Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 5958640, 13 pages
http://dx.doi.org/10.1155/2016/5958640

Research Article

An Improved Version of Discrete Particle Swarm Optimization for Flexible Job Shop Scheduling Problem with Fuzzy Processing Time

Song Huang,[1] Na Tian,[1] Yan Wang,[1] and Zhicheng Ji[1,2]

[1]*School of Internet of Things Engineering, Jiangnan University, Wuxi, China*
[2]*Engineering Research Center of Internet of Things Technology Applications, Ministry of Education, Jiangnan University, Wuxi 214122, China*

Correspondence should be addressed to Yan Wang; wangyan88@jiangnan.edu.cn

Received 11 May 2016; Revised 12 September 2016; Accepted 22 September 2016

Academic Editor: Erik Cuevas

The fuzzy processing time occasionally exists in job shop scheduling problem of flexible manufacturing system. To deal with fuzzy processing time, fuzzy flexible job shop model was established in several papers and has attracted numerous researchers' attention recently. In our research, an improved version of discrete particle swarm optimization (IDPSO) is designed to solve flexible job shop scheduling problem with fuzzy processing time (FJSPF). In IDPSO, heuristic initial methods based on triangular fuzzy number are developed, and a combination of six initial methods is applied to initialize machine assignment and random method is used to initialize operation sequence. Then, some simple and effective discrete operators are employed to update particle's position and generate new particles. In order to guide the particles effectively, we extend global best position to a set with several global best positions. Finally, experiments are designed to investigate the impact of four parameters in IDPSO by Taguchi method, and IDPSO is tested on five instances and compared with some state-of-the-art algorithms. The experimental results show that the proposed algorithm can obtain better solutions for FJSPF and is more competitive than the compared algorithms.

1. Introduction

Flexible job shop scheduling problem (FJSP) is a general form of job shop scheduling problem (JSP), which is completely NP-hard problem [1]. Up to date, many different mathematical methods and techniques, such as mixed integer programming [2], disjunctive graph [3], priority dispatch rules [4, 5], and neural networks [6], are developed to optimize FJSP. Nature-inspired algorithms have made great progress in solving optimization problems in the past two decades [7, 8] and are extended to solve a wide range of applications such as scheduling optimization problems [9–12], transportation problems [13, 14], traveling salesman problems [15–17], and engineering optimization design [18]. Due to their validity and robustness, various nature-inspired algorithms (particle swarm optimization (PSO) [19], differential evolution (DE) [20], firefly algorithm (FA) [21], artificial bee colony (ABC) [22, 23], harmony search (HS) [24, 25], evolutionary algorithm (EA) [26], and biogeography-based optimization (BBO) [27]) also attract a large number of researches' attention on the application of FJSP. By defining the fitness of particles using Pareto ranking and crowding distance, Shao et al. presented hybrid discrete algorithms for addressing FJSP, which had a global search procedure of particle swarm optimization and a local search procedure of simulated annealing [19]. Utilizing differential evolution and local search, Yuan and Xu presented a combined differential evolution (HDE) and speed-up local search based on critical path to improve the efficiency [20]. Recently, firefly algorithm attracted many researches' attention and Karthikeyan et al. proposed a hybrid discrete firefly algorithm (HDFA) to optimize the FJSP [21]. In HDFA, some discrete transformations of attractiveness, distance, and movement were developed and local search was applied to improve the accuracy of the solutions of FJSP. Focusing on scheduling problems with maintenance activities, Li et al. proposed discrete artificial bee colony

(DABC) [22]. In DABC, an effective decoding method was studied to decode the representation of FJSP with maintenance activities and food sources of employed bees, onlooker bees, and scout bees were produced by Tabu search. Wang et al. presented an artificial bee colony to minimize the makespan [23]. A population updating mechanism for generating the scout bees and local search strategy based on critical path for onlooker bees were employed in ABC to achieve good performance. To minimize the makespan and the mean of earliness and tardiness, a discrete harmony search (DHS) was developed by Gao et al. [24]. In this algorithm, a discrete function to generate new harmonies was designed and some local search approaches were used to strengthen exploitation ability. To minimize makespan, Yuan et al. developed a hybrid harmony search (HHS), which employed local search process to perform exploitation [25]. In HHS, a new decoding method was applied to reduce the search space and neighborhood structure of local search was based on common critical operations.

In most job shop models, processing time is assumed as a deterministic value. However, in uncertain environment, processing time is frequently acquired in a certain range. Therefore, flexible job shop scheduling model with fuzzy processing time (FJSPF) is a closer approximation to the real manufacturing system with uncertain situation and the model of FJSPF was built in some researches to address this situation. As a FJSP model with fuzzy processing time, FJSPF is also highly difficult to solve and several intelligent algorithms based on fuzzy number have been developed to optimize it. Lei presented a decomposition-integration genetic algorithm (DIGA), which provided a two-string representation and decomposed the chromosome into two parts (job sequencing part and machine assigning part), and the two parts of the population evolved independently in DIGA [28]. To minimize the fuzzy makespan, Lei also proposed a coevolutionary genetic algorithm (CGA), which designed a novel representation, crossover operator, and a modified tournament selection. In CGA, job sequencing and machine assignment evolved independently and then cooperated to determine the scheduling scheme [29]. To solve the FJSPF, an estimation of distribution algorithm (EDA) based on probability model was proposed by Wang et al. [30]. In EDA, probability matrix was firstly generated and then updated with the elites. Moreover, roulette selection was used to produce new individuals.

In this paper, an improved version of PSO (called IDPSO), which consists of discrete operators, is introduced to solve FJSPF. First, triangular fuzzy number (TFN) is used to express fuzzy processing time and several initial methods based on TFN are developed in initialization phase. Then six initial methods are applied to initialize machine assignment and random method is used to initialize operation sequence. Then simple discrete operators are embedded in IDPSO to update the particles' positions. To guide particles by the personal best positions and the global best position in IDPSO, discrete operator f_2 is applied to update the current particles' positions using the personal best positions or the global best position. Moreover, the global best position is extended to contain several global best positions in the predefined global best set and that can make the guidance more effective.

The remainder of this paper is organized as follows: Section 2 introduces the problem statement of FJSPF. Section 3 describes the implementation of IDPSO for FJSPF. The simulation results and analysis of the proposed algorithm are provided in Section 4. A conclusion is finally given in Section 5.

2. FJSPF Model

2.1. Problem Statement. Suppose that the system consists of n jobs (denoted by $J = \{J_i \mid i = 1, 2, \ldots, n\}$) and m machines (denoted by $M = \{M_k \mid k = 1, 2, \ldots, m\}$). Job i has n_i operations (denoted by $O_i = \{O_{ij} \mid j = 1, 2, \ldots, n_i\}$). Each operation can be processed on a set of machines (denoted by M_s) and processing time is also given. Therefore, two types of FJSP, partial flexibility and total flexibility, are defined as follows:

$$\text{FJSP} \begin{cases} \text{Partial Flexibility} & \text{IF } M_s \in M \\ \text{Total Flexibility} & \text{IF } M_s = M. \end{cases} \tag{1}$$

For FJSPF model, processing time on any machine is fuzzy and it can be called fuzzy processing time. Generally, fuzzy processing time can be treated as TFN and is expressed as follows:

$$s = \left(s^1, s^2, s^3\right), \tag{2}$$

where s^1, s^2, and s^3 are the minimal probable processing time, the most probable processing time, and the maximal probable processing time. The optimization task is to determine an assignment on machines and a sequence of operations to optimize one or more criteria. The maximal completion time is a significantly important factor to impact due date. Therefore, the criterion is to minimize the maximal fuzzy completion time in this paper and its expression is as follows:

$$C_M = \min\left\{\max_{1 \le k \le m} C_k\right\}, \tag{3}$$

where C_k is a TFN and means the fuzzy completion time on machine k.

2.2. Operations on Triangular Fuzzy Number. Some operations for real number are hardly applied to TFN since TFN includes three values, and then special operations should be designed to operate TFNs. Therefore, researchers define three essential operations (addition operation, max operation, and rank operation) to dispose TFNs [31]. Specifically, two TFNs are added by addition operation; max operation is required to obtain the maximal TFN and rank operation is applied to sort TFNs. Max operation and rank operation need to compare two TFNs. In order to compare TFNs, three criteria should be defined through a TFN $s = (s^1, s^2, s^3)$ as follows:

Criterion 1 ($c1$):

$$c1\,(s) = \frac{\left(s^1 + 2s^2 + s^3\right)}{4} \tag{4}$$

```
(1) Input: s, t
(2) Output: The relationship of s and t
(3) begin
(4)     if c1(s) ≠ c1(t)
(5)                 if c1(s) < c1(t)
                                s < t
                    else
                                s > t
(6)                 end
(7)     else if c2(s) ≠ c2(t)
(8)                 if c2(s) < c2(t)
                                s < t
                    else
                                s > t
(9)                 end
        else if c3(s) ≠ c3(t)
(10)                if c3(s) < c3(t)
                                s < t
                    else
                                s > t
(11)                end
        else
                s = t
(12)    end
(13) end
```

Pseudocode 1: Pseudocode of comparing TFNs.

Criterion 2 ($c2$):

$$c2(s) = s^2 \tag{5}$$

Criterion 3 ($c3$):

$$c3(s) = s^3 - s^1 \tag{6}$$

According to the above three criteria, we can use two TFNs, $s = (s^1, s^2, s^3)$ and $t = (t^1, t^2, t^3)$, to demonstrate the comparing procedure as shown in Pseudocode 1.

For instance, two TFNs, $s = (3, 5, 6)$ and $t = (2, 5, 7)$, are used to illustrate the comparing procedure. First, Criterion 1 should be calculated and then we have $c1(s) = c1(t) = 4.75$. Since Criterion 1 values of two TFNs are equal to each other, we should calculate Criterion 2 and then we have $c2(s) = c2(t) = 5$. As Criterion 2 values of two TFNs are also the same, Criterion 3 should be calculated and we have $c3(s) = 3$ and $c3(t) = 5$. Due to $c3(s) < c3(t)$, thus $s < t$. According to the comparing procedure of two TFNs, the procedure of max operation and rank operation can be obtained easily. Besides, for two TFNs, $s = (s^1, s^2, s^3)$ and $t = (t^1, t^2, t^3)$, the addition operation is calculated by the following formula:

$$s + t = \left(s^1 + t^1, s^2 + t^2, s^3 + t^3\right). \tag{7}$$

3. IDPSO for FJSPF

3.1. Theory of Particle Swarm Optimization. Particle swarm optimization (PSO) is an intelligent algorithm and employs a population with several particles to search the optimal solution. Each particle has three attributes: velocity, position, and personal best position, and the population has a global best position. For a specific issue, suppose n particles are in m dimensional space. Position represents a potential solution of the issue and velocity represents the move of particle. Velocity and position of particle i can be denoted by $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{im})$ and $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})$. Personal best position (denoted by $\mathbf{p}_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,m})$) and global best position (denoted by $\mathbf{g}_{\text{best}} = (g_{\text{best},1}, g_{\text{best},2}, \ldots, g_{\text{best},m})$) represent the best position obtained by particle i and the best position obtained by all particles. The velocity $\mathbf{v}_i^{t+1}$ and the position $\mathbf{x}_i^{t+1}$ can be manipulated according to the following:

$$\begin{aligned} \mathbf{v}_i^{t+1} &= \omega \mathbf{v}_i^t + c_1 r_1 \left(\mathbf{p}_i^t - \mathbf{x}_i^t\right) + c_2 r_2 \left(\mathbf{g}_{\text{best}}^t - \mathbf{x}_i^t\right), \\ \mathbf{x}_i^{t+1} &= \mathbf{x}_i^t + \mathbf{v}_i^{t+1}, \end{aligned} \tag{8}$$

where c_1 and c_2 are cognitive coefficient and social coefficient. ω is inertia weight. r_1 and r_2 are random numbers in (0, 1). t is current iteration number. The particles will obtain new promising positions until a satisfactory solution is found or the maximal iteration number is met.

3.2. Detail of IDPSO. PSO was firstly introduced by Kennedy and Eberhart in 1995 and many variants have strong abilities to solve nonlinear and continuous optimization problems [32]. FJSPF is a class of discrete optimization problem, which has many discrete variables that need to operate. Therefore, an improved PSO with several discrete operators is introduced in discrete PSO version. According to the mechanism of PSO, personal best positions and global best position are the guiders of the population and they make all particles move to global optimum. Therefore, the position $\mathbf{x}_i^{t+1}$ of IDPSO can be manipulated according to the following:

$$\begin{aligned} \mathbf{x}_i^{t+1} = \omega \otimes f_1 \left(\mathbf{x}_i^t\right) + c_1 \otimes f_2 \left(\mathbf{x}_i^t, \mathbf{p}_i^t\right) + c_2 \\ \otimes f_2 \left(\mathbf{x}_i^t, \mathbf{g}_{\text{best}}^t\right), \end{aligned} \tag{9}$$

where $\otimes$ is probability operation, indicating that the following operator carries on with the corresponding probability. + is an operation, which indicates that the current operator is completed and the next operator is carrying on. ω, c_1, and c_2 are the probabilities and f_1 and f_2 are two discrete operators to operate discrete variables. In detail, c_2 equals $1 - c_1$ in our algorithm. f_2 is firstly applied to the current position $\mathbf{x}_i^t$ and the personal best position $\mathbf{p}_i^t$. Then, f_2 is applied to the current position $\mathbf{x}_i^t$ and the global best position $\mathbf{g}_{\text{best}}^t$. This makes the current particle guided by personal best position and global best position in IDPSO. In addition, the details of f_1 and f_2 are presented in Section 3.5. Perturbation operator f_3 is then adopted and will also be introduced in Section 3.5.

Two positions, personal best position $\mathbf{p}_i$ and global best position $\mathbf{g}_{\text{best}}$, are the key factors to affect particles'

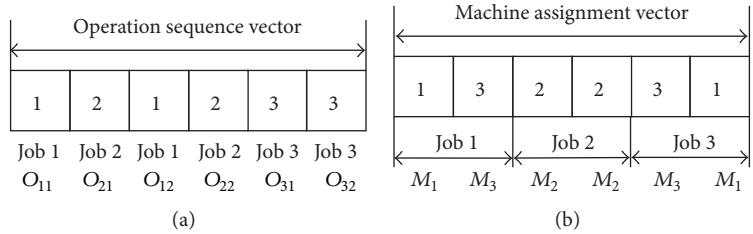

Figure 1: Two-vector representation.

convergence curve. According to PSO, the personal best position of particle i is also updated as follows:

$$\mathbf{p}_i = \begin{cases} \mathbf{x}_i & \text{IF } C_M(\mathbf{x}_i) < C_M(\mathbf{p}_i) \\ \mathbf{p}_i & \text{otherwise.} \end{cases} \tag{10}$$

In order to avoid the weakness of PSO being trapped into local optima, the global best position is extended to a predefined global best set which contains several global best positions in IDPSO. Suppose global best set has Na global best positions. Then the global best set updates itself as follows: add current population and global best set to form a new set, and then find Na best positions from new set to the next global best set. Finally, global best position $\mathbf{g}_{\text{best}}$ is randomly selected from global best set.

3.3. Encoding and Decoding. Two-vector representation (operation sequence vector and machine assignment vector) regularly represents operation sequence and machine assignment. Numerous researches adopted operation-based representation to represent the operation sequence vector since it is simple and no illegal representation needs to be repaired. For the operation-based representation, the length of operation sequence vector equals the total number of operations. Each operation is denoted by job number and the ith appearance of job number denotes the ith operation of the corresponding job. For machine assignment vector, assign the machines to all operations with ascending order of job number. One example of two-vector representation is illustrated in Figure 1. For instance, the sequence [1 2 1 2 3 3] in Figure 1(a) represents the operation sequence $[O_{11}, O_{21}, O_{12}, O_{22}, O_{31}, O_{32}]$, and the sequence [1 3 2 2 3 1] in Figure 1(b) represents the machine assignments (O_{11}, M_1), (O_{12}, M_3), (O_{21}, M_2), (O_{22}, M_2), (O_{31}, M_3), and (O_{32}, M_1). Table 1 shows the processing times of 3 jobs and 3 machines' instance. Figure 2 shows the Gantt chart of this representation scheme.

3.4. Initialization. In IDPSO, operation sequence vector is initialized randomly and several initial methods based on TFNs are developed to initialize machine assignment. It is obvious that initial methods from FJSP cannot be directly applied to FJSPF due to the particularity of TFNs. For machine assignment, we deal with the minimal probable processing time, the most probable processing time, and the maximal probable processing time of TFN separately and

Table 1: Processing times for 3 jobs and 3 machines instance.

		Processing time		
		M_1	M_2	M_3
J_1	O_{11}	(1, 4, 5)	(2, 3, 4)	(4, 5, 6)
	O_{12}	(3, 5, 6)	(1, 2, 5)	(4, 6, 7)
J_2	O_{21}	(2, 4, 5)	(2, 3, 6)	(1, 2, 5)
	O_{22}	(1, 3, 5)	(1, 2, 4)	(2, 4, 5)
J_3	O_{31}	(3, 4, 5)	(5, 6, 7)	(2, 4, 7)
	O_{32}	(2, 3, 5)	(1, 3, 4)	(1, 4, 5)

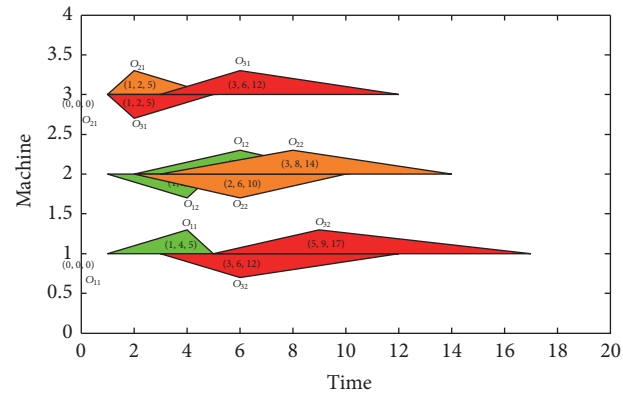

Figure 2: Gantt chart of one solution of 3 jobs and 3 machines.

then new initial methods (Rules 1, 2, and 3) are formed. These rules can decompose a TFN into three processing times and find a promising machine assignment for each processing time separately.

Rule 1. Select all minimal probable processing times of TFNs in fuzzy processing timetable to form a new processing timetable. In the new table, find the minimal processing time for each operation, and assign the corresponding machine to the operation. Repeat the procedure until all operations are assigned.

For instance, the first element "1" of O_{11} in Table 2 is selected from the minimal probable processing time of the first element (1, 4, 5) of O_{11} in Table 1, and the second element "2" of O_{11} in Table 2 is selected from the minimal probable processing time of the second element (2, 3, 4) of O_{11} in Table 1. Therefore, Table 2 is formed by selecting all the minimal probable processing times of Table 1. And then the machine assignment is obtained by Rule 1 and is shown

Table 2: Minimal probable processing times for 3 jobs and 3 machines instance.

		Processing time		
		M_1	M_2	M_3
J_1	O_{11}	1	2	4
	O_{12}	3	1	4
J_2	O_{21}	2	2	1
	O_{22}	1	1	2
J_3	O_{31}	3	5	2
	O_{32}	2	1	1

Table 3: Machine assignment for 3 jobs and 3 machines instance.

		Machine assignment		
		M_1	M_2	M_3
J_1	O_{11}	1	0	0
	O_{12}	0	1	0
J_2	O_{21}	0	0	1
	O_{22}	1	0	0
J_3	O_{31}	0	0	1
	O_{32}	0	1	0

in Table 3. In Table 3, the number "1" represents the machine assigned on the corresponding operation.

Rule 2. The procedures of Rules 2 and 3 are the same as that of Rule 1. The difference is that Rule 2 selects the most probable processing times of TFNs to form a new processing timetable and Rule 3 selects the maximal probable processing times of TFNs to form a new processing timetable.

Three existing rules, global minimal processing time rule (Rule 4) [33], local minimal processing time rule (Rule 5) [34], and earliest completion machine (Rule 6) [35], were proved to be effective initial methods in FJSP and are also modified to initialize the population in this paper. Finally, empirical combination of the above six heuristic methods (10% by Rule 1, 10% by Rule 2, 10% by Rule 3, 30% by Rule 4, 30% by Rule 5, and 10% by Rule 6) is used to initialize machine assignment vector of the population. The details of the three existing rules are as follows.

Rule 4. Firstly, find the machine with the minimal TFN in the processing timetable, and assign the operation on this machine. Then, add the assigned processing time to other processing times in the same column and delete the row in which the operation is assigned. Repeat the procedure until all operations are assigned.

Rule 5. For each job, find the machine with the minimal TFN for every operation O_{ij} of the job, and assign the corresponding operation on this machine. Then, add the assigned time to other processing times in the same column and delete the row in which the operation is assigned. Repeat the procedure until all operations are assigned.

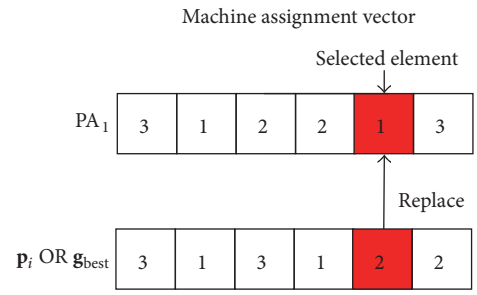

Figure 3: Detail of f_2.

Rule 6. For each operation in order of operation sequence vector, assign the operation to the machine that can complete the operation with the earliest fuzzy completion time.

3.5. Designed Operators. According to formula (9), the function of f_1 is to make the particles unchanged and ω is the unchanged probability. Otherwise, the particles are the changed particles. In IDPSO, these unchanged particles are only operated by f_2 and the other particles are only operated by perturbation operator f_3. The function of f_2 is to exchange the information from personal best positions and global best position with probabilities c_1 and c_2, respectively. Since many promising particles are produced in the initial phase, exchanging multipoints information can significantly change the scheduling of FJSPF and may destroy the useful information of promising particles. Consequently, exchanging single-point information from the best positions contributes to the reservation of particles' self-information and acquiring helpful information from the best positions. Therefore, f_2 adopts simple and effective single-point information exchanging method. In addition, f_2 is used to the current particle's position and its personal best position with the probability c_1. Then, f_2 is used to the current particle's position and the global best position with the probability c_2. It is worth pointing out that f_2 is only implemented on machine assignment vector.

f_2 works as follows. For the particle PA_1, keep the operation sequence unchanged. Then randomly select an element in machine assignment vector of PA_1, and replace this element with the element of the personal best position $\mathbf{p}_i$ or the global best position $\mathbf{g}_{\mathbf{best}}$ in the same place. How f_2 works is illustrated in Figure 3.

f_2 with single-point information interchange has a weak exploration ability. To avoid this weakness, perturbation operator f_3 is used to explore other space and can make the algorithm have a good ability to balance exploitation and exploration. Perturbation operation f_3 can be executed as follows.

Method 1 and Method 2 should be defined in perturbation operator f_3. For the particle PA_1, keep machine assignment unchanged. Randomly select an operation O_{ij} in operation sequence vector and record its place pl in operation sequence vector of PA_1, and then find the place pl$'$ where the operation O_{ij} locates in operation sequence vector of PA_2. Finally,

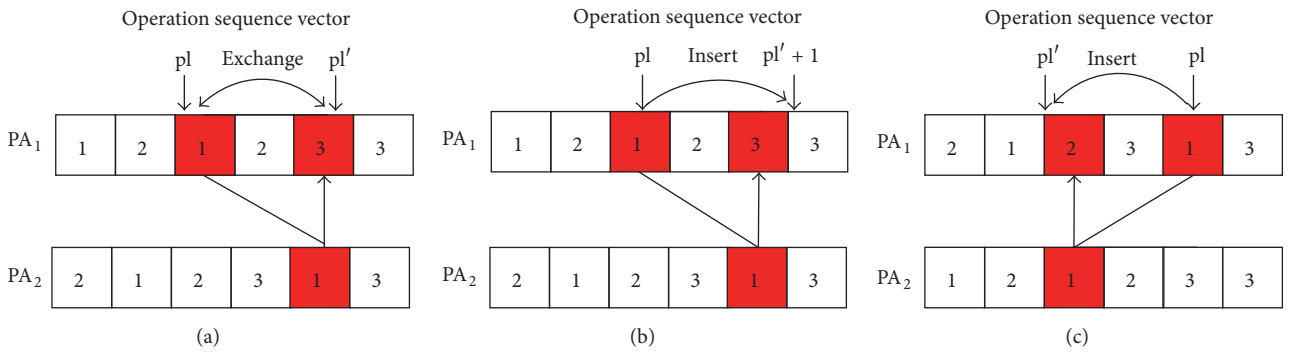

Figure 4: Detail of f_3.

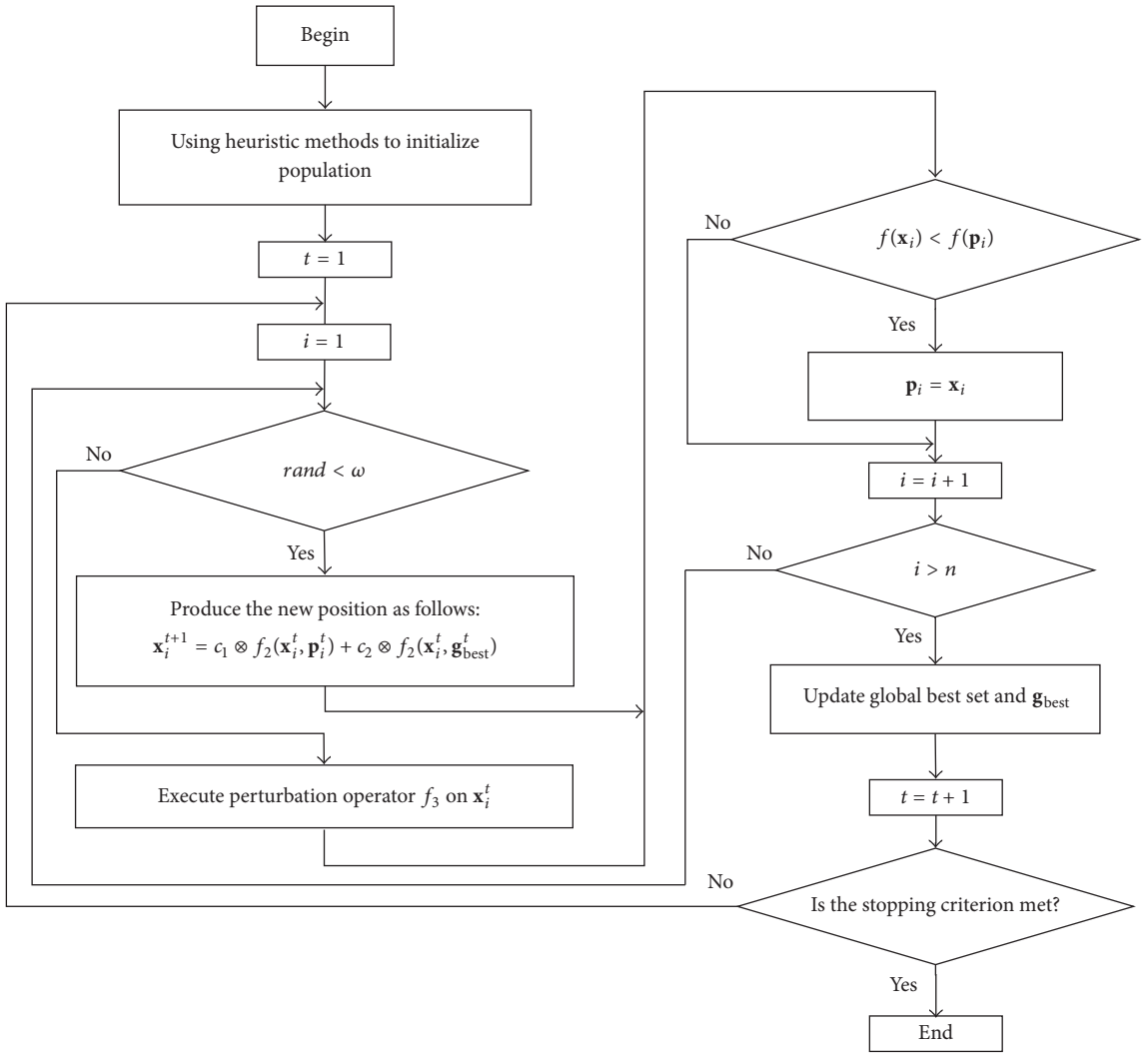

Figure 5: Flowchart of IDPSO algorithm.

Method 1 is exchanging the operation O_{ij} with the operation in the place pl' of operation sequence vector of particle PA_1. Method 2 is inserting the operation O_{ij} into the place $\text{pl}' + 1$, if $\text{pl} < \text{pl}'$, or inserting the operation O_{ij} into the place pl', if $\text{pl}' < \text{pl}$. The operation of Method 1 is illustrated in Figure 4(a) and Method 2 is illustrated in Figures 4(b)-4(c). f_3 works as follows. If rand < 0.5, f_3 will execute Method 1; otherwise, f_3 will execute Method 2.

From the above description, the framework of IDPSO is shown in Figure 5.

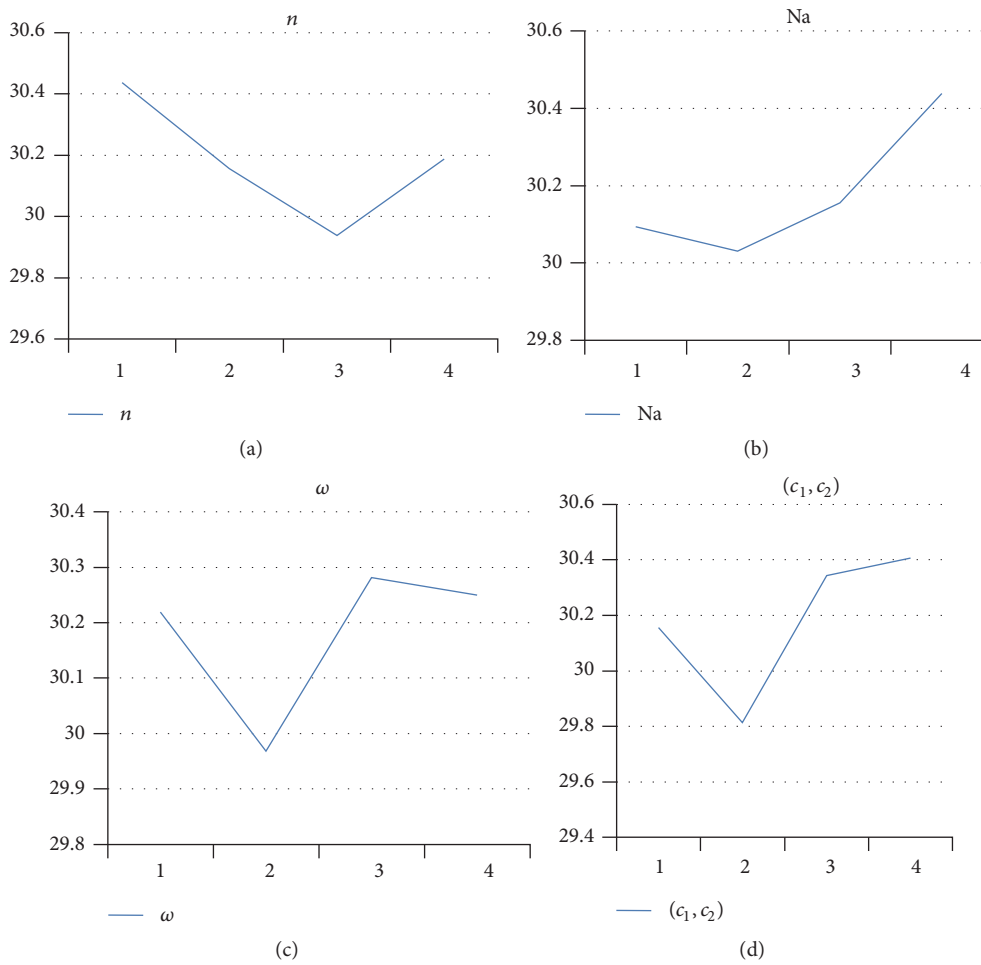

Figure 6: Trend of factor levels for each parameter.

Table 4: Value ranges of four factor parameters.

Parameters	n	Na	ω	(c_1, c_2)
Value range	50~200	1~15	0.90~0.98	0.2~0.8

4. Experiments and Results

4.1. Parameter Setting. In this section, the impacts of four parameters in our algorithm are investigated and some experiments are conducted using Taguchi method. The experiment design involves four parameters: n, ω, (c_1, c_2), and Na, and each parameter is divided into four levels. Value ranges of four parameters are listed in Table 4. According to the experiment design of four factors and four levels, experiments of an orthogonal array (4^4) are obtained. The factor levels of each parameter are listed in Table 5, and the orthogonal experiments are listed in Table 6. According to the orthogonal table, the designed experiments are conducted on Instance 1 [28] and each designed experiment runs 20 times independently. The maximal generation number is 1000. After the experiments, the results of $c1$ for each designed experiment are also listed in the last column of Table 6.

Table 5: Factor levels of parameters.

Parameters	Factor level			
	1	2	3	4
n	50	100	150	200
Na	1	5	10	15
ω	0.9	0.94	0.96	0.98
(c_1, c_2)	(0.2, 0.8)	(0.4, 0.6)	(0.6, 0.4)	(0.8, 0.2)

According to the results of $c1$ in Table 6, we can obtain the average value of $c1$ (AVC1) for each factor level of each parameter in Table 7. For each parameter, the effect on performance is analyzed by "Rank" in Table 7. In Table 7, "Delta" denotes the maximal AVC1 minus the minimal AVC1 for each parameter, and "Rank" denotes the rank of "Delta." "Rank" reflects the significance of each parameter. For comparisons, (c_1, c_2) rank first, and n rank second. Therefore, the parameters (c_1, c_2) are the most significant factor on the performance of our algorithm. According to Table 7, the trends of each factor level are illustrated in Figures 6(a)–6(d). In Figure 6, it is obvious that parameter combinations of the proposed

Table 6: Orthogonal table of designed experiments.

Experiment number	Factor level				$c1$
	n	Na	ω	(c_1, c_2)	
1	1	1	3	4	31
2	1	2	4	3	30.25
3	1	3	2	1	30.25
4	1	4	1	2	30.25
5	2	1	2	4	29.875
6	2	2	3	1	30.25
7	2	3	4	2	29.875
8	2	4	1	3	30.625
9	3	1	1	1	29.75
10	3	2	2	2	29.375
11	3	3	3	3	30.125
12	3	4	4	4	30.5
13	4	1	3	2	29.75
14	4	2	1	4	30.25
15	4	3	4	1	30.375
16	4	4	2	3	30.375

Table 7: Rank of each parameter by $c1$.

Factor	n	Na	ω	(c_1, c_2)
1	30.4375	30.09375	30.21875	30.15625
2	30.15625	30.03125	29.96875	29.8125
3	29.9375	30.15625	30.28125	30.34375
4	30.1875	30.4375	30.25	30.40625
Delta	0.5	0.40625	0.3125	0.59375
Rank	2	3	4	1

algorithm contribute to the performance of FJSPF and we can choose appropriate parameters from Figures 6(a)–6(d).

4.2. Numerical Computation and Comparison. Five instances ranging from 10 jobs-40 operations to 15 jobs-80 operations (Instance 1–Instance 5) [28, 29] are considered to evaluate the proposed algorithm and IDPSO is compared with some state-of-the-art algorithms. The test is implemented in Matlab 7.1 on Lenovo computer with an Intel 2.4 G processor and 4 G RAM. The proposed algorithm will run 30 times on each instance. The parameters are selected as suggested in Section 4.1. The suitable parameter settings of the proposed algorithm are summarized as follows: the population size is 100; the size of global best set is 10; the maximal generation number is 1000; the values of ω, c_1, and c_2 are 0.94, 0.4, and 0.6, respectively; the compared algorithms are DIGA [28], CGA [29], EDA [30], PEGA [36], PSO+SA [37], and HABC [38] and their results on the five instances are from the corresponding literature.

As shown in Table 8, it lists the average value (Average value), the best solution (Best solution), and the worst solution (Worst solution) obtained by seven algorithms. For Instances 1–4, the proposed algorithm outperforms all other six algorithms in terms of "Average value," "Best solution," and "Worst solution." For Instance 5, the results of "Average value," "Best solution," and "Worst solution" obtained by the proposed algorithm rank second among these compared algorithms. Besides, the average CPU time of 20 runs is listed in Table 9. For Instances 1–4, the proposed algorithm spends the second shortest average CPU time among all seven algorithms. For Instance 5, the proposed algorithm costs the shortest average CPU time. In summary, for all five instances, the proposed algorithm is the most effective algorithm among these compared algorithms. In addition, the Gantt charts of best solution obtained by the proposed algorithm are illustrated in Figures 7(a)–7(e).

Wilcoxon rank sum test can confirm whether one data set is superior to another data set in statistics and two values (p value, h value) can be acquired by the test to assess the

Table 8: Results of the five instances.

Instance	Algorithm	Average value	Best solution	Worst solution
Instance 1	IDPSO	**(20.6, 29.87, 40.17)**	**(19, 28, 40)**	**(21, 31, 41)**
	EDA	(20.3, 30.5, 41.6)	(20, 28, 40)	(22, 32, 43)
	HABC	(21.0, 32.0, 43.6)	(19, 30, 43)	(23, 33, 46)
	CGA	(23.1, 33.1, 43.4)	(21, 29, 41)	(25, 37, 47)
	DIGA	(22.8, 33.4, 44.6)	(20, 31, 40)	(25, 37, 49)
	PEGA	(25.3, 35.7, 47.8)	(22, 33, 42)	(28, 41, 55)
	PSO+SA	(26.1, 37.4, 48.2)	(25, 32, 40)	(32, 43, 55)
Instance 2	IDPSO	**(32.47, 46.17, 57.53)**	**(30, 46, 58)**	**(35, 47, 57)**
	EDA	(33.7, 46.9, 57.9)	(32, 46, 57)	(34, 48, 58)
	HABC	(33.0, 47.8, 62.2)	(33, 46, 58)	(36, 48, 65)
	CGA	(35.0, 47.1, 60.6)	(32, 47, 57)	(38, 49, 64)
	DIGA	(35.4, 48.4, 62.3)	(33, 48, 57)	(37, 50, 65)
	PEGA	(37.5, 51.8, 66.8)	(38, 48, 61)	(43, 58, 73)
	PSO+SA	(36.8, 51.6, 65.5)	(38, 49, 61)	(44, 60, 77)
Instance 3	IDPSO	**(32.47, 46.20, 60.90)**	**(33, 45, 60)**	**(35, 49, 64)**
	EDA	(32.8, 47.2, 62.9)	(31, 46, 60)	(34, 49, 66)
	HABC	(33.9, 50.8, 67.3)	(33, 47, 64)	(36, 54, 70)
	CGA	(36.4, 50.8, 66.0)	(34, 47, 63)	(38, 53, 71)
	DIGA	(37.3, 53.0, 66.9)	(37, 49, 64)	(41, 58, 75)
	PEGA	(40.6, 56.4, 73.3)	(36, 51, 68)	(40, 59, 77)
	PSO+SA	(40.9, 57.4, 74.5)	(38, 51, 65)	(42, 60, 79)
Instance 4	IDPSO	**(24.90, 36.67, 51.00)**	**(23, 35, 49)**	**(27, 38, 51)**
	EDA	(24.8, 37.2, 51.9)	(21, 36, 50)	(24, 39, 57)
	HABC	(25.5, 40.0, 56.3)	(23, 38, 53)	(25, 44, 59)
	CGA	(27.4, 40.4, 55.0)	(26, 37, 51)	(29, 42, 59)
	DIGA	(29.2, 42.9, 57.5)	(29, 41, 56)	(29, 46, 60)
	PEGA	(35.2, 50.1, 66.4)	(34, 48, 61)	(39, 55, 73)
	PSO+SA	(35.8, 48.9, 65.3)	(33, 48, 64)	(35, 52, 72)
Instance 5	IDPSO	(40.60, 58.03, 78.97)	(35, 56, 81)	(43, 60, 82)
	EDA	**(38.6, 56.9, 78.3)**	**(36, 55, 73)**	**(40, 60, 81)**
	HABC	—	—	—
	CGA	(47.0, 65.4, 86.0)	(42, 62, 82)	(49, 70, 91)
	DIGA	(45.8, 66.3, 88.7)	(42, 63, 84)	(49, 71, 92)
	PEGA	(50.3, 74.0, 96.5)	(48, 68, 94)	(50, 74, 100)
	PSO+SA	(51.2, 74.6, 97.6)	(48, 72, 93)	(52, 73, 101)

Table 9: CPU times of the five instances.

	Time(s)				
	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
IDPSO	4.53	4.52	5.20	5.19	**7.40**
EDA	**3.65**	**3.63**	**4.86**	**4.56**	9.83
ABC	9.87	10.88	14.80	13.85	—
CGA	8.29	8.26	10.66	10.77	23.87
DIGA	15.36	15.57	18.87	19.02	37.82
PEGA	12.56	12.67	15.23	15.71	30.15
PSO+SA	12.40	12.33	15.24	15.66	30.90

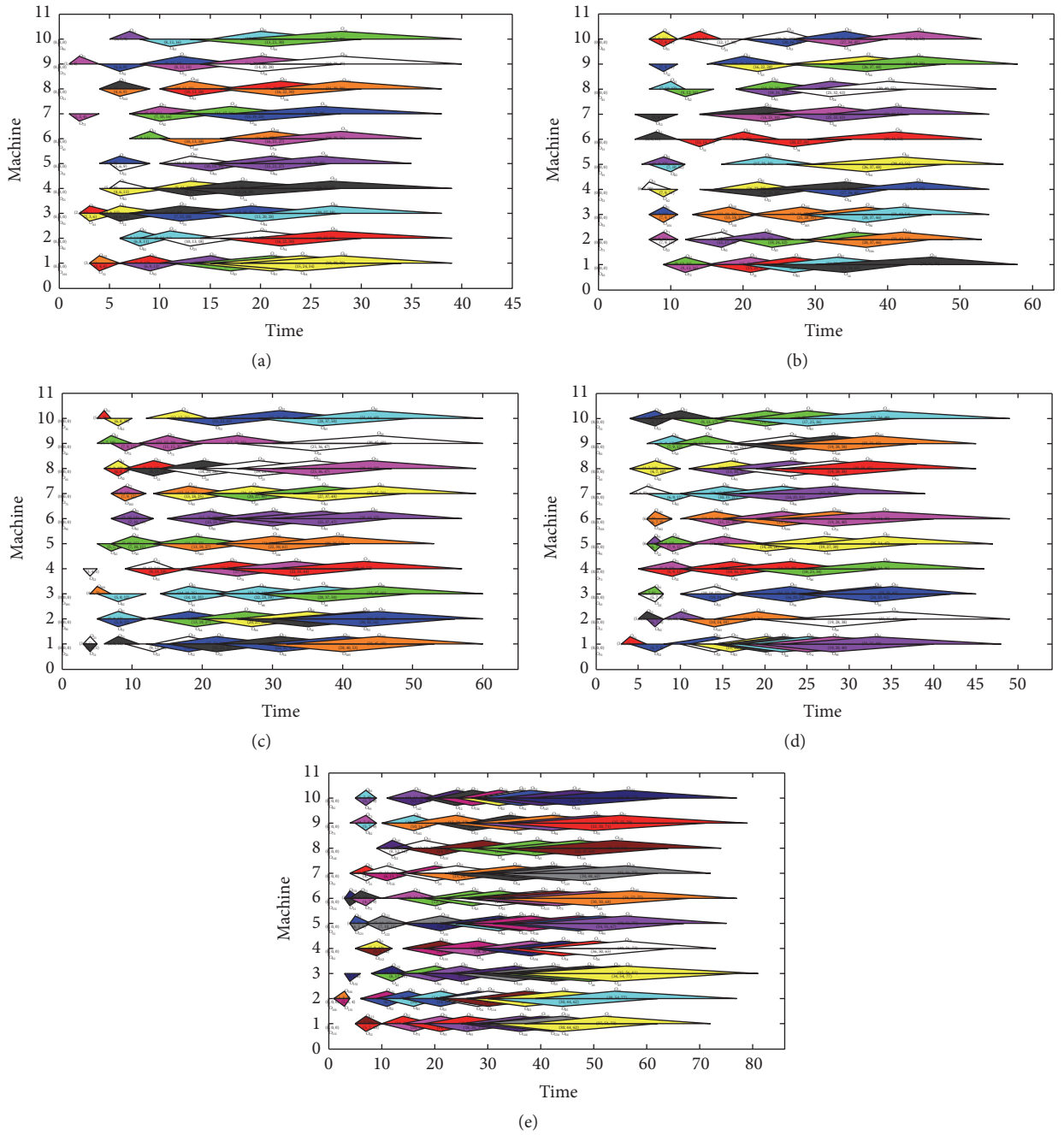

Figure 7: (a) Gantt chart of Instance 1 obtained by IDPSO (19, 28, 40). (b) Gantt chart of Instance 2 obtained by IDPSO (30, 46, 58). (c) Gantt chart of Instance 3 obtained by IDPSO (33, 45, 60). (d) Gantt chart of Instance 4 obtained by IDPSO (23, 35, 49). (e) Gantt chart of Instance 5 obtained by IDPSO (35, 56, 81).

superiority between two data sets. Using $c1$ as the criterion, we will do statistical test and Wilcoxon signed rank test to assess IDPSO by comparisons of IDPSO and EDA [28] and IDPSO and DIGA [30]. In our test, significance level is 0.05. Also, the experiment of IDPSO optimizing Instance 1 to Instance 5 under noise conditions is also conducted to assess the ability to optimize FJSPF with noise and the noise is Gaussian noise, of which the average value is 0 and the standard deviation is 1.

Table 10 lists the average value of $c1$ (denoted by AV($c1$)) and the standard deviation of $c1$ (denoted by STD($c1$)) obtained by IDPSO, EDA, and DIGA. Under noise condition, the average value of $c1$ (denoted by $\mathrm{AV}(c1)_{\mathrm{noise}}$) and the standard deviation of $c1$ (denoted by $\mathrm{STD}(c1)_{\mathrm{noise}}$) obtained by IDPSO are also listed in Table 10 for comparison. p value_1 and p value_2, h value_1 and h value_2 denote p value and h value of Wilcoxon signed rank test obtained by comparisons of IDPSO and EDA, IDPSO and DIGA, respectively. From

Table 10: Results of statistical analysis and noise conditions of five instances.

		Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
IDPSO	AV($c1$)	**3.006e + 001**	**4.577e + 001**	**4.643e + 001**	**3.740e + 001**	**5.882e + 001**
	STD($c1$)	**6.014e − 001**	**4.795e − 001**	**9.669e − 001**	**4.281e − 001**	**9.863e − 001**
	$\text{AV}(c1)_{\text{noise}}$	3.047e + 001	4.627e + 001	4.652e + 001	3.790e + 001	5.941e + 001
	$\text{STD}(c1)_{\text{noise}}$	5.945e − 001	3.425e − 001	8.696e − 001	8.010e − 001	7.637e − 001
	p value_1	2.751e − 011	4.043e − 011	3.357e − 011	1.786e − 004	1.726e − 004
	h value_1	1	1	1	1	1
	p value_2	2.786e − 011	2.471e − 011	2.933e − 011	1.786e − 004	1.746e − 004
	h value_2	1	1	1	1	1
EDA	AV($c1$)	3.379e + 001	4.871e + 001	5.295e + 001	4.097e + 001	6.437e + 001
	STD($c1$)	1.114e + 000	1.826e + 000	1.966e + 000	1.591e + 000	1.022e + 000
DIGA	AV($c1$)	3.441e + 001	4.969e + 001	5.353e + 001	4.310e + 001	6.670e + 001
	STD($c1$)	2.676e + 000	2.232e + 000	3.100e + 000	2.852e + 000	4.395e + 000

AV($c1$) and STD($c1$) in Table 10, it is clearly seen that IDPSO obtains the best value among three algorithms for all five instances. For comparing AV($c1$) and $\text{AV}(c1)_{\text{noise}}$, STD($c1$) and $\text{STD}(c1)_{\text{noise}}$ obtained by IDPSO, the results under noise conditions are slightly worse than the results with no noise. Therefore, the noise hardly damages the performance of IDPSO to some degree. Also, since all h value_1 and h value_2 are equal to 1, this indicates that IDPSO statistically outperforms EDA and DIGA with the probability of 95%. From the above analysis, IDPSO has a better comprehensive performance than other algorithms to optimize FJSPF and also can solve FJSPF under noise conditions.

Some superiorities of the proposed algorithm are summarized as follows: Several initial methods based on TFNs and three initial methods from other papers are applied to produce abundant initial particles and more promising solutions. High-quality and high-diversity initial population can make the particles reach more promising area in the initial phase. Simple and effective discrete operators are introduced to IDPSO for exchanging the information of particles. Discrete operator f_2 is to exchange the machine information from personal best positions and global best positions, and then perturbation operator f_3 can find better solution from other space and increase the exploration of the population. The appropriate combination of discrete operators f_2 and f_3 is a valid method to balance the exploration and exploitation capabilities. Moreover, global best position is a significant factor to guide the population to the global optimum. Several global best positions are used to update the particles' positions and that contributes to correcting some global best positions' errors in search process and avoiding being trapped into local optima. Consequently, this comprehensive strategy makes IDPSO have a better performance on FJSPF.

5. Conclusion

In this work, an improved version of PSO with discrete operators is provided to solve FJSPF. Firstly, several heuristic initial methods based on TFNs are developed to initialize machine assignment and random method is used to initialize operation sequence. Secondly, simple and effective discrete operators are employed in IDPSO to update particle's position and perturbation operator f_3 is applied to explore more area. Thirdly, several global best positions retained in global best set can make the population obtain more information about global best area and correct some global best positions' errors. Finally, the impacts of four parameters are investigated by Taguchi method and parameter selection is suggested by designed experiments on Instance 1. Furthermore, five instances of FJSPF are used to evaluate IDPSO and the comparisons with several previous published algorithms are also performed. The experimental results show that IDPSO is able to obtain better solutions on Instance 1 to Instance 5 and is a competitive method to solve FJSPF. How to address dynamic scheduling problem and establish more valid models for job shop manufacturing system needs our future efforts.

Competing Interests

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

Acknowledgments

This work is supported by National Natural Science Foundation of China (Project no. 61572238), National High-Tech Research and Development Projects of China (Project no. 2014AA041505), and Provincial Outstanding Youth Foundation of Jiangsu Province (Project no. BK20160001).

References

[1] 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.

[2] C. Özgüven, Y. Yavuz, and L. Özbakır, "Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times," *Applied Mathematical Modelling*, vol. 36, no. 2, pp. 846–858, 2012.

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

[4] J. C. Tay and N. B. Ho, "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems," *Computers and Industrial Engineering*, vol. 54, no. 3, pp. 453–473, 2008.

[5] N. B. Ho and J. C. Tay, "Evolving dispatching rules for solving the flexible job-shop problem," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 3, pp. 2848–2855, Edinburgh, Scotland, September 2005.

[6] X. Xu, Q. Guan, W. Wang, and S. Chen, "Transient chaotic discrete neural network for flexible job-shop scheduling," in *Proceedings of the 2nd International Symposium on Neural Networks (ISNN '05)*, pp. 762–769, June 2005.

[7] H. Zang, S. Zhang, and K. Hapeshi, "A review of nature-inspired algorithms," *Journal of Bionic Engineering*, vol. 7, pp. S232–S237, 2010.

[8] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm and Evolutionary Computation*, vol. 16, pp. 1–18, 2014.

[9] M. Cheng, P. R. Tadikamalla, J. Shang, and S. Zhang, "Bicriteria hierarchical optimization of two-machine flow shop scheduling problem with time-dependent deteriorating jobs," *European Journal of Operational Research*, vol. 234, no. 3, pp. 650–657, 2014.

[10] C.-L. Chen, S.-Y. Huang, Y.-R. Tzeng, and C.-L. Chen, "A revised discrete particle swarm optimization algorithm for permutation flow-shop scheduling problem," *Soft Computing*, vol. 18, no. 11, pp. 2271–2282, 2014.

[11] L. Guo, S. Zhao, S. Shen, and C. Jiang, "Task scheduling optimization in cloud computing based on heuristic algorithm," *Journal of Networks*, vol. 7, no. 3, pp. 547–553, 2012.

[12] S. Burnwal and S. Deb, "Scheduling optimization of flexible manufacturing system using cuckoo search-based approach," *International Journal of Advanced Manufacturing Technology*, vol. 64, no. 5–8, pp. 951–959, 2013.

[13] M. M. Lotfi and R. Tavakkoli-Moghaddam, "A genetic algorithm using priority-based encoding with new operators for fixed charge transportation problems," *Applied Soft Computing Journal*, vol. 13, no. 5, pp. 2711–2726, 2013.

[14] K. A. A. D. Raj and C. Rajendran, "A genetic algorithm for solving the fixed-charge transportation model: two-stage problem," *Computers & Operations Research*, vol. 39, no. 9, pp. 2016–2032, 2012.

[15] R. I. Bolaños, M. G. Echeverry, and J. W. Escobar, "A multiobjective non-dominated sorting genetic algorithm (NSGA-II) for the multiple traveling salesman problem," *Decision Science Letters*, vol. 4, no. 4, pp. 559–568, 2015.

[16] D. Karaboga and B. Gorkemli, "A combinatorial Artificial Bee Colony algorithm for traveling salesman problem," in *Proceedings of the International Symposium on Innovations in Intelligent Systems and Applications (INISTA '11)*, pp. 50–53, June 2011.

[17] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 5, pp. 682–691, 2012.

[18] G. Kanagaraj, S. G. Ponnambalam, N. Jawahar, and J. M. Nilakantan, "An effective hybrid cuckoo search and genetic algorithm for constrained engineering design optimization," *Engineering Optimization*, vol. 46, no. 10, pp. 1331–1351, 2014.

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

[20] Y. Yuan and H. Xu, "Flexible job shop scheduling using hybrid differential evolution algorithms," *Computers & Industrial Engineering*, vol. 65, no. 2, pp. 246–260, 2013.

[21] S. Karthikeyan, P. Asokan, S. Nickolas, and T. Page, "A hybrid discrete firefly algorithm for solving multi-objective flexible job shop scheduling problems," *International Journal of Bio-Inspired Computation*, vol. 7, no. 6, pp. 386–401, 2015.

[22] J.-Q. Li, Q.-K. Pan, and M. F. Tasgetiren, "A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities," *Applied Mathematical Modelling*, vol. 38, no. 3, pp. 1111–1132, 2014.

[23] 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. 60, no. 1–4, pp. 303–315, 2012.

[24] K. Z. Gao, P. N. Suganthan, Q. K. Pan, T. J. Chua, T. X. Cai, and C. S. Chong, "Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives," *Journal of Intelligent Manufacturing*, vol. 27, no. 2, pp. 363–374, 2016.

[25] Y. Yuan, H. Xu, and J. Yang, "A hybrid harmony search algorithm for the flexible job shop scheduling problem," *Applied Soft Computing Journal*, vol. 13, no. 7, pp. 3259–3272, 2013.

[26] T.-C. Chiang and H.-J. Lin, "A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling," *International Journal of Production Economics*, vol. 141, no. 1, pp. 87–98, 2013.

[27] S. H. A. Rahmati and M. Zandieh, "A new biogeography-based optimization (BBO) algorithm for the flexible job shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 58, no. 9–12, pp. 1115–1129, 2012.

[28] D. Lei, "A genetic algorithm for flexible job shop scheduling with fuzzy processing time," *International Journal of Production Research*, vol. 48, no. 10, pp. 2995–3013, 2010.

[29] D. Lei, "Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling," *Applied Soft Computing*, vol. 12, no. 8, pp. 2237–2245, 2012.

[30] 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.

[31] G. Bortolan and R. Degani, "A review of some methods for ranking fuzzy subsets," *Fuzzy Sets and Systems*, vol. 15, no. 1, pp. 1–19, 1985.

[32] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Washington, DC, USA, December 1995.

[33] 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.

[34] J.-Q. Li, Q.-K. Pan, P. N. Suganthan, and T. J. Chua, "A hybrid tabu search algorithm with an efficient neighborhood structure

for the flexible job shop scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 52, no. 5-8, pp. 683–697, 2011.

[35] J. Lin, "A hybrid biogeography-based optimization for the fuzzy flexible job-shop scheduling problem," *Knowledge-Based Systems*, vol. 78, no. 1, pp. 59–74, 2015.

[36] 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.

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

[38] L. Wang, G. Zhou, Y. Xu, and M. Liu, "A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem," *International Journal of Production Research*, vol. 51, no. 12, pp. 3593–3608, 2013.