Computers & Industrial Engineering 73 (2014) 96–114

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A hybrid biogeography-based optimization algorithm for job shop scheduling problem

Xiaohua Wang, Haibin Duan *

State Key Laboratory of Virtual Reality Technology and Systems, School of Automation Science and Electrical Engineering, Beihang University (BUAA), Beijing 100191, PR China

ARTICLE INFO

Article history:
Received 8 July 2013
Received in revised form 30 December 2013
Accepted 15 April 2014
Available online 9 May 2014

Keywords:
Job-shop scheduling problem (JSP)
Biogeography-based optimization (BBO)
Chaos theory
Hybrid biogeography-based optimization (HBBO)

ABSTRACT

In this paper, a hybrid biogeography-based optimization (HBBO) algorithm has been proposed for the job-shop scheduling problem (JSP). Biogeography-based optimization (BBO) is a new bio-inpired computation method that is based on the science of biogeography. The BBO algorithm searches for the global optimum mainly through two main steps: migration and mutation. As JSP is one of the most difficult combinational optimization problems, the original BBO algorithm cannot handle it very well, especially for instances with larger size. The proposed HBBO algorithm combines the chaos theory and "searching around the optimum" strategy with the basic BBO, which makes it converge to global optimum solution faster and more stably. Series of comparative experiments with particle swarm optimization (PSO), basic BBO, the CPLEX and 14 other competitive algorithms are conducted, and the results show that our proposed HBBO algorithm outperforms the other state-of-the-art algorithms, such as genetic algorithm (GA), simulated annealing (SA), the PSO and the basic BBO.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Knowledge and services have been emphasized in modern manufacturing in the present decade. Many enterprises are facing challenges of unprecedented and abrupt changes, such as intense global competition, rapid IT infrastructure changes and development of advanced management technologies. Due to its importance in the field of manufacturing industries, the job-shop scheduling problem (JSP) attracts considerable attention (Ma, Chu, & Zuo, 2010; Michael et al., 2013; Zhang, 2010).

The JSP is one of the most difficult combinatorial optimization problems and it has been well known as an NP-hard problem (Lawler et al., 1982; Xia & Wu, 2005). The JSP can be described briefly as follows. There are a number of jobs that need to be processed on a number of machines. Each job has several operations which have to be processed in a predefined sequence on different machines and each machine can only process one job at a time (Sels et al., 2012). The goal is to schedule the jobs on the machines to minimize the completion time needed for processing all jobs.

Because of the practical importance of the JSP, huge research effort has been devoted to fields of engineering, computer science, management science and business. Numerical efficient methods have been proposed, such as guided local search with shifting bottleneck (SB) (Balas & Vazacopoulos, 1998), tabu search (TS) (Azzouz et al., 2012; Brucker & Neyer, 1998; Eshlaghy & Sheibatolhamdy, 2011; Eugeniusz & Czesław, 2005; Nowicki & Smutnicki, 1996; Watson, Christopher Beck, Howe, & Darrell Whitley, 2003) and simulated annealing (SA) (Atabak et al., 2011; Van Laarhoven, Aarts, & Lenstra, 1992). Since that biologically inspired techniques have been advanced in the literature (Duan, Shao, Su, & Zhang, 2010; Duan, Xu, & Xing, 2010; Liu, Duan, & Deng, 2012; Sun, Duan, & Shi, 2013; Xu, Duan, & Liu, 2010; Yu & Duan, 2013; Zhang & Duan, 2014), there are also many bio-inspired optimization algorithms applied to the JSP, such as genetic algorithm (GA) (Azzouz et al., 2012; De Giovanni & Pezzella, 2010; Pezzella, Morganti, & Ciaschetti, 2007; Sun et al., 2010; Zhang, Shao, Lia, & Gao, 2009), particle swarm algorithm (PSO) (Gu, Tang, & Zheng, 2012; Lian, Jiao, & Gu, 2006; Xia, Wu, Zhang, & Yang, 2004), ant colony algorithm (ACO) (Anitha & Karpagam, 2013; Azzouz et al., 2012; Chen & Zhang, 2013; Xing & Chena, 2010), and artificial bee colony (ABC) algorithm (Deming, 2012; Tasgetiren et al., 2011; Wang, Duan, & Luo, 2013; Wang, Wang, Yu, & Zhang, 2013). In this paper, a hybrid biogeography-based optimization (HBBO), which takes advantages of the chaos theory (Xu et al., 2010) and the "searching around the optimum" strategy, is structured to solve the JSP.

Biogeography-based optimization (BBO) (Simon, 2008) is a new algorithm, which emulates the geographical distribution and the migration of species in an ecosystem. In this method each feasible

* Corresponding author. Tel.: +86 10 82317318.
E-mail address: hbduan@buaa.edu.cn (H. Duan).

http://dx.doi.org/10.1016/j.cie.2014.04.006
0360-8352/© 2014 Elsevier Ltd. All rights reserved.

solution is represented by a habitat. The habitat suitability index (HSI) is utilized to measure the goodness of the habitat. In order to find a solution with the best aspects, the concepts and models of biogeography are applied. The basic BBO works mainly based on the two mechanisms: migration and mutation. With the migration mechanism, poor solutions can accept a lot of new features, which may improve the quality of those solutions. Furthermore, solutions do not have the tendency to clump together in similar groups due to the new type of mutation operation. Elitism operation (Simon et al., 2009), which retains the best solutions in the population from one generation to the next, can also make the basic BBO algorithm more efficient in the both aspects of migration and mutation.

The chaos theory and "searching around the optimum" strategy are integrated into the basic BBO. Therefore, the exploration ability of the algorithm can be enhanced and the diversity of population can be improved. Furthermore, premature converge can be avoided. Thus the potential weakness lying in the basic method is avoided. We verify experimentally that our proposed HBBO approach outperforms the basic BBO approach, and is significantly better than the PSO and several state-of-art approaches.

The remainder of this paper is organized as follows. In Section 2, the JSP is formulated. In Section 3, the main principle of the basic BBO with elitism is presented. Section 4 describes the chaos theory and the principle of the "searching around the optimum" strategy. In Section 5, the detailed implementation procedures of the HBBO algorithm for the JSP are specified. In Section 6, a series of comparative experiments are given to verify that the HBBO approach outperforms the CPLEX, the PSO, the basic BBO and numerical existing algorithms. Our concluding remarks are contained in the final section.

2. The mathematical description of the JSP

Suppose that there are m machines $M = \{M_1, M_2, \ldots, M_m\}$ and n jobs $J = \{J_1, J_2, \ldots, J_n\}$, where job i has p operations $O_i = \{O_{i1}, O_{i2}, \ldots, O_{ip}\}$, $i = 1, 2, \ldots, n$. In this paper we make the number of the operations in each job the same, and it is also same to the number of the machines. Thus, $ip = m$, $i = 1, 2, \ldots, n$ and the operation set can be described as $O = \{\sigma_1, \sigma_2, \ldots, \sigma_{n\times m}\}$. Makespan $C_{\max}$ is the total time cost in completing operations of all jobs under conventional assumptions. The conventional assumptions can be described as following (Allaoui & Artiba, 2004; Wang, Duan, et al., 2013; Wang, Wang, et al., 2013).

(1) Each operation of one job must be processed exactly once on each of m machines.
(2) Each job is routed through the m machines in a given order to ensure that the j-th operation of job i O_{ij} will be processed only when the processing of the $(j-1)$-th operation O_{ij-1} has been finished.
(3) Every machine can process only one job at any time, and every operation cannot be interrupted.
(4) The machines cannot be breakdown.
(5) The jobs are independent from each other.

Matrix $S\,(m^*n)$ is developed to represent that the s_{ij}-th operation of the job i is processed on machine j, s is the element of the matrix, and i is the column and j is the row of the matrix.

$$S = \begin{bmatrix} s_{11} & s_{12} & \ldots & s_{1m} \\ s_{21} & s_{22} & \ldots & s_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \ldots & s_{nm} \end{bmatrix}$$

Another matrix $T\,(m^*n)$ is developed to represent the time cost t_{ij} of the j-th operation of the job i, i is the column and j is the row of the matrix.

$$T = \begin{bmatrix} t_{11} & t_{12} & \ldots & t_{1m} \\ t_{21} & t_{22} & \ldots & t_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} \ldots & t_{nm} & \end{bmatrix}$$

The processing of operation σ_j on the machine i is denoted by G_{ij}. Suppose that the p-th operation on machine i, G_{ip}, is directly connected by the q-th operation G_{iq}, the relation can be represented by the operator $G_{ip} \to G_{iq}$. The completion time of $G_{ip} \to G_{iq}$ can be calculated with $C_{iq} = C_{ip} + t_{iq} + t_k$, where t_{iq} is the time cost of the operation σ_q on the machine i and t_k is the space time on machine i during the convert of the two operations. Thus the completion time of all the jobs can be computed as $C_{\max} = \max_{allG_{ij}}(C_{ij})$. Therefore, the aim of the JSP is to make the makespan $C_{\max}$ minimized.

3. Biogeography-based optimization

3.1. Principles of the basic BBO algorithm

BBO is a bio-inspired computation algorithm inspired by the geographical distribution and the migration of species in an ecosystem. The problem can be of any area in life (Engineering, Economics, Medicine, Business, Urban Planning, Sports, etc.) as long as we have a qualitative measure of the suitability of a given solution (Bansal et al., 2013; Jain et al., 2012; Jamuna & Swarup, 2011; Li, Wang, Zhou, & Yin, 2011; Ma & Simon, 2011; Sayed, Saad, Emara, & Abou El-Zahab, 2013; Silva, Coelho, Lebensztajn, & Lebensztajn, 2012; Wang & Duan, 2013). The BBO technique is utilized to solve the JSP in this paper.

BBO is developed based on the mathematics models of biogeography, which explains how species emigrate and immigrate within the habitats, how new species arise, and how species become extinct. A quantitative performance index HSI is used to evaluate if a habitat is suitable for biological survival. Features that correlate with HSI include such factors as rainfall, diversity of vegetation, diversity of topographic features, land area and temperature (Simon, 2008). The variables that characterize habitability are named suitability index variables (SIVs). SIVs are the independent variables of the habitat, while HSI are the dependent variable.

A set of habitats is used to present the candidate solutions in the BBO. The basic BBO works mainly on two mechanisms, migration and mutation. HSI is the evaluation criteria to measure if a solution is good, analogous to fitness in other population-based optimization algorithms. The species migrate from one habitat to other habitats that have good geographical conditions. Habitats with a high HSI tend to have a larger number of species, more species that emigrate to nearby habitats, and a lower species immigration rate. The solutions with high HSI tends to share their features with those with low HSI, and poor solutions can accept a lot of new features from good solutions. Mutation is a probabilistic operator that randomly modifies habitat SIVs based on the habitats a priori probability of existence. Suppose that we have a habitat H, a vector of SIVs, following the migration and mutation steps to reach the optimal solution. In this way, new candidate habitat is generated from all of the salutation in population.

3.2. The migration strategy

BBO migration is a probabilistic operator that adjusts each solution H_i by sharing features between solutions. In the BBO, the migration strategy is similar to the evolutionary strategy in which many parents can contribute to a single offspring (Li et al., 2011).

Migration can be expressed as $H_i(SIV) \leftarrow H_j(SIV)$ (Ma & Simon, 2011). Each individual has its own immigration rate λ and emigration rate μ, which are related with the number of species on the island. When more species inhabit the island, the immigration rate reduces while the emigration rate increases (Wang, Duan, et al., 2013; Wang, Wang, et al., 2013). Therefore, the immigration and emigration rates can be calculated as follows when there are k species in the habitat.

$$\begin{cases} \mu_s = \frac{ES}{S_{max}} \\ \lambda_s = I\left(1 - \frac{S}{S_{max}}\right) \end{cases} \quad (1)$$

where I is the maximum possible immigration rate, which occurs when there are zero species on the island. The largest possible number of species that the habitat can support is S_{max}, at which point the immigration rate becomes zero and the maximum possible emigration rate E occurs. The immigration and emigration curves are shown in Fig. 1.

The equilibrium species number S_0 is reached when the emigration rate μ is equal to the immigration rate λ.

Suppose that there are N habitats, H_i is one of them, whose immigration rate is λ_i, and H_j is another habitat whose emigration rate is μ_j. The migration process can be presented in Fig. 2.

The probability P_s, which represents that the habitat contains exactly species S, changes from time t to time $t + \Delta t$. It is updated as follows:

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

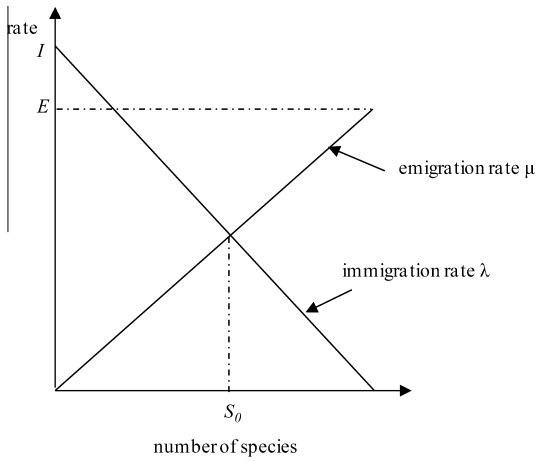

Fig. 1. The relationship between the number of species and the migration rates.

3.3. The mutation strategy

As the HSI of a habitat can change suddenly due to apparently random events, there is mutation of SIVs in the BBO. Solutions with very high HSI and very low HSI are both equally improbable, while medium HSI solutions are relatively probable to mutate. Mutation is used to enhance the diversity of the population, which helps to decrease the chances of getting trapped in local optima. Moreover, elitism (copying some of the fittest individuals to the next generation) is applied to guarantee the survival of the best individual(s) (Simon et al., 2009). Mutation is a probabilistic operator that randomly modifies an SIV of a solution based on its a priori probability of existence. Species count probabilities P_S, computed from λ_s and μ_s with equation (2), is used to determine mutation rates. Suppose a habitat with S species is selected to execute the mutation operation, randomly modify a chosen variable (SIV) based on its associated probability P_S. The mutation rate $m(s)$ can be computed according to the following function proportional to P_S:

$$m(s) = m_{max}\left(1 - \frac{p_s}{p_{max}}\right) \quad (3)$$

where m_{max} is a user-defined parameter, and P_{max} is the maximum species count probability, while P_S is the probability of existence of S species in the habitat.

Suppose that habitat $H_i \in SIV^R$ represents a feasible solution to some problem, the mutation process can be presented in Fig. 3.

4. The HBBO method

The BBO is a flexible, versatile and robust algorithm. However, there are still some weak points of the basic algorithm when used to solve the JSP, such as requiring a large number of iterations to reach the global optimal solution and the tendency to converge prematurely. Therefore, the chaos theory and "searching around the optimum" strategy are introduced into the basic BBO to accelerate the process of looking for the scheduling solution.

4.1. Chaos theory

Chaos theory, which is discovered by a meteorologist named Edward Lorenz (Lorenz, 1963), studies the behavior of dynamical systems that are highly sensitive to initial conditions. It is an effect which is popularly referred to as the butterfly effect. The butterfly effect phenomenon, common to chaos theory, is also known as sensitive to initial conditions, which means that each point in such a system is arbitrarily closely approximated by other points with significantly different in future.

Small differences in initial conditions yield widely diverging outcomes in chaotic systems, rendering long-term prediction

```
Begin
 for i = 1: N
     use λ_i to probabilistically decide whether to immigrate to H_i
     if rand (0,1) < λ_i then
       select the emigrating island H_j with probability ∝ μ_j, j ∈ [1,N]
       if rand (0,1) < μ_j then
       H_j(SIV) ← H_i(SIV)
       end if
     end if
 end for
End
```

Fig. 2. The migration process of the BBO.

```
Begin
    for j =1 : R
     Compute the probability Ps with equation (2)
     Compute the  mutation rate m(s) with equation (3)
      Select H_i(j) with probability ∝ m(s)
      if  H_i(j) is selected then
        Replace H_i(j) with randomly generated SIV
      end if
    end for
End
```

Fig. 3. The mutation process of the BBO.

impossible in general. Besides the sensitivity to initial conditions, chaotic systems have another two properties. One is that an infinite number of unstable periodic orbits embed in the underlying chaotic set. The other is that the dynamics in the chaotic system is ergodic, which means that the system ergodically visits a small neighborhood of every point in each one of the unstable periodic orbits embedded in its temporal evolution.

The criterion for a dynamical system to be classified as a chaotic system is described as follows. It must be sensitive to initial conditions, and it must be topologically mixing, moreover its periodic orbits must be dense. The basic idea of chaos theory can be understood by considering the following well-known one-dimensional logistic map, which is one of the best studied chaotic systems (Liu et al., 2012; Xu et al., 2010).

$$x_{n+1} = \eta x_n(1 - x_n) \tag{4}$$

where n is the number of habitats, x_n is restricted to the unit interval [0, 1] where 1 represents the maximum possible population and 0 represents extinction. η is the growth rate of x_n. A very small difference in the initial value of x_n would give rise to large difference in its long-time behavior, which is the basic characteristic of chaos. Moreover, when the value of $\eta = 4$ and $x_n \neq 0.25, 0.5, 0.75$ will eventually visit every neighborhood in a subinterval of [0, 1] with the best periodicity. Hence the logistic map is defined as following.

$$x_{n+1} = 4x_n(1 - x_n) \tag{5}$$

This logistic map develops chaos by the period-doubling bifurcation route. The track of chaotic variable can travel ergodically over the whole space of interest in the neighborhood of a periodic orbit. Despite its variation looks like in disorder, the variation of the chaotic variable has a delicate inherent rule. In the proposed HBBO algorithm, chaos theory is used to initialize the habitats. In this way, the ergodicity and irregularity of the chaotic variable can increase the diversity of the habitats to increase the speed of reaching the optimal solution.

4.2. The "searching around the optimum" strategy

As the basic BBO has a tendency to converge prematurely, a strategy of "searching around the optimum" is proposed. In this strategy, an effective method that identifies premature stagnation is embedded to the BBO. Thus, once premature stagnation happens, a randomized solution, as a substitute for current optimum, is used to change the current searching locus. Hence, the habitats can get out of the local optima.

In each generation of the BBO an optimal solution can be reached. When the optimal solution has no change in the process of continuous *MAX* iterations, we think that the algorithm has the potential of stagnation and the habitats have been or will soon get into local optimal solution. *MAX* is a threshold given by the user according the specified problem and the experiment results. A greater *MAX* indicates that the standard of the premature stagnation judgment is more easing.

A counter C is used to record the number of continuous iterations that have the same optimal solution. If the current iteration has the same optimal solution with the last iteration, $C = C + 1$. Otherwise, $C = 0$. When $C = MAX$, we think that the algorithm cannot jump out of the local optima. Then use a random value to replace a randomly selected dimension of the current optimal solution, which can be described as follows:

$$U_i = (u_{i1}, u_{i2}, \ldots, u_{iD}) \tag{6}$$

where $u_{ik} \in [\min_k, \max_k]$, $k \in [1, D]$ is randomly chosen, and U_i is the optimal solution of current iteration.

This strategy can make a fine tuning for the global optimal solution with a random disturbance and help the algorithm jump out of the local optimal solution. With this stagnation judgment and the random disturbance, the invalid iteration can be effectively reduced. Thus the convergence speed of the algorithm can be improved and the accuracy of the optimization results can be higher, which make the algorithm more stable.

5. Implementation of the HBBO for the JSP

5.1. Encoding strategy

Because the JSP is a combinatorial optimization problem, encoding is necessary in the HBBO algorithm. Encoding refers to the habitats expression, which is a key problem in the application of the HBBO method for the JSP. Think about the combination characteristics and complicated constraint of the JSP, the encoding of the HBBO for the JSP is much more complicated than other schedule problems such as the flow-shop scheduling problem. The Lamarkian character, the complexity of the decoding, coding space characteristics, and the storage requirements all have to be taken into account. Hence, encoding based on the process is used in the HBBO algorithm.

For a JSP with n jobs processed on m machines, the encoding of habitats can be described as following.

$$[h_1, h_2, \ldots, h_i, \ldots, h_{n^*m}] \tag{7}$$

where h_i is the sequence number of a job. If h_i is the n-th appearance of the job number, the n-th operation is on processing. This kind of encoding can guarantee the feasibility of random scheduling results.

5.2. Decoding strategy

After the completion of the optimization, the scheduling result has to be decoded from the habitats. Take a JSP with 3 jobs to be processed on 3 machines for example. There are 3*3 operations

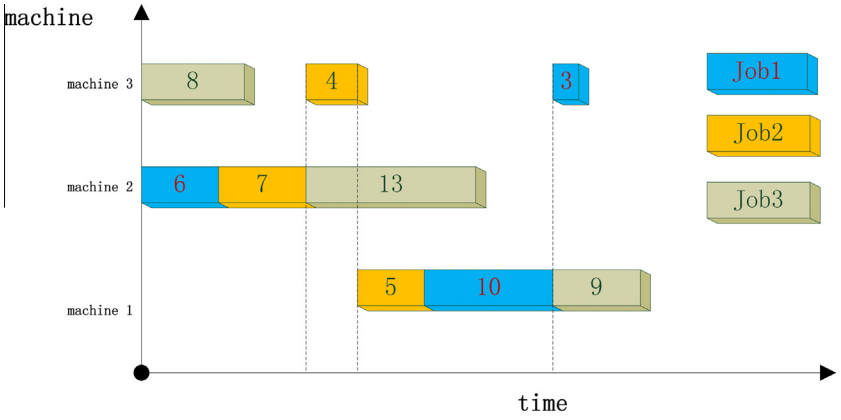

Fig. 4. GATT chart of the 3*3 instance.

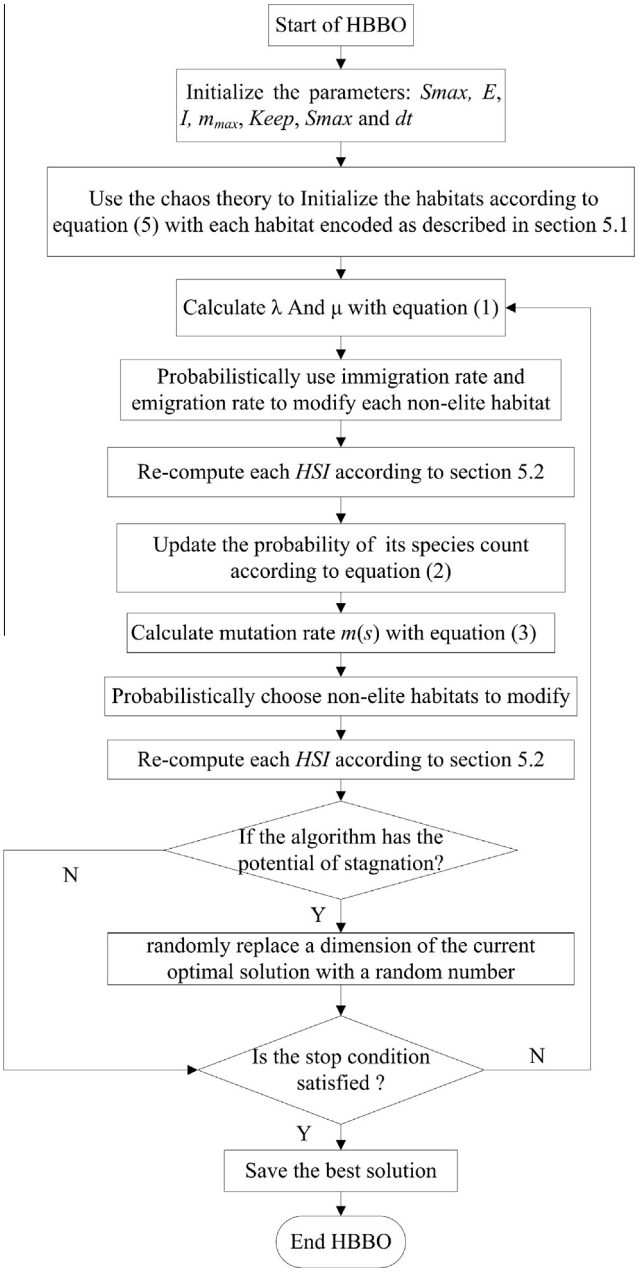

Fig. 5. Flow chart of our proposed HBBO for the JSP.

Table 1
Information of the 6*6 JSP instance.

Job	Operation sequence	Machine assigned	Processing time	Job	Operation sequence	Machine assigned	Processing time
Job1	O11	3	1	Job4	O41	2	5
	O12	1	3		O42	1	5
	O13	2	6		O43	3	5
	O14	4	7		O44	4	3
	O15	6	3		O45	5	8
	O16	5	6		O46	6	9
Job2	O21	2	8	Job5	O51	3	9
	O22	3	5		O52	2	3
	O23	5	10		O53	5	5
	O24	6	10		O54	6	4
	O25	1	10		O55	1	3
	O26	4	4		O56	4	1
Job3	O31	3	5	Job6	O61	2	3
	O32	4	4		O62	4	3
	O33	6	8		O63	6	9
	O34	1	9		O64	1	10
	O35	2	1		O65	5	4
	O36	5	7		O66	3	1

in a complete scheduling, thus there should be 3*3 SIVs in each habitats. Suppose that the initial scheduling of the HBBO is [1,2,3,2,2,3,1,3,1], and the operation-machine matrix is given as following:

$$\begin{bmatrix} 2 & 1 & 3 \\ 2 & 3 & 1 \\ 3 & 2 & 1 \end{bmatrix}.$$

The scheduling result of the habitat is presented as following.

(1) Execute the first operation of job1 on the machine 2.
(2) Execute the first operation of job2 on the machine 2.
(3) Execute the first operation of job3 on the machine 3.
(4) Execute the second operation of job2 on the machine 3.
(5) Execute the third operation of job2 on the machine 1.
(6) Execute the second operation of job3 on the machine 2.
(7) Execute the second operation of job1 on the machine 1.
(8) Execute the third operation of job3 on the machine 1.
(9) Execute the third operation of job1 on the machine 3.

In addition, suppose that the time cost matrix is given as following.

$$\begin{bmatrix} 10 & 6 & 3 \\ 5 & 7 & 4 \\ 9 & 13 & 8 \end{bmatrix}$$

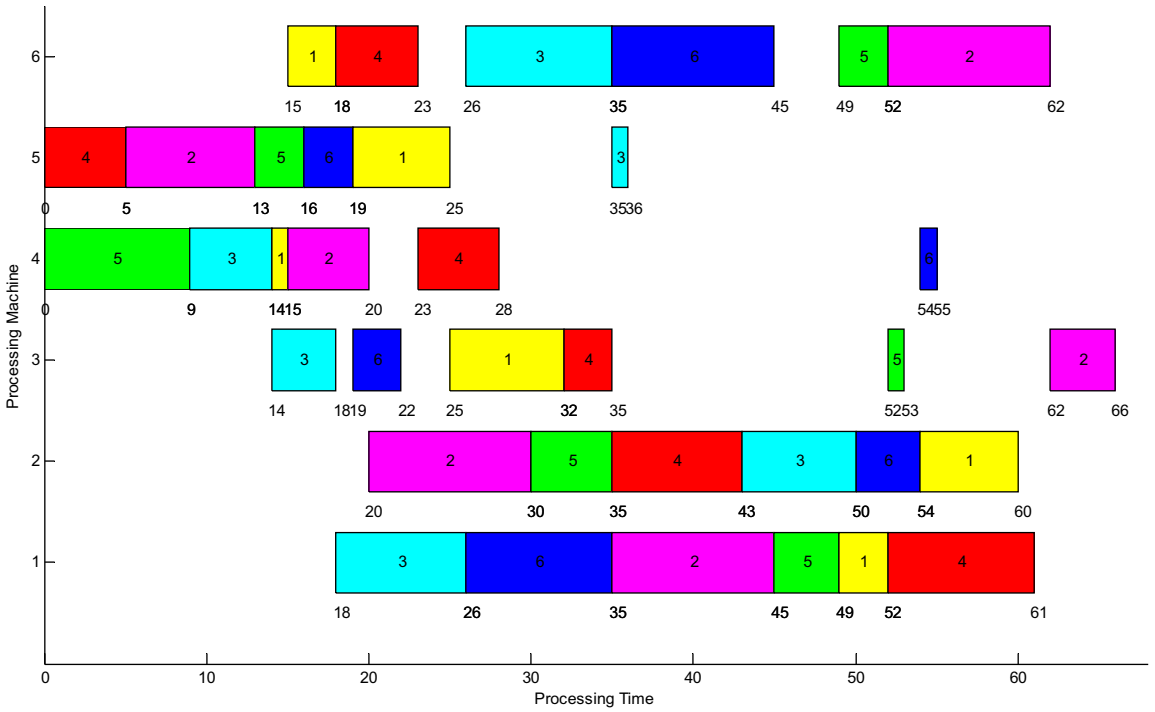

Fig. 6. GATT chart for a possible schedule of the 6*6 JSP without optimization.

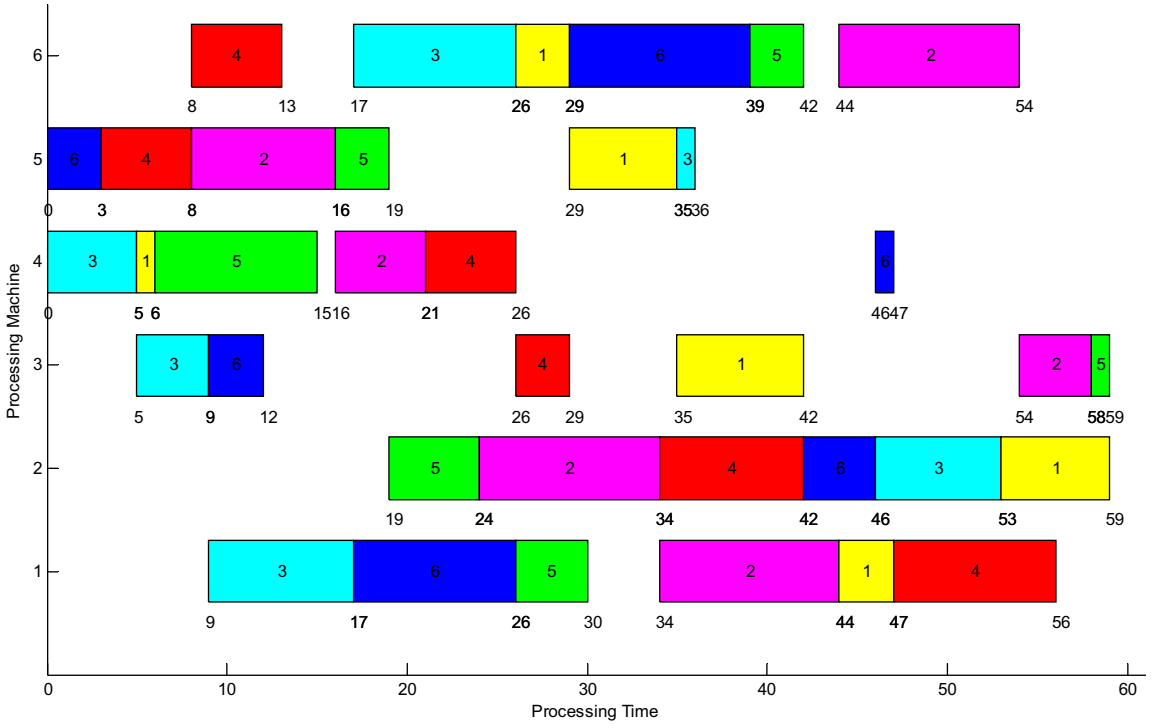

Fig. 7. GATT chart for the schedule of the 6*6 JSP optimized by the PSO method.

The makespan time of the whole processing is 41 s.

The GATT chart is presented in Fig. 4.

5.3. The procedure of the HBBO for the JSP

Our proposed HBBO combines the basic BBO with the characteristics of random of chaos theory, efficiency and accuracy of the "searching around the optimum" strategy. The proposed HBBO algorithm has better properties with high accuracy and efficiency than the basic BBO. The main operations involved in this process are given below.

- **Step 1:** Initialize HBBO parameters.

Map the problem solutions to SIVs and habitats according to the problem. Initialize the maximum species count S_{max}, the maximum

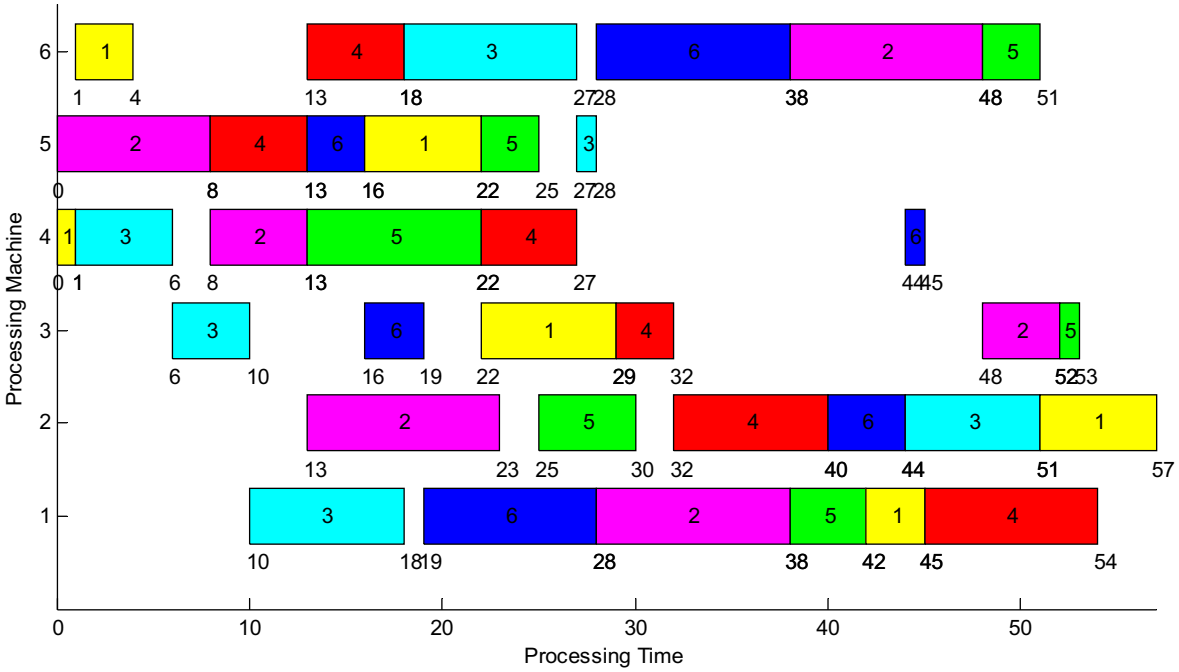

Fig. 8. GATT chart for the schedule of the 6*6 JSP optimized by the BBO method.

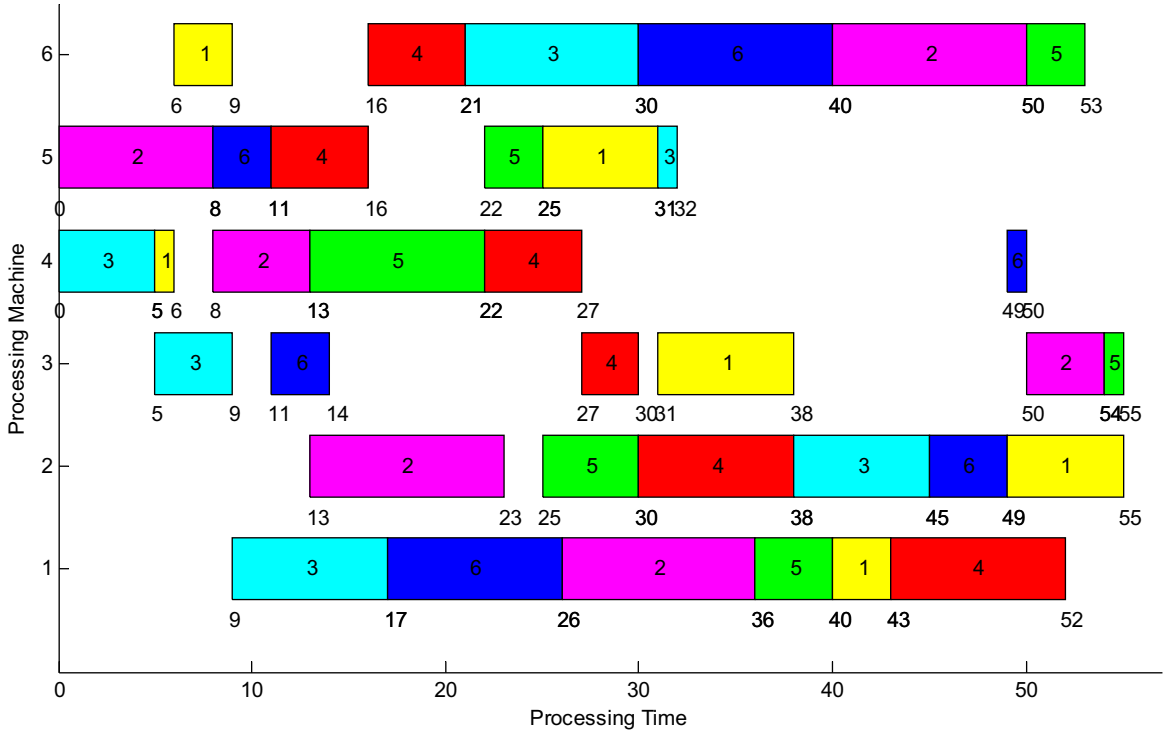

Fig. 9. GATT chart for the schedule of the 6*6 JSP optimized by the HBBO method.

migration rates E and I, the maximum mutation rate m_{max}, the elitism parameter *Keep*, and the step size used for numerical integration of probabilities dt .

- **Step 2:** Initialize habitats.

Use the chaos theory to initialize a set of habitats with each habitat corresponding to a potential solution according to equation (5) and the encoding strategy described in Section 5.1.

- **Step 3:** Calculate λ and μ.

For each habitat, map the HSI to the number of species S, the immigration rate λ, and the emigration rate μ according to equation (1).

- **Step 4:** Migrate.

Probabilistically use immigration rate λ_i and emigration rate μ_i to modify each non-elite habitat. Re-compute each HSI according to Section 5.2.

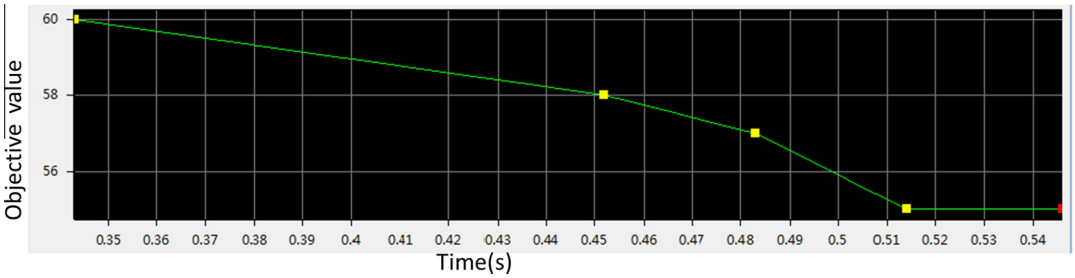

Fig. 10. Statistics of the CPLEX for the schedule of the 6*6 JSP. (The green line shows the tendency of the objective value's changing with time. The yellow points represent objective values of possible solutions and the last red point is objective value of the best solution.) (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

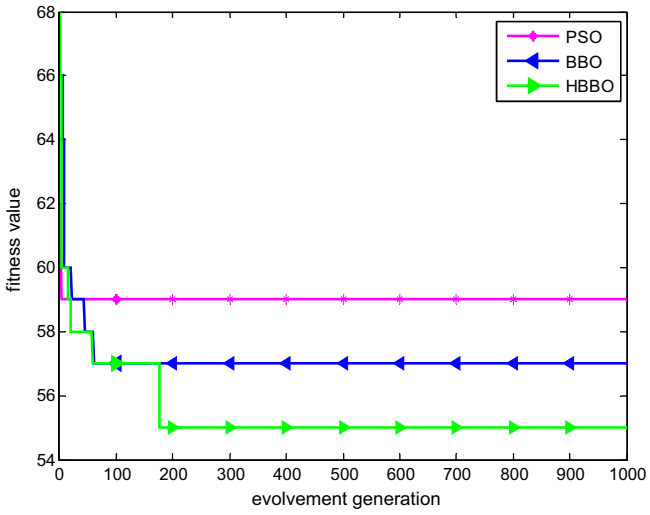

Fig. 11. Evolution curves of methods for the 6*6 JSP in comparison.

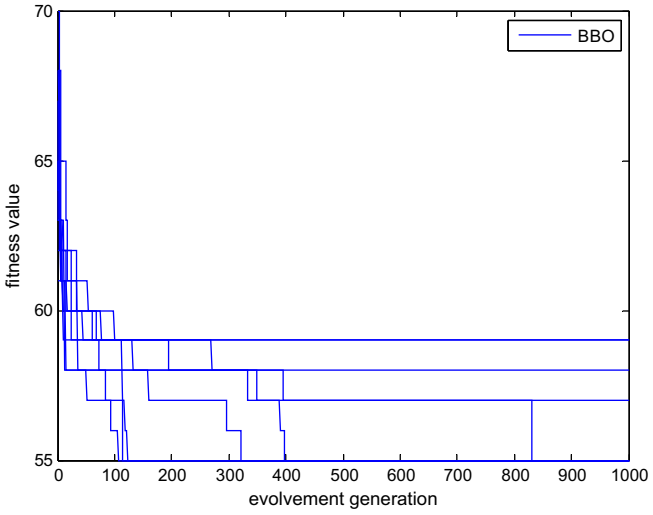

Fig. 13. Evolution curves of the basic BBO.

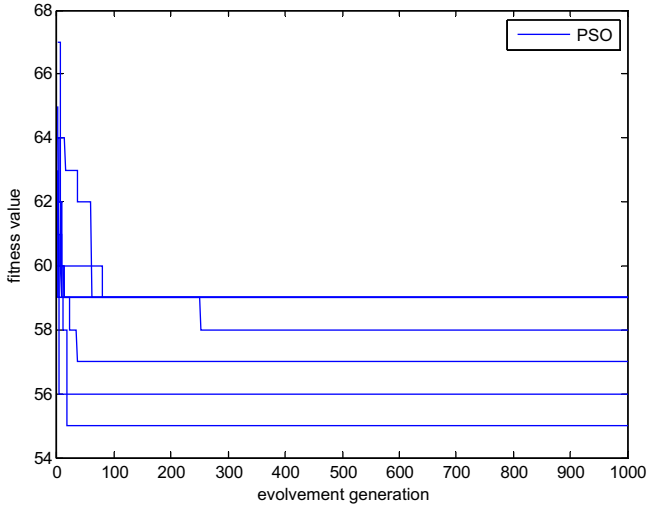

Fig. 12. Evolution curves of the PSO.

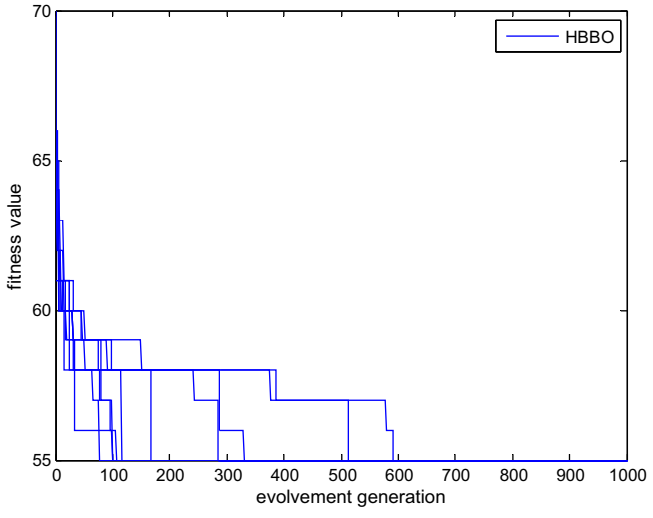

Fig. 14. Evolution curves of the HBBO.

- **Step 5:** Mutate.

For each habitat, update the probability of its species count according to equation (2). Mutate each non-elite habitat based on its probability according to equation (3), and re-compute each HSI according to Section 5.2.

- **Step 6:** Searching around the optimum.

If the algorithm has the potential of stagnation, use the strategy of "searching around the optimum" to randomly replace a dimension of the current optimal solution with a random number to help the algorithm jump out of the local optima according to Section 4.2. Re-compute each HSI according to Section 5.2.

Table 2
Information of the 10*5 JSP instance.

Job	Operation sequence	Machine assigned	Processing time	Job	Operation sequence	Machine assigned	Processing time
Job1	O11	3	54	Job6	O61	2	53
	O12	4	79		O62	3	99
	O13	1	16		O63	5	60
	O14	2	66		O64	1	13
	O15	5	58		O65	4	53
Job2	O21	2	83	Job7	O71	1	38
	O22	1	23		O72	2	60
	O23	3	89		O73	3	23
	O24	4	58		O74	5	59
	O25	5	56		O75	4	41
Job3	O31	3	15	Job8	O81	3	27
	O32	2	11		O82	2	25
	O33	5	49		O83	1	57
	O34	1	31		O84	4	49
	O35	4	20		O85	5	69
Job4	O41	2	71	Job9	O91	5	87
	O42	4	99		O92	3	56
	O43	1	15		O93	4	64
	O44	5	68		O94	2	85
	O45	3	85		O95	1	13
Job5	O51	3	36	Job10	O101	4	76
	O52	1	70		O102	2	30
	O53	2	45		O103	5	27
	O54	4	91		O104	3	85
	O55	5	35		O105	1	86

- **Step 7:** If the stopping criterion is satisfied, stop the iterations and output the solution, otherwise, go to **Step 3** for the next iteration. This loop can be terminated after a predefined number of iterations, or after an acceptable problem solution has been found.

The detailed flow chart of the proposed HBBO approach for the JSP is shown in Fig. 5.

6. Comparative experimental results

In order to verify the feasibility and effectiveness of the proposed algorithm HBBO for solving the JSP, a set of comparative experiments are conducted. To prove that our proposed HBBO performs better than other algorithms, some competitive intelligent methods should be chosen. The performance of the proposed algorithm is compared with that of the CPLEX, the PSO, and the basic BBO algorithm. In addition, the results are also compared with that of numerical existing methods.

The methods are all coded in Matlab, which run on a 2.4 GHz Intel Core i3 PC with 4.0 GB of RAM under Windows 2007 operating system. As it is generally accepted that the difficulty to solve the JSP is closely associated with the problem size, the parameters for different instance should be various. Therefore, the initial parameters of the HBBO method are set as following based on the tests and practical experience. For smaller instances (i.e., instance 1 and 2 in experiment 1; FT06 and LA01to LA 05 in experiment 2), we set total population size $P = 50$, number of generations $N = 1000$, step size used for numerical integration of probabilities $dt = 1$, the maximum migration rates $I = 1$, $E = 1$, the maximum mutation probability $m_{max} = 0.001$, and the number of elitisms $keep = 3$. For larger instances (i.e., instance 3 in experiment 1; FT10–FT20, LA06–LA26, LA31 and LA36 in experiment 2; all the instances in experiment 3) the total population size $P = 100$, number of generations $N = 3000$, step size used for numerical integration of probabilities $dt = 1$, the maximum migration rates $I = 1$, $E = 1$, the maximum mutation probability $m_{max} = 0.005$. The population sizes and the numbers of iterations of the PSO and the basic BBO are set to be the same with that in the HBBO in each experiment.

6.1. Experiment 1

To test the performance of the HBBO, three JSP instances with different scales are selected in the first experiment. Comparative experiments are conducted by using the PSO, and the basic BBO algorithm and an exact commercial solver IBM ILOG CPLEX Optimization Studio version 12.2. The population sizes and the numbers of iterations of all the three bio-inspired intelligent methods are set to be the same. And the job shop scheduling example in the CPLEX is executed with basic configuration. The GATT charts of scheduling results conducted by different intelligent algorithms and the statistics of the CPLEX are presented. In addition the evolution curves of each intelligent algorithm are given in comparison.

The processing time and operation sequence for the first 6*6 JSP instance (Fisher & Thompson, 1963) are given as following in

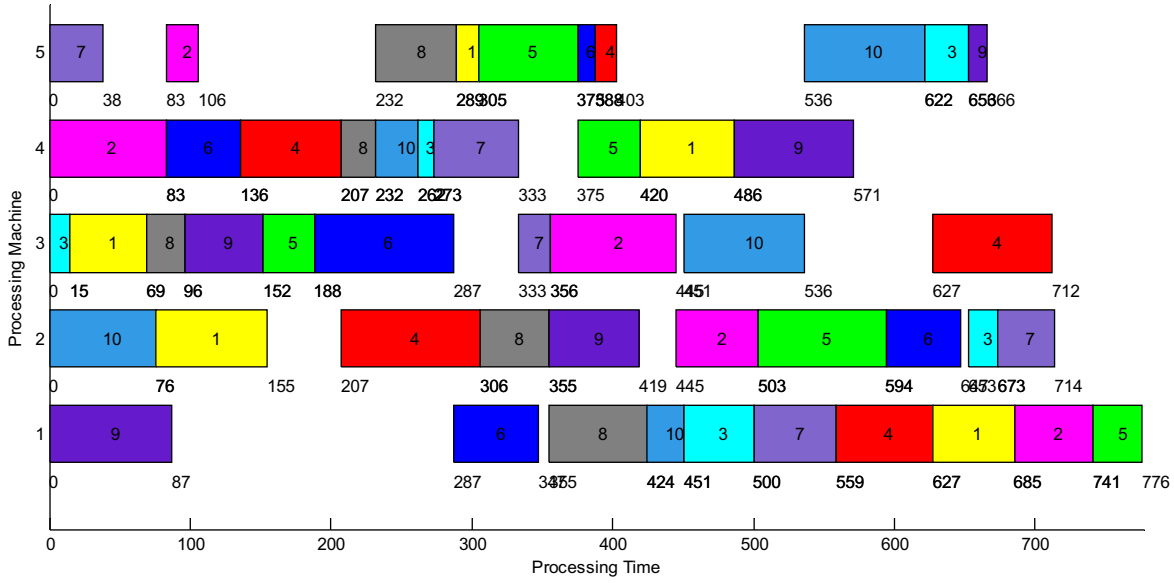

Fig. 15. GATT chart for a possible schedule of the 10*5 JSP without optimization.

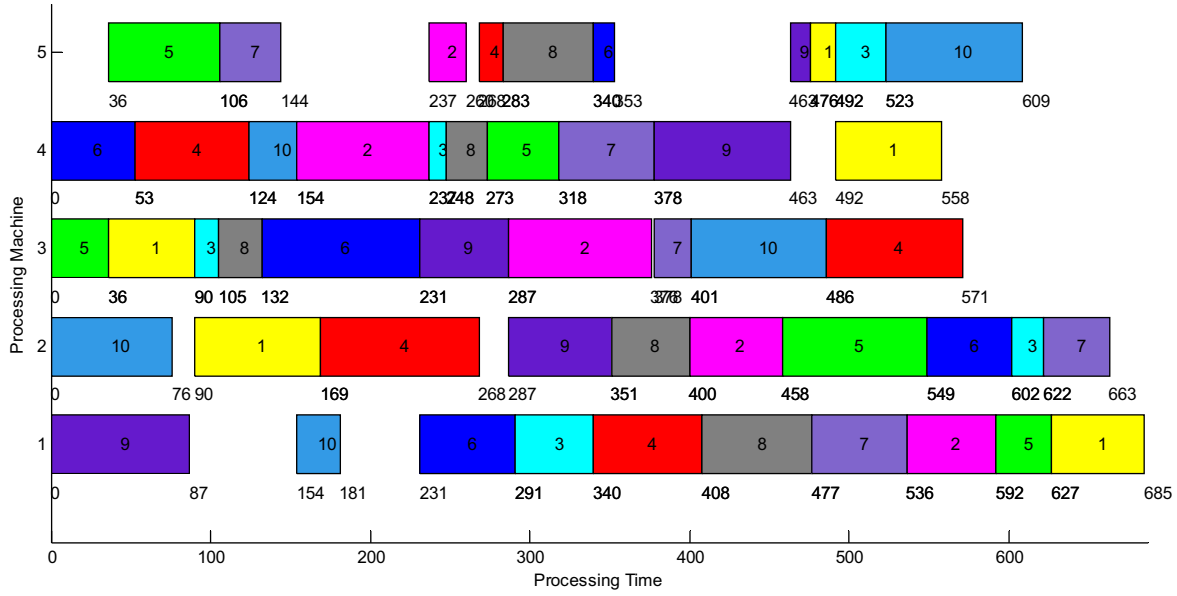

Fig. 16. GATT chart for the schedule of the 10*5 JSP optimized by the PSO method.

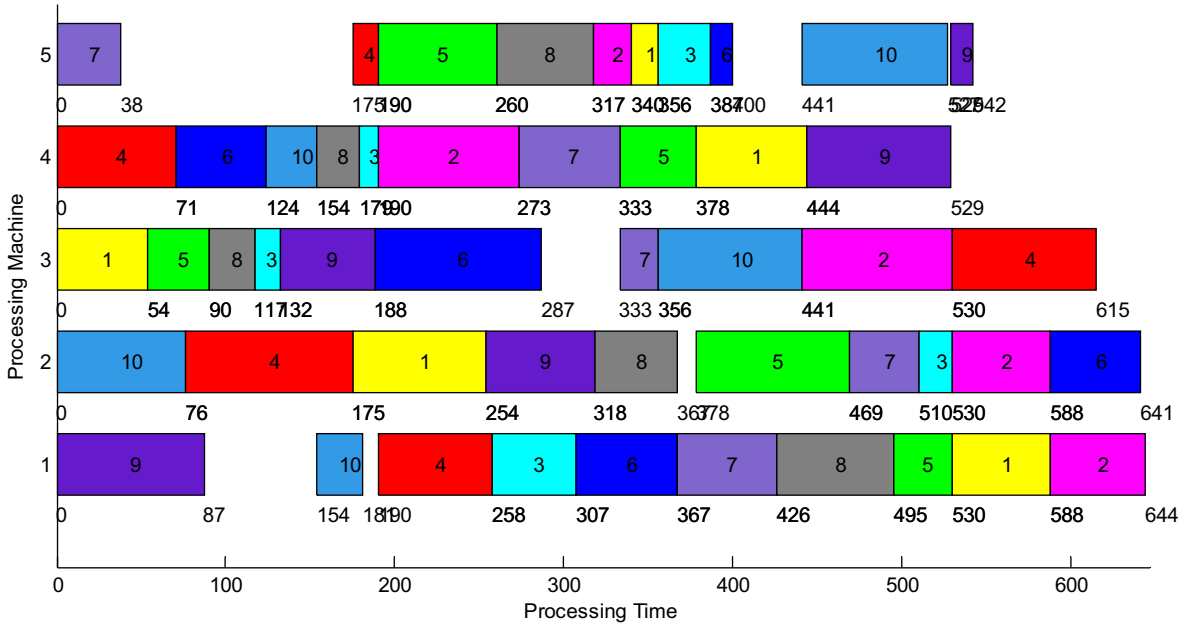

Fig. 17. GATT chart for the schedule of the 10*5 JSP optimized by the basic BBO method.

Table 1. Figs. 6–11 present the experimental results of different methods for the JSP with 6 jobs and 6 machines.

Figs. 6–9 show the GATT charts for the schedule of the 6*6 JSP with the PSO, the basic BBO and the HBBO respectively, and Fig. 10 gives statistics of the CPLEX for the schedule of the first instance, while Fig. 11 presents the comparative evolution curves of the three bio-inspired intelligence methods. From the simulation results, it is obvious that the performance of the proposed algorithm HBBO is better. As the minimum of the makespan (55) is found by the CPLEX and the HBBO, and the basic BBO finds a worse solution with a higher makespan (57), while the PSO obtains the worst solution with the highest makespan (59).

To get the average performance of the bio-inspired computation algorithms, more runs on the problem instance should be performed. Further experiments with the PSO, the basic BBO and the HBBO are given from Figs. 12–14 to verify the stability and advantage of the proposed method. For the first instance, each of the algorithms runs for 10 times independently and the evolution curves are obtained to test if the algorithm is stable.

Figs. 12–14 present the evolution curves of the PSO, the basic BBO and the HBBO for 10 runs. Among the three methods, the average converging rate of the PSO is the fastest apparently. However, the PSO algorithm gets into local optimal easily and the average makespan value of it is the largest among that of the three methods. Only in one of the 10 runs the PSO obtains the best known solution for the problems so far. The basic BBO algorithm has better performance than the PSO, but the schedule result is not the best as it is not very stable. In 6 of 10 runs the basic BBO obtains the optimal solution and the HBBO gets the best schedule in all the 10runs. From the comparative results, it is obvious that the

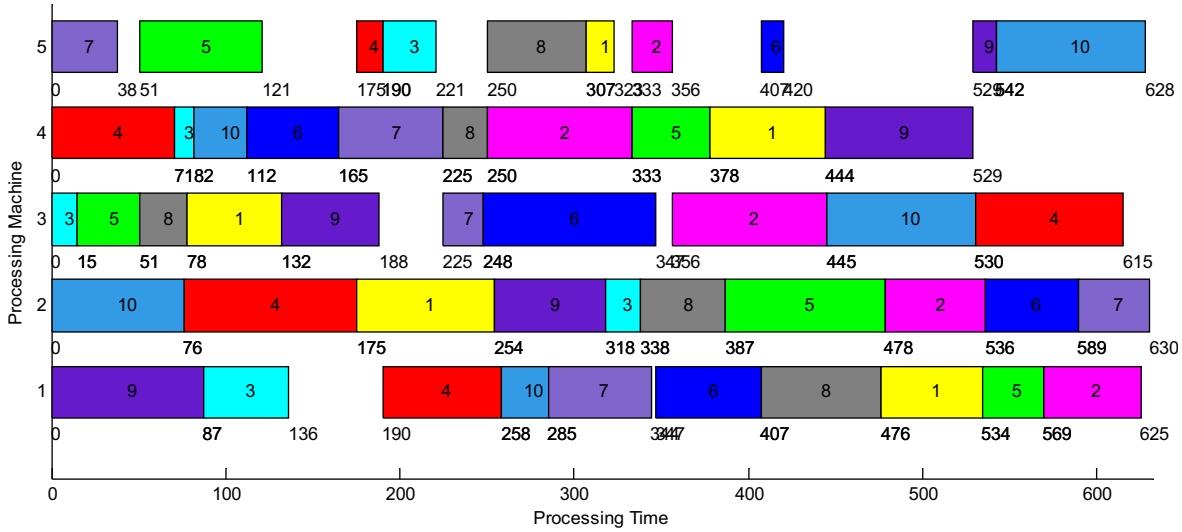

Fig. 18. GATT chart for the schedule of the 10*5 JSP optimized by the HBBO method.

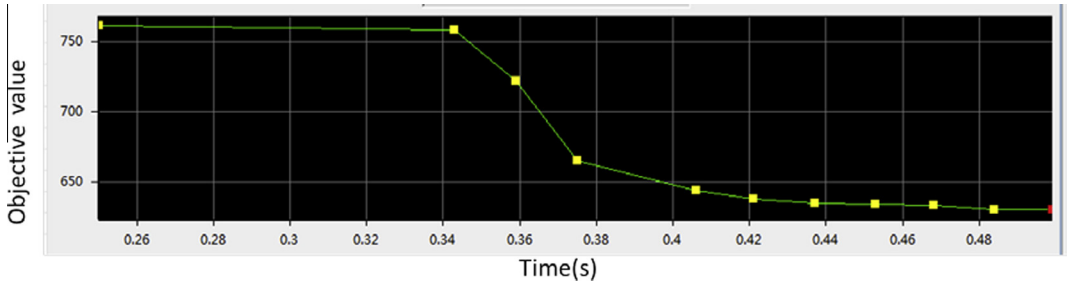

Fig. 19. Statistics of the CPLEX for the schedule of the 10*5 JSP. (The green line shows the tendency of the objective value's changing with time. The yellow points represent objective values of possible solutions and the last red point is objective value of the best solution.) (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

HBBO method is more reliable than the other two algorithms. Therefore, the HBBO can solve different JSP problems effectively and stably as it can find the best schedule in all runs.

The processing time and operation sequence for the 10*5 JSP instance are given in Table 2. Figs. 15–20 present the experimental results of different methods for the JSP with 10 jobs and 5 machines.

Figs. 15–18 show the GATT charts for the schedule of the 10*5 JSP without optimization and with the PSO, the basic BBO and the HBBO respectively. Fig. 19 gives statistics of the CPLEX for the schedule of the second instance, and Fig. 20 compares the evolution curves of the three methods. From the experimental results, it is obvious that the proposed algorithm performs better than the PSO and the basic BBO as the makespan of it is least.

Test on a more complicated JSP instance is conducted. Figs. 21–26 present the experimental results of different methods for the JSP with 20 jobs and 10 machines. The processing time and operation sequence for the 20*10 JSP instance is given in Table 3.

Figs. 21–24 give the GATT charts for the schedule of the 20*10 JSP with the PSO, the basic BBO and the HBBO respectively, and Fig. 25 gives statistics of the CPLEX for the schedule of the third instance, while the evolution curves of the three bio-inspired methods are given in Fig. 26. The same to the results of the 6*6 and 10*5 JSPs, the schedule of the HBBO algorithm is still the best. Moreover, the best makespan values of different algorithms for the three problem instances are provided in Table 4 in comparing to make the result more clear and convincing. From the experimental

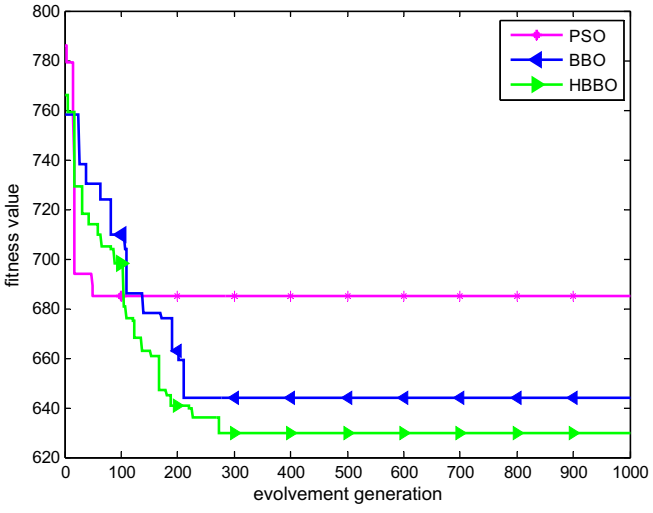

Fig. 20. Evolution curves of methods for the 10*5 JSP in comparison.

results mention above, it is obvious that the CPLEX performs better than the PSO and the basic BBO as it obtains the best solution for 6*6 and 10*5 instances. However for the 20*10 instance, the makespan value of the CPLEX is higher than the HBBO. The results show that the HBBO is more powerful than the PSO and the basic BBO, and is at least not worse than the CPLEX in solving the JSP.

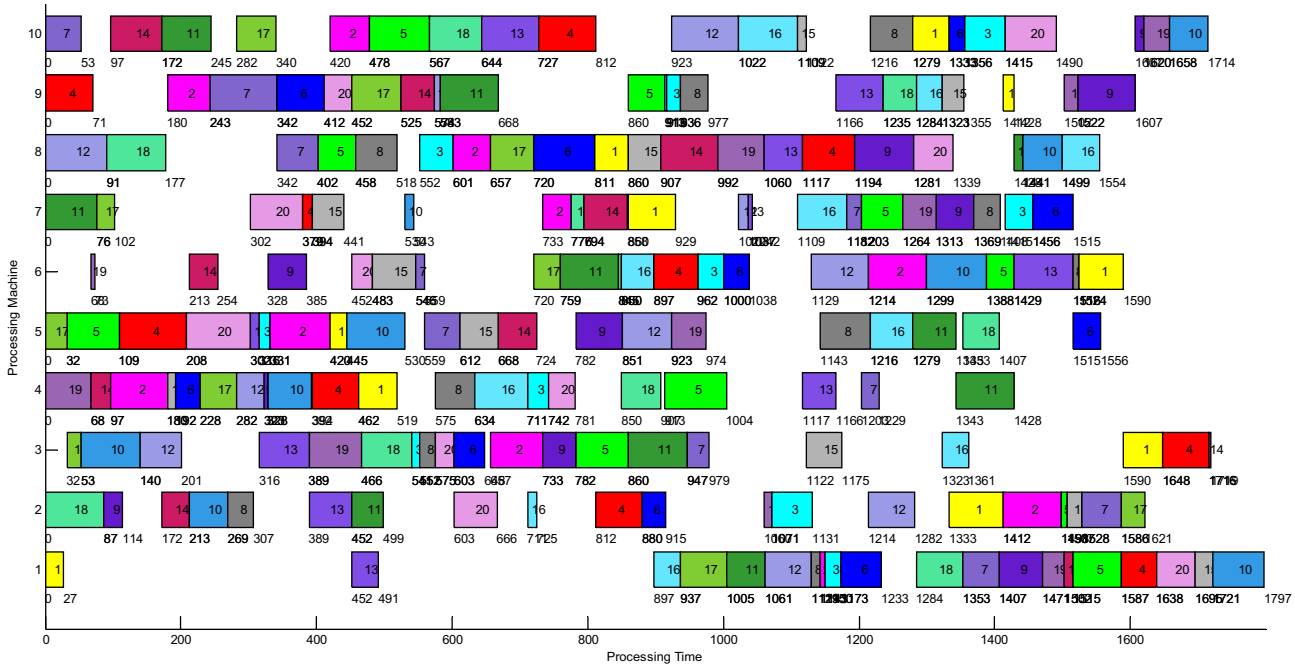

Fig. 21. GATT chart for a possible schedule of the 20*10 JSP without optimization.

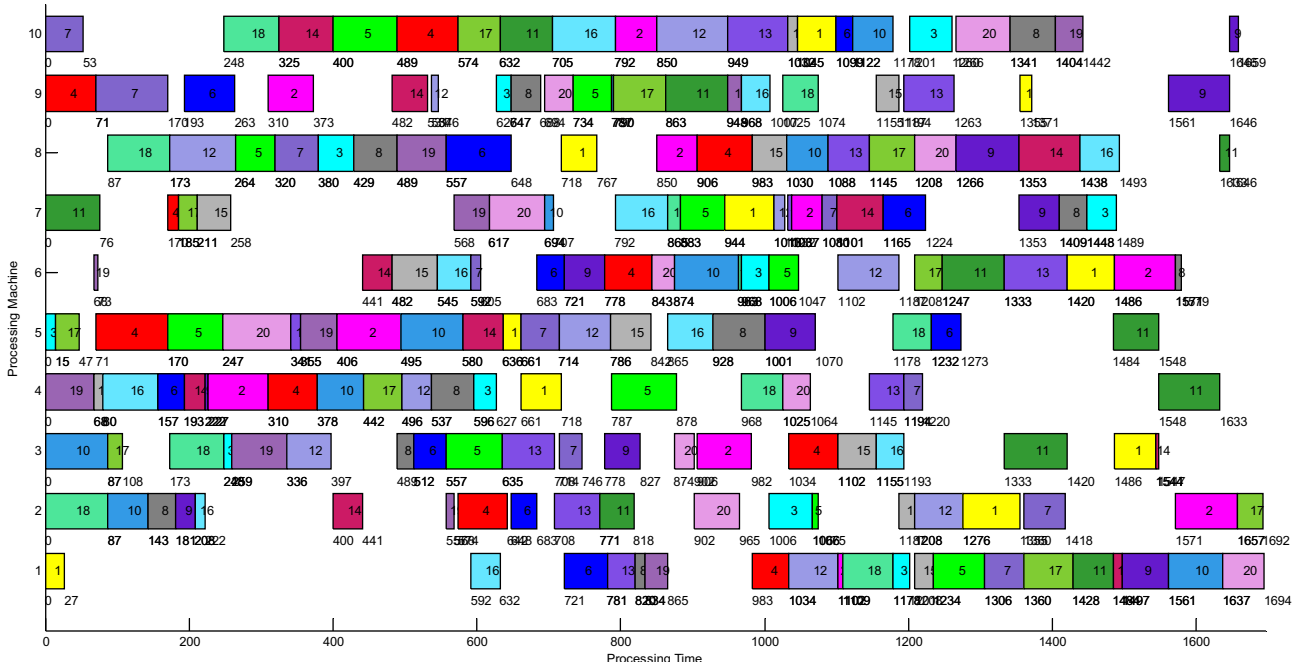

Fig. 22. GATT chart for the schedule of the 20*10 JSP optimized by the PSO method.

6.2. Experiment 2

Previous work has shown that there is rarely an ultimate method that can outperform other methods in all the problem cases. The second series of experiments aims at illustrating the effectiveness and performance of the HBBO for the JSP to minimize makespan. Several well-known benchmark problems with different scales are selected, including FT06, FT10, and FT20 (Fisher & Thompson, 1963), LA01–LA26 (Lawrence, 1984). These problems are widely used in the literature and available on the OR-Library web site (Beasley, 1990) (URL: http://people.brunel.ac.uk/~mastjjb/jeb/info.html). Numerical analysis based on those well-known benchmark problems can give some convincing explanations that our proposed method shows advantages.

In this experiment the results of the proposed HBBO algorithm are compared with that of numerical comparative algorithms from literatures, such as, GA(Zhao, 2011), SA (Lu, 2005), HEA (Ge, Du, & Qian, 2007), MPSO (Lin, Horng, Kao, Chen, & Run, 2010), HIA (Ge, Sun, Liang, & Qian, 2008), HGA-Param (Goncalves, Magalhaes Mendes, & Mauricio, 2005), SBI (Joseph

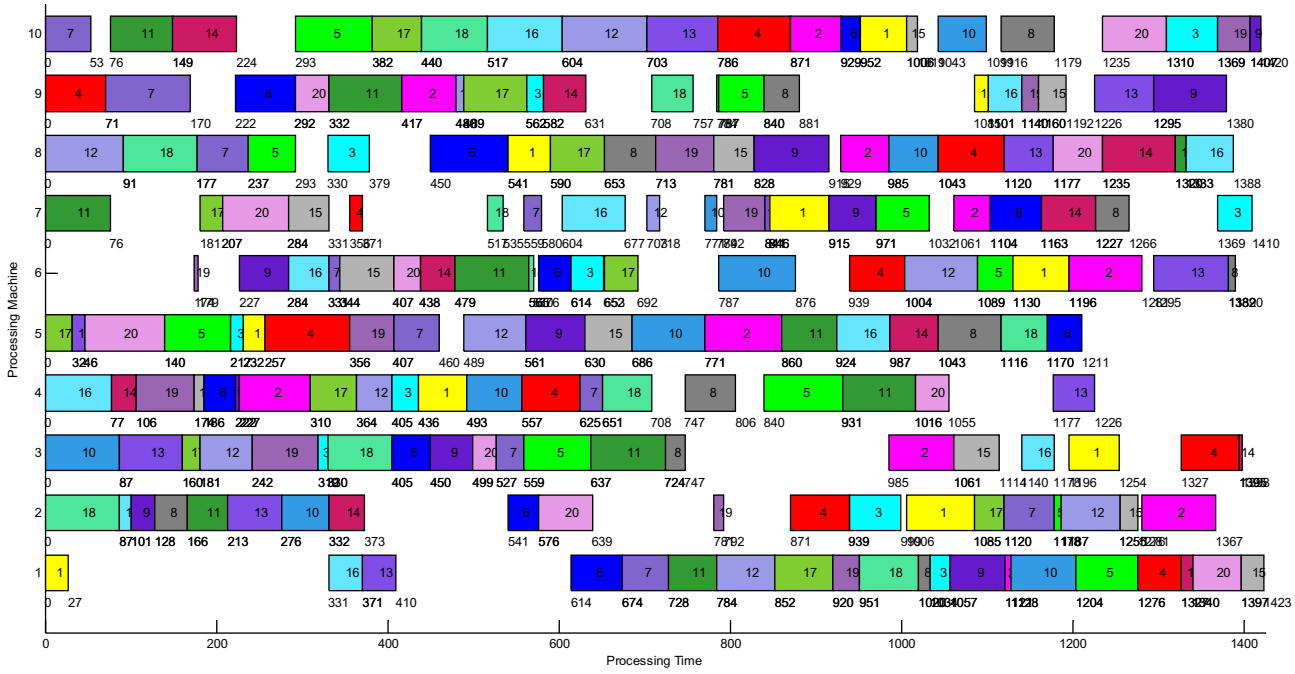

Fig. 23. GATT chart for the schedule of the 20*10 JSP optimized by the basic BBO method.

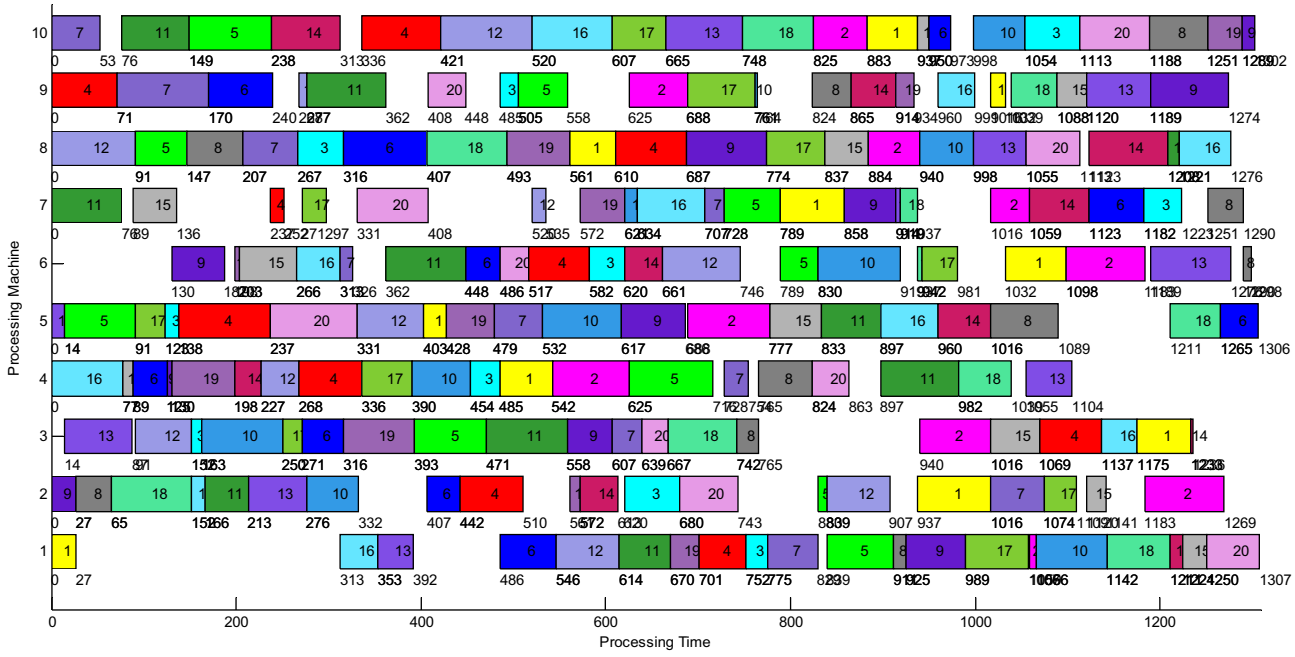

Fig. 24. GATT chart for the schedule of the 20*10 JSP optimized by the HBBO method.

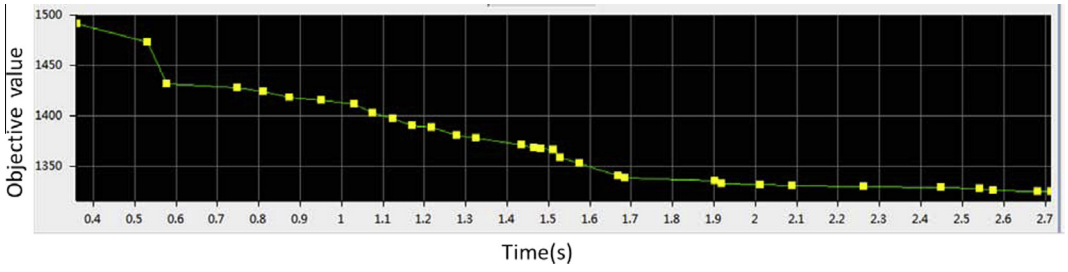

Fig. 25. Statistics of the CPLEX for the schedule of the 20*10 JSP. (The green line shows the tendency of the objective value's changing with time. The yellow points represent objective values of possible solutions and the last red point is objective value of the best solution.) (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

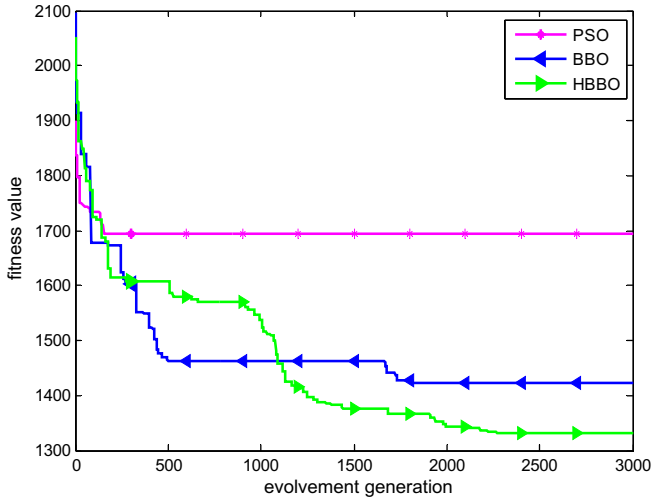

Fig. 26. Evolution curves of methods for the 20*10 JSP in comparison.

Table 3
Information of the 20*10 JSP instance.

Job	Operation sequence	Machine assigned	Processing time	Job	Operation sequence	Machine assigned	Processing time
Job1	O11	10	27	Job11	O1101	3	76
	O12	6	25		O1102	8	73
	O13	7	57		O1103	7	47
	O14	3	49		O1104	2	85
	O15	4	69		O1105	6	86
	O16	1	54		O1106	1	87
	O17	9	79		O1107	4	56
	O18	2	16		O1108	10	64
	O19	5	66		O1109	5	85
	O110	8	58		O1110	9	13
Job2	O21	7	83	Job12	O1201	6	91
	O22	2	63		O1202	8	61
	O23	6	89		O1203	9	41
	O24	1	58		O1204	10	9
	O25	3	56		O1205	1	72
	O26	8	76		O1206	4	99
	O27	4	43		O1207	3	15
	O28	10	7		O1208	7	68
	O29	5	85		O1209	2	85
	O210	9	86		O1210	5	68
Job3	O31	6	15	Job13	O1301	7	14
	O32	8	11		O1302	1	73
	O33	3	49		O1303	9	63
	O34	7	31		O1304	5	39
	O35	2	20		O1305	2	83
	O36	5	38		O1306	6	5
	O37	9	60		O1307	4	57
	O38	10	23		O1308	3	49
	O39	1	59		O1309	10	69
	O310	4	41		O1310	8	87
Job4	O41	2	71	Job14	O1401	7	29
	O42	6	99		O1402	4	75
	O43	4	15		O1403	5	41
	O44	7	68		O1404	6	41
	O45	1	85		O1405	3	49
	O46	9	68		O1406	1	56
	O47	5	65		O1407	8	64
	O48	3	77		O1408	2	85
	O49	10	51		O1409	9	13
	O410	8	68		O1410	10	3
Job5	O51	6	77	Job15	O1501	7	12
	O52	3	56		O1502	9	47
	O53	1	89		O1503	5	63
	O54	8	78		O1504	10	56
	O55	2	53		O1505	1	47
	O56	7	91		O1506	4	13
	O57	4	61		O1507	6	53

Table 3 (*continued*)

Job	Operation sequence	Machine assigned	Processing time	Job	Operation sequence	Machine assigned	Processing time
	O58	5	41		O1508	2	32
	O59	9	9		O1509	8	21
	O510	10	72		O1510	3	26
Job6	O61	7	36	Job16	O1601	7	77
	O62	2	70		O1602	9	14
	O63	8	45		O1603	5	47
	O64	3	91		O1604	10	40
	O65	9	35		O1605	1	87
	O66	5	38		O1606	4	73
	O67	10	60		O1607	6	63
	O68	1	23		O1608	2	39
	O69	4	59		O1609	8	38
	O610	6	41		O1610	3	55
Job7	O71	1	53	Job17	O1701	6	32
	O72	2	99		O1702	8	21
	O73	3	60		O1703	4	26
	O74	5	13		O1704	7	54
	O75	6	53		O1705	1	58
	O76	8	32		O1706	2	73
	O77	4	21		O1707	3	63
	O78	7	26		O1708	5	39
	O79	10	54		O1709	10	68
	O710	9	58		O1710	9	35
Job8	O81	9	38	Job18	O1801	9	87
	O82	3	60		O1802	3	86
	O83	8	23		O1803	8	75
	O84	7	59		O1804	1	77
	O85	2	41		O1805	4	18
	O86	10	14		O1806	5	5
	O87	6	73		O1807	7	57
	O88	1	63		O1808	2	49
	O89	4	39		O1809	10	69
	O810	5	8		O1810	6	54
Job9	O91	9	27	Job19	O1901	7	68
	O92	7	5		O1902	5	5
	O93	5	57		O1903	8	77
	O94	8	49		O1904	6	51
	O95	6	69		O1905	3	68
	O96	3	87		O1906	9	11
	O97	4	56		O1907	4	49
	O98	10	64		O1908	10	31
	O99	2	85		O1909	2	20
	O910	1	13		O1910	1	38
Job10	O101	8	87	Job20	O2001	6	94
	O102	9	56		O2002	4	77
	O103	7	64		O2003	2	40
	O104	6	85		O2004	5	31
	O105	4	13		O2005	8	28
	O106	2	3		O2006	9	63
	O107	5	89		O2007	7	39
	O108	3	58		O2008	3	58
	O109	1	56		O2009	1	75
	O1010	10	76		O2010	10	57

Table 4
The best makespan values of different algorithms for the three instances.

Problem instance	Without optimization	CPLEX	PSO	BBO	HBBO
6*6	66	55	55	55	55
10*5	776	630	685	644	630
20*10	1797	1325	1694	1423	1307

et al.,1988), SBII (Joseph et al.,1988), TSAB (Nowicki & Smutnicki, 1996), TSSB (Pezzella & Merelli, 2000), PSO-priority based (Sha & Hsu, 2006), PSO-permutation based (Sha & Hsu, 2006), HPSO (Sha & Hsu, 2006), and TS (Mauro & Marco, 1993). The CPLEX, the PSO, and the basic BBO are also tested on the benchmarks. The parameters of all the three bio-inspired intelligent algorithms (the PSO, the basic BBO and the HBBO) and the exact solver CPLEX are given in the beginning of Section 6.

Table 5
Makespans of HBBO, PSO, BBO, CPLEX and some literature results.

Instance	$n*m$	BK	GA	SA	HEA	MPSO	HIA	HGA-Param	SBI	SBII	TSAB	TSSB
FT06	6*6	55	–	–	55	55	55	55	55	55	55	55
FT10	10*10	930	–	–	930	930	930	930	1015	930	930	930
FT20	20*5	1165	–	–	1169	1165	1165	1165	1290	1178	1165	1165
LA01	10*5	666	666	674	666	666	666	666	666	666	666	666
LA02	10*5	655	666	671	655	655	655	655	720	669	655	655
LA03	10*5	597	666	619	597	597	597	597	623	605	597	597
LA04	10*5	590	–	602	590	590	590	590	597	593	590	590
LA05	10*5	593	–	593	593	593	593	593	593	593	593	593
LA06	15*5	926	926	926	926	926	926	926	926	926	926	926
LA07	15*5	890	–	–	890	890	890	890	890	890	890	890
LA08	15*5	863	–	–	863	863	863	863	868	863	863	863
LA09	15*5	951	–	–	951	951	951	951	951	951	951	951
LA10	15*5	958	–	–	958	958	958	958	959	959	958	958
LA11	20*5	1222	1222	1222	1222	1222	1222	1222	1222	1222	1222	1222
LA12	20*5	1039	1070	–	1039	1039	1039	1039	1039	1039	1039	1039
LA13	20*5	1150	1180	–	1150	1150	1150	1150	1150	1150	1150	1150
LA14	20*5	1292	1292	–	1292	1292	1292	1292	1292	1292	1292	1292
LA15	20*5	1207	1215	–	1207	1207	1207	1207	1207	1207	1207	1207
LA16	10*10	945	979	1005	945	945	945	945	1021	978	945	945
LA17	10*10	784	784	793	784	784	784	784	796	787	784	784
LA18	10*10	848	848	–	848	848	848	848	891	859	848	848
LA19	10*10	842	855	–	–	842	842	842	875	860	842	842
LA20	10*10	902	950	1007	–	902	902	907	924	914	902	902
LA21	15*10	1046	1097	1134	1046	1046	1046	1046	1172	1084	1047	1046
LA22	15*10	927	940	1122	935	932	932	935	1040	944	927	927
LA23	15*10	1032	1040	–	1032	1032	1032	1032	1061	1032	1032	1032
LA24	15*10	935	953	–	–	941	950	953	1000	976	939	938
LA25	15*10	977	990	–	–	977	979	986	1048	1017	977	979
LA26	20*10	1218	1231	1355	1218	1218	1218	1218	1304	1224	1218	1218
LA31	30*10	1784	–	–	1784	1784	1784	1784	1784	1784	1784	1784
LA36	15*15	1268	–	–	1287	1278	1281	1279	1351	1305	1268	1268

Instance	PSO-priority based	PSO-permutation based	HPSO	TS	CPLEX	PSO		BBO		HBBO	
						MS*	Iter	MS	Iter	MS	Iter
FT06	55	55	55	55	55	55	1000	55	354	55	253
FT10	1007	937	930	935	945	1040	3000	1001	3000	930	439
FT20	1242	1165	1165	1165	1270	1238	3000	1182	3000	1165	497
LA01	681	666	666	666	666	666	496	666	310	666	203
LA02	694	655	655	655	666	665	1000	655	724	655	309
LA03	633	597	597	597	597	619	1000	608	1000	597	373
LA04	611	590	590	590	590	607	1000	603	1000	590	389
LA05	593	593	593	593	593	593	785	593	873	593	243
LA06	926	926	926	926	926	926	3000	926	1989	926	522
LA07	890	890	890	890	890	890	1877	890	1520	890	580
LA08	863	863	863	863	863	863	1722	863	1358	863	493
LA09	953	951	951	951	951	956	3000	951	2105	951	649
LA10	958	958	958	958	958	958	2055	958	1799	958	652
LA11	1222	1222	1222	1222	1222	1222	3000	1222	2005	1222	973
LA12	1039	1039	1039	1039	1039	1103	3000	1053	3000	1039	1169
LA13	1150	1150	1150	1150	1150	1173	3000	1168	3000	1150	1204
LA14	1292	1292	1292	1292	1292	1292	2300	1292	2298	1292	822
LA15	1232	1207	1207	1207	1207	1231	3000	1223	3000	1207	1254
LA16	1006	945	945	945	985	1033	3000	998	3000	945	1203
LA17	833	784	784	784	785	803	3000	797	3000	784	1223
LA18	901	848	848	848	861	870	3000	867	3000	848	925
LA19	895	842	842	842	860	867	3000	866	3000	842	873
LA20	963	907	902	902	907	1002	3000	918	3000	902	643
LA21	1201	1055	1046	1046	1095	1173	3000	1146	3000	1046	799
LA22	1046	935	927	933	937	977	3000	951	3000	933	3000
LA23	1146	1032	1032	1032	1032	1082	3000	1045	3000	1032	1062
LA24	1082	937	935	941	963	992	3000	979	3000	935	1179
LA25	1107	983	977	979	995	1035	3000	1013	3000	977	1300
LA26	1409	1218	1218	1218	1237	1358	3000	1275	3000	1218	1266
LA31	1784	1784	1784	1784	1784	1868	3000	1813	3000	1784	2398
LA36	1437	1291	1268	1278	1312	1373	3000	1331	3000	1268	2466

Note: MS represents makespan and Iter represents iteration.

The comparative results are shown in Tables 5–7 and Fig. 27. The first column of Table 5 gives the problem names. The second column $n*m$ means that the JSP instance involves n jobs and m machines. BK represents the best known solution for each instance. In Table 5 the makespan time of 18 algorithms are given. The parameters are set according to the rule that is mentioned in the beginning of Section 6. As the iteration numbers can reflect the efficiency directly and also reveal the time consuming proportion of the method, the iteration numbers of the PSO, the basic BBO and the HBBO are also presented. Each instance is executed for 5 runs. Since that the convergence character of each method is different, not all the three algorithms can

Table 6
Bias% of HBBO, PSO, BBO, CPLEX and some literature results.

Instance	GA	SA	HEA	MPSO	HIA	HGA-Param	SBI	SBII	TSAB
FT06	–	–	0	0	0	0	0	0	0
FT10	–	–	0	0	0	0	9.1398	0	0
FT20	–	–	0.3433	0	0	0	10.7296	1.1159	0
LA01	0	1.2012	0	0	0	0	0	0	0
LA02	1.6794	2.4427	0	0	0	0	9.9237	2.1374	0
LA03	11.5578	3.6851	0	0	0	0	4.3551	1.3400	0
LA04	–	2.0339	0	0	0	0	1.1864	0.5085	0
LA05	–	0	0	0	0	0	0	0	0
LA06	0	0	0	0	0	0	0	0	0
LA07	–	–	0	0	0	0	0	0	0
LA08	–	–	0	0	0	0	0.5794	0	0
LA09	–	–	0	0	0	0	0	0	0
LA10	–	–	0	0	0	0	0.1044	0.1044	0
LA11	0	0	0	0	0	0	0	0	0
LA12	2.9836	–	0	0	0	0	0	0	0
LA13	2.6087	–	0	0	0	0	0	0	0
LA14	0	–	0	0	0	0	0	0	0
LA15	0.6628	–	0	0	0	0	0	0	0
LA16	3.5979	6.3492	0	0	0	0	8.0423	3.4921	0
LA17	0	1.1480	0	0	0	0	1.5306	0.3827	0
LA18	5.3215	–	0	0	0	0	5.0708	1.2972	0
LA19	4.8757	–	–	0	0	0	3.9192	2.1378	0
LA20	1.4024	11.6408	–	0	0	0.5543	2.4390	1.3304	0
LA21	1.0673	8.4130	0	0	0	0	12.0459	3.6329	0.0956
LA22	2.6087	21.0356	0.8630	0.5394	0.5394	0.8630	12.1899	1.8339	0
LA23	0	–	0	0	0	0	2.8101	0	0
LA24	0.6628	–	–	0.6417	1.6043	1.9251	6.9519	4.3850	0.4278
LA25	3.5979	–	–	0	0.2047	0.9212	7.2671	4.0942	0
LA26	0	11.2479	0	0	0	0	7.0608	0.4926	0
LA31	–	–	0	0	0	0	0	0	0
LA36	–	–	1.4984	0.7886	1.0252	0.8675	6.5457	2.9180	0
Average	2.3838	5.3229	0.1002	0.0635	0.1088	0.1655	3.6094	1.0065	0.0169

Instance	TSSB	PSO-priority based	PSO-permutation based	HPSO	TS	CPLEX	PSO	BBO	HBBO
FT06	0	0	0	0	0	0	0	0	0
FT10	0	8.2796	0.7527	0	0.5376	1.6129	11.8280	7.6344	0
FT20	0	6.6094	0	0	0	9.0129	6.2661	1.4592	0
LA01	0	2.2523	0	0	0	0	0	0	0
LA02	0	5.9542	0	0	0	1.6794	1.5267	0	0
LA03	0	6.0302	0	0	0	0	3.6851	1.8425	0
LA04	0	3.5593	0	0	0	0	2.8814	2.2034	0
LA05	0	0	0	0	0	0	0	0	0
LA06	0	0	0	0	0	0	0	0	0
LA07	0	0	0	0	0	0	0	0	0
LA08	0	0	0	0	0	0	0	0	0
LA09	0	0.2103	0	0	0	0	0.5258	0	0
LA10	0	0	0	0	0	0	0	0	0
LA11	0	0	0	0	0	0	0	0	0
LA12	0	0	0	0	0	0	6.1598	1.3474	0
LA13	0	0	0	0	0	0	2.0000	1.5652	0
LA14	0	0	0	0	0	0	0	0	0
LA15	0	2.0713	0	0	0	0	1.9884	1.3256	0
LA16	0	6.4550	0	0	0	4.2328	9.3122	5.6085	0
LA17	0	6.2500	0	0	0	0.1276	2.4235	1.6582	0
LA18	0	6.2500	0	0	0	1.5330	2.5943	2.2406	0
LA19	0	6.2945	0	0	0	2.1378	2.9691	2.8504	0
LA20	0	6.7627	0.5543	0	0	0.5543	11.0865	1.7738	0
LA21	0	14.8184	0.8604	0	0	4.6845	12.1415	9.5602	0
LA22	0	12.8371	0.8630	0	0.6472	1.0787	5.3937	2.5890	0.6472
LA23	0	11.0465	0	0	0	0	4.8450	1.2597	0
LA24	0.3209	15.7219	0.2139	0	0.6417	2.9947	6.0963	4.7059	0
LA25	0.2047	13.3060	0.6141	0	0.2047	1.8424	5.9365	3.6847	0
LA26	0	15.6814	0	0	0	1.5599	11.4943	4.6798	0
LA31	0	0	0	0	0	0	4.7085	1.6256	0
LA36	0	13.3281	1.8139	0	0.7886	3.4700	8.2808	4.9685	0
Average	0.0170	5.2812	0.1830	0	0.0910	1.1781	4.0046	2.0833	0.0209

obtain the best solution of each instance. Therefore, if the final best solutions are obtained within the given iterations, the columns of 'Iter' in Table 5 give the least iterations that the algorithms cost to obtain the optimal solution. However if the algorithm can never achieve the optima, the numbers of iterations set at the beginning of the algorithm are given in that column. The *bias%* between every algorithm and the best known solution is given in Table 6 and it is computed according to the following formula.

$$bias\% = \frac{MS_i^x - BK_i}{BK_i} * 100 \tag{8}$$

Table 7
Statistics of HBBO, PSO, BBO, CPLEX and some literature results.

Algorithm	GA	SA	HEA	MPSO	HIA	HGA-Param	SBI	SBII	TSAB	TSSB
N_OBK	5	3	23	27	26	25	12	15	28	28
N_IT	15	13	27	31	31	31	31	31	31	31
N_OBK/N_IT	0.3333	0.2308	0.8519	0.8710	0.8387	0.8065	0.3871	0.4839	0.9032	0.9032
sgap	308	652	31	21	35	51	1073	295	5	5
sgap%	2.1131	5.8893	0.1211	0.0718	0.1196	0.1743	3.6675	1.0083	0.0171	0.0171

Algorithm	PSO-priority based	PSO-permutation based	HPSO	TS	CPLEX	PSO	BBO	HBBO
N_BK	11	23	31	25	17	9	11	30
N_IT	31	31	31	31	31	31	31	31
N_OBK/N_IT	0.3548	0.7419	1	0.8065	0.5484	0.2903	0.3548	0.9678
sgap	1594	60	0	29	376	1273	651	6.0000
sgap%	5.4483	0.2051	0	0.0991	1.2852	4.3511	2.2251	0.0205

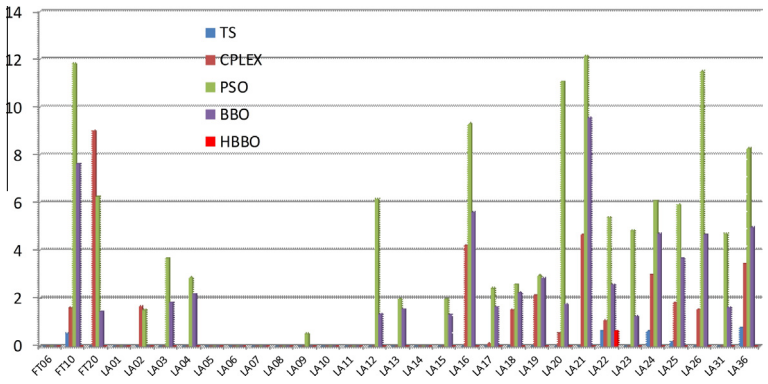

Fig. 27. The *bias%* of CPLEX, PSO, BBO and HBBO for each instance.

Table 8
Makespans of CPLEX, PSO, BBO, and HBBO.

Instance	n^*m	BK	CPLEX	PSO	BBO	HBBO
ORB1	10*10	1059	1089	1137	1126	1059
ORB2	10*10	888	900	992	945	919
ABZ5	10*10	1234	1270	1286	1272	1250
ABZ6	10*10	943	948	984	979	948
ABZ9	20*15	679	758	844	821	734

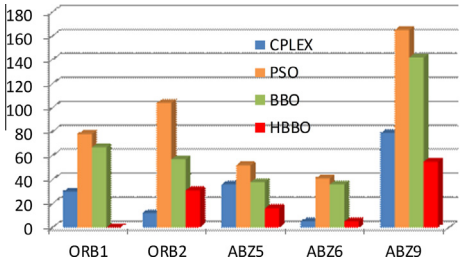

Fig. 28. Gaps between the obtained and the best known makepan of each method.

where MS_i^x is the makespan of instance i that is obtained by method x, BK_i is the best known makespan of instance i. The last row of Table 6 gives the average *bias%* of each method of all the 31 instances. The *bias%*s of the CPLEX, the PSO, the basic BBO and the HBBO for each instance are also given in Fig. 27 to make it more convenient for comparing.

Since that the number of instances tested by each method is different, it is not very convincing to compare the result by average *bias%* directly. Therefore, the statistics of the number that best known solution is obtained (N_OBK) and the total number of instance tested by each method (N_IT) is given in Table 7. The sum of gaps (*sgap*) between the obtained and the best known makepan of all the instances that are tested by each method is given. Similar to the *bias%*, the *sgap%* is obtained by

$$Sgap\% = \frac{\sum_i gap_i}{\sum_i BK_i} * 100 \tag{9}$$

where i represents that the instance i is tested by the corresponding method, *sgap* denotes the sum of all the gaps between the makespan obtained by each method and the best known.

From Tables 5–7, it is obvious that our proposed HBBO method outperforms all other algorithms except the HPSO. The HBBO obtained the best-known solution for 30 of the 31 instances, while the basic BBO obtained 11 of the 31 instances. The iterations of the HBBO are apparently less than that of the PSO and the basic BBO as given in Table 5 for each instance. In addition the *bias%* and *sgap%* of the HBBO is lower than most of the competitors.

6.3. Experiment 3

Our proposed HBBO is tested on the ORB (Joseph et al., 1988) and ABZ (Applegate & Cook, 1991) benchmarks to further illuminate the advantage of the HBBO. The computational results of the HBBO are particularly compared with that of the CPLEX, the PSO, and the basic BBO. The CPLEX, the PSO, the basic BBO and the HBBO are tested on 5 tougher instances (ORB1, ORB2, ABZ5, ABZ6 and ABZ9) and the experimental results are presented in Table 8. The gaps between the makepan obtained by each method and the best known makepan of all the instances are given in Fig. 28. From the comparative results, we can see that the HBBO approach dramatically outperforms the PSO and the basic BBO for all the instances, and can also obtain better solutions than the CPLEX for four of the five instances.

From the three series of experiments, we can see that the HBBO can make better performance than the CPLEX, the PSO, the basic BBO and most of the literature results. From the results of the basic BBO and other basic methods we can see that the basic BBO performs better than the PSO, the GA and the SA as it can obtain smaller makespan. Moreover, the convergence character of the basic BBO is also better than that of the PSO. Therefore, the basic BBO is a competitive bio-inspired algorithm. Comparing the results of the HBBO algorithm with that of the basic BBO, we can see that the benefits from the chaos theory and the "searching around the optimum" strategy for solving the job shop scheduling problems is significant. The CPLEX can report feasible solutions for most instances, but the bias and gaps of it is larger than that of the HBBO. In addition, the number of the best-known solutions obtained by the CPLEX is less than that of the HBBO, as indicated in Tables 6 and 7. Results of other 14 methods from literature are also given but most of their results are inferior to ours except the HPSO which achieves the best-known solutions for all the 31instances as shown in experiment 2. To sum up in conclusion the proposed HBBO is demonstrated to be competitive as it can make better schedule within less iterations.

7. Conclusions

This paper develops a novel HBBO algorithm to solve the JSP problem. BBO is a new simulated bio-inspired intelligent algorithm which has some features that are unique to other biology-based optimization methods. To reduce the computation time and to guarantee avoidance of the local minimum, the proposed HBBO method integrates the chaos theory and the "searching around the optimum" strategy into the basic BBO algorithm and takes advantages of the efficiency, accuracy and stability of them.

Different types of experiments are conducted to highlight the advantages of our approach over the existing methods. Our proposed HBBO is tested and approved in numerical simulation based on both self-designed numerical examples and 36 standard benchmark instances that are taken from the OR-Library. Series of the simulation results highlight the advantages of our approach over numerical existing methods. It should be noticed that the HBBO generates satisfactory job-shop scheduling successfully within less iterations, while the basic BBO and the PSO algorithms converge to local best solutions easily. Therefore, the chaos theory and the "searching around the optimum" strategy can consistently improve the performance of the original algorithm. It may finally be concluded that our aggregation method HBBO is a more effective and robust method for the JSP as it outperforms the PSO, the basic BBO, the CPLEX, and the other state-of-the-art numerical methods.

Our future work will focus on applying this novel technique to more complicated JSPs, and implement this hybrid approach in enterprise systems.

Acknowledgements

This work was partially supported by National Key Basic Research Program of China (973 Project) under grant #2013CB035503 and #2014CB046401, Natural Science Foundation of China (NSFC) under grant #61333004, #61273054 and #61175109, National Magnetic Confinement Fusion Research Program of China under grant #2012GB102006, Top-Notch Young Talents Program of China, Aeronautical Foundation of China under grant #20135851042, and the Fundamental Research Funds for the Central Universities of China under grant #YWF-11-03-Q-012.

References

Allaoui, H., & Artiba, A. (2004). Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. *Computers and Industrial Engineering, 47*(4), 431–450.

Anitha, J., & Karpagam, M. (2013). Ant colony optimization using pheromone updating strategy to solve job shop scheduling. *Proceedings of the 7th international conference on intelligent systems and control, Coimbatore, Tamil Nadu, India*, 367–372.

Applegate, D., & Cook, W. (1991). A computational study of the job shop scheduling problem. *ORSA Journal on Computing, 3*(2), 149–156.

Atabak, E., Solimanpurb, M., Topaloglua, S., & Elmic, A. (2011). A simulated annealing algorithm for the job shop cell scheduling problem with intercellular moves and reentrant parts. *Computers and Industrial Engineering, 61*(1), 171–178.

Azzouz, A., Ennigrou, M., Jlifi, B., & Ghédira, K. (2012). Combining tabu search and genetic algorithm in a multi-agent system for solving flexible job shop problem. *Proceedings of the 11th Mexican international conference on artificial intelligence. Advances in artifical intelligence and applications, San Luis Potosi*, 83–88.

Balas, E., & Vazacopoulos, A. (1998). Guided local search with shifting bottleneck for job shop scheduling. *Management Science, 44*(2), 262–275.

Bansal, A. K., Kumar, R., & Gupta, R. A. (2013). Economic analysis and power management of a small autonomous hybrid power system (SAHPS) using biogeography based optimization (BBO) algorithm. *IEEE Transactions on Smart Grid, 4*(1), 638–648.

Beasley, J. E. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society, 41*, 1069–1072.

Brucker, P., & Neyer, J. (1998). Tabu-search for the multi-mode job-shop problem. *Operations Research Spektrum, 20*(1), 21–28.

Chen, W., & Zhang, J. (2013). Ant colony optimization for software project scheduling and staffing with an event-based scheduler. *IEEE Transactions on software engineering, 39*(1), 1–17.

Costa e Silva, M. A., Coelho, L. D. S., Lebensztajn, L., & Lebensztajn, Luiz (2012). Multiobjective biogeography-based optimization based on predator-prey approach. *IEEE Transactions on Magnetics, 48*(2), 951–954.

De Giovanni, L., & Pezzella, F. (2010). An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. *European Journal of Operational Research, 200*(2), 395–408.

Deming, L. (2012). Multi-objective artificial bee colony for interval job shop scheduling with flexible maintenance. *International Journal of Advanced Manufacturing Technology, 6*(9–12), 1–9.

Duan, H., Shao, S., Su, B., & Zhang, L. (2010a). New development thoughts on the bio-inspired intelligence based control for unmanned combat aerial vehicle. *Science China Technological Sciences, 53*(8), 2025–2031.

Duan, H., Xu, C., & Xing, Z. (2010b). A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems. *International Journal of Neural Systems, 20*(1), 39–50.

Eshlaghy & Sheibatolhamdy (2011). Scheduling in flexible job-shop manufacturing system by improved tabu search. *African Journal of Business Management, 5*(12), 4863–4872.

Eugeniusz, N., & Czesław, S. (2005). An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling, 8*(2), 145–159.

Fisher, H., & Thompson, G. L. (1963). *Industrial scheduling*. Englewood Cliffs, NJ: Prentice-Hall.

Ge, H., Du, W., & Qian, F. (2007). A hybrid algorithm based on particle swarm optimization and simulated annealing for job shop scheduling. *Proceedings of the 3rd international conference on natural computation, Haikou, China*, 715–719.

Ge, H., Sun, L., Liang, Y., & Qian, F. (2008). An effective PSO and AIS-based hybrid intelligent algorithm for job-shop scheduling. *IEEE Transactions on Systems, Man and Cybernetics, Part A, 38*(2), 358–368.

Goncalves, J., Magalhaes Mendes, J., & Mauricio, M. G. C. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research, 167*(1), 77–95.

Gu, W., Tang, D., & Zheng, K. (2012). Minimizing makespan in job-shop scheduling problem using an improved adaptive particle swarm optimization algorithm. *Proceedings of the 24th chinese control and decision conference, Taiyuan, China*, 3189–3193.

Jain, P., Agarwal, A., Gupta, N., Sharma, R., Paliwal, U., & Bhakar, R. (2012). Profit maximization of a generation company based on biogeography based optimization. *Proceedings of the IEEE power and energy society general meeting, San Diego, USA*, 1–6.

Jamuna, K., & Swarup, K. S. (2011). Biogeography based optimization for optimal meter placement for security constrained state estimation. *Swarm and Evolutionary Computation, 1*(6), 89–96.

Joseph, A., Egon, B., & Daniel, Z. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science, 34*(3), 391–401.

Lawler, E. L., Lenstra, J. K., & Rinnooy Kan Alexander, H. G. (1982). *Recent developments in deterministic sequencing and scheduling*. Afdeling Mathematische Besliskunde: Mathematisch Centrum.

Lawrence, S. R. (1984). *Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques*. Pittsburgh, PA: Graduate School of Industrial Administration (GSIA), Carnegie Mellon University.

Li, Xiangtao, Wang, W., Zhou, J., & Yin, M. (2011). A perturb biogeography based optimization with mutation for global numerical optimization. *Applied Mathematics and Computation, 218*(2), 598–609.

Lian, Z., Jiao, B., & Gu, X. (2006). A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan. *Applied Mathematics and Computation, 183*(2), 1008–1017.
Lin, T., Horng, S., Kao, T., Chen, Y., & Run, R. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications, 37*(3), 2629–2636.
Liu, F., Duan, H., & Deng, Y. (2012). A chaotic quantum-behaved particle swarm optimization based on lateral inhibition for image matching. *Optik, 123*(21), 1955–1960.
Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences, 20*(3), 130–141.
Lu, T. (2005). *Genetic algorithm on solving job shop scheduling problem*. Dalian University of Technology. Master Thesis.
Ma, Y., Chu, C., & Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. *Computers and Industrial Engineering, 58*(2), 199–211.
Ma, H., & Simon, D. (2011). Blended biogeography-based optimization for constrained optimization. *Engineering Applications of Artificial Intelligence, 24*(3), 517–525.
Mauro, D., & Marco, T. (1993). Applying tabu search to the job shop scheduling problem. *Annals of Operations Research, 41*(3), 231–252.
Michael, Y., Bulysheva, L. A., Emelyanenko, A. A., & Emelyanenko, V. A. (2013). *Enterprise Information Systems, 7*(2), 169–186.
Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the job-shop problem. *Management Science, 42*(6), 797–813.
Pezzella, F., & Merelli, E. (2000). A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research, 120*(2), 297–310.
Pezzella, F., Morganti, G., & Ciaschetti, G. (2007). A genetic algorithm for the flexible job-shop scheduling problem. *Computers and Operations Research, 35*(10), 3202–3212.
Sayed, M. M., Saad, M. S., Emara, H. M., & Abou El-Zahab, E. E. (2013). A novel method for type-2 fuzzy logic controller design using a modified biogeography-based optimization. *Proceedings of the IEEE international conference on industrial technology (ICIT), Cape Town, Western Cape, South Africa*, 28–33.
Sels, V., Gheysen, N., & Vanhoucke, M. (2012). A comparison of priority rules for the job shop scheduling problem under different flow time - and tardiness-related objective functions. *International Journal of Production Research, 50*(15), 4255–4270.
Sha, D. Y., & Hsu, C. (2006). A hybrid particle swarm optimization for job shop scheduling problem. *Computers and Industrial Engineering, 51*(4), 791–808.
Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*(6), 702–713.
Simon, D., Ergezer, M., & Du, D. (2009). Population distributions in bio-geography-based optimization algorithms with elitism. *Proceedings of the IEEE international conference on systems, man, and cybernetics. San Antonio, TX, USA*, 991–996.
Sun, C., Duan, H., & Shi, Y. (2013). Optimal satellite formation reconfiguration based on closed-loop brain storm optimization. *IEEE Computational Intelligence Magazine, 8*(4), 39–51.
Sun, W., Pan, Y., Lu, X., & Ma, Q. (2010). Research on flexible job-shop scheduling problem based on a modified genetic algorithm. *Journal of Mechanical Science and Technology, 24*(10), 2119–2125.
Tasgetiren, M. F., Pan, Q. K., Suganthan, P. N., & Chen, A. H. L. (2011). A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. *Information Sciences, 181*(12), 3459–3475.
Van Laarhoven, P. J. M., Aarts, E. H. L., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations Research, 40*(1), 113–125.
Wang, X., & Duan, H. (2013). Predator-prey biogeography-based optimization for bio-inspired visual attention. *International Journal of Computational Intelligence Systems, 6*(6), 1151–1162.
Wang, W., Wang, Y., Yu, H., & Zhang, C. (2013a). Dynamic balance adaptive ant colony optimization algorithm for job-shop scheduling. *Proceedings of the 5th conference on measuring technology and mechatronics sutomation, Hong Kong, China*, 496–499.
Wang, X., Duan, H., & Luo, D. (2013b). Cauchy biogeography-based optimization based on lateral inhibition for image matching. *Optik, 124*(22), 5447–5453.
Watson, J. P., Christopher Beck, J., Howe, A. E., & Darrell Whitley, L. (2003). Problem difficulty for tabu search in job-shop scheduling. *Artificial Intelligence, 143*(2), 189–217.
Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering, 48*(2), 409–425.
Xia, W., Wu, Z., Zhang, W., & Yang, G. (2004). Applying particle swarm optimization to job-shop scheduling problem. *Journal of Mechanical Engineering, 17*(3), 437–441.
Xing, L., & Chena, Y. (2010). A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Applied Soft Computing, 10*(3), 888–896.
Xu, C., Duan, H., & Liu, F. (2010). Chaotic artificial bee colony approach to uninhabited combat air vehicle (UCAV) path planning. *Aerospace Science and Technology, 14*(8), 535–541.
Yu, J., & Duan, H. (2013). Artificial bee colony approach to information granulation-based fuzzy radial basis function neural networks for image fusion. *Optik, 124*(17), 3103–3111.
Zhang, X. (2010). Job-shop scheduling problems using timed planning. *Proceedings of the 4th IEEE international conference on secure software integration and reliability improvement companion, Singapore*, 110–117.
Zhang, Q., & Duan, H. (2014). Biological weight delection of multi-scale retinex via artificial bee colony algorithm. *Optik, 125*(3), 1434–1438.
Zhang, G., Shao, Xinyu, Lia, P., & Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers and Industrial Engineering, 56*(4), 1309–1318.
Zhao, W. (2011). Job shop scheduling based on genetic algorithm and simulated annealing algorithm. *Computer Simulation, 28*(7), 361–364.