Int J Adv Manuf Technol
DOI 10.1007/s00170-011-3437-9

ORIGINAL ARTICLE

A new biogeography-based optimization (BBO) algorithm for the flexible job shop scheduling problem

Seyed Habib A. Rahmati · M. Zandieh

Received: 5 February 2011 / Accepted: 6 June 2011

Abstract Biogeography-based optimization (BBO) algorithm is a new kind of optimization technique based on biogeography concept. This population-based algorithm uses the idea of the migration strategy of animals or other species for solving optimization problems. In this paper, the BBO algorithm is developed for flexible job shop scheduling problem (FJSP). It means that migration operators of BBO are developed for searching a solution area of FJSP and finding the optimum or near-optimum solution to this problem. In fact, the main aim of this paper was to provide a new way for BBO to solve scheduling problems. To assess the performance of BBO, it is also compared with a genetic algorithm that has the most similarity with the proposed BBO. This similarity causes the impact of different neighborhood structures being minimized and the differences among the algorithms being just due to their search quality. Finally, to evaluate the distinctions of the two algorithms much more elaborately, they are implemented on three different objective functions named makespan, critical machine work load, and total work load of machines. BBO is also compared with some famous algorithms in the literature.

Keywords Biogeography-based optimization algorithm · Flexible job shop scheduling problem · Makespan · Machine work load

S. H. A. Rahmati
Industrial and Mechanical Engineering Faculty,
Islamic Azad University, Qazvin Branch,
Qazvin, Iran

M. Zandieh (✉)
Department of Industrial Management,
Management and Accounting Faculty, Shahid Beheshti University,
G.C. Tehran, Iran
e-mail: m_zandieh@sbu.ac.ir

1 Introduction

Flexible job shop scheduling problem (FJSP) as a branch of production planning problems is a modified version of job shop scheduling problem (JSP) [1]. In JSP, operations can be processed on a predetermined and fixed processing order through all machines. However, in FJSP, this assumption is released and an operation is allowed to be processed by any machine from a given set. FJSP is more complex than JSP because of the additional need for determining the assignment of operations to machines. Therefore, since JSP belongs to the NP-hard class of problems, FJSP as a more complicated problem is also categorized in this class.

FJSP, to minimize a predefined objective, faces two main difficulties, including (1) assigning each operation to a machine and (2) scheduling the assigned operations of each machine. Therefore, two sub-problems can be considered for FJSP, which are called (1) machine assignment sub-problem and (2) operation sequencing sub-problem.

For solving a FJSP, some studies considered these two sub-problems separately. This approach, which is named hierarchical approach, divides a hard problem into two simpler sub-problems. Brandimarte [2] solved an operation sequence sub-problem by using some dispatching rules and solved the machine assignment sub-problem through a tabu search (TS) algorithm. The TS algorithm was also used by Barnes and Chambers [3] in their hierarchical approach. Xia and Wu [4] used simulated annealing for operation sequence and particle swarm optimization (PSO) for the machine assignment sub-problem within a multi-objective FJSP.

On the other hand, more studies consider this two sub-problems simultaneously and solve FJSP through an integrated approach. Generally, integrated approaches are more complicated, but result in better solutions. Hurink et al. [5] and Scrich et al. [6], in their integrated approach,

Published online: 15 September 2011

used the TS algorithm for solving FJSP. Chen et al. [7] proposed a genetic algorithm (GA) with a special type of chromosome in which for each sub-problem of the FJSP a separate vector was considered. It means that their chromosome consisted of two vectors including the machine assignment vector and the operation sequence vector. Mastrolilli and Gambardella [8] proposed two neighborhood structures in their TS technique. TS was also used by Saidi-Mehrabad and Fattahi [9]. Kacem et al. [10–12] studied single- and multi-objective FJSP based on the localization approach. Mati et al. [13] developed a greedy algorithm for FJSP. Ho et al. [14] proposed a learnable GA for FJSP. Gao et al. [15] developed a hybrid genetic algorithm to solve multi-objective FJSP. Gao et al. [16] developed the general PSO algorithm for solving FJSP. Zhang et al. [17] developed an algorithm called VNGA in which a variable neighborhood search algorithm as a local search is used for improving the quality of the GA's solutions. Zhang et al. [18], to deal with a multi-objective FJSP (MOFJSP), proposed a hybrid version of the PSO algorithm in which TS algorithm is used as local search. Zhang et al. [19, 20] developed a combination of TS with GA and variable neighborhood genetic algorithm for solving MOFJSP, respectively. They [21] also developed an effective GA (eGA) with a special initialization method. Their algorithm reached most of the best solutions that were found in the literature and also improved some of them. Wang et al. [22] proposed a multi-objective GA which utilizes entropy and immune concept for MOFJSP.

Biogeography-based optimization (BBO) algorithm, such as GA or PSO algorithm, is a naturally inspired algorithm in which the migration strategy of species is used for solving engineering problems. Biogeography science can be referred to the studies of two naturalists named Alfred Wallace [23] and Charles Darwin [24]. Robert MacArthur and Edward Wilson [25] started a mathematical modeling of biogeography in 1960 and introduced it as an important area of research. These types of mathematical models represent how species migrate among different islands, how new species arise, and how species become extinct. In the literature of biogeography, an island is referred to as any habitat that is geographically isolated from other habitats. It should be mentioned that in this paper, island and habitat are considered the same. Although BBO is a naturally inspired algorithm, it has some fundamental distinctions from common natural algorithms such as GA, PSO, or ant colony optimization. In BBO, the initial population is not discarded among different generations. Instead, the migration concept is used to modify the population. As another distinction, in each generation, the fitness function is not used directly to modify the population, and BBO used fitness to determine the immigration and emigration rates.

BBO was firstly presented by Simon [26] for solving engineering problems. In his first paper, Simon introduced the main idea, definitions, and steps of BBO and proved its good performance. Since then, many researchers have used BBO in their studies. In his next paper, Simon [27] introduced his algorithm much more simply by means of a simple version of BBO and analyzed its population by means of probability theory. Then, he showed how a BBO with a low mutation rate outperforms GA with a low mutation rate. Du et al. [28] improved the BBO's performance by inserting distinctive features of other heuristic algorithms into the BBO. Ergezer et al. [29], by using opposition-based learning (OBL) alongside BBO's migration rates, proposed a new version of BBO which is called oppositional BBO. They mathematically proved that among all OBL methods, their algorithm has the highest expected probability to get close to the problem's solution. Ma and Chen [30] explored the performance of six migration models on BBO by generalizing the equilibrium species count of biogeography theory and showed that the sinusoidal migration model outperforms other models. Ma and Simon [31] proposed a new blended crossover and mutation operator in which a solution is adapted by a linear combination of itself with another solution. They also developed their blended BBO for constrained problems. BBO has also shown a good performance on real-world optimization problems such as classification of satellite images [32], groundwater resource detection [33], or even on an economical problem to solve the economic load dispatch problem [34].

In this paper, the BBO algorithm is developed and introduced to scheduling area, especially for FJSP. To explain the performance of this algorithm more explicitly, it is compared with a newly developed genetic algorithm. In both of these algorithms, we have tried to implement similar neighborhood structures (taken from [22]) in order to minimize the impact of neighborhood structures on the performance of the algorithms. Therefore, different results of the algorithms are just due to their search ability. In addition, since different objectives can change the performance of the algorithms, our algorithms are implemented for three common objective functions, namely, makespan, critical machine work load, and total work load of machines. Finally, this simple biogeography-based algorithm, which is not a hybrid with other algorithms nor included learning ability, is compared with three famous algorithms [2, 14, 21] reported in the literature to show which one of them [21] introduced the most number of best solutions that have not been obtained until now. The algorithms are tested on different classical problems of FJSP and the results are presented.

The rest of the paper is organized as follows. In the next section, classical FJSP is introduced. In Section 3, the proposed BBO is developed. A similar GA is also explained in this section. Section 4 presents and discusses computational results; finally, Section 5 concludes the paper and suggests some future work opportunities.

Table 1 Example of FJSP with three jobs and four machines

FJSP		Processing times			
		M1	M2	M3	M4
J1	O1,1	2	–	1	6
	O1,2	5	3	–	2
	O1,3	–	2	4	–
J2	O2,1	7	–	–	11
	O2,2	4	4	12	8
J3	O3,1	2	–	7	9
	O3,2	3	5	8	1
	O3,3	4	3	–	5

2 Problem definition

A FJSP is a scheduling model in which n jobs J (J_i, $i \in \{1,2,\ldots,n\}$) are supposed to be processed on m machine M (M_k, $k \in \{1,2,\ldots,m\}$). For each job, one or more operations (O_{ij}, $j \in \{1,2,\ldots,n_i\}$) (where n_i represents the total number of operations for job J_i) can be considered. For each operation of one specific job, a predetermined set of capable machines is considered, and each operation (O_{ij}) of that job (J_i) can be processed by one machine out of its set of capable machines (M_{ij}). For job J_i, P_{ijk} denotes the processing time of operation j (O_{ij}) on machine k. Therefore, FJSP has two main goals, including assigning each operation to a suitable machine and determining the sequence of the assigned operation on each machine in order to minimize common objective functions like maximal makespan ($C_{\max}$), critical machine work load (CWL), or total work load (TWL) of machines. For more detail, Ho et al. [14] can be consulted. The following assumptions are also considered:

1. Operations of each job have a fixed and predetermined order.
2. Jobs have the same priority.
3. There is no priority restriction among operations of different jobs.
4. Jobs are released at time 0 and machines are available at time 0.
5. Move time between operations and setup time of machines are ignored.
6. At any specific time, only one job can be processed on each machine.
7. During the process, operations cannot be broken off.

The FJSP, which consisted of three jobs and four machines, is shown in Table 1. In Table 1, the numbers present the processing times of operations on different machines of their set of capable machines, and symbol "–" means the operation cannot be processed on a corresponding machine.

3 The proposed algorithm

3.1 Biogeography-based optimization

BBO is a new naturally inspired algorithm that is based on biogeography science. Biogeography, as a subset of biology, studies the distribution of species over space and time [26, 27]. Simon [26] develops biogeography science for solving optimization problems. BBO, just like GA or PSO, is a population-based algorithm in which a population of candidate solutions (individuals) is used for solving a global optimization problem [26]. In GA, each chromosome is considered as an individual and has its fitness value. Likewise, in BBO, each habitat is considered as an individual and has its habitat suitability index (HSI) instead of fitness value to show the degree of its goodness. High-HSI habitat represents a good solution and low-HSI habitat represents a poor solution. Solution features emigrate from high-HSI habitats (emigrating habitat) to low-HSI habitats (immigrating habitat). In other words, low-HSI habitats

Table 2 BBO's definitions and concepts vs. GA's definitions and concepts

	BBO	GA
1	Population-based	Population-based
2	Habitat (individual)	Chromosome (individual)
3	SIV	Gen
4	Habitats consisted of SIV	Chromosomes consisted of Gens
5	Mutation operator	Mutation operator
6	Migration operators (immigration and emigration)	Crossover operator
	No reproduction	Reproduction with P_{re} rate
7	Good solution is characterized by high HSI	Good solution is characterized by high fitness
8	A good habitat is one which has more diversity and species	A good chromosome is the one which has more value of fitness function
9	No individual of initial population discard during iterations but it is modified	Initial individuals can be discarded by GA operators during iterations

Operation Sequence	3	1	2	3	1	2	3	1
	O_{31}	O_{11}	O_{21}	O_{32}	O_{12}	O_{22}	O_{33}	O_{13}
Machine Assignment	1	2	2	1	2	3	2	4
	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{31}	O_{32}	O_{33}
	First Job			Second Job		Third Job		

Fig. 1 A two-vector habitat representation for three jobs, four machines, and eight operations for FJSP

accept a lot of new features from high-HSI habitats through an immigration process. Therefore, the migration operators, which are emigration and immigration, are used to improve and evolve a solution to the optimization problem. Generally, in an optimization problem like FJSP, the objective function is considered as HSI and the evolutionary procedure of BBO is to determine those solutions which maximize the HSI by using the immigration and emigration features of the habitats.

Table 2 represents and redefines the extent of BBO on FJSP. Meanwhile, since GA is considered as a popular population-based algorithm, Table 2 compares BBO's characteristics with GA's characteristics. The performance comparison of these two algorithms is also considered to explain BBO much more explicitly.

As is clear from Table 2, BBO has two main operators, which are migration (including emigration and immigration) and mutation. For implementing these two operators, there are different options, but the one option that is used for each of them is described in Sections 3.1.4 and 3.1.5.

3.1.1 Initialization of the BBO algorithm

For initializing this algorithm, the method that Wang et al. [22] proposed is used. In this approach, first, the operation sequence is generated randomly, and then from the set of capable machines, two machines are selected for each operation. Finally, if a random generated number (Rand ∈ [0,1]) is <0.8, a machine with a shorter process time is chosen; otherwise, a machine with a longer process time is chosen.

3.1.2 Representing and decoding scheme of habitats

Although in this algorithm an individual is called habitat, the performance and structure of an individual is just like the chromosomes of GA. Consequently, we use a common representation of the literature that Wang et al. used in their GA [22]. This representation includes two vectors: one vector used for representing the processing sequence of all operations and another vector used for representing the assignment of a suitable machine to each operation. The integration of these two vectors creates a feasible solution for FJSP. A scheme of this chromosome (according to Table 1) is shown in Fig. 1. As can be seen in this figure, the length of both vectors is the same as the number of all operations of the jobs.

The sequence vector is a vector like [31231231] (according to the example of presented in Table 1) in which each number is repeated at equal times as the number of related job operations. For example, since the first job has three operations, number 1 is repeated three times that *kth*1 shows the placement of the *kth* operation of the first job. Therefore, the corresponding sequence can be represented as [O_{31}, O_{11}, O_{21}, O_{32}, O_{12}, O_{22}, O_{33}, O_{13}].

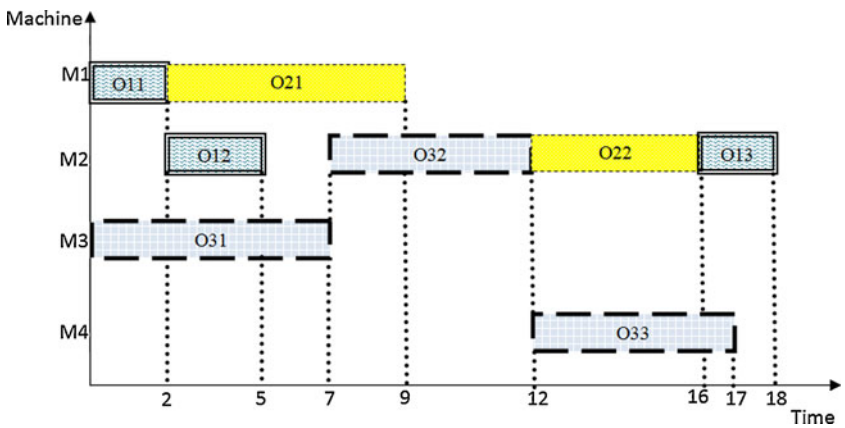

Fig. 2 Decoding Gantt chart for the habitat of Fig. 1

The assignment vector is a vector like [122-12-324] (according to example of presented in Table 1) that shows the assigned machine of each operation successively. It means that the first three suitability index variable (SIV) show the assigned machines to operations of the first job successively, the next part for the second job, and so on. Therefore, the assigned machines to each SIVof operation sequence vector is as:

$$\begin{bmatrix}(O_{31},M_3),(O_{11},M_1),(O_{21},M_1),(O_{32},M_2),\\(O_{12},M_2),(O_{22},M_2),(O_{33},M_4),(O_{13},M_2)\end{bmatrix}.$$

Each habitat should be transferred to a solution of the problem during a process called decoding. For more notational convenience, assume that each operation O_{ij} (SIV) of the habitat is denoted by OP. Then, its process and start time are denoted by p^{OP} and S^{OP} respectively, and complementation time can be calculated as $S^{\mathrm{OP}} + p^{\mathrm{OP}}$. Now, denoting job and machine predecessor as JP and MP, respectively, the start time of any new operation can be calculated as Eq. 1. Of course, it is assumed that all jobs are started at time 0 and S_{JP}^{OP} and S_{MP}^{OP} in the beginning are assumed as zero.

$$S^{OP} = \max\left\{\left(S_{JP}^{OP} + P_{JP}^{OP}\right), \left(S_{MP}^{OP} + P_{MP}^{OP}\right)\right\} \quad (1)$$

The decoding process starts from the first SIV of the operation sequence vector by assigning a corresponding machine from the assignment vector to that operation. The operation should be located in the earliest capable time of that machine (which, for first operation, the earliest time is time 0). For other SIVs or operations of the operation sequence vector from left to right, a similar process is done, but locating the operation on the earliest capable time of the corresponding machine is in accordance with Eq. 1. Following this decoding process, create an active schedule for each habitat or solution of FJSP. The decoding process for the habitat of Fig. 1 is done in Fig. 2 schematically with a Gantt chart.

3.1.3 Selection strategies

This step is one of the distinctive steps of BBO with other algorithms, which is executed through two different strategies, one for migration and one for mutation. Details of these strategies are explained in the two next subsections.

```
Select Rand ( Rand ∈ [0 1] )
If Rand < λ_i
   For j= 1 to n
      Select H_j through Rolette wheel process
      If H_j is selected
         Using H_i and H_j, the migration operator is done
      End
   End
End
```

Fig. 3 Selection strategy of the migration operator in the BBO algorithm

```
Select H_i(SIV) according to mutation probability
If H_i(SIV) is selected
   The mutation operator is done
End
```

Fig. 4 Selection strategy of the mutation operator in the BBO algorithm

Selection strategies of migration To explain these strategies, the first two new notations should be defined. These two notations, which are denoted by λ_i and μ_j, represent the immigration and emigration rates, respectively. Now, the solutions are selected for immigrating or emigrating according to these two rates. Of course, it should be noticed that these rates are explained in the next subsection (Section 3.1.4) more completely.

According to the concept of the BBO algorithm, during the migration process, we face two types of selection. Firstly, we should determine whether a special habitat H_i should be immigrated or not. To do so, a simple comparison of λ_i with a random number is done. Secondly, we should select habitat H_j for emigrating to H_j. Details of the selection algorithm for migration are shown in Fig. 3.

Selection strategies of mutation Figure 4 explains how the mutation selection strategy is performed in the BBO algorithm.

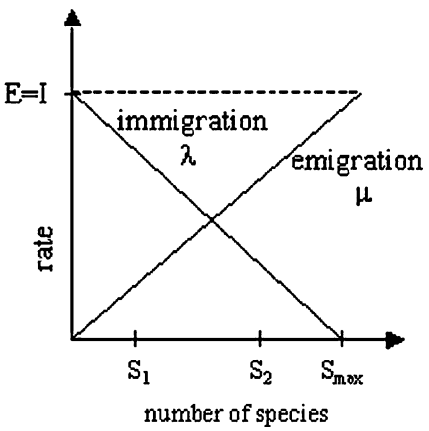

Fig. 5 Variation of the immigration and emigration rates for different species abundance in a habitat [26]

Division: Divide the set of jobs into two non empty groups G1 and G2

Direct copies: According to members of G1 determine direct copies (same position) of immigrating habitat to its modified version

Indirect copies: Members of G2 should be copied from emigrating habitat to modified version of immigrating habitat in the same order

Fig. 6 IPOX operator for operation sequence migration [22]

3.1.4 Migration operator

Migration is a probabilistic operator that is used for modifying each solution H_i by sharing features among different solutions. The idea of a migration operator is based on the migration in biogeography which shows the movement of species among different habitats. Solution H_i is selected as immigrating habitat with respect to its immigration rate λ_i, and solution H_j is selected as emigrating habitat with respect to its emigration rate μ_j. It means that the probability that a solution is selected for immigrating or emigrating depends on its immigration rate λ_i or emigration rate μ_j; the migration process can be shown as:

$$H_i(\text{SIV}) \leftarrow H_j(\text{SIV}) \tag{2}$$

The immigration rate λ_i and emigration rate μ_j and the species abundance of a habitat can be modeled as Fig. 5 (taken from [26]). As was mentioned, the features of high-HSI solutions (good solutions) tend to emigrate to low-HSI solutions (poor solutions). This tendency causes, by increasing the species, as in Fig. 5, the immigration rate to decrease and the emigration rate to increase. In this figure, E and I denote the maximum of immigration and emigration rates, respectively, $S_{\max}$ denotes the largest number of species that the habitat can support, and S_0 denotes the equilibrium point in which the immigration rate and the emigration rate are equal. Although in this figure the immigration and emigration are considered linear, they can be replaced with other curves if needed. It should be noticed that E and I are mostly set to 1.

Equation 2 shows how a feature or SIV of a solution is adjusted with a feature or SIV of another solution through migration operation. As mentioned in Table 1, a SIV shows a feature of the solution (just like a gene in GA) and is used as a search variable. Therefore, a set of all possible SIVs is considered as the search space from which a solution is determined.

After calculating the HSI for each solution H_i, the immigration rate λ_i and the emigration rate μ_j can be evaluated as Eqs. 3 and 4, respectively. It means that these two rates are the functions of fitness or HSI of the solution. Since, according to the biogeography, the SIVs of a high-HSI solution tend to emigrate to low-HSI solutions, a high-HSI solution has a relatively high μ_j and low λ_i, while in a poor solution, a relatively low μ_j and a high λ_i are expected.

$$\lambda_i = I\left(1 - \frac{k_i}{n}\right) \tag{3}$$

$$\mu_i = E\left(\frac{k_i}{n}\right) \tag{4}$$

In Eqs. 3 and 4, k_i represents the rank of the ith habitat after sorting all habitats according to their HSIs and n represents the size of the population. It is clear that since more HSI represents a better solution, more k_i represents the better solution. Therefore, the 1th solution is the worst and the nth solution is the best. Now, by calculating λ_i and μ_j, the selection strategy and migration operator are done as in Fig. 10.

Another parameter that needs to be calculated is the probability of existence of S species in the habitat, which is denoted by P_S. This parameter is obtained through an equation like Eq. 5 because to model changes from time t to $t + \Delta t$, one of the three following states should happen:

1. S species at time t and this amount does not change during $[t,t + \Delta t]$.
2. $S - 1$ species at time t and one immigrating during $[t,t + \Delta t]$.

Random String Generation: Generate a random string (Rand$\in$ [0 or1]) as long as habitats

Copy Type1: If Rand=0 SIVs copies directly from immigrating habitat to its modified version

Copy Type2: If Rand=1 SIVs copies directly from emigrating habitat to modified version of immigrating habitat

Fig. 7 MPX operator for machine assignment migration [22]

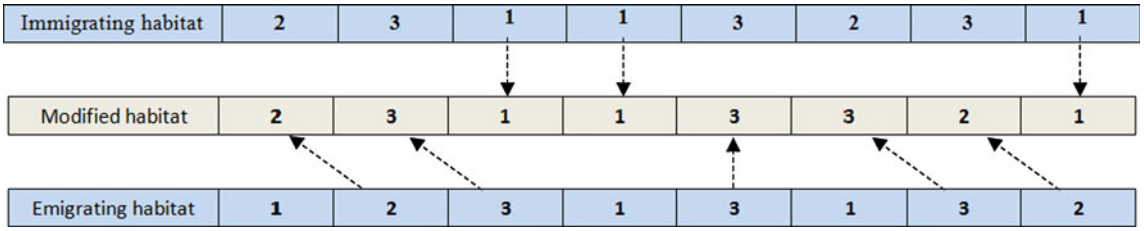

Fig. 8 IPOX operator of operation sequence vector for G1; {1} and G2: {2, 3}

3. $S+1$ species at time t and one emigrating during $[t,t+\Delta t]$.

$$P_S(t+\Delta t) = P_S(t)(1-\lambda_S\Delta t-\mu_S\Delta t) + P_{S-1}\lambda_{S-1}\Delta t + P_{S+1}\mu_{S+1}\Delta t \quad (5)$$

By solving Eq. 5 in a steady state, P_i is calculated as Eq. 6 [26], where v_i is a function of population size (n) and formulated by Eq. 7. i' in Eq. 7 is the smallest integer that is greater than or equal to $\frac{n(n+1)}{2}$.

$$P_i = \frac{v_i}{\sum_{i=1}^{n} v_i} \quad (6)$$

$$v_i = \begin{cases} \dfrac{n!}{(n+1-i)!(i-1)!} i = 1,2,3\ldots,i' \\ v_n + 2 - ii = i'+1\ldots,n+1 \end{cases} \quad (7)$$

Now, according to what was mentioned, migration operator should be done. To do so, after selecting the immigrating and emigrating habitats, the migration operator is done just like the crossover operator of GA. For migrating our two-vector representation, two popular crossover operators called improved precedence operation crossover (IPOX) and multipoint preservative crossover (MPX), developed by Zhang et al. [35] (taken from [22]), are used for the operation sequence and machine assignment, respectively. Whenever each of these operators is processed, the other one is stopped. It means that when MPX is processed on operation sequence, IPOX is unchanged and vice versa. Figures 6 and 7 represent the IPOX operator and the MPX operator, respectively, and Figs. 8 and 9 illustrate these operators schematically.

3.1.5 Mutation operator

In BBO, mutation is a probabilistic operator which is used for modifying one or more randomly selected SIV of a solution based on its priori probability of existence P_i. In BBO, just like GA, this operator is used for increasing diversity among the population. In this algorithm, the mutation probability m_i is calculated according to the solution probability [26], as in Eq. 8. Therefore, mutation probability and solution probability are proportioned inversely.

$$m_i = m_{\max}\left(1-\frac{p_i}{p_{\max}}\right) \quad (8)$$

Now, according to the mutation probability (m_i), the selection strategy and mutation operator can be done. Again, for each vector of the habitat, a special mutation operator is determined (taken from [22]). Figures 10, 11, and 12 illustrate the mutation operator for both vectors of each habitat.

3.1.6 Main algorithm of BBO algorithm

The main algorithm of the BBO is shown in Fig. 13.

3.2 The GA

As mentioned, to assess the performance of the proposed BBO much more clearly, it is compared with a GA. To do so, the operators of GA are considered just like the BBO's operator to minimize the impact of the different operators on the performance of the algorithms. Therefore, in the

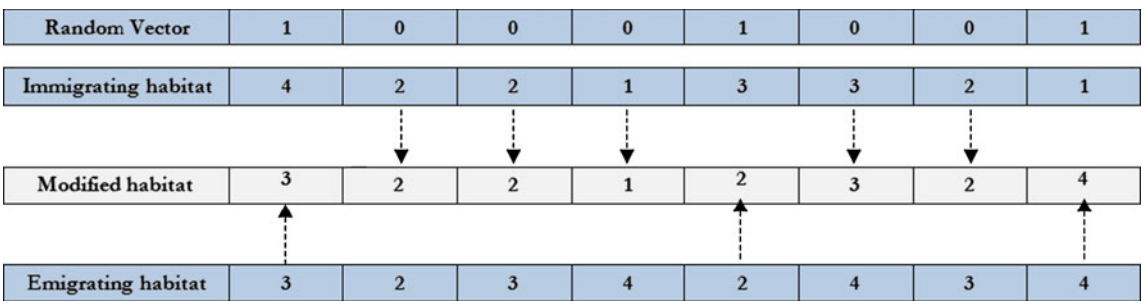

Fig. 9 MPX operator of the machine assignment vector

If the habitat should be mutated do mutation as follows:

1-Operation sequence mutation: Choose a SIV randomly and insert it in a position before a random operation.

2-Machine assignment mutation: Choose two SIVs randomly and change each SIV with a machine from the set of capable machines of that SIV (or operation)

Fig. 10 Steps of mutation for each chosen habitat [22]

proposed GA, the initialization method is the same as the BBO, the crossover is like the migration of the BBO (MPX and IPOX), and mutation structures are also the same. It is worth reminding that their most difference is in their selection strategies. In GA, the selection strategy is tournament selection [22] in which two parents are selected and a random number is generated (Rand ∈ [0,1]). If this random number is <0.8, a better parent is selected; otherwise, the worst parent is selected. It should be mentioned that these two parents are not deleted from the population and can be selected again as parents.

4 Computational results

In this section, we show how our non-equipped BBO (without adding any hybrid algorithm or learning ability) is comparable with three famous equipped algorithms named TS by Brandimarte [2], learnable GA (LEGA) by HO et al. [14], and eGA by Zhang et al. [21]. The algorithm of Zhang et al. presents the most number of best solutions that have been obtained in the literature and also improved some of them. As mentioned, BBO is also compared with simple GA that have most the similar operators with BBO (except in selecting) to show the search quality of algorithms more explicitly. Of course, since the algorithms differ by the different fitness function, we compare these two algorithms for different objectives including makespan, critical machine work load, and total work load of machines. Comparison of BBO with the similar GA of the paper is also done through some illustrations, which are plotted in Fig. 14. The algorithms are tested on Kacem [10, 12], Brandimart [2], and Barnes and Chamber [3] library. The results are presented in Tables 3, 4, and 5 and Tables 6 and 7 in the Appendix and run in MATLAB on a PC with 4-GB RAM and 2.4-GHz CPU. It should be noticed that for the Kacem and Brandimart library, the population and iteration size are set to 200 and for Barn library are set to 350 and 300, respectively. Besides, P_c = 85% (crossover rate) and P_m = 10% (mutation rate).

Tables 3 and 4, which are related to the Brandimarte library (BRdata) and Barnes and Chambers library (BCdata), respectively, show how this simple version of BBO can acquire solutions near or equal to the best solutions that have been obtained in the literature by Zhang et al. [21]. Therefore, it is expected that by adding an immune operator, improving the operator of BBO, or using any other way that is used for making an algorithm more intelligent, the BBO can easily reach higher quality solutions. Assessing this type of improvement is one of our future works for this algorithm. A comparison of BBO with other algorithms (Table 5) shows that it is at least as good as Ho's and Brandimart's algorithm. It means that our BBO outperforms Ho's and Brandimarte's algorithms on their whole tested library. In these tables, the following notations are used:

n	Number of jobs
m	Number of machine types
Flex	Average number of equivalent machines per operation
LB	Lower bound for the problem
C_{max}	Best solution for makespan
Av (C_{max})	Average of the solution for makespan
t	Average of the complementation times (roundup time)
T_0	Total number of operations

Tables 6 and 7 in the Appendix and Fig. 14 compare the proposed BBO with a newly proposed GA in which neighborhood structures are designed like the BBO's neighborhood structures. To do so, the migration strategy of BBO and the crossover operator of GA is designed

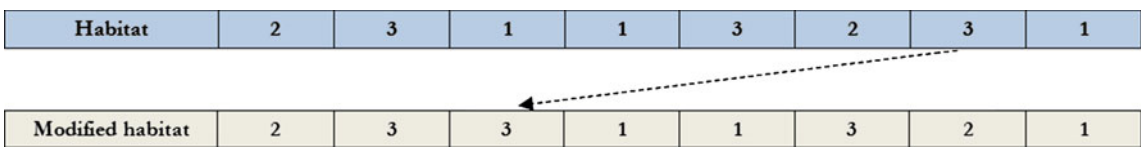

Fig. 11 Scheme of mutation operator of the operation sequence vector

Fig. 12 Scheme of mutation operator for the machine assignment vector

similarly. The mutation operators of both algorithms are also designed similarly. Therefore, the impact of the different operators of the two algorithms is extremely minimized, and most of the differences in the results of the two algorithms are just due to their different search process. On the other hand, since our algorithms are programmed in MATLAB, this comparison is vital. In this way, we can compare the BBO with a well-known algorithm (GA) in a similar situation. Table 6 in the Appendix compares these two algorithms according to the value of three different objective functions. In this table, every three rows belongs to one specific test problem in which the first, second, and third rows present the results of algorithms for makespan ($C_{\max}$), TWL of machines, and CWL, respectively. The last two columns of the table (named Count) count the amount of being better for the three criteria (minimum, average, and maximum). For instance, in MK01, the BBO at one criterion is better (the average of obtained results). In Table 6 in the Appendix, two algorithms are completely comparable. However, in Table 7 in the Appendix, which is designed like Table 6 (in the Appendix), but for the computational time of the algorithms, the BBO proves itself clearly, and although the effectiveness of the algorithms is very similar as in Table 6 (in the Appendix), the BBO shows better efficiency in Table 7 (in the Appendix). Figure 14, for the three different objective functions on the test problem 15*10 of Kacem, presents the performance of the two algorithms simultaneously and schematically. In Fig. 14, for each algorithm, two curves are plotted, one figure for best of the objective function (vs. time) and one figure for average of the objective function (vs. time).

It is worth mentioning that all operators of our BBO or GA are designed like the algorithm of Wang et al. [22]. On the other hand, their algorithm is a multi-objective GA which implemented the entropy and immune concept.

Parameter setting: $E = 1, I = 1, m_{max} = 1$, Pop. size=200, Num. iteration=200
Initialization: Generating habitats as size as population size (Like GA)
Population evaluation: evaluate habitats just like chromosomes of the GA
Sort the population according to the HSI of the habitats increasingly
For i=1: Num. iteration
  Calculate λ_j, μ_j, P_j, & m_j according to habitat's rank
  For j=1: Pop. Size
    Generate $Rand \in [0,1]$
    If Rand<=λ_j
      $H_i(SIV)$ = Choose a habitat randomly through a Roulette wheel of μ $(\mu_1, \mu_2, \ldots, \mu_n)$
      Execute migration operator like crossover of GA $H_j(SIV) \leftarrow H_i(SIV)$
    Else
      The habitat keeps unchanged
    End if
    Generate $Rand \in [0,1]$
    If Rand<=m_j
      Execute mutation operator like mutation operator of GA
    Else
      The habitat keeps unchanged
    End if
  End for
  Calculate λ_j, μ_j, P_j, & m_j according to habitat's rank
End for

Fig. 13 Main algorithm of the BBO

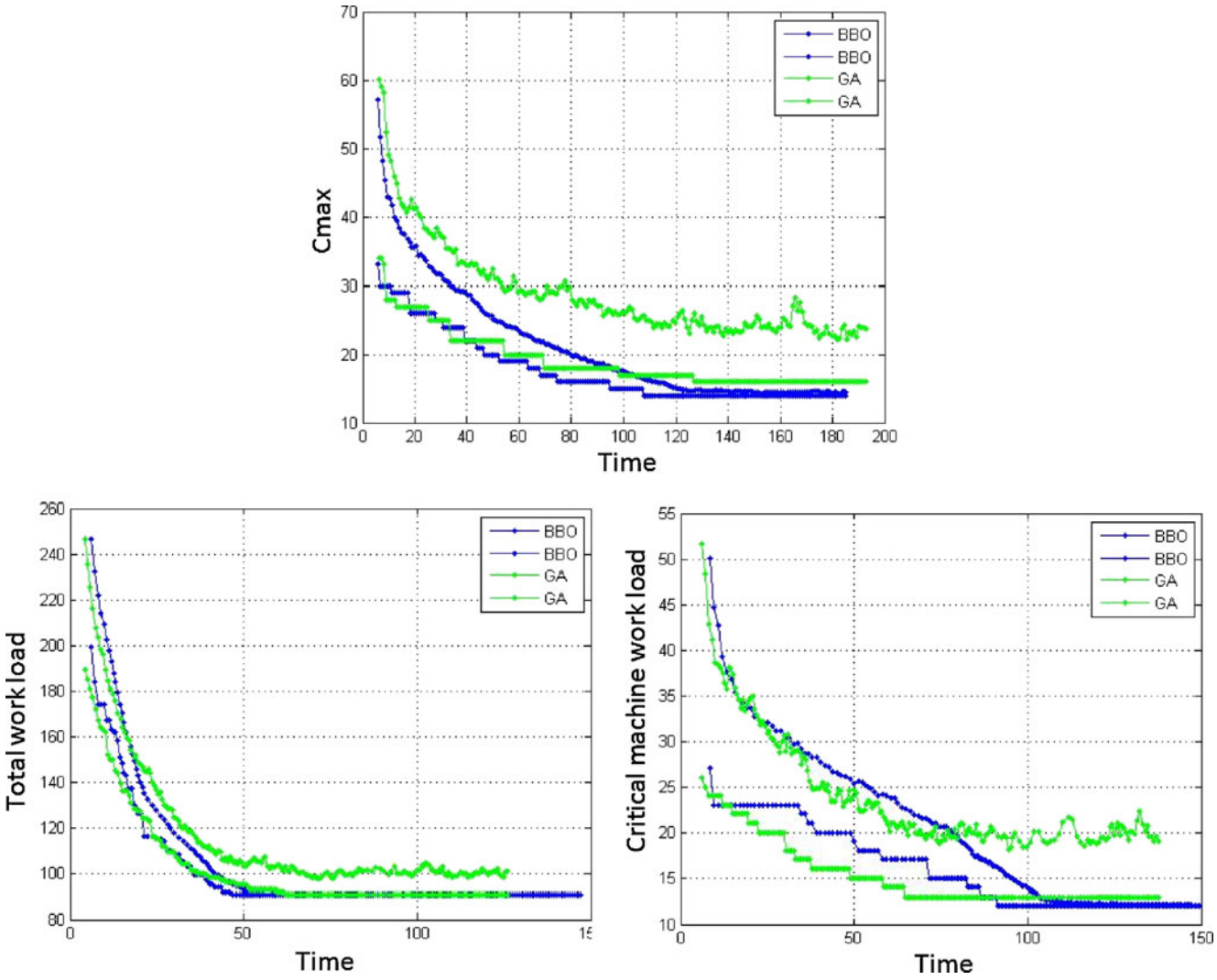

Fig. 14 Proposed BBO vs. the proposed GA with 200 population size and generation for 10*15 of Kacem

Table 3 Results of Brandimart library (BRdata)

Problem	$n \times m$	T_0	Flex.	LB	eGA				Proposed BBO			
					Pop	C_{max}	Av(C_{max})	t(C++)	Pop	C_{max}	Av(C_{max})	t(Matlab)
MK01	10*6	58	2.09	36	100	40	40	1.6	200	40	41	121
MK02	10*6	150	4.10	24	300	26	26	2.6	200	28	28.25	123
MK03	15*8	90	3.01	48	50	204	204	1.3	200	204	204	441
MK04	15*8	106	1.91	204	100	60	60	6.2	200	64	66	242
MK05	15*4	150	1.71	168	200	173	173	7.3	200	173	173.5	209
MK06	10*15	100	3.27	33	200	58	58	15.7	200	66	66.5	411
MK07	20*5	225	2.83	133	200	144	145	17.3	200	144	144.25	196
MK08	20*10	240	1.43	523	50	523	523	2.2	200	523	523	452
MK09	20*10	240	2.53	299	300	307	307	30.2	200	310	310.75	653
MK10	20*15	58	2.98	165	300	198	199	36.6	200	230	232.75	709

Table 4 Results of Barnes and Chamber library (BCdata)

Problem	$n \times m$	T_0	Flex.	LB	eGA				Proposed BBO			
					Pop	*Cmax*	Av(*Cmax*)	*t*(C++)	Pop	*Cmax*	Av(*Cmax*)	*t*(Matlab)
mt10c1	10*11	100	1.1	655	200	927	928	23.25	350	946	947	401
mt10cc	10*12	100	1.2	655	200	910	910	19.27	350	946	946	405
mt10x	10*11	100	1.1	655	1,000	918	918	21.45	350	955	961	416
mt10xx	10*12	100	1.2	655	1,000	918	918	20.38	350	939	945	480
mt10xxx	10*13	100	1.3	655	1,000	918	918	25.39	350	954	954.5	497
mt10xy	10*12	100	1.2	655	300	905	906	24.37	350	951	951	458
mt10xyz	10*13	100	1.3	655	1,000	847	847	30.24	350	858	858	495
setb4c9	15*11	150	1.1	857	1,000	914	914	12.81	350	959	959	762
setb4cc	15*12	150	1.2	857	1,000	909	910	20.16	350	944	950	770
setb4x	15*11	150	1.1	846	200	925	925	8.92	350	942	951	749
setb4xx	15*12	150	1.2	847	300	925	925	54.06	350	967	967	761
setb4xxx	15*13	150	1.3	846	1,000	925	925	62.81	350	991	991	797
setb4xy	15*12	150	1.2	845	1,000	916	916	27.78	350	978	982	778
setb4xyz	15*13	150	1.3	838	1,000	905	908.1	40.26	350	930	930.5	651
seti5c12	15*16	225	1.07	1,027	1,000	1,174	1,174	70.69	350	1,198	1,202	1,460
seti5cc	15*17	225	1.13	955	1,000	1,136	1136.2	69.53	350	1,199	1202.5	1,370
seti5x	15*16	225	1.07	955	1,000	1,209	1,209	67.56	350	1,249	1,254	1,382
seti5xx	15*17	225	1.13	955	1,000	1,204	1,204	78.29	350	1,266	1,273	1,429
seti5xxx	15*18	225	1.2	955	1,000	1,204	1,204	105.25	350	1,227	1,228	1,415
seti5xy	15*17	225	1.13	955	1,000	1,136	1136.3	70.47	350	1,170	1,195	1,419
seti5xyz	15*18	225	1.2	955	1,000	1,125	1126.5	70.56	350	1,175	1180.5	1,391

Besides, according to the results, our BBO's times are generally less than our GA's times. Therefore, the times used by Wang et al. can be considered as some samples for the upper bounds of our times if our algorithm was programmed in C++.

Table 5 Computational results on Kacem and Brandimart data for makespan (Cmax)

		Proposed BBO	Ho [14]	Brandimart [2]
Kacem	4*5	11	11	–
	8*8	14	–	–
	10*10	7	7	–
	10*15	12	12	–
Brandimart	MK01	40	40	42
	MK02	28	29	32
	MK03	204	-	204
	MK04	66	67	81
	MK05	173	176	186
	MK06	64	67	86
	MK07	144	147	157
	MK08	523	523	523
	MK09	310	320	369
	MK10	230	229	269

5 Conclusion and future works

During the last decades, for solving optimization problems, different new types of algorithm have been developed. Most of them are inspired by natural phenomena like the genetic mechanism of our body, or manner of ant, fish, or bee. This paper introduced a new naturally inspired algorithm to scheduling area in which biogeography theories are used for solving optimization problems. To do so, we adjusted the operators of the biogeography-based algorithm (BBO) including migration and mutation for flexible job shop scheduling problem. Then, we compared it with a newly developed GA, which has similar operators, for three different objectives. Finally, BBO was compared by three famous algorithms reported in the literature. In all of the assessments, our simple and non-equipped BBO shows a good performance. In comparison with a similar GA of the paper, the BBO was completely comparable; in comparison with three other famous algorithms, the BBO's best solution is at least as good as the LEGA of Ho et al. and the TS of Brandimart, and equal or nearly equal to the eGA of Zhang et al. (which introduced the most number of best solutions of the literature). Therefore, according to the results, BBO can be introduced as a capable algorithm for FJSP that

should be studied more to obtain better results from its potential capabilities in scheduling or other types of optimization problems.

Different development can still be considered for this algorithm, including:

- Developing BBO for other scheduling problems such as flow shop, job shop, open shop, etc.
- Adding a learning ability to BBO and creating a knowledge-based BBO (KNBBO)
- Proposing a hybrid version of BBO for solving different scheduling problems
- Combining BBO with local search to create immune BBO
- Improving BBO's functions, for example by considering nonlinear curves for immigration and emigration rates
- Developing a fuzzy version of BBO through fuzzy fitness or fuzzy process time
- Developing a multi-objective BBO for scheduling problems or even other optimization problems

Appendix

Table 6 Comparing the different objective functions of GA and BBO

		Proposed GA			Proposed BBO			Count	
		Min	Ave.	Max	Min	Ave.	Max	GA	BBO
4*5	C_{max}	11	11	11	11	11	11	–	–
	TWL	32	32	32	32	32	32	–	–
	CWL	7	7.5	8	8	8	8	2	–
8*8	C_{max}	15	15.5	16	14	14.75	15	–	3
	TWL	73	73	73	73	73	73	–	–
	CWL	13	13.25	14	11	11.4	12	–	3
10*10	C_{max}	7	7.75	8	7	7.75	8	–	–
	TWL	43	44	45	43	43.25	44	–	2
	CWL	6	6.5	8	5	5.25	8	–	2
10*15	C_{max}	14	14.25	15	13	13	13	–	3
	TWL	91	91	91	91	91	91	–	–
	CWL	14	15.25	16	12	12.75	14	–	3
Mk01	C_{max}	40	41.5	42	40	41	42	–	1
	TWL	153	153	153	153	153	153	–	–
	CWL	36	36	36	36	36	36	–	–
Mk02	C_{max}	28	28.25	29	28	28.25	29	–	–
	TWL	140	140	140	140	140	140	–	–
	CWL	26	26	26	26	26	26	–	–
Mk03	C_{max}	204	204	204	204	204	204	–	–
	TWL	812	812	812	813	813.25	814	3	–
	CWL	204	204	204	204	204	204	–	–
Mk04	C_{max}	64	64.75	65	64	66	67	2	–
	TWL	324	324	324	324	324	324	–	–
	CWL	60	60	60	60	60	60	–	–
Mk05	C_{max}	173	173.5	175	173	173.5	175	–	–
	TWL	272	272	272	272	272	272	–	–
	CWL	173	173	173	173	173	173	–	–
Mk06	C_{max}	64	65.5	67	66	66.5	67	3	–
	TWL	330	330	330	330	330	330	–	–
	CWL	54	54	54	54	54	54	–	–
Mk07	C_{max}	143	143.75	144	144	144.25	145	3	–
	TWL	649	649	649	649	649	649	–	–
	CWL	141	142.75	144	140	140.5	141	–	3

Table 6 (continued)

		Proposed GA			Proposed BBO			Count	
		Min	Ave.	Max	Min	Ave.	Max	GA	BBO
Mk08	C_{max}	523	523	523	523	523	523	–	–
	TWL	2,484	2,484	2,484	2,284	2,284	2,284	–	–
	CWL	523	523	523	523	523	523	–	–
Mk09	C_{max}	307	310	311	310	310.75	311	3	–
	TWL	2,210	2210.5	2,211	2,210	2210.25	2,211	–	1
	CWL	299	299	299	299	299	299	–	–
Mk10	C_{max}	220	220.25	221	230	232.75	236	3	–
	TWL	1,847	1,847	1,847	1,847	1,847	1,847	–	–
	CWL	197	197.5	199	197	198.25	199	1	–
				Sum				20	21

Table 7 Comparing the computational time of the proposed GA and BBO

		Proposed GA			Proposed BBO			Count	
		Min	Ave.	Max	Min	Ave.	Max	GA	BBO
4*5	C_{max}	55	57.75	66	34	40.5	47	–	3
	TWL	43	44.25	46	41	42.5	44	–	3
	CWL	48	49	51	37	38.75	41	–	3
8*8	C_{max}	148	168.75	177	145	150.75	156	–	3
	TWL	63	79	88	59	79	92	–	3
	CWL	44	45.25	47	38	38.5	40	–	3
10*10	C_{max}	79	88	99	84	87	91	3	–
	TWL	64	68.4	78	51	59.8	69	–	3
	CWL	36	43.6	56	41	44.8	57	3	–
10*15	C_{max}	63	93.25	121	61	85	109	–	3
	TWL	117	123	177	116	117.75	165	–	3
	CWL	117	127.75	159	115	121.5	143	–	3
Mk01	C_{max}	136	138.75	141	118	121.5	126	–	3
	TWL	108	117.25	140	105	127	156	–	3
	CWL	60	84.5	138	52	59.75	78	–	3
Mk02	C_{max}	123	127.25	133	117	123.5	137	–	3
	TWL	129	137	142	106	117.75	133	–	3
	CWL	61	81.75	135	56	77.5	125	–	3
Mk03	C_{max}	285	426.5	659	287	441.75	630	3	–
	TWL	230	295.5	335	291	378.75	450	3	–
	CWL	129	217	307	130	205.25	285	3	–
Mk04	C_{max}	169	236.25	348	162	242	361	–	3
	TWL	187	261.25	338	191	242.5	364	3	–
	CWL	84	132.25	206	79	105	178	–	3
Mk05	C_{max}	183	262	342	178	209.5	238	–	3
	TWL	209	215.75	224	192	215	247	–	3
	CWL	95	162	224	90	147	206	–	3
Mk06	C_{max}	281	357.25	463	359	410	464	3	–
	TWL	229	301	337	219	331.25	435	–	3

Table 7 (continued)

		Proposed GA			Proposed BBO			Count	
		Min	Ave.	Max	Min	Ave.	Max	GA	BBO
	CWL	123	172.25	282	126	193.5	264	3	–
Mk07	C_{max}	186	194.25	204	184	196	209	–	3
	TWL	203	214	224	187	192.25	199	–	3
	CWL	96	132.5	198	86	132.25	183	–	3
Mk08	C_{max}	453	468.25	504	450	451.5	454	–	3
	TWL	557	691.5	921	515	650.5	932	–	3
	CWL	194	277.5	446	178	241.25	415	–	3
Mk09	C_{max}	548	614.25	744	502	653	766	–	3
	TWL	535	547	569	511	568.5	709	–	3
	CWL	207	334.25	460	198	282.5	450	–	3
Mk10	C_{max}	587	723	1,103	596	709	1,002	3	–
	TWL	626	747.5	802	521	821	1,046	–	3
	CWL	195	216.5	245	205	209.75	217	3	–
				Sum				30	96

References

1. Brucker P, Schlie R (1990) Job-shop scheduling with multipurpose machines. Computing 45(4):369–375
2. Brandimarte P (1993) Routing and scheduling in a flexible job shop by tabu search. Annual Operation Research 41:157–183
3. Barnes JW, Chambers JB (1996) Flexible job shop scheduling by tabu search. Graduate Program in Operations Research and Industrial Engineering. University of Texas, Austin, Technical Report Series, ORP96-09
4. Xia WJ, Wu ZM (2005) An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Computer and Industrial Engineering 48(2):409–425
5. Hurink E, Jurisch B, Thole M (1994) Tabu search for the job shop scheduling problem with multi-purpose machine. Operations Research Spektrum 15(4):205–215
6. Scrich CR, Armentano VA, Laguna M (2004) Tardiness minimization in a flexible job shop: a tabu search approach. Intelligent Journal of Advance Manufacturing Technology 15(1):103–115
7. Chen JC, Chen KH, Wu JJ, Chen CW (2008) A study of the flexible job shop scheduling problem with parallel machines and reentrant process. Intelligent Journal of Advance Manufacturing Technology 39(3–4):344–354
8. Mastrolilli M, Gambardella LM (2000) Effective neighborhood functions for the flexible job shop problem. J Sched 3(1):3–20
9. Saidi-Mehrabad M, Fattahi P (2007) Flexible job shop scheduling with tabu search algorithms. Intelligent Journal of Advance Manufacturing Technology 32(5–6):563–570
10. Kacem I, Hammadi S, Borne P (2002) Approach by localization multi-objective evolutionary optimization for flexible job-shops scheduling problems. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 32(1):1–13
11. Kacem I, Hammadi S, Borne P (2002) Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 32 (2):172–172
12. Kacem I, Hammadi S, Borne P (2002) Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Mathematics and Computer Simulation 60(3–5):245–276
13. Mati Y, Rezg N, Xie XL (2001) An integrated greedy heuristic for a flexible job shop scheduling problem. Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics. OPAC, Tucson, pp 2534–2539
14. Ho NB, Tay JCJ, Lai E (2007) An effective architecture for learning and evolving flexible job-shop schedules. Eur J Oper Res 179:316–333
15. Gao J, Gen M, Sun LY, Zhao XH (2007) A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Computer and Industrial Engineering 53(1):149–162
16. Gao L, Peng CY, Zhou C, Li PG (2006) Solving flexible job-shop scheduling problem using general particle swarm optimization. Proceedings of the 36th International Conference on Computers & Industrial Engineering (ICCIE 2006), Jun 20–23, Taipei, China, pp 3018–3027
17. Zhang GH, Gao L, Li X, Li P (2008) Variable neighborhood genetic algorithm for the flexible job shop scheduling problems. Proceedings of Intelligent Robotics and Applications (ICIRA 08), LNCS Press, October, pp 503–512. doi:10.1007/978-3-540-88518-4-54
18. Zhang GH, Shao GH, Li PG, Gao L (2009) An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. Computer & Industrial Engineering 56:1309–1318
19. Zhang GH, Gao L, Shi Y (2010) A genetic algorithm and tabu search for multi objective flexible job shop scheduling problems. Proceedings of the 1st International Conference on Computing, Control and Industrial Engineering (CCIE 2010), Wuhan, China, June 5–6, pp 215–254
20. Zhang GH, Gao L, Shi Y (2010) A novel variable neighborhood genetic algorithm for multi-objective flexible job-shop scheduling problems. Adv Mater Res 118–120:369–373
21. Zhang GH, Gao L, Shi Y (2011) An effective genetic algorithm for the flexible job-shop scheduling problem. Experts System with Applications 38(4):3563–3573

22. Wang X, Gao L, Zhang G, Shao X (2010) A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. Int J Adv Manuf Technol 51(5–8):757–767
23. Wallace A (2005) The geographical distribution of animals (two volumes). Adamant Media Corporation, Boston
24. Darwin C (1995) The origin of species. Gramercy, New York
25. MacArthur R, Wilson E (1967) The theory of biogeography. Princeton University Press, Princeton
26. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12:702–713
27. Simon D (2009) A probabilistic analysis of a simplified biogeography-based optimization algorithm. http://academic.csuohio.edu/simond/bbo/simplified/bbosimplified.pdf
28. Du D, Simon D, Ergezer M (2009) Biogeography-based optimization combined with evolutionary strategy and immigration refusal. IEEE International Conference on Systems, Man, and Cybernetics. San Antonio, TX, pp 1023–1028
29. Ergezer M, Simon D, Du DW (2009) Oppositional biogeography-based optimization. IEEE Conference on Systems, Man, and Cybernetics, San Antonio, TX, pp 1035–1040
30. Ma H, Chen X (2009) Equilibrium species counts and migration model tradeoffs for biogeography-based optimization. 48th IEEE Conference on Decision and Control
31. Ma H, Simon D (2011) Blended biogeography-based optimization for constrained optimization. Eng Appl Artif Intell 24:517–525. doi:10.1016/j.engappai.2010.08.005
32. Panchal V, Singh P, Kaur N, Kundra H (2009) Biogeography based satellite image classification. Int J Comp Sci Inform Secur 6 (2):269–274
33. Kundra H, Kaur A, Panchal V (2009) An integrated approach to biogeography based optimization with case based reasoning for retrieving groundwater possibility. Proceedings of the Eighth Annual Asian Conference and Exhibition on Geospatial Information, Technology and Applications, August 2009, Singapore
34. Bhattacharya A, Chattopadhyay PK (2010) Solving complex economic load dispatch problems using biogeography-based optimization. Expert Systems with Applications 37:3605–3615
35. Zhang CY, Rao YQ, Li PG, Shao XY (2007) Bilevel genetic algorithm for the flexible job-shop scheduling problem. Jixie Gongcheng Xuebao/Chin J Mech Eng 43(4):119–124 (in Chinese)