Research Article

A Bee Evolutionary Guiding Nondominated Sorting Genetic Algorithm II for Multiobjective Flexible Job-Shop Scheduling

Qianwang Deng, Guiliang Gong, Xuran Gong, Like Zhang, Wei Liu, and Qinghua Ren

State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Hunan University, Changsha 410082, China

Correspondence should be addressed to Guiliang Gong; gong_guiliang@163.com

Received 11 November 2016; Accepted 20 February 2017; Published 28 March 2017

Copyright © 2017 Qianwang Deng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Flexible job-shop scheduling problem (FJSP) is an NP-hard puzzle which inherits the job-shop scheduling problem (JSP) characteristics. This paper presents a bee evolutionary guiding nondominated sorting genetic algorithm II (BEG-NSGA-II) for multiobjective FJSP (MO-FJSP) with the objectives to minimize the maximal completion time, the workload of the most loaded machine, and the total workload of all machines. It adopts a two-stage optimization mechanism during the optimizing process. In the first stage, the NSGA-II algorithm with $T$ iteration times is first used to obtain the initial population $N$, in which a bee evolutionary guiding scheme is presented to exploit the solution space extensively. In the second stage, the NSGA-II algorithm with $GEN$ iteration times is used again to obtain the Pareto-optimal solutions. In order to enhance the searching ability and avoid the premature convergence, an updating mechanism is employed in this stage. More specifically, its population consists of three parts, and each of them changes with the iteration times. What is more, numerical simulations are carried out which are based on some published benchmark instances. Finally, the effectiveness of the proposed BEG-NSGA-II algorithm is shown by comparing the experimental results and the results of some well-known algorithms already existed.

1. Introduction

As a part of production scheduling and combinatorial optimization problems, job-shop scheduling problem (JSP) attracts more and more researchers from all walks of life (e.g., mechanical engineering, mathematics, and computer software engineering) in the recent decades [1–5].

Flexible job-shop scheduling problem (FJSP) inherits the characteristics of the JSP, in which each operation is allowed to be processed by any machine in a given set rather than one specified machine, and it has been proved that the FJSP is strong NP-hard [6]. FJSP consists of two subproblems: one is the routing subproblem that each operation is assigned to one machine of a set of machines, and the other is the scheduling subproblem that a feasible schedule is obtained by sequencing the assigned operations on all machines. Therefore the FJSP is more difficult to be solved than the classical JSP because of its need to determine the assignment of operations in related machines [7].

The FJSP is firstly addressed by Brucker and Schlie [8]. And they presented a polynomial algorithm with only two jobs and identical machines. Brandimarte [9] proposed a hybrid tabu search (TS) algorithm which was based on decomposition to solve the FJSP. Dauzère-Péres and Paulli [10] provided a TS algorithm which was based on the integrated approach and developed a new neighbourhood function [11] for the FJSP in terms of solution quality and computation time. Gao et al. [12] proposed a hybrid genetic algorithm to solve the FJSP with nonfixed availability constraints. And in order to enhance the inheritability, this genetic algorithm uses an innovative representation method and applies genetic operations to phenotype space. Saidi-Mehrabad and Fattahi [13] developed a TS algorithm that took the operation sequences and sequence-dependent setups into consideration to solve the FJSP. A genetic algorithm (GA) combined with a variable neighbourhood search (VNS) was presented by Gao et al. [14], and a GA with different strategies was proposed by Pezzella et al. [15]. Yazdani et al. [16] developed a parallel VNS...
algorithm based on the application of multiple independent searches which increased the exploration of search space. Xing et al. [17] put forth a knowledge-based ant colony optimization algorithm. Recently, a novel artificial bee colony (ABC) algorithm [18] and a discrete harmony search (DHS) algorithm [19] were brought forward to solve the FJSP.

As is shown above, the single-objective optimization of FJSP (SO-FJSP) has been extensively studied, which generally minimizes the makespan that is the time required to complete all jobs. However, many industries (e.g., aircraft, semiconductors manufacturing, and electronics) have trade-offs in their scheduling problems in which multiple objectives need to be considered to optimize the overall performance of the system. Therefore, the MO-FJSP may be closer to the realistic production environments and needs to be further studied. In recent years, the MO-FJSP has captured more and more interests of numerous domain researchers, and a great many of algorithms have been presented [20]. Compared with SO-FJSP, MO-FJSP has two problems to be dealt with: incommensurability between objectives and contradiction between objectives (i.e., optimizing a single-objective generally results in deterioration of another objective) [13].

For the MO-FJSP, Schaffer [21] provided a genetic algorithm with vector evaluated. Jurisch [22] presented a branch-and-bound algorithm and some heuristic algorithms. By combining the VNS with particle swarm optimization (PSO), Liu et al. [23] presented a hybrid metaheuristic to solve the MO-FJSP. A new genetic algorithm (GA) which hybridized with a bottleneck shifting procedure was developed by Gao et al. [24] to solve the MO-FJSP. Zhang et al. [25] embedded tabu search (TS) in PSO as a local search to deal with the MO-FJSP. Xing et al. [26] advanced an efficient search method for the MO-FJSP. González-Rodríguez et al. [27] proposed a generic multiobjective model which was based on lexicographical minimization of expected values for FJSP. By introducing several metrics of the multiobjective evaluation in the MO-FJSP literature, Rahmati et al. [28] adopted two multiobjective evolutionary algorithms for the MO-FJSP.

Recently, some studies based on the Pareto dominance relation have been used to solve the MO-FJSP and they are more desirable when compared with the prior linear weighted summation ones [29]. The nondominated sorting genetic algorithm (NSGA) [30] was one of the first methods used to solve the multiobjective problem. Kacem et al. [31] brought up a Pareto approach based on the hybridization of fuzzy logic (FL) and evolutionary algorithms to solve the MO-FJSP. Ho and Tay [32] integrated a guiding local search procedure and an elitism memory mechanism into the evolutionary algorithm to solve the MO-FJSP.

Deb et al. [33] came up with a nondominated sorting-based multiobjective evolutionary algorithm (MOEA), called nondominated sorting genetic algorithm II (NSGA-II). Wang et al. [34] proposed a multiobjective GA based on immune and entropy principle for the MO-FJSP. By employing simulated annealing (SA) algorithm as a local search process, Frutos et al. [35] introduced a memetic algorithm (MA) based on the NSGA-II. Wang et al. [36] presented an effective Pareto-based estimation of distribution algorithm (P-EDA), in which various strategies are integrated to maintain quality and diversity of solutions. Rohaninejad et al. [37] advanced an MO-FJSP problem with machines capacity constraints to minimize the makespan and overtime costs of machines. Kaplanoğlu [38] put forth an object-oriented (OO) approach combined with SA optimization algorithm to solve the MO-FJSP, in which a two-string encoding scheme was used to express this problem.

Our review of the above literatures reveals that the NSGA-II algorithm has been widely used to solve the MO-FJSP for its advantages such as high efficiency to optimize the complex problems and the ability to gain widespread Pareto-optimal solutions. And the algorithms with a two-stage optimization scheme have been also widely studied to solve the MO-FJSP for it could fully tap the optimization potentials of various metaheuristic algorithms. However, we found that the NSGA-II algorithm has the disadvantage of premature convergence to local solution and the algorithms with a two-stage optimization scheme have the disadvantages of being unable to gain stable and high quality initial population in the first stage. Hence, in this paper, we propose a bee evolutionary guiding NSGA-II (BEG-NSGA-II) with a two-stage optimization scheme to solve the MO-FJSP, which aims to fully play the respective advantages of NSGA-II algorithm and the algorithms with a two-stage optimization scheme and to overcome the disadvantages of them. In the first stage, the NSGA-II algorithm with \( T \) iteration times is first used to obtain the initial population \( N \) which consists of three parts changing with the iteration times. In order to extensively exploit the solution space, an effective local search operator is invented in this stage. In the second stage, the NSGA-II algorithm with \( GEN \) iteration times is used to obtain the Pareto-optimal solutions. In order to enhance the searching ability and avoid the premature convergence, an updating mechanism and some useful genetic operators were employed in this stage. Four famous benchmarks that include 53 open problems of FJSP are chosen to estimate the performance of the proposed algorithm. Moreover, by comparing the results of our algorithm and some existing well-known algorithms, the virtues of our algorithm can be clearly demonstrated.

The rest of this paper is organized as follows. The definition and formalization of the MO-FJSP are given in the next section. In Section 3, NSGA-II is briefly introduced, and then the overview and implementation details of the proposed BEG-NSGA-II are presented, respectively. Afterwards, Experimental Studies and Discussions are made in Section 4. Finally, Conclusions and Future Studies are described in Section 5.

### 2. Problem Definition

The FJSP is commonly defined as follows. There is a set of \( n \) jobs \( (J_i, i \in \{1, 2, \ldots, n\}) \) and a set of \( m \) machines \( (M_k, k \in \{1, 2, \ldots, m\}) \). One or more operation(s) \( (O_{ij}, j \in \{1, 2, \ldots, n_j\}, n_j \) is the total number of operations for job \( J_i \) is/are allowed to be processed by one machine of \( M_{k} \), which consists of a set of machines of the \( j \)th operations for job \( J_i \). \( F_{ijk} \) denotes the processing time of the \( j \)th operation for job \( J_i \), which is processed by machine \( k \). Generally, the FJSP consists of two subproblems: the routing subproblem of assigning
Table 1: The 4 × 5 problem.

|       | M1 | M2 | M3 | M4 | M5 |
|-------|----|----|----|----|----|
| Job 1 | O₁₁| 2  | 5  | 4  | 1  | 2  |
|       | O₁₂| 5  | 4  | 5  | 7  | 5  |
|       | O₁₃| 4  | 5  | 4  | 5  | 4  |
| Job 2 | O₂₁| 2  | 5  | 4  | 2  | 8  |
|       | O₂₂| 5  | 6  | 9  | 8  | 5  |
|       | O₂₃| 4  | 5  | 4  | 5  | 4  |
| Job 3 | O₃₁| 9  | 8  | 6  | 7  | 9  |
|       | O₃₂| 6  | 1  | 2  | 5  | 4  |
|       | O₃₃| 2  | 5  | 4  | 2  | 4  |
|       | O₃₄| 4  | 5  | 2  | 1  | 5  |
| Job 4 | O₄₁| 1  | 5  | 2  | 4  | 12 |
|       | O₄₂| 5  | 1  | 2  | 4  | 2  |

Each operation to a machine among a set of machines available and the scheduling subproblem of sequencing the assigned operations on all machines to obtain a feasible schedule for optimizing a certain objective function [39, 40].

One classical 4 × 5 FJSP is shown in Table 1. In this paper, we aim to minimize the following three objectives:

\[ \text{max} \quad f_1 = \max_{1 \leq k \leq m} (C_k) \]
\[ \text{max} \quad f_2 = \max_{1 \leq k \leq m} (W_k) \]
\[ \text{min} \quad f_3 = \sum_{k=1}^{m} W_k \]

Subject to:
\[ C_{ij} - C_{i(j-1)} \geq P_{ijk} x_{ijk}, \quad j = 2, \ldots, n_i; \quad \forall i, j \]
\[ [(C_{hg} - C_{ij} - t_{ijk}) x_{hgg} x_{ijk} \geq 0] \]
\[ \lor [(C_{ij} - C_{hg} - t_{ijk}) x_{hgg} x_{ijk} \geq 0], \quad \forall (i, j), (h, g), k \]
\[ \sum_{k \in M_{ij}} x_{ijk} = 1 \]
\[ x_{ijk} = \begin{cases} 1, & \text{if machine } k \text{ is selected for operation } O_{ij} \\ 0, & \text{otherwise.} \end{cases} \]

Equation (1) indicates the three optimizing objectives. Inequality (2) ensures the operation precedence constraint. Inequality (3) ensures that each machine processes only one operation at each time. Equation (4) states that one machine can be selected from the set of available machines for each operation [25]. According to whether machine \( k \) is selected to process step \( O_{ij} \) or not, equation (5) is used to determine the value of \( x_{ijk} \).

3. The Proposed Algorithm

3.1. Brief Introduction of NSGA-II. The nondominated sorting genetic algorithm II (NSGA-II) is a population-based multiobjective evolutionary algorithm, which is widely used in the optimization of multiobjective problems. The core procedure of NSGA-II can be briefly formulated as follows. First, using \( P_t \) and \( U_t \) presents the current parent and
The sizes of $P_t$ and $U_t$ are both $N$. Then a new population $R_t = P_t \cup U_t$ (of size $2N$) is formed by combining $P_t$ with $U_t$. Furthermore, an operator called nondominated sorting is executed, which defines $R_t$ as different nondominated levels ($rank1$, $rank2$, etc.), to choose the best $N$ members as the new population named $P_{t+1}$ for the next generation. In NSGA-II, the crowding distance is computed by a special operator, and then the solutions with larger crowding distance values are selected. More details of the NSGA-II could refer to [33].

3.2. The BEG-NSGA-II

3.2.1. Framework. In this paper, a bee evolutionary guiding nondominated sorting genetic algorithm II (BEG-NSGA-II) with a two-stage optimization scheme is proposed for the FJSP, in which a bee evolutionary guiding scheme is presented that focus on the exploitation of solution space, and some mechanisms are used to enhance the searching ability and avoid the premature convergence. Its framework is shown in Figure 1. More details of steps are described as follows.

The First Stage

Step 1. Generate $N$ individuals as the initial population randomly, and set $t = 1$.

Step 2. Calculate the fitness value of individuals based on objective 1, objective 2, and objective 3; then select the best fitness individuals as the queen 1, queen 2, and queen 3 correspondingly; if $t > 1$, compare the fitness value of the parent queens and offspring queens, and select the best ones as the new queens correspondingly.

Step 3. Use roulette wheel method to select $P$ individuals according to the fitness value of objective 1, objective 2, and objective 3.

Step 4. Randomly generate $R$ individuals, and combine these individuals with the $P$ individuals generated from Step 3, respectively.

Step 5. The new queens selected from Step 2 take crossover and mutation with other corresponding individuals generated from Step 4 to produce the offspring population 1, offspring population 2, and offspring population 3, respectively.

Step 6. Combine the three offspring populations into a combining population.

Step 7. Fast nondominated sorting and congestion computing.

Step 8. Select the top $N$ individuals as the offspring population.

Step 9. Set $t = t + 1$; if $t > T$, continue next step; otherwise return Step 2, and the population is replaced by the selected $N$ individuals.

The Second Stage

Step 10. Select $S$ top individuals (from the top $N$ individuals) as the elite individuals.

Step 11. Use a binary tournament selection method to select cross individuals from offspring population (i.e., the selected $N$ individuals); if $D(i, j) > u$, use a precedence operation crossover; otherwise, use job-based crossover.

Step 12. If $P_m <= 0.1$ and if offspring rank = 1, then choose swapping mutation; else, choose two binding mutation or reverse mutation; else, the population does not take mutation.

Step 13. Randomly generate $R'$ new individuals.

Step 14. Combine the three-part individuals which get from Steps 10, 12, and 13.

Step 15. Use Step 7 to deal with the combining population to generate offspring population.

Step 16. If $t > GEN$, output the result; otherwise, $t = t + 1$, and return Step 10.

The parameters used in this algorithm are defined as follows:

At the first stage,

$$\alpha = \frac{t}{T},$$

$$P = \alpha \times \frac{N}{2},$$

$$R = \frac{N (1 - \alpha)}{2}.$$  \hfill (6)

At the second stage,

$$\alpha' = \frac{t}{GEN},$$

$$R' = \frac{N (1 - \alpha')}{2},$$

$$\chi' + S + 0.5N >= N.$$  \hfill (7)

$$D(i, j) = \sum_{l=1}^{L} \left\lfloor \min \left[ \max \left(0, a_g - a_p \right), 1, 0 \right] \right\rfloor,$$  \hfill (8)

$$D(i, j) <= u, \text{ Precedence operation crossover}$$

$$D(i, j) > u, \text{ Job-based crossover.}$$  \hfill (9)

In (6)–(9), where $N$ is the number of individuals in the initial population, $t$ is the current iteration, $T$ is the iteration times in the first stage, and $GEN$ is the iteration times in the second stage. Equation (9) ensures that there are enough numbers of individuals to be selected in the following offspring, and we set the value of $S$ as $0.3 \times N$. Equation (10)
Figure 1: The framework of BEG-NSGA-II for MO-FJSP.
is a difference degree function which is used to compute the similarity between chromosome $i$ and chromosome $j$ (the two chromosomes that will be crossed), where $L$ is the length of a chromosome and $a_{ij}$ denotes the $i$th gene in chromosome $i$. In (11), $u$ is a parameter set as 0.7. The chromosomes tend to select the multipoint preservative crossover with the decrease of $u$.

3.2.2. Chromosome Encoding and Decoding. In this paper, we use the encoding method proposed by Gao et al. [12]. The FJSP problem includes two subproblems: one is operation sequence part and the other is machine assignment part. Chromosome that corresponds to the solution of the FJSP also consists of two parts. The first one is the operation sequence vector and the second one is the machine assignment vector. Two different encoding methods are used to generate the two vectors.

In terms of the operation sequence vector, the operation-based representation method is used, which is composed of the jobs’ numbers. This representation uses an array of uninterrupted integers from 1 to $n$, where $n$ is the total number of jobs, and each integer appears $O_{ni}$ times, where $O_{ni}$ is the total number of operations of job $i$; therefore, the length of the initial operation sequence population is equal to $\sum_{i=1}^{n} O_{ni}$. By scanning the operation sequence from left to right, the $n$th occurrence of a job number expresses the $m$th operation of this job. The operation sequence vector of every initial individual of population is generated with the randomly encoding principle. By using these representation features, any permutation of the operation sequence vector can be decoded to a feasible solution.

The machine assignment vector indicates the selected machines that are assigned to the corresponding operations for all jobs. It contains $n$ parts, and the length of $i$th part is $O_{ni}$, hence the length of this vector also equals $\sum_{i=1}^{n} O_{ni}$. The $i$th part of this vector expresses the machine assignment set of $i$th job. Supposing a machine set $S_{Ri} = \{m_{j1i}, m_{j2i}, \ldots, m_{jbi}\}$ can be selected to process the $i$th operation of job $i$, a gene set $\{g_{i1}, g_{i2}, \ldots, g_{ini}\}$ can be decoded by the $j$th machine $m_{jbi}$ from $S_{Ri}$. The machine assignment vector of every initial individual of population is generated by selecting the available machine randomly for each operation of every job. An example is shown in Figure 2(a), and the operation and machine sequence are shown in it as follows: $(O_{31},M_{4}), (O_{11},M_{1}), (O_{21},M_{2}), (O_{32},M_{1}), (O_{22},M_{3}), (O_{23},M_{4}), (O_{13},M_{2}), (O_{34},M_{4})$ and $(O_{23},M_{3})$; then we can get the value of objectives of the work by referring to Table I.

When the chromosome representation is decoded, each operation starts as soon as possible following the precedence and machine constraints [31]. A schedule generated by using this decoding method can be ensured to be active schedule [40]. The procedure of decoding is implemented as follows.

Step 1. Identify the machine of all operations based on the machine assignment vector.

Step 2. Identify the set of machines used to process every job.

Step 3. Identify the set of operations for every machine.

Step 4. Determine the allowable starting time of every operation. $AS_{ij} = C_{i(j−1)}$, where $AS_{ij}$ denotes the allowable starting time of operation $O_{ij}$, and $C_{i(j−1)}$ is the completion time of operation $O_{i(j−1)}$ for the same job.

Step 5. Calculate the idle time of the machine of operation $O_{ij}$, and get the idle areas $[t_{start}, t_{end}]$, where $t_{start}$ is the start time of these idle areas and $t_{end}$ is the end time of these idle areas. Scanning these areas from left to right, if max ($AS_{ij}, t_{start}$) + $t_{ijk}$ $\leq$ $t_{end}$, the earliest starting time is $S_{ij} = t_{start}$; else $S_{ij} = \text{max}(AS_{ij}, C_{i(j−1)})$.

Step 6. Calculate the completion time of every operation. $C_{ij} = S_{ij} + t_{ijk}$.

Step 7. Generate the sets of starting time and completion time for every operation of every job.

By using the above procedure, a feasible schedule for the FJSP is obtained. Figure 2 shows the examples of encoding.
and decoding methods, and the processing time and machine date of jobs can be seen from Table 1. This example contains 4 jobs and 5 machines. Job 1 and job 2 both have 3 operations; job 3 and job 4 contain 4 and 2 operations, respectively. Figure 2(a) shows a chromosome which contains two parts: the operation sequence vector and the machine assignment vector. The operation sequence vector is a permutation with repetitions of job numbers. It contains three 1s, three 2s, four 3s, and two 4s, because there are four jobs: job 1 contains 3 operations, job 2 contains 3 operations, job 3 contains 4 operations, and job 4 contains 2 operations. Its length is 12. The machine assignment vector consists of 4 parts because of 4 jobs. Its length is also 12. Each part presents the machines selected for the corresponding operations of job. For example, the first part contains 3 numbers which are 4, 3, and 5. Number 4 means that machine 4 is selected for operation 1 of job 1, number 3 means machine 3 is selected for operation 2 of job 1, and number 5 means machine 5 is selected for operation 3 of job 1. A Gantt chart of a schedule based on the chromosome in Figure 2(a) is shown in Figure 2(b).

3.2.3. Crossover Operators. In this paper, the proposed algorithm contains two optimization stages. In order to expand the searching space and avoid premature of local optimal solutions, we use different crossover operators in these two stages.

At the first stage, a single-point crossover (SPX) or multipoint crossover (MPX) operator is selected randomly (50%) for the operation sequence vector, and a two-point crossover (TPX) operator is selected for the machine assignment vector.

The procedure of SPX is described as follows ($P_1$ and $P_2$ are used to denote two parents; $O_1$ and $O_2$ are used to denote two offspring).

Step 1. A random parameter $k$ that meets the inequality $0 < k < P$ (the length of operation sequence vector) is generated to determine the position of the crossover.

Step 2. The elements from 1 to $k$ in $P_1$ are duplicated to $O_1$ in the same positions; and the elements from 1 to $k$ in $P_2$ are duplicated to $O_2$ in the same positions.

Step 3. Calculate the total number of each element (in this example, the total number of 1, 2, 3, and 4 is three, three, four, and two, resp.).

Step 4. The elements in $P_1$ are appended to the remaining empty positions in $O_1$ from left to right until the total number of each element in $O_1$ equals each one in $P_1$. The elements in $P_2$ are appended to the remaining empty positions in $O_2$ from left to right until the total number of each element in $O_2$ equals each one in $P_2$.

The procedure of MPX is described as follows ($P_1$ and $P_2$ are used to denote two parents; $O_1$ and $O_2$ are used to denote two offspring).

Step 1. Two random parameters $k_1$ and $k_2$ that meet the inequality $0 < k_1 < k_2 < P$ as well as $k_1 \neq k_2$ are generated to determine the positions of crossover.

Step 2. Append the elements between $k_1$ and $k_2$ in $P_1$ to the leftmost positions of $O_1$; and the elements from $k_1$ to $k_2$ in $P_2$ are appended to the leftmost positions of $O_2$.

Step 3. Calculate the total number of each element (in this example, the total number of 1, 2, 3, and 4 is three, three, four, and two, resp.).

Step 4. The elements in $P_2$ are appended to the remaining empty positions in $O_2$ from left to right until the total number of each element in $O_2$ equals each one in $P_2$. The elements in $P_1$ are appended to the remaining empty positions in $O_2$ from left to right until the total number of each element in $O_2$ equals each one in $P_2$.

The examples of SPX and MPX are, respectively, shown in Figures 3 and 4.

For the machine assignment vector, a two-point crossover (TPX) has been adopted here as the crossover operator. The procedure of it is described as follows ($P_1$ and $P_2$ are used to denote two parents; $O_1$ and $O_2$ are used to denote two offspring).

Step 1. Two random parameters $k_1$ and $k_2$ that meet the inequality $0 < k_1 < k_2 < P$ as well as $k_1 \neq k_2$ are generated to determine the positions of crossover.

Step 2. Append the elements between $k_1$ and $k_2$ positions in $P_2$ to the same positions in $O_1$, append the elements before $k_1$
and after $k_2$ positions in $P_1$ to the same positions in $O_1$, and use the same process to generate $O_2$.

Based on the above procedure, we can obtain feasible offspring if the parents are feasible. An example of TPX crossover is shown in Figure 5.

At the second stage, a precedence operation crossover (POX) or job-based crossover (J BX), which is determined by equation (11), is selected for operation sequence vector and the TPX is also selected for machine assignment vector.

The main procedure of POX is described as follows ($P_1$ and $P_2$ are used to denote two parents; $O_1$ and $O_2$ are used to denote two offspring).

**Step 1.** The Job set is randomly divided into two subsets: Jobset1 and Jobset2.

**Step 2.** The element(s) which belong(s) to Jobset1 in $P_1$ is (are) appended to the same position(s) in $O_1$ and deleted in $P_1$; and the element(s) which belong(s) to Jobset2 in $P_2$ is (are) appended to the same positions in $O_2$ and deleted in $P_2$.

**Step 3.** Append the elements remaining in $P_1$ to the remaining empty positions in $O_2$ from left to right; and append the elements remaining in $P_2$ to the remaining empty positions in $O_1$ from left to right.

The main procedure of J BX is described as follows ($P_1$ and $P_2$ are used to denote two parents; $O_1$ and $O_2$ are used to denote two offspring).

**Step 1.** The Job set is randomly divided into two subsets: Jobset1 and Jobset2.

**Step 2.** The element(s) which belong(s) to Jobset1 in $P_1$ is (are) appended to the same position(s) in $O_1$ and deleted in $P_1$; and the element(s) which belong(s) to Jobset2 in $P_2$ is (are) appended to the same positions in $O_2$.

**Step 3.** Append the elements remaining in $P_1$ to the remaining empty positions in $O_2$ from left to right; and append the elements remaining in $P_2$ to the remained empty positions in $O_1$ from left to right.

The examples of POX and J BX are, respectively, shown in Figures 6 and 7.
The main procedure of MPM operator is described as follows ($P_1$ and $O_1$ are used to denote a parent and offspring, resp.).

**Step 1.** Randomly select $k$ positions in $P_1$ ($k$ equals half of the length of the machine assignment vector).

**Step 2.** Change the value of these selected positions according to their optional machine sets selected for process the corresponding operations.

At the second stage, if the individual is at the forefront of Pareto, it will select the swapping mutation; otherwise choose the two binding mutation (TBM) or reverse mutation for operation sequence vector. The MPM operator is also selected for the machine assignment vector.

The main procedure of TBM is described as follows ($P_1$ and $O_1$ are used to denote a parent and offspring, resp.).

**Step 1.** A random parameter $m$ ($m < \text{the length of the operation sequence vector subtract } 3$) is generated in $P_1$.

**Step 2.** Exchange the elements $m$ with $m + 3$ and $m + 1$ with $m + 2$ in $P_1$ to generate $O_1$.

The examples of swapping mutation operator, reverse mutation operator, TBM operator, and MPM operator are, respectively, shown in Figures 8, 9, 10, and 11.

### 4. Experimental Studies and Discussions

The proposed BEG-NSGA-II algorithm was coded in MATLAB R2014a and implemented on a computer configured with Intel Core i3 CPU with 2.67 GHz frequency and 4 GB RAM. Four famous benchmarks that include 53 open problems of FJSP are chosen to estimate the proposed algorithm, which were also used by many researchers to evaluate their approaches. In order to illustrate the performance of the proposed algorithm, we compare our algorithm with other state-of-the-art reported ones. The computational time used to solve these benchmarks is also compared to show the good efficiency of the proposed method. Because the computation time and implementing performance are not only affected by the algorithm itself but also affected by the computer hardware, implementing software, and coding skills, we also append the information of hardware and software, as well as the original computational time with the corresponding algorithms. All experimental simulations were run 20 times, respectively, for each problem of these benchmarks. The adopted parameters of the BEG-NSGA-II are listed in Table 2.

#### 4.1. Experiment 1

The data of Experiment 1 are taken from Kacem et al. [31]. It contains 4 representative instances (problem $4 \times 5$, problem $8 \times 8$, problem $10 \times 10$, and problem $15 \times 10$). The experimental results and comparisons with other well-known algorithms are shown in Table 3 ($n \times m$ means that the problem includes $n$ jobs and $m$ machines; $f_1$, $f_2$, and $f_3$ mean the optimization objectives of makespan, MW, and TW, resp. $T$ means the average computer-independent CPU times in minute spent on each problem of these benchmarks; the symbol “—” means the time has not been given in the paper). BEG-NSGA-II denotes the proposed algorithm. The results of AL which are taken from Kacem et al. [31], PSO + SA which are taken from Xia and Wu [41], hGA which are taken from Gao et al. [24], MOGA which are taken from Wang et al. [34], hPSO which are taken from Shao et al. [42], and OO approaches which are taken from Kaplanoglu [38] are used to make comparison with the proposed algorithm. The bolded results are thenew Pareto-optimal solutions found in our algorithms.

For the problem $4 \times 5$, the proposed BEG-NSGA-II algorithm obtains not only all best solutions in these compared algorithms, but also another new Pareto-optimal solution ($f_1 = 13, f_2 = 7$, and $f_3 = 33$). The Gantt chart of this new Pareto solution is shown in Figure 12. For the problems $8 \times 8$, $10 \times 10$, and $15 \times 10$, the proposed algorithm gets more Pareto-optimal solutions than other listed algorithms but hPSO, and
Table 3: Results on Experiment 1.

| Problem | ALa | PSO + SAb | MOGAc | hPSOd | hGAe | OO approachf | BEG-NSGA-IIg |
|---------|-----|-----------|-------|-------|------|-------------|--------------|
| n x m   | f1  | f2  | f3   | T f1 | f2  | f3   | T f1 | f2  | f3   | T f1 | f2  | f3   | T f1 | f2  | f3   | T |
| 4 x 5   |     |     |     |     |     |      |     |     |     |     |     |      |     |     |     |     |
|         | 11  | 10  | 32  | 0.10| 11  | 10  | 32  | 0.03| 11  | 10  | 32  | 0.10 |
|         | 12  | 9   | 34  |     | 12  | 8   | 32  |     | 13  | 7   | 33  |     |
| 8 x 8   |     |     |     |     |     |      |     |     |     |     |     |      |     |     |     |     |
|         | 16  | 13  | 75  |     | 15  | 12  | 75  |     | 15  | 12  | 75  | 0.01 |
|         | 16  | 13  | 73  |     | 15  | 12  | 75  |     | 15  | 12  | 75  | 0.04 |
| 16  | 11  | 77  |     | 16  | 13  | 73  | 16  | 11  | 77  | 16  | 11  | 77  |     |
| 10 x 10 | 7   | 5   | 45  |     | 8   | 5   | 42  | 0.24| 7   | 5   | 43  | 0.20 |
|         | 7   | 6   | 42  |     | 7   | 6   | 42  |     | 8   | 5   | 42  | 7  |
|         | 8   | 7   | 41  |     | 8   | 7   | 41  |     | 7   | 7   | 43  | 8  |
| 10   | 7   | 5   | 45  |     | 8   | 7   | 41  |     | 8   | 7   | 41  |     |
| 15 x 10 | 23  | 11  | 91  |     | 12  | 10  | 95  |     | 12  | 10  | 93  |     |
|         | 12  | 10  | 95  |     | 12  | 10  | 93  |     | 14  | 12  | 91  |     |
|         | 11  | 10  | 98  |     | 11  | 10  | 95  |     | 11  | 10  | 95  |     |

a The computation configuration is not listed in AL.
b The computation configuration is not listed in PSO + SA.
c The CPU time on a PC with 2 GHz CPU and 2 GB of RAM memory in C++.
d The CPU time on a personal computer with 2 GHz CPU and 2 GB of RAM memory in C++.
e The CPU time on a 3.0 GHz Pentium in Delphi.
f The CPU time on an Intel Core i5, 2.67 GHz processor with 4 GB RAM of memory in Java.
g The CPU time on an Intel Core i3, 2.67 GHz processor with 4GB RAM in MATLAB R2014a.
although it obtains the same results with hPSO, it seems to consume less computation time than the hPSO does.

4.2. Experiment 2. The data of Experiment 2 are taken from Dauzère-Pérès and Paulli [10]. It contains 18 problems. The experimental results and comparisons with other well-known algorithms are shown in Table 4 (\(n \times m\) means that the problem includes \(n\) jobs and \(m\) machines, \(f_1, f_2,\) and \(f_3\) mean the optimization objectives of makespan, MW, and TW, resp. \(T\) means the average computer-independent CPU times in minute spent on each problem of these benchmarks). BEG-NSGA-II denotes the proposed algorithm. The results of MOGA are adopted from Wang et al. [34]. The bolded results are the new Pareto-optimal solutions found in our algorithms. From Table 4, except the problem 01a, we can see that our algorithm can obtain some new Pareto-optimal solutions which are better for objectives \(f_2\) or \(f_3\), or both of them, but a little worse for the objective \(f_1\). For the problem 01a, our algorithm gets the same solutions as MOGA. For all problems, our algorithm consumes less computational time than MOGA. Figure 13 illustrates the Gantt chart of the problem 18a (\(f_1 = 2,634, f_2 = 2,156,\) and \(f_3 = 21,005\)).

4.3. Experiment 3. The data of Experiment 3 are taken from Brandimarte [9]. It contains 10 problems. The experimental results and comparisons with other well-known algorithms are shown in Table 5 (\(n \times m\) means that the problem includes \(n\) jobs and \(m\) machines, \(f_1, f_2,\) and \(f_3\) mean the optimization objectives of makespan, MW, and TW, resp. \(T\) means the average computer-independent CPU times in minute spent on each problem of these benchmarks). BEG-NSGA-II denotes the proposed algorithm. The results of SM and MOGA are adopted from [26, 34]. The bolded results are the new Pareto-optimal solutions found in our algorithms. From Table 5,
Table 4: Results on Experiment 2.

| Problem (n x m) | MOGA* | BEG-NSGA-IIb | |
|-----------------|--------|---------------|---|
| f₁  | f₂  | f₃  | T | f₁  | f₂  | f₃  | T |
| 01a (10 x 5) | 2,568  | 2,505  | 11,137 | 2.04 | 2,568  | 2,505  | 11,137 | 1.91 |
| 02a (10 x 5) | 2,289  | 2,263  | 11,137 | 2.56 | 2,487  | 2,237  | 11,137 | 2.13 |
| 03a (10 x 5) | 2,287  | 2,248  | 11,137 | 2.90 | 2,492  | 2,247  | 11,137 | 2.5  |
| 04a (10 x 5) | 2,550  | 2,503  | 11,090 | 2.07 | 2,882  | 2,572  | 11,069 | 2.03 |
| 05a (10 x 5) | 2,292  | 2,252  | 11,077 | 2.37 | 2,740  | 2,250  | 10,981 | 2.12 |
| 06a (10 x 5) | 2,250  | 2,233  | 11,009 | 3.09 | 2,673  | 2,194  | 10,893 | 2.86 |
| 07a (15 x 8) | 2,450  | 2,413  | 16,485 | 7.63 | 2,927  | 2,187  | 16,485 | 6.21 |
| 08a (15 x 8) | 2,187  | 2,102  | 16,485 | 8.27 | 2,621  | 2,091  | 16,485 | 6.35 |

* MOGA: Multi-Objective Genetic Algorithm
b BEG-NSGA-II: Bi-Objective Genetic Algorithm
Table 4: Continued.

| Problem \((n \times m)\) | MOGA* | BEG-NSGA-IIb |
|------------------------|-------|-------------|
| \(f_1\) | \(f_2\) | \(f_3\) | \(T\) | \(f_1\) | \(f_2\) | \(f_3\) | \(T\) |
| \hline
| 09a (15 \times 8) | 2,157 | 2,113 | 16,485 | 10.16 | 2,442 | 2,103 | 16,485 | 7.26 |
| | 2,144 | 2,119 | 16,485 | | 2,470 | 2,074 | 16,485 | |
| | 2,158 | 2,102 | 16,485 | | 2,455 | 2,081 | 16,485 | |
| | 2,461 | 2,433 | 16,505 | 7.55 | 2,998 | 2,317 | 16,487 | 6.43 |
| | 2,470 | 2,310 | 16,537 | | 3,036 | 2,363 | 16,482 | |
| | 2,478 | 2,330 | 16,533 | | 3,091 | 2,491 | 16,473 | |
| | 2,482 | 2,360 | 16,499 | | 2,984 | 2,319 | 16,486 | |
| | 2,501 | 2,265 | 16,547 | | 3,052 | 2,377 | 16,477 | |
| | 2,501 | 2,312 | 16,528 | | 3,098 | 2,491 | 16,472 | |
| | 2,547 | 2,476 | 16,490 | | 3,013 | 2,293 | 16,487 | |
| 10a (15 \times 8) | 3,064 | 2,734 | 16,464 | | 3,136 | 2,457 | 16,471 | |
| | | | | | 3,025 | 2,377 | 16,480 | |
| | | | | | 3,041 | 2,351 | 16,481 | |
| | | | | | 2,980 | 2,258 | 16,494 | |
| | | | | | 3,021 | 2,319 | 16,484 | |
| 11a (15 \times 8) | 2,182 | 2,170 | 16,449 | 10.14 | 2,551 | 2,116 | 16,229 | 7.25 |
| | 2,202 | 2,114 | 16,476 | | 2,599 | 2,131 | 16,224 | |
| | 2,210 | 2,113 | 16,442 | | 2,612 | 2,146 | 16,222 | |
| | 2,337 | 2,185 | 16,377 | | 2,573 | 2,058 | 16,231 | |
| | 2,874 | 2,389 | 16,247 | | 2,753 | 2,186 | 16,217 | |
| | 2,894 | 2,330 | 16,247 | | 2,628 | 2,102 | 16,224 | |
| | 2,962 | 2,312 | 16,247 | | 2,702 | 2,098 | 16,223 | |
| | | | | | 2,658 | 2,089 | 16,224 | |
| | | | | | 2,555 | 2,111 | 16,229 | |
| | | | | | 2,664 | 2,063 | 16,228 | |
| 12a (15 \times 8) | 2,161 | 2,107 | 16,295 | 11.92 | 2,624 | 2,049 | 16,055 | 8.22 |
| | 2,168 | 2,130 | 16,220 | | 2,620 | 2,022 | 16,068 | |
| | 2,191 | 2,084 | 16,355 | | 2,587 | 2,035 | 16,065 | |
| | 2,210 | 2,103 | 16,331 | | 2,614 | 2,057 | 16,051 | |
| | 2,315 | 2,125 | 16,292 | | 2,585 | 2,043 | 16,061 | |
| | 2,366 | 2,105 | 16,237 | | 2,659 | 2,038 | 16,055 | |
| | 2,493 | 2,297 | 16,124 | | 2,617 | 2,024 | 16,065 | |
| | 2,631 | 2,309 | 16,112 | | 2,575 | 2,057 | 16,057 | |
| | 2,637 | 2,303 | 16,113 | | 2,587 | 2,050 | 16,059 | |
| | 2,683 | 2,397 | 16,104 | | 2,582 | 2,098 | 16,052 | |
| | | | | | 2,595 | 2,036 | 16,059 | |
| | | | | | 2,566 | 2,044 | 16,061 | |
| | | | | | 2,894 | 2,277 | 21,610 | 10.27 |
| | | | | | 2,928 | 2,233 | 21,610 | |
| | | | | | 2,896 | 2,244 | 21,610 | |
| | | | | | 2,877 | 2,282 | 21,610 | |
| 13a (20 \times 10) | 2,408 | 2,326 | 21,610 | 23.99 | 2,649 | 2,186 | 21,610 | 12.39 |
| | | | | | 2,649 | 2,186 | 21,610 | |
| | | | | | 2,664 | 2,207 | 21,610 | |
| | | | | | 2,738 | 2,182 | 21,610 | |
| 14a (20 \times 10) | 2,285 | 2,247 | 21,610 | 33.29 | 2,627 | 2,216 | 21,610 | 13.60 |
| | 2,287 | 2,218 | 21,610 | | 2,681 | 2,196 | 21,610 | |
| | | | | | 2,645 | 2,215 | 21,610 | |
| | | | | | 2,652 | 2,201 | 21,610 | |
| | | | | | 2,668 | 2,197 | 21,610 | |
| | | | | | 2,682 | 2,178 | 21,610 | |
Table 4: Continued.

| Problem (n x m) | MOGA$^a$ | BEG-NSGA-II$^b$ |  |
|----------------|----------|-----------------|---|
|                | $f_1$    | $f_2$           | $f_3$ | $f_1$    | $f_2$           | $f_3$ | $T$ |
|                |          |                 |       |          |                 |       |     |
| 16a (20 x 10)  | 2,447    | 2,354           | 21,602| 21.52    | 2,965           | 2,279 | 21,534| 11.48|
|                | 2,450    | 2,380           | 21,590|          | 3,158           | 2,303 | 21,504|       |
|                | 2,487    | 2,454           | 21,584|          | 3,184           | 2,400 | 21,490|       |
|                | 2,492    | 2,417           | 21,576|          | 3,039           | 2,279 | 21,523|       |
|                | 2,540    | 2,396           | 21,547|          | 3,050           | 2,318 | 21,507|       |
|                | 2,550    | 2,492           | 21,545|          | 3,071           | 2,286 | 21,513|       |
|                | 2,568    | 2,428           | 21,540|          | 3,275           | 2,281 | 21,511|       |
|                | 3,013    | 2,588           | 21,478|          | 3,105           | 2,352 | 21,496|       |
|                | 3,106    | 2,548           | 21,478|          | 3,175           | 2,344 | 21,497|       |
|                |          |                 |       |          |                 |       |     |
| 17a (20 x 10)  | 2,322    | 2,240           | 21,433| 28.47    | 2,806           | 2,195 | 21,096| 13.92|
|                | 2,322    | 2,280           | 21,362|          | 2,750           | 2,165 | 21,120|       |
|                | 2,323    | 2,238           | 21,454|          | 2,791           | 2,141 | 21,114|       |
|                | 2,343    | 2,224           | 21,420|          | 2,716           | 2,157 | 21,129|       |
|                | 2,480    | 2,285           | 21,344|          | 2,849           | 2,148 | 21,107|       |
|                | 2,528    | 2,231           | 21,313|          | 2,769           | 2,166 | 21,111|       |
|                | 2,789    | 2,448           | 21,198|          | 2,856           | 2,183 | 21,098|       |
|                | 2,808    | 2,303           | 21,200|          | 2,773           | 2,145 | 21,111|       |
|                | 2,816    | 2,370           | 21,197|          | 2,856           | 2,142 | 21,105|       |
|                |          |                 |       |          |                 |       |     |
| 18a (20 x 10)  | 2,267    | 2,235           | 21,483| 33.01    | 2,634           | 2,156 | 21,005| 16.23|
|                | 2,269    | 2,206           | 21,408|          | 2,642           | 2,126 | 21,006|       |
|                | 2,320    | 2,208           | 21,354|          | 2,638           | 2,185 | 20,999|       |
|                | 2,437    | 2,221           | 21,311|          | 2,638           | 2,148 | 21,002|       |
|                | 2,531    | 2,310           | 21,285|          | 2,674           | 2,148 | 21,000|       |
|                | 2,545    | 2,305           | 21,282|          | 2,644           | 2,145 | 21,002|       |
|                |          |                 |       |          |                 |       |     |
|                |          |                 |       |          |                 |       |     |

$^a$The CPU time on a PC with 2 GHz CPU and 2 GB of RAM memory in C++.

$^b$The CPU time on an Intel Core i3, 2.67 GHz processor with 4 GB RAM in MATLAB R2014a.

we can see that our algorithm can obtain some new Pareto-optimal solutions in problems MK01, MK04, MK05, MK08, and MK10 compared with SM and MOGA and the same solutions in MK2, MK3, MK6, MK7, and MK9 as MOGA. For all problems, our algorithm consumes less computational time than MOGA. Figure 14 illustrates the Gantt chart of the problem MK08 ($f_1 = 541, f_2 = 533, and f_3 = 2,516$).

4.4. Experiment 4. The data of Experiment 4 are taken from Barnes and Chambers [43]. It contains 21 problems. The experimental results and comparisons with other well-known algorithms are shown in Table 6 ($n \times m$ means that the problem includes $n$ jobs and $m$ machines, $f_1$, $f_2$, and $f_3$ mean the optimization objectives of makespan, MW, and TW, resp. $T$ means the average computer-independent CPU
Table 5: Results on Experiment 3.

| Problem (n × m) | SM<sup>a</sup> | MOGA<sup>b</sup> | BEG-NSGA-II<sup>b</sup> |
|-----------------|-----------------|-----------------|--------------------------|
|                 | f<sub>1</sub>   | f<sub>2</sub>   | f<sub>3</sub> | T           | f<sub>1</sub> | f<sub>2</sub> | f<sub>3</sub> | T           | f<sub>1</sub> | f<sub>2</sub> | f<sub>3</sub> | T           |
| MK01 (10 × 6)   | 42 42 162 4.78  | 42 39 158 0.49  | 42 39 158 0.36        |
|                 | 44 40 154 44    | 44 40 154 44    | 44 40 154 44          |
|                 | 43 40 155 43    | 43 40 155 43    | 43 40 155 43          |
|                 | 40 36 169 40    | 40 36 169 40    | 40 36 169 40          |
|                 | 47 36 167 47    | 47 37 165 47    | 47 37 164 47          |
|                 | 49 37 164 50    | 49 38 162 50    | 49 38 162 50          |
| MK02 (10 × 6)   | 28 28 155 3.02  | 26 26 151 0.75  | 26 26 151 0.70        |
|                 | 27 27 146 27    | 27 27 146 27    | 27 27 146 27          |
|                 | 29 27 145 29    | 29 27 145 29    | 29 27 145 29          |
|                 | 29 29 143 29    | 29 29 143 29    | 29 29 143 29          |
|                 | 31 31 141 31    | 31 31 141 31    | 31 31 141 31          |
|                 | 33 33 140 33    | 33 33 140 33    | 33 33 140 33          |
| MK03 (15 × 8)   | 204 204 852 26.14 | 204 199 855 4.75 | 204 199 855 3.66       |
|                 | 204 144 871 204 | 204 144 871 204 | 204 144 871 204        |
|                 | 204 135 882 204 | 204 135 882 204 | 204 135 882 204        |
|                 | 204 133 884 204 | 204 133 884 204 | 204 133 884 204        |
|                 | 213 199 850 213 | 213 199 850 213 | 213 199 850 213        |
|                 | 214 210 849 214 | 214 210 849 214 | 214 210 849 214        |
|                 | 221 199 847 221 | 221 199 847 221 | 221 199 847 221        |
|                 | 222 199 848 222 | 222 199 848 222 | 222 199 848 222        |
|                 | 223 188 848 223 | 223 188 848 223 | 223 188 848 223        |
|                 | 230 177 848 230 | 230 177 848 230 | 230 177 848 230        |
| MK04 (15 × 8)   | 68 67 352 17.74 | 66 63 345 1.76  | 66 63 345 1.63         |
|                 | 65 63 362 65    | 65 63 362 65    | 65 63 362 65           |
|                 | 63 61 371 63    | 63 61 371 63    | 63 61 371 63           |
|                 | 62 61 373 62    | 60 59 390 60    | 60 59 390 60           |
|                 | 61 60 382 61    | 74 54 349 74    | 74 54 349 74           |
|                 | 60 59 390 60    | 74 55 348 74    | 74 55 348 74           |
|                 | 73 55 350 72    | 72 72 342 72    | 72 72 342 72           |
|                 | 74 54 349 78    | 78 78 339 78    | 78 78 339 78           |
|                 | 74 55 348 74    | 72 66 348 72    | 72 66 348 72           |
|                 | 90 76 331 90    | 73 72 343 73    | 73 72 343 73           |
| MK05 (15 × 4)   | 177 177 702 8.26 | 173 173 683 2.34 | 173 173 683 1.96       |
|                 | 175 175 682 175 | 175 175 682 175 | 175 175 682 175        |
|                 | 183 183 677 183 | 183 183 677 183 | 183 183 677 183        |
|                 | 185 185 676 185 | 179 179 679 179 | 179 179 679 179        |
|                 | 179 179 679 179 | 199 199 674 199 | 199 199 674 199        |
|                 | 193 193 675 193 | 193 193 675 193 | 193 193 675 193        |
|                 | 183 183 677 183 | 183 183 677 183 | 183 183 677 183        |
| MK06 (10 × 15)  | 67 431 18.79 67 | 62 55 424 1.93  | 62 55 424 1.85         |
|                 | 65 54 417 65    | 65 54 417 65    | 65 54 417 65           |
|                 | 60 58 441 60    | 60 58 441 60    | 60 58 441 60           |
|                 | 62 60 440 62    | 62 60 440 62    | 62 60 440 62           |
|                 | 76 60 362 76    | 76 60 362 76    | 76 60 362 76           |
|                 | 76 74 356 76    | 76 74 356 76    | 76 74 356 76           |
|                 | 78 60 361 78    | 78 60 361 78    | 78 60 361 78           |
|                 | 73 72 360 73    | 73 72 360 73    | 73 72 360 73           |
|                 | 72 72 361 72    | 72 72 361 72    | 72 72 361 72           |
|                 | 100 90 330 100  | 100 90 330 100  | 100 90 330 100         |
Table 5: Continued.

| Problem $(n \times m)$ | SM$^a$ | MOGA$^b$ | BEG-NSGA-II$^b$ |
|------------------------|--------|----------|-----------------|
|                        | $f_1$  | $f_2$  | $f_3$ | $T$ | $f_1$  | $f_2$  | $f_3$ | $T$ | $f_1$  | $f_2$  | $f_3$ | $T$ |
| MK07 $(20 \times 5)$   | 150    | 150    | 717   | 5.68 | 139    | 139    | 693   | 4.92 | 139    | 139    | 693   | 3.12 |
|                        | 140    | 138    | 686   |      | 140    | 138    | 686   |      | 144    | 144    | 673   |      |
|                        | 151    | 151    | 667   |      | 151    | 151    | 667   |      | 157    | 157    | 662   |      |
|                        | 162    | 162    | 659   |      | 162    | 162    | 659   |      | 166    | 166    | 657   |      |
| MK08 $(20 \times 10)$  | 523    | 523    | 2,524 | 67.67| 523    | 515    | 2,524 | 12.04| 523    | 515    | 2,524 | 8.34 |
|                        | 523    | 497    | 2,534 |      | 523    | 497    | 2,534 |      | 524    | 524    | 2,519 |      |
|                        | 578    | 578    | 2,489 |      | 578    | 578    | 2,489 |      | 587    | 587    | 2,484 |      |
|                        | 587    | 587    | 2,484 |      | 587    | 587    | 2,484 |      | 541    | 533    | 2,516 |      |
|                        | 542    | 542    | 2,511 |      | 542    | 542    | 2,511 |      | 560    | 560    | 2,505 |      |
| MK09 $(20 \times 10)$  | 311    | 299    | 2,374 | 77.76| 311    | 299    | 2,290 | 19.48| 311    | 299    | 2,290 | 10.15|
|                        | 310    | 299    | 3514  |      | 310    | 299    | 3514  |      | 311    | 301    | 2,287 |      |
|                        | 314    | 299    | 2,315 |      | 314    | 299    | 2,315 |      | 315    | 299    | 2,283 |      |
|                        | 315    | 299    | 2,265 |      | 332    | 302    | 2,265 |      | 329    | 301    | 2,266 |      |
|                        | 328    | 305    | 2,259 |      | 328    | 305    | 2,259 |      | 325    | 299    | 2,275 |      |
| MK10 $(20 \times 15)$  | 227    | 221    | 1,989 | 122.52| 224    | 219    | 1,980 | 17.87| 224    | 219    | 1,980 | 12.38|
|                        | 225    | 211    | 1,976 |      | 225    | 211    | 1,976 |      | 233    | 214    | 1,919 |      |
|                        | 233    | 214    | 1,919 |      | 240    | 215    | 1,905 |      | 235    | 218    | 1,897 |      |
|                        | 235    | 218    | 1,897 |      | 246    | 215    | 1,896 |      | 235    | 225    | 1,895 |      |
|                        | 240    | 215    | 1,905 |      | 252    | 224    | 1,884 |      | 240    | 216    | 1,888 |      |
|                        | 242    | 214    | 1,913 |      | 266    | 254    | 1,864 |      | 246    | 215    | 1,896 |      |
|                        | 246    | 215    | 1,896 |      | 246    | 215    | 1,896 |      | 246    | 224    | 1,884 |      |
|                        | 256    | 211    | 1,919 |      | 246    | 205    | 1,940 |      | 256    | 211    | 1,919 |      |
|                        | 260    | 244    | 1,869 |      | 251    | 202    | 1,948 |      | 260    | 244    | 1,869 |      |
|                        | 266    | 254    | 1,864 |      | 253    | 207    | 1,930 |      | 268    | 264    | 1,858 |      |
|                        | 276    | 256    | 1,857 |      | 255    | 202    | 1,943 |      | 281    | 268    | 1,854 |      |
|                        | 217    | 207    | 2,064 |      | 250    | 205    | 1,935 |      | 214    | 204    | 2,082 |      |
|                        | 254    | 203    | 1,942 |      | 244    | 207    | 1,935 |      |        |        |      |      |

$^a$The CPU time on a Pentium IV 2.4 GHz processor with 1.0 GB of RAM memory in MATLAB.

$^b$The CPU time on a PC with 2 GHz CPU and 2 GB of RAM memory in C++.

$^c$The CPU time on an Intel Core i3, 2.67 GHz processor with 4 GB RAM in MATLAB R2014a.
### Table 6: Results on Experiment 4.

| Problem \((n \times m)\) | HGTS \(^a\) | HA \(^b\) | BEG-NSGA-II \(^c\) |
|--------------------------|-------------|------------|-------------------|
| \(mt10c1\ (10 \times 11)\) | \(927\) | \(927\) | \(1,092\) |
| \(mt10cc\ (10 \times 12)\) | \(908\) | \(908\) | \(1,005\) |
| \(mt10x\ (10 \times 11)\) | \(918\) | \(918\) | \(1,117\) |
| \(mt10xx\ (10 \times 12)\) | \(918\) | \(918\) | \(1,117\) |
| \(mt10xy\ (10 \times 12)\) | \(905\) | \(905\) | \(1,061\) |
| \(mt10xyz\ (10 \times 13)\) | \(847\) | \(847\) | \(925\) |
| \(mtb4c9\ (15 \times 11)\) | \(914\) | \(914\) | \(1,035\) |
| \(mtb4cc\ (15 \times 12)\) | \(907\) | \(907\) | \(1,023\) |
| \(setib4\ (15 \times 11)\) | \(905\) | \(905\) | \(1,020\) |
| \(setib4cc\ (15 \times 12)\) | \(905\) | \(905\) | \(1,011\) |
| \(setib4x\ (15 \times 11)\) | \(910\) | \(910\) | \(1,001\) |
| \(setib4xx\ (15 \times 12)\) | \(917\) | \(917\) | \(1,346\) |
| \(setib4xxx\ (15 \times 13)\) | \(917\) | \(917\) | \(1,346\) |
| \(seti5c12\ (15 \times 16)\) | \(1,170\) | \(1,170\) | \(1,319\) |
| \(seti5cc\ (15 \times 17)\) | \(1,136\) | \(1,136\) | \(1,301\) |
| \(seti5x\ (15 \times 16)\) | \(1,199\) | \(1,199\) | \(1,368\) |
| \(seti5xx\ (15 \times 17)\) | \(1,197\) | \(1,197\) | \(1,346\) |
| \(seti5xxx\ (15 \times 18)\) | \(1,136\) | \(1,136\) | \(1,301\) |
| \(seti5xy\ (15 \times 17)\) | \(1,125\) | \(1,125\) | \(1,270\) |

\(^a\)The CPU time on a Xeon E5520 processor with 24 GB of RAM memory in C++.

\(^b\)The CPU time on an Intel 2.0 GHz Core (TM) 2 Duo processor with 8.0 GB of RAM memory in C++.

\(^c\)The CPU time on an Intel Core i3, 2.67 GHz processor with 4 GB RAM in MATLAB R2014a.
BEG-NSGA-II denotes the proposed algorithm. The results of HGTS and HA are adopted from [44, 45], and there are no other multiobjective optimization algorithms to compare for these problems in the up to date literatures. The bolded results are the new multiobjective optimization solution found in our algorithms. From Table 6, we can see that our algorithm can solve these problems with little worse objective \( f_1 \) but simultaneously get the other two objectives \( f_2 \) and \( f_3 \) with less computation time. Figure 15 illustrates the Gantt chart of the problem set i5xyz \((f_1 = 1270, f_2 = 835, \) and \( f_3 = 11,472)\).

4.5. Discussions. From the simulation results of test examples 1–3, we can see that our proposed BEG-NSGA-II could obtain the same or more different Pareto-optimal solutions for the most benchmarks of MFJSP with less computation time. That means more schemes can be chosen by the production managers when they make scheduling decisions with high efficiency. From the simulation results of test example 4, we can see that our proposed algorithm of BEG-NSGA-II works better in multiobjective optimization as well as single-objective optimization.

5. Conclusions and Future Studies

In this paper, in order to fully play the respective advantages of nondominated sorting genetic algorithm II (NSGA-II) algorithm and the algorithms with a two-stage optimization scheme and to overcome the disadvantages of them, we developed a bee evolutionary guiding nondominated sorting genetic algorithm II (BEG-NSGA-II) for solving the multi-objective flexible job-shop scheduling problem with the optimization objectives of minimizing the maximal completion time, the workload of the most loaded machine, and the total workload of all machines. A two-stage optimization mechanism is constructed in the optimization process. In the first stage, the NSGA-II algorithm with \( T \) iteration times is first used to obtain the initial population \( N \) which consists of three parts changing with the iteration times. In this stage, an effective local search operator is invented to extensively exploit the solution space. In the second stage, the NSGA-II algorithm with \( GEN \) iteration times is used to obtain the Pareto-optimal solutions, in which an updating mechanism and some useful genetic operators were employed to enhance the searching ability and avoid the premature convergence. From the simulation results, we can get the conclusions that our proposed algorithm BEG-NSGA-II could obtain more different Pareto-optimal solutions for most benchmarks of MO-FJSP with less computation time. Hence, it could provide more schemes for the production managers to choose when they make scheduling decisions. This proposed computational intelligence method (BEG-NSGA-II) could be widely used in flexible job-shop scheduling problems, especially the multiobjective optimization problems in scheduling filed.

In the future, we will concentrate on the dynamic and real-time scheduling problems which possibly include newly inserted jobs during the production process. Meanwhile, the redistribution of job operations and machine breakdowns may also be taken into consideration.

Notations

- \( n \): Total number of jobs
- \( m \): Total number of machines
- \( n_i \): Total number of operations of job \( i \)
- \( O_{ij} \): The \( j \)th operation of job \( i \)
- \( M_{ij} \): The set of available machines for the operation \( O_{ij} \)
- \( P_{ijk} \): Processing time of \( O_{ij} \) on machine \( k \)
- \( t_{ijk} \): Starting time of operation \( O_{ij} \) on machine \( k \)
- \( C_{ijk} \): Completion time of the operation \( O_{ij} \)
- \( i, h \): Index of jobs, \( i, h = 1, 2, \ldots, n \)
- \( k \): Index of machines, \( k = 1, 2, \ldots, m \)
- \( f_j \): Index of operation sequence, \( f_j = 1, 2, \ldots, n_i \)
- \( C_k \): The completion time of \( M_k \)
- \( W_k \): The workload of \( M_k \)
Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 71473077); National High Technology Research and Development Program of China (863 Program) (Grant no. 2013AA040206); National Key Technology R&D Program of China (2015BAF01B00).

References

[1] J. Blazewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: conventional and new solution techniques," European Journal of Operational Research, vol. 93, no. 1, pp. 1–33, 1996.

[2] E. Pérez, M. Posada, and F. Herrera, "Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling," Journal of Intelligent Manufacturing, vol. 23, no. 3, pp. 341–356, 2012.

[3] X. Qiu and H. Y. K. Lau, "An AIS-based hybrid algorithm for static job shop scheduling problem," Journal of Intelligent Manufacturing, vol. 25, no. 3, pp. 489–503, 2014.

[4] L. Asadzadeh, "A local search genetic algorithm for the job shop scheduling problem with intelligent agents," Computers and Industrial Engineering, vol. 85, article no. 4004, pp. 376–383, 2015.

[5] W.-Y. Ku and J. C. Beck, "Mixed Integer Programming models for job shop scheduling: a computational analysis," Computers and Operations Research, vol. 73, pp. 165–173, 2016.

[6] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," Mathematics of Operations Research, vol. 1, no. 2, pp. 117–129, 1976.

[7] S. Wang, L. Wang, Y. Xu, and M. Liu, "An effective estimation of distribution algorithm for the flexible job-shop scheduling problem with fuzzy processing time," International Journal of Production Research, vol. 51, no. 12, pp. 3778–3793, 2013.

[8] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," Computing, vol. 45, no. 4, pp. 369–375, 1990.

[9] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," Annals of Operations Research, vol. 41, no. 3, pp. 157–183, 1993.

[10] S. Dauzère-Pérès and J. Paulli, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search," Annals of Operations Research, vol. 70, no. 1, pp. 281–306, 1997.

[11] M. Mastrolilli and L. M. Gambardella, "Effective neighbourhood functions for the flexible job shop problem," Journal of Scheduling, vol. 3, no. 1, pp. 3–20, 2000.

[12] J. Gao, M. Gen, and L. Sun, "Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm," Journal of Intelligent Manufacturing, vol. 17, no. 4, pp. 493–507, 2006.

[13] M. Saidi-Mehrabad and F. Fattahi, "Flexible job shop scheduling with tabu search algorithms," The International Journal of Advanced Manufacturing Technology, vol. 32, no. 5–6, pp. 563–570, 2007.

[14] L. Gao, C. Y. Peng, C. Zhou, and P. G. Li, "Solving flexible job shop scheduling problem using general particle swarm optimization," in Proceedings of the 36th CIE Conference on Computers & Industrial Engineering, pp. 3018–3027, Taipei, Taiwan, 2006.

[15] 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.

[16] M. Yazdani, M. Amiri, and M. Zandieh, "Flexible job-shop scheduling with parallel variable neighborhood search algorithm," Expert Systems with Applications, vol. 37, no. 1, pp. 678–687, 2010.

[17] L.-N. Xing, Y.-W. Chen, P. Wang, Q.-S. Zhao, and J. Xiong, "An knowledge-based ant colony optimization for flexible job shop scheduling problems," Applied Soft Computing, vol. 10, no. 3, pp. 888–896, 2010.

[18] L. Wang, G. Zhou, Y. Xu, S. Wang, and M. Liu, "An effective artificial bee colony algorithm for the flexible job-shop scheduling problem," The International Journal of Advanced Manufacturing Technology, vol. 56, no. 1, pp. 1–8, 2012.

[19] K. Z. Gao, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time," International Journal of Production Research, vol. 53, no. 19, pp. 5896–5911, 2015.

[20] M. Rabiee, M. Zandieh, and P. Ramezani, "Bi-objective partial flexible job shop scheduling problem: NSGA-II, NPGA, MOGA and PAES approaches," International Journal of Production Research, vol. 50, no. 24, pp. 7327–7342, 2012.

[21] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications, pp. 93–100, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1985.

[22] B. Jurisch, "Scheduling jobs in shops with multi-purpose machines," in Software Technologies for Embedded and Ubiquitous Systems, pp. 114–125, Springer, Berlin, Germany, 1992.

[23] H. Liu, A. Abraham, O. Choi, and S. H. Moon, "Variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems," in SEAL 2006, vol. 4247 of Lecture Notes in Computer Science, pp. 197–204, Springer, Berlin, Germany, 2006.

[24] J. Gao, M. Gen, L. Sun, and X. Zhao, "A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems," Computers & Industrial Engineering, vol. 53, no. 1, pp. 149–162, 2007.

[25] G. Zhang, X. Shao, P. Li, and L. Gao, "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem," Computers & Industrial Engineering, vol. 56, no. 4, pp. 1309–1318, 2009.

[26] L.-N. Xing, Y.-W. Chen, and K.-W. Yang, "An efficient search method for multi-objective flexible job shop scheduling problems," Journal of Intelligent Manufacturing, vol. 20, no. 3, pp. 283–293, 2009.

[27] I. González-Rodríguez, C. R. Vela, and J. Puente, "A genetic solution based on lexicographical goal programming for a multiobjective job shop with uncertainty," Journal of Intelligent Manufacturing, vol. 21, no. 1, pp. 65–73, 2010.

[28] S. H. A. Rahmati, M. Zandieh, and M. Yazdani, "Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem," The International Journal of Advanced Manufacturing Technology, vol. 64, no. 5–8, pp. 915–932, 2013.
[29] Y. Yuan and H. Xu, "Multiobjective flexible job shop scheduling using memetic algorithms," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 336–353, 2015.

[30] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[31] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic," *Mathematics and Computers in Simulation*, vol. 60, no. 3–5, pp. 245–276, 2002.

[32] N. B. Ho and J. C. Tay, "Solving multiple-objective flexible job shop problems by evolution and local search," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 38, no. 5, pp. 674–685, 2008.

[33] 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.

[34] X. Wang, L. Gao, C. Zhang, and X. Shao, "A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 51, no. 5–8, pp. 757–767, 2010.

[35] M. Frutos, A. C. Olivera, and F. Tohmé, "A memetic algorithm based on a NSGAII scheme for the flexible job-shop scheduling problem," *Annals of Operations Research*, vol. 181, pp. 745–765, 2010.

[36] L. Wang, S. Wang, and M. Liu, "A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem," *International Journal of Production Research*, vol. 51, no. 12, pp. 3574–3592, 2013.

[37] M. Rohaninejad, A. Kheirkhah, P. Fattahi, and B. Vahedi-Nouri, "A hybrid multi-objective genetic algorithm based on the ELECTRE method for a capacitated flexible job shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 77, no. 1–4, pp. 51–66, 2015.

[38] V. Kaplanoğlu, "An object-oriented approach for multi-objective flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 45, pp. 71–84, 2016.

[39] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Transactions on Systems, Man & Cybernetics Part C: Applications & Reviews*, vol. 32, no. 1, pp. 1–13, 2002.

[40] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation," *Computers & Industrial Engineering*, vol. 30, no. 4, pp. 983–997, 1996.

[41] W. Xia and Z. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems," *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 409–425, 2005.

[42] X. Shao, W. Liu, Q. Liu, and C. Zhang, "Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 67, no. 9–12, pp. 2885–2901, 2013.

[43] J. W. Barnes and J. B. Chambers, "Flexible job shop scheduling by tabu search, Graduate program in operations research and industrial engineering," Tech. Rep., University of Texas, Austin, Tex, USA, 1996.