Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 6012737, 12 pages
https://doi.org/10.1155/2020/6012737

Research Article

A Mathematical Model and Self-Adaptive NSGA-II for a Multiobjective IPPS Problem Subject to Delivery Time

Li Ba, Mingshun Yang, Xinqin Gao, Yong Liu, Zhoupeng Han, Erbao Xu, and Yan Li

Key Laboratory of NC Machine Tools and Integrated Manufacturing Equipment of the Education Ministry, Xi'an University of Technology, Xi'an 70048, China

Correspondence should be addressed to Li Ba; xautbali@163.com

Received 25 May 2020; Accepted 1 August 2020; Published 24 August 2020

Academic Editor: Miguel A. Salido

Process planning and scheduling are two important components of manufacturing systems. This paper deals with a multiobjective just-in-time integrated process planning and scheduling (MOJIT-IPPS) problem. Delivery time and machine workload are considered to make IPPS problem more suitable for manufacturing environments. The earliness/tardiness penalty, maximum machine workload, and total machine workload are objectives that are minimized. The decoding method is a crucial part that significantly influences the scheduling results. We develop a self-adaptive decoding method to obtain better results. A non-dominated sorting genetic algorithm with self-adaptive decoding (SD-NSGA-II) is proposed for solving MOJIT-IPPS. Finally, the model and algorithm are proven through an example. Furthermore, different scale examples are tested to prove the good performance of the proposed method.

1. Introduction

Process planning and scheduling are two crucial parts in manufacturing industries. They are commonly performed independently in serial mode [1, 2]. In the product design stage, process planning is used to determine the raw materials, machine tools, and process routes. In the scheduling stage, machines are scheduled according to the results from process planning [3]. The independent and serial modes of the two subsystems may lead to unrealistic process routes, uneven resource utilization, and bottlenecks in scheduling [4].

In recent years, IPPS problems have gradually become a research focus around the world. The integration of process planning and scheduling is an effective way to eliminate resource conflicts, decrease the flow time of workpieces, improve machine utilization, and reduce the production cycle [3].

Currently, the basic IPPS problem, which considers makespan as an objective, has been a research focus in production scheduling. With the continuous attention of the whole society on energy conservation and waste elimination, just-in-time production has become the general consensus of modern manufacturing enterprises [5]. Delivery time is a key factor that has been considered in real production. Moreover, there are other important consideration factors in addition to the time factor. Therefore, it is difficult to meet actual production demand if optimization is based only on a single objective.

In conclusion, to make IPPS problems more in line with real manufacturing situations, this paper focuses on a MOJIT-IPPS problem. The earliness/tardiness penalty, maximum machine workload, and total machine workload are regarded as objectives. A mathematical model of the MOJIT-IPPS problem is established. The decoding method is an important part that influences the JIT scheduling solution. Given the complexity of the problem, a SD-NSGA-II is proposed for the problem. Finally, the model and algorithm are proven through several examples.

This paper initially provides a literature review related to IPPS problems in Section 2. A mathematical model of MOJIT-IPPS is established in Section 3. SD-NSGA-II is designed in Section 4. Computational results are analysed

in Section 5. The paper ends with some conclusions in Section 6.

2. Literature Review

Researchers began studying the IPPS problem in the 1980s [6]. The basic IPPS problem can be defined as follows [7]: "given a set of n parts that are to be processed on m machines with operations including alternative manufacturing resources, select suitable manufacturing resources, and sequences of operations to determine a schedule in which the precedence constraints among operations can be satisfied and the corresponding objectives can be achieved."

Compared to the job shop scheduling problem (JSP) and flow shop scheduling problem (FSP), the IPPS problem has a larger solution space and is more complex because of flexibilities in process routes and machines. This problem has been identified as NP-hard [8]. Currently, an increasing number of studies have focused on the IPPS problem. Algorithm improvement is a research direction of IPPS. Many researchers have proposed advanced algorithms for improving the quality of solutions. In addition, more factors in real production have been considered in the basic IPPS problem to make it more realistic. Therefore, we summarized these studies in two aspects: algorithm improvement and problem improvement.

2.1. Improvement of Algorithms. Li et al. [1] proposed a novel hybrid algorithm based on particle swarm optimization (PSO) and a genetic algorithm (GA) for an IPPS problem with an uncertain processing time. Several different sizes of examples were involved to demonstrate the algorithm. Petrovic et al. [3] used an AND/OR graph to express flexible process plans. An algorithm based on a PSO and chaos theory was proposed to solve an IPPS problem. The advantage of the algorithm was proven by comparing it with a GA and a simulated annealing algorithm (SAA) through a set of examples. Seker et al. [8] proposed a hybrid heuristic algorithm based on a GA and artificial neural network (ANN) to solve an IPPS problem. A clustered chromosome structure was designed to improve the algorithm performance. Yin et al. [9] proposed an improved GA for solving a dynamic IPPS problem with consideration of machine breakdown and new job arrival. Zhang et al. [10] proposed an object-coding GA to solve an IPPS problem. An effective object-coding representation was designed, and experiments showed good performance of the algorithm. Jain et al. [11] proposed a simulation-based GA to solve an IPPS problem. A regeneration scheme was embedded in the algorithm to prevent premature convergence. Phanden et al. [12] established an IPPS model composed of a process route selection module, scheduling module, analysis module, and process route modification module. A GA was designed to solve the model. Li [13] proposed an active learning GA to solve an IPPS problem. The algorithm was proven to be a promising and effective method through experimental results. Lian et al. [14] proposed an imperialist competitive algorithm (ICA) to solve an IPPS problem. The performance of the ICA was evaluated on four sets of experiments. Özgüven et al. [15] proposed a mixed-integer linear programming model for an IPPS problem. Several examples were introduced and the results showed the superiority of the model. Zhou et al. [16] proposed a game-theoretic approach for generating optimal process plans for multiple jobs in networked manufacturing.

2.2. Improvement of Problems. Zhang et al. [2] studied an extended IPPS problem in a cloud manufacturing context, and they proposed three frameworks to cope with IPPS. Zhang [17] proposed an IPPS problem in a remanufacturing context and designed a simulation-based GA for the problem. Shokouhi [18] proposed and studied an IPPS problem considering precedence constraints, and a GA was designed for the problem. Xia et al. [19] proposed a dynamic IPPS problem with consideration of machine breakdown and new job arrival, and a model of the problem was established. A hybrid GA with variable neighbourhood search (VNS) was designed to solve the problem. Multiple objectives are considered in IPPS problem and studied by Manupati et al. [20, 21]. They proposed some novel algorithms for solving the problem.

Currently, energy conservation and environmental protection have reached a consensus in many countries. Energy consumption has become a critical aspect in industries. Zhang et al. [22] established an IPPS model with total energy consumption as the objective. A GA-based approach was proposed to solve the problem. Dai et al. [23] created a mathematical model of an IPPS problem. Energy consumption and makespan were two objectives to be minimized. A modified GA was adopted to explore the optimal solution between the two objectives.

In real production, all processes may not be able to be processed in the same workshop due to different process types. Transportation is another factor that must be considered in some real production environments. For this aspect, Manupati et al. [4, 24] considered transportation in IPPS and proposed algorithms to solve extended problems.

Presently, on the one hand, the JIT production mode has become the general consensus of manufacturing enterprises. Each order should be completed on time. Many workshop scheduling problems consider delivery time, such as the JSP [25, 26], FSP [27, 28], parallel-machine scheduling problem (PMSP) [5, 29], and single-machine scheduling problem [30, 31]. On the other hand, machine workload is another key indicator in workshop scheduling.

Based on the above literature, this paper considers delivery time in the IPPS problem to make it more in line with actual production. A mathematical model of the MOJIT-IPPS problem is established. The earliness/tardiness penalty, maximum machine workload, and total machine workload are objectives that are minimized.

The decoding method is a critical module that influences the JIT scheduling solution. Due to the complexity of MOJIT-IPPS, this paper proposes a self-adaptive decoding method for MOJIT-IPPS. An NSGA-II with the self-adaptive decoding method is designed. The effectiveness of the proposed approach is demonstrated through a set of

different scale examples by comparison with another decoding method.

3. Mathematical Model of MOJIT-IPPS

There are different jobs that must be processed in a workshop. All jobs should be finished according to the delivery time. Each job can be processed through more than one process route, and each process can be processed by more than one machine. Under these conditions, the earliness/tardiness penalty, maximum machine workload, and total machine workload are regarded as the objectives. An optimal solution is obtained by choosing process routes, choosing machines, and sorting process sequences for each job.

Based on the above description, a MOJIT-IPPS model is proposed. The following parameters are defined to establish the model for the problem:

(i) n: number of jobs.

(ii) m: number of machines.

(iii) p: number of process routes.

(iv) R_k: process route k, $k \in [1, p]$.

(v) ps_k: number of processes of R_k.

(vi) TMS_j: available starting time of machine j, $j \in [1, m]$.

(vii) TSJ_{ikvj}: R_k is chosen for job i, starting time on machine j of the vth process, $i \in [1, n]$, $v \in [1, ps_k]$.

(viii) TMJ_{ikvj}: R_k is chosen for job i, process time on machine j of the vth process.

(ix) TFFJ_i: the final completion time of job i.

(x) TJD_i: delivery time of job i.

(xi) WLM_j: workload of machine j.

(xii) RJP_{ik}: Boolean variable representing the relationship between a job and process route. If R_k is chosen for job i, the variable is equal to 1; otherwise, it is 0.

(xiii) RPM_{kvj}: Boolean variable representing the relationship between a process route and machine. If the vth process of R_k is processed on machine j, the variable is equal to 1; otherwise, it is 0.

Based on the above parameters, the earliness/tardiness penalty, maximum machine workload, and total machine workload are considered the objectives. The objective functions are established as follows.

The earliness/tardiness penalty is minimized:

$$\mathrm{Min}\left\{\sum_{i=1}^{n}\left|\mathrm{TJD}_i - \mathrm{TFFJ}_i\right|\right\}. \tag{1}$$

The maximum machine workload is minimized:

$$\mathrm{Min}\left\{\mathrm{Max}\left\{\mathrm{WLM}_j\right\}\right\}. \tag{2}$$

The total machine workload is minimized:

$$\mathrm{Min}\left\{\sum_{j=1}^{m}\mathrm{WLM}_j\right\}, \tag{3}$$

where

$$\mathrm{WLM}_j = \sum_{i=1}^{n}\sum_{k=1}^{p}\sum_{v=1}^{ps_k}\mathrm{RJP}_{ik}\cdot\mathrm{RPM}_{kvj}\cdot\mathrm{TMJ}_{ikvj}, \quad i \in [1,n], j \in [1,m], k \in [1,p] \text{ and } v \in [1, ps_k]. \tag{4}$$

The following constraints are considered in the model:

3.1. One Machine Can Only Perform One Process on a Job at a Time. If time starts at t,

$$\mathrm{RJP}_{ik}\cdot\mathrm{RPM}_{kv'j} = 1. \tag{5}$$

Then, during time slot $[t, t + \mathrm{TMJ}_{ikv'j}]$,

$$\sum_{v=1}^{ps_k}\mathrm{RJP}_{ik}\cdot\mathrm{RPM}_{kvj} = 1, \tag{6}$$

where $v' \in [1, ps_k]$.

3.2. One Process Cannot Be Performed by More than One Machine. If time starts at t,

$$\mathrm{RJP}_{ik}\cdot\mathrm{RPM}_{kvj'} = 1. \tag{7}$$

Then, during time slot $[t, t + \mathrm{TMJ}_{ikvj'}]$,

$$\sum_{j=1}^{m}\mathrm{RJP}_{ik}\cdot\mathrm{RPM}_{kvj} = 1, \tag{8}$$

where $j' \in [1, m]$.

3.3. The Job Is Processed Strictly according to the Process Sequence, and the Starting Time of Each Process Cannot Be Earlier than the Available Starting Time of the Machine That Is Chosen to Complete the Process. Assuming $\mathrm{RJP}_{ik} = 1$ and $\mathrm{RJP}_{ik}\cdot\mathrm{RPM}_{kv'j} = 1$, $\mathrm{RJP}_{ik}\cdot\mathrm{RPM}_{k(v'-1)j'} = 1$. Currently, the v'th process of job i will be performed; then,

$$\mathrm{TSJ}_{ikv'j} = \begin{cases} \mathrm{TMS}_j, & \mathrm{TMS}_j \geq \mathrm{TFJ}_{ik(v'-1)j}, \\ \mathrm{TFJ}_{ik(v'-1)j'}, & \mathrm{TMS}_j < \mathrm{TFJ}_{ik(v'-1)j}, \end{cases} \tag{9}$$

where $v' \in [2, ps_k]$.

According to constraint (3), the related parameters are updated by the following equation.

The completion time of the v'th process of job i is calculated as follows:

$$\mathrm{TFJ}_{ikv'j} = \mathrm{TSJ}_{ikv'j} + \mathrm{TMJ}_{ikv'j}. \quad (10)$$

The available starting time of machine j is updated as follows:

$$\mathrm{TMS}_j = \mathrm{TFJ}_{ikv'j}. \quad (11)$$

If $v' = ps_k$, TFFJ_i is updated as follows:

$$\mathrm{TFFJ}_i = \mathrm{TFJ}_{ikv'j}. \quad (12)$$

Equations (1)–(3) show the objectives that will be minimized. The workload of each machine is calculated by equation (4). The relationships between processes and machines are constrained through equations (5)–(8) to ensure the rationality of the manufacturing process. The starting time of each process is calculated by equation (9); the completion time of each process is calculated by equation (10); the available starting time of each machine is calculated by equation (11); and the final completion time of each job is calculated by equation (12).

4. SD-NSGA-II for MOJIT-IPPS

Based on the description and modelling of the above problem, the MOJIT-IPPS problem is NP-hard. Deb et al. [32] proposed NSGA-II in 2002. To date, the algorithm has been applied to solve many multiobjective optimization problems. Considering the complexity and discreteness of the MOJIT-IPPS problem, this study proposes a SD-NSGA-II to solve the problem.

4.1. Encoding. A four-layer encoding structure is proposed to construct solutions. The structure consists of four layers: the workpiece, machine, process time, and process route. An example of the encoding structure of MOJIT-IPPS is provided in Table 1.

In Table 1, for example, the codes in the first column mean that R_3 is adopted for J_2. Because the code "2" first appears in the workpiece layer, this code represents the first process of J_2. M_4 is chosen to complete this process. The process time is 0.8, and so on.

4.2. Decoding. Decoding is a key step that influences solutions. The semiactive decoding method is commonly used for many basic workshop scheduling problems where the makespan is the objective to be minimized, such as the basic JSP, FSP, and IPPS. However, this method is not suitable for MOJIT-IPPS because delivery time is taken into account. To improve the decoding performance, this paper proposes a self-adaptive decoding method for MOJIT-IPPS. We divide the method into three parts: active decoding, self-adaptive operation for solutions, and reverse active decoding.

4.2.1. Part 1: Active Decoding. Compared with semiactive decoding, active decoding is more effective in compressing the completion time of each workpiece. A sample coding is shown in Table 2. Two workpieces will be processed on two machines. Two processes will be performed for each workpiece. Sample coding is decoded from the first column to the end. A comparison between these two decoding methods is shown in Figure 1 (DT: delivery time).

If using semiactive decoding, processes will be performed according to start time of machines only. For example, the second process of J_2 will be carried out by M_1. When this process is finished on M_1, the start time of M_1 is updated to seven and the first process of J_1 will be started at this time. However, M_1 is idle from time zero to four. The first process of J_1 can be performed during this time interval.

Therefore, for reducing finished time of each workpiece, we adopt active decoding instead of semiactive decoding. Before introducing this decoding method, some parameters are defined as follows:

(i) SC: a sample coding that will be decoded

(ii) L_{sc}: the number of columns of SC

(iii) Job_k: the workpiece in the kth column of SC, $k = 1, 2, \ldots, L_{sc}$

(iv) Mach_k: the machine in the kth column of SC

(v) Ts (Mach_k): the earliest start time of Mach_k;

(vi) Nop (Job_k): the number of processes of Job_k

Supposing that the kth column represents the vth process of Job_k, the steps of active decoding are as follows:

(i) Step 1: set counters $k = 1$ and $f = 0$.

(ii) Step 2: check the idle time and work time of Mach_k from time zero to Ts (Mach_k); if exiting an idle time segment, $f = f + 1$.

(iii) Step 3: if $f > 0$, go to Step 4; otherwise, go to Step 6.

(iv) Step 4: set a counter $g = 0$. If the vth process of Job_k can be carried out in the gth idle time segment, then deal with this process in the idle time segment, and go to Step 7; otherwise, $g = g + 1$, and go to Step 5.

(v) Step 5: if $g > f$, go to Step 6; otherwise, go to Step 4.

(vi) Step 6: perform the vth process of Job_k at time Ts (Mach_k).

(vii) Step 7: $k = k + 1$.

(viii) Step 8: if $k > L_{\mathrm{sc}}$, active decoding for SC is completed; otherwise, go to Step 2.

4.2.2. Part 2: Self-Adaptive Operation for Solutions. The solution of all processes of workpieces must be adjusted to correct the location of each column. If not, the completion time will be extended, and workpieces may be finished late in reverse active decoding. A comparison is shown in Figure 2.

To present the flow of self-adaptive operation, "TScn" is defined as a set, $\mathrm{TScn} = \{\mathrm{TScn}_1, \mathrm{TScn}_2, \ldots, \mathrm{TScn}_{L_{\mathrm{sc}}}\}$. For example, "$\mathrm{TScn}_1$" denotes the start time of the process and is located in the first column of SC.

The steps of self-adaptive operation are as follows:

Table 1: Sample coding of MOJIT-IPPS

J_i	Code of the workpiece							
	2	1	2	1	2	2	1	2
M_j	Code of the machine							
	4	2	1	4	3	1	1	2
TMJ_{ikvj}	Process time of the workpiece							
	0.8	0.9	0.7	1.3	1	0.6	1.1	0.7
R_k	Code of the process route							
	3	2	1	2	3	1	2	3

Table 2: Sample coding.

J_i	2	2	1	1
M_i	2	1	1	2
TMJ_{ikvj}	4	3	2	2

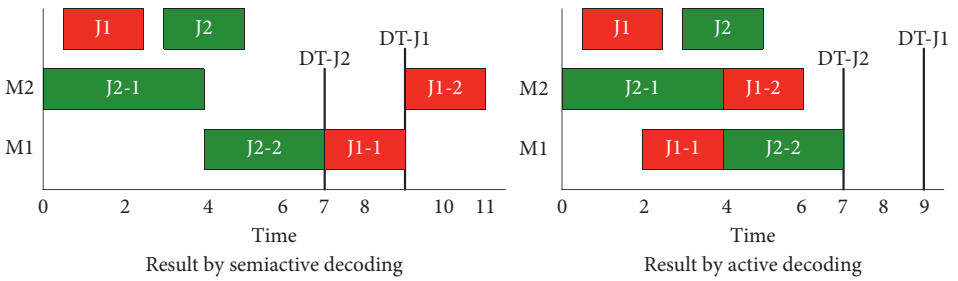

Figure 1: Comparison between semiactive decoding and active decoding. (a) Result by semiactive decoding. (b) Result by active decoding.

(i) Step 1: make a copy of SC, named SCTem, and clear all columns of SCTem

(ii) Step 2: set a counter $u = L_{sc}$

(iii) Step 3: choose the maximum value in TScn, put the column that corresponds to this value into the uth column of SCTem, and clear this value in TScn

(iv) Step 4: $u = u-1$; if u is equal to zero, go to Step 5; otherwise, go to Step 3

(v) Step 5: replace SC with SCTem, and the self-adaptive operation for SC is completed

4.2.3. Part 3: Reverse Active Decoding. In this part, solutions are decoded in reverse. The main decoding steps are same as those in part 1, but the start time of each workpiece is not equal to zero and each solution is decoded in reverse.

The start time of workpiece h for reverse active decoding is denoted by "$TSRev_h$." The final completion time of workpiece h in part 1 is denoted by "$TFFJ_h$." The delivery time of workpiece h is denoted by "TJD_h." $TSRev_h$ is calculated by the following equation:

$$TSRev_h = \text{Max}\{TFFJ_h, TJD_h\}, \quad h \in [1, n]. \tag{13}$$

An example of reverse active decoding is shown in Figure 3.

4.3. Selection Operation. First, the Pareto dominance relationship is taken into consideration, each solution is ranked, and the rank of each solution is obtained. Second, the crowding distance of each solution is calculated. Finally, a crowded comparison operator is proposed. A tournament selection method is adopted to perform the selection operation.

4.3.1. Ranking Operation. Before performing the ranking operation, the Pareto dominance relationship is defined as follows [33].

Pareto dominance relationship: for a multiobjective minimization problem with N optimization objectives, assume a solution set P is composed of M individuals, $P = \{x_1, x_2, \ldots, x_M\}$. For any individual x_i, the target value of the jth objective is $f_j(x_i)$. Then, if solution x_u dominates solution x_v, the following two conditions must be met:

(1) For any target values f_k, $f_k(x_u) \le f_k(x_v)$

(2) $\exists k'$, $f_{k'}(x_u) < f_{k'}(x_v)$

where $i, u, v \in [1, M]$, $j, k, k\prime \in [1, N]$.

The definition of the Pareto dominance relationship shows that if solution x_u dominates solution x_v, solution x_u is better than solution x_v in the multiobjective minimization problem.

Assuming that the population is $P = \{x_1, x_2, \ldots, x_M\}$, the rank set is $\text{Rank} = \{\text{rank}_1, \text{rank}_2, \ldots, \text{rank}_M\}$, and the number of individuals that dominate x_i ($i \in [1, M]$) is $DN = \{dn_1, dn_2, \ldots, dn_M\}$. Some variables are initialized as follows: $\text{rank}_h = -1$, $dn_h = 0$, and $h \in [1, M]$. The ranking operation is shown in Figure 4.

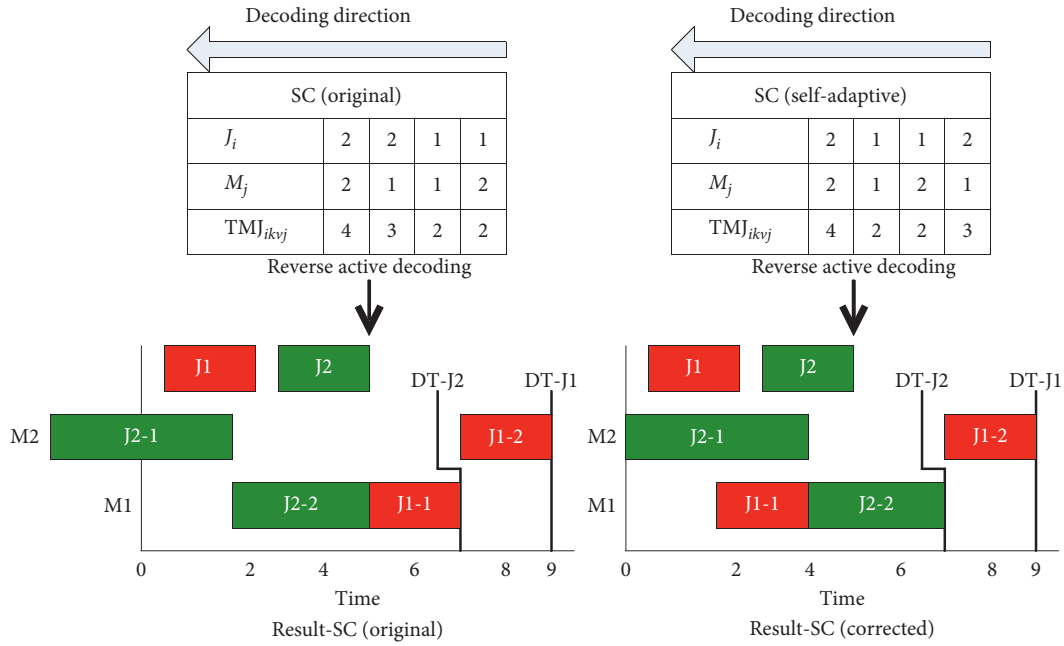

Figure 2: Comparison before and after self-adaptive operation.

Figure 4 shows that if solution x_u dominates solution x_v, the rank of x_u is smaller than the rank of x_v in the multiobjective minimization problem.

4.3.2. Crowding Distance. In addition to rank, crowding distance is another important indicator. If the ranks of two individuals are equal, to maintain the diversity of the population, it is commonly believed that an individual with a larger crowding distance between the two individuals is a better individual [32].

Assuming a multiobjective minimization problem, the number of optimization objectives is N, the population P is composed of M individuals, $P = \{x_1, x_2, \ldots, x_M\}$, and the crowding distance set is $\mathrm{CD} = \{\mathrm{cd}_1, \mathrm{cd}_2, \ldots, \mathrm{cd}_M\}$. The crowding distance of the jth target of x_i is cdf_{ij}. The individual that is ranked one position before x_i and the individual that is ranked one position behind x_i are named x_{front} and x_{behind}, respectively. The individual with the maximum value of the jth target and the individual with the minimum value of the jth target are named $x_{\max}$ and $x_{\min}$, respectively. Then, the crowding distance cd_i of individual x_i is calculated as the following two equations:

$$\mathrm{cdf}_{ij} = \frac{f_j(x_{\mathrm{behind}}) - f_j(x_{\mathrm{front}})}{f_j(x_{\max}) - f_j(x_{\min})}, \tag{14}$$

$$\mathrm{cd}_i = \sum_{j=1}^{N} \mathrm{cdf}_{ij}. \tag{15}$$

4.3.3. Tournament Selection. Before tournament selection, a crowded comparison operator is defined as follows [32].

Crowded comparison operator: for a multiobjective minimization problem with N optimization targets, assuming a solution set P is composed of M individuals, $P = \{x_1, x_2, \ldots, x_M\}$, the rank of solution x_u is rank_u, and the rank of solution x_v is rank_v. The crowding distances of solutions x_u and x_v are cd_u and cd_v, respectively. The notation of the crowded comparison operator is defined as " $<$ "; then, $x_u < x_v$ must meet the following condition (1) or (2):

(1) $\mathrm{rank}_u < \mathrm{rank}_v$

(2) $\mathrm{rank}_u = \mathrm{rank}_v$ and $\mathrm{cd}_u > \mathrm{cd}_v$

where $u, v \in [1, M]$.

The crowded comparison operator shows that if $x_u < x_v$, solution x_u is better than x_v.

Based on the crowded comparison operator, tournament selection will be conducted for selecting individuals which will be crossed.

4.4. Crossover Operation. This paper adopts a precedence operation crossover (POX) operator [34] to crossover individuals. Assuming workpiece 2 will undergo crossover, an example of POX is shown in Figure 5.

Because different process routes may be chosen for the same workpiece in different individuals, the same workpiece may have different numbers of operations in different individuals. Thus, the code length of generated individuals may be changed after crossover, as shown in Figure 5.

4.5. Mutation Operation. Two kinds of mutation operators are adopted in SD-NSGA-II for the MOJIT-IPPS problem. First, swapping is performed on codes of two columns according to a swap probability. Two locations are generated randomly. The two columns of codes in the locations are exchanged. Second, single-point mutation is performed on the machine layer according to a single-point mutation probability.

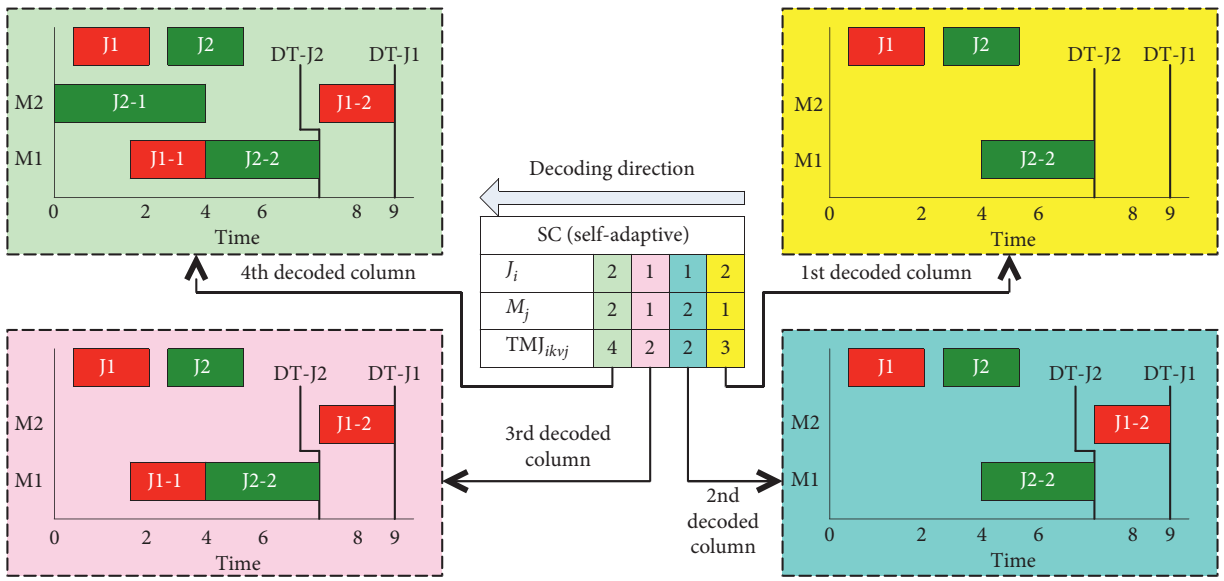

Figure 3: An example of reverse active decoding.

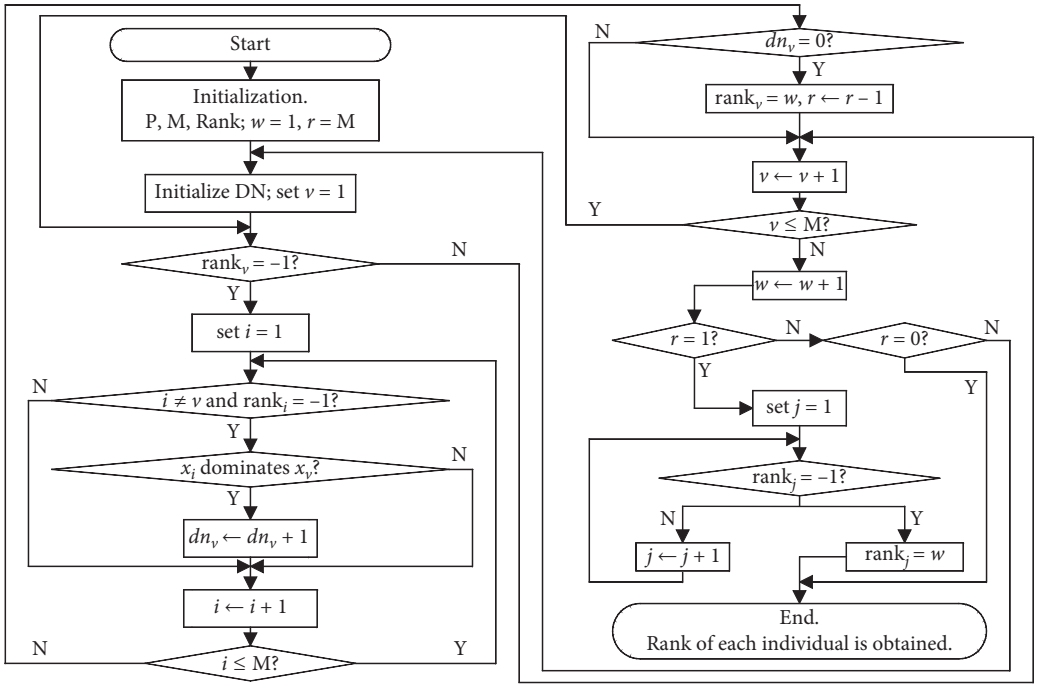

Figure 4: Flowchart of ranking operation.

4.6. Pareto Set. Assuming that the population size is equal to M, the number of iterations is equal to D. The following steps are adopted to construct a Pareto set:

(i) Step 1: initialize the Pareto set, PA = $\{\varphi\}$, and iteration counter $e = 0$.

(ii) Step 2: the individuals of the first generation P are ranked, and the crowding distance of each individual is calculated.

(iii) Step 3: the selection, crossover, and mutation operations are performed on population P, and then population P' of the next generation is obtained.

(iv) Step 4: the temporary population Q is established, $Q = P \cup P'$.

(v) Step 5: the individuals of population Q are ranked, and the crowding distance of each individual is calculated.

(vi) Step 6: each individual whose rank is equal to 1 in population Q is compared with all individuals that belong to PA. If an individual is different from any individual in PA, it is added to PA.

(vii) Step 7: $e++$. If $e < D$, skip to Step 8; otherwise, skip to Step 9.

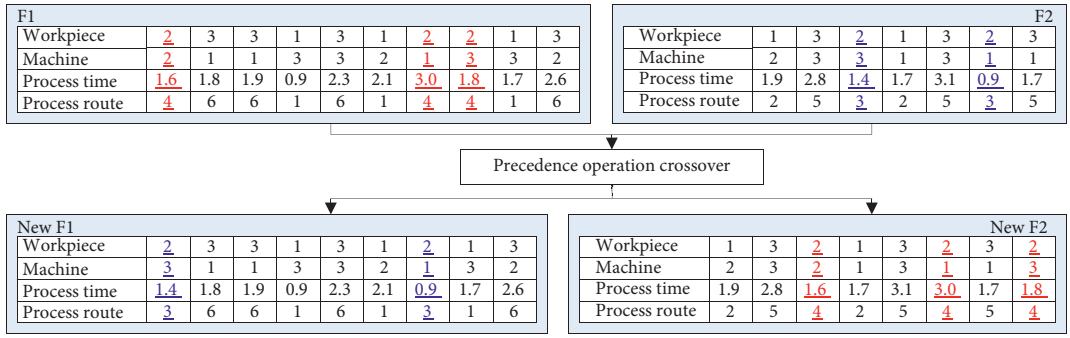

Figure 5: Crossover operation for MOJIT-IPPS.

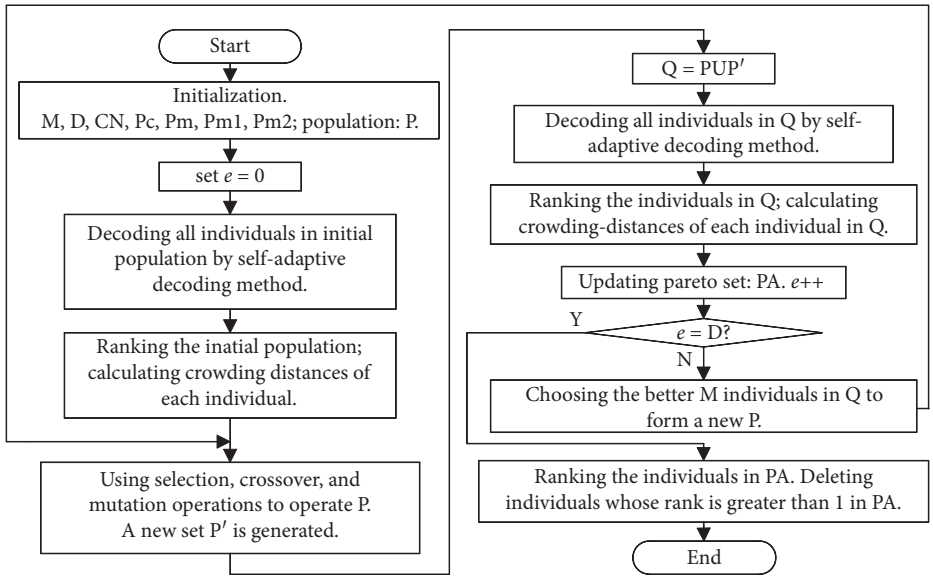

Figure 6: Flowchart of SD-NSGA-II for MOJIT-IPPS.

(viii) Step 8: the better M individuals in population Q are chosen to form the new generation P; skip to Step 3.

(ix) Step 9: PA is ranked, and individuals whose ranks are larger than 1 are deleted. The Pareto set is formed.

4.7. Flowchart of SD-NSGA-II. To present the flowchart of SD-NSGA-II, some parameters are defined as follows:

(i) M: population size, controlling the number of individuals in the population.

(ii) CN: number of compared individuals, CN < M.

(iii) D: iterations, controlling the number of iterations of the algorithm.

(iv) Pc: crossover probability of the two selected individuals, decimals in (0, 1) interval.

(v) Pm: mutation probability of individuals, decimals in (0, 1) interval.

(vi) Pm_1: probability that the swap operation is performed on codes of two columns of individuals, decimals in (0, 1) interval.

(vii) Pm_2: probability that the single-point mutation is performed on the machine layer of the current individual, decimals in (0, 1) interval.

A flowchart of SD-NSGA-II for the MOJIT-IPPS is shown in Figure 6.

5. Experimental Results and Analyses

5.1. Experiment 1: Multiobjective Optimization. There are seven types of shaft parts: $J_1 - J_7$. The delivery time of each part is 30, 22, 21, 24, 27, 25, and 30, respectively. There are seven machines: $M_1 - M_7$. There are six process methods: $P_1 - P_7$. There are fourteen process routes: $R_1 - R_{14}$. The earliness/tardiness penalty, maximum machine workload, and total machine workload are the objectives that will be minimized. The process routes of each workpiece are provided in Table 3.

In this study, the parameters of the SD-NSGA-II are set as $D = 500$, $M = 200$, CN = 2, Pc = 0.85, Pm = 0.1, Pm1 = 0.5, and Pm2 = 0.5. The C# programming language is used to create the algorithm. The results of the Pareto set are shown in Table 4. There are 32 solutions in the Pareto set.

Table 3: Process routes.

Ji	Ri	Details of Ri (machine code-process time)				
J_1	R_1	P_3 3–3.2, 7–4.4	P_2 2–2, 6–2.4	P_4 1–4.6, 5–4.4, 6–6.6	P_6 6–5.2, 7–5	P_1 1–2.9, 3–5.4, 4–3.2
	R_2	P_1 1–5.8, 3–2.1, 4–6.7	P_2 2–4.6, 6–3	P_5 2–4.1, 4–4.4	P_4 1–5.1, 5–5.3, 6–5.5	— —
J_2	R_3	P_7 4–3.2, 5–4.7	P_1 1–3.4, 3–6.3, 4–7.8	P_3 3–6.1, 7–4.4	P_6 6–2.4, 7–5.3	P_5 2–5.5, 4–4.3
	R_4	P_7 4–2.9, 5–3	P_1 1–4.5, 3–7, 4–6.2	P_2 2–6.2, 6–2	P_5 2–3.9, 4–4.7	— —
J_3	R_5	P_3 3–2.1, 7–3.5	P_6 6–5.9, 7–5.1	P_7 4–7.3, 5–6.7	P_4 1–6.6, 5–5.8, 6–3	— —
	R_6	P_6 6–3.4, 7–7.1	P_1 1–5.5, 3–5.4, 4–4.9	P_4 1–2.7, 5–7.8, 6–6.3	P_2 2–6.2, 6–3.6	— —
J_4	R_7	P_4 1–3, 5–5, 6–6	P_1 1–6.8, 3–5.2, 4–2.3	P_7 4–6.7, 5–7.7	— —	— —
	R_8	P_6 6–5.5, 7–3.3	P_5 2–6.1, 4–3.4	P_7 4–3.8, 5–7	P_2 2–2.1, 6–6.3	— —
J_5	R_9	P_2 2–2.2, 6–2.2	P_4 1–2.1, 5–7.9, 6–3.8	P_1 1–7, 3–7.9, 4–6	— —	— —
	R_{10}	P_1 1–6, 3–5, 4–2.1	P_2 2–5.1, 6–2.6	P_7 4–2.1, 5–4.7	— —	— —
J_6	R_{11}	P_5 2–7.5, 4–7.6	P_3 3–3.7, 7–4.3	P_1 1–2, 3–2.3, 4–2.6	P_4 1–3.8, 5–5.8, 6–3	P_6 6–4.2, 7–3.2
	R_{12}	P_1 1–6.8, 3–6.7, 4–4	P_4 1–5.9, 5–7.3, 6–3.2	P_7 4–7.3, 5–2.2	P_5 2–5.2, 4–5.3	— —
J_7	R_{13}	P_7 4–3.7, 5–7	P_5 2–4.3, 4–4.6	P_1 1–7, 3–4.2, 4–6.2	— —	— —
	R_{14}	P_2 2–4.8, 6–4.6	P_7 4–4.3, 5–2.1	P_6 6–2.7, 7–5.5	— —	— —

Table 4: Results of the Pareto set.

E/T penalty			Maximum machine workload			Total machine workload		
Max	Min	Average	Max	Min	Average	Max	Min	Average
7.3	0	1.9	25	15.2	18.9	96.6	85.2	89.4

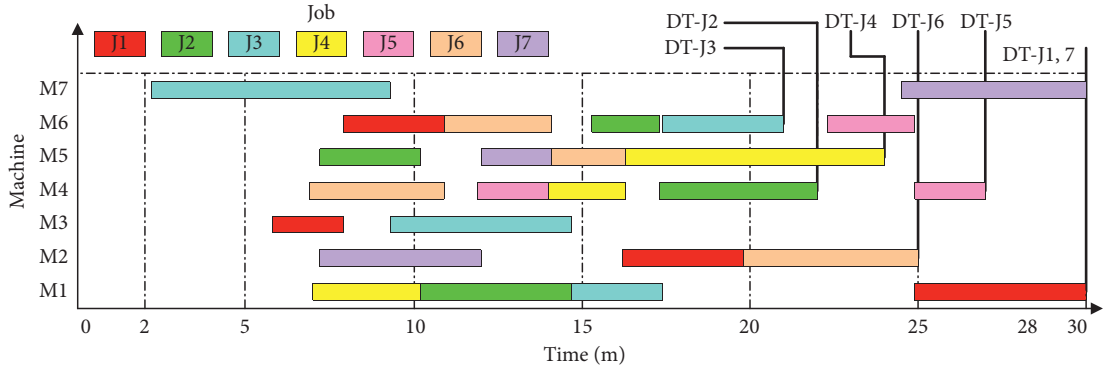

Figure 7: Gantt chart of a solution in the Pareto set.

To express the results more directly, we select a solution in the Pareto set whose E/T penalty is equal to zero. A Gantt chart of the solution is shown in Figure 7.

The mathematical model and SD-NSGA-II for MOJIT-IPPS are proven through experiment 1. The earliness/tardiness penalty, maximum machine workload, and total machine workload are minimized simultaneously. Compared with optimizing only one objective, the scheduling solutions obtained by considering multiple objectives are more feasible for determining the real scheduling solution.

Table 5: Examples of experiment 2.

Parameters of example 1 (Ex1)							
n	m	p	VRN	VRT	VRD (Ex1-1)	VRD (Ex1-2)	VRD (Ex1–3)
3	4	6	2–4	2.0–8.0	11–20	21–30	31–40
Parameters of example 2 (Ex2)							
n	m	p	VRN	VRT	VRD (Ex2-1)	VRD (Ex2-2)	VRD (Ex2-3)
6	6	12	3–4	2.0–8.0	11–20	21–30	31–40
Parameters of example 3 (Ex3)							
n	m	p	VRN	VRT	VRD (Ex3–1)	VRD (Ex3-2)	VRD (Ex3-3)
9	8	18	4–6	2.0–8.0	11–30	21–40	31–50
Parameters of example 4 (Ex4)							
n	m	p	VRN	VRT	VRD (Ex4–1)	VRD (Ex4–2)	VRD (Ex4-3)
12	10	24	5–7	2.0–8.0	11–30	21–40	31–50

Table 6: Algorithm parameters for the four sets of examples.

Algorithm parameters for Ex1			
M	D	Pc	Pm
100	200	0.9	0.1
Algorithm parameters for Ex2			
M	D	Pc	Pm
300	400	0.9	0.1
Algorithm parameters for Ex3			
M	D	Pc	Pm
400	500	0.9	0.1
Algorithm parameters for Ex4			
M	D	Pc	Pm
500	500	0.9	0.1

Table 7: Results for each example.

Example no.	Self-adaptive decoding method			Decoding method in [26]		
	E/T penalty			E/T penalty		
	Max	Min	Average	Max	Min	Average
Ex1-1	0	0	0	0	0	0
Ex1-2	0	0	0	0	0	0
Ex1-3	0	0	0	0	0	0
Ex2-1	**10**	**0.7**	**3.2875**	**12.9**	**4.3**	**7.306**
Ex2-2	**1.8**	**0**	**0.0825**	**2**	**0**	**0.316**
Ex2-3	0	0	0	0	0	0
Ex3-1	**52.5**	**9.8**	**28.768**	**65.9**	**35.1**	**50.6725**
Ex3-2	**5.2**	**0**	**0.2095**	**10.3**	**0**	**1.188**
Ex3-3	0	0	0	0	0	0
Ex4-1	**108.9**	**33.1**	**73.2435**	**153.1**	**104.1**	**125.904**
Ex4-2	**32.3**	**5.4**	**16.5465**	**62.6**	**10.7**	**33.1735**
Ex4-3	**2.2**	**0**	**0.02**	**10.1**	**0**	**0.479**

5.2. Experiment 2: Self-Adaptive Decoding Method. To evaluate the performance of the self-adaptive decoding method, a set of different scale examples is presented. As space is limited, several important parameters of the examples are provided in Table 5.

In Table 5, "VRN" represents the value range of the number of procedures of each process route, "VRT" represents the time range of each process, and "VRD" represents the delivery time range of each workpiece.

The algorithm parameters for the four sets of examples are shown in Table 6. Each set of examples is calculated 200 times by the two compared decoding methods. A comparison between the self-adaptive decoding method and a decoding method in [26] is shown in Table 7. For the twelve examples, the proposed self-adaptive decoding method provides better performance than the compared method [26] in Ex2-1, Ex2-2, Ex3-1, Ex3-2, Ex4-1, Ex4-2, and Ex4-3.

6. Conclusions

The main works in this paper is summarized as follows:

(1) The earliness/tardiness penalty, maximum machine workload, and total machine workload are minimized simultaneously in the MOJIT-IPPS problem. A mathematical model of MOJIT-IPPS is established.

(2) Due to the complexity and particularity of the MOJIT-IPPS problem, a SD-NSGA-II is proposed to solve the problem. Each module of the algorithm is designed, and a flowchart of the algorithm is drawn.

(3) The above model and algorithm are validated experimentally. The Pareto set is obtained through the proposed SD-NSGA-II. The results indicate the good performance of the proposed method.

Additionally, some future works will be carried out as follows:

(1) Algorithm improvement is a critical research direction for the IPPS problem. In this paper, a SD-NSGA-II is proposed for MOJIT-IPPS. We will continue research in this direction for the IPPS problem with delivery time. Specifically, more advanced algorithms will be developed to address different scale MOJIT-IPPS problems.

(2) The MOJIT-IPPS problem is a static scheduling problem in this paper. In future work, some dynamic factors will be considered in MOJIT-IPPS, such as inserting new workpieces, cancelling workpieces, and machine breakdown.

Data Availability

All data generated or analysed during this study are included in this article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant no. 51575443), Key Scientific Research Project of Education Department of Shaanxi Provincial Government, China (Grant no. 20JY047),

Doctoral Scientific Research Foundation of Xi'an University of Technology (Grant no. 102-451117013), China Postdoctoral Science Foundation (Grant no. 2020M673612XB), and Scientific Research Program Funded by Shaanxi Provincial Education Department, China (Grant no. 20JS114).

References

[1] X. Li, L. Gao, W. Wang, C. Wang, and L. Wen, "Particle swarm optimization hybridized with genetic algorithm for uncertain integrated process planning and scheduling with interval processing time," *Computers & Industrial Engineering*, vol. 135, pp. 1036–1046, 2019.

[2] L. P. Zhang, C. X. Yu, and T. N. Wong, "Cloud-based frameworks for the integrated process planning and scheduling," *International Journal of Computer Integrated Manufacturing*, vol. 32, no. 12, pp. 1–15, 2019.

[3] M. Petrovic, N. Vukovic, M. Mitic et al., "Integration of process planning and scheduling using chaotic particle swarm optimization algorithm," *Expert Systems with Applications*, vol. 64, pp. 569–588, 2016.

[4] V. K. Manupati, G. D. Putnik, M. K. Tiwari, P. Ávila, and M. M. Cruz-Cunha, "Integration of process planning and scheduling using mobile-agent based approach in a networked manufacturing environment," *Computers & Industrial Engineering*, vol. 94, pp. 63–73, 2016.

[5] C.-Y. Cheng and L.-W. Huang, "Minimizing total earliness and tardiness through unrelated parallel machine scheduling using distributed release time control," *Journal of Manufacturing Systems*, vol. 42, pp. 1–10, 2017.

[6] T. Wei and B. Khoshnevis, "Integration of process planning and scheduling-a review," *Journal of Intelligent Manufacturing*, vol. 11, no. 1, pp. 51–63, 2000.

[7] Y. W. Guo, W. D. Li, A. R. Mileham, and G. W. Owen, "Applications of particle swarm optimisation in integrated process planning and scheduling," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 2, pp. 280–288, 2009.

[8] A. Seker, S. Erol, and R. Botsali, "A neuro-fuzzy model for a new hybrid integrated Process Planning and Scheduling system," *Expert Systems with Applications*, vol. 40, no. 13, pp. 5341–5351, 2013.

[9] L. J. Yin, L. Gao, X. L. Li et al., "An improved genetic algorithm with rolling window technology for dynamic integrated process planning and scheduling problem," in *Proceedings of the 2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design*, pp. 414–419, Wellington, New Zealand, April 2017.

[10] L. Zhang and T. N. Wong, "An object-coding genetic algorithm for integrated process planning and scheduling," *European Journal of Operational Research*, vol. 244, no. 2, pp. 434–444, 2015.

[11] R. K. Phanden and A. Jain, "Assessment of makespan performance for flexible process plans in job shop scheduling," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 1948–1953, 2015.

[12] R. K. Phanden, A. Jain, and R. Verma, "An approach for integration of process planning and scheduling," *International Journal of Computer Integrated Manufacturing*, vol. 26, no. 4, pp. 284–302, 2013.

[13] X. Li, L. Gao, and X. Shao, "An active learning genetic algorithm for integrated process planning and scheduling," *Expert Systems with Applications*, vol. 39, no. 8, pp. 6683–6691, 2012.

[14] K. Lian, C. Zhang, L. Gao, and X. Li, "Integrated process planning and scheduling using an imperialist competitive algorithm," *International Journal of Production Research*, vol. 50, no. 15, pp. 4326–4343, 2012.

[15] C. Özgüven, L. Özbakır, and Y. Yavuz, "Mathematical models for job-shop scheduling problems with routing and process plan flexibility," *Applied Mathematical Modelling*, vol. 34, no. 6, pp. 1539–1548, 2010.

[16] G. Zhou, Z. Xiao, P. Jiang, and G. Q. Huang, "A game-theoretic approach to generating optimal process plans of multiple jobs in networked manufacturing," *International Journal of Computer Integrated Manufacturing*, vol. 23, no. 12, pp. 1118–1132, 2010.

[17] R. Zhang, S. K. Ong, and A. Y. C. Nee, "A simulation-based genetic algorithm approach for remanufacturing process planning and scheduling," *Applied Soft Computing*, vol. 37, pp. 521–532, 2015.

[18] E. Shokouhi, "Integrated multi-objective process planning and flexible job shop scheduling considering precedence constraints," *Production & Manufacturing Research*, vol. 6, no. 1, pp. 61–89, 2018.

[19] H. Xia, X. Li, and L. Gao, "A hybrid genetic algorithm with variable neighborhood search for dynamic integrated process planning and scheduling," *Computers & Industrial Engineering*, vol. 102, pp. 99–112, 2016.

[20] V. K. Manupati, P. C. Chang, and M. K. Tiwari, "Intelligent search techniques for network-based manufacturing systems: multi-objective formulation and solutions," *International Journal of Computer Integrated Manufacturing*, vol. 29, no. 8, pp. 850–869, 2016.

[21] V. K. Manupati, J. J. Thakkar, K. Y. Wong, and M. K. Tiwari, "Near optimal process plan selection for multiple jobs in networked based manufacturing using multi-objective evolutionary algorithms," *Computers & Industrial Engineering*, vol. 66, no. 1, pp. 63–76, 2013.

[22] Z. Zhang, R. Tang, T. Peng, L. Tao, and S. Jia, "A method for minimizing the energy consumption of machining system: integration of process planning and scheduling," *Journal of Cleaner Production*, vol. 137, pp. 1647–1662, 2016.

[23] M. Dai, D. B. Tang, Y. C. Xu et al., "Energy-aware integrated process planning and scheduling for job shops," *Proc IMechE Part B: J Engineering Manufacture*, vol. 229, no. S1, pp. 13–26, 2015.

[24] C. Moon and Y. Seo, "Evolutionary algorithm for advanced process planning and scheduling in a multi-plant," *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 311–325, 2005.

[25] M. Yazdani, A. Aleti, S. M. Khalili, and F. Jolai, "Optimizing the sum of maximum earliness and tardiness of the job shop scheduling problem," *Computers & Industrial Engineering*, vol. 107, pp. 12–24, 2017.

[26] H. A. Yang, Q. F. Sun, and J. Y. Li, "Novel decoding method for Job Shop earliness and tardiness scheduling problem," *Computer Integrated Manufacturing Systems*, vol. 17, no. 12, pp. 2652–2659, 2011.

[27] J. Schaller and J. Valente, "Branch-and-bound algorithms for minimizing total earliness and tardiness in a two-machine permutation flow shop with unforced idle allowed," *Computers & Operations Research*, vol. 109, pp. 1–11, 2019.

[28] I. Ribas, R. Companys, and X. Tort-Martorell, "An iterated greedy algorithm for solving the total tardiness parallel blocking flow shop scheduling problem," *Expert Systems with Applications*, vol. 121, pp. 347–361, 2019.

[29] R. O. Marinho Diana and S. R. de Souza, "Analysis of variable neighborhood descent as a local search operator for total weighted tardiness problem on unrelated parallel machines," *Computers and Operations Research*, vol. 117, pp. 1–16, 2020.

[30] R. Cordone and P. Hosteins, "A bi-objective model for the single-machine scheduling problem with rejection cost and total tardiness minimization," *Computers & Operations Research*, vol. 102, pp. 130–140, 2019.

[31] F. Jaramillo and M. Erkoc, "Minimizing total weighted tardiness and overtime costs for single machine preemptive scheduling," *Computers & Industrial Engineering*, vol. 107, pp. 109–119, 2017.

[32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[33] H. Panahi and R. Tavakkoli-Moghaddam, "Solving a multi-objective open shop scheduling problem by a novel hybrid ant colony optimization," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2817–2822, 2011.

[34] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 10, pp. 3202–3212, 2008.