Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 1257060, 11 pages
http://dx.doi.org/10.1155/2016/1257060

Research Article

A Pareto-Based Adaptive Variable Neighborhood Search for Biobjective Hybrid Flow Shop Scheduling Problem with Sequence-Dependent Setup Time

Huixin Tian,[1] Kun Li,[2] and Wei Liu[3]

[1]*School of Electrical Engineering & Automation, Tianjin Polytechnic University, Tianjin, China*
[2]*School of Management, Tianjin Polytechnic University, Tianjin, China*
[3]*Guangxi Colleges and University Key Laboratory of Minerals Engineering, Guangxi University, Guangxi, China*

Correspondence should be addressed to Kun Li; lk_neu@163.com

Received 29 December 2015; Revised 7 September 2016; Accepted 12 October 2016

Academic Editor: Yan-Jun Liu

Different from most researches focused on the single objective hybrid flowshop scheduling (HFS) problem, this paper investigates a biobjective HFS problem with sequence dependent setup time. The two objectives are the minimization of total weighted tardiness and the total setup time. To efficiently solve this problem, a Pareto-based adaptive biobjective variable neighborhood search (PABOVNS) is developed. In the proposed PABOVNS, a solution is denoted as a sequence of all jobs and a decoding procedure is presented to obtain the corresponding complete schedule. In addition, the proposed PABOVNS has three major features that can guarantee a good balance of exploration and exploitation. First, an adaptive selection strategy of neighborhoods is proposed to automatically select the most promising neighborhood instead of the sequential selection strategy of canonical VNS. Second, a two phase multiobjective local search based on neighborhood search and path relinking is designed for each selected neighborhood. Third, an external archive with diversity maintenance is adopted to store the nondominated solutions and at the same time provide initial solutions for the local search. Computational results based on randomly generated instances show that the PABOVNS is efficient and even superior to some other powerful multiobjective algorithms in the literature.

1. Introduction

In a typical hybrid flow shop scheduling (HFS) problem (Figure 1), a set of n jobs need to be processed through M production stages and at each stage k there are m_k identical parallel machines. Once completed at one stage, a job can be directly sent to the immediately following stage if at least one machine in this stage is available or can be stored at the infinite buffer between consecutive stages. The task of the HFS problem is to establish a production schedule so that some performance can be optimized (e.g., minimization of makespan, total weighted completion time, and tardiness of jobs [1]).

The HFS problem has been one of the important research issues in the production scheduling since proposed because many practical production scheduling problems can be modeled as a HFS problem [2, 3]. For example, in the iron and steel industry, most products are generally obtained by processing slabs through several consecutive stages, that is, hot rolling, cold rolling, and the continuous annealing (Figure 2).

Different from the single objective HFS problem in the literature, in practical production decision makers usually need to consider multiple objectives during scheduling, for example, (1) minimization of the total sequence-dependent setup times between consecutive jobs and (2) minimization of the total tardiness of all jobs. The two objectives are generally conflicting with each other. For example, to guarantee jobs can be completed before their due dates, urgent jobs should be arranged for production as early as possible, which in turn often causes huge transition of dimension or setup times. On the contrary, a good transition of dimension or small

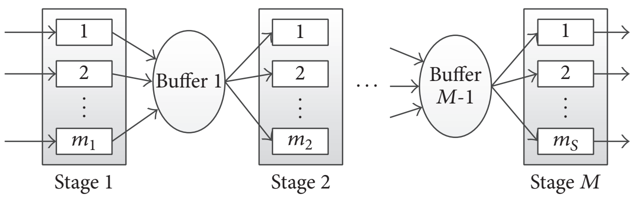

FIGURE 1: Structure of the hybrid flow shop with infinite buffers.

setup time also often causes large tardiness of jobs in practical production.

Therefore, in this paper we consider the biobjective HFS and develop a Pareto-based adaptive biobjective variable neighborhood search (PABOVNS) algorithm to solve it. The major features of the proposed PABOVNS algorithm are as follows.

(i) In the PABOVNS, a solution is coded as a sequence of all jobs and a decoding procedure is designed to obtain the corresponding complete schedule.

(ii) Since the adaptive strategy has been successfully used in the control system design and intelligent algorithms [4–11], an adaptive selection strategy of neighborhoods is proposed in the PABOVNS to select the most promising neighborhood, instead of the sequential selection strategy of canonical variable neighborhood search (VNS).

(iii) A two-phase multiobjective local search based on neighborhood search and path relinking [12] is designed for each selected neighborhood.

(iv) An external archive with diversity maintenance based on Pareto concept is adopted to store the nondominated solutions obtained by PABOVNS and at the same time it can provide initial solutions for the local search of PABOVNS.

The rest of this paper is organized as follows. Section 2 reviewed the related research results on HFS problem and biobjective HFS problem in the literature. The description of the biobjective HFS problem is given in Section 3. In Section 4, the proposed PABOVNS is described in detail. Computational results on randomly generated instances are presented in Section 5. Finally, the paper is concluded in Section 6.

2. Literature Review

Since proposed, the HFS problem has drawn a great deal of attention from researchers. Detailed reviews on the complexity, scheduling criteria, solving approaches, and applications of HFS problem can be found in Linn and Zhang [13], Ruiz and Vazquez-Rodriguez [14], and Ribas et al. [1]. Based on these reviews, it can be found that most previous researches focused on the single objective and that the solution methods in the literature can be classified into three categories: exact methods, heuristic methods, and metaheuristic methods.

Among the exact methods for the HFS problem, branch-and-bound (B&B) is the most preferred solution method. Dessouky et al. [15] first presented B&B method for the two- and three-stage HFS with uniform parallel machines, and the other kinds of B&B methods have been proposed for the general HFS problem with any amount of stages and parallel machines in many references [16–18].

Although the B&B method can solve the HFS problem to optimality, it should be noted that the size of solved problems is relatively small. Consequently, many researchers turned to develop dispatching rules and constructive heuristics so as to quickly obtain a near-optimal solution for the complex HFS problem. Brah [19] compared 10 different dispatching rules for the HFS with multiple stages to minimize the maximum tardiness. Guinet and Solomon [20] presented several dispatching rules and tailored heuristics for the M-stage HFS problem to minimize makespan and maximum tardiness. Sevastianov [21] and Kyparisis and Koulamas [22] tackled the HFS problem with multiple stages and uniform parallel machines in each stage and developed some heuristics to minimize makespan. Botta-Genoulaz [23] presented six heuristics for the HFS problem with precedence constraints so as to minimize the maximum lateness of jobs. Yang et al. [24] compared three heuristics based decomposition and local search methods for the HFS problem to minimize the total weighted tardiness. Recently, Lee et al. [25] developed an efficient heuristic based on the beam search and NEH method for the HFS problem.

The advantage of constructive heuristics is that they can provide a feasible solution quickly; however, the quality of the obtained solution is often not good enough for practical production. So metaheuristics become more and more popular for HFS problems in the literature. Grabowski and Pempera [26] investigated a no-wait HFS derived from real manufacturing system and developed a tabu search algorithm. Both Sawik [27] and Wardono and Fathi [28] proposed tabu search algorithm for the HFS problem with limited buffers. Wang and Tang [29] developed a hybrid tabu search that incorporated scatter search for the HFS with finite intermediate buffers. Cui and Gu [30] presented a discrete group search optimizer for HFS with random breakdown. Xiao et al. [31] designed a genetic algorithm for the *M-stage* HFS problem to minimize makespan. For the HFS problem with sequence-dependent setup times, Kurz and Askin [32] presented a genetic algorithm with the random keys representation. Ruiz and Maroto [33] further considered both the sequence-dependent setup times and machine eligibility in HFS problems and proposed a genetic algorithm. Zhang et al. [34] studied the HFS problem derived from a practical aeronautic production environment and developed a genetic algorithm hybrid with clustering. An artificial immune system was presented by Engin and Döyen [35] for HFS problem to minimize makespan. Tang and Wang [36] proposed an improved particle swarm optimization algorithm for the HFS problem. Ying and Lin [37] developed an ant colony optimization algorithm for the multiprocessor task problem with job precedence.

As reviewed above, different kinds of solution methods have been proposed in the literature for the HFS problem and most of them are focused on the single objective HFS problem. Although some researchers have paid more

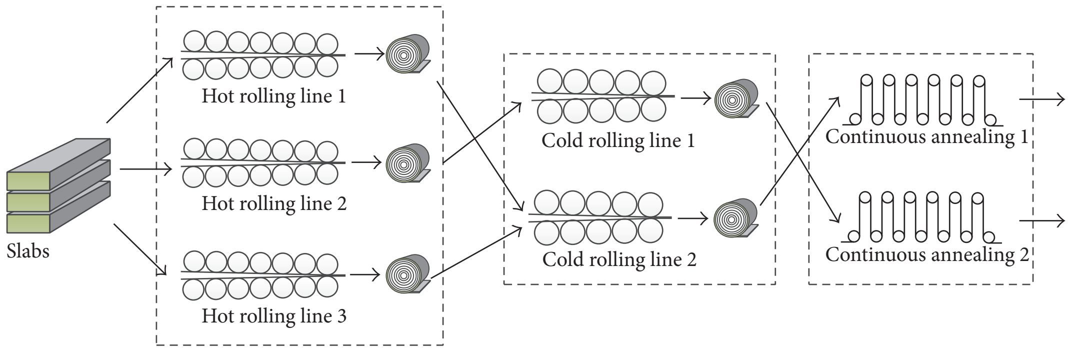

Figure 2: Illustration of hybrid flow shop in iron and steel industry.

attention to the multiobjective HFS problem, the related references are very few in the literature. Behnamian et al. [38] presented a three-phase hybrid metaheuristic based on Pareto optimal concept for a biobjective HFS to minimize the makespan and the earliness and tardiness of jobs. Rashidi et al. [39] developed a parallel multiobjective genetic algorithm for the biobjective HFS problem to minimize the makespan and total tardiness of jobs by dividing the population into several different subpopulations. Then, each subpopulation was assigned with different weights that were used to aggregate the two objectives. Abyaneh and Zandieh [40] proposed several methods for the biobjective HFS with sequence-dependent setup times and limited buffers. A biobjective local search algorithm was presented by Mousavi et al. [41] for the HFS to minimize the objectives of makespan and total tardiness. Recently, Marichelvam et al. [42] developed a discrete firefly algorithm for the biobjective HFS problem to minimize the makespan and mean flow time.

3. Problem Statement

The biobjective HFS problem considered in this paper can be stated as follows. There are a set of n jobs to be processed successively through M production stages, each of which has m_k identical parallel machines and any machine can be used to process jobs. It is assumed that all jobs have arrived as time zero. Each job i ($i = 1, 2, \ldots, n$) has a positive processing time p_{ik} at each stage k ($k = 1, 2, \ldots, M$), a due date d_i ($d_i > 0$), and a weight w_i ($w_i > 0$). Once a job has completed the required processing at a stage, it can be stored at the infinite buffer between consecutive stages or sent to the immediately following stage if at least one machine in this stage is available. In each stage k, whenever two jobs i and j are adjacent in the processing sequence of a machine (i.e., job j is processed immediately after job i on the same machine in stage k), a setup time t_{ijk} ($t_{ijk} \geq 0$) will be needed. Different from the objectives considered in previous literature [38–42] (e.g., makespan and total tardiness), the task of our HFS problem is to minimize the total sequence-dependent setup times and the total weighted tardiness of all jobs. The two objectives are adopted based on the description described for the practical production scheduling of iron and steel industry (Figure 2).

4. Pareto-Based Biobjective Variable Neighborhood Search

4.1. Canonical Single Objective VNS. VNS is a simple but powerful search algorithm for combinatorial optimization problems. The procedure of canonical VNS for the single objective optimization problem is given in Algorithm 1. The main search mechanism of VNS is to systematically change the neighborhood among candidate neighborhoods so as to enhance the search diversity.

Since proposed by Mladenović and Hansen [43], VNS has been successfully adopted to solve many kinds of difficult combinatorial optimization problems, especially the scheduling problems [44–46]. Therefore, in this paper we also adopt VNS and extend it to solve the biobjective HFS problem based on the Pareto dominance concept.

4.2. Multiobjective Optimization Based on Pareto Dominance. To give a clear description of our PABOVNS algorithm, we first give some explanations on the multiobjective optimization based on the concept of Pareto dominance.

Instead of optimizing a single objective, the multiobjective optimization endeavors to simultaneously achieve the optimization of multiple objectives that are conflicting with each other. For two given solutions, namely, X_1 and X_2, let the kth ($k = 1, \ldots, K$) objective of them be denoted by $f_k(X_1)$ and $f_k(X_2)$, and then solution X_1 is said to dominate X_2 if and only if the following two conditions are satisfied: (1) $f_k(X_1) \leq f_k(X_2)$ for every objective k; and (2) at the same time there exists at least one objective j such that $f_j(X_1) < f_j(X_2)$. If each one of a set of solutions cannot dominate any other solution, then the set of solutions are called the nondominated solutions. So if a solution X is not dominated by any other solution in the solution space, then solution X is called a Pareto optimal solution and the set of all Pareto optimal solutions is called the Pareto optimal set. Correspondingly, the objective vectors of the Pareto optimal set in the objective space are called the Pareto front. The task of multiobjective optimization is to achieve the Pareto optimal set so that the corresponding Pareto front can be distributed as evenly as possible in the objective space.

4.3. PABOVNS Algorithm. Based on the four features described above, the overall framework of the proposed

```
(1)  Input: initial solution $s_0$, a set of candidate neighborhoods $N = \{n_1, n_2, \ldots, n_q\}$, $g_{\max}$
(2)  Set the iteration number $g = 1$.
(3)  while $g < g_{\max}$
(4)      $i \coloneqq 1$
(5)      while $i \le q$ do
(6)          $s \coloneqq Shaking(s_0, n_i)$          //a random solution $s$ is generated in neighborhood $n_i$
(7)          $s' \coloneqq LocalSearch(s, n_i)$      //improve $s$ in neighborhood $n_i$ and get new solution $s'$
(8)          if $s'$ is better than $s_0$
(9)              $s_0 \coloneqq s'$
(10)             $i \coloneqq 1$                     //next search will start from the first neighborhood
(11)         else
(12)             $i \coloneqq i + 1$                 //next neighborhood is selected
(13)         end if
(14)     end while
(15)     $g = g + 1$
(16) end while
```

Algorithm 1: Search process of canonical VNS.

```
(1)  Input: a set of $q$ candidate neighborhoods $N = \{n_1, n_2, \ldots, n_q\}$, the maximum iterations $g_{\max}$, the
     selection probability of each neighborhood $j$ to be $p_j \coloneqq 1/q$, the iteration number $g \coloneqq 0$, the
     external archive $A$ to be empty, and $i = 0$.
(2)  Initialization of external archive $A$:
(3)      Generate an initial solution $S_0 \coloneqq CreateInitialSolution(\,)$ and set $A \coloneqq \{S_0\}$
(4)      while $i \le q$                             //$q$ is the sum of candidate neighborhoods
(5)          $NS \coloneqq LocalSearch(S_0, n_i)$          //perform multi-objective local search on $S_0$ in $n_i$ and
                                                   obtains a set of non-dominated solutions $NS$
(6)          $A \coloneqq UpdateArchive(A, NS)$        //update the external archive $A$ with $NS$
(7)          $i \coloneqq i + 1$
(8)      end while
(9)  while $g < g_{\max}$
(10)     $l \coloneqq SelectNeighborhood(\,)$            //select a neighborhood based on selection probabilities
(11)     Multi-objective local search: Phase I – neighborhood search:
(12)         $S \coloneqq SelectSolution(A)$             //randomly select a solution from $A$
(13)         $S' \coloneqq Shaking(S, n_l)$              //a random solution $S'$ is generated in neighborhood $n_l$
(14)         $NS_1 \coloneqq LocalSearch(S', n_l)$       //perform multi-objective local search on $S'$ in $n_l$
(15)     Multi-objective local search: Phase II – path relinking:
(16)         $S_1 \coloneqq SelectSolution(A)$           //randomly select a solution $S_1$ from $A$
(17)         $S_2 \coloneqq SelectSolution(A)$           //randomly select another solution $S_2$ from $A$
(18)         $NS_2 \coloneqq PathRelinking(S_1, S_2, n_l)$  //perform multi-objective path relinking from $S_1$ to $S_2$
(19)     Update external archive and selection probability of the selected neighborhood:
(20)         $A \coloneqq UpdateArchive(A, NS_1, NS_2)$  //update the external archive $A$ with $NS_1$ and $NS_2$
(21)         $UpdateProbability(n_l)$                //update the selection probability of neighborhood $n_l$
(22)     $g \coloneqq g + 1$
(23) end while
```

Algorithm 2: Overall procedure of the proposed PABOVNS.

PABOVNS algorithm can be given in Algorithm 2 and the components of it are described in the following subsections.

In the PABOVNS, we first generate an initial solution s_0 by the *CreateInitialSolution*( ) method. Then, a multiobjective neighborhood search *LocalSearch*( ) is performed on s_0 from neighborhood n_1 to n_q so as to initialize the external archive A. Subsequently, at each iteration of the VNS algorithm we will first select a neighborhood n_l by the adaptive selection method *SelectNeighborhood*( ) based on selection probabilities of neighborhoods and then perform the two-phase multiobjective local search. In the first phase, *LocalSearch*( ) is carried out on a perturbed solution, which is randomly selected from A, in the selected neighborhood n_l and a set of nondominated solutions (i.e., NS_1) is obtained. In the second phase, two random solutions s_1 and s_2 are first selected from A and then the multiobjective path-relinking method *PathRelinking*( ) based on the selected neighborhood n_l is adopted to obtain another set of nondominated solutions (i.e., NS_2). Finally, the nondominated solutions of NS_1 and NS_2 obtained in the two-phase local search are used to update the external archive A. Whenever A can be improved by NS, the selection probability of the selected neighborhood n_l will be increased. On the contrary, if NS fails to update A, the selection probability of neighborhood n_l will be decreased.

4.3.1. Solution Representation. The decision variables in HFS consist of two parts: the allocation of jobs to machines in each stage and the scheduling of jobs assigned to each machine. In the literature, the random key representation is often adopted [39, 41]. In this representation, each job is assigned to a real number that is generated within $[1, 1 + m_j)$ for each stage j. The integer part denotes the machine number to which this job is assigned and the fractional part is used to determine the sequence of jobs assigned to the same machine (e.g., sort these jobs in the nondescending order of the factional part). Although this representation method is simple, it is still difficult to define neighborhood structures. For example, for the job sequence of a certain machine, the insertion of a job from its originally assigned position to another position can be achieved by changing the fractional part of a job. However, such an insertion move is quite random because we have to sort the fractional part of jobs first if we want to accurately insert this job to a designated position. Therefore, we prefer to adopt a discrete version of solution representation so as to make it easy for solution space construction and neighborhood search. In this kind of representation, a solution is denoted as a sequence of all jobs (just like the solution for a canonical flow shop scheduling problem) and a decoding procedure is presented to obtain the corresponding complete schedule.

For a given solution represented by $S = (s_1, \ldots, s_k, \ldots, s_n)$ in which s_k denotes the job arranged at the kth position, the decoding procedure to construct a complete schedule of HFS can be given as follows.

Step 1. Set the earliest available time of all machines to be zero.

Step 2. Set $i = m_1$, and allocate the first m_1 jobs in s to the m_1 machines in Stage 1.

Step 3. Calculate the completion time of each job currently processed in Stage 1 and then make the complete schedule of the first completed job in Stage 1 on the next stages based on the first available machine rule (i.e., assign the first completed job to the first available machine in each of the next stages). Calculate the completion time of this job on each stage and then update the first available time of each machine in each stage.

Step 4. Set $i = i + 1$. If $i > n$, stop; otherwise, allocate job s_i to the first available machine in Stage 1 and go to Step 2.

In the above greedy decoding heuristic, the principle is that we select the first completed job in Stage 1 and then make its schedule on the next stages based on the first available machine rule. After this, next job is assigned to the first available machine in Stage 1. This process will be repeated until all the jobs have been scheduled. Since the intermediate buffer is infinite, the heuristic is quite simple. Based on this decoding method, we can deal with the neighborhood construction and search using the same way as for the flow shop scheduling whose solution is also represented as a sequence of jobs.

4.3.2. Initialization of the External Archive A

(1) CreateInitialSolution( ): Generation of Initial Solution. The initial solution s_0 is generated by a modified version of NEH method proposed by Nawaz et al. [47]. Since NEH is used in the single objective environment, we use the linear combination of the two objectives (i.e., $f_1 + f_2$, where f_1 and f_2 denote the total setup times and the total weighted tardiness). The procedure of this method can be described as follows.

Step 1. Sort all the jobs with the nondescending order of their due dates and denote the obtained job sequence as $S = (s_1, s_2, \ldots, s_n)$.

Step 2. Select the first two jobs s_1 and s_2 and determine their best sequence as if there are only the two jobs to be scheduled. Let the obtained partial sequence be S'.

Step 3. Select the next job from S and insert it into S' at the optimal position that can result in a minimal increase of the sum of setup time and weighted tardiness of jobs in S'.

Step 4. Repeat Step 3 until all jobs have been inserted to S', and then calculate the two objectives of the obtained solution S'. Set $s_0 = S'$.

In the above procedure, the decoding method described above will be used to transform each partial solution into the corresponding HFS schedule so as to evaluate the increase value of setup time and job tardiness.

(2) LocalSearch( ): Multiobjective Local Search Based on Neighborhood Search. As shown in [48], the multiobjective local search is very important for scheduling problems. So based on the initial solution s_0, we next turn to generate a set of nondominated solutions by a neighborhood based multiobjective local search. Since a solution is denoted as a job sequence, we adopt four kinds of neighborhoods: *insertion*, *swap*, *block insertion*, and *block swap*. The first two neighborhoods are often used for flow shop scheduling problems. The description of each neighborhood is given as follows.

(i) The insertion move removes a job in its current position and then inserts it to another position in a solution.

(ii) The swap move just swaps two jobs each time in a solution.

(iii) The block insertion move removes two adjacent jobs from their current position and then inserts them to another two adjacent positions.

(iv) The block swap move swaps two adjacent jobs with the other two adjacent jobs.

A neighborhood of a solution consists of all the possible neighborhood solutions that can be obtained by performing this kind of neighborhood move on this solution. In our algorithm, the four neighborhood are denoted as n_1, n_2, n_3, and, n_4.

During the local search, whenever a new solution is generated, it will be added to a solution set named *NS*. After the search of a neighborhood, *NS* will be truncated to only contain the nondominated solutions and then *NS* is used to update the external archive A.

(3) UpdateArchive( ): Update of the External Archive. In our algorithm, the external archive A is adopted to store the obtained nondominated solutions. Besides, the starting solution for each iteration of the VNS is also selected from it (see the *SelectSolution*( ) function) so as to enhance the ability of escaping from local optimum. For a given new solution s, the external archive A is updated as follows: if s is dominated by at least one solution in A, then s is discarded; otherwise, s is inserted into A. Since we have an upper bound on the size of A, whenever the size of A exceeds the upper bound, we will delete the most crowded solution from A based on the crowding metric of each solution in A. For the biobjective HFS, we first sort the solutions in A with the ascending order of the first objective (e.g., let the solution sequence as $S_1, S_2, \ldots, S_{|A|}$) and then the crowding metric of a solution S_i is calculated as the sum of Euclidean distances to solution S_{i-1} and S_{i+1}. Please note that the two objectives are normalized when calculating the Euclidean distances, and the normalized objectives are calculated as $f'_k(S) = f_k(S)/f_{k,\max}$ in which $f_k(S)$ is the original kth objective function value of solution S and $f_{k,\max}$ is the maximum value of the kth objective function value in the current external archive A.

4.3.3. Adaptive Selection Strategy: SelectNeighborhood( ) and UpdateProbability( ). In this section, we first give the definition of *success* and *failure* of a neighborhood search so as to determine and update the selection probability of each neighborhood.

As is shown in Algorithm 1, initially we set the selection probability of each neighborhood to be $1/q$. Once the two-phase multiobjective local search based on a neighborhood n_l is completed (the nondominated solutions obtained in each phase are stored in NS_1 and NS_2, resp.), we use the two sets of nondominated solutions in NS_1 and NS_2 to update the external archive A. If A is updated (i.e., new nondominated solutions are added into A), the neighborhood n_l is viewed as successful; otherwise, it is viewed as unsuccessful. During the iteration of our algorithm, we memorize the number of successful count succ_l and the unsuccessful count fail_l of each neighborhood n_l. Based on these counts, the selection probability of each neighborhood n_l can be calculated as $P_l = S_l/\sum_{i=1}^{q} S_l$, in which $S_l = \text{succ}_l/(\text{succ}_l + \text{fail}_l) + 0.01$ is called the success ratio. We add 0.01 to each S_l so as to avoid that the denominator has a value of zero. Based on the selection probability, we then use the roulette wheel method to select the neighborhood.

4.3.4. Two-Phase Multiobjective Local Search. After the neighborhood is selected, the two-phase multiobjective local search will be performed to generate new nondominated solutions. The motivation to adopt two-phase local search is that we want to achieve a balance between exploitation and exploration.

In Phase I, an initial solution S is randomly selected and then a shaking function (i.e., *Shaking*(S, n_l)) is used to perturb it by performing a random move selected from n_l. Finally, the full neighborhood search is performed on it to obtain a set of new solutions that are stored in NS_1. This full neighborhood search is the same one described in Section 4.3.2 and its search is focused on the exploitation.

Instead of focusing on exploitation, we prefer to develop a multiobjective path-relinking search in Phase II (*PathRelinking*(S_1, S_2, n_l)) so as to enhance the exploration ability of the local search. Path relinking (PR) is proposed by Glover [12] to generate new solutions by exploring a path that connects an *initial solution* and a *target solution* with a given kind of neighborhood move. At each step of PR, a move is performed so as to gradually decrease the difference between the initial solution and the target solution. Once the initial solution becomes the same one as the target solution, the search process of PR terminates. So the search behavior of PR is different from classical full neighborhood search and it can be used to enhance the search diversity (i.e., exploration ability) of local search. In our algorithm, we extend the classical single objective PR into multiobjective PR. For simplicity, we just give the procedure of multiobjective PR for the *swap* move as follows.

Step 1. Randomly select two solutions, namely, $S_1 = (s_{11}, s_{12}, \ldots, s_{1n})$ and $S_2 = (s_{21}, s_{22}, \ldots, s_{2n})$ in which s_{1k} and s_{2k} denote the job index arranged at the kth position in the two solutions, from the external archive, and let S_1 and S_2 be the initial solution and the target solution, respectively. Set $k = 1$ and NS_2 to be empty.

Step 2. If $s_{1k} \neq s_{2k}$, find the job whose index is s_{2k} in S_1 and swap it with s_{1k} to generate a new solution S'. Evaluate this new solution and use it to update NS_2 using the follow way: if S' is not dominated by any solution in NS_2, then add it to NS_2 and then delete all the solutions that are dominated by S'; otherwise, discard S'.

Step 3. Set $k = k + 1$. If $k > n$, go to Step 4; otherwise, go to Step 2.

After the two-phase multiobjective local search, two sets of nondominated solutions (NS_1 and NS_2) are used to update the external archive.

Table 1: Comparison results of the IGD metric between PABOVNS and PBOVNS.

M	n	Algorithms		Improvement	Sig	CPU (seconds)	
		PBOVNS	PABOVNS			PBOVNS	PABOVNS
3	30	**0.0014**	**0.0014**	0.00%	−	45.08	45.03
	80	0.0035	**0.0034**	2.86%	−	120.27	120.16
	100	0.0054	**0.0048**	11.11%	+	150.24	150.19
5	30	**0.0035**	0.0037	−5.71%	−	75.31	75.39
	80	0.0082	**0.0078**	4.88%	−	200.14	200.19
	100	0.0135	**0.0117**	13.33%	+	250.22	250.18
8	30	0.0124	**0.0119**	4.03%	+	120.09	120.14
	80	0.0438	**0.0387**	11.64%	+	320.34	320.21
	100	0.0614	**0.0560**	8.79%	+	400.26	400.34

5. Computational Experiments

5.1. Experiment Setting. To test the performance of the PABOVNS algorithm, computational experiments were carried out based on a set of randomly generated instances. In the experiments, our PABOVNS algorithm was implemented in C++ and all the experiments were carried out on a personal computer with Intel i7 4770 CPU (3.4 GHz) and 8 GB memory. The parameters used in our algorithm are set as follows: the size of the external archive A is set to 50 and the stopping criterion is adopted as the maximum available CPU time that is set as $0.5 \times n \times M$ seconds (please note that n is the number of jobs and M is the number of stages). In the experiments, all the testing algorithms share the same stopping criterion.

For the randomly generated instances, the number of stages is selected from $\{3, 5, 8\}$, the number of machines in each stage is uniformly generated in $[1, 5]$, and the number of jobs to be scheduled is selected from $\{50, 80, 100\}$. In addition, the processing time of each job is uniformly generated in $[1, 100]$, the sequence-dependent setup time of jobs is uniformly generated in $[1, 100]$, and the weight of each job is generated in $[1, 5]$. For the due date of each job j, we follow the generation method in [49] for the multiobjective permutation flow shop scheduling problem with setup times. That is, the due date of each job is generated by $d_j = (P_j + S_j) \times (1 + random \times 3)$, where P_j is the total processing times of job j on all stages, S_j is the sum of average setup time for all possible following jobs on all machines, and *random* is a random number uniformly distributed in $[0, 1]$. For each problem size (denoted by the number of stages and the number of jobs), we generate 30 random instances and thus there are a total of 270 instances tested in the experiments.

5.2. Performance Metrics. To evaluate the performance of the proposed PABOVNS algorithm, the performance metric named inverse general distance (IGD) is adopted in the experiments because this metric has been often adopted in the literature for multiobjective optimization (Zhou et al. [50]). The IGD metric can be defined as $\mathrm{IGD}(A, P^*) = \sum_{v \in P^*} d(v, A)/|P^*|$, where P^* is the Pareto optimal front or the reference Pareto front and $d(v, A)$ is the minimum Euclidean distance (in objective space) between point v and the points in A. Based on the definition, it can be seen that the IGD metric can measure both the convergence of A to P^* and the distribution diversity of points in A. Therefore, it is more favorable for A to have a small value of IGD.

It should be noted that it is impossible to obtain the true Pareto optimal set and the corresponding Pareto optimal front for the instances because the problem is NP-hard. So for each instance, we tested all the testing algorithms with a maximum available CPU time of $5 \times n \times M$ seconds and combine all the obtained external archives. Then, the nondominated solutions selected from the union of these external archives are used as the reference Pareto optimal set and the corresponding Pareto front is adopted as the reference Pareto front P^*. In addition, since the objectives in the biobjective HFS problem have different dimensions, we prefer to normalize the objective values of the nondominated solutions obtained by different testing algorithms into $[0, 1]$ so that the comparison results can be clear.

5.3. Computational Results. In this section, we first carried out preliminary experiments to illustrate the efficiency of the proposed improvement strategies, that is, the adaptive selection strategy of neighborhoods and the two-phase multiobjective local search. Then, we compared the PABOVNS algorithm to some other algorithms in the literature.

5.3.1. Efficiency of the Adaptive Selection Strategy of Neighborhoods. To analyze the impact of the adaptive selection strategy of neighborhoods on the performance of PABOVNS, we compared the proposed PABOVNS to another version of it in which the adaptive selection strategy is not adopted. In this experiment, the version without the adaptive selection strategy is denoted as PBOVNS, and the selection sequence of each neighborhood is n_1, n_2, n_3, and n_4. That is, in the two-phase multiobjective local search process of PBOVNS, neighborhood n_1 is firstly used. If the obtained NS_1 and NS_2 cannot update the external archive A, it turns to adopt the next neighborhood. But whenever the external archive A is updated, the algorithm turns back to adopt the first neighborhood n_1.

The comparison results of IGD metric between the two algorithms are given in Table 1, in which n is the number of jobs and M is the number of stages and Sig is the statistical

Table 2: Comparison results of the IGD metric between different algorithms.

M	n	Algorithms			Improvement		Sig	
		PABOVNS$_I$	PABOVNS$_{II}$	PABOVNS	Versus PABOVNS$_I$	Versus PABOVNS$_{II}$	Versus PABOVNS$_I$	Versus PABOVNS$_{II}$
3	30	**0.0014**	0.0017	**0.0014**	0.00%	17.65%	−	−
	80	0.0037	0.0039	**0.0034**	8.11%	12.82%	−	+
	100	0.0053	0.0057	**0.0048**	9.43%	15.79%	+	+
5	30	**0.0035**	0.0044	0.0037	−5.71%	15.91%	−	+
	80	0.0079	0.0086	**0.0078**	1.27%	9.30%	−	+
	100	0.0124	0.0135	**0.0117**	5.65%	13.33%	+	+
8	30	0.0130	0.0137	**0.0119**	8.46%	13.14%	+	+
	80	0.0403	0.0412	**0.0387**	3.97%	6.07%	+	+
	100	0.0580	0.0609	**0.0560**	3.45%	8.05%	+	+

Table 3: Comparison results of the IGD metric between different algorithms.

M	n	Algorithms			Improvement		Sig	
		MOPGA	NSGA-II	PABOVNS	Versus MOPGA	Versus NSGA-II	Versus MOPGA	Versus NSGA-II
3	30	**0.0014**	0.0015	**0.0014**	0.00%	6.67%	−	−
	80	0.0035	0.0036	**0.0034**	2.86%	5.56%	−	−
	100	0.0050	0.0053	**0.0048**	4.00%	9.43%	−	+
5	30	**0.0037**	0.0040	**0.0037**	0.00%	7.50%	−	+
	80	0.0080	0.0083	**0.0078**	2.50%	6.02%	−	+
	100	0.0126	0.0122	**0.0117**	7.14%	4.10%	+	+
8	30	0.0124	0.0128	**0.0119**	4.03%	7.03%	+	+
	80	0.0409	0.0415	**0.0387**	5.38%	6.75%	+	+
	100	0.0584	0.0596	**0.0560**	4.11%	6.04%	+	+

t-test result to show whether the two algorithms have significant performance difference. In Tables 1, 2, and 3, please note that the result is the average value of the ten instances for each problem size and the better results are shown in bold type. From the results, it can be seen that the PBOVNS can obtain the best results for only two groups of small size problems, namely, 3 × 30 and 5 × 30, while the PABOVNS succeeds in achieving the best results for 8 out of the 9 problem groups. The average improvement of PABOVSN over the PBOVNS is about 5.66%, which illustrates the efficiency of the proposed adaptive selection strategy of neighborhoods. In addition, the t-test with 95% confidence level show that the PABOVSN obtains significantly better results over PBOVNS for 5 problem groups ("+" means the performance difference is significant, while "−" means that the performance difference is insignificant). These experimental results show that the adaptive selection strategy of neighborhoods can help to improve the performance of PBOVNS. The major reason behind this phenomenon is that the adaptive selection strategy of neighborhoods can adaptively select the most promising neighborhood for current problem, which in turn improves the search efficiency.

5.3.2. Efficiency of the Two-Phase Multiobjective Local Search. As described in Section 4.3.4, the motivation of designing the two-phase multiobjective local search is to achieve a local search with a good balance of exploitation and exploration. So in this section, we further tested two variants of PABOVNS: the first variant only adopts Phase I as the local search and the second variant only adopts Phase II as the local search.

The comparison results between the three algorithms are given in Table 2, in which the first variant is denoted as PABOVNS$_I$ and the second variant is denoted as PABOVNS$_{II}$. Since the three algorithms have similar CPU times as is shown in Table 1, in this table the CPU time of each algorithm is not provided. When comparing PABOVNS$_I$ and PABOVNS$_{II}$, it appears that the PABOVNS$_I$ algorithm is superior to the PABOVNS$_{II}$ for all the testing problem groups. In addition, the PABOVNS$_I$ algorithm can obtain the best results for 2 problem groups, namely, 3 × 30 and 5 × 30, among all the three testing algorithms. The proposed PABOVNS can succeed to obtain the best results for 8 problem groups. More specifically, the proposed PABOVNS algorithm obtains better results over PABOVNS$_I$ for 7 problem groups, among which the performance difference is significant for 5 problem groups. In addition, the PABOVNS algorithm obtains better results over PABOVNS$_{II}$ for all the problem groups, among which the performance difference is significant for 8 problem groups. The reason is that the local search of Phase II (path relinking) has a better search diversity and it can help to enhance the exploration ability when combined with the local search of Phase I.

5.3.3. Comparison with Other Algorithms. In this section, we further compared our algorithm to the other two powerful algorithms in the literature. Since the biobjective HFS in [37]

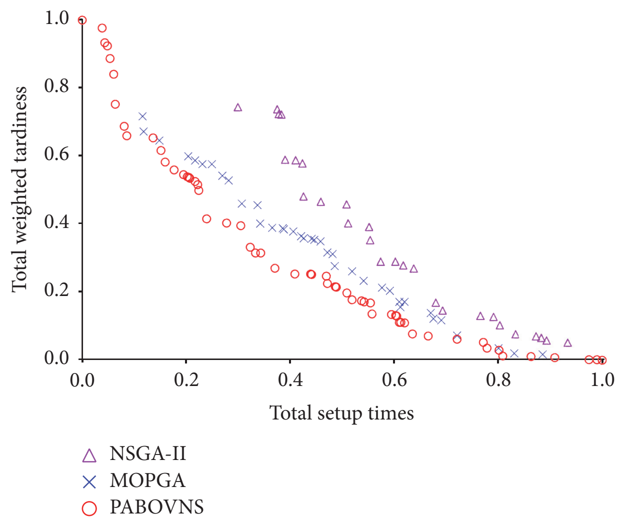

Figure 3: Pareto fronts obtained by three algorithms for a problem of 3×100 size.

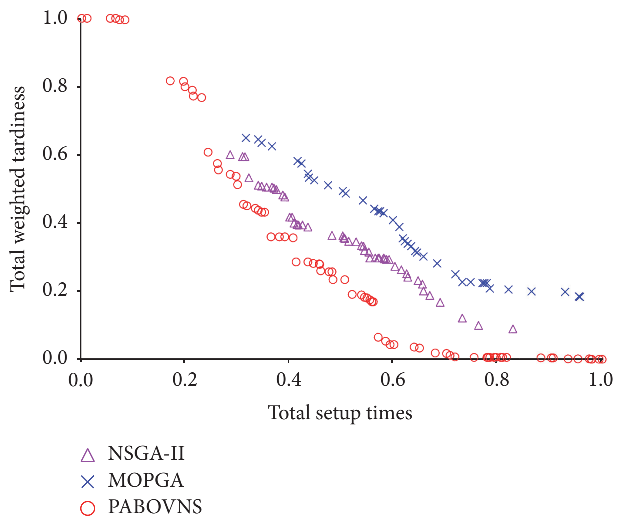

Figure 4: Pareto fronts obtained by three algorithms for a problem of 5×100 size.

is similar to our problem (only the objectives are different), the first algorithm for comparison is the multiobjective parallel genetic algorithm (MOPGA) proposed by Rashidi et al. [39] (in this experiment, we reimplemented this algorithm using suggested parameter settings in [39]). Besides the MOPGA, we developed another comparison algorithm based on the NSGA-II (Deb et al. [51]) which is the most famous multiobjective algorithm in the literature. To make the NSGA-II able to solve our problem, the following modifications are made. In the modified NSGA-II, the solution representation is the same one used in our PABOVNS, and the two-phase multiobjective local search is applied on a randomly selected solution from the first Pareto front (the NSGA-II ranks solutions in different Pareto front according to the objectives of solutions and the first Pareto front is the nondominated solutions obtained by NSGA-II). The crossover operator used in NSGA-II is the traditional two-cutting crossover operator and the mutation operator is a random insertion move performed on a given solution. The size of the population in NSGA-II is set to 500 and the mutation probability is set to $1/n$. At each iteration of NSGA-II, the new solutions generated by the local search and the new solutions generated by crossover and mutation are used to update the population. In this experiment, both of the two comparison algorithms adopt the same stopping criterion of our PABOVNS, that is, the maximum available CPU time of $0.5 \times n \times M$ seconds.

The comparison results for the three algorithms are presented in Table 3. Based on the results shown in Table 3, the following observations can be obtained.

(1) Both the MOPGA and our PABOVNS algorithms are superior to the modified NSGA-II algorithm. The MOPGA algorithm obtains better results than the NSGA-II for 8 out of the 9 problem groups.

(2) With the increase of problem size, the IGD values of the three algorithms tend to deteriorate due to the fact that large size problems are more difficult to solve.

(3) Our PABOVNS succeeds to achieve the best results for all the 9 problem groups. The average improvement achieved by the PABOVNS is 3.34% over the MOPGA and 6.57% over the modified NSGA-II, respectively.

(4) The PABOVNS can obtain significantly better results over MOPGA for 4 large size problem groups, namely, 5×100, 8×30, 8×80, and 8×100. With comparison to the modified NSGA-II algorithm, the PABOVNS achieves significantly better result for 7 problem groups.

To give a better understanding of the performance difference among the three algorithms, we further give the graphical illustration of the results obtained by the three algorithms for problem groups of 3×100, 5×100, and 8×100 in Figures 3, 4, and 5, respectively. From these figures, it can also be seen that our PABOVNS algorithm obtains the best Pareto front for the three problems. More specifically, the Pareto fronts obtained by the PABOVNS algorithm are closer to the referenced Pareto optimal front, and their distribution is also much better with comparison to the Pareto fronts obtained by the MOPGA and the NSGA-II.

6. Conclusions

In this paper, we investigate the biobjective HFS to minimize the total setup times and the total weighted tardiness. To efficiently solve this problem, we developed a Pareto-based adaptive biobjective variable neighborhood search algorithm with four major features: job sequence based coding and decoding method, adaptive selection strategy of neighborhood, two-phase multiobjective local search, and external archive with diversity maintenance. Computational experiments based on a set of randomly generated problems were carried out and the obtained results demonstrated that the proposed algorithm is effective and efficient for the biobjective HFS problem. In addition, the comparison

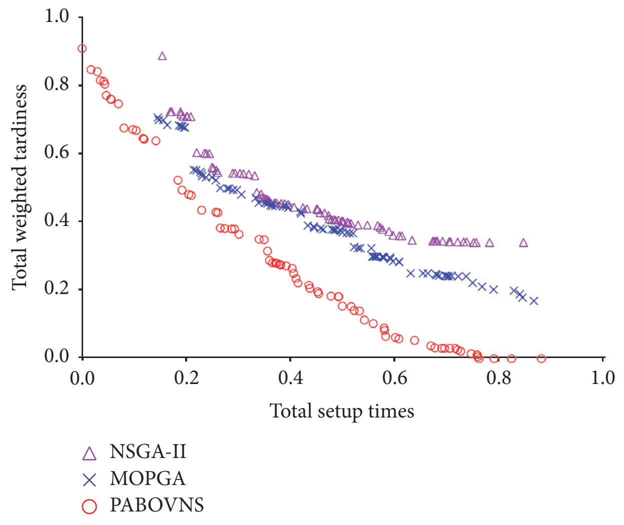

Figure 5: Pareto fronts obtained by three algorithms for a problem of 8 × 100 size.

results of the proposed algorithm to the other powerful metaheuristics in the literature also showed the proposed algorithm's efficiency and superiority.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grants 61403277 and 71602143).

References

[1] 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," *Computers and Operations Research*, vol. 37, no. 8, pp. 1439–1454, 2010.

[2] Z. H. Jin, K. Ohno, T. Ito, and S. E. Elmaghraby, "Scheduling hybrid flowshops in printed circuit board assembly lines," *Production and Operations Management*, vol. 11, no. 2, pp. 216–230, 2002.

[3] D. E. Deal, T. Yang, and S. Hallquist, "Job scheduling in petrochemical production: two-stage processing with finite intermediate storage," *Computers and Chemical Engineering*, vol. 18, no. 4, pp. 333–344, 1994.

[4] X. W. Chen, J. G. Zhang, and Y. J. Liu, "Research on the intelligent control and simulation of automobile cruise system based on fuzzy system," *Mathematical Problems in Engineering*, vol. 2016, Article ID 420308, 12 pages, 2016.

[5] C. L. P. Chen, Y.-J. Liu, and G.-X. Wen, "Fuzzy neural network-based adaptive control for a class of uncertain nonlinear stochastic systems," *IEEE Transactions on Cybernetics*, vol. 44, no. 5, pp. 583–593, 2014.

[6] Y.-J. Liu, J. Li, S. Tong, and C. L. Chen, "Neural network control-based adaptive learning design for nonlinear systems with full-state constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 7, pp. 1562–1571, 2016.

[7] Y.-J. Liu, Y. Gao, S. Tong, and Y. Li, "Fuzzy approximation-based adaptive backstepping optimal control for a class of nonlinear discrete-time systems with dead-zone," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 1, pp. 16–28, 2016.

[8] Y. J. Liu, S. C. Tong, D. J. Li, and Y. Gao, "Fuzzy adaptive control with state observer for a class of nonlinear discrete-time systems with input constraint," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 5, pp. 1147–1158, 2016.

[9] G. Y. Lai, Z. Liu, Y. Zhang, C. L. P. Chen, S. L. Xie, and Y. J. Liu, "Fuzzy adaptive inverse compensation method to tracking control of uncertain nonlinear systems with generalized actuator dead zone," *IEEE Transactions on Fuzzy Systems*, 2016.

[10] L. X. Tang and X. P. Wang, "A hybrid multi-objective evolutionary algorithm for multi-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 20–45, 2013.

[11] X. P. Wang and L. X. Tang, "An adaptive multi-population differential evolution algorithm for continuous multi-objective optimization," *Information Sciences*, vol. 348, pp. 124–141, 2016.

[12] F. Glover, "Tabu search and adaptive memory programming—advances, applications and challenges," in *Interfaces in Computer Science and Operations Research*, R. S. Barr, R. V. Helgason, and J. L. Kennington, Eds., vol. 7 of *Operations Research/Computer Science Interfaces Series*, pp. 1–75, Kluwer Academic, 1997.

[13] R. Linn and W. Zhang, "Hybrid flow shop scheduling: a survey," *Computers and Industrial Engineering*, vol. 37, no. 1, pp. 57–61, 1999.

[14] R. Ruiz and J. A. Vazquez-Rodriguez, "The hybrid flow shop scheduling problem," *European Journal of Operational Research*, vol. 205, no. 1, pp. 1–18, 2010.

[15] M. M. Dessouky, M. I. Dessouky, and S. K. Verma, "Flowshop scheduling with identical jobs and uniform parallel machines," *European Journal of Operational Research*, vol. 109, no. 3, pp. 620–631, 1998.

[16] S. A. Brah and J. L. Hunsucker, "Branch and bound algorithm for the flow shop with multiple processors," *European Journal of Operational Research*, vol. 51, no. 1, pp. 88–99, 1991.

[17] O. Moursli and Y. Pochet, "Branch-and-bound algorithm for the hybrid flowshop," *International Journal of Production Economics*, vol. 64, no. 1, pp. 113–125, 2000.

[18] E. Néron, P. Baptiste, and J. N. D. Gupta, "Solving hybrid flow shop problem using energetic reasoning and global operations," *Omega*, vol. 29, no. 6, pp. 501–511, 2001.

[19] S. A. Brah, "A comparative analysis of due date based job sequencing rules in a flow shop with multiple processors," *Production Planning and Control*, vol. 7, no. 4, pp. 362–373, 1996.

[20] A. G. P. Guinet and M. M. Solomon, "Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time," *International Journal of Production Research*, vol. 34, no. 6, pp. 1643–1654, 1996.

[21] S. V. Sevastianov, "Geometrical heuristics for multiprocessor flowshop scheduling with uniform machines at each stage," *Journal of Scheduling*, vol. 5, no. 3, pp. 205–225, 2002.

[22] G. J. Kyparisis and C. Koulamas, "Flexible flow shop scheduling with uniform parallel machines," *European Journal of Operational Research*, vol. 168, no. 3, pp. 985–997, 2006.

[23] V. Botta-Genoulaz, "Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness," *International Journal of Production Economics*, vol. 64, no. 1, pp. 101–111, 2000.

[24] Y. Yang, S. Kreipl, and M. Pinedo, "Heuristics for minimizing total weighted tardiness in flexible flow shops," *Journal of Scheduling*, vol. 3, no. 2, pp. 89–108, 2000.

[25] G.-C. Lee, J. M. Hong, and S.-H. Choi, "Efficient heuristic algorithm for scheduling two-stage hybrid flowshop with sequence-dependent setup times," *Mathematical Problems in Engineering*, vol. 2015, Article ID 420308, 10 pages, 2015.

[26] J. Grabowski and J. Pempera, "Sequencing of jobs in some production system," *European Journal of Operational Research*, vol. 125, no. 3, pp. 535–550, 2000.

[27] T. J. Sawik, "Scheduling algorithm for flexible flow lines with limited intermediate buffers," *Applied Stochastic Models and Data Analysis*, vol. 9, no. 2, pp. 127–138, 1993.

[28] B. Wardono and Y. Fathi, "A tabu search algorithm for the multi-stage parallel machine problem with limited buffer capacities," *European Journal of Operational Research*, vol. 155, no. 2, pp. 380–401, 2004.

[29] X. P. Wang and L. X. Tang, "A tabu search heuristic for the hybrid flowshop scheduling with finite intermediate buffers," *Computers & Operations Research*, vol. 36, no. 3, pp. 907–918, 2009.

[30] Z. Cui and X. Gu, "A discrete group search optimizer for hybrid flowshop scheduling problem with random breakdown," *Mathematical Problems in Engineering*, vol. 2014, Article ID 621393, 11 pages, 2014.

[31] W. Xiao, P. Hao, S. Zhang, and X. Xu, "Hybrid flow shop scheduling using genetic algorithms," in *Proceedings of the 3th World Congress on Intelligent Control and Automation*, pp. 537–541, IEEE Press, July 2000.

[32] M. E. Kurz and R. G. Askin, "Scheduling flexible flow lines with sequence-dependent setup times," *European Journal of Operational Research*, vol. 159, no. 1, pp. 66–82, 2004.

[33] R. Ruiz and C. Maroto, "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility," *European Journal of Operational Research*, vol. 169, no. 3, pp. 781–800, 2006.

[34] Y. Zhang, S. Liu, and S. Sun, "Clustering and genetic algorithm based hybrid flowshop scheduling with multiple operations," *Mathematical Problems in Engineering*, vol. 2014, Article ID 167073, 8 pages, 2014.

[35] O. Engin and A. Döyen, "A new approach to solve hybrid flow shop scheduling problems by artificial immune system," *Future Generation Computer Systems*, vol. 20, no. 6, pp. 1083–1095, 2004.

[36] L. Tang and X. Wang, "An improved particle swarm optimization algorithm for the hybrid flowshop scheduling to minimize total weighted completion time in process industry," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 6, pp. 1303–1314, 2010.

[37] K.-C. Ying and S.-W. Lin, "Multiprocessor task scheduling in multistage hybrid flow-shops: an ant colony system approach," *International Journal of Production Research*, vol. 44, no. 16, pp. 3161–3177, 2006.

[38] J. Behnamian, S. M. T. Fatemi Ghomi, and M. Zandieh, "A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic," *Expert Systems with Applications*, vol. 36, no. 8, pp. 11057–11069, 2009.

[39] E. Rashidi, M. Jahandar, and M. Zandieh, "An improved hybrid multi-objective parallel genetic algorithm for hybrid flow shop scheduling with unrelated parallel machines," *International Journal of Advanced Manufacturing Technology*, vol. 49, no. 9-12, pp. 1129–1139, 2010.

[40] S. H. Abyaneh and M. Zandieh, "Bi-objective hybrid flow shop scheduling with sequence-dependent setup times and limited buffers," *International Journal of Advanced Manufacturing Technology*, vol. 58, no. 1–4, pp. 309–325, 2012.

[41] S. M. Mousavi, M. Mousakhani, and M. Zandieh, "Bi-objective hybrid flow shop scheduling: a new local search," *International Journal of Advanced Manufacturing Technology*, vol. 64, no. 5–8, pp. 933–950, 2013.

[42] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 301–305, 2014.

[43] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers < Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[44] P. Hansen, N. Mladenović, and J. A. Moreno Pérez, "Variable neighbourhood search: methods and applications," *Annals of Operations Research*, vol. 175, no. 1, pp. 367–407, 2010.

[45] W. M. Cheng, P. Guo, Z. Q. Zhang, M. Zeng, and J. Liang, "Variable neighborhood search for parallel machines scheduling problem with step deteriorating jobs," *Mathematical Problems in Engineering*, vol. 2012, Article ID 928312, 20 pages, 2012.

[46] X. Wang and L. Tang, "A population-based variable neighborhood search for the single machine total weighted tardiness problem," *Computers and Operations Research*, vol. 36, no. 6, pp. 2105–2110, 2009.

[47] M. Nawaz, E. E. Enscore Jr., and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[48] X. P. Wang and L. X. Tang, "A machine-learning based memetic algorithm for the multi-objective permutation flowshop scheduling problem," *Computers & Operations Research*, vol. 79, pp. 60–77, 2017.

[49] M. Ciavotta, G. Minella, and R. Ruiz, "Multi-objective sequence dependent setup times permutation flowshop: a new algorithm and a comprehensive study," *European Journal of Operational Research*, vol. 227, no. 2, pp. 301–313, 2013.

[50] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhangd, "Multiobjective evolutionary algorithms: a survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[51] 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.