IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received November 26, 2020, accepted December 10, 2020, date of publication December 14, 2020, date of current version December 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3044606

Integrated Optimization Approach of Hybrid Flow-Shop Scheduling Based on Process Set

XIXING LI[1], HONGTAO TANG[2], ZHIPENG YANG[2], RUI WU[1], AND YABO LUO[2]

[1]Hubei Key Laboratory of Modern Manufacturing and Quality Engineering, School of Mechanical Engineering, Hubei University of Technology, Wuhan 430068, China

[2]Hubei Key Laboratory of Digital Manufacturing, School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan 430070, China

Corresponding author: Hongtao Tang (615912286@qq.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 51805152, Grant 51705384, Grant 52075401, and Grant 51875430; and in part by the Scientific Research Foundation for High-level Talents of Hubei University of Technology under Grant GCRC2020009.

ABSTRACT Considering that process planning and production scheduling are independent of each other in a hybrid flow-shop, this study categorizes the process route into parallel process-set, batch set and unordered process-set, and builds a multi-objective optimization model to minimize the maximum completion time and the minimum processing cost. An improved artificial bee colony algorithm has been developed to solve the model. A segmented decoding method based on the insertion principle and the release time of the predecessor process is proposed to effectively use the idle time of the machine. A dynamic triggering neighborhood mechanism is introduced to enhance the local searchabilit of the algorithm. Finally, the feasibility and effectiveness of this algorithm to solve such problems are verified via simulation experiments.

INDEX TERMS Process planning, production scheduling, hybrid flow-shop, improved artificial bee colony algorithm.

I. INTRODUCTION

An appropriate feasible workshop scheduling plan is key to transform and upgrade the manufacturing industry and improve manufacturing efficiency, particularly in process industries, such as chemical processes, textiles, metallurgical, printed circuit boards, and automobile manufacturing enterprises [1]. Generally, a hybrid flow-shop (HF) consists of multiple processing stages. There are strict order constraints between different processing stages, with multiple parallel machines in at least one processing stage. The research objective of HF scheduling (HFS) is to strategize a feasible and effective arrangement of a processing sequence of different manufacturing tasks to satisfy the goals pursued by managers, such as the equipment load balance in each processing stage, minimum total flow time, and minimum maximum completion time. This has been proven to be an NP-hard problem [2]. Most studies consider certain assumptions (e.g., ignoring the setup times of operations, non-sequence-dependent) to simplify the construction model [3]. However, in a real-time production process, there are different production constraints (e.g., one stage of the machine is a batch machine instead of a single parallel machine, or the processing sequence can be interchanged) and different dynamic events (e.g., new manufacturing tasks or machine failure) that increases the corresponding processing time to delay the total completion time of the original scheduling scheme.

Recently, the methods to solve HFS have been generally divided into two categories: accurate method and approximate method [4]. Accurate methods include mathematical programming method [5], [6], branch and bound method [7], [8] and Lagrange Relaxation algorithm [9], which can solve small scale simple problems. However, the solution space of practical problems is generally large and its calculation time is unacceptable. The approximate method is an experience-based solution algorithm, generally a range of solving time, space, and feasible solutions has been given. The solution speed is relatively fast, and the result is a feasible approximate optimal solution. The method mainly includes a heuristic approach and hybrid intelligence optimization algorithm. For the integrated optimization model of permutation flow shop scheduling problems with HFS, Pang *et al.* developed an improved fireworks algorithm to minimize the makespan [10]. Mirsanei *et al.* proposed a novel

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiwu Li.

simulated annealing algorithm to produce a reasonable manufacturing schedule within an acceptable computational time for solving the HFS with sequence-dependent setup times [2]. Zhou *et al.* proposed a hybrid different algorithm with estimation of distribution algorithm to solve a reentrant HFS, where inspection and repair operations are carried out as soon as a layer has completed fabrication [11]. Yu *et al.* presented a genetic algorithm incorporating a new decoding method to solve the HFS with unrelated machines and machine eligibility to minimize the total tardiness [12]. Marichelvam *et al.* developed a cuckoo search metaheuristic algorithm to minimize the makespan for the multistage HFS scheduling problem [13]. Liu *et al.* combined the estimation of distribution algorithm and differential evolution algorithm to address a specialized two-stage HFS scheduling problem with parallel batching machines; a job-dependent deteriorating effect and non-identical job sizes were considered simultaneously [14]. Choong *et al.* combined particle swarm optimization with simulated annealing and tabu search, respectively, which were applied to the HFS scheduling problem [15].

The artificial bee colony (ABC) algorithm is a metaheuristic algorithm based on relative populations. It was first introduced by Karaboga to solve multi-variable, multi-modal continuous functions [16]. It was inspired by the behavior of bees collecting honey, it has fast convergence speed and strong optimization ability when compared with other metaheuristic algorithms [17]– [20]. Therefore, further intensive studies on the application of ABC algorithm have been conducted in the research field of job scheduling, such as single machine scheduling [21], multi-machine parallel scheduling [22], flexible job shop scheduling [23], open shop scheduling [24], flow shop scheduling [25].

The ABC algorithm has received much attention for its application in the HFS scheduling problem. Lin *et al.* developed a hybrid ABC algorithm with bi-directional planning to minimize the makespan of scheduling multistage HFS with multiprocessor tasks. Computational evaluations of two well-known benchmark problem sets supported the proposed hybrid ABC algorithm with high performance against the best-so-far algorithm [26]. Cui *et al.* proposed an improved discrete ABC algorithm that combined a novel differential evolution and modified variable neighborhood search to minimize the makespan of HFS [27]. Li *et al.* proposed an improved discrete ABC algorithm to solve the hybrid flexible flowshop scheduling problem with dynamic operation skipping features in molten iron systems. A dynamic encoding mechanism, flexible decoding strategy, and right-shift strategy were proposed [28]. Li *et al.* proposed a hybrid ABC algorithm to solve a parallel batching-distributed flowshop problem with deteriorating jobs. Two types of problem-specific heuristics were introduced, and five types of local search operators were designed [29]. To solve the large-scale HFS scheduling problem with limited buffers, Li *et al.* combined the ABC algorithm with tabu search to minimize the maximum completion time [30]. Considering a two-stage HFS with multilevel product structures and requirement operations, Kheirandish *et al.* developed an ABC algorithm along with a genetic algorithm to obtain near-optimal solutions to minimize the maximum completion time in reasonable run-times [31]. Pan *et al.* developed an effective discrete ABC algorithm with a hybrid representation and combination of forward decoding and backward decoding methods to solve the HFS scheduling problem in order to minimize the makespan [32].

Generally, several unexpected disruptions occur in realistic production systems. Peng *et al.* studied a real-world HFS rescheduling problem in which machine breakdown was considered as the disruption. They developed an improved ABC algorithm with a population initialization heuristic and worst solution replacement strategy [33]. Li *et al.* addressed the steelmaking scheduling problem with continuous casting constraint and resource constraints simultaneously. They proposed several heuristics and developed a discrete ABC algorithm with a two-phased-based encoding mechanism and local search procedure [34]. To save energy in sustainable manufacturing, Zhang *et al.* studied an HFS green scheduling problem with variable machine processing speeds to minimize the mankespan and total energy consumption, and developed a decomposition-based multiobjective discrete ABC algorithm [35]. Generally, a conventional foundry manufacturing process (as shown in **FIGURE 1**) demonstrates partial sequence flexibility and batch processing machines. Furthermore, several other complicated characteristics are as follows: (1) different processes of the same job can be processed at the same time (e.g., modeling and core-making stage), (2) the same process of different jobs can be processed in batches (e.g., melting stage), and (3) the processing order can be exchanged between different processes of the same job (e.g., detection phase).

Therefore, our research aims to propose an integrated optimization approach for HFS scheduling based on the process-set division scheme to support production process tracking and monitoring. The main contributions of our study are as follows: (1) a feasible division method of process-set is proposed to satisfy the requirement of integrating the process route with scheduling, (2) an HFS scheduling model with batch processors in the middle stage was considered, (3) an initialization method based on opposite learning was proposed, and (4) four effective neighborhood search strategies were formulated to improve the optimization ability of the solving algorithm. This paper is organized as follows: problem description and mathematical modeling are presented in Section 2, an improved ABC algorithm with dynamic trigger neighborhood search (DTNS) is developed in Section 3, a case analysis is demonstrated in Section 4, and conclusions and scope for future work are provided in Section 5.

II. PROBLEM DESCRIPTION AND MATHEMATICAL MODELING

A. PROCESSES-SET DEFINITION

Assuming n jobs, and each job J_i has m operations, and that its technological route is $\{O_{i,1}, O_{i,2} \ldots, O_{i,j} \ldots, O_{i,k} \ldots,$

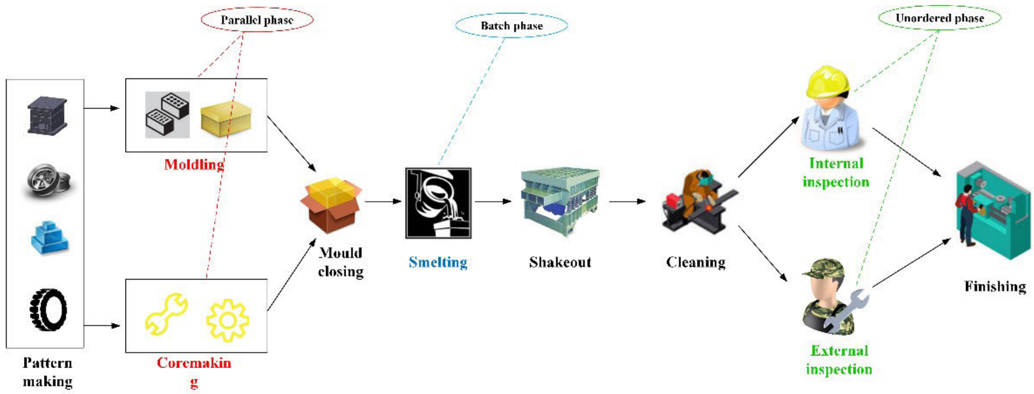


FIGURE 1. Conventional foundry manufacturing process.

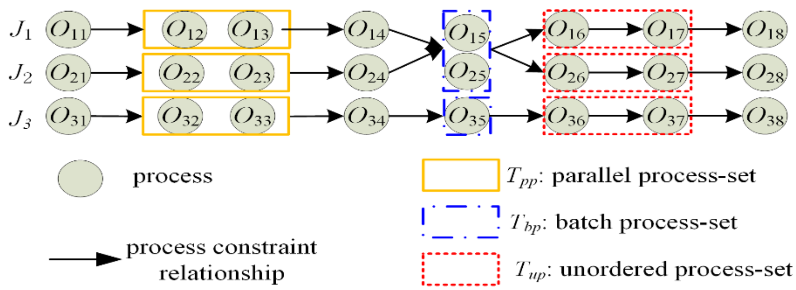


FIGURE 2. Process roadmap.

$O_{i,n}, \ldots, O_{i,m}\}$, there are many different process-set in the process route. However, there is only one process-set for each job. As shown in **FIGURE 2**, based on the directed acyclic graph [36], this work studies the phenomenon where the process route contains three special process-set: parallel process-set (T_{pp}), batch process-set (T_{bp}), and unordered process-set (T_{up}). The difference between T_{pp} and T_{up} is whether the internal processes can be processed at the same time. There is no processing order requirement for the processes in T_{pp} and T_{up}.

1) **Parallel process-set** T_{pp}: $T_{ppi} =< \frac{O_{i,j}}{O_{i,j+1}} >$, i.e., the processes $O_{i,j}$ and $O_{i,j+1}$ of the job J_i in the process route can be processed by different parallel machines simultaneously; $O_{i,j}$ and $O_{i,j+1}$ are called parallel processes. As shown in **FIGURE 2**, the parallel processes $O_{1,2}$ and $O_{1,3}$ form a parallel process-set T_{pp1}.
2) **Batch process-set** T_{bp}: $T_{bpn} =\ll J_{1,k}, \ldots, J_{n,k} > n >$, i.e., the k process of n jobs in the batch stage can be processed in batches on a batch processing machine. The corresponding process is called batch process. As shown in **FIGURE 2**, the batch process $O_{1,5}$ and $O_{2,5}$ form the batch process-set T_{bp1}, and $O_{3,5}$ form the batch process-set T_{bp2}.
3) **Unordered process-set** T_{up}: $T_{upi} =< O_{i,n}, O_{i,n+1} >$, i.e., the process $O_{i,n}$ and $O_{i,n+1}$ of the job J_i in the process route have no processing order requirements, but they cannot be simultaneously processed, $O_{i,n}$ and $O_{i,n+1}$ are called unorder processes. As shown in **FIGURE 2**, the unordered processes $O_{1,6}$ and $O_{1,7}$ form the unordered process-set T_{up1}.

B. PROBLEM DESCRIPTION

This work studies the HFS scheduling problem with batch processors and flexibly sequenced processes. Each job passes through multiple processing stages, in sequence, of a same process route; each stage has at least one parallel machine. This problem contains three special processing stages: parallel processing, batch processing, and unorder processing stages. The remaining stages are non-parallel single processing stages. In the parallel processing and unorder processing stages, we should determine the processing sequence. In the batch processing stage, we should determine the grouping and batching of a job. In the grouping phase, the batch processor is divided into several job clusters based on the job selection first, and then further decomposed according to the threshold method. Therefore, the described problem can be divided into three sub-problems: (1) determining each job's process route, (2) assigning a processing machine to each job, and (3) determining each job's status on the machine processing order.

Moreover, the following assumptions are considered for the problem addressed in this study:

(1) All machines are available at the initial time 0.
(2) Each job has strict processing order constraints between the other processes except the parallel process set and unordered process set.

TABLE 1. Notations.

Notation	Signification	Notation	Signification
n	number of jobs	E_{ij}	completion time of $O_{i,j}$
m	number of operations in each job	P_{ijt}	processing time of O_{ij} on M_t
b	number of batches	RC_i	raw material cost of J_i
l	number of machines	SC_t	static waiting cost of M_t
$O_{i,j}$	jth operation in J_i	DC_t	dynamic processing cost of M_t
W_i	weight of the J_i	T_t^s	static waiting time of M_t
W_{Bk}	weight of batch B_k	T_t^d	dynamic processing time of M_t
M_t	machine set	S_{ij}	start time of $O_{i,j}$
Q	threshold of batch machines	S_{Tpi}	start time of subsequent T_{ppi}
E_{Tpi}	completion time of precursor T_{ppi}	S_{Tbpi}	start time of T_{bpi}
E_{Tbi}	completion time of all precursor T_{bpi}	α	fixed processing time
P_{Tbpk}	processing time of batch B_k	β	coefficient of weight

(3) The processing time of each parallel process set and unordered process set should be processed between its predecessor and subsequent processes.
(4) Except for the batch process set, the remaining operations are processed on a single parallel machine.
(5) The batch processing machine has a processing threshold, and the weight of each batch of jobs cannot exceed the processing threshold of the batch processing machine.
(6) The start time of the batch process set is not earlier than the maximum completion time of all its predecessor processes and the previous batch process set.
(7) The completion time of the batch process set should be earlier than the start time of all subsequent processes.
(8) The processing time of the batch processing stage is related to the weight of the batch of jobs, and the processing time of the jobs in the remaining stages is fixed.
(9) There is no influence on each other.
(10) Each job can belong to only one processing batch.
(11) Each machine will run until all jobs pertaining to the machine are processed and stopped.

C. MATHEMATICAL MODELING

1) NOTATIONS

The notations used throughout this paper are list in TABLE 1.

2) DECISION VARIABLES

$$X_{ijt}\begin{cases}1, & O_{i,j} \text{ is processed in } M_t\\ 0, & \text{else}\end{cases}$$

$$Y_{ijk}\begin{cases}1, & O_{i,j} \text{ is belong to } B_k\\ 0, & \text{else}\end{cases}$$

3) OBJECTIVES

The objective of this study is to minimize the makespan and cost. The mathematical model is described according to the aforementioned assumptions and notations.

a: MAKESPAN

The completion time is the longest time consumed after the completion of all processes, expressed as f_1, as shown below.

$$\min f_1 = \max(E_{ij}) \quad \forall 1 \le i \le n,\ 1 \le j \le m \tag{1}$$

b: PRODUCTION COST

The completion cost is determined in two parts: material and processing costs. Further, the processing cost includes two parts: standby state processing cost and working state processing cost, denoted by f_2, as shown below.

$$\min f_2 = \sum_{i=1}^{n} RCi + \sum_{t=1}^{m} (SCt \times T_t^s + DCt \times T_t^d) \tag{2}$$

4) CONSTRAINTS

$$E_{T_{pi}} \le S_{ij} \quad \forall 1 \le i \le n,\ O_{i,j} \in (T_{pp} \cup T_{dp}) \tag{3}$$

$$S_{ij} + P_{ij} \le S_{T_{pi}} \quad \forall 1 \le i \le n,\ O_{i,j} \in (T_{pp} \cup T_{dp}) \tag{4}$$

$$S_{ij} + X_{ijt} \times P_{ijt} \le E_{ij} \quad \forall O_{i,j} \notin (T_{pp} \cup T_{dp}),\ \ t \in M \tag{5}$$

$$\sum_{j=1}^{m} X_{ijt} = 1 \quad \forall 1 \le i \le n,\ \ t \in M \tag{6}$$

$$\sum_{i=1}^{n} Y_{ijk} = 1 \quad \forall 1 \le k \le b,\ O_{i,j} \in T_{bp} \tag{7}$$

$$W_{Bk} = Y_{ijk} \times W_i \quad \forall 1 \le i \le n,\ 1 \le k \le b,\ \ O_{i,j} \in T_{bp} \tag{8}$$

$$W_{Bk} \leq Q \quad \forall 1 \leq k \leq b \tag{9}$$
$$P_{Tbpk} = \alpha + \beta \times W_{Bk} \quad \forall 1 \leq k \leq b \tag{10}$$
$$\max(E_{Tbk}) \leq S_{Tbpk} \quad \forall 1 \leq k \leq b \tag{11}$$

Constraints (3) and (4) indicate that the processing time of the parallel process set and unordered process set is between their predecessor and successor processes. Constraint (5) indicates that except the parallel process and unordered process sets, there are strict processing order constraints. Constraint (6) indicates that only one processing machine can be selected for each process of the job. Constraint (7) indicates that each job can only belong to one processing batch. Constraint (8) indicates each batch total weight is the sum of the weights of all the jobs in the batch. Constraint (9) indicates that the weight of each batch cannot exceed the processing threshold of the batch machine. Constraint (10) indicates that the processing time of each batch set depends on the batch weight and basic melting time constant. Constraint (11) indicates that the start time of each batch process set is earlier than the completion time of all its predecessor processes.

III. SOLUTION REPRESENTATION

The ABC algorithm has the advantages of few parameters and strong versatility. It has been widely applied in solving scheduling problems. Three types of bees are defined for the original ABC algorithm: employee, onlook, and scouter bees. The number of employee and onlook bees are equal. While the employee bees correspond to the honey source (the solution number of the problem), the richness of the honey source represents the adaptation of the solution degree.

Employee stage: Employee bees are responsible for finding new honey sources. If the quality of the new honey sources is better than the original, the latter will be replaced and the new honey source information will be shared with the onlook bees.

Onlook stage: Onlook bees work by sharing the honey source information with employee bees and deciding whether to follow the employee bee to collect honey through roulette.

Scouter stage: Scouter bees update the honey source by updating the unimproved honey source several times.

The HFS problem based on the process set studied here is a multi-objective optimization problem. Therefore, this study first proposes an improved artificial bee colony (IABC) based on process coding. Then, the DTNS is introduced to improve the local optimization ability of the algorithm during the iteration process. The details are as follows.

A. ENCODING

Coding is used to solve problems represented by a set of vectors. A feasible coding method can increase the speed of convergence of an algorithm to easily find an optimal solution of a given problem. This study employs the process-machine-based coding method to generate the job scheduling sequence. Each group of vectors $X = [Xj|Xm]$,which represents an effective solution to the problem, where X_j represents the job. The appears order represents the process sequence of the job, X_m represents the machine sequence of the optional machine set corresponding to the process, and not the machine number. For instance, considering a set of vectors encoding in TABLE 2, $X = [Xj|Xm] = [31222112323231321331 1 \mid 132131112212111213231]$, the first position of X_j represents $O_{3,1}$ of job 3, and the fourth position represents $O_{2,2}$ of job 2. The corresponding fourth position in X_m represents processing on the machine M_2 of the optional machine set $\{M_2, M_3, M_4\}$.

B. DECODING

Decoding is a practical solution to a set of vector mapping problems. Because the scheduling problem studied in this work has three special processing stages, different corresponding decoding methods are employed.

1) PARALLEL DECODING

The jobs in this stage form a set of parallel operations and can be machined simultaneously. The final machining order of parallel operations is determined based on the principle of *Sort Before Insert* (SBI), as shown in **FIGURE 3**. Considering $O_{i,2}$ and $O_{i,3}$ of the parallel process-set as an example, the front operation is $O_{i,1}$ and follow-up operation is $O_{i,4}$, decoding steps as follows:

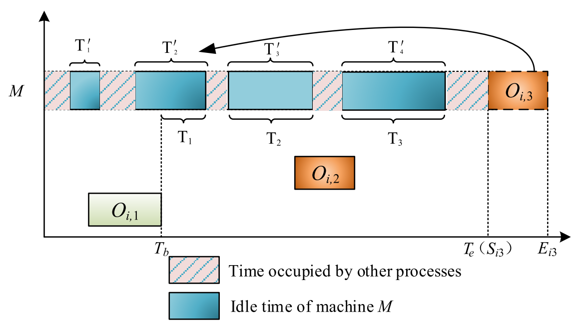


FIGURE 3. Schematic diagram of parallel decoding.

Step1: Determine the processing time period of each parallel process-set and its front operations in order regardless of the parallel processes. (In **FIGURE 3**, parallel process $O_{i,3,}$ start time $T_e = S_{i3}$, and processing time period is $[S_{i3}, E_{i3}]$);

Step2: Determine the idle time segment of processing machine M from $T_b - T_e$, and sort it in the order of the earliest idle time to obtain $T = \{T_1, \ldots T_n\}$;

Step3: Select the later process in the parallel process to the processing time period $[S_{i3}, E_{i3}]$ and insert each of them in turn with T_j until it can be put into a certain period of time. If there is no such period of time, maintain the original start-up time of $O_{i,3}$ unchanged;

Step4: Determine the start time of $O_{i,4}$. The start time is not earlier than the maximum finish time of the parallel process-set.

TABLE 2. Workpiece machining information table.

Job	Weight	Operation	Processing time						
			M_1	M_2	M_3	M_4	M_5	M_6	M_7
J_1	2	$O_{1,1}$	6	7	9	-	-	-	-
		$O_{1,2}$ (T_{pp})	-	6	7	11	-	-	-
		$O_{1,3}$ (T_{pp})	-	4	7	9	-	-	-
		$O_{1,4}$ (T_{bp})	-	-	-	-	-	-	NaN
		$O_{1,5}$ (T_{up})	-	-	5	6	7	-	-
		$O_{1,6}$ (T_{up})	-	-	4	8	9	-	-
		$O_{1,7}$	-	-	-	3	8	6	-
J_2	3	$O_{2,1}$	8	7	-	-	-	-	-
		$O_{2,2}$ (T_{pp})	-	6	5	8	-	-	-
		$O_{2,3}$ (T_{pp})	-	8	5	6	-	-	-
		$O_{2,4}$ (T_{bp})	-	-	-	-	-	-	NaN
		$O_{2,5}$ (T_{up})	-	7	-	8	-	-	-
		$O_{2,6}$ (T_{up})	-	7	9	11	-	-	-
		$O_{2,7}$ (T_{up})	-	-	-	8	8	6	-
J_3	1	$O_{3,1}$	6	8	-	-	-	-	-
		$O_{3,2}$ (T_{pp})	-	9	10	7	-	-	-
		$O_{3,3}$ (T_{pp}))	6	9	4		-	-	-
		$O_{3,4}$ (T_{bp})	-	-	-	-	-	-	NaN
		$O_{3,5}$ (T_{up})	-	8	-	9	-	-	-
		$O_{3,6}$ (T_{up})	-	6	7	4	-	-	-
		$O_{3,7}$	-	-	-	-	8	9	-

- NaN: Representative processing time is variable

2) BATCH DECODING

The processes in this stage are batch operations; different jobs can be processed in batches. This job first divides the job cluster based on the number of batch machines (each job cluster is processed on only one batch), and then groups batches according to the threshold method in each job cluster. The processing time of each batch machine is determined according to Constraint (10). Each job cluster, as shown in **FIGURE 4,** is grouped using the rules of *Early Release Time Fit*, assuming that there are a total of n jobs, where $O_{i,4}$ can be processed on the batch machine. The decoding steps are as follows:

Step1: Determine the completion time of the forward operations for all batch set of each job cluster and increments by the completion time to obtain the sorted artifacts. If its precursor operation is a parallel process-set, the entire parallel process-set is the precursor operation.

Step2: Create a new batch and place the sorted jobs in the batch in turn until the machining threshold for the batch machine is met or the batch is assigned to each job. The start time for each batch is selected as the maximum of the operations contained in the batch and completion time of the previous batch.

3) UNORDERED DECODING

The processes in this stage constitute an unordered process-set. The processing order can be exchanged between the processes, but they cannot be processed simultaneously. As shown in **FIGURE 5**, this study is based on the principle of SBI. Final processing order of unordered operations. The composition of the unordered process $O_{i,5}$ and $O_{i,6}$ is considered as an example, where the precursor process is $O_{i,4}$. The subsequent process is $O_{i,7}$, the decoding steps are as follows:

Step1: Determine the processing time period of each unordered process-set and its precursor operations in order regardless of the sequence of operations (in **FIGURE 5**, the start time of $O_{i,6}$, $T_e = S_{i3}$, and processing time period is $[S_{i3}, E_{i3}]$);

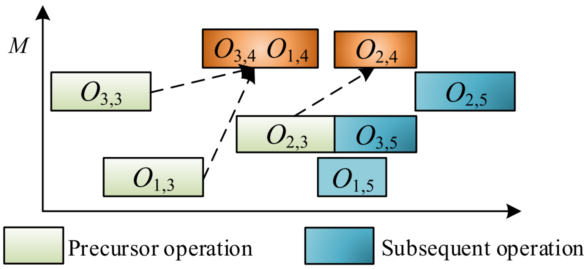


FIGURE 4. Schematic diagram of batch decoding.

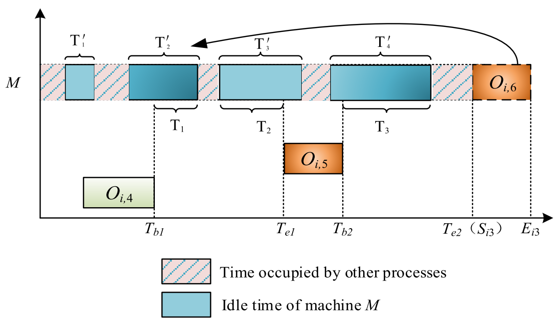


FIGURE 5. Schematic diagram of unordered decoding.

Step2: Determine the idle time segments of processing machine M in $T_{b1} \sim T_{e1}$ and $T_{b2} \sim T_{e2}$, and obtain the $T = \{T_1, \ldots, T_n\}$ in order by the earliest idle time;

Step3: Select the backward operation in the unordered operation to input its processing period of time $[S_{i3}, E_{i3}]$, until it can be placed in a certain period of time. If there is no such period of time, then maintain the original start time of $O_{i,6}$ unchanged;

Step4: Determine the start time of $O_{i,7}$. The start time is not earlier than the maximum completion time of an unordered process-set.

Considering the job information in TABLE 2 as an example, each job has seven processes, of which $O_{i,2}$ and $O_{i,3}$ are parallel processes, processes $O_{i,4}$ are batch processes, and $O_{i,5}$ and $O_{i,6}$ are unordered processes, $M_1 \sim M_6$ are single parallel machines, and M_7 is a batch machine. The processing threshold of the batch machine is 4. Assuming the processing time of two batches is $PTbpk = \alpha + \beta \times W_{Bk}$ (if $\alpha = 1$, $\beta = 0.5$, $W_{Bk} = 10000kg$), $X = [Xj|Xm] ===$ [3122211232323132133111321311122121112132311]. The scheduling Gantt chart after decoding is shown in **FIGURE 6**.

C. INITIALIZATION STRATEGY

The quality of the initial population affects the convergence speed of the algorithm and quality of the final solution. This work studies the multi-objective optimization problem; hence, a population initialization strategy based on opposite learning is proposed to improve the quality of the initial population. The opposite learning mechanism is a machine learning method proposed by Tizhoosh [37], where an algorithm can be identified faster by considering the solution of the current problem and distance of the opposite solution from the optimal solution. The concept of opposite learning is introduced below:

1) OPPOSITION NUMBER

If $X \in [a, b]$, then its inverse number $X^* = a + b - X$.

2) OPPOSITION POINT

If the individual $\overline{X} = \{X_1, X_2 \cdots, X_n\}$, its opposite individual $\overline{X}^* = \left\{X_1^*, X_2^* \cdots, X_n^*\right\}$, then $X_i^* = a + b - X_i, X_i \in [a, b]$.

In this study, all the *SN* (the number of honey sources) population is first produced, and then the same amount opposite population is produced based on initial population. The opposite population is produced according to the order of the forward population and machine number, e.g., a vector solution of seven processes for three jobs $X = [Xj|Xm] =$ [312221123232313213311|132131112212111213231], Its inverse solution $X^* = [Xj^* \mid Xm^*] =$ [13222332121213123 1133 | 121212313131111221311]. Finally, the two populations are combined to sort the Pareto non-dominant The preceding SN is selected as the initial population.

D. EMPLOYEE STAGE

In the employee bee stage, the precedence operation crossover (POX) cross method was employed, as shown **FIGURE 7,** to produce a new honey source. The parents of the cross are the current honey source while the other are randomly selected in the population. If the number selected is consistent with the current honey source, then it is crossed with the current optimal individual to produce two children.

Through the aforementioned cross-method used to produce two new honey sources, we compare the two new sources of honey to check their dominance over the original honey source, and the source of raw honey is chosen as a new solution. If two new sources of honey dominate the source of raw honey, one of the new sources is randomly selected; else, the source of raw honey is not replaced.

E. ONLOOK STAGE

To find a better honey source, this stage first creates a new honey source X^* by inserting the current honey source X (randomly producing two locations and inserting the post-position job into the front position). Then, the DTNS mechanism is introduced based on the fitness of the new honey source. The method is as follows: first, the tournament method is considered to select solution X. Then, an insert operation is performed on X to produce a new solution X^*. If the new solution X^* is dominated by X, then X is replaced; else, the four neighborhood structures defined in this study are triggered in turn until a better solution is identifies or after executing the four neighborhood structures. The steps are shown in **FIGURE 8**.

In the conventional ABC algorithm, the adaptation function is determined by the target value. This study mainly

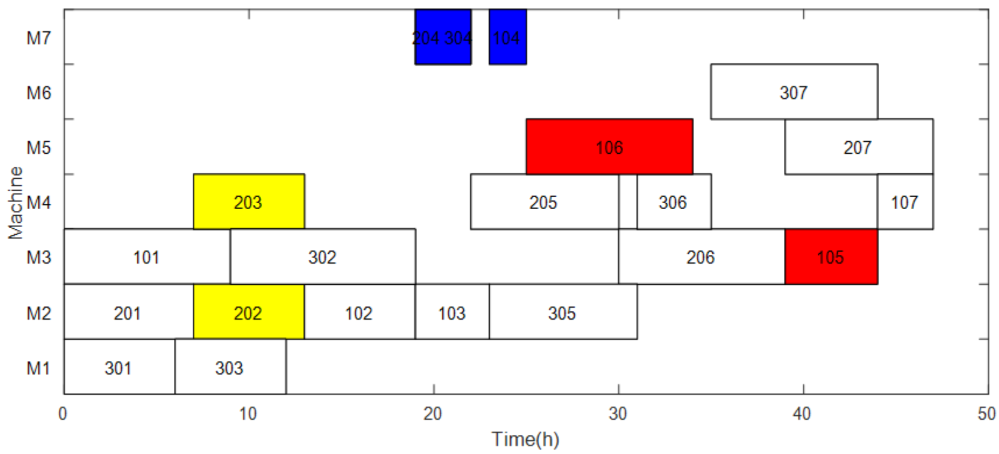


FIGURE 6. Gantt chart of example.

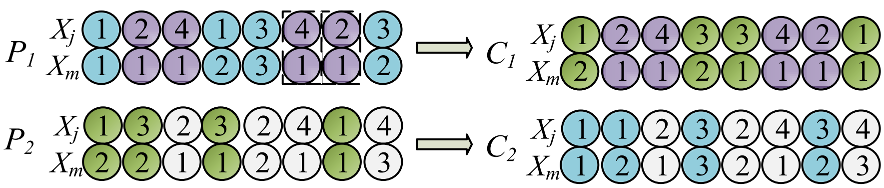


FIGURE 7. Schematic diagram of POX crossover operator.

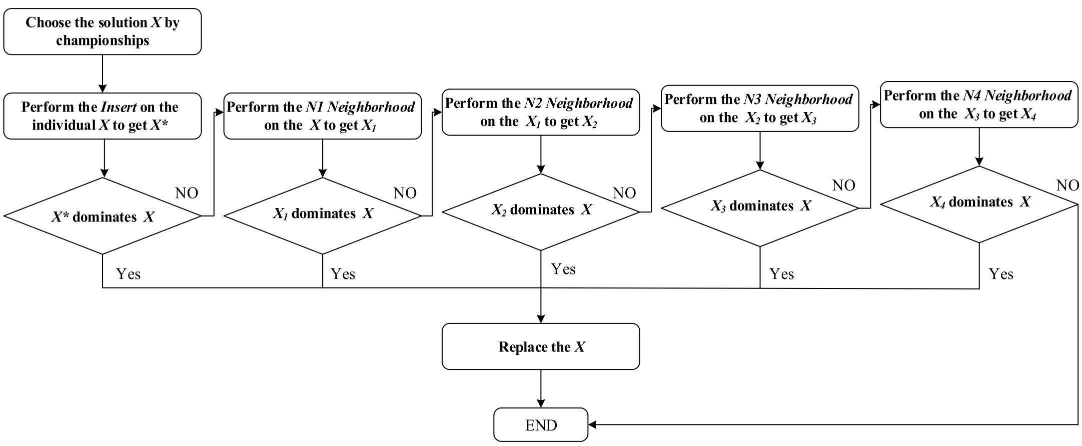


FIGURE 8. Schematic diagram of dynamic triggering neighborhood.

focuses on the multi-objective optimization problem. Therefore, a concept based on the Pareto dominance quantity is proposed to determine the fitness function, i.e., to determine the number of remaining solutions dominated by each solution, and finally, to determine the following probability of following bees according to the fitness function, as shown in the Eq. (12) and Eq. (13):

$$fit(i) = dom(i) \Big/ Foodnum \tag{12}$$

$$P(i) = fit(i) \Bigg/ \sum_{i=1}^{FoodNum} fit(i) \tag{13}$$

For the mathematical model constructed in this study, the following four neighborhood searches are defined. After using the following neighborhood structure to generate a new solution, we should check the process to avoid illegal solutions. Each neighborhood may generate more neighborhood solutions, and the smallest Pareto non-dominated level is the new solution. If there are multiple solutions, we randomly select one of them.

NS1: Batch process exchanges neighborhoods—Randomly select several batch processes and reassign process positions in ascending order of job weight;

NS2: Neighborhood based on critical path—Select the key block with more than two operations in the critical path (do not select if it does not exist), and insert the block head or block end operation into a point that is not adjacent to it in the path;

NS3: Random full neighborhood—Randomly select three procedures in a vector solution to generate all possible neighborhood solutions after the entire arrangement of the selected procedures;

NS4: Neighbor reselection of processing resources—Randomly select several parallel processes or unordered processes and redistribute the processing machines for the selected processes.

F. SCOUTER STAGE

The scouter bee is responsible for searching for new honey sources instead of the unknown honey source, as the quality of the random searched honey source may be poor. Therefore, this study performs an insertion operation and swap operation on the optimal honey source [38], [39] to replace the randomly generated honey source.

G. DEVELOPED IABC+ DTNS ALGORITHM

Based on the aforementioned theories and methods, the developed IABC+DTNS algorithm in this study is as follows:

Step1: Set the number of iterations of the IABC-DTNS algorithm *Maxcycle*, number of searches *Limit*, number of honey sources *SN*, and size of the external archive set *SN*. According to the concept of opposite learning, two populations are produced, two populations are integrated, and Pareto is sorted as non-dominant. The pre-SN is selected as the initial population; it is placed in an external archive set with a Pareto non-dominant rating of 1;

Step2: Generate a new solution based on the crossover method in Section III-D and perform Pareto non-dominated sorting on the population;

Step3: Determine the following probability of each individual according to the sorted population;

Step4: Perform neighborhood search for each individual according to Section III-E and perform Pareto non-dominated sorting on the updated population;

Step5: Update the external file set, remove the solutions that are dominated, and leave only the non-dominated solutions. If the size of the external file set is reached, the solution with the smaller crowding distance is replaced.

Step6: Reinitialize according to Section III-F to reach the limit nonupdated solution.

Step7: Determine whether the specified number of searches is reached. If the output Pareto non-dominated solution set is not reached, repeat ***Step2*** − ***Step6***.

The flowchart of the IABC+DTNS algorithm proposed in this study is shown in **FIGURE 9** below:

H. COMPLEXITY OF IABC-DTNS

Algorithm complexity is an important index to evaluate algorithm performance, which determines the efficiency of algorithm execution and affects the solving ability of computer. Suppose the problem size is D, the population size is SN, and the maximum number of iterations is G. For each iteration of the IABC-DTNS, the computational complexity is analyzed as follows.

In the initial phase, the computational complexity of generating SN individuals, generating SN opposite individuals, evaluating the initial population, and performing Non-dominant sort are $O(SN^*D)$, $O(SN^*D)$, $O(SN^*D)$ and $O(8^*SN^2)$, respectively. Then, the computational complexity of generating SN individuals is $O(3^*SN^*D+8^*SN^2)$. In employee stage, the computational complexity of generating new individuals by crossing, evaluating newly generated individuals, and selecting better individuals are $O(SN^*D)$, $O(2^*SN^*D)$, $O(SN)$, respectively. Then, the computational complexity of employee stage is $O(3^*SN^*D+SN)$. Similar to the employee stage, the computational complexity of onlooker stage is $O(10^*SN^*D+5SN)$. In scouter stage, because of its less executing times and simple operation, computational complexity of this part is negligible. Besides, the computational complexity of archive maintenance is $O(SN+2^*SN^*logSN)$.

In summary, in G iteration, the computation complexity of IABC-DTNS is shown as follows: $O(D,SN,G) = O(G)^* O(8^*SN^2+2^*SN^* \ logSN+16^*SN^*D+7^*SN) \approx G^*O(SN^2+SN^*logSN+SN^*D)$.

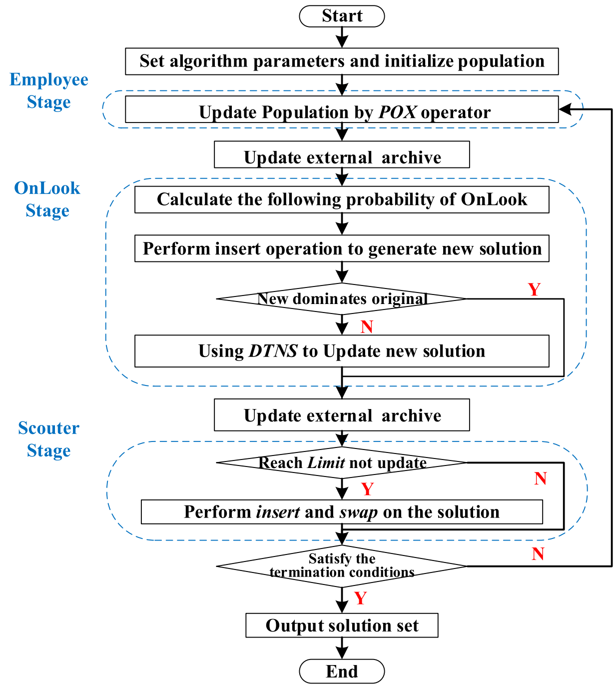


FIGURE 9. Flowchart of IABC+DTNS algorithm.

IV. EXPERIMENTS AND DISCUSSION

This study focuses on the HFS scheduling problems based on three proposed process sets. There are no benchmark instances to verify it. Therefore, a practical casting shop

TABLE 3. **Process division.**

	Operation									
	Pattern Making	Molding	Coremaking	Mold Assembling	Smelting	Shakeout	Cleaning	External inspection	Internal inspection	Refinement
M	[M_1-M_4]	[M_5-M_7]	[M_7-M_{10}]	[M_{11}, M_{12}]	[M_{24}, M_{25}]	[M_{13}, M_{14}]	[M_{14}-M_{16}]	[M_{17}-M_{20}]	[M_{19}-M_{21}]	[M_{22}-M_{25}]
			T_{pp}		T_{bp}			T_{up}		

TABLE 4. **Raw material cost and processing time of different jobs.**

Job	Raw material cost (¥)	Processing time (hour) $O_{i,1}$	$O_{i,2}$	$O_{i,3}$	$O_{i,4}$	$O_{i,5}$	$O_{i,6}$	$O_{i,7}$	$O_{i,8}$	$O_{i,9}$	$O_{i,10}$
Job1	3460	[11,3,7,18]	[6,13,12]	[11,3,7,18]	[16,16]	NaN	[8,20]	[12,16]	[11,3,7,18]	[16,12,10]	[15,17]
Job2	2560	[4,12,13,16]	[12,8,15]	[4,12,13,16]	[13,8]	NaN	[12,20]	[4,12]	[4,12,13,16]	[18,9,5]	[5,4]
Job3	3765	[8,10,5,11]	[15,14,11]	[8,10,5,11]	[12,8]	NaN	[11,15]	[17,17]	[8,10,5,11]	[18,7,18]	[5,5]
Job4	4030	[14,12,6,16]	[5,10,20]	[14,12,6,16]	[16,8]	NaN	[8,16]	[10,17]	[14,12,6,16]	[18,7,3]	[4,4]
Job5	4835	[16,6,9,6]	[4,9,13]	[16,6,9,6]	[10,11]	NaN	[17,16]	[6,19]	[16,6,9,6]	[17,13,17]	[11,14]
Job6	2860	[20,4,11,13]	[14,11,20]	[20,4,11,13]	[12,10]	NaN	[6,14]	[9,4]	[20,4,11,13]	[16,20,9]	[20,19]
Job7	1430	[11,14,6,18]	[17,16,8]	[11,14,6,18]	[13,19]	NaN	[9,5]	[17,5]	[11,14,6,18]	[7,13,20]	[7,11]
Job8	6300	[11,16,4,4]	[16,13,20]	[11,16,4,4]	[17,15]	NaN	[7,3]	[6,7]	[11,16,4,4]	[17,20,12]	[6,7]
Job9	907	[12,3,3,17]	[20,17,20]	[12,3,3,17]	[10,18]	NaN	[9,6]	[12,20]	[12,3,3,17]	[12,17,18]	[13,15]
Job10	924	[16,3,20,16]	[18,5,18]	[16,3,20,16]	[20,16]	NaN	[9,16]	[18,4]	[16,3,20,16]	[10,12,19]	[20,11]
Job11	3000	[5,11,7,9]	[9,18,10]	[5,11,7,9]	[3,18]	NaN	[10,20]	[7,14]	[5,11,7,9]	[4,4,4]	[13,5]
Job12	2700	[6,4,11,3]	[19,9,5]	[6,4,11,3]	[20,6]	NaN	[4,13]	[8,9]	[6,4,11,3]	[15,8,15]	[9,7]
Job13	1580	[13,10,18,10]	[12,11,11]	[13,10,18,10]	[19,18]	NaN	[3,11]	[14,8]	[13,10,18,10]	[10,16,6]	[5,17]
Job14	3500	[16,9,10,5]	[20,18,3]	[16,9,10,5]	[6,16]	NaN	[14,12]	[7,3]	[16,9,10,5]	[20,20,12]	[18,14]
Job15	2700	[13,6,6,4]	[6,6,17]	[13,6,6,4]	[16,3]	NaN	[4,19]	[3,3]	[13,6,6,4]	[10,11,5]	[10,19]

NaN: Batch process and time determined by formula

scheduling case is employed to evaluate and verify the effectiveness of the constructed model and proposed algorithm. The operating environment of the algorithm is a 2.7 GHz CPU, 8 GB memory, 64-bit Win7 system computer; the programming environment was MATLAB 2016.

A. CASE DESCRIPTION

For one production cycle in the foundry workshop, there are 10 different operations with 25 machines. The process division is shown in TABLE 3. T_{pp} contains molding and coremaking operation, T_{bp} contains smelting operation, T_{up} contains external inspection and internal inspection operations. The raw material cost and processing time of different jobs are shown in TABLE 4. There are different processing times of one operation processed on different machines in TABLE 5.

B. PERFORMANCE METRICS

(1) Mean ideal distance (MID) is measured by calculating the distance between a non-dominant solution and an ideal solution. This metric measures the convergence rate of the algorithm. Lower the MID value, better is the quality and performance of the algorithm.

(2) Spread of a non-dominated solution (SNS): This metric measures the diversity of the solutions. A higher value of SNS denotes a better diversity of solutions.

(3) Percentage of domination (POD): This metric measures the ability of an algorithm to dominate the solutions of other algorithms. A higher value of POD implies that the algorithm is more effective than other algorithms.

Further detailed illustrations and formulations of these metrics are found in References [40], [41].

C. EFFECTIVENESS OF THE OPPOSITE LEARNING STRATEGY

In this study, the initialization method based on opposite learning is used to improve the quality of the initial population. To verify the effectiveness of the initialization strategy, a random initialization strategy (RI) and an initialization strategy based on opposite learning (OL) are employed in

TABLE 5. DC_T **and** SC_T **of different machines (¥).**

	Machine												
	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}
DC_t	83	70	66	180	155	150	120	190	165	180	65	54	63
SC_t	20	25	30	40	35	37	28	80	70	65	10	18	12
	M_{14}	M_{15}	M_{16}	M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}	M_{25}	
DC_t	78	103	88	90	78	90	103	158	145	78	210	250	
SC_t	15	25	34	20	20	25	66	30	26	10	70	50	

TABLE 6. **The value of all metrics of IABC+DTNS based on OL and RI separately.**

Algorithm	MID	SNS	POD (%)
IABC+DTNS (OL)	**5980.398**	211597.32	**0.93**
IABC+DTNS (RI)	33749.94	211709.37	0.07

TABLE 7. **The orthogonal array and ARV values.**

Experiment number	parameter			*ARV*
	SN	*Limit*	P_m	
1	60	5	0.1	17.38
2	60	10	0.2	17.42
3	60	15	0.3	18.54
4	100	5	0.2	16.36
5	100	10	0.3	16.38
6	100	15	0.1	16.68
7	150	5	0.3	16.11
8	150	10	0.1	16.03
9	150	15	0.2	15.99

TABLE 8. **The value of all metrics obtained by compared algorithms.**

Algorithm	MID	SNS	POD
IABC+DTNS	**4824.158**	209924.6	**0.5000**
NSGA-II	11749.24	**209226.6**	0.0714
SPEA2	80010	209334.7	0.1429
MODVOA	48984.33	209302.3	0.0714
HPSO	85565.2	210683	0.0714
IABC	7928.878	210023.7	0.0714
IMOMA-II	18057	211334.2	0.0714

the initialization stage of the proposed IABC+DTNS algorithm. The initialization parameters of the two algorithms are same; the population number is $SN = 100$, $Limit = 10$, and algorithm running time is 500 s. To avoid randomness, the

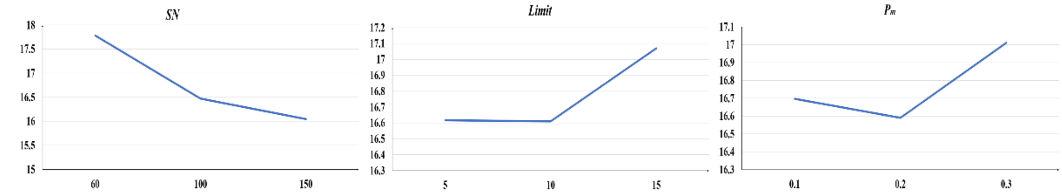


FIGURE 10. Fact level trend of IABC-DTNS.

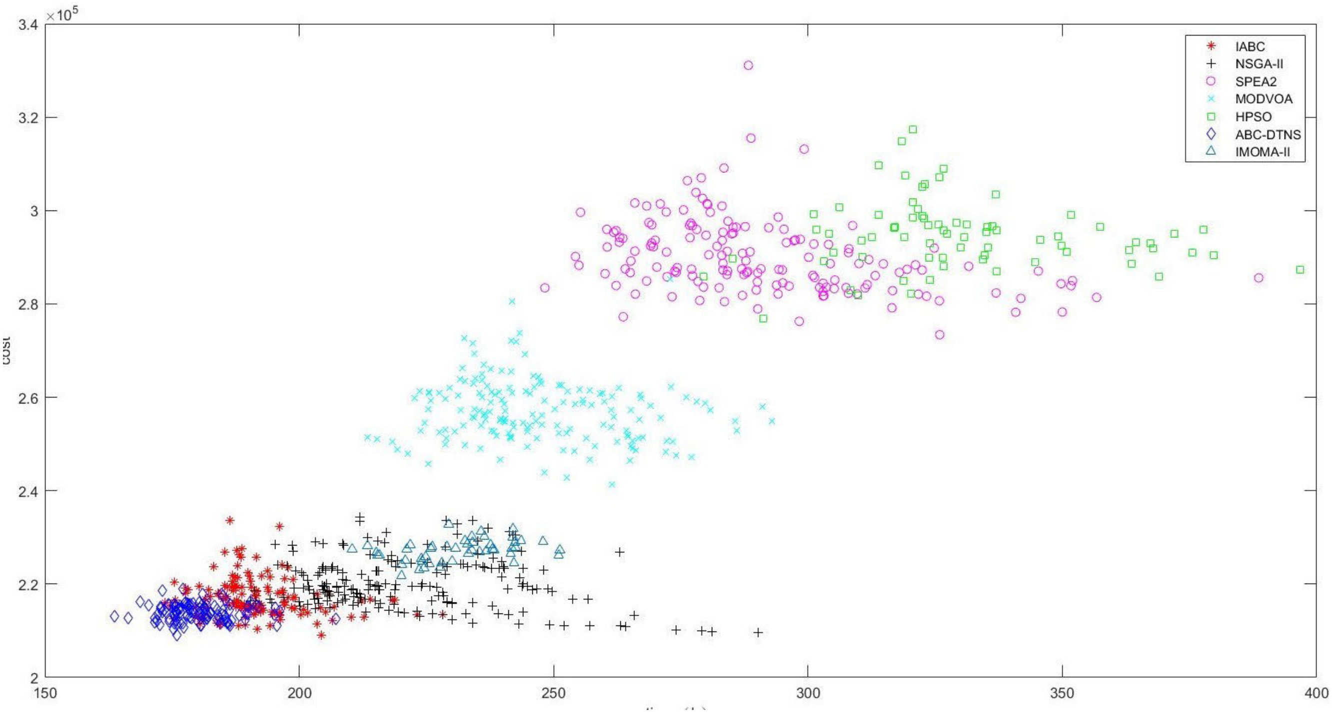


FIGURE 11. Pareto front scatter plot.

two algorithms run independently 30 times. TABLE 6 lists the results of the two different strategies.

TABLE 6 shows that the MID and POD values based on OL initialization are better than those of the RI, and the value of the initialization method based on opposite learning is much less than random initialization. Although the value of SNS based on OL initialization is less than that of the RI, the differences between them are too small to affect the diversity of feasible solutions. Therefore, the proposed initialization method based on OL is feasible and effective.

D. PARAMETER SETTINGS

To investigate the influence of different parameters on the performance of the algorithm, an orthogonal experiment with a scale of (33) was selected to optimize the algorithm parameters. To avoid the randomness of the results, each algorithm ran independently 30 times, and the average value was taken (Eq. (14)) as the evaluation index. The experimental results are shown in TABLE 7:

$$ARV = -10 \times \log(\frac{MID}{SNS}) \tag{14}$$

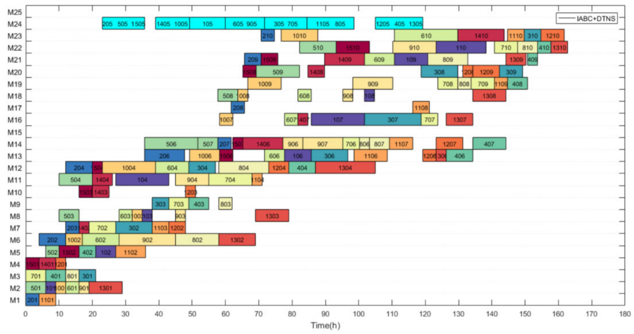
TABLE 9. Completion time and cost of compared algorithms.

Algorithm	Completion time (h)	Cost (￥)
IABC+DTNS	163.74	213068.39
NSGA-II	190.42	216450.56
SPEA2	248.28	283434.56
MODVOA	213.38	251417.53
HPSO	279.41	285906.98
IABC	173.58	216103.68
IMOMA-II	210.43	22175.52

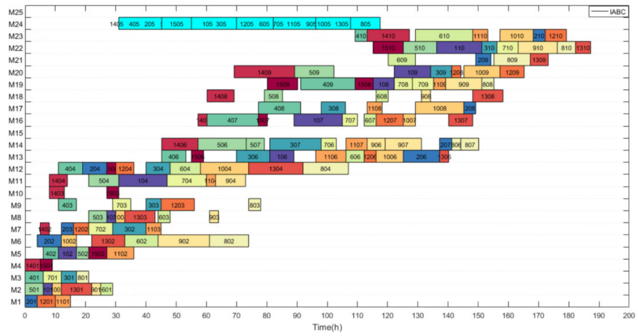
According to the orthogonal test table, the horizontal trend chart of parameters, as shown in **FIGURE 10**, is drawn. From the diagram, the performance of the algorithm is the best when $SN = 150$, $Limit = 10$, $P_m = 0.2$.

E. COMPARISONS ANALYSIS

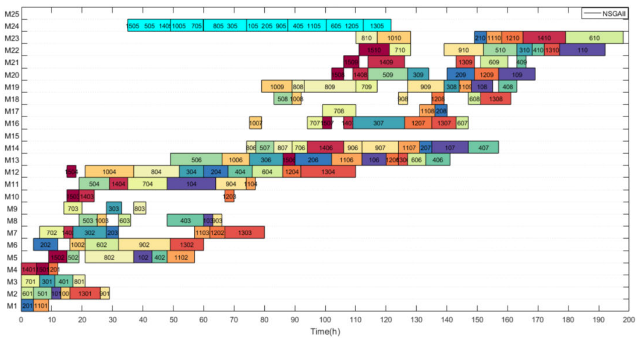
To verify the validity of the proposed algorithm (IABC-DTNS), a comparison with other algorithms



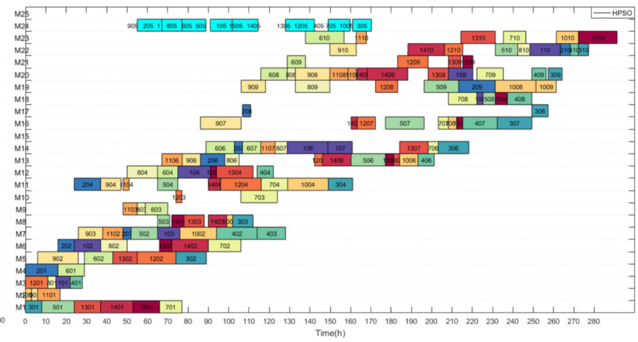
a IABC-DTNS



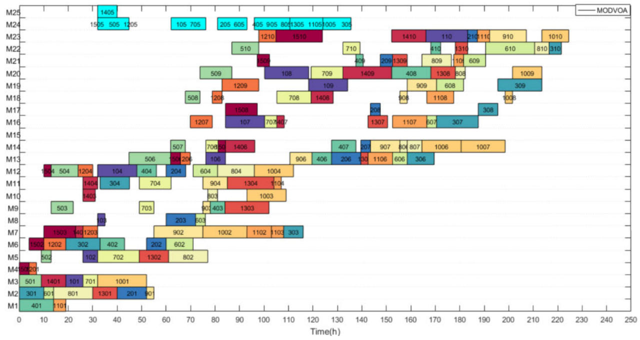
b IABC



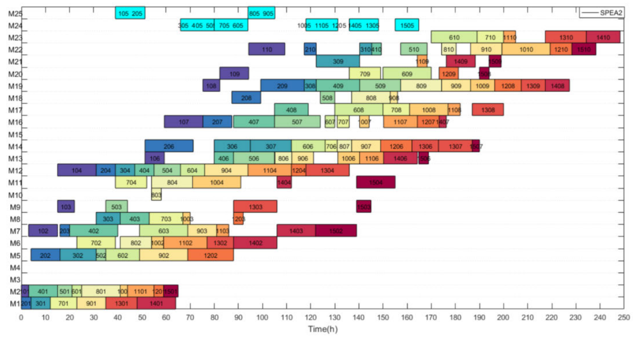
c NSGA-II



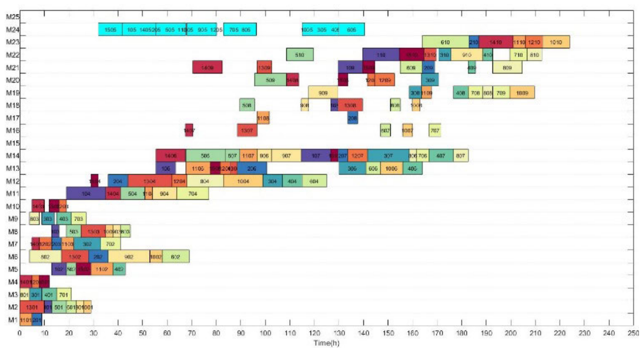
d HPSO



e MODVOA



f SPEA2



g IMOMA-II

FIGURE 12. Scheduling Gantt chart.

(NSGA-II [42], SPEA2 [43], MODVOA [44], HPSO [45], and IMOMA-II [46]) of the multi-objective solution was conducted. Simultaneously, to verify the validity of the DTNS, the IABC algorithm without DTNS is tested; its parameters are consistent with those of the IABC-DTNS. Each algorithm runs independently 30 times, and the termination time is set to 1500 s. The values of MID, SNS, and POD are shown in TABLE 8. It can be seen that the IABC-DTNS algorithm achieved the best results of MID and POD, which indicates that the convergence ability and effectiveness of IABC-DTNS are better than those of the other algorithms.

The Pareto front scatterplot of the six compared algorithms is shown in **FIGURE 11**. It can be seen that the proposed IABC-DTNS algorithm proposed is similar to the real Pareto front. The scheduling Gantt chart of the six compared algorithms' non-dominated solution sets is shown in **FIGURE 12**. Each batch in the **FIGURE** is processed on the batch process machine 24. The maximum completion time and cost are listed in TABLE 8. For the algorithms' running time, the proposed algorithm consumes a shorter running time than those of the other algorithms. Further, the cost value is smaller than those of the other algorithms. In conclusion, the IABC-DTNS has better performance than the other algorithms.

V. CONCLUSION AND FUTURE WORK

In this study, an improved ABC algorithm based on the ABC algorithm and DTNS was developed to solve the HFS scheduling problem. Three process sets have been introduced to address the multi-production procedure, such as parallel operation, batch operation, and unordered operation. The three process sets are parallel process set, batch process set, and unorder process set. For the IABC algorithm, the operation-based three-stage decoding method is designed, an effective initialization strategy based on opposite learning is proposed to improve the quality of the initial population, and four neighborhood search strategies based on DTNS are introduced to improve the local optimization ability of the algorithm during the iteration process. Comparisons with other published algorithms are conducted on a practical casting HFS scheduling case. The analysis results show that the IABC-DTNS algorithm has better exploitation capability, exploration capability, and performance reliability in solving the HFS scheduling problem.

In consideration of further exploration, the following conclusions can be drawn. (1) Further work can be conducted on constructing an effective green scheduling mathematical model of HFS, particularly in certain high-energy-consumption flow manufacturing enterprises. (2) Constructing a dynamic decision system with real-time scheduling data and developing a corresponding prototype system is extremely important for enterprise managers.

REFERENCES

[1] A. Noroozi, H. Mokhtari, and I. N. Kamal Abadi, "Research on computational intelligence algorithms with adaptive learning approach for scheduling problems with batch processing machines," *Neurocomputing*, vol. 101, pp. 190–203, Feb. 2013.

[2] H. S. Mirsanei, M. Zandieh, M. J. Moayed, and M. R. Khabbazi, "A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times," *J. Intell. Manuf.*, vol. 22, no. 6, pp. 965–978, Dec. 2011.

[3] I. Ribas, R. Leisten, and J. M. Framiñan, "Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective," *Comput. Oper. Res.*, vol. 37, no. 8, pp. 1439–1454, Aug. 2010.

[4] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *Eur. J. Oper. Res.*, vol. 205, no. 1, pp. 1–18, 2010.

[5] Y.-Y. Tan, Y.-L. Huang, and S.-X. Liu, "Two-stage mathematical programming approach for steelmaking process scheduling under variable electricity price," *J. Iron Steel Res. Int.*, vol. 20, no. 7, pp. 1–8, Jul. 2013.

[6] R. F. Teixeira, Jr., F. C. F. Fernandes, and N. A. Pereira, "Binary integer programming formulations for scheduling in market-driven foundries," *Comput. Ind. Eng.*, vol. 59, no. 3, pp. 425–435, 2010.

[7] S. Wang, M. Liu, and C. Chu, "A branch-and-bound algorithm for two-stage no-wait hybrid flow-shop scheduling," *Int. J. Prod. Res.*, vol. 53, no. 4, pp. 1143–1167, Feb. 2015.

[8] L. Hidri, S. Elkosantini, and M. M. Mabkhot, "Exact and heuristic procedures for the two-center hybrid flow shop scheduling problem with transportation times," *IEEE Access*, vol. 6, pp. 21788–21801, 2018.

[9] T. Nishi, Y. Isoya, and M. Inuiguchi, "An integrated column generation and Lagrangian relaxation for solving flowshop problems to minimize the total weighted tardiness," *Int. J. Innov. Comput., Inf. Control*, vol. 7, no. 11, pp. 6453–6471, 2011.

[10] X. Pang, H. Xue, M.-L. Tseng, M. K. Lim, and K. Liu, "Hybrid flow shop scheduling problems using improved fireworks algorithm for permutation," *Appl. Sci.*, vol. 10, no. 3, p. 1174, Feb. 2020.

[11] B.-H. Zhou, L.-M. Hu, and Z.-Y. Zhong, "A hybrid differential evolution algorithm with estimation of distribution algorithm for reentrant hybrid flow shop scheduling problem," *Neural Comput. Appl.*, vol. 30, no. 1, pp. 193–209, Jul. 2018.

[12] C. Yu, Q. Semeraro, and A. Matta, "A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility," *Comput. Oper. Res.*, vol. 100, pp. 211–229, Dec. 2018.

[13] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan," *Appl. Soft Comput.*, vol. 19, pp. 93–101, Jun. 2014.

[14] S. Liu, J. Pei, H. Cheng, X. Liu, and P. M. Pardalos, "Two-stage hybrid flow shop scheduling on parallel batching machines considering a job-dependent deteriorating effect and non-identical job sizes," *Appl. Soft Comput.*, vol. 84, Nov. 2019, Art. no. 105701.

[15] F. Choong, S. Phon-Amnuaisuk, and M. Y. Alias, "Metaheuristic methods in hybrid flow shop scheduling problem," *Expert Syst. Appl.*, vol. 38, no. 9, pp. 10787–10793, Sep. 2011.

[16] X. Li, Z. Peng, B. Du, J. Guo, W. Xu, and K. Zhuang, "Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems," *Comput. Ind. Eng.*, vol. 113, pp. 10–26, Nov. 2017.

[17] N. Karaboga, "A new design method based on artificial bee colony algorithm for digital IIR filters," *J. Franklin Inst.*, vol. 346, no. 4, pp. 328–348, May 2009.

[18] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Appl. Math. Comput.*, vol. 214, no. 1, pp. 108–132, Aug. 2009.

[19] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Oct. 2007.

[20] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 687–697, Jan. 2008.

[21] A. Yurtkuran and E. Emel, "A discrete artificial bee colony algorithm for single machine scheduling problems," *Int. J. Prod. Res.*, vol. 54, no. 22, pp. 6860–6878, Nov. 2016.

[22] R. Zhang, P.-C. Chang, S. Song, and C. Wu, "A multi-objective artificial bee colony algorithm for parallel batch-processing machine scheduling in fabric dyeing processes," *Knowl.-Based Syst.*, vol. 116, pp. 114–129, Jan. 2017.

[23] K. Z. Gao, P. N. Suganthan, T. J. Chua, C. S. Chong, T. X. Cai, and Q. K. Pan, "A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion," *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7652–7663, Nov. 2015.

[24] Y.-M. Huang and J.-C. Lin, "A new bee colony optimization algorithm with idle-time-based filtering scheme for open shop-scheduling problems," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5438–5447, May 2011.

[25] D. Gong, Y. Han, and J. Sun, "A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems," *Knowl.-Based Syst.*, vol. 148, pp. 115–130, May 2018.

[26] S.-W. Lin, K.-C. Ying, and C.-Y. Huang, "Multiprocessor task scheduling in multistage hybrid flowshops: A hybrid artificial bee colony algorithm with bi-directional planning," *Comput. Oper. Res.*, vol. 40, no. 5, pp. 1186–1195, May 2013.

[27] Z. Cui and X. Gu, "An improved discrete artificial bee colony algorithm to minimize the makespan on hybrid flow shop problems," *Neurocomputing*, vol. 148, pp. 248–259, Jan. 2015.

[28] J.-Q. Li, Q.-K. Pan, and P.-Y. Duan, "An improved artificial bee colony algorithm for solving hybrid flexible flowshop with dynamic operation skipping," *IEEE Trans. Cybern.*, vol. 46, no. 6, pp. 1311–1324, Jun. 2016.

[29] J.-Q. Li, M.-X. Song, L. Wang, P.-Y. Duan, Y.-Y. Han, H.-Y. Sang, and Q.-K. Pan, "Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2425–2439, Jun. 2020.

[30] J.-Q. Li and Q.-K. Pan, "Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm," *Inf. Sci.*, vol. 316, pp. 487–502, Sep. 2015.

[31] O. Kheirandish, R. Tavakkoli-Moghaddam, and M. Karimi-Nasab, "An artificial bee colony algorithm for a two-stage hybrid flowshop scheduling problem with multilevel product structures and requirement operations," *Int. J. Comput. Integr. Manuf.*, vol. 28, no. 5, pp. 437–450, May 2015.

[32] Q. K. Pan, L. Wang, and J. Q. Li, "A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation," *Omega*, vol. 45, pp. 42–56, Jun. 2014.

[33] K. Peng, Q.-K. Pan, L. Gao, B. Zhang, and X. Pang, "An improved artificial bee colony algorithm for real-world hybrid flowshop rescheduling in Steelmaking-refining-Continuous casting process," *Comput. Ind. Eng.*, vol. 122, pp. 235–250, Aug. 2018.

[34] J. Li, P. Duan, H. Sang, S. Wang, Z. Liu, and P. Duan, "An efficient optimization algorithm for resource-constrained steelmaking scheduling problems," *IEEE Access*, vol. 6, pp. 33883–33894, 2018.

[35] B. Zhang, Q.-K. Pan, L. Gao, X.-Y. Li, L.-L. Meng, and K.-K. Peng, "A multiobjective evolutionary algorithm based on decomposition for hybrid flowshop green scheduling problem," *Comput. Ind. Eng.*, vol. 136, pp. 325–344, Oct. 2019.

[36] D. Gerbner, B. Keszegh, C. Palmer, and D. Pálvölgyi, "Topological orderings of weighted directed acyclic graphs," *Inf. Process. Lett.*, vol. 116, no. 9, pp. 564–568, Sep. 2016.

[37] S. Mahdavi, S. Rahnamayan, and K. Deb, "Opposition based learning: A literature review," *Swarm Evol. Comput.*, vol. 39, pp. 1–23, Apr. 2018.

[38] J.-Q. Li, S.-C. Bai, P.-Y. Duan, H.-Y. Sang, Y.-Y. Han, and Z.-X. Zheng, "An improved artificial bee colony algorithm for addressing distributed flow shop with distance coefficient in a prefabricated system," *Int. J. Prod. Res.*, vol. 57, no. 22, pp. 6922–6942, Nov. 2019.

[39] B. Niu, Y. Chen, L. Tan, H. Wang, and L. Li, "Discrete artificial bee colony algorithm for low-carbon traveling salesman problem," *J. Comput. Theor. Nanosci.*, vol. 9, no. 10, pp. 1766–1771, Oct. 2012.

[40] A. M. Fathollahi-Fard, M. Hajiaghaei-Keshteli, and R. Tavakkoli-Moghaddam, "The social engineering optimizer (SEO)," *Eng. Appl. Artif. Intell.*, vol. 72, pp. 267–293, Jun. 2018.

[41] R. B. Govindan, A. N. Massaro, T. Al-Shargabi, N. N. Andescavage, T. Chang, P. Glass, and A. J. du Plessis, "Detrended fluctuation analysis of non-stationary cardiac beat-to-beat interval of sick infants," *EPL (Europhys. Lett.)*, vol. 108, no. 4, p. 40005, Nov. 2014.

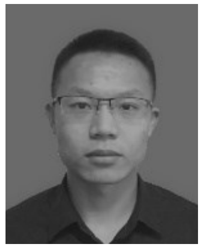
[42] H. Han, R. Yu, B. Li, and Y. Zhang, "Multi-objective optimization of corrugated tube inserted with multi-channel twisted tape using RSM and NSGA-II," *Appl. Thermal Eng.*, vol. 159, Aug. 2019, Art. no. 113731.

[43] H. Amin-Tahmasbi and R. Tavakkoli-Moghaddam, "Solving a bi-objective flowshop scheduling problem by a Multi-objective Immune System and comparing with SPEA2+ and SPGA," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 772–779, 2011.

[44] C. Lu, X. Li, L. Gao, W. Liao, and J. Yi, "An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times," *Comput. Ind. Eng.*, vol. 104, pp. 156–174, Feb. 2017.

[45] K. Mason, J. Duggan, and E. Howley, "Multi-objective dynamic economic emission dispatch using particle swarm optimisation variants," *Neurocomputing*, vol. 270, pp. 188–197, Dec. 2017.

[46] J. Sun, Z. Miao, and D. Gong, "Interval multiobjective optimization with memetic algorithms," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3444–3457, Aug. 2020.



XIXING LI received the M.S. and Ph.D. degrees in mechanical engineering from the Wuhan University of Technology, Wuhan, China, in 2014 and 2017, respectively. He is currently a Lecturer with the School of Mechanical Engineering, Hubei University of Technology, Wuhan. He has published about ten journal articles. His current research interests include production planning and scheduling, manufacturing informatization, and optimization modeling.

HONGTAO TANG received the M.S. degree in mechanical engineering from the Wuhan University of Technology, Wuhan, China, in 2008, and the Ph.D. degree in mechanical engineering from the Huazhong University of Science and Technology, Wuhan, in 2014. He is currently an Associate Professor of industrial engineering with the Wuhan University of Technology. He has published more than 50 academic articles. His current research interests include intelligent manufacturing systems and complex system optimization.

ZHIPENG YANG received the B.Sc. degree in mechanical engineering from the Wuhan University of Technology, in 2018, where he is currently pursuing the M.S. degree. He has published two articles in related journals. His current research interest includes intelligent manufacturing systems.

RUI WU received the B.Sc. degree and the Ph.D. degree in mechanical engineering from the Wuhan University of Technology, Wuhan, China, in 2012 and 2019, respectively. He is currently a Lecturer with the School of Mechanical Engineering, Hubei University of Technology, Wuhan. His current research interests include manufacturing scheduling and intelligent optimization algorithms.

YABO LUO received the M.S. degree in petroleum engineering from Yangtze University, Jingzhou, China, in 1994, and the Ph.D. degree in mechanical engineering from the Wuhan University of Technology, Wuhan, China, in 2001. He is currently a Professor of industrial engineering with the Wuhan University of Technology. He has published three academic works or college textbooks, and more than 70 academic articles. His current research interests include complex system optimization and bionic algorithm developing.