Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 184643, 10 pages
http://dx.doi.org/10.1155/2015/184643

Research Article

A Modified Biogeography-Based Optimization for the Flexible Job Shop Scheduling Problem

Yuzhen Yang

School of Electrical Engineering, Shanghai Dianji University, Shanghai 200240, China

Correspondence should be addressed to Yuzhen Yang; young_goodstar@126.com

Received 20 May 2015; Accepted 28 September 2015

Academic Editor: George S. Dulikravich

The flexible job shop scheduling problem (FJSSP) is a practical extension of classical job shop scheduling problem that is known to be NP-hard. In this paper, an effective modified biogeography-based optimization (MBBO) algorithm with machine-based shifting is proposed to solve FJSSP with makespan minimization. The MBBO attaches great importance to the balance between exploration and exploitation. At the initialization stage, different strategies which correspond to two-vector representation are proposed to generate the initial habitats. At global phase, different migration and mutation operators are properly designed. At local phase, a machine-based shifting decoding strategy and a local search based on insertion to the habitat with best makespan are introduced to enhance the exploitation ability. A series of experiments on two well-known benchmark instances are performed. The comparisons between MBBO and other famous algorithms as well as BBO variants prove the effectiveness and efficiency of MBBO in solving FJSSP.

1. Introduction

The job shop scheduling problem (JSSP) is one of the most difficult combination optimization problems in manufacturing systems. As an extension of JSSP, the flexible job shop scheduling problem (FJSSP) is much closer to the practical production because of the ability of machines to perform more than one type of operations. Therefore, the problem that FJSSP should deal with is not only the sequencing of operations like JSSP but also the assignments of machines for operations. It makes solving FJSSP much more difficult than classical JSSP. Hence, the study on FJSSP theoretically or methodologically would have positive influence in application field.

Since the first study on FJSSP was proposed by Brucker and Schlie in 1990 [1], a plenty of research has been carried out and most of them were focused on designing metaheuristics to solve FJSSP with makespan or tardiness minimization. Since FJSSP consists of two subproblems, which are operation sequence subproblem and machine assignment subproblem, the conventional method for solving this problem is to treat these two subproblems separately with different strategies. By doing this, the internal relationship between these two subproblems has been neglected. As research continues, the integrated approaches by considering the two subproblems simultaneously are proposed with high complexity and better solutions. Brandimarte [2] used some dispatching rules to solve the machine assignment problems and tabu search (TS) for the sequencing problems. Mastrolilli and Gambardella [3] proposed TS based on several neighborhood structures. Pezzella et al. [4] proposed a genetic algorithm (GA) integrating different strategies. Ho and Tay did a serial of studies on FJSSP. They first designed genetic programming (GP) to combine dispatching rules [5] and came up with GENACE via incorporating composite dispatching rules (CDRs) and cultural algorithm [6] and then LEGA based on GA and machine learning mechanism [7]. Recently, Yazdani et al. [8] developed a parallel variable neighborhood search (PVNS) algorithm based on six neighborhood structures. Lei designed a coevolutionary GA [9] and swarm-based neighborhood search algorithm [10] for fuzzy FJSSP. Wang et al. proposed single population and bipopulation based estimation of distribution algorithm (EDA [11] and BEDA [12]) for the FJSSP and came up with better results. Yuan and Xu [13] presented a hybrid differential evolution (HDE) algorithm with new speed-up strategy on some local search.

With regard to the multiobjective FJSSP, weighted methods and Pareto-based methods are commonly studied. The objectives generally are makespan, total work load of machines, and maximum work load of machine. Kacem et al. [14, 15] proposed several effective evolutionary algorithms. Ho et al. [16] used CDRs, GA, and machine learning mechanism to optimize several objectives. Gao [17, 18] added shifting bottleneck and variable neighborhood search to GA. Zhang et al. [19] developed a hybrid particle swarm optimization (PSO) incorporated with a novel initialization. Li et al. [20] introduced a hybrid TS algorithm and discrete artificial bee colony (ABC) algorithm for the MJSSP. Wang et al. [21] further developed Pareto-based EDA (PEDA).

Biogeography-based optimization (BBO), a combination of biogeography and engineering science, was proposed by Simon in 2008 [22]. Due to the similar features with other biology-based optimization algorithms, such as GA and PSO, BBO is applicable to many types of problems that GA and PSO are applied for. However, BBO also has some unique features from other types of biology-based optimization. The initial populations in BBO survive forever and their characteristics change gradually as the search process progressed. And the main parameters of individuals are decided by their ranking in population, which would cause no extra setting process.

So far, on count of all these features, BBO has been applied to many academic and application problems [23–25]. However, there is little research about BBO to solve scheduling problem. In this paper, we propose a modified BBO (MBBO) to solve the FJSSP with makespan minimization. When designing the algorithm, we focus on the balance between exploration and exploitation. Considering two-vector coding, different strategies are introduced to generate the initial solutions. A machine-based shifting decoding method is proposed to convert the vectors to feasible schedule at local exploitation phase. Different migration and mutation operators are designed at global exploration phase. Particularly, a local search is applied to the solution with the best makespan to speed up the convergence of the algorithm while maintaining the good character of the best solution at local exploitation phase. In addition, experiments are performed to examine how the parameters affect the performance of the algorithm. Last but not least, we compare MBBO with other existing famous algorithms and BBO variants on two sets of benchmark instances to test the efficiency and effectiveness of BBO in solving FJSSP.

The reminder of the paper is organized as follows. In Section 2, we explain the flexible job shop scheduling problem as well as the concept and structure of basic BBO algorithm in Section 3. In Section 4, the modified BBO is developed to FJSSP subsequently. Section 5 analyzes the performance results of MBBO when applied to solve common benchmark problems in literature. At last, we come to our conclusion and some possible future directions.

2. Problem Formulation

FJSSP is one of the most practical and hardest combinatorial optimization problems. During past two decades, a bunch of literature has been published, but no efficient algorithm has been presented yet for solving it to optimality in polynomial time.

Table 1: Example of FJSSP with 3 jobs, 4 machines, and 9 operations.

FJSSP		Processing Time			
		M_1	M_2	M_3	M_4
J_1	O_{11}	3	4	∞	5
	O_{12}	4	6	5	∞
	O_{13}	3	∞	∞	6
J_2	O_{21}	4	∞	∞	5
	O_{22}	∞	5	∞	3
	O_{23}	5	3	5	2
	O_{24}	3	6	∞	∞
J_3	O_{31}	5	3	4	3
	O_{32}	4	2	1	∞

Suppose we are given n jobs and m machines. Each machine can handle one job at most at a time. Each job i consists of a sequence of operations and needs to be processed during an uninterrupted time period on one from a set of feasible machines. The purpose is to find a scheme, that is, the job sequence on the machines as to optimize one or more performance measurements, makespan in our case. An example of 3 jobs, 4 machines, and 9 operations is shown in Table 1, where ∞ indicates that the machine can process the operation, however, with infinity processing time. And the FJSSP could be divided into two types, total FJSSP and partial FJSSP, based on whether each operation can be processed on all of the machines.

The mathematical model of FJSSP with makespan minimization can be formulated as [20]

$$\begin{aligned}
&\text{Minimise} \quad C_M = \max_{1\le i\le n}\left\{C_{i,q_i}\right\}\\
&\text{Subject to} \quad C_{i,j} - C_{i,j-1} \ge p_{i,j,k}x_{i,j,k}, \quad j = 2,3,\ldots,q_i;\ \forall i,k\\
&\qquad \sum_{k\in M_{i,j}} x_{i,j,k} = 1, \quad \forall i,j\\
&\qquad x_{i,j,k}\\
&\qquad = \begin{cases} 1, & \text{if operation } O_{i,j} \text{ is processed on machine } k\\ 0, & \text{otherwise}\end{cases}\\
&\qquad C_{i,j} \ge 0, \quad \forall i,j.
\end{aligned} \tag{1}$$

The indices and variables of the model are enumerated as follows: m is number of machines; n is number of jobs; i is job index; j is operation number index; k is machine index; q_i is number of operations of job i; $C_{i,j}$ is finish time of jth operation of job i; $p_{i,j,k}$ is processing time of jth operation of job i on machine k; $M_{i,j}$ is set of machines that can process jth operation of job i.

3. Biogeography-Based Optimization

BBO algorithm, proposed by Simon in 2008, is inspired by the mathematics of biogeography and mainly the work

```
I : ∅ → {H^n, HSI^n}
While not T
  Ψ = (m, n, λ, μ, Ω, M)
end
```

Pseudocode 1: Pseudocode of BBO for optimization problems.

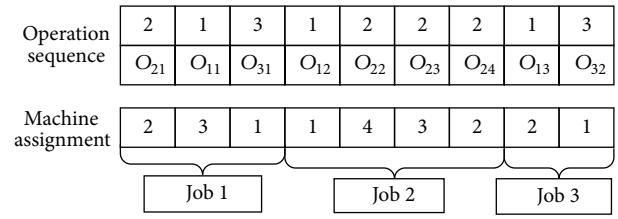

Figure 1: The coding strategy for MBBO.

from MacArthur and Wilson [26]. Later, a large amount of theoretical, methodological, and practical studies on BBO have come into being.

The two main concepts of BBO are habitat suitability index (HSI) and suitability index variables (SIVs). Features that correlate with HSI include rainfall, diversity of topographic features, land area, and temperature. And SIVs are considered as the independent variables of the habitat. Geographical areas that are well suited for species are said to own a high HSI. Considering the optimization algorithm, a population of candidate solutions can be represented as vectors. Each integer in the solution vector is considered to be an SIV. After assessing performance of the solutions, good solutions are considered to be habitats with a high HSI, and poor ones are considered to be habitats with a low HSI. Therefore, HSI is analogous to fitness in other population-based optimization algorithms. A BBO algorithm can be described as in Pseudocode 1.

A BBO algorithm is a 3-tuple evolutionary algorithm that proposes a solution to an optimization problem. $I : \varnothing \rightarrow \{H^n, \mathrm{HSI}^n\}$ is a function that creates an initial ecosystem of habitats and computes each corresponding HSI. $\Psi = (m, n, \lambda, \mu, \Omega, M)$ is a 6-tuple ecosystem transition function that modifies the ecosystem from one optimization iteration to the next, which includes m: the number of SIVs; n: the size of habitats; λ: the immigration rate; μ: the emigration rate; Ω: the migration operator; M: the mutation operator.

As it is clear from Pseudocode 1, the two main operators of BBO are migration and mutation. Here we present a brief introduction of these two operators and more details can be found in the literature.

Migration operator, including immigration and emigration, bridges the communication of habitats in an ecosystem. The emigration and immigration rates of each solution are used to probabilistically share information between habitats. Deciding whether or not a habitat performs emigration or immigration is up to its HSI. A habitat with high HSI meaning more species has more opportunities to emigrate to neighboring habitats. And a habitat with low HSI meaning sparse species need immigration to increase the diversity of its species. Therefore, habitats with high HSI have larger emigration rates and smaller immigration rates while ones with low HSI have larger immigration rates and smaller emigration rates. A model of immigration and emigration rate could be set as linear or nonlinear.

Since cataclysmic events can drastically change the HSI of a natural habitat, mutation operator is designed to model these random events. Moreover this operator could increase the diversity of BBO to further help jump out of local optimal to some extent. The mutation rate of each habitat here, unlike the one in GA, is not set as the same value randomly. It is decided by probability of species numbers of each habitat. $P_s(t)$ refers to the possibility of a habitat having s species. Then the $P_s(t + \Delta t)$ can be calculated as

$$P_s(t+\Delta t) = P_s(t)\left(1 - \lambda_s \Delta t - \mu_s \Delta t\right) + P_{s-1}\lambda_{s-1}\Delta t + P_{s+1}\mu_{s+1}\Delta t. \tag{2}$$

Here give (3) derived from (2) by taking $\Delta t \rightarrow 0$

$$P_s' = \begin{cases} -\left(\lambda_s + \mu_s\right)P_s + \mu_{s+1}P_{s+1}, & s = 1 \\ -\left(\lambda_s + \mu_s\right)P_s + \lambda_{s-1}P_{s-1} + \mu_{s+1}P_{s+1}, & 1 < s \le n-1 \\ -\left(\lambda_s + \mu_s\right)P_s + \lambda_{s-1}P_{s-1}, & s = n. \end{cases} \tag{3}$$

And then the mutation rate of each habitat is calculated according to its P_s:

$$m_s = m_{\max}\left(1 - \frac{P_s}{P_{\max}}\right), \tag{4}$$

where $m_{\max}$ is a predefined value meaning the maximum mutation rate.

4. MBBO for FJSSP

In this section, we will propose a modified biogeography-based optimization (MBBO) to solve the FJSSP. Different from the traditional shifting decoding strategy, a novel machine-based shifting one, which performs a kind of local search, will be presented to decode the representations to feasible schedules more effectively.

4.1. Habits Representation. It is clear that the FJSSP has two subproblems, which are the allocation of operations on machines and the sequencing of operations. To handle these subproblems, our MBBO operates on two chromosomes which correspond to the two problems, respectively, which has achieved good results by cooperating with GA [9]. Each of them is coded as a D-dimensional real-parameter vector $[x_1, x_2, x_3, \ldots, x_D]$, where D is the total number of operations. A scheme of this coding strategy is shown in Figure 1.

For the operation sequence vector, the operations which belong to the same job are signed with the same symbol and

the kth occurrence of a job number refers to the kth operation of the job. For the instance in Table 1, J_1 has three operations O_{11}, O_{12}, and O_{13}; J_2 has four operations O_{21}, O_{22}, O_{23}, and O_{24}; J_3 has two operations O_{31} and O_{32}. One possible coding vector of operation sequence $[2\ 1\ 3\ 1\ 2\ 2\ 2\ 1\ 3]$ in Pseudocode 1 can be translated into a sequence represented as $[O_{21}, O_{11}, O_{31}, O_{12}, O_{22}, O_{23}, O_{24}, O_{13}, O_{32}]$.

And the machine assignment vector presents the assigned machine of each operation successively. For instance, in Table 1, the operations for the three jobs are 3, 4, and 2, respectively. Hence the first three SIVs stand for the assigned machines for J_1 successively, the following four SIVs for J_2, and the last two SIVs for J_3. Therefore, the two vectors in Pseudocode 1 can be translated as $[(O_{21}, M_1), (O_{11}, M_2), (O_{31}, M_2), (O_{12}, M_3), (O_{22}, M_4), (O_{23}, M_3), (O_{24}, M_2), (O_{13}, M_1), (O_{32}, M_1)]$.

This representation has an advantage that any habitat can be decoded to a feasible schedule. There does not exist the special case that processes the schedule operations whose technological predecessors have not been scheduled yet. Moreover, it shows another advantage that the search template of the evolutionary search has no concern with details of particular scheduling problems.

4.2. Decoding Vectors to Feasible Schedules. After the representation, the coding vectors must be transformed into a feasible schedule and a powerful decoding strategy plays an important role in improving the final solutions.

The most common decoding strategy for FJSSP, existing in literatures, is called forcing or left shifting [27], which is widely applied in classical JSSP. The decoding allocates each operation on its assigned machine one by one in the order represented by the coding. When operation $O_{i,j}$ is scheduled on machine k, the idle time between operations that have already been processed on that machine is examined from left to right to find the earliest one that is not shorter than the process time of operation $O_{i,j}$. If such an interval exists, it is allocated there; otherwise, it is allocated at the current end of machine k.

Obviously, the left shifting only shifts operations on the predefined machine. Considering the fact that a machine has the ability to perform more than one operation in FJSSP, a machine-based shifting decoding strategy is proposed, where the shifting to other capable machines is also examined. The process works as follows, where π_1 and π_2 refer to the operation sequence vector and machine assignment vector, respectively.

(1) For the first operation of operation sequence vector $\pi_1(1)$, schedule it on the machine k with the shortest processing time.

(2) For each operation $\pi_1(l)$, $l = 2, 3, \ldots, n$, the starting time of corresponding operation $O_{i,j}$ is calculated as the following situations, where $L_{\pi_2(i,j)}$ stands for the order of operation $O_{i,j}$ on the machine determined by machine assignment vector π_2; $\mathrm{St}(O_{i,j})$ refers to the starting time of operation $O_{i,j}$:

(a) If $L_{\pi_2(i,j)} = 1 \& j = 1$, set $\mathrm{St}(O_{i,j}) = 0$ and schedule it on the machine k with the shortest processing time.

(b) If $L_{\pi_2(i,j)} = 1 \& j \neq 1$, set $\mathrm{St}(O_{i,j}) = C_{i,j-1}$ and schedule it on machine $\pi_2(O_{i,j})$.

(c) If $L_{\pi_2(i,j)} \neq 1 \& j = 1$, set $\mathrm{St}(O_{i,j}) = \min(C_{i,j}) - p_{i,j,k}$ and schedule it on machine k.

(d) If $L_{\pi_2(i,j)} \neq 1 \& j \neq 1$, set $\mathrm{St}(O_{i,j}) = \min(C_{i,j}) - p_{i,j,k}$ while $\mathrm{St}(O_{i,j}) > C_{i,j-1}$ and schedule it on machine k.

(3) Modify the two vectors according to the final schedule.

The schedules generated by shifting decoding and machine-based shifting decoding of the example shown in Table 1 and Figure 1 are depicted in Figures 2 and 3, respectively. The makespan obtained by machine-based shifting is 13 while the one obtained by shifting strategy is 20. It is clear that machine-based strategy functions effectively as a kind of local search to improve the final schedule.

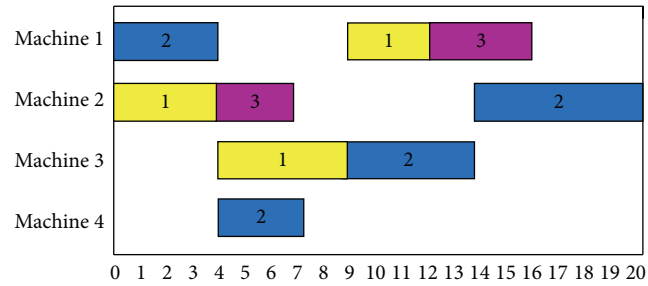

Figure 2: The Gantt chart of shifting decoding process for FJSSP.

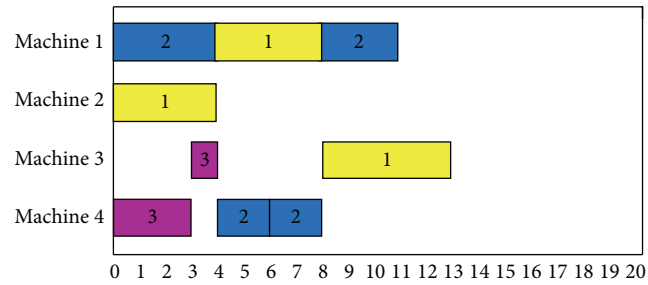

Figure 3: The Gantt chart of machine-shifting decoding process for FJSSP.

4.3. Initial Habitats. At the beginning of MBBO, two different methods, which correspond to two vectors, are designed to generate initial habitats.

To generate initial operation ones, the simple random rule is served due to the strong decoding strategy in this paper. However, this is not suitable for the machine assignment vectors initialization because of the existence of partial flexibility. At least one operation cannot be processed on some machine. Hence, random rule might generate infeasible solutions. Here the following 2-tournament strategy is designed for the

```
For i = 1 : NP
  if Rand(0, 1) < λ_i
      using Tournament selection to choose H_j;
      Migration (H_i, H_j);
  end
end
```

Pseudocode 2: Selection process of the migration operator.

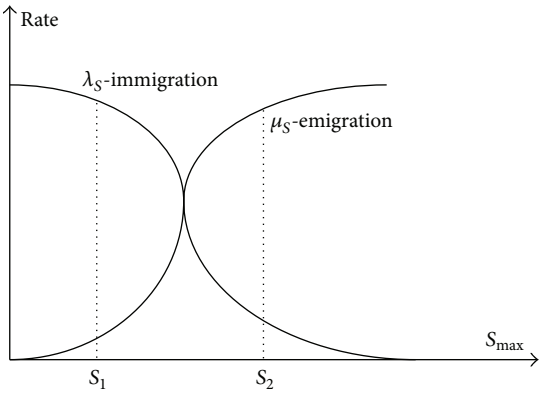

Figure 4: Cosine migration model curve of BBO.

machine assignment vectors. For each operation, two capable machines are selected randomly. Then the one with shorter processing time is chosen as the processing machine.

4.4. Migration Operator. Migration operator is a probabilistic one to share information between habitats based on emigration and immigration rates. Each habitat H_i should be checked whether it performs immigration with respect to its immigration rate λ_i. If so, habitat H_j is selected as emigration habitat with respect to its emigration rate μ_j. The process is shown in Pseudocode 2.

And the immigration rate λ and emigration rate μ are calculated as (5) and (6), respectively, shown in Figure 4. One has

$$\lambda_s = \left(\frac{I}{2}\right) * \left(\cos\left(\frac{s\pi}{n}\right) + 1\right), \tag{5}$$

$$\mu_s = \left(\frac{E}{2}\right) * \left(-\cos\left(\frac{s\pi}{n}\right) + 1\right), \tag{6}$$

where s refers to the species size of the habitat; n is the maximum species size; I and E are the immigration and emigration coefficient, respectively, and generally set to be 1. Literature [28] proves this advantage of this cosine model over linear ones.

With regard to the two-vector representation of MBBO, the IPOX [29] and 0-1 uniform crossover are applied to the operation sequence vector and machine assignment vector, respectively. The processes are depicted in Figures 5 and 6. By doing so, there are no needs for any modification, which avoid burdening the algorithm with extra computation time.

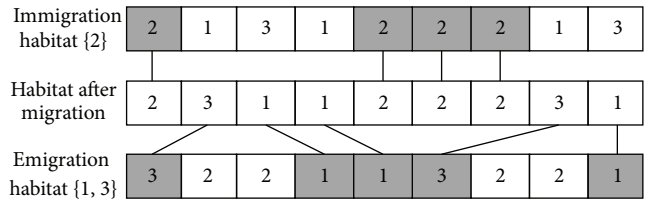

Figure 5: IPOX operator on operation sequence vector.

```
For i = 1 : NP
   using m_i selecting H_i & Rank_i ≠ 1
   Mutation (H_i);
end
```

Pseudocode 3: Selection process of the mutation operator.

4.5. Mutation Operator. The mutation operator in BBO performs the same role as the one in GA to increase diversity of the habitats. Similarly, the mutation operator should undergo a selection process. Whether or not a habitat performs mutation operator is determined by its mutation rate m_i calculated by (2), (3), and (4). The selection process of mutation operator is shown in Pseudocode 3. Note that an elitism approach is employed to save the features of the habitat that has the best solution in MBBO process. The habitat with the best solution has a mutation rate of 0.

Likewise, mutation operator on operation sequence vector and machine assignment vector should be designed. A random arrangement on part of the vector is performed on operation sequence vector, shown in Figure 7, and no further modification is needed. With regard to the machine assignment vector, random method is not proper because of the existence of partial flexible jobs. A subsequence replacement is applied to the machine assignment vector. A subsequence of the vector is selected randomly. And then for each SIV in the subsequence, the Rolette wheel process is performed to select one machine from a set of capable machines. The current machine is replaced by the selected machine. The detailed description of this process is shown in Figure 8.

4.6. Local Search for the Best Habitat. Local search is used to search the neighbor habitats of the habitat with the best HSI. By doing so, the good characters of habitats with better HSI are reserved, and furthermore the convergence speed of the search process is greatly accelerated. The local search is realized as follows. The habitat with the best HSI is selected and performed insertion n times at most. The operator insert(π, l, u, v) refers to insert l variants after position μ to position ν in the vector π. Parameter l is randomly generated among $[1, L/5]$ where L is length of the vector. If current neighbor habitat has a better HSI than the original habitat, terminate the iterations and replace the original habitat with the neighbor one; if not, continue the iterations until the termination criteria are satisfied. The pseudocode of this local search is shown in Pseudocode 4.

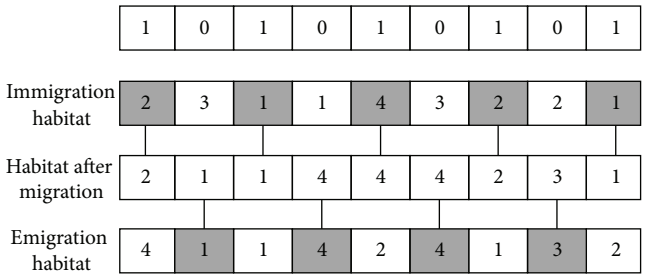

Figure 6: Uniform crossover operator on machine assignment vector.

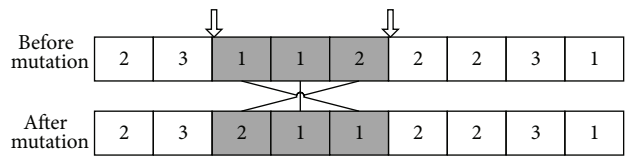

Figure 7: Process of the mutation operator on operation sequence vector.

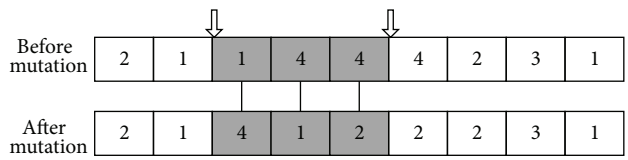

Figure 8: Process of the mutation operator on machine assignment vector.

```
select H_i ⊆ {Rank_i = 1}
loop = 1;
do {
        randomly select μ, ν & μ ≠ ν;
        po = randperm(ceil(L/5));
        H_i' = insert(H_i, po(1), u, v);
        if H_i' < H_i
        H_i = H_i';
        break;
        end
        loop++;
} while loop ≦ n
Return H_i
```

Pseudocode 4: The pseudocode of local search based on insertion.

4.7. Framework of MBBO. With the above design, the entire procedure of MBBO to solve the FJSSP is depicted in Figure 9.

5. Computation Results and Comparisons

Computational experiments are carried out to investigate the performance of our proposed modified BBO algorithm. In order to evaluate the performance of MBBO, we run the algorithm on a series of widely used benchmark problems including four Kacem instances [14, 30] and ten Brandimarte instances [2].

All the programs in the experiments were written in Matlab and all the experiments were running on platform using Intel Core 4 Quad 2.4 GHZ CPU with 2 GB RAM. First we applied some experiments to examine parameter settings of MBBO. And then we compared MBBO with other well-known evolutionary algorithms for FJSSP. Last but not least, several BBO variants were compared and discussed.

5.1. Parameter Discussion. The parameters have a great influence on the performance of algorithms. There are several parameters that should be set properly in most evolutionary algorithms in order to make the algorithms more effective, such as population size, crossover possibility, and mutation possibility in GA. Things are different with BBO. Note that the three main parameters in BBO are immigration rate, emigration rate, and mutation rate of each habitat, which are calculated by their rankings. Thus, there is no need to set these three parameters. The remaining two parameters in BBO that should be designed are habitats size and termination rule. Since the trend in metaheuristic methods is to reduce the number of design parameters, MBBO has the advantage of scalability than other algorithms.

Considering the different sizes of benchmark problems, a series of experiments were performed on each instance to investigate how habitat sizes and termination rules affect MBBO. The results on four Kacem instances are shown in Figures 10–13 and the sizes of habitats were set as 10, 50, 100, and 200.

It is obvious that the size of habitats has little impact on the algorithm in Kacem instances. In the first three instances, four sizes of habitats reached the optimal solution within 50 generations. In accordance with the results of Kacem04 in Figure 13, larger size of habitats accelerated the convergence speed of MBBO at the expense of a little computation time increment. Through overall consideration, the size of habitats and termination generation are set as 50 and 100 separately for all four Kacem instances.

Similarly, the two parameters for ten BRdata instances were tested by experiments. The results showed that the optimal parameters for different instances were not the same. Figure 14 displayed the situation with MK01 under different sizes of habitats. The situation was similar to the one with Kacem04. The settings for all the ten BRdata instances were listed in Table 2.

5.2. Results of Kacem Instances. Kacem instances consist of four problems with the scale ranging from 4 jobs 5 machines to 15 jobs 10 machines. And our MBBO algorithm was compared with several famous algorithms for FJSSP including LEGA [7], BBO [31], and BEDA [12]. The results are listed in Table 3. For each instance, we run MBBO 50 times independently and record the best makespan C and average makespan AV(C). The results of other three algorithms are directly taken from relative literatures.

From Table 3, it is obvious that MBBO solved the Kacem instances with both effectiveness and stability. In all four instances, MBBO and BEDA reached the recorded optimal values in each of the 50 independent runs.

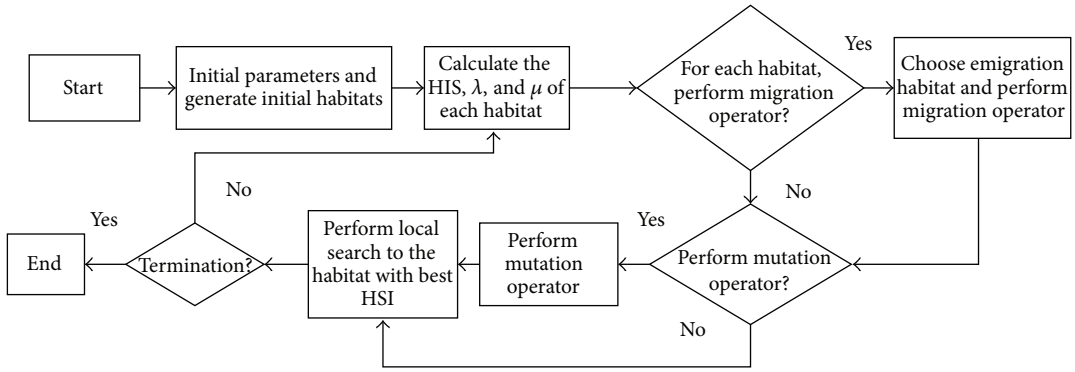

Figure 9: The framework of the MBBO for FJSSP.

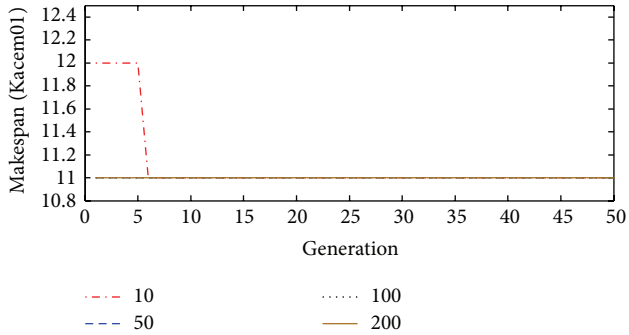

Figure 10: The performance of MBBO with different habitat sizes on Kacem01.

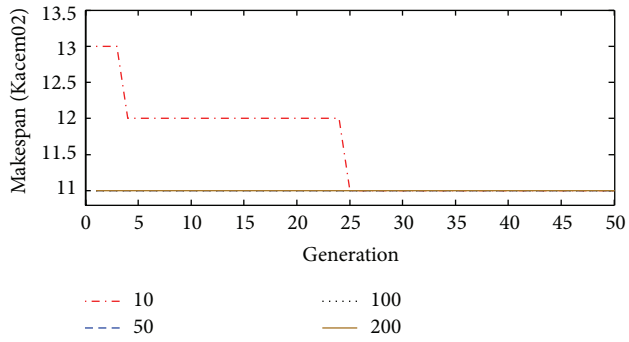

Figure 11: The performance of MBBO with different habitat sizes on Kacem02.

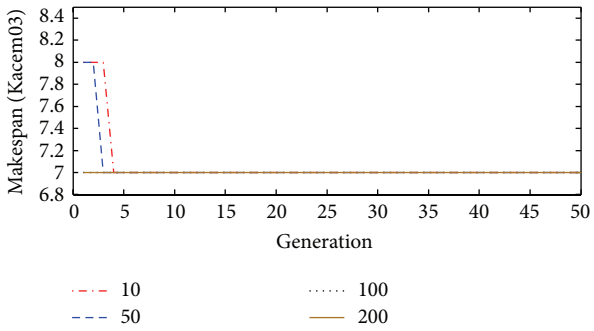

Figure 12: The performance of MBBO with different habitat sizes on Kacem03.

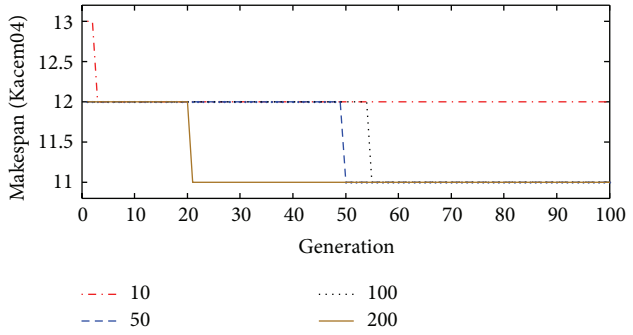

Figure 13: The performance of MBBO with different habitat sizes on Kacem04.

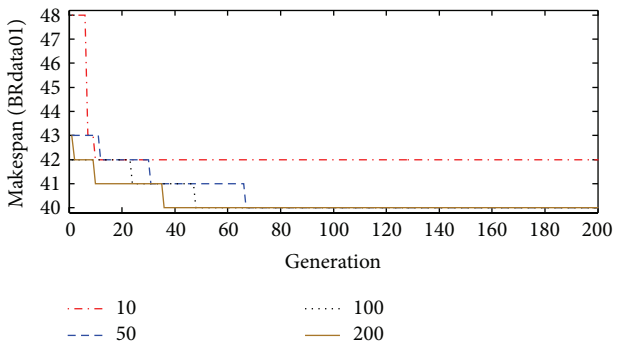

Figure 14: The performance of MBBO with different habitat sizes on BRdata01.

5.3. Results of BRdata Instances. Brandimart data consists of ten problems with number of jobs ranging from 10 to 20, number of machines ranging from 4 to 15, and number of operations for each job ranging from 5 to 15. Similarly, for each instance, we run MBBO 50 times independently with different seeds. The results are presented in Table 4, where the results of other three algorithms are directly taken from relative literatures.

The results in Table 4 showed that MBBO had a distinct advantage in solving BRdata. In MK01, MK02, MK03, MK 04, MK08, and MK09, MBBO achieved the known optimal values and acquired makespan much close to the optimal one

Table 2: The parameters settings for BRdata instances.

BRdata	$n * m$	NP	T
MK01	10 * 6	100	100
MK02	10 * 6	100	200
MK03	15 * 8	50	100
MK04	15 * 8	200	300
MK05	15 * 4	100	200
MK06	10 * 15	200	200
MK07	20 * 5	200	200
MK08	20 * 10	50	100
MK09	20 * 10	150	100
MK10	20 * 15	200	300

Table 3: The results for several algorithms on Kacem with makespan minimization.

	Instance	LEGA		BBO		BEDA		MBBO	
		C	AV(C)	C	AV(C)	C	AV(C)	C	AV(C)
Kacem	4 * 5	11	11	11	11	11	11	**11**	**11**
	10 * 7	11	11	11	11.34	11	11	**11**	**11**
	10 * 10	7	7.56	7	7.5	7	7	**7**	**7**
	10 * 15	12	12.04	12	12.56	11	11	**11**	**11**

Table 4: The results for several algorithms on BRdata with makespan minimization.

Instance	LEGA		BBO		BEDA		MBBO	
	C	AV(C)	C	AV(C)	C	AV(C)	C	AV(C)
MK01	40	41.5	40	41	40	41.02	**40**	**40**
MK02	29	29.1	28	28.25	**26**	27.25	**26**	**26.8**
MK03	—	—	204	204	204	204	**204**	**204**
MK04	67	67.34	64	66	**60**	63.69	**60**	**62**
MK05	176	178.1	173	173.5	**172**	173.38	173	173.5
MK06	67	68.82	66	66.5	**60**	**62.83**	61	63.1
MK07	147	152.9	144	144.25	**139**	**141.55**	140	142.75
MK08	523	523.34	523	523	523	523	**523**	**523**
MK09	320	327.74	310	310.75	307	310.35	**307**	**307.2**
MK10	229	235.72	230	232.75	**206**	**211.92**	221	221.75

in the rest instances. MBBO performed better than LEGA and BBO in all instances both effectively and stably. Comparing with LEGA, best results of MBBO in four instances are a little larger, but the average of results are much better in six out of ten instances, which indicates that MBBO owns the advantage of stability and is very robust.

5.4. Comparisons between MBBO and Other BBO Variants. To further verify the effectiveness of the machine-based shifting and local search in MBBO, we compared the MBBO with several variants on four Kacem instances and ten BRdata instances. BBO stands for the algorithm proposed by Habib et al. [32]. And the DBBO refers to the BBO with machine-based shifting decoding strategy. The habitat sizes for these three algorithms are set to 100 for Kacem instances and in accordance with Table 2 for BRdata instances. Figures 15, 16, and 17 depict the convergence curves of the best makespan of BBO, DBBO, and MBBO when solving Kacem02, Kacem04, and MK04, separately.

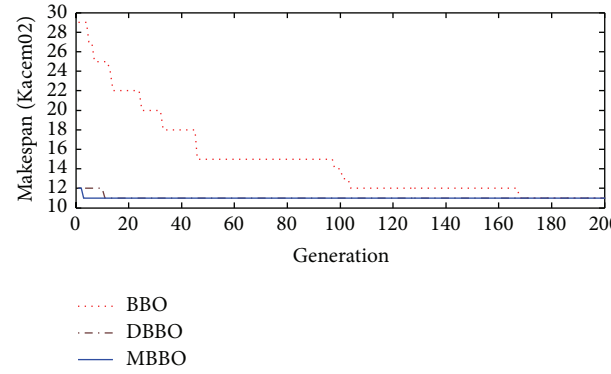

Figure 15: The comparison of convergence between BBO, DBBO, and MBBO on Kacem02.

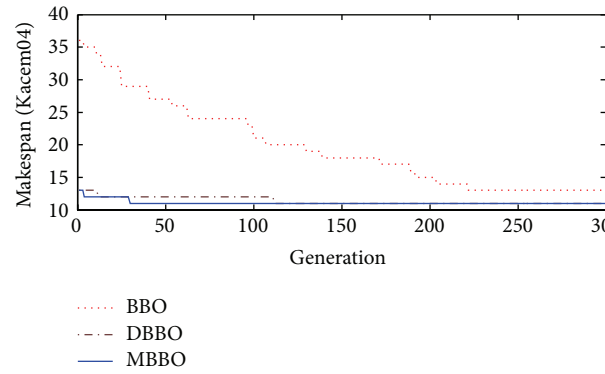

Figure 16: The comparison of convergence between BBO, DBBO, and MBBO on Kacem04.

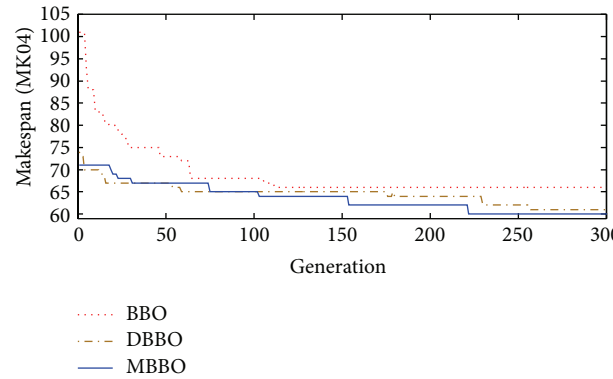

Figure 17: The comparison of convergence between BBO, DBBO, and MBBO on MK04.

Obviously, from the three figures, it can be seen that MBBO is of faster convergence speed than BBO and DBBO. Moreover, MBBO is more powerful to jump out of local optimal and reach the global optimal value. It is because that MBBO applied a local search to the best habitat instead of mutation operator, which keeps the good characters of better

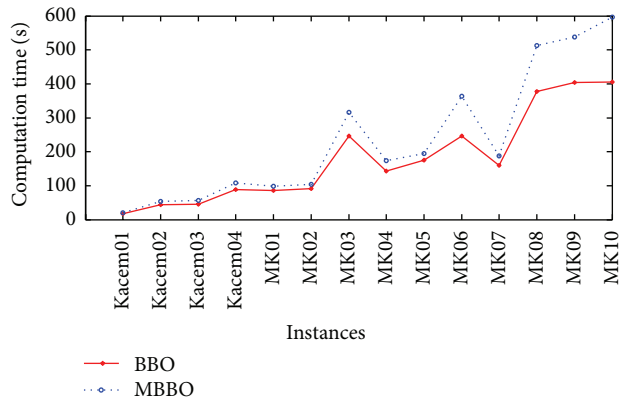

Figure 18: The comparison of computation time between BBO and MBBO on two datasets.

habitats, as well as the novel machine-based decoding process. The good performance of DBBO indicates effectiveness of machine-based shifting strategy.

It is worth noticing that the faster convergence of MBBO is at expense of computation time. Figure 18 shows that, under the same generations, the time MBBO spent is a little more than BBO on both Kacem and BRdata instances. However, as shown in Figure 16, MBBO reached the optimal makespan after 30 generations while BBO still did not converge to the optimal value 11 after 200 generations. Therefore, there is no doubt that MBBO has great superiority over BBO on convergence speed and quality of solutions in solving FJSSP.

6. Conclusions

In this paper, a modified biogeography-based optimization algorithm with machine-based shifting decoding strategy was proposed for solving the flexible job shop scheduling problem with makespan minimization. The MBBO algorithm applied a double coding scheme with operation sequence and machine assignment vector. And a special machine-based shifting strategy was proposed to decode the vectors to a scheduling solution. Besides, several discrete operators in BBO, such as emigration, immigration, and mutation, were designed. Moreover, a special local search, instead of mutation, was applied to the best habitat to speed up the search process while maintaining the good characters of better habitats. The parameters were investigated and chosen by experiments. Comparisons on Kacem and BRdata instances with several existing famous algorithms indicate the superiority of the proposed MBBO algorithm in terms of effectiveness and efficiency.

More comprehensive studies can be applied to extend MBBO algorithm. Other possible criteria in multiobjective optimization will be considered. Furthermore, more local search methods will be analyzed to integrate to the MBBO.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The author appreciates the support of scientific research start-up fund provided by Shanghai Dianji University and Climbing Peak Discipline Project of Shanghai Dianji University (Project no. 15DFXK01).

References

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

[2] 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.

[3] 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.

[4] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers and Operations Research*, vol. 35, no. 10, pp. 3202–3212, 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, UK, September 2005.

[6] N. B. Ho and J. C. Tay, "GENACE: an efficient cultural algorithm for solving the flexible job-shop problem," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, vol. 2, pp. 1759–1766, IEEE, June 2004.

[7] N. B. Ho and J. C. Tay, "LEGA: an architecture for learning and evolving flexible job-shop schedules," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, vol. 2, pp. 1380–1387, September 2005.

[8] 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.

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

[10] D. Lei and X. Guo, "Swarm-based neighbourhood search algorithm for fuzzy flexible job shop scheduling," *International Journal of Production Research*, vol. 50, no. 6, pp. 1639–1649, 2012.

[11] L. Wang and C. Fang, "An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem," *Computers and Operations Research*, vol. 39, no. 2, pp. 449–460, 2012.

[12] L. Wang, S. Wang, Y. Xu, G. Zhou, and M. Liu, "A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem," *Computers & Industrial Engineering*, vol. 62, no. 4, pp. 917–926, 2012.

[13] 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.

[14] 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 and Cybernetics Part C: Applications and Reviews*, vol. 32, no. 1, pp. 1–13, 2002.

[15] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach based on uniform design and fuzzy evolutionary

algorithms for flexible job-shop scheduling problems (FJSPs)," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 7, pp. 1–7, IEEE, Hammamet, Tunisia, October 2002.

[16] N. B. Ho, J. C. Tay, and E. M.-K. Lai, "An effective architecture for learning and evolving flexible job-shop schedules," *European Journal of Operational Research*, vol. 179, no. 2, pp. 316–333, 2007.

[17] 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.

[18] J. Gao, L. Sun, and M. Gen, "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems," *Computers & Operations Research*, vol. 35, no. 9, pp. 2892–2907, 2008.

[19] 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.

[20] J.-Q. Li, Q.-K. Pan, and Y.-C. Liang, "An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems," *Computers & Industrial Engineering*, vol. 59, no. 4, pp. 647–662, 2010.

[21] L. Wang, S. Y. 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.

[22] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[23] H. Kundra and M. Sood, "Cross-country path finding using hybrid approach of PSO and BBO," *International Journal of Computer Applications*, vol. 7, no. 6, pp. 15–19, 2010.

[24] K. Jamuna and K. S. Swarup, "Multi-objective biogeography based optimization for optimal PMU placement," *Applied Soft Computing*, vol. 12, no. 5, pp. 1503–1510, 2012.

[25] S. Singh, E. Mittal, and G. Sachdeva, "Multi-objective gain-impedance optimization of Yagi-Uda antenna using NSBBO and NSPSO," *International Journal of Computer Applications*, vol. 56, no. 15, pp. 1–6, 2012.

[26] R. MacArthur and E. Wilson, *The Theory of Biogeography*, Princeton University Press, Princeton, NJ, USA, 1967.

[27] R. Nakano and T. Yamada, "Conventional genetic algorithm for job shop problems," in *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA '91)*, pp. 474–479, San Diego, Calif, USA, July 1991.

[28] H. P. Ma, X. Li, and S. D. Lin, "Analysis of migratorate models for biogeography-based optimization," *Journal of Southeast University (Natural Science Edition)*, vol. 39, supplement, pp. 16–21, 2009 (Chinese).

[29] C. Y. Zhang, Y. Q. Rao, P. G. Li, and X. Shao, "Bilevel genetic algorithm for the flexible job-shop scheduling problem," *Chinese Journal of Mechanical Engineering*, vol. 43, no. 4, pp. 119–124, 2007 (Chinese).

[30] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach based on uniform design and fuzzy evolutionary algorithms for flexible job-shop scheduling problems (FJSPs)," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 7, pp. 1–7, Yasmine Hammamet, Tunisia, October 2002.

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

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