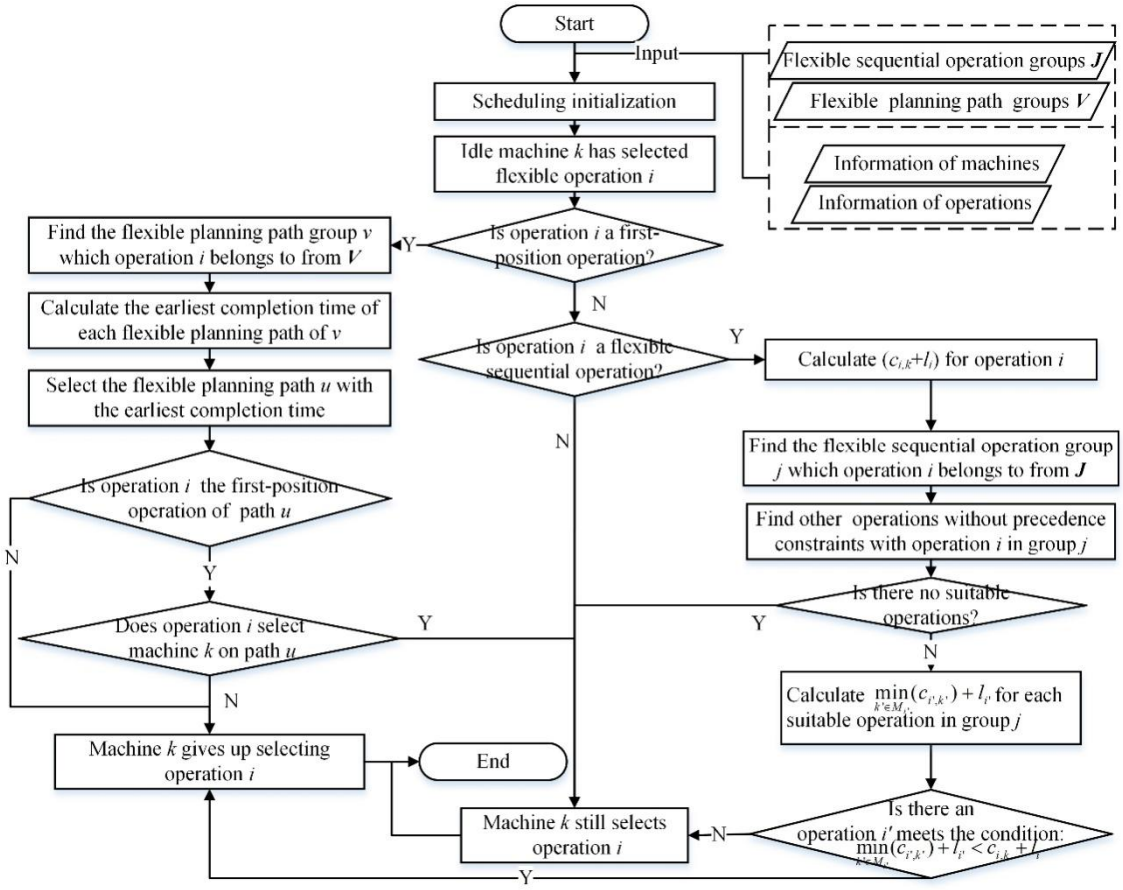


Figure 2. Flowchart of the FODS.

6) Select the flexible planning path with the earliest completion time.

7) If the flexible operation i is the first-position operation of the selected flexible planning path, then go to Step 8. Otherwise, go to Step 15.

8) If the idle machine k is selected by operation i on the selected flexible planning path, then go to Step 14. Otherwise, go to Step 15.

9) Calculate the sum of the completion time of operation i on machine k and the length of its relative subsequent path: $(c_{i,k} + l_i)$.

10) Find the flexible sequential operation group j to which the flexible operation i belongs.

11) If there are other flexible operations without a precedence constraint relationship with operation i in group j, then go to Step 14. Otherwise, go to Step 12.

12) Calculate the sum of the earliest completion time for each operation on its available machines and the length of its relative subsequent path: $\min_{k' \in M_{i'}}(c_{i',k'}) + l_{i'}$, where $i' \in O_j \ \& \ \forall (i,i'),(i',i) \notin R$.

13) If there is an operation i' that satisfies the condition $\min_{k' \in M_{i'}}(c_{i',k'}) + l_{i'} < c_{i,k} + l_k$, then go to Step 15. Otherwise, go to Step 14.

14) If the idle machine k still chooses the flexible operation i, go to Step 16.

15) If the idle machine k no longer selects the flexible operation i, go to Step 16.

16) End.

3.3. Operation optimization allocation strategy

3.3.1. Design ideas

At a driven time, for an idle machine, create a set of its competitive operations if they exist in the schedulable operation set. According to the definition of competitive operations, all operations have the characteristics that the completion time on the idle machine is better than that on the other available machines. Therefore, choosing an appropriate competitive operation can reduce the workload of the machine. Meanwhile, to complete the operations on the critical path as early as possible and avoid the accumulation of a large number of schedulable operations with priority constraints due to deferred processing, the principle of the relative long subsequent path (RLSP) has been designed and adopted in the OOAS. Select the operation with the longest relative subsequent path from the competitive operation set; if the operation is not unique, then select the operation with the earliest completion time. If the operation is still not unique, then the operation with a short setup time is preferred. If the selected operation is a flexible operation, then the FODS is adopted to determine whether the operation is still selected by the idle machine. If the operation is abandoned, then select another operation for the machine according to the principle of the RLSP. If all of the idle machines are executed in the above procedure once, the first round of allocation is over. If unassigned idle machines still exist after the first round of allocation, then the second round of operation allocation continues and the operations of the first-round allocation are processed on their selected machines by default. Repeat the above implementation until all of the idle machines have assigned operations or the remaining idle machines do not have any competitive operations.

3.3.2. Implementation of the strategy

The specific implementation steps of the OOAS are as follows, and its flowchart is shown in Figure 3.

1) Schedule initialization and input the information on the operations and machines, the idle machine set $\boldsymbol{M}_I$ and the schedulable operation set $\boldsymbol{O}_S$. And, create two empty sets: the planned operation set $\boldsymbol{O}_P$ and the planned machine set $\boldsymbol{M}_P$.

2) Set the initial values of the parameters: $m = |\boldsymbol{M}_I|$, $r = 1$. $|\boldsymbol{M}_I|$ indicates the number of idle machines.

3) Select the r^{th} machine from $\boldsymbol{M}_I$ and generate its competitive operation set $\boldsymbol{O}_C$ according to the definition of the competitive operation.

4) If $\boldsymbol{O}_C$ is empty, then m- -, and go to Step 10. Otherwise, go to Step 5.

5) Select an operation from $\boldsymbol{O}_C$ based on the principle of the RLSP.

- The principle of the RLSP: Select the operation with the longest relative subsequent path from the competitive operation set. If the operation is not unique, then select the operation with the earliest completion time. If the operation is still not unique, then the operation with a short setup time is preferred.

6) If the selected operation is a flexible operation, then go to Step 7. Otherwise, go to Step 9.

7) Adopt the FODS, and if the machine still selects the operation, then go to Step 9. Otherwise, go to Step 8.

8) Remove the operation from $\boldsymbol{O}_C$, update $\boldsymbol{O}_C$ and go to Step 4.

9) If the selected operation is determined to be the planned operation of the machine, then add the operation and the machine to $\boldsymbol{O}_P$ and $\boldsymbol{M}_P$ respectively, and remove the operation from $\boldsymbol{O}_S$.

10) r++。

11) If $r < |\boldsymbol{M}_I|$, then go to Step 12. Otherwise, go to Step 3.

12) If $m = 0$ or $m = |\boldsymbol{M}_I|$, then go to Step 14. Otherwise, go to Step 13.

13) Update the information on the machines and operations in $\boldsymbol{O}_P$ and $\boldsymbol{M}_P$ respectively; then, remove them from $\boldsymbol{M}_I$ and go to Step 2.

14) End.

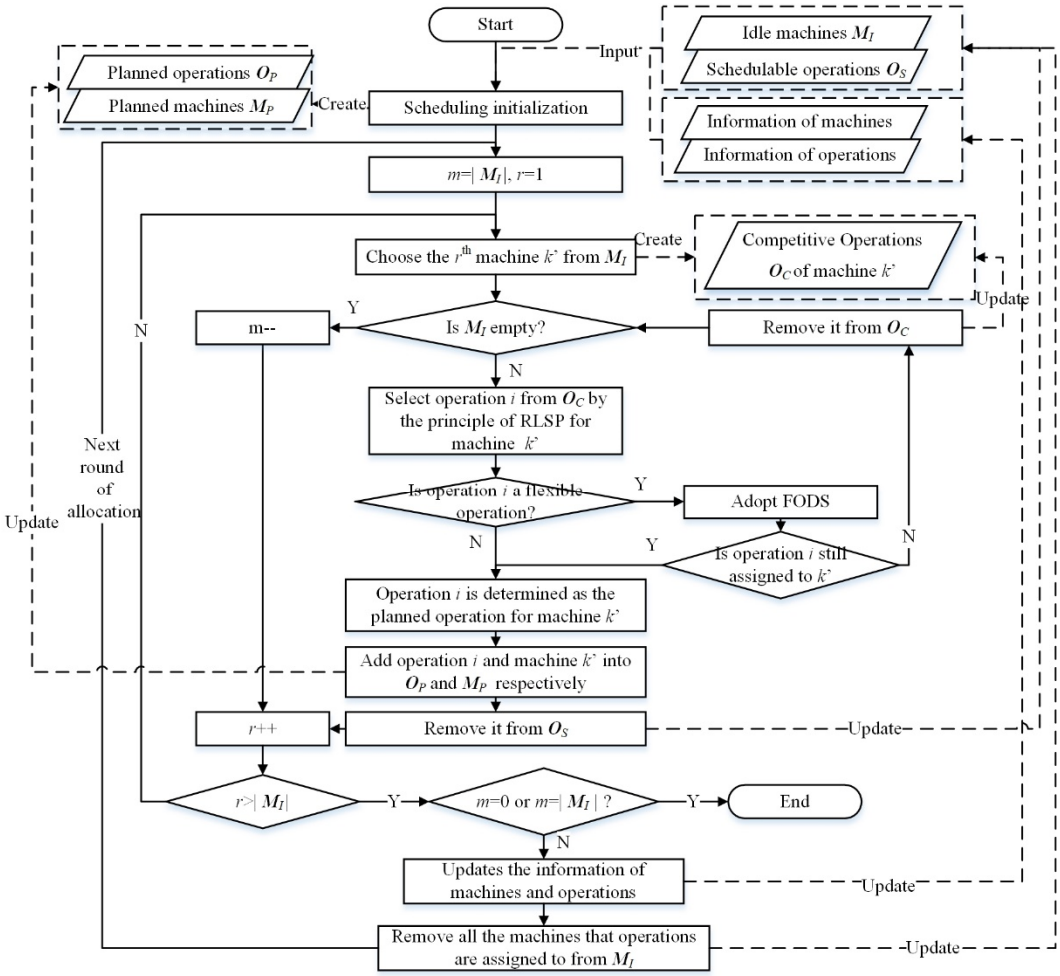


Figure 3. Flowchart of the OOAS.

3.4. Parallel optimization strategy

3.4.1. Design ideas

At a driven time, for an idle machine k with no competitive operation, if it is an available machine for some planned operations, then select operation i which can be finished earliest on machine k from these operations, and mark its planned machine k'. Assign operation i to machine k temporarily. If the competitive operation set of machine k' is not empty, then select the competitive operation with the earliest completion time on machine k'. If the selected operation i' is a flexible operation, then use the FODS to determine whether the machine k' will still select it. If the selection continues, then operation i' is determined to be the new planned operation for machine k', while operation i is finally determined to be the planned operation for machine k. Otherwise, select the other competitive operation with the earlier completion time to machine k', and use the FODS to repeat the above steps. Finally, if there is no competitive operation that meets the requirements, then cancel the movement of operation i to machine k. This strategy maximizes the avoidance of the empty load of idle machines at the driven time by adjusting the scheduling of the planned operations, so as to improve the parallelism between parallel processing operations.

3.4.2. Implementation of the strategy

The specific implementation steps of the POS are as follows, and its flowchart is shown in Figure 4.

1) Schedule initialization and input the information on the operations and machines, the idle machine set $\boldsymbol{M}_I$, the schedulable operation set $\boldsymbol{O}_S$, the planned operation set $\boldsymbol{O}_P$ and the planned machine set $\boldsymbol{M}_P$.

2) Set the initial value of the parameter: $r = 1$.

3) Select the r^{th} machine k from $\boldsymbol{M}_I$.

4) If there are operations of $\boldsymbol{O}_P$ that can be processed on machine k, then go to Step 5. Otherwise, go to Step 14.

5) Select the planned operation i that can be finished earliest on machine k and mark its original planned machine k'.

6) Generate the competitive operation set $\boldsymbol{O}_C$ for machine k'.

7) If $\boldsymbol{O}_C$ is empty, then operation i remains to be the planned operation for machine k'; go to Step 14. Otherwise, go to Step 8.

8) Select the earliest completed operation i' on machine k' from $\boldsymbol{O}_C$.

9) If operation i' is a flexible operation, then go to Step 10. Otherwise, go to Step 12.

10) Adopt the FODS, and if machine k' still selects operation i', then go to Step 12. Otherwise, go to Step 11.

11) Remove the operation from $\boldsymbol{O}_C$ and update it; then, go to Step 7.

12) Determine operation i' as the new planned operation for machine k' and assign the planned scheduling process i to the machine k; then, update $\boldsymbol{O}_P$ and $\boldsymbol{M}_P$.

13) Remove operation i' from $\boldsymbol{O}_S$ and update it.

14) r ++.

15) If $r > |\boldsymbol{M}_I|$, then go to Step 16. Otherwise, go to Step 3. $|\boldsymbol{M}_I|$ is the number of machines in $\boldsymbol{M}_I$.

16) End.

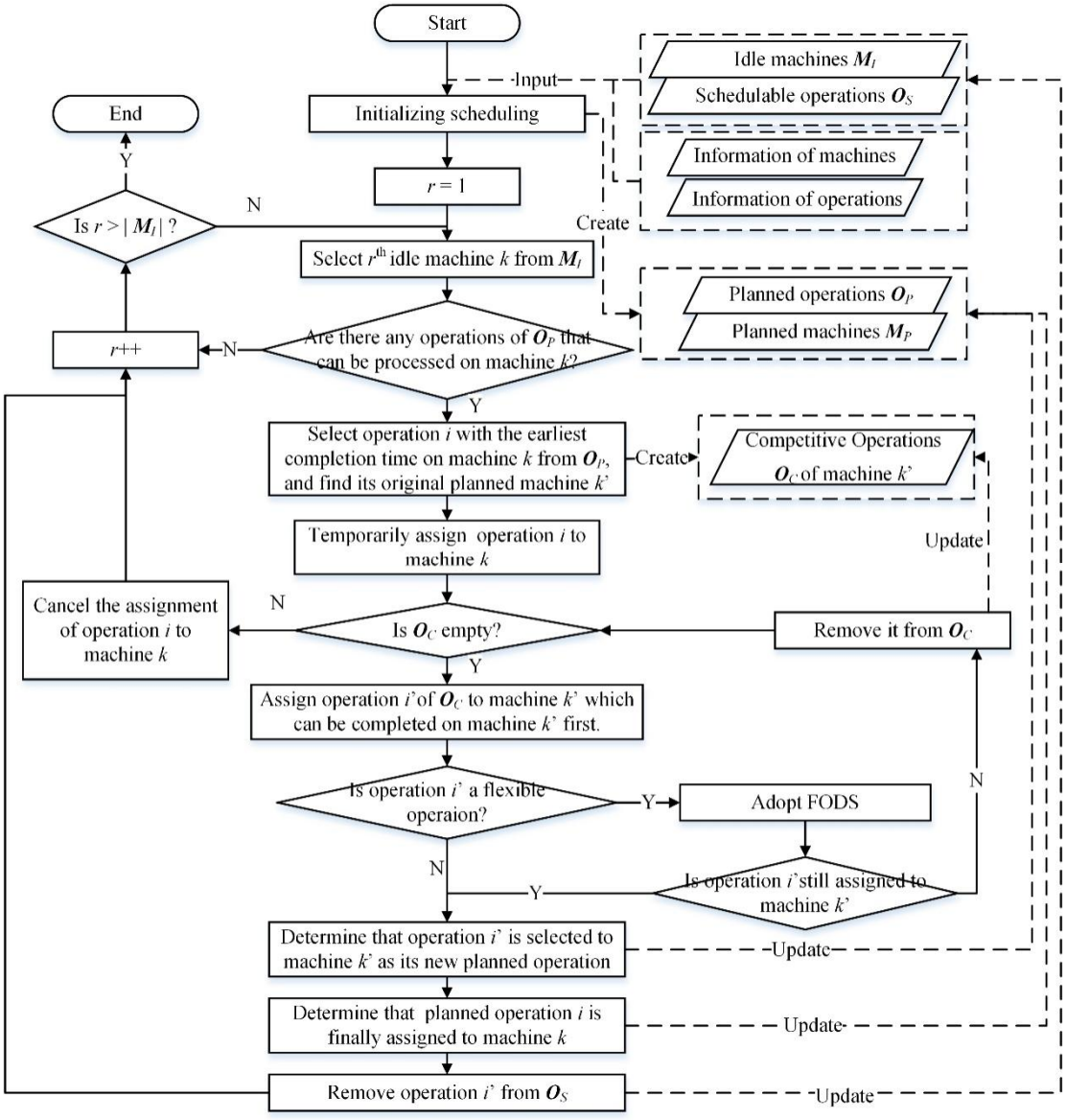


Figure 4. Flowchart of the POS.

3.5. Potential operation preemption strategy

3.5.1. Design ideas

At a driven time, create an empty interrupted operation set. For a planned operation on its planned machine, if its potential operations exist, then compute the competition indicators of these operations in the descending order by the relative subsequent path length. If there is a potential operation for which the competition indicator is less than α, then the planned operation on the planned machine can

be interrupted by it. Otherwise, the planned operation will not be interrupted. Repeat the above procedure and add all of the interrupted operations into the interrupted operation set. This strategy makes up for the shortage of neglecting the preemption of the new schedulable operations which are generated at other driven times.

3.5.2. Implementation of the strategy

The specific implementation steps of the POPS are as follows, and its flowchart is shown in Figure 5.

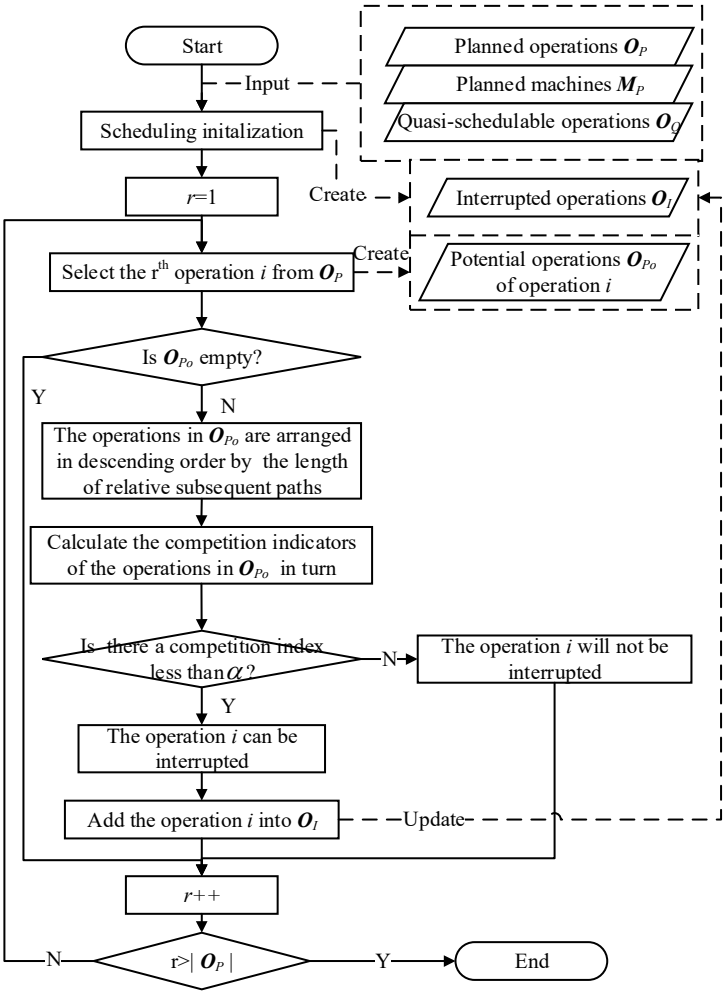


Figure 5. Flowchart of the POPS.

1) Schedule initialization and input the quasi-scheduling operation set O_Q, the planned operation set O_P and the planned machine set M_P; then, create an empty interrupted operations set O_I.

2) Set the initial value of the parameter: $r = 1$.

3) Select the r^{th} planned operation from O_P and generate the corresponding potential operation set O_{Po} according to the definition of the potential operation.

4) If O_{Po} is empty, then go to Step 9. Otherwise, go to Step 5.

5) Arrange the operations in $\boldsymbol{O}_{Po}$ in descending order according to the length of their relative subsequent paths.

6) Calculate the competition indicator for each potential operation in turn.

7) If there is an operation for which the competition indicator is less than α, then the planned operation can be interrupted; go to Step 8. Otherwise, the planned operation will not be interrupted; go to Step 9.

8) Add the planned operation into $\boldsymbol{O}_I$.

9) r++.

10) If $r >$ | $\boldsymbol{O}_P$ |, then go to Step 11. Otherwise, go to Step 3. | $\boldsymbol{O}_P$ | indicates the scale of $\boldsymbol{O}_P$.

11) End.

4. Algorithm design and complexity

4.1. Algorithm design and implementation

According to the above analysis of the main strategies, the implementation steps of the proposed algorithm, named MFISA, have been designed as follows.

1) Schedule initialization and input the information on the operations and machines and setup times; then, initialize the data.

2) Generate the idle machine set $\boldsymbol{M}_I$, the schedulable operation set $\boldsymbol{O}_S$ and quasi-schedulable operation set $\boldsymbol{O}_Q$ at the drive time T_d.

- Machine status mark: There are two marks for machine status. If the machine is idle, then its status mark is 1; if the machine is occupied, then its status mark is 0. At the initial time $T_d = 0$, the status mark of all machines is 1.
- Operation status mark: There are three marks for operation status. If the operation is unprocessed, then the operation status is marked as 0; if the operation is completed, then the operation status is marked as 1; if the operation is under processing, then the operation status is marked as 0.5. At the initial time $T_d = 0$, the status mark of all operations is 1.
- Mark conversion: At driven time T_d, when an operation is assigned to a machine, the machine status mark is converted from 1 to 0 and the operation status mark is converted from 1 to 0.5. When the next driven time T_{d+1} is confirmed, the status mark of the machine, for which the ending time is equal to T_{d+1}, is converted from 0 to 1; and, the status mark of the corresponding operation is converted from 0.5 to 0.
- Generation of $\boldsymbol{M}_I$: Traverse the status marks of all machines, and then add the machines with status marks of 1 to $\boldsymbol{M}_I$.
- Generation of $\boldsymbol{O}_S$: Traverse the status marks of all operations, and then add to $\boldsymbol{O}_S$ the operations for which the status marks are 1 and the status marks of their predecessors are 0. In particular, at the initial moment, in the processing tree, all node operations with zero-entry degree are schedulable operations. Since the MFISA adopts the reverse-order scheduling mode, only the root node operation is schedulable at the initial time.
- Generation of $\boldsymbol{O}_Q$: Traverse the status marks of all operations and add to $\boldsymbol{O}_Q$ the operations for which the status marks are 1 and the status marks of their immediate predecessors are 0.5.

- Driven time: The moment that the status mark of a machine is converted to 1 is known as a driven time, and the scheduling system is driven to have a scheduling event, that is, a schedulable operation is decided to be assigned for the idle machines.

3) Adopt the OOAS to assign the planned operations for idle machines and generate the planned operation set $\boldsymbol{O}_P$ and the planned machine set $\boldsymbol{M}_P$.

4) If all idle machines have been assigned planned operations, then go to Step 6. Otherwise, go to Step 5.

5) Adopt the POS to deal with the unassigned idle machines and update $\boldsymbol{O}_P$ and $\boldsymbol{M}_P$.

6) The POPS is used to determine whether the current planned operation can be preempted by the other operations, and to generate the interrupted operation set $\boldsymbol{O}_I$.

7) If $\boldsymbol{O}_I$ is empty, then go to Step 9. Otherwise, go to Step 8.

8) The planned machines of the operations in $\boldsymbol{O}_I$ are locked for them at T_d, that is, each operation cannot be selected by its planned machine until the next driven time. Then, update $\boldsymbol{O}_S$, retreat all machine statuses and all operation statuses back to the driven time T_{d} and go to Step 3.

- The locked machine for the operation is recorded as "operation number L (machine number)". For example, the corresponding machine 3 of operation 26 is locked and recorded as 26L (3).

9) Calculate the next driven time T_{d+1} and update the driven time $T_d = T_{d+1}$. Then, update $\boldsymbol{M}_I$, $\boldsymbol{O}_S$ and $\boldsymbol{O}_Q$.

- The driven time $T_{d+1} = \min_{i \in \boldsymbol{O}_P}(c_i)$, where i is the operation in $\boldsymbol{O}_P$; that is, the next driven time T_{d+1} is equal to the earliest completion time of the planned operations.
- Convert the status marks of all planned operations to 0.5 and convert the status marks of the operations for which the completion time equals to T_{d+1} to 0.
- Convert the status marks of all planned machines to 0 and convert the status marks of the machines for which the completion time equals to T_{d+1} to 1.

10) If all operations have been processed, that is, if the status mark of each operation is 0, then go to Step 11. Otherwise, go to Step 2.

11) Output the scheduling information and the Gantt chart.

12) End.

The flowchart for the MFISA is shown in Figure 6.

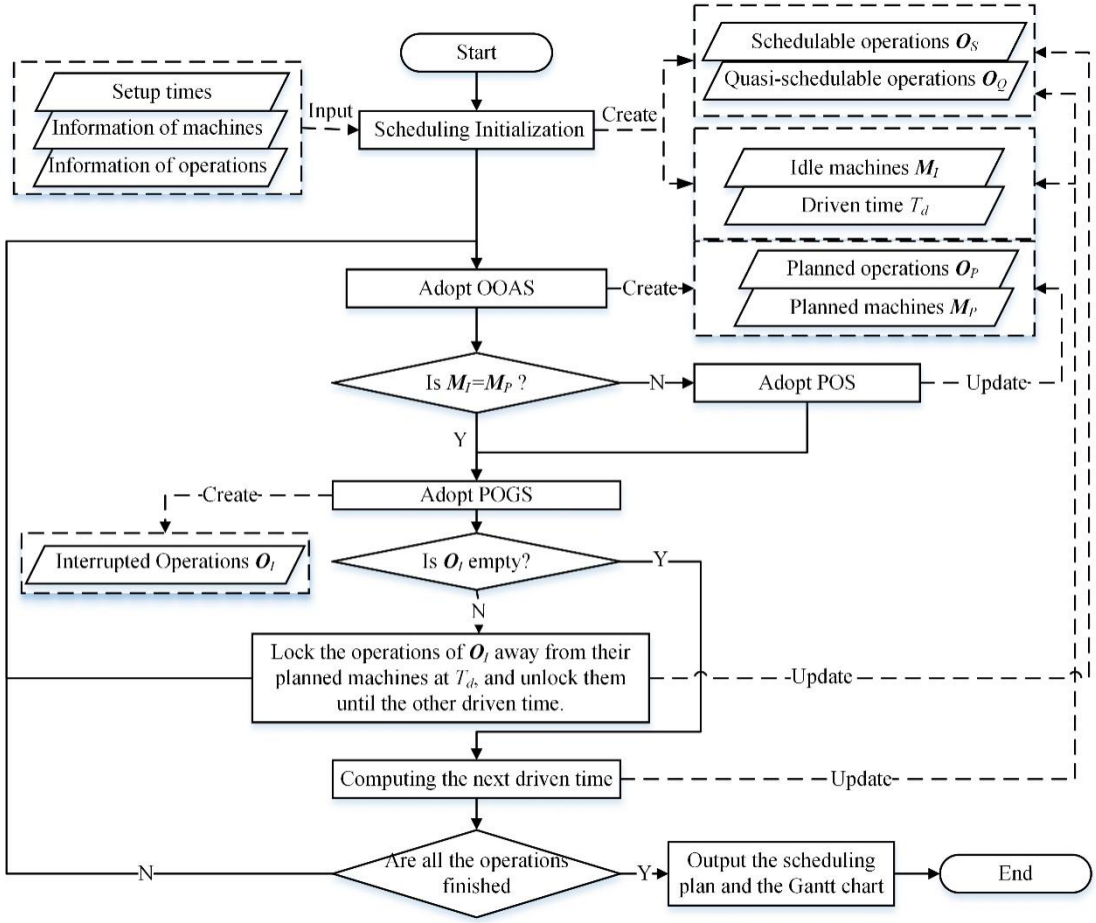


Figure 6. Flowchart of the MFISA.

4.2. Complexity analysis

The total number of operations of a complex product is set to be n, where the total number of the flexible operations is n_1, and the total number of the involved machines is m. Then, the complexities of the FODS, the POS, the POAS and the POPS involved in the MFISA are as follows:

1) In the FODS, the calculation of the earliest completion time of the flexible planning path requires n_1 times, or the calculation of the sum of the earliest completion time of the flexible sequential operation and its relative subsequent path length requires n_1 times. The comparison with the sum of the completion time of the planned operation and the relative subsequent path length requires (n_1 - 1) times. Because $n_1 \leq n$, if and only if $n_1 = n$, the complexity will be O(n).

2) There are at most m idle machines involved in the OOAS, and the calculation of the competitive operations for each machine needs nm times. Each operation needs to be sorted once to find the available machine with the shortest completion time, so $nm^2\log_2(m)$ times are required in total. When adopting the principle of the RLSP, the length of the relative subsequent path needs to be calculated at most n times and be sorted once, so it requires $n\log_2(n)$ times executions. During the execution of the OOAS, the FODS is adopted at most n_1 times. Therefore, the OOAS involves

($nm^2\log_2(m) + n\log_2(n) + n_1(n_1 - 1)$) times executions, so the maximum complexity is $O(n^2)$ if and only if $n_1 = n$.

3) There are at most (m - 1) idle machines involved in the POS, so, to generate the competitive process set for each planned machine, it needs to calculate the completion times at most (n - 1)m times, and each operation should be sorted once to find the machine with the shortest completion time. Thus, it requires $(n\text{-}1)m^2\log_2(m)$ times executions. Then, all competitive operations need to be sorted with $(n\text{-}1)\log_2(n\text{-}1)$ times executions. During the execution of the POS, the FODS is adopted at most n_1 times. Therefore, the POS involves $((n\text{-}1)m^2\log_2(m) + (n\text{-}1)\log_2(n\text{-}1) + n_1(n_1\text{-}1))$ times executions, so the maximum complexity is $O(n^2)$ if and only if $n_1 = n$.

4) There are at most m planned operations that need to have the preemptive judgment in the POPS. It needs to calculate the earliest starting times for at most (n-m) operations to generate the potential operations for each planned operation. Then, it needs at most $m(n\text{-}m)$ times for comparison with the completion times of the planned operations. In order to judge the preemption of the potential operations, the competition indicators need to be calculated (n-m) times for each planned machine, so $m(n\text{-}m)$ times executions are required in total, and m operations are added into the interrupted operation set. Therefore, the POPS involves at most $(m(n\text{-}m) + m(n\text{-}m) + m)$ times executions, so the complexity is $O(n)$.

5) When a preemption event occurs at the driven time, it is necessary to lock m operations and the corresponding machines, and to reassign the operations to the idle machines at the driven time. Thus, $((n\text{-}1) + (n\text{-}2) + \ldots + 1) = n(n\text{-}1) / 2$ times executions are needed in total for all driven times.
In conclusion, the complexity of the MFISA in this paper is $\max\{n(n\text{-}1)/2\ O(n), n(n\text{-}1)/2\ O(n^2), n(n\text{-}1)/2\ O(n^2), n(n\text{-}1)/2\ O(n)\}$. Therefore, the complexity is $O(n^4)$ if $n_1 = n$, or $O(n^3)$ if $n_1 \ll n$.

5. Experimental analysis

5.1. Numerical example for MFISP

The above analysis and design of the MFISA are not based on a particular example, so the algorithm and strategies in this paper have more general applicability and application significance. In order to further explain the execution processing and the final effect of this algorithm, this paper will explain it through an example and highlight the scheduling procedure that involves the scheduling strategies.

The processing operation tree of a complex product A is shown in Figure 7. Each node represents an operation with three common attributes: the operation number, the available machine numbers and the processing time of each machine. In addition, the special operation contains its operation type information. Except for the ordinary operations, there are four special operation types; information on the setup times is shown in Table 2. The flexible planning path groups and the flexible sequential operation groups are represented by the AND sub-graphs and the OR sub-graphs in the processing operation tree, respectively.

Table 2. Setup times between different types of operations.

Type	Machine 1				Machine 2				Machine 3				Machine 4			
	I	II	III	IV	I	II	III	IV	I	II	III	IV	I	II	III	IV
I	0	2	2	4	0	3	4	3	0	7	7	2	0	5	8	3
II	2	0	3	2	3	0	5	2	6	0	4	3	5	0	2	5
III	3	3	0	5	4	5	0	2	7	4	0	3	8	2	0	5
IV	3	2	3	0	4	3	2	0	5	3	3	0	3	5	5	0

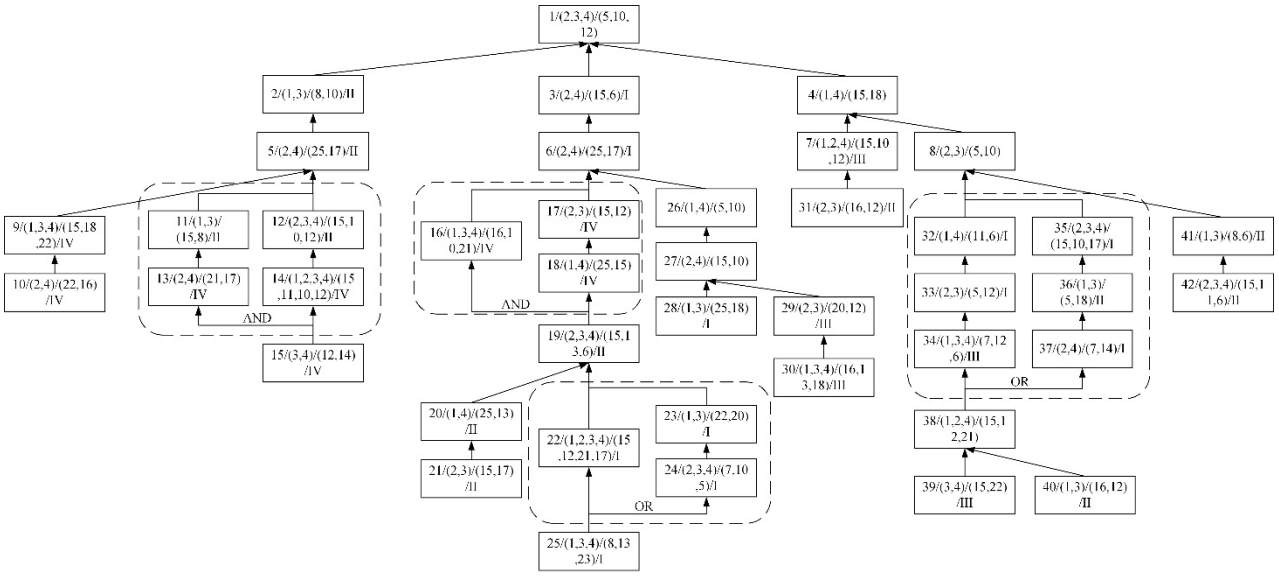


Figure 7. Processing operation tree of Product A.

Now, the MFISA with α = 0.94 is applied to solve the scheduling problem of Product A. The scheduling procedure of each driven time is shown in Table 3. To make the strategies easier to understand, a few driven times are described in detail.

First, we need to reverse the processing operation tree, that is, to perform priority scheduling of the root node and reverse scheduling along the arrow direction. At the initial time, $t = 0$, the schedulable operation set $\boldsymbol{O}_S = \{1\}$ and the idle machine set $\boldsymbol{M}_I = \{1, 2, 3, 4\}$. Only Machine 2 has the competitive operation set $\boldsymbol{O}_C = \{1\}$, so, according to the OOAS, Machine 2 selects Operation 1 as its planned operation. At this time, the planned operation set $\boldsymbol{O}_P = \{1\}$ and planned machine set $\boldsymbol{M}_P = \{2\}$. Since there is no potential operation of Machine 2, the final scheduling scheme is as follows: $\boldsymbol{O}_P = \{1\}$ and $\boldsymbol{M}_P = \{2\}$.

At $t = 5$, the schedulable operation set $\boldsymbol{O}_S = \{2, 3, 4\}$ and the idle machine set $\boldsymbol{M}_I = \{1, 2, 3, 4\}$. Then, the OOAS is adopted, in the first round of allocation, the competitive operation set of Machine 1 is $\boldsymbol{O}_C = \{2, 4\}$, $l_2 = 122.7$, $l_4 = 60.7$. According to the principle of the RLSP, Machine 1 selects Operation 2, and Machine 4 selects Operation 3. By updating the earliest ending time of each machine after the first round of allocation, the idle machine set $\boldsymbol{M}_I = \{2, 3\}$ and there is no competitive operation for them; so, the second round of allocation is not required. At this time, the planned operation set $\boldsymbol{O}_P = \{2, 3\}$ and planned machine set $\boldsymbol{M}_P = \{1, 4\}$. Then, the POS is adopted, the planned operation 3 on the planned machine 4 is an available operation for the idle machine 2. Operation 2 is temporarily assigned to Machine 3. Meanwhile, the competition operation set of Machine 4 is {4}, so Machine 2

selects Operation 3 as the planned operation, and Machine 4 selects Operation 4 as its new planned operation. At this time, the planned operation set $\boldsymbol{O}_P = \{2, 3, 4\}$ and planned machine set $\boldsymbol{M}_P = \{1, 2, 4\}$.

There is a potential operation set $\boldsymbol{O}_{Po} = \{5, 6\}$ for Machine 4. According to the POPS, the competition indicator for Operation 4 is $\theta_{5,4,4} = 0.907 < 0.94$, which means that Operation 4 may be preempted by Operation 5 on Machine 4. Then, lock Operation 4 up from being selected by Machine 4 until the next driven time. Trace all of the scheduled information back to $t = 5$ and reschedule the idle machines and operations at $t = 5$.

Backtracking to $t = 5$, the schedulable operation set $\boldsymbol{O}_S = \{2, 3, 4\}$ and the interrupted operations set $\boldsymbol{O}_I = \{4L(4)\}$. According to the OOAS, the planned operation set $\boldsymbol{O}_P = \{2, 3\}$ and planned machine set $\boldsymbol{M}_P = \{1, 4\}$ are obtained. The planned operation 2 on the planned machine 1 is an available operation for the idle machine 3. According to the POS, Operation 2 is temporarily assigned to Machine 3. Meanwhile, the competition operation set of Machine 1 is {4}, so Machine 2 selects Operation 3 as the planned operation and Machine 1 selects Operation 4 as its new planned operation. Thus, the planned operation set $\boldsymbol{O}_P = \{2, 3, 4\}$ and planned machine set $\boldsymbol{M}_P = \{3, 4, 1\}$. Since there is no potential operation set for $\boldsymbol{M}_P$, the final scheduling scheme at $t = 5$ is as follows: $\boldsymbol{O}_P = \{2, 3, 4\}$ and $\boldsymbol{M}_P = \{3, 4, 1\}$.

At $t = 30$, the schedulable operation set is {16/17, 26, 32/35, 41}, the idle machine set $\boldsymbol{M}_I = \{2, 3\}$ and the interrupted operations set $\boldsymbol{O}_I = \{26L(4)\}$. Operations 16 and 17 are flexible operations in the same flexible sequential operation group, and Operations 32 and 35 are the first-position flexible operations on two flexible planning paths of the same flexible planning path group. The competitive operation set of Machine 3 is {16/17, 35, 41}, and the competitive operation set of Machine 4 is {32}. According to the OOAS, the planned operations of Machines 3 and 4 are Operations 16 and 32, respectively. Then, according to the FODS, Operation 16 is better than Operation 17, and the planning path of Operation 32 is better than that of Operation 35; so, Machines 3 and 4 still keep their selection. At this time, the planned operation set $\boldsymbol{O}_P = \{16, 32\}$ and planned machine set $\boldsymbol{M}_P = \{3, 4\}$. Finally, according to the POPS, it is known that neither Machine 3 nor Machine 4 will be preempted, so the final scheduling scheme at $t = 30$ is as follows: $\boldsymbol{O}_P = \{16, 32\}$ and $\boldsymbol{M}_P = \{3, 4\}$.

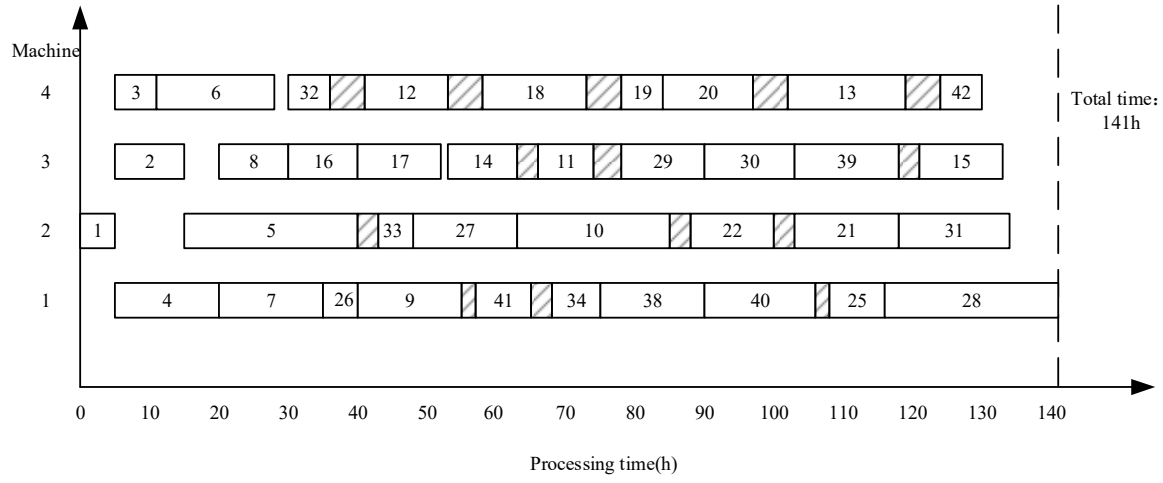


Figure 8. Gantt chart of Product A obtained by the MFISA.

The Gantt chart is shown in Figure 8, where a gray square indicates the setup time. The final completion time of Product A is 141 hours; the selected orders for the two flexible sequential operation groups are {16≻17≻18} and {12≻11≻14≻13}, and the selected planning paths of the two flexible planning path groups are {32≻33≻34} and {22}. The final positive scheduling order of operation and the corresponding machines for Product A are {42, 15, 31, 28, 25, 39, 21, 13, 30, 40, 22, 20, 38, 29, 19, 34, 11, 10, 41, 18, 14, 27, 12, 17, 33, 9, 26, 32, 16, 8, 7, 5, 6, 2, 3, 4, 1} and {4, 1, 1, 3, 1, 1, 2, 4, 2, 3, 3, 1, 2, 1, 1, 2, 2, 4, 4, 4, 3, 3, 3, 3, 4, 4, 3, 2, 1, 3, 1, 4, 2, 1, 4, 3, 2}.

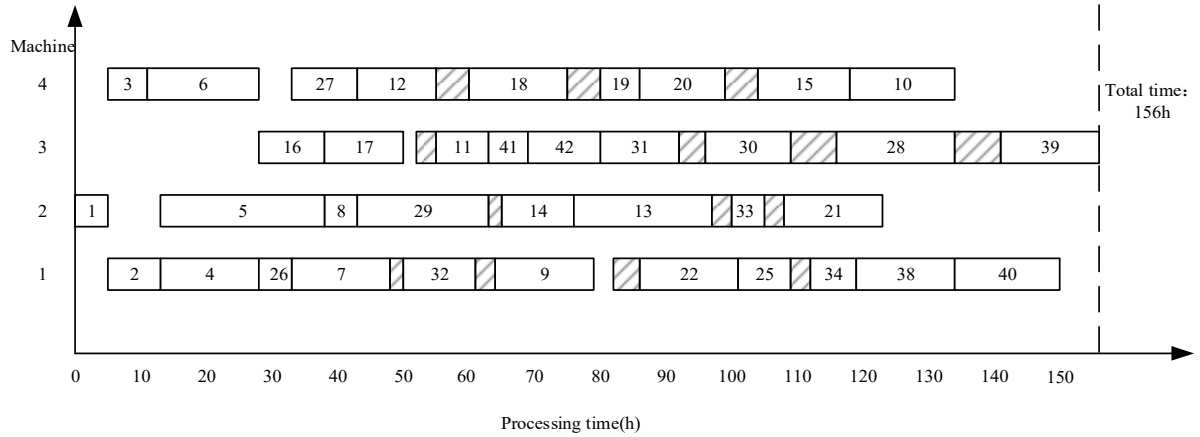


Figure 9. Gantt chart of Product A obtained by the MFISA without using the POS.

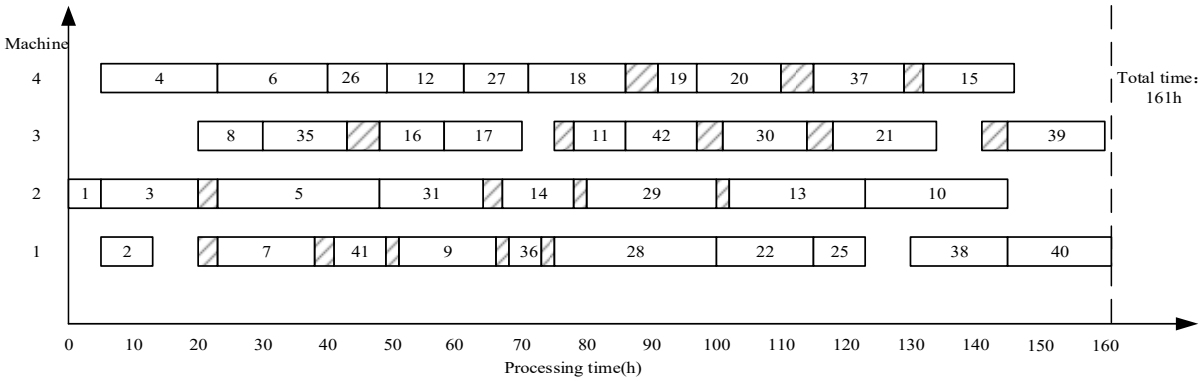


Figure 10. Gantt chart of Product A obtained by the MFISA without using the POPS.

To visualize the effects of POS and POPS, Figures 9 and 10 show the Gantt charts obtained without the POS and POPS, respectively. The makespans of Product A are 156 hours and 161 hours, respectively. Obviously, their makespans are much longer than that obtained via the MFISA. The reasons are as follows.

Without using the POS, many operations cannot be processed in parallel with optional machines. For example, in Figure 9, without using the POS, Operation 4 is not assigned to any machine at $t = 5$, which delays all of the starting times of the successor operations, and Machine 3 remains idle until time 28.

Without using the POPS, the processing states at some driven times will not backtrack to a better scheme to consider the competition of some potential operations, which can preempt the machine. For example, in Figure 10, without using the POPS, the processing state is not backtracked to driven time

5 by because Operation 6 can preempt Machine 4 from Operation 4 at time 11. Operation 6 will be processing on Machine 4 at time 23, which is 12 hours later than it in Figure 8.

Table 3. Operation scheduling table.

Driven time	Schedulable operation	Machine				Next driven time	Interrupted operation
		1	2	3	4		
0	1		1			5	
5	2, 3, 4	2 II	3I		4	5	4
5	2, 3, 4L(4)	4	3I		2 II	11	
11	6				6I	15	
15	5		5II			20	
20	7, 8	7III		8		28	
28	16/17, 26				26	28	26
28	16/17, 26L(4)*					30	
30	16/17, 26L(4), 32/35, 41			16IV	32I	35	
35	26, 41, 31	26				36	
36	33, 41, 31					40	
40	33, 41, 31, 27, 17, 9, 11/12	9IV	33I	17IV	12II	40	
48	41, 31, 27 , 34		27			52	
52	41, 31, 34, 18			41II		52	41
52	41L(3), 31, 34, 18			31II		52	31
52	41L(3), 31L(3), 34, 18					53	
53	41L(3), 31L(3), 34, 18, 11/14			14IV	18IV	55	
55	41, 31, 38, 34, 10	41II				63	
63	11, 31, 38, 34, 10, 29		10IV	11II		65	
65	28, 29, 31, 34, 42	34III				73	
73	31, 28, 39, 42, 19				19IV	74	
74	41, 31, 28, 39, 13			29III		75	
75	13, 28, 31, 38, 42	38				84	
84	13, 28, 31, 20, 22/23, 42				20II	85	
85	13, 22/23, 28, 31, 42		22I	42II		90	
90	13 , 28, 30, 31, 40, 42	40II		30III		97	
97	31, 28, 39, 42, 13, 21				13IV	100	
100	28, 31, 39, 42, 21, 25		21II			103	
103	28, 39, 42, 31, 25			39III		106	
106	25, 28, 31, 42	25I				116	
116	28, 31, 42	28I				118	
118	31, 42		31II			119	
119	42, 15			15IV	42II		

*: 26L(4) indicates Operation 26 cannot be selected by Machine 4

Table 4. Statistics and results of the instances.

No. of instances	OPs.[a]	MAs.[b]	FPPGs.[c]	FSOGs.[d]	FOs.[e]	JMRW		MLW		ECT		MFISA		
						MS.[f]	CT.[g]	MS.[f]	CT.[g]	MS.[f]	CT.[g]	Average MS.[f]	MS.[f] (best known)	Average CT.[g]
1	50	5	3	2	25	614	0.08	606	0.02	783	0.03	558.55	543	0.23
2	50	5	4	2	26	414	0.19	400	0.02	442	0.00	290.73	287	0.25
3	66	5	4	4	44	724	0.13	520	0.03	576	0.00	454.55	441	0.26
4	59	5	4	2	35	713	0.13	709	0.05	631	0.02	531.73	496	0.22
5	62	5	3	3	38	833	0.09	695	0.00	702	0.02	487.27	480	0.21
6	144	10	4	4	52	719	0.73	649	0.09	753	0.08	432.64	424	1.10
7	127	10	2	3	32	528	0.36	730	0.09	825	0.02	484.73	432	0.55
8	144	10	4	4	52	747	0.39	847	0.09	956	0.09	542.36	513	0.72
9	134	10	3	3	40	563	0.33	697	0.09	630	0.05	360.64	335	1.05
10	160	10	2	3	35	775	0.33	1093	0.09	1051	0.03	600.73	575	0.98
11	300	15	4	7	83	834	1.03	1139	0.36	1229	0.09	625.09	572	3.18
12	307	15	8	3	85	868	1.92	1300	0.22	1264	0.08	644.73	596	3.62
13	293	15	5	6	101	1122	1.38	1137	0.19	1390	0.13	623.64	593	3.93
14	294	15	2	7	84	775	1.38	829	0.33	780	0.19	465.27	429	4.97
15	279	15	3	2	47	710	1.17	1079	0.19	967	0.11	477.18	456	2.82
16	435	20	6	3	105	689	2.56	1031	0.33	1027	0.14	489.00	431	8.49
17	390	20	8	6	93	552	2.05	550	0.33	614	0.14	360.55	343	8.79
18	411	20	8	2	72	666	2.53	799	0.28	815	0.08	440.55	416	7.95
19	371	20	2	6	67	703	2.25	1065	0.38	943	0.14	495.18	463	7.95
20	372	20	3	2	34	763	2.06	1694	0.19	1613	0.09	637.27	621	3.48
21	491	25	6	3	69	581	4.17	686	0.31	713	0.16	349.45	329	15.77
22	487	25	6	5	96	707	2.73	1301	0.19	1122	0.14	510.91	488	10.37
23	448	25	2	4	45	600	3.45	1042	0.27	889	0.16	443.55	420	11.92
24	526	25	4	8	124	741	4.06	1002	0.22	1156	0.20	465.00	421	23.99

Continued on next page

No. of instances	OPs.[a]	MAs.[b]	FPPGs.[c]	FSOGs.[d]	FOs.[e]	JMRW		MLW		ECT		MFISA		
						MS.[f]	CT.[g]	MS.[f]	CT.[g]	MS.[f]	CT.[g]	Average MS.[f]	MS.[f] (best known)	Average CT.[g]
25	504	25	7	5	113	608	3.09	906	0.17	849	0.17	407.00	392	14.80
26	667	30	8	6	166	652	5.92	787	0.28	872	0.28	387.73	368	32.10
27	687	30	8	6	153	733	6.25	710	0.30	758	0.30	377.27	363	31.44
28	647	30	7	6	157	792	5.13	1481	0.22	1315	0.38	557.55	525	27.80
29	662	30	5	8	143	711	6.47	981	0.38	788	0.42	464.27	433	32.45
30	622	30	7	6	117	806	6.14	1555	0.28	1501	0.34	629.45	610	18.17

a: number of operations; b: number of machines; c: number of flexible planning path groups; d: number of flexible sequential operation groups
e: number of flexible operations; f: makespan; g: CPU time

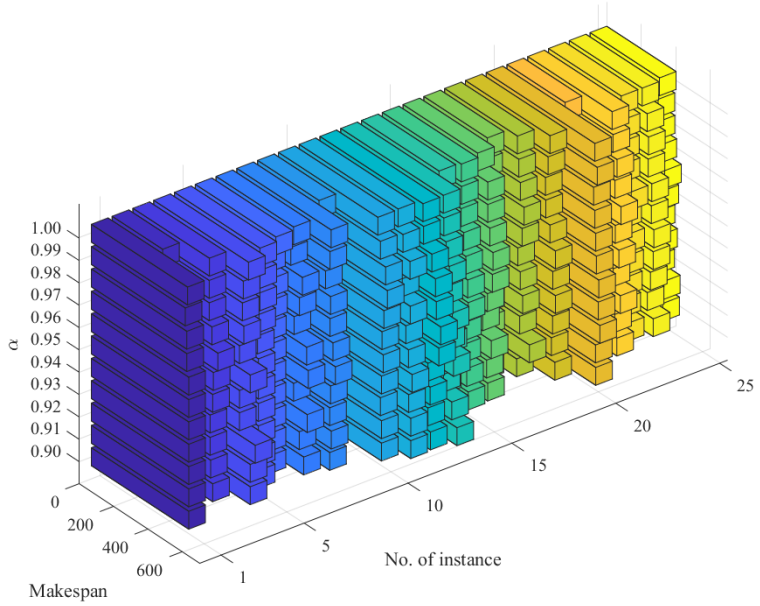


Figure 11. Results of MFISA with different values of α.

5.2. Experiments for MFISP

In order to research the effectiveness of the proposed MFISA for the MFISP, we use the hypothesis test cases. Because there is no published integrated algorithm for the MFISP, we chose three typical dispatching rules: earliest completion time (ECT), job with most remaining work (JMRW) and machine with least workload (MLW); these were used to enable comparison with the MFISA. Because the MFISA does not describe the operations by jobs due to the reprocessing of some subassemblies, in this study, the dispatching rule JMRW was modified to give priority to the operation with the longest relative subsequence path. The number of actual scheduling operations of a complex product in the MFISP cannot be fixed, so 30 complex products with different scales and quantities of operation and equipment were selected. The process parameters of these products and the results of the above four algorithms are shown in Table 4. We chose $\alpha \in [0.9,1]$ with the step length of 0.01 for the proposed algorithm, where a total of 11 groups of experiments were carried out. Table 4 shows the best results and average values of these 11 groups of experiments. The results of these 11 groups of experiments are shown in Figure 11. For further intuitive illustration, the confidence intervals of the optimal solutions and CPU time are shown in Figure 12.

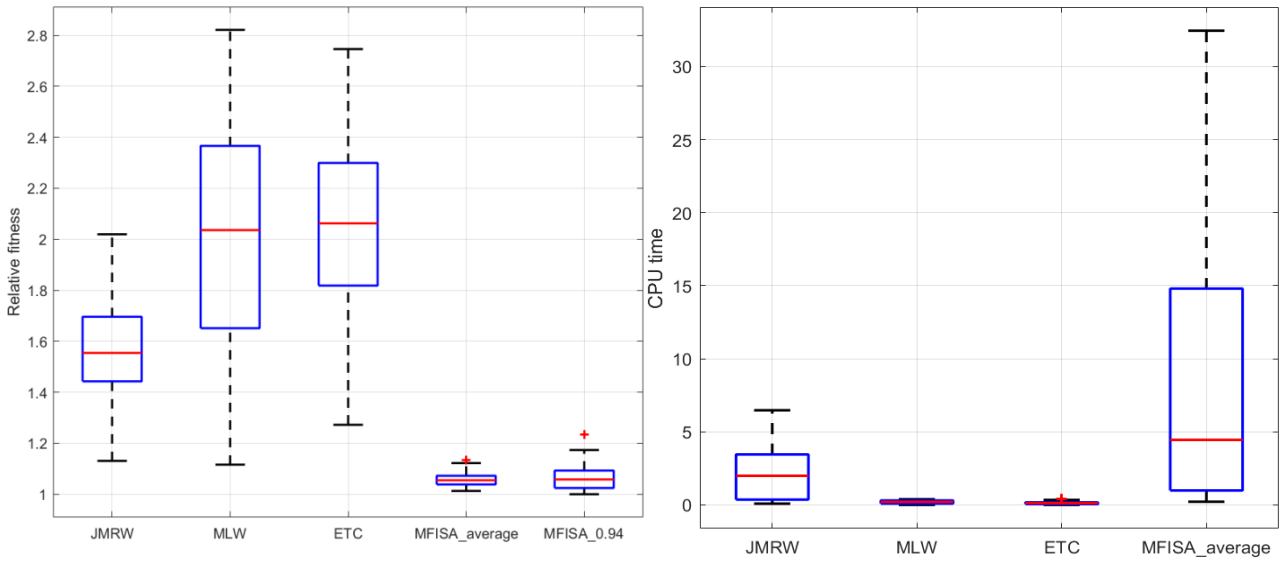


Figure 12. Comparison of JMRW, MLW, ETC and MFISA results for the MFISP: (a) relative fitness and (b) CPU time.

In Figure 11, it is obvious that the optimal result of each product obtained by the MFISA may differ with different α. This is because these products are structured differently and the MFISP is NP-hard. Also, the differences between the results were not huge. Thus, we randomly chose the case with $\alpha = 0.94$ as an example. In Figure 12(a), it is clear that the average results of the MFISA are better than those of the other three algorithms, and all of the optimal results (best-known) were obtained by the MFISA. Also, the MFISA with $\alpha = 0.94$ performed better than the other three algorithms. Most of the results of the other three algorithms were far from the optimal solutions, among which the MLW performed the worst. However, in Figure 12(b), we can see that the CPU times of the MFISA are much slower than those of the other algorithms. If there is a dynamic disturbance, the MLW is the fastest

algorithm to respond, but it also sacrifices the makespans of the complex products.

The results show that the scheduling results of the proposed algorithm are the most favorable. As the operation scale increases, this advantage becomes more obvious. In fact, all of the optimal solutions were obtained by it with a suitable $\alpha \in [0.9,1]$. Among all of the comparisons, the MFISA had the best performance.

5.3. Numerical example for FISP

When the number of flexible operations is zero and there are no special types of operations, the MFISP will degenerate into the normal FISP, which only considers the flexibility of machines. The MFISA can also be applied to solve the scheduling for the normal FISP.

Now, we take the processing operation tree of Product B in Figure 13 as an example. The other three flexible integrated algorithms, i.e., the DPFISA [36] and CMFISA [37], which have better scheduling effects on the FISP, and the RLFISA [40], which is the latest algorithm, were compared with the MFISA. The Gantt chart for each is shown in Figure 14. The completion time of the scheduling scheme obtained by the MFISA was 120 hours, which is the same as that of the DPFISA and much less than the 155 hours obtained by the CMFISA. The result shows that the MFISA can also better solve the FISP.

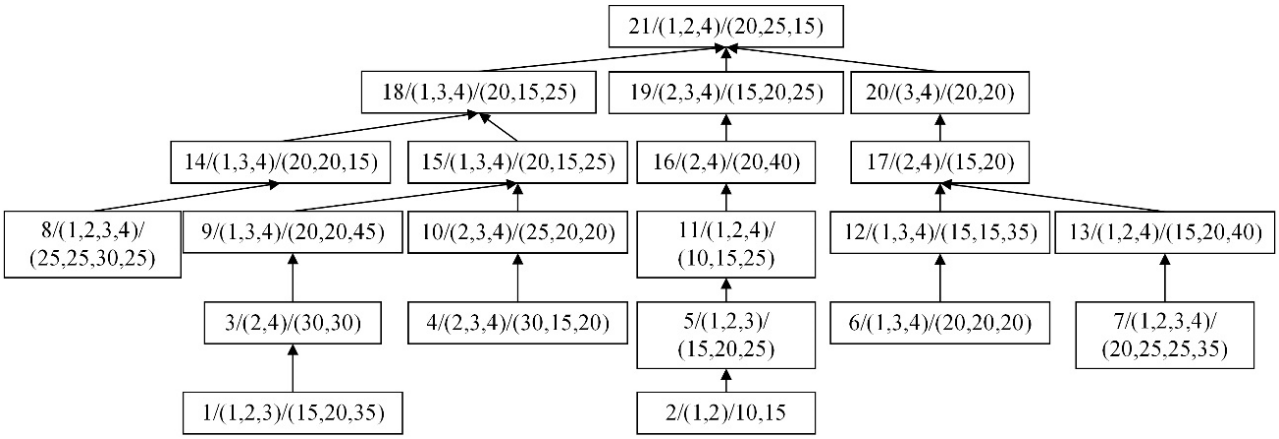


Figure 13. Processing operation tree of Product B.

5.4. Experiments for FISP

To further illustrate the effectiveness of the MFISA for solving the normal FISP, we randomly generated 13 groups of test instances with the operation and equipment combination of [50,100,200] × [5,10,15] and [400,600] × [15,20]. Each group contained 10 test instances of the same scale, for a total of 130 instances. In this study, 13 groups of test instances were divided into low density, medium density and high density based on the average number of operations per machine. Low density refers to the instances in which the average number of operations per machine is less than 20. High density refers to the instances in which the average number of operations per machine is more than 30, and the remaining instances are at medium density. The scales of all instances are shown in Table 5.

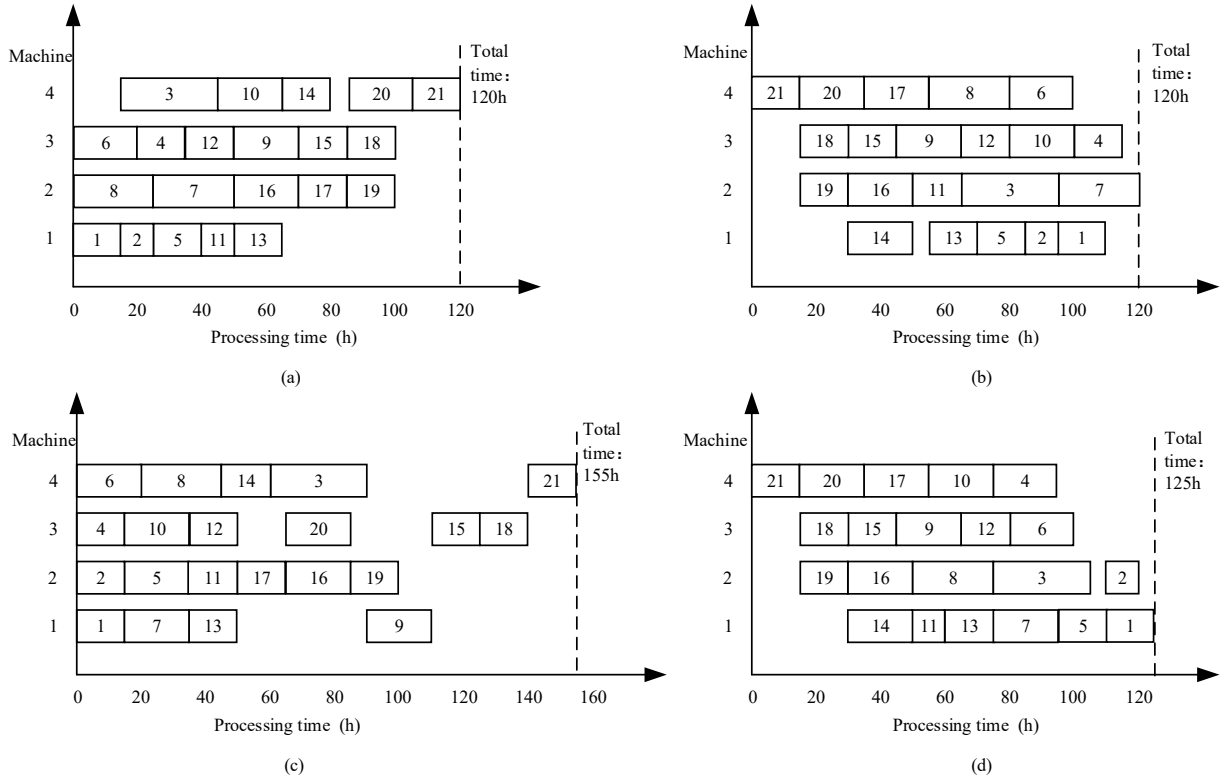


Figure 14. Gantt charts obtained by four algorithms: (a) Gantt chart of the DPFISA, (b) Gantt chart of the MFISA, (c) Gantt chart of the CMFISA and (d) Gantt chart of the RLFISA.

Table 5. Scales of 13 test groups.

Test	Number of operations	Number of machines	Density level
FIS_1	50	5	low
FIS_2	50	10	low
FIS_3	50	15	low
FIS_4	100	10	low
FIS_5	100	15	low
FIS_6	200	15	low
FIS_7	200	10	medium
FIS_8	100	5	medium
FIS_9	400	15	medium
FIS_10	400	20	medium
FIS_11	200	5	high
FIS_12	600	20	high
FIS_13	600	15	high

The final results of the MFISA with $\alpha = 0.94$, the DPFISA, the CMFISA and the RLFISA are shown in Figures 15 and 16. In order to make the results more intuitive, the confidence intervals of the optimal solutions and the CPU times for the four algorithms under different production densities are

shown in Figures 17–19.

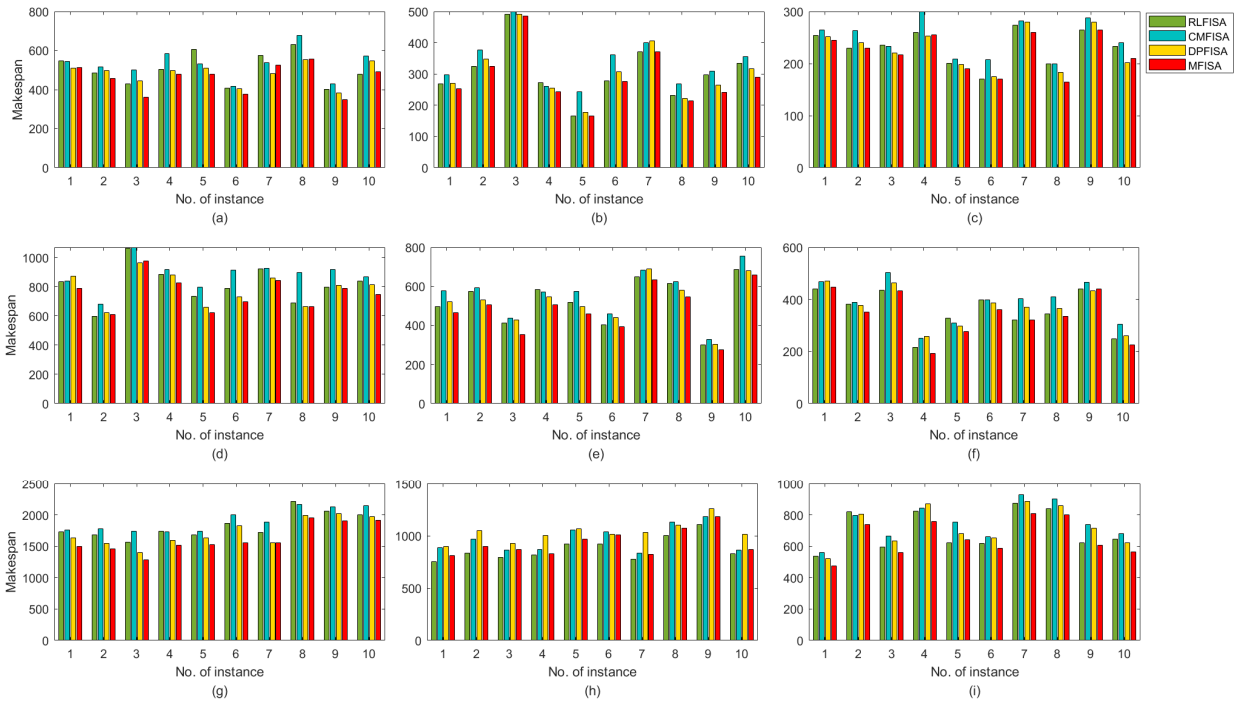


Figure 15. Results of the four algorithms on a combination of operations and equipment of [50,100,200] × [5,10,15]: (a) 50 operations on five machines, (b) 50 operations on 10 machines, (c) 50 operations on 10 machines, (d) 100 operations on five machines, (e) 100 operations on 10 machines, (f) 100 operations on 10 machines, (g) 200 operations on five machines, (h) 200 operations on 10 machines and (i) 200 operations on 10 machines.

As shown in Figure 15, when the production scale is low-density, the average result of the MFISA with different values of α is the best. Compared with the average result, the solution with α = 0.94 was not the best among all α values, but it was still better than the other three algorithms. It indicates that, regardless of the α value, overall, the MFISA outperforms the other algorithms. Among the other algorithms, the RLFISA is the most recently published algorithm, but its performance is worse than the DPFISA. However, the RLFISA had the best running time of the four algorithms, while the MFISA was the slowest.

As shown in Figure 16, when the production scale is medium-density, the average results of the MFISA are still the best, but the effect is weaker than that of low density. When α is 0.94, the performance of the MFISA is still better than that of the other three algorithms. On the whole, most of the results of the MFISA are better than those of the other algorithms. The optimization effect of the RLFISA was better than that of the other two algorithms. However, the CMFISA had the best running time of the four algorithms, while the MFISA was the slowest.

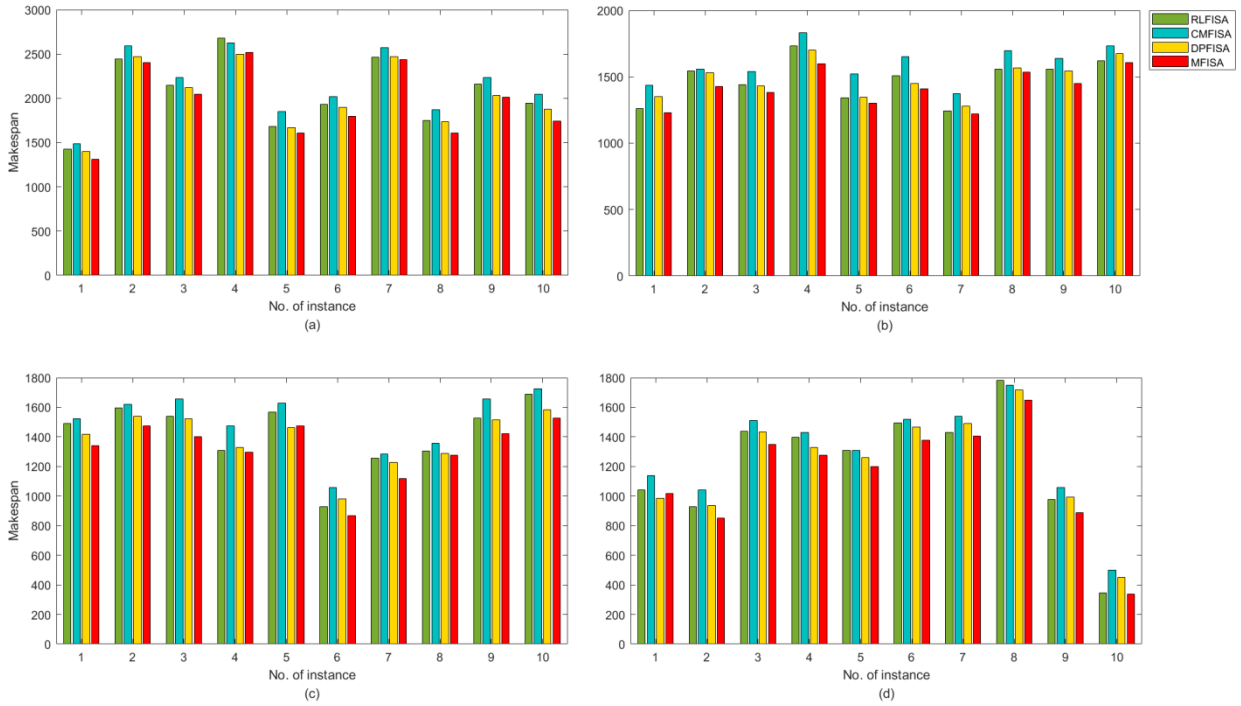
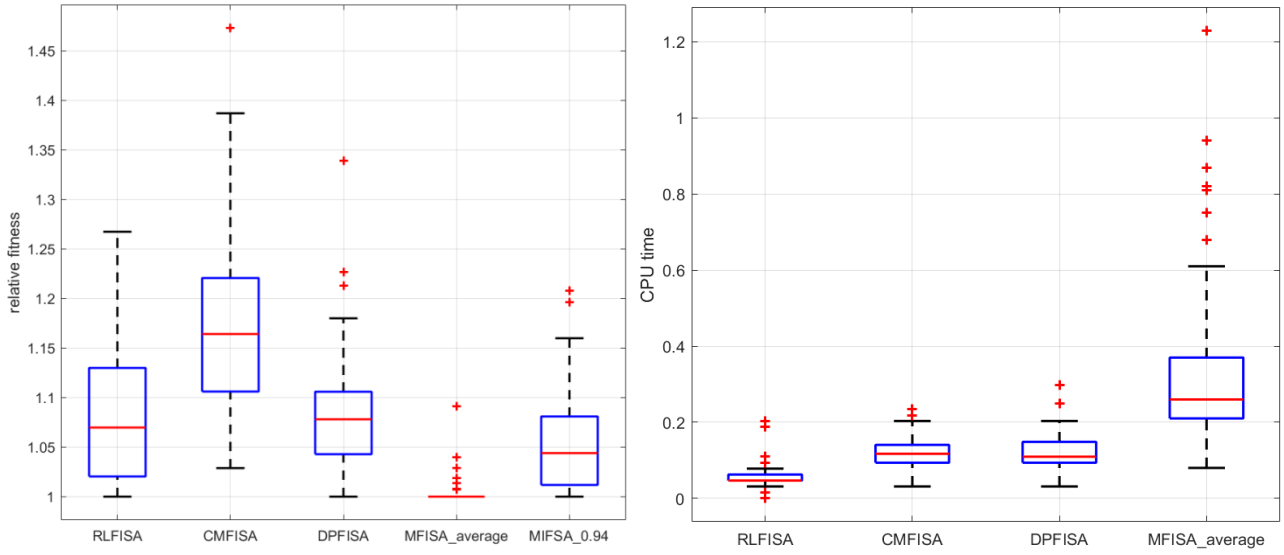
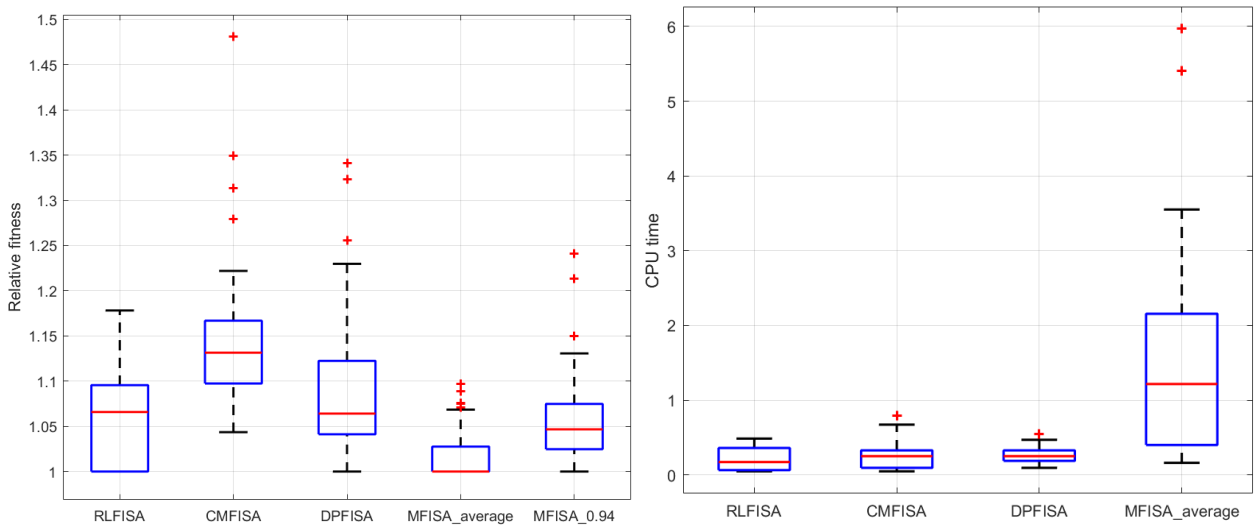


Figure 16. Results of the four algorithms on a combination of operations and equipment of [400,600] × [15,20]: (a) 400 operations on 15 machines, (b) 400 operations on 20 machines, (c) 600 operations on 15 machines and (d) 600 operations on 20 machines.



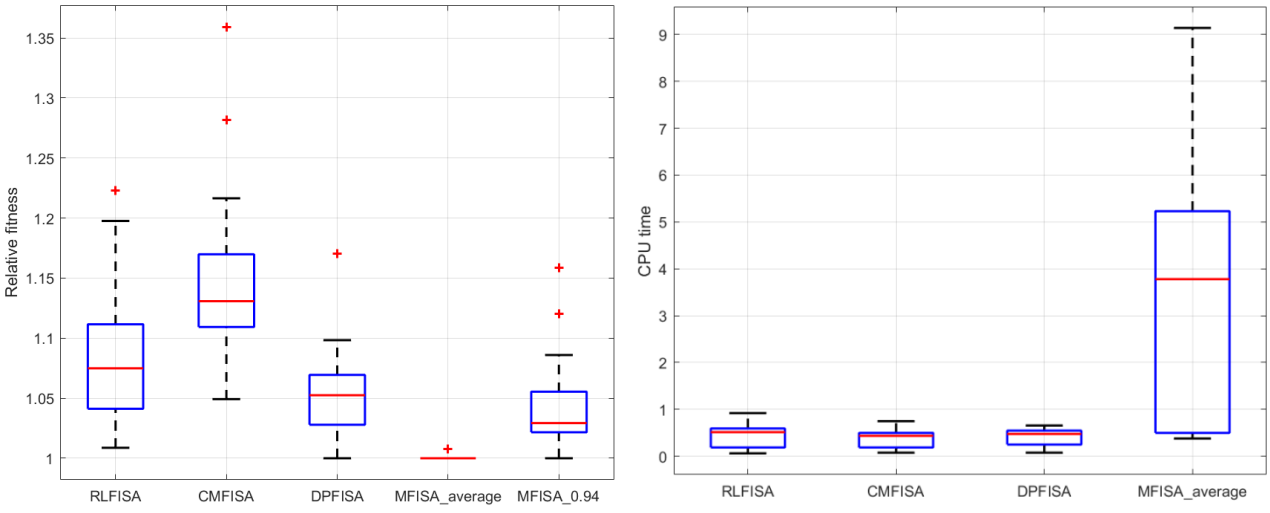
(a) (b)

Figure 17. Comparison results for four algorithms for low density: (a) relative fitness and (b) CPU time.



(a) (b)

Figure 18. Comparison results for four algorithms for medium density: (a) relative fitness and (b) CPU time.



(a) (b)

Figure 19. Comparison results for four algorithms for high density: (a) relative fitness and (b) CPU time.

As shown in Figure 19, the average results of the MFISA are still the best when the production scale is high-density. When α is 0.94, the performance of the MFISA is still better than the other three algorithms, but the running time is much longer than those of the other algorithms. Overall, the MFISA outperformed the other algorithms for the most instances. Among the other three algorithms, the DPFISA had a better optimization effect.

Based on Figures 17–19, it can be found that the optimization effect of the MFISA always keeps a high level no matter the value of α. The optimization effects of the RLFISA and the CMFISA

obviously decrease with the increase of the production density. The optimization effect of the DPFISA improves with the increase of the production density. The running times of the four algorithms increase greatly with the increase of production density.

6. Discussion

In summary, the complementarity of all strategies has great influence on the performance of the proposed strategies when applied to solve both the MFISP and FISP.

For the FISP, the previous approaches involved studying the characteristics of the processing tree by focusing on the shortest processing time of each operation to simplify the flexible problem, such as using the DPFISA and CMFISA. This ignores the influence of other processing times on all processing, artificially reducing the effect of optimization. The proposed algorithm makes up for this shortcoming and uses the average processing time to define the remaining workload of the processing tree, that is, the RSP. The OOAS was designed based on the RSP; it reduces the workload of the machines by selecting the proper competitive operations, and it avoids the accumulation of serial operations by making the operations on the critical path complete earlier.

Previous event-driven-based approaches only dispatch operations per driven time according to one certain principle, such as CMFISA. This may cause device load imbalance, resulting in longer processing times for concurrent operations. Besides, some algorithms only focus the schedulable operations at a current driven time or current layer priority in the processing tree, such as the DPFISA, RLFISA and CMFISA. This may cause new schedulable processes generated at other driven times or layers to lose potentially idle devices. To avoid the above shortages, the proposed algorithm uses the POS to supplement the results of employing the OOAS in parallel, and it uses the POPS to backtrack the driven times. The POS maximizes the avoidance of the empty load of idle machines at the driven time by adjusting the scheduling of the planned operations, so as to improve the parallelism between parallel processing operations. The POPS achieves the preemption of the new schedulable operations, providing a better solution to previous driven times.

The experimental results in Figures 17–19 show that the MFISA performs better than the comparison algorithms without considering the production scale density. And, a suitable α can control it to find a more optimal solution. Since the change of α is controlled in a small range (0.9~1) and there is no obvious rule between the results, α can be randomly selected within this range. Also, since the running time of the proposed algorithm is short, it can also be run several times with different α to get a more optimized result.

For the MFISP, in addition to the above advantages, the FODS is very helpful for the proposed algorithm when dynamically determining whether the current flexible operation is optimal. However, typical dispatching rules have no special strategy to deal with the flexible planning path group, so flexible operations cannot be optimally scheduled. The experimental results in Table 4 show that the MFISA is better than the comparison algorithms even when the operation scale is increased.

7. Conclusions and future research

In this paper, we address the MFISP for complex products. We establish the relevant mathematical model and build the extension processing operation tree by mapping the flexible sequential operation groups and the flexible planning path groups to the AND sub-graphs and OR sub-graphs. Then, we put

forward the MFISA to solve the scheduling of the MFISP. Therefore, the conclusions are as follows.

1) The FODS ensures the compact serial processing between flexible operations and shortens the completion time of products as much as possible.

2) The OOAS reduces the workload of the machines by selecting the proper competitive operations and avoids the accumulation of serial operations by making the operations on the critical path complete earlier.

3) The POS avoids the empty load of the machines at the driven time by readjusting the scheduling scheme.

4) The POPS achieves the preemption of the new schedulable operations.

5) The MFISA can not only effectively solve the MFISP, but it can also better solve the FISP.

Based on the MFISA proposed in this paper, it can also be mixed with strategies for multi-workshops or other constrains to solve an extension of the MFISP, and it can be further extended to dynamic environments.

Acknowledgments

We would like to acknowledge the support of the National Natural Science Foundation of China (Grant no. 61772160).

Conflict of interest

The authors declare that there is no conflict of interest.

References

1. V. M. Valenzuela-Alcaraz, M. A. Cosio-Leon, A. D. Romero-Ocano, C. A. Brizuela, A cooperative coevolutionary algorithm approach to the no-wait job shop scheduling problem, *Expert Syst. Appl.*, **194** (2022). https://doi.org/10.1016/j.eswa.2022.116498
2. M. A. Salido, J. Escamilla, F. Barber, A. Giret, D. Tang, M. Dai, Energy efficiency, robustness, and makespan optimality in job-shop scheduling problems, *AI EDAM*, **30** (2016), 118–125. https://doi.org/10.1017/S0890060415000335
3. M. Dai, Z. Zhang, A. Giret, M. A. Salido, An enhanced estimation of distribution algorithm for energy-efficient job-shop scheduling problems with transportation constraints, *Sustainability*, **11** (2019), 3085. https://doi.org/10.3390/su11113085
4. X. Hao, M. Gen, L. Lin, G. A. Suer, Effective multiobjective EDA for bi-criteria stochastic job-shop scheduling problem, *J. Intell. Manuf.*, **28** (2017), 833–845. https://doi.org/10.1007/s10845-014-1026-0
5. Y. Wang, X. Wang, Inventory based two-objective job shop scheduling model and its hybrid genetic algorithm, *Appl. Soft Comput.*, **13** (2013), 1400–1406. http://dx.doi.org/10.1016/j.asoc.2012.03.073
6. S. Nguyen, M. Zhang, M. Johnston, K. C. Tan, Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming, *IEEE Trans. Evol. Comput.*, **18** (2013), 193–208. http://dx.doi.org/10.1109/TEVC.2013.2248159

7. H. Wang, B. R. Sarker, J. Li, J. Li, Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q-learning, *Int. J. Prod. Res.*, **59**(2021), 5867–5883. https://doi.org/10.1080/00207543.2020.1794075
8. K. C. Ying, P. Pourhejazy, C. Y. Cheng, C. H. Wang, Cyber-physical assembly system-based optimization for robotic assembly sequence planning, *J. Manuf. Syst.*, **58** (2021), 452–466. https://doi.org/10.1016/j.jmsy.2021.01.004
9. A. De Giorgio, A. Maffei, M. Onori, L. Wang, Towards online reinforced learning of assembly sequence planning with interactive guidance systems for industry 4.0 adaptive manufacturing, *J. Manuf. Syst.*, **60** (2021), 22–34. https://doi.org/10.1016/j.jmsy.2021.05.001
10. A. Baykasoglu, F. S. Madenoglu, A. Hamzadayi, Greedy randomized adaptive search for dynamic flexible job-shop scheduling, *J. Manuf. Syst.*, **56** (2020), 425–451. https://doi.org/10.1016/j.jmsy.2020.06.005
11. A. Baykasoglu, F. S. Madenoglu, Greedy randomized adaptive search procedure for simultaneous scheduling of production and preventive maintenance activities in dynamic flexible job shops, *Soft Comput.*, **25** (2021), 14893–14932. https://doi.org/10.1007/s00500-021-06053-0
12. A. Vital-Soto, A. Azab, M. F. Baki, Mathematical modeling and a hybridized bacterial foraging optimization algorithm for the flexible job-shop scheduling problem with sequencing flexibility, *J. Manuf. Syst.*, **54**(2020), 74–93. https://doi.org/10.1016/j.jmsy.2019.11.010
13. G. Gong, R. Chiong, Q. Deng, X. Gong, A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility, *Int. J. Prod. Res.*, **58** (2020), 4406–4420. https://doi.org/10.1080/00207543.2019.1653504
14. R. Li, W. Gong, C. Lu, A reinforcement learning based RMOEA/D for bi-objective fuzzy flexible job shop scheduling, *Expert Syst. Appl.*, **203** (2022), 117380. https://doi.org/10.1016/j.eswa.2022.117380
15. R. Li, W. Gong, C. Lu, Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time, *Comput. Ind. Eng.*, **168** (2022), 108099. https://doi.org/10.1016/j.cie.2022.108099
16. R. Liu, R. Piplani, C. Toro, Deep reinforcement learning for dynamic scheduling of a flexible job shop, *Int. J. Prod. Res.*, **60** (2022), 4049–4069. https://doi.org/10.1080/ 00207543.2022.2058432
17. K. Lei, P. Guo, W. Zhao, Y. Wang, L. Qian, X. Meng, et al., A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem, *Expert Syst. Appl.*, **205** (2022), 117796. https://doi.org/10.1016/j.eswa.2022.117796
18. Y. Du, J. Li, X. Chen, P. Duan, Q. Pan, Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem, *IEEE Trans. Emerging Top. Comput. Intell.*, **2022** (2022). https://doi.org/10.1109/TETCI.2022.3145706
19. X. Li, K. Xing, Iterative widen heuristic beam search algorithm for scheduling problem of flexible assembly systems, *IEEE Trans. Ind. Inf.*, **17** (2021), 7348–7358. https://doi.org/10.1109/TII.2021.3049338
20. C. Finetto, M. Faccio, G. Rosati, A. Rossi, Mixed-model sequencing optimization for an automated single-station fully flexible assembly system (F-FAS), *Int. J. Adv. Manuf. Technol.*, **70** (2014), 797–812. https://doi.org/10.1007/s00170-013-5308-z
21. A. Hottenrott, M. Schiffer, M. Grunow, Flexible assembly layouts in smart manufacturing: An impact assessment for the automotive industry, *IISE Trans.*, **2022** (2022). https://doi.org/10.1080/24725854.2022.2124470

22. W. Ren, J. Wen, Y. Yan, Y. Hu, Y. Guan, J. Li, Multi-objective optimisation for energy-aware flexible job-shop scheduling problem with assembly operations, *Int. J. Prod. Res.*, **59** (2021), 7216–7231. https://doi.org/10.1080/00207543.2020.1836421
23. W. Lin, Q. Deng, W. Han, G. Gong, K. Li, An effective algorithm for flexible assembly job-shop scheduling with tight job constraints, *Int. Trans. Oper. Res.*, **1** (2020). https://doi.org/10.1111/itor.12767
24. P. Fattahi, N. B. Rad, F. Daneshamooz, S. Ahmadi, A new hybrid particle swarm optimization and parallel variable neighborhood search algorithm for flexible job shop scheduling with assembly process, *Assem. Autom.*, **40** (2020), 419–432. https://doi.org/10.1108/AA-11-2018-0178
25. X. Wu, X. Liu, N. Zhao, An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem, *Memet. Comput.*, **11** (2019), 335–355. https://doi.org/ 10.1007/s12293-018-00278-7
26. S. Zhang, S. Wang, Flexible assembly job-shop scheduling with sequence-dependent setup times and part sharing in a dynamic environment, constraint programming model, mixed-integer programming model, and dispatching rules, *IEEE Trans. Eng. Manage.*, **65** (2018), 487–504. https://doi.org/ 10.1109/TEM.2017.2785774
27. X. Li, J. Lu, C. Yang, J. Wang, Research of flexible assembly job-shop batch-scheduling problem based on improved artificial bee colony, *Front. Bioeng. Biotechnol.*, **10** (2022). https://doi.org/10.3389/fbioe.2022.909548
28. Z. Xie, X. Zhang, Y. Xia, J. Yang, Y. Xin, A hybrid method of heuristic algorithm and constraint programming for no-wait integrated scheduling problem, *J. Int. Technol.*, **22** (2021), 1085–1092. https://doi.org/10.53106/160792642021092205012
29. Z. Xie, W. Zhou, Z. Yu, Integrated scheduling algorithm for dynamic adjustment of equipment maintenance start time, *J. Mech. Eng.*, **57** (2021), 240–246. https://doi.org/10.3901/JME.2021.04.240
30. Z. Xie, N. Lv, Integrated scheduling algorithm with pre-start device, *J. Mech. Eng.*, **57** (2021), 217–225. https://doi.org/ 10.3901/JME.2021.17.217
31. Z. Xie, H. Teng, J. Ming, X. Yue, A two-workshop collaborative, integrated scheduling algorithm considering the prescheduling of the root-subtree processes, *Comput. Intell. Neurosci.*, **2022** (2022). https://doi.org/10.1155/2022/9065638
32. X. Zhan, Z. Xie, D. Yao, Integrated scheduling algorithm of two workshops based on process end time driven and processing area priority, *Electronics*, **11** (2022), 2594. https://doi.org/10.3390/electronics11162594
33. Y. Gao, Z. Xie, X. Yu, A hybrid algorithm for integrated scheduling problem of complex products with tree structure, *Multimedia Tools Appl.*, **79** (2020), 32285–32304. https://doi.org/doi.org/10.1007/s11042-020-09477-2
34. Z. Wang, C. Lu, An integrated job shop scheduling and assembly sequence planning approach for discrete manufacturing, *J. Manuf. Syst.*, **61** (2021), 27–44. https://doi.org/10.1016/j.jmsy.2021.08.003
35. Y. Gao, Z. Xie, D. Yang, X. Yu, Flexible integrated scheduling algorithm based on remaining work probability selection coding, *Expert Syst.*, **38** (2021), e12683. https://doi.org/10.1111/exsy.12683
36. Z. Xie, Z. Gui, J. Yang, Dynamic parallel integrated flexible scheduling algorithm based on device driver and essential path, *J. Mech. Eng.*, **50** (2014), 203–212. https://doi.org/10.3901/JME.2014.18.203

37. Z. Xie, H. Zhou, J. Yu, Z. Gui, Conflict mediation algorithm of integrated flexible scheduling based on device driver, *Trans. Beijing Inst. Technol.*, **34** (2014), 1150–1156. https://doi.org/10.3969/j.issn.1002-137X.2013.04.042
38. H. Lu, G. Huang, H. Yang, Integrating order review/release and dispatching rules for assembly job shop scheduling using a simulation approach, *Int. J. Prod. Res.*, **49** (2011), 647–669. https://doi.org/10.1080/00207540903524490
39. Z. Xie, S. Hao, G. Ye, G. Tan, A new algorithm for complex product flexible scheduling with constraint between jobs, *Comput. Ind. Eng.*, **57** (2009), 766–772. https://doi.org/10.1016/j.cie.2009.02.004
40. Z. Xie, Q. Wang, Flexible integrated scheduling algorithm based on reverse order layer priority, *J. Electron. Inf. Technol.*, **44** (2022), 1554–1562. https://doi.org/10.11999/JEIT211378
41. R. El-Khalil, Z. Darwish, Flexible manufacturing systems performance in U.S. automotive manufacturing plants, a case study, *Prod. Plann. Control*, **30** (2019), 48–59. https://doi.org/10.1080/09537287.2018.1520318
42. X. Yang, J. Liu, Q. Chen, N. Mao, Performance analysis of flexible assembly job shop scheduling under variable distrurbance intensity, *Comput. Integr. Manuf. Syst.*, **27** (2021), 800–814. https://doi.org/10.13196/j.cims.2021.03.013

AIMS Press