Scientific Research and Essays Vol. 6(10), pp. 2078-2100, 18 May, 2011
Available online at http://www.academicjournals.org/SRE
DOI: 10.5897/SRE10.818

Full Length Research Paper

A hybrid bio-geography based optimization for permutation flow shop scheduling

Minghao Yin[1,2] and Xiangtao Li[1,2]*

[1]College of Computer Science, Northeast Normal University, Changchun, 130117, P. R. China.
[2]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, 130012, Changchun, P. R. China.

Accepted 30 November, 2010

The permutation flow shop problem (PFSSP) is an NP-hard problem of wide engineering and theoretical background. In this paper, a biogeography based optimization (BBO) based on memetic algorithm, named HBBO is proposed for PFSSP. Firstly, to make BBO suitable for PFSSP, a new LRV rule based on random key is introduced to convert the continuous position in BBO to the discrete job permutation. Secondly, the NEH heuristic was combined with the random initialization to initialize the population with certain quality and diversity. Thirdly, a fast local search is used for enhancing the individuals with a certain probability. Fourthly, the pair wise based local search is used to enhance the global optimal solution and help the algorithm to escape from local minimum. Additionally, simulations and comparisons based on PFSSP benchmarks are carried out, showing that our algorithm is both effective and efficient.

Key words: Biogeography based optimization, permutation flow shop scheduling, memetic algorithm, local search.

INTRODUCTION

Scheduling problems play an important role in both manufacturing systems and industrial process for improving the utilization of resources, and therefore it is crucial to develop efficient scheduling technologies (Stadtler, 2005). The permutation flow shop problem (PFSSP), one of the best known production scheduling problems, can be viewed as a simplified version of the flow shop problem, and had been proved to be non-deterministic-polynomial-time (NP)-hard (Garey and Johnson, 1979; Rinnooy, 1976). Due to its significance in both academic and engineering applications, for the permutation flow shop with the criterion of minimizing the makespan or maximum lateness of jobs, different kinds of approaches have been proposed to solve PFSSP and obtained some achievements.

Since the pioneering work of Johnson (Johnson, 1954) on the two machine permutation flow shop problem, many methods have been introduced for solving PFSSP with the objective of minimizing the makespan and minimizing maximum lateness. However, due to unacceptable computation time, exact algorithms such as branch and bound method (Bansal, 1977; Croce et al., 1996; 2002; Ignall and Schrage, 1965) and mixed integer linear programming method (Stafford, 1988) can only solve problems with instances of relatively small size.

Heuristic algorithms were then proposed to solve the large-sized scheduling problems. This kind of algorithms can be broadly classified into three categories: Constructive heuristic algorithms, improvement heuristic algorithms and hybrid heuristic algorithms (Liu and Wang, 2007). The constructive heuristics are mainly simple heuristics that build a feasible scheduling from scratch, as is seen in Palmer (1985), Gupta (1972), Ho and Chang (1991), Campbell et al. (1970), Rajendran (1993), Nawaz et al. (1983), Taillard (1990) and Framinan and

*Corresponding author. E-mail: lixt314@gmail.com. Tel: +86-0431-84536338. Fax: +86-0431-84536338.

Leisten (2003). Constructive heuristics usually can obtain a nearly optimal solution in a reasonable computational time, while the solution qualities are not satisfactory.

These heuristics are mainly meta-heuristics that start from previous generated solutions and subsequently approach the optimal solution by improving the solutions with domain dependent knowledge. The meta-heuristics mainly include simulated annealing algorithm (SA) (Ogbu and Smith, 1990; Osman and Potts, 1989), genetic algorithm (GA) (Reeves, 1995; Reeves and Yamada, 1998), artificial immune system algorithm (AIS) (Orhan and Alper, 2004), particle swarm optimization algorithm (PSO) (Tasgetiren et al., 2007), ant colony algorithm (ACO) (Rajendran and Ziegler, 2004; Stützle, 1998), tabu search algorithm (Grabowski and Wodecki, 2004; Nowicki and Smutnicki, 1996; Revees, 1993), iterated local search algorithm(ILS) (Stützle, 1998), estimation of distribution algorithm (EDA) (Bassem and Eddaly, 2009) and biogeography based optimization (Simon, 2008). Improvement heuristics usually can obtain fairly satisfactory solutions, while the solution processes are always time-consuming.

Rather recently, it has become evident that the concentration on a sole meta-heuristic has some limitations. Researchers have found that a skilled combination of two or more meta-heuristic techniques, as called hybrid heuristics, can improve the performance especially when dealing with real-world and large scale problems. A lot of hybrid heuristic based algorithms have been investigated in the past few years. In the paper (Nearchou, 2004), a hybrid SA algorithm was introduced combining the operators of GA with local searches. In the paper (Tseng and Lin, 2009), the genetic algorithm is integrated into a novel local search scheme resulting into two hybrid algorithms: The insertion search and the insertion search with cut-and repair (ISCR).

In the paper (Zhang et al., 2009), two effective heuristics are used during the local search to improve all generated chromosomes in every generation. In Wang and Zhang (2003), Wang Ling used the well-known Nawaz-Enscore-Ham (NEH) combined with GA to generate the initial population, and applied multi-crossover operators to enhance the exploring potential. T. Murata proposed a hybrid genetic algorithm with local search (Murata et al., 1996). In the paper (Dipak et al., 2007), a probabilistic hybrid heuristic that combined NEH with SA was proposed for solving PFSSP. In the paper Tasgetiren et al. (2007), applied the PSO algorithm to solve PFSSP by using a small position value rule, and the proposed algorithm, as called PSOVNS, was combined with the variable neighborhood-based local search algorithm. In the paper, Liu and Wang (2007), an efficient particle swarm optimization based mimetic algorithm (MA) for PFSSP to minimize the maximum completion time was proposed. In this algorithm, the PSO-based searching operators and some special local search operators were used to balance the exploration and exploitation abilities. In the paper (Stützle, 1998), an ant-colony based algorithm was proposed to solve PFSSP by combining the fast local search to enhance the solutions. In the paper (Dong and Huang, 2009), a hybrid algorithm that combined Framinan-Leisten (FL) heuristic with iterated local search algorithm is proposed.

Regarding minimizing maximum lateness of permutation flow shop scheduling, within our knowledge, only few researchers have used the minimizing maximum lateness as the performance measures of proposed algorithm. In the paper (Tasgetiren et al., 2004), this paper uses the PSO_{VNS} to find the optimal solution. This algorithm can find 156 out of 160 upper bounds where 155 of them were improved. In the paper, Zheng and Yamashiro (2010), proposed a new quantum differential evolutionary algorithm. This algorithm based on the basic quantum- inspired evolutionary algorithm (QEA) is encoded and decoded by using the quantum rotating angle and a simple strategy. This algorithm can find 157 out of 160 upper bounds where 156 of them were improved.

Among the existing meta-heuristic algorithms, an evolution technique, biogeography based optimization (Simon, 2008) is a new global optimization algorithm that is simple to be implement and has little or no parameters to be tuned based on biogeography theory, which is the study of the geographical distribution of the biological organisms. The BBO has a way of sharing information which is like other biology based algorithms. One of the remarkable advantages of BBO is that this algorithm can use migration, mutation operators to increase the population diversity. Compared with the 14 benchmarks, BBO is the fifth faster of the eight algorithm including ACO, BBO, DE, ES, GA, PBIL, PSO, and SGA. Up to now, most published works on BBO mainly focused on solving the complex continuous optimization problem (Ma and Simon, 2010; Ma, 2010; Gong et al., 2010; Bhattacharya and Chattopadhyay, 2010; Bhattacharya and Chattopadhyay, 2010; Roy et al., 2010). Within our knowledge, only few researchers have used the BBO algorithm to solve PFSSP. Therefore, this field of study is still in its early days, a large number of future researches are necessary in order to develop BBO based algorithms for solving PFSSP other than only for those areas the inventors originally focused on.

In this paper, we propose a new hybrid BBO (HBBO) algorithm combing BBO with some local search mechanisms as well as fast local search for solving PFSSP. The crucial idea behind HBBO can be summarized as follows. Firstly, to make HBBO suitable to solve PFSSP, a new LRV rule is proposed based on random key. This rule can help to convert the continuous encoding of BBO to a job permutation. Secondly, The NEH heuristic was combined with the random initialization to initialize the population with certain quality and diversity. Thirdly, multiple different neighborhoods are designed and incorporated as a local search scheme into the searching process to enrich the searching behaviors and to avoid premature convergence. Fast local search is proposed as a component of BBO, for the sake of improving BBO's local search performance. Fourthly, the pair wise based local search is used to enhance the global optimal solution and help the algorithm to escape from local minimum.

PERMUTATION FLOW SHOP SCHEDULING PROBLEM

The permutation flow shop scheduling problem (PFSSP) in the paper consists of a set of jobs on a set of machines with the objective of minimizing the makespan. In permutation flow shop problem, n jobs $\mathrm{N} = J_1, J_2, ..., J_n$ are to be processed on a series of m machines $\mathrm{M} = M_1, M_2, ..., M_n$ sequentially. Each job consists of a set of operations $J_i = \{O_{i1}, ..., O_{jm}\}$. The processing time of job J_i on machines M_i is denoted by $P_{i,j}$ *(i=1,….,m, j=1,…,n)*. Each job can be processed on only one machine at a time and each machine can be processed on only one job at a time. Moreover, the operation cannot be pre-empted. The sequence in which the jobs to be processed are identical for each machine. The objective of the scheduling is to find way to minimize makespan.

The permutation flow shop scheduling problem is often denoted by the symbols $n/m/P/C_{\max}$. A job permutation is denoted by $\pi = \{\pi_1, \pi_2, ..., \pi_n\}$, where n jobs will be sequenced through m machines. Let $C(\pi_j, m)$ denote the completion time of job π_j on machine *m*. The completion time of the permutation flow shop scheduling problem according to the processing sequence $\pi = \{\pi_1, \pi_2, ..., \pi_n\}$ is shown as follows:

$$C(\pi,1) = p_{\pi_1,1},$$

$$\mathrm{C}(\pi_j,1) = \mathrm{C}(\pi_{j-1},1) + p_{\pi,1}, \mathrm{j} = 2,...,n$$

$$\mathrm{C}(\pi_1, i) = \mathrm{C}(\pi_1, i-1) + p_{\pi,i}, i = 2,...,m$$

$$C(\pi_j, i) = \max(C(\pi_{j-1}, i), C(\pi_j, i-1)) + p_{\pi,i}, j = 2,...n, i = 2,...m \quad (1)$$

Then makespan can be defined as:

$$C_{\max}(\pi) = C(\pi_n, m) \quad (2)$$

The goal of the permutation flow shop problem is to find the most suitable arrangement of π *:

$$C_{\max}(\pi^*) \le C(\pi_n, m) \ \forall \pi \in \prod . \quad (3)$$

As for the flow shop scheduling with the due date constraint, let $L(\pi_j)$ be denoted as the lateness of jobs π_j and be defined as,

$$L(\pi_j) = C(\pi_j, m) - d(\pi_j) . \quad (4)$$

Maximum lateness $L_{\max}(\pi)$ of a permutation can be defined as

$$L_{\max}(\pi) = \max(C(\pi_j, m) - d(\pi_j)) , \quad (5)$$

where $d(\pi_j)$ is the due date of jobs π_j . The optimal solution $\pi*$ should satisfy the following criterion:

$$L_{\max}(\pi^*) \le L_{\max}(\pi) \qquad \forall \pi \in \prod \quad (6)$$

and the sum of flow times of all jobs can be describes as

$$C_{sum}(\pi^*) = \sum_{j=1}^{n} C(\pi_j, m) , \quad (7)$$

The optimal solution $\pi*$ should satisfy the following criterion:

$$C_{sum}(\pi^*) \le C_{sum}(\pi) \ \forall \pi \in \prod \quad (8)$$

BIOGEOGRAPHY-BASED OPTIMIZATION

Biogeography based optimization (Simon, 2008) is a new evolution algorithm developed for the global optimization. It is inspired by the immigration and emigration of species between islands in search of more friendly habitats. Each solution is called an "habitat" with an habitat suitability index (HSI) and represented by an n-dimension real vector. An initial individual of the habitat vectors is randomly generated. Those solutions that are good are considered to be habitats with a high HSI. Those that are poor are considered to be habitats with a low HSI. The high HSI tends to share their features with low HSI. Low HSI solutions accept a lot of new features from high HSI solutions. In BBO, habitat H is a vector of N (SIVs) initialized randomly and then follows migration and mutation step to reach the optimal solution. The new candidate habitat is generated from all of the solution in population by using the migration and mutation operators.

In BBO, the migration strategy is similar to the

```
procedure Habitat migration
Begin
    for i=1 to NP
        Select X_i with probability based on λ_i
        if rand(0,1)< λ_i then
            for j=1 to NP
            Select X_j with probability based on μ_j
            if rand(0,1)< μ_j then
                Randomly select an SIV σ from X_j
                Replace a random SIV in X_i with σ
            end if
            end for
        end if
    end for
End.
```

Scheme A. The algorithm of Habitat migration model.

```
procedure Mutation
Begin
    for i=1 to NP
        Compute the probability P_i
        Select SIV X_i(j)with probability based on P_i
        if rand(0,1)< m_i then
            Replace X_i(j) with a randomly generated SIV
        end if
    end for
End.
```

Scheme B. The algorithm of mutation model.

evolutionary strategy in which many parents can contribute to a single offspring. BBO migration is used to change existing solution and modify existing island. Migration is a probabilistic operator that adjusts habitat X_i. The probability X_i modified is proportional to its immigration rate λ_i, and the source of the modified probability from X_j is proportional to the emigration rate μ_j.Migration can be described as shown in Scheme A.

Mutation is a probabilistic operator that randomly modifies habitat SIVs based on the habitat priori probability of existence. Very high HSI solutions and very low HSI solutions are equally improbable. Medium HSI solutions are relatively probable. The mutation rate m is expressed as:

$$m = m_{\max} \left(\frac{1 - P_s}{P_{\max}}\right) \quad (9)$$

where $m_{\max}$ is a user-defined parameter.

This mutation scheme tends to increase diversity among the population. Mutation can be described as in Scheme B. The basic structure of BBO algorithm can be informally described with the algorithm in Scheme C.

BBO for PFSSP

Solution representation

In BBO, the standard continuous encoding scheme of BBO can not be used to solve PFSSP directly. In order to apply BBO to PFSSP, one of the key issues is to construct a direct relationship between the job sequences and the vector of individuals in BBO. Qian and wang (2008) proposed a new a largest-order-value (LOV) rule based on random key representation to convert the individual to the job permutation. However, in this paper, we propose a random key based largest-ranked-value (LRV) representation. The LRV rule has the same effect as the LOV rule. Using the LRV rule, we can convert the

procedure *Biogeography based optimization*

Begin

Initialize the population P randomly and each habitat corresponding to a potential solution to the given problem.

Evaluate the fitness for each individual in P

G=1

While *the termination criteria is not satisfied* ***do***

Sort the population from best to worst.

For each habitat, map the HSI to the number of species S, the immigration rate λ *, and the emigration rate* μ *.*

Probabilistically use Immigration Island based on the immigration rates.

Modify the population with the migration operator shown in Habitat migration.

Update the probability for each individual.

Mutate the population with the mutation operation.

Evaluate the fitness for each individual in P.

Sort the population from best to worst.

G=G+1;

end while

End.

Scheme C. The algorithm of Biogeography based optimization.

Table 1. Solution representation of particle X_i^t .

Job, dimension	1	2	3	4	5	6
Position x_{ij}^t	1.25	0.85	0.63	1.45	0.23	1.32
Job, π_{ij}^t	3	4	5	1	6	2

continuous real-code vector in BBO to a discrete job permutation. Specifically, in our LRV rule, the largest value of a vector is firstly picked as the first order of a job permutation. After that, the second largest value is picked as the second one. In this way, all the values of the vector will be handled to convert the vector to a job permutation. We use a simple example to illustrate the LRV rule in Table 1. In this example, because the largest value is 1.45, the dimension j=4 is picked and assigned a rank value of one; then the dimension j=6 is picked because the second largest value is 1.32; in the similar way, the job permutation can be obtained, that is, $\pi=[3,4,5,1,6,2]$. As we can see, such a conversion process is really simple, and it makes BBO applicable for solving PFSSP.

Initial population

This section will cover the important issues of initialization and follow the subsequent section by the essential concept of implementation. The BBO-based search is applied for exploration. Initial swarm is often generated randomly. In order to enhance the solution, in our paper, we take advantage of the NEH heuristic to produce 10% vector and the rest of the vectors are initialized with random vector values. Nawaz et al.'s (1983) NEH heuristic is regarded as the best heuristic for the PFSP. The NEH algorithm is based on the idea that the high processing time on all machines should be scheduled as early in the sequence as possible. The NEH heuristic has two phases: (1) The jobs are sorted in non increasing sums of their processing time, and (2) A job sequence is established by evaluating the partial schedules based on the initial order of the first phase. The NEH can be described by the following three steps:

Step 1: Compute the total processing time for each job on m machine:

$$\forall \text{ job } i,\ i=1,\cdots,n,\ P_i=\sum_{j=1}^{m} p_{ij} \tag{10}$$

Step 2: Sort the jobs in non increasing order of P_i, then the first two jobs are taken and the two partial possible schedules are evaluated. Choose the better sequence as a current sequence.

Step 3: Take job *i*, $i=3,\cdots,n$, and find the best schedule by placing it in all possible *i* positions in the sequence of jobs that are already scheduled. The best sequence would be selected for the next iteration.

The NEH algorithm generated the job permutation, which should be converted to the position values of a certain

---

procedure *Biogeography based optimization for permutation flow shop*

Begin

Using NEH heuristic to produce 10% vectors and the rest of the vectors are initialized with random vector values and each habitat corresponding to a potential solution to the given problem.

Apply the LRV rule to convert the individual X_i^0 *to the job permutation* $\pi_i^0 = [\pi_{i1}^0, \pi_{i2}^0, \cdots, \pi_{in}^0]$ *(i=1, ⋯, NP). Evaluate fitness for every individual and determine the best and suboptimial individual with the objective value;*

G = 1

While *the termination criteria is not satisfied* ***do***

Sort the population from best to worst.

For each habitat, map the HSI to the number of species S, the immigration rate λ *, and the emigration rate* μ *.*

Begin *Apply emigration operation*

Probabilistically use Immigration Island based on the immigration rates.

Modify the population with the migration operator shown in Habitat migration.

end *emigration operation*

Update the probability for each individual.

Begin *mutation operation*

Mutate the population with the mutation operation.

end *mutation operation*

Apply the LRV rule to convert the individual X_i^G *to the job permutation* $\pi_i^G = [\pi_{i1}^G, \pi_{i2}^G, \pi_{i3}^G, \cdots, \pi_{in}^G]$ *(i= 1, ⋯, NP). Evaluate fitness for every individual and determine the best and suboptimial individual with the objective value;*

Sort the population from best to worst.

G = G + 1;

end while

End.

Scheme D. Biogeography based optimization for permutation flow shop scheduling.

individual so that it can adapt to the BBO search. The conversion is implemented using the following equation:

$$x_{NEH,i} = x_{\max i} - \frac{(x_{\max i} - x_{\min i})}{n} \cdot (sq_{NEH,i} - 1), i = 1, 2, \cdots, n \qquad (11)$$

where $x_{NEH,i}$ is the position value in the *i*th dimension of the individual. $sq_{NEH,i}$ is the job index in the ith dimension of the permutation.

The rest of the populations are initialized with random vector values. The initial continuous position values of other vector are first calculated by the following formula:

$$x_{ij}^I = (x_{\max} - x_{\min}) * r + x_{\min} \qquad (12)$$

where $x_{\min} = 0$, $x_{\max} = 4.0$ and r is a uniform random number between 0 and 1.

BBO-BASED SEARCH

The biogeography based optimization is a population based stochastic optimization algorithm proposed by Simon in 2008, and the algorithm is similar to the genetic algorithm. The BBO algorithm adopts the real number encoding scheme, migration, mutation based on differential individuals and it has the better ability of overall search ability. Recently, the BBO algorithm was developed to optimize multi-variable and multi-modal continuous functions. Since then, there had been no published work to deal with the permutation flow shop problem by using the biogeography based optimization. However, the BBO algorithm is performed on a continuous space. Therefore, the searching space of the BBO is not the permutation based solution space and we use the LRV rule to convert the continuous individual to the permutation vector. For the initial population, The NEH heuristic combined the random initialization to initialize the population with certain quality and diversity.

The procedure of the biogeography based optimization for PFSSP can be summarized as shown in Scheme d.

Fast local search

Due to the parallel evolution framework of BBO, local search is easy to be incorporated for exploitation. Here, we present a fast local search which is embedded in BBO for solving PFSSP. The purpose of the local search is to find a better solution from the neighborhood of a solution. In this paper, three neighborhoods, that is, 'Inverse, Insert and Swap' are used to improve the diversity of population and enhance the quality of the solution. The three

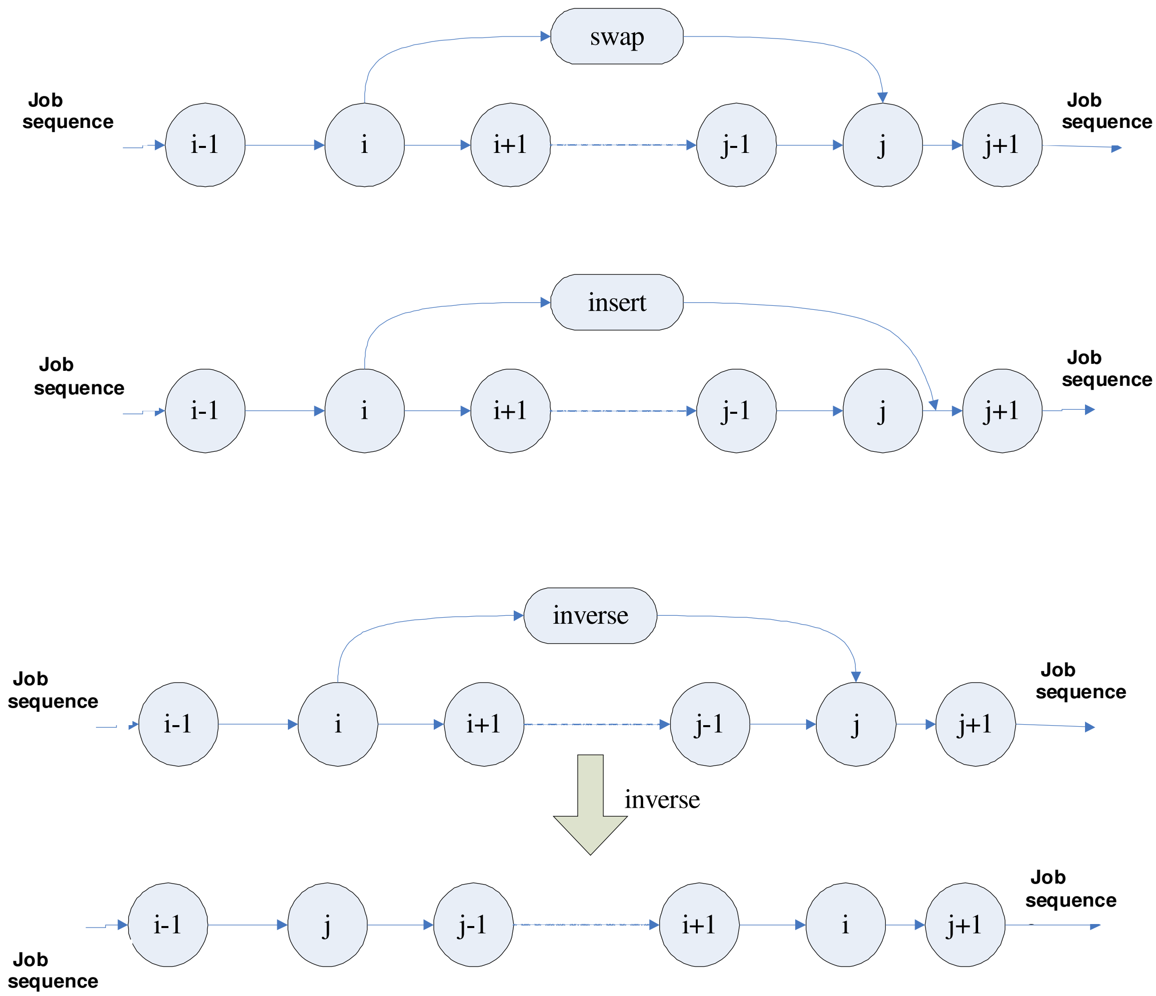

Figure 1. The neighborhood operations for local search.

neighborhood operations are shown in Figure 1. The detail of these neighborhoods is as follows:

Swap: Choose two different positions from a job permutation randomly and swap them. This neighborhood operation is illustrated in Figure 1.
Insert: Choose two different positions from a job permutation randomly and insert the back one before the front. This neighborhood operation is illustrated in Figure 1.
Inverse: Inverse the subsequence between two different random positions of a job permutation. This neighborhood operation is illustrated in Figure 1.

In order to enhance the local search ability and get a better solution, we propose a new fast local search to enhance the makespan of every vector with the certain probability. The algorithm is performed by using the above three operations alternatively to avoid trapping in local optimal points, sometimes.

A new individual enhancement scheme is proposed to combine the insert, swap and inverse operations. This method first selects an operation scheme from the three operations to operate an individual. The selected operation starts from an initial solution and attempts to move from the current solution x to its neighborhood x'. If the objective fitness of x' is smaller than the fitness of the current solution, x' is accepted as a new basic solution. After finishing one scheme, the search process keeps generating the individual's neighborhood randomly and the solution is accepted until the stopping criterion is reached (Scheme g).

Remarks

A partial algorithm about 3 type operations is listed in this algorithm. pr_s is the probability of executing swapping operation, pr_i is the probability of executing inserting operation, pr_{inv} is the probability of executing inversing operation.

Pair wise-based local search

Here, a pair wise-based local search (Liu and Wang, 2007) is employed for the global optimal solution. The crucial idea of pair wise-based local search is to swap the job orders of two adjacent jobs of an individual, and to compare the previous and new

procedure *Pairwise-based local search*

Begin

X_{best} *permutation of global best solution at iteration t.*

Apply the LRV rule to convert the individual X_{best} *to the job permutation* $\pi_{best} = [\pi_{best1}, \pi_{best2}, \cdots, \pi_{bestn}]$. *Evaluate fitness* $f(\pi_{best})$ *for* X_{best} *individual.*

$X_0 = X_{best}$

$\pi_0 = \pi_{best}$

$X_0 = insert(X_0)$

$\pi_0 = insert(\pi_0)$

i=0;

for *i=1 to n*

i=i+1;

j=0;

for *j=1 to n*

j=j+1;

Execute swapping the pair of jobs on position$_i$ *and position*$_j$ *in the individual* X_0, *and obtain the individual* X_{new}.

Apply the LRV rule to convert the individual X_{new} *to the job permutation* $\pi_{new} = [\pi_{new1}, \pi_{new2}, \cdots, \pi_{newn}]$. *Evaluate fitness* $f(\pi_{new})$ *for* X_{new} *individual.*

If $(f(\pi_{new}) - f(\pi_{best}) < 0)$

$X_{best} = X_{new}$;

$f(\pi_{best}) = f(\pi_{new})$;

$X_0 = X_{new}$;

$\pi_{best} = \pi_{new}$;

End if

End for *(j=the number of jobs)*

End for *(i=the number of jobs)*

End.

Scheme F. The procedure of the pair wise-based local search algorithm.

Table 2. The execution time for every operation of every job on every machine.

	Job 1	Job 2	Job 3	Job 4	Job 5
M1	3	4	7	8	10
M2	12	15	6	10	8

makespan. If the new makespan is lower than the previous, we will accept the new individual. This method can be also regarded as a local search for BBO to enhance the global optimal solution in every generation. It examines each possible pair wise interchange of the job in the first position. Then other position is given the same operation. Whenever there is an improvement in the objective function, the orders of the jobs are interchanged. Pair wise-based search can be viewed as a detailed neighborhood searching process to enhance the exploitation ability. We will use an example to describe the pair wise-based local search. Table 2 shows the execution time for every operation of every job on every machine. It is assumed that we want to improve the individual $\pi = (5,4,2,1,3)$, whose makespan is 42 and the corresponding Gantt chart is shown in Figure 2. By using the proposed pair wise-based local search, the next individual is $\pi = (4,5,2,1,3)$, whose makespan is 41 and the corresponding Gantt chart is shown in Figure 3. The new individual is lower than the pervious individual. Thus, we will accept the new individual. Table 3 shows the makespans of the example in Table 2.

The procedure of the pair wise-based local search algorithm for PFSSP is as shown in Scheme f.

BBO-based hybrid algorithm

Here, structure of this hybrid algorithm was discussed. There are mainly four strategies to update the individuals in the proposed algorithm. The LRV rule is used to convert the continuous position in BBO to the discrete job permutation. The NEH heuristic combined the random initialization to initialize the population with certain quality and diversity. The fast–based local search is used for enhancing the individuals with a certain probability, and therefore has a higher ability to approximate the optimal solution fast. The pair wise based local search is used to enhance the global optimal solution and help the algorithm to escape from local minimum.

Based on the above section, solution representation, initial population, BBO-based local search, fast local search, pair wise-based local search, the procedure of HBBO is proposed as follows:

Step1: Initialize the parameters of BBO algorithm, containing the maximum migration rates E and I, the maximum rate $m_{\max}$,and the minimal emigration rate θ. Set G which denotes a generation, G=0. $G_{\max}$ is the evolution generation. NP is the population size. D is the dimension or the number of the jobs. Keep = 2.

Step 2: Initialize the population, using NEH heuristic to produce 10% vectors and the rest of the vectors are initialized with random

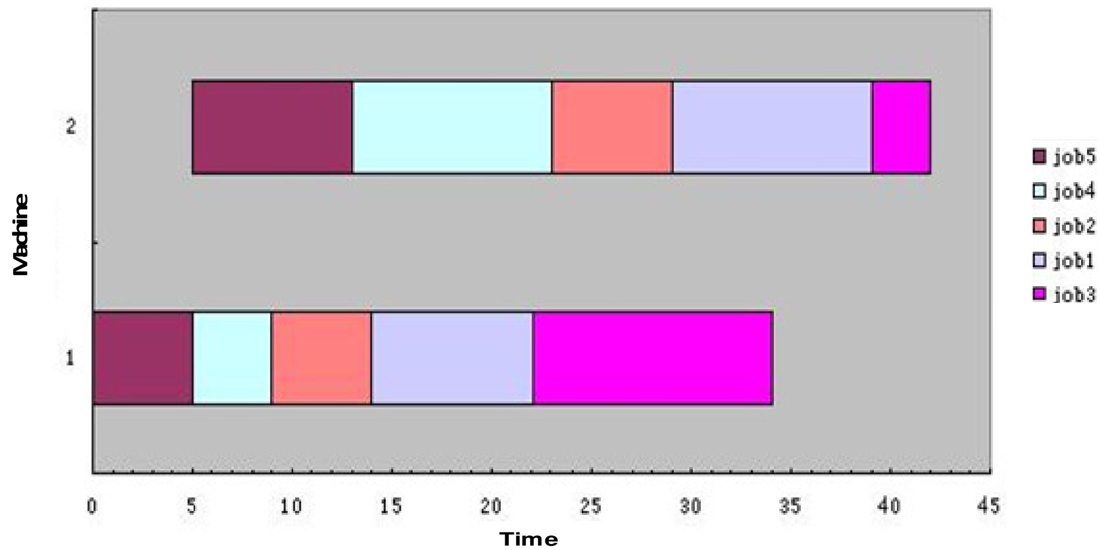

Figure 2. The Gantt chart of the original scheduling.

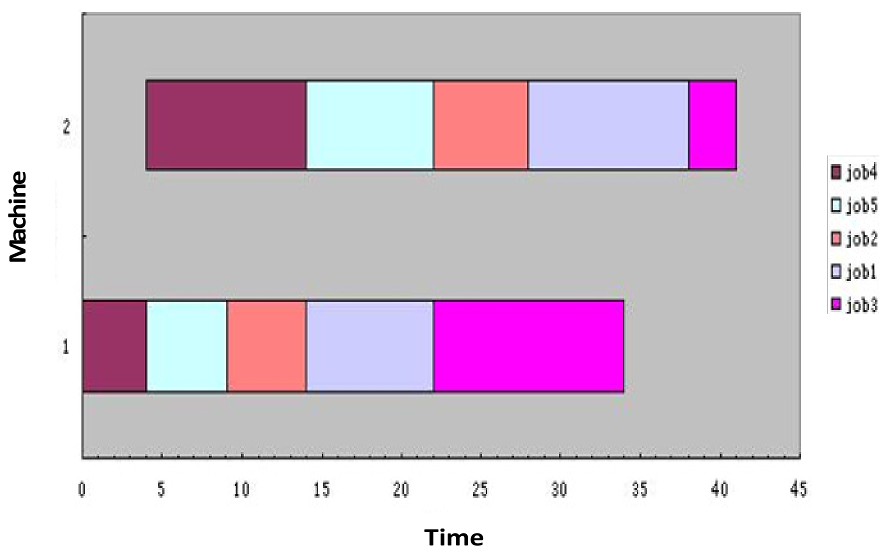

Figure 3. The Gantt chart of the active scheduling.

Table 3. Makespan of the example shown in Table 2.

	Job 1	Job 2	Job 3	Job 4	Job 5
M1	4	9	14	22	34
M2	14	22	28	38	41

vector values and each habitat corresponding to a potential solution to the given problem. Let $x_{\min} = 0$, $x_{\max} = 4.0$ Generate $x_{i,j}(0) = x_{\min} + random(0,1) * (x_{\max} - x_{\min})$, $i = 1, \cdots, NP$, $j = 1, \cdots, D$.

Step 3: Evaluate the population. Apply the LRV rule to convert the individual X_i^0 to the job permutation $\pi_i^0 = [\pi_{i1}^0, \pi_{i2}^0, \pi_{i3}^0, \cdots, \pi_{in}^0]$ (i=1, $\cdots$, NP). Evaluate fitness for every individual and sort the fitness from best to worst. Then determine the best and suboptimal individual with the objective value.

Step 4: Perform the evolution.

Step 4.1 (Compute phase): Compute immigration rate λ and emigration rate μ for each species count. $\lambda(i)$ is the immigration rate for habitat i, $\mu(i)$ is the emigration rate for habitat i.

Step 4.2: (Migration phase): The probability X_i modified is

```
procedure fast local search
Begin
    X_p the individual to be enhanced
    Apply the LRV rule to convert the individual X_p to the job
    permutation π_p = [π_p1, π_p2, ··· , π_pn]. Evaluate fitness f(π_p) for X_p individual.
  for i=1 to n × (n − 1)
   q = rand();
   Select r1 and r2 randomly, where r1 ≠ r2
   If (0 ≤ q ≤ pr_s)
   Execute swapping scheme for the individual X_p ,and obtain the individual X'_p
   Else If (pr_s ≤ q ≤ pr_s + pr_i)
   Execute inserting scheme for the individual X_p , and obtain the individual X'_p
   Else (pr_s + pr_i ≤ q ≤ pr_s + pr_i + pr_inv)
   Execute inversing scheme for the individual X_p , and obtain the individual X'_p
   End if
   Apply the LRV rule to convert the individual X'_p to the job
   permutation π'_p = [π'_p1, π'_p2, ··· , π'_pn]. Evaluate fitness f(π'_p) for X'_p individual.
   If ( f(π'_p) − f(π_p) <= 0)
      X_p = X'_p ;
      f(π_p) = f(π'_p) ;
      π_p = π'_p ;
  End if
  End for
End.
```

Scheme G. The procedure of the fast local search algorithm.

proportional to its immigration rate λ_i, and the source of the modified probability from X_j is proportional to the emigration rate μ_j.

Step 4.3 (Mutation phase): For each habitat, update the probability of its species count. Then mutate each habitat using the equation $m = m_{\max}(\frac{1-P_s}{P_{\max}})$ and recompute each habitat's fitness.

Step 4.3 (Compute phase): Make sure that there are no duplicate individuals in the population. Any duplicates that are found are randomly mutated, so there should be a good chance that there are no duplicates in the population.

Step 4.4 (Evaluating phase): Evaluate the population. Apply the LRV rule to convert the individual X_i^G to the job permutation $\pi_i^G = [\pi_{i1}^G, \pi_{i2}^G, \pi_{i3}^G, \cdots, \pi_{in}^G]$ (i=1, $\cdots$, NP). Evaluate fitness for every individual and sort the fitness from best to worst. Then determine the best and suboptimal individual with the objective value.

Step 4.5 (Local search phase): Use the fast local search to enhance the best individual, and the pair wise-based local search operations on the suboptimal individual. Through these operations, we save the best one for the next generation.

Step 5 (Stopping criteria): Set G=G+1.if G< $G_{\max}$ then go to Step 4.

NUMERICAL SIMULATION RESULTS AND COMPARISONS

To test the performance of the proposed HBBO for the permutation flow shop scheduling problem, computational simulations are carried out with some well-studied problems taken from the OR-Library. In this paper, 29 problems from two classes of PFFSP test problems are selected. The first eight problems are instances car1, car2 through to car8 designed by Cariler (1978). The second 21 problems are instances rec01, rec03 through to rec41 designed by Reeves and Yamada (1998). The third 120 instances are from Taillard (1993) and the last problems sets are called DMU problems from Demirkol et al. (1998), containing 160 problems for our experiments. So far, these problems have been widely used as benchmarks to certify the performance of algorithms by many researchers.

The HBBO is coded in MATLAB 7.0, and in our simulation, numerical experiments are performed on a PC with Pentium 3.0 GHz Processor and 1.0 GB memory. In HBBO, each instance is independently executed 15 times for every algorithm for comparison. The parameters are set as follows: The population size=30, $G_{\max}$ =300, mutation probability P=0.01.

Comparisons of BBO, HBBO

The statistical performances of BBO and HBBO are shown in Table 4. In this table, *C** is the optimal makespan or lower bound value known so far. BRE

Table 4. Comparisons of BBO and HBBO.

Problem	n/m	C^*	BBO				HBBO			
			BRE	ARE	WRE	t_{avg}	BRE	ARE	WRE	t_{avg}
Car1	11/5	7038	0	0	0	0	0	0	0	0
Car2	13/4	7166	0	0	0	0.7813	0	0	0	0.06
Car3	12/5	7312	0	1.0708	1.1898	0.8125	0	0	0	0.26
Car4	14/4	8003	0	0	0	0.0313	0	0	0	0
Car5	10/6	7720	0	0.9547	1.3083	0.7656	0	0	0	0.03
Car6	8/9	8505	0.7643	0.9500	2.6220	0.5938	0	0	0	0.06
Car7	7/7	6590	0	0	0	0	0	0	0	0
Car8	8/8	8366	0	0	0	0.1094	0	0	0	0.03
Rec01	20/5	1247	0.1604	0.8099	2.6464	1.1563	0	0.016	0.1604	11.45
Rec03	20/5	1109	0.6312	0.7484	1.1722	1.1719	0	0	0	0.8875
Rec05	20/5	1242	0.2415	0.3704	0.8857	1.1406	0.2415	0.2415	0.2415	11.6875
Rec07	20/10	1566	1.1494	3.3844	3.8314	1.2031	0	0	0	2.2813
Rec09	20/10	1537	2.2121	2.3878	2.4073	1.1875	0	0	0	0.9688
Rec11	20/10	1431	0.2795	2.4319	3.2145	1.2188	0	0	0	0.5781
Rec13	20/15	1930	1.2953	2.3109	2.9534	1.2031	0	0.2383	0.9845	14.5625
Rec15	20/15	1950	1.1795	1.1846	1.2308	1.1875	0	0.1744	0.6667	14.6563
Rec17	20/15	1902	2.2608	3.2282	4.5216	1.2188	0	0.2261	1.1567	14.5313
Rec19	30/10	2093	1.3856	2.6899	3.7745	1.6250	0.2867	0.4013	0.8600	32.4219
Rec21	30/10	2017	1.6361	1.9683	2.8260	1.5781	0.1487	1.2692	1.4378	32.2344
Rec23	30/10	2011	2.1382	3.1328	5.5196	1.7031	0.3987	0.4525	0.4973	32.2031
Rec25	30/15	2513	3.2630	4.3016	5.0537	1.6250	0	0.5412	1.1540	36.3750
Rec27	30/15	2373	1.6435	2.7729	3.5398	1.6563	0.2107	0.4256	0.9692	36.8281
Rec29	30/15	2287	1.6178	3.1351	3.9790	1.6563	0	0.7346	1.9182	36.8750
Rec31	50/10	3045	2.5944	3.0049	3.3498	2.5156	0.2627	0.4499	1.1494	108.8438
Rec33	50/10	3114	0.9313	0.9891	0.9955	2.4375	0	0.4110	0.8349	108.9063
Rec35	50/10	3277	0.2441	0.4333	0.7629	2.5938	0	0	0	3.6719
Rec37	75/20	4951	4.7667	5.3282	5.9584	3.7188	1.3533	1.7592	2.2622	368.1250
Rec39	75/20	5087	3.4598	4.0849	4.4820	3.7344	0.8649	1.2168	1.9265	365.7188
Rec41	75/20	4960	4.9194	5.3750	5.7661	3.7656	1.5121	1.9516	2.7823	370.0625
Average			1.3370	1.9672	2.5514	1.4618	0.1820	0.3624	0.6207	55.3210

represents the best relative error to C^*, ARE denotes the average relative error to C^*, and WRE represents the worst relative error to C^*. t_{avg} Denoted the time to reach the best solution in each run average over R runs in seconds. The performance measures employed in our experiment, BRE, ARE, WRE are defined:

$$BRE = \frac{Heu_{best} - Best_{know}}{Best_{know}} \times 100\ \%$$

$$\text{ARE}=\sum_{r=1}^{n} \left(\frac{Heu_{i} - Best_{know}}{Best_{know}} \times 100\right) \times \frac{1}{n} (\%)$$

$$WRE = \frac{Heu_{worst} - Best_{worst}}{Best_{worst}} \times 100\%$$

From Table 4, it can be seen that both HBBOs provide better performance than BBO for almost all benchmarks. The HBBO can provide better solutions than the BBO algorithm. This demonstrates the effectiveness of the fast local search and pairwise-based local search in HBBO. From Table 4, the t_{avg} of HBBO is more than the BBO. The cause of consuming the time is that the local searches need to consume some time. Additionally, in Table 4, we can also see that, for the problems of the rec01-rec41, the optimal results obtained by the HBBO are closer to C^* than BBO. Therefore, in the final HBBO algorithm, these two local searches will be used in the algorithm. From the results, this demonstrates that the global searching ability of HBBO is effective and HBBO is very suitable for PFSSP. To illustrate the experimental results more intuitively, we will set the car06 as an example.

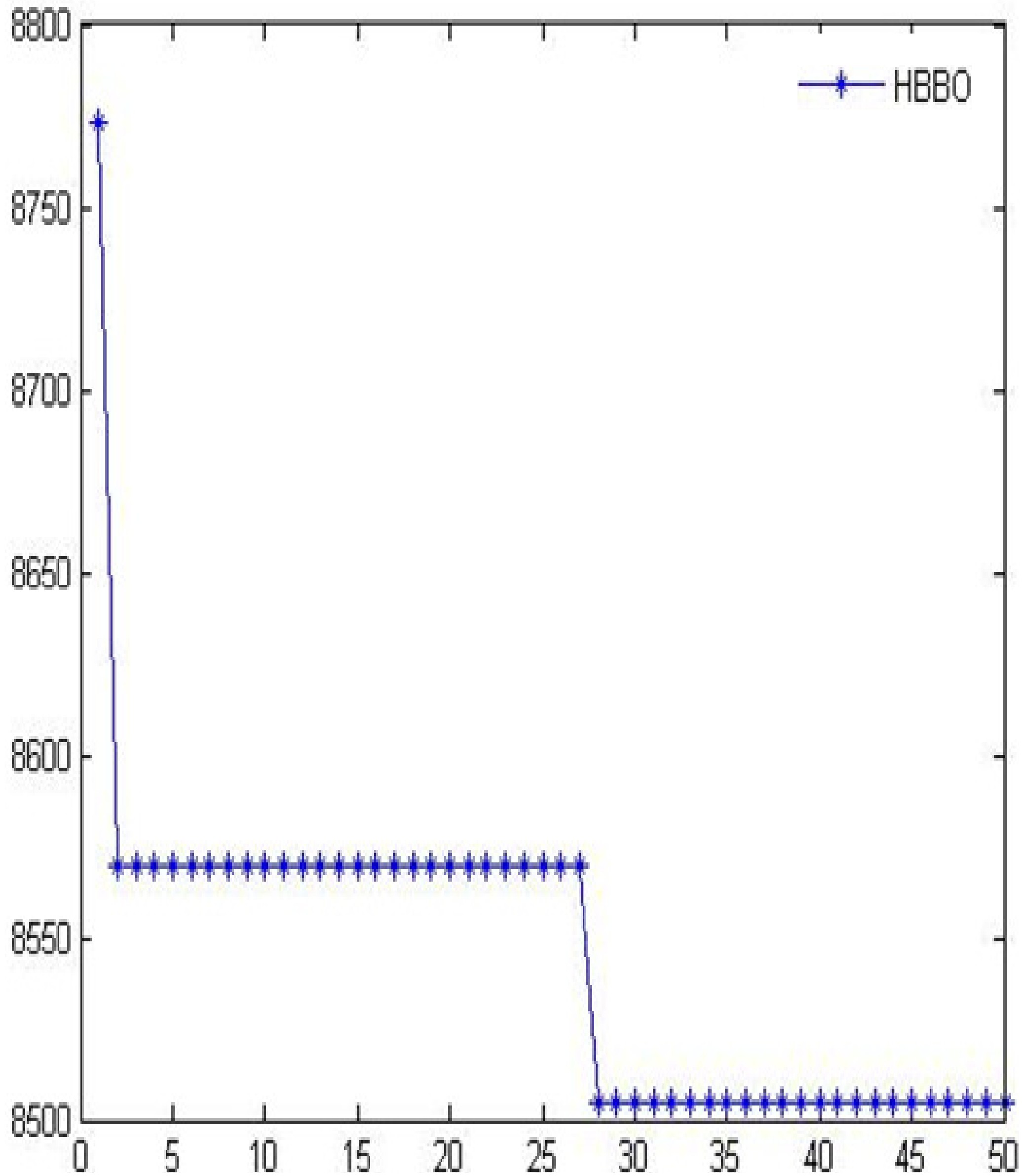

Figure 4. Convergence curves for different algorithms for Car06.

This instance has a lot of local optimal so that it is difficult to find the optimal solution. The convergence rate of HBBO algorithm could be found in Figure 4. It can be seen that at the iteration of 28, HBBO can find the Car's optimal of 8366.

Comparisons of HBBO, HDE (Qian and wang, 2008) and OSA

In order to show the effectiveness of HBBO, we carry out a simulation to compare our HBBO with another DE based algorithm HDE (Qian and Wang, 2008) and OSA (Osman and Potts, 1989). HDE applies the parallel evolution mechanism of DE to perform effective exploration (global search), but it also adopts problem-dependent local search methodology to adequately perform exploitation (local search). OSA's performance is an efficient algorithm which uses insert to construct the neighborhood. The experimental result is listed in Table 5. SD denotes the standard deviation of the makespan. From Table 5, it is shown that the BRE, and ARE values obtained by HBBO are better than those resulting from HDE and OSA for all instances. For the OSA, this algorithm only performs better than the HBBO for instance rec39. For the rest problems, HBBO can provide better

Table 5. Comparisons of HDE, OSA and HBBO.

Problem	HDE			OSA			HBBO		
	BRE	ARE	SD	BRE	ARE	SD	BRE	ARE	SD
Car1	0	0	0	0	0	0	0	0	0
Car2	0	0	0	0	0	0	0	0	0
Car3	0	0.536	44.352	0	0.625	47.188	0	0	0
Car4	0	0	0	0	0	0	0	0	0
Car5	0	0.593	49.537	0	0.801	50.725	0	0	0
Car6	0	0.153	27.406	0	2.093	274.705	0	0	0
Car7	0	0.448	60.219	0	1.483	114.208	0	0	0
Car8	0	0	0	0	2.297	254.627	0	0	0
Rec01	0	0.152	0.447	0.160	0.160	0	0	0.016	0.6325
Rec03	0	0.153	1.252	0	0.189	1.853	0	0	0
Rec05	0.242	0.386	3.327	0.242	0.588	4.620	0.2415	0.2415	0
Rec07	0	0.920	7.589	0	0.434	11.593	0	0	0
Rec09	0	0.273	11.593	0	0.690	12.385	0	0	0
Rec11	0	0	0	0	2.215	37.600	0	0	0
Rec13	0.259	0.705	8.566	0.311	1.793	14.691	0	0.2383	5.6411
Rec15	0.051	0.995	14.961	0.718	1.569	16.071	0	0.1744	5.3375
Rec17	0.368	1.309	11.874	1.840	3.796	36.721	0	0.2261	7.0404
Rec19	0.287	0.908	9.104	0.287	0.803	9.484	0.2867	0.4013	5.0596
Rec21	0.198	1.284	9.171	1.438	1.477	1.687	0.1487	1.2692	8.3293
Rec23	0.497	0.696	10.625	0.497	0.845	10.822	0.3987	0.4525	0.7379
Rec25	0.676	1.429	16.489	1.194	1.938	15.063	0	0.5412	8.1131
Rec27	0.843	1.197	4.600	0.843	1.845	21.055	0.2107	0.4256	6.4541
Rec29	0.525	1.299	13.342	0.612	2.882	38.831	0	0.7346	6.6299
Rec31	0.427	1.192	13.107	0.296	1.333	30.394	0.2627	0.4499	8.0836
Rec33	0.353	0.787	4.743	0.128	0.732	7.315	0	0.4110	12.2638
Rec35	0	0	0	0	0	0	0	0	0
Rec37	1.697	2.632	33.410	2.000	2.751	25.433	1.3533	1.7592	13.6255
Rec39	1.278	1.543	7.836	0.767	1.240	12.306	0.8649	1.2168	18.9529
Rec41	1.714	2.615	36.387	1.734	2.726	39.378	1.5121	1.9516	18.8668
Average	0.325	0.766	13.792	0.451	1.287	37.543	0.1820	0.3624	4.3368

solutions. The SD value of the proposed algorithm is also much better than the HDE and OSA for most instances expect Rec01, Rec21, Rec27, Rec33, and Rec 39. It can be concluded that the performance of HBBO is better than HDE and OSA.

Comparisons of HBBO, PSOMA, PSOVNS

The performance of HBBO is also compared with other two state-of-art algorithms, that is PSOMA proposed by Bo Liu (Liu and Wang, 2007), and PSOVNS proposed by Tasgetiren (Tasgetiren et al., 2004). We accept the results of those papers and do not ourselves program these algorithms. The experimental result is listed in Table 6. As can be seen in Table 6, the BRE and ARE values obtained by HBBO are much better than PSOMA and PSOVNS. All WRE values obtained by HBBO are better than PSOMA and PSOVNS. It can be concluded that HBBO is more effective than PSOMA and PSOVNS in an acceptable time.

Figure 5 shows the means plot with LSD intervals for the above six algorithms, that is, HDE, OSA, PSOVNS, PSOVNS, BBO and HBBO. Form the results we can see that the HBBO produces statistically better results than all others. Therefore, the HBBO is a robust algorithm for solving PFSSP.

Comparisons of SGA, SGA+NEH, HGA (Wang and Zheng, 2003)

In order to further show the effectiveness of HBBO, we carry out some comparisons with SGA, HGA and SGA+NEH in previous papers. We accept the results of those papers and do not ourselves program these

Table 6. Comparisons of PSOMA, PSOVNS and HBBO.

Problem	PSOVNS			PSOMA			HBBO		
	BRE	ARE	WRE	BRE	ARE	WRE	BRE	ARE	WRE
Car1	0	0	0	0	0	0	0	0	0
Car2	0	0	0	0	0	0	0	0	0
Car3	0	0.420	1.189	0	0	0	0	0	0
Car4	0	0	0	0	0	0	0	0	0
Car5	0	0.039	0.389	0	0.018	0.375	0	0	0
Car6	0	0.076	0.764	0	0.114	0.764	0	0	0
Car7	0	0	0	0	0	0	0	0	0
Car8	0	0	0	0	0	0	0	0	0
Rec01	0.160	0.168	0.321	0	0.144	0.160	0	0.016	0.1604
Rec03	0	0.158	0.180	0	0.189	0.721	0	0	0
Rec05	0.242	0.249	0.420	0.242	0.249	0.402	0.2415	0.2415	0.2415
Rec07	0.702	1.095	1.405	0	0.986	1.149	0	0	0
Rec09	0	0.651	1.366	0	0.621	1.691	0	0	0
Rec11	0.071	1.153	2.656	0	0.129	0.978	0	0	0
Rec13	1.036	1.79	2.643	0.259	0.893	1.502	0	0.2383	0.9845
Rec15	0.769	1.487	2.256	0.051	0.628	1.076	0	0.1744	0.6667
Rec17	0.999	2.453	3.365	0	1.330	2.155	0	0.2261	1.1567
Rec19	1.529	2.099	2.532	0.43	1.313	2.102	0.2867	0.4013	0.8600
Rec21	1.487	1.671	2.033	1.437	1.596	1.636	0.1487	1.2692	1.4378
Rec23	1.343	2.106	2.884	0.596	1.310	2.038	0.3987	0.4525	0.4973
Rec25	2.388	3.166	3.780	0.835	2.085	3.233	0	0.5412	1.1540
Rec27	1.728	2.463	3.203	1.348	1.605	2.402	0.2107	0.4256	0.9692
Rec29	1.968	3.109	4.067	1.442	1.888	2.492	0	0.7346	1.9182
Rec31	2.594	3.232	4.237	1.510	2.254	2.692	0.2627	0.4499	1.1494
Rec33	0.835	1.007	1.477	0	0.645	0.834	0	0.4110	0.8349
Rec35	0	0.038	0.092	0	0	0	0	0	0
Rec37	4.383	4.949	5.736	2.101	3.537	4.039	1.3533	1.7592	2.2622
Rec39	2.850	3.371	3.951	1.553	2.426	2.830	0.8649	1.2168	1.9265
Rec41	4.173	4.867	5.585	2.641	3.684	4.052	1.5121	1.9516	2.7823
Average	1.0089	1.4420	1.9493	0.4981	0.9532	1.3560	0.1820	0.3624	0.6207

algorithms. SGA is the standard genetic algorithm. HGA is a hybrid genetic algorithm that uses multi-crossoveroperators to effect on a subpopulation and uses the SA to enhance it. SGA+NEH is a hybrid genetic algorithm using NEH to improve algorithm performance. The experi-mental results are listed in Table 6. As can be seen in Table 7, the BRE and ARE values of HBBO are better than those obtained by SGA, SGA+NEH and SGA for all problem expect Rec05. For the HGA, this algorithm only performs better than the HBBO for instance rec05. For the rest problems, HBBO can provide better solutions. Therefore, it can be concluded that HBBO is more effective than these algorithm in an acceptable time.

Figure 6 shows the means plot with LSD intervals for the above five algorithms, that is, SGA, SGA+NEH, HGA, BBO and HBBO. From the results, we can see that the HBBO produces statistically better results than all others. Therefore, the HBBO is a robust algorithm for solving PFSSP.

Comparisons of HQEA, QDEA and HBBO

The performance of HBBO is also compared with other quantum-inspired algorithms. The hybrid quantum-inspired evolution algorithm (HQEA) proposed by Wang et al. (2005); the quantum differential evolution algorithm (QDEA) was proposed by Zheng and Yamashiro, (2010). We accept the results of those papers and do not ourselves program these algorithms. The experimental results are listed in Table 8. From Table 8, we can find the Car problem, the HQEA, QDEA, HBBO all can find the optimal solution. For the Rec problem, HBBO also can provide better solutions than the other solution

Table 7. Comparisons of SGA, SGA+NEH, HGA and HBBO.

Problem	SGA		SGA+NEH		HGA		HBBO	
	BRE	ARE	BRE	ARE	BRE	ARE	BRE	ARE
Car1	0	0.27	0	0	0	0	0	0
Car2	0	4.07	2.93	2.93	0	0	0	0
Car3	1.09	2.95	0.82	1.21	0	0	0	0
Car4	0	2.36	0	0.07	0	0	0	0
Car5	0	1.46	0	1.14	0	0	0	0
Car6	0	1.86	0	2.82	0	0.04	0	0
Car7	0	1.57	0	1.36	0	0	0	0
Car8	0	2.59	0	0.03	0	0	0	0
Rec01	2.81	6.96	2.25	6.13	0	0.14	0	0.016
Rec03	1.89	4.45	1.26	4.27	0	0.09	0	0
Rec05	1.93	3.82	2.33	2.90	0	0.29	0.2415	0.2415
Rec07	1.15	5.31	3.38	5.27	0	0.69	0	0
Rec09	3.12	4.73	0.39	2.13	0	0.64	0	0
Rec11	3.91	7.39	1.19	3.66	0	1.10	0	0
Rec13	3.68	5.97	1.92	4.41	0.36	1.68	0	0.2383
Rec15	2.21	4.29	2.87	4.02	0.56	1.12	0	0.1744
Rec17	3.15	6.08	2.16	4.02	0.95	2.32	0	0.2261
Rec19	4.01	6.07	2.05	4.35	0.62	1.32	0.2867	0.4013
Rec21	3.42	6.07	3.52	3.58	1.44	1.57	0.1487	1.2692
Rec23	3.83	7.46	3.63	5.12	0.40	0.87	0.3987	0.4525
Rec25	4.42	7.20	3.14	4.89	1.27	2.54	0	0.5412
Rec27	4.93	6.85	3.16	5.12	1.10	1.83	0.2107	0.4256
Rec29	6.21	8.48	3.32	4.93	1.40	2.70	0	0.7346
Rec31	6.17	8.02	5.94	6.66	0.43	1.34	0.2627	0.4499
Rec33	3.08	5.12	2.70	3.38	0	0.78	0	0.4110
Rec35	1.46	3.30	1.89	2.58	0	0	0	0
Rec37	7.89	10.07	7.14	7.94	3.75	4.90	1.3533	1.7592
Rec39	7.32	8.51	6.25	7.09	2.20	2.79	0.8649	1.2168
Rec41	8.51	10.03	7.49	8.47	3.64	4.92	1.5121	1.9516
Average	2.9721	5.2866	2.4734	3.8097	0.6248	1.1610	0.1820	0.3624

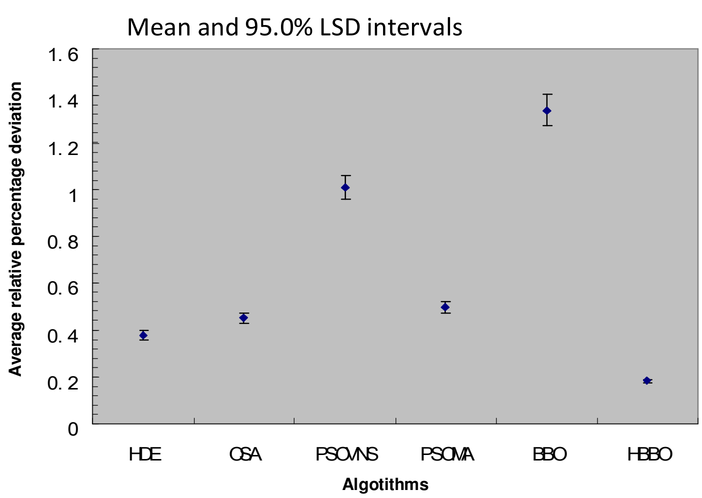

Figure 5. The means plot with LSD intervals for six algorithms (HDE, OSA, PSOVNS, PSOMA, BBO, HBBO).

Table 8. Comparisons of HQEA, QDEA and HBBO.

Problem	HQEA		QDEA		HBBO	
	BRE	ARE	BRE	ARE	BRE	ARE
Car1	0	0	0	0	0	0
Car2	0	0	0	0	0	0
Car3	0	0	0	0	0	0
Car4	0	0	0	0	0	0
Car5	0	0	0	0	0	0
Car6	0	0	0	0	0	0
Car7	0	0	0	0	0	0
Car8	0	0	0	0	0	0
Rec01	0	0.140	0	0	0	0.016
Rec03	0	0.170	0	0	0	0
Rec05	0.240	0.340	0.242	0.242	0.2415	0.2415
Rec07	0	1.020	0	0	0	0
Rec09	0	0.640	0	0	0	0
Rec11	0	0.670	0	0	0	0
Rec13	0.160	1.070	0.104	0.225	0	0.2383
Rec15	0.050	0.970	0	0.158	0	0.1744
Rec17	0.630	1.680	0	0.126	0	0.2261
Rec19	0.290	1.430	0.287	0.435	0.2867	0.4013
Rec21	1.440	1.630	0.149	1.041	0.1487	1.2692
Rec23	0.500	1.200	0.348	0.597	0.3987	0.4525
Rec25	0.770	1.870	0.119	0.454	0	0.5412
Rec27	0.970	1.830	0.253	0.954	0.2107	0.4256
Rec29	0.350	1.970	0	0.824	0	0.7346
Rec31	1.050	2.500	0.263	0.565	0.2627	0.4499
Rec33	0.830	0.910	0	0.297	0	0.4110
Rec35	0	0.150	0	0	0	0
Rec37	2.520	4.330	1.717	2.771	1.3533	1.7592
Rec39	1.630	2.710	0.845	1.485	0.8649	1.2168
Rec41	3.130	4.150	1.190	1.965	1.5121	1.9516
Average	0.502	1.082	0.1902	0.428	0.1820	0.3624

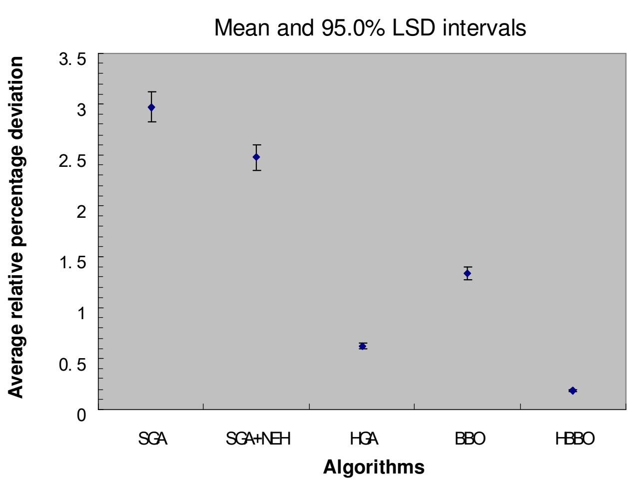

Figure 6. The means plot with LSD intervals for six algorithms (SGA, SGA+NEH, HGA, BBO, HBBO).

expect Rec23 and Rec41. For these two problems, our algorithm cannot obtain the better solution than the QDEA. But the ARE of HBBO is better or equal to the QDEA. Therefore, the HBBO algorithm is an effective and robust approach for the PFSP. Figure 7 shows the means plot with LSD intervals for the above algorithms, that is, HQEA, QDEA and HBBO. From the results, we can see that the HBBO produces statistically better results than the other algorithm. Therefore, the HBBO is an effective algorithm for solving PFSSP.

Comparisons on minimizing maximum lateness of PFSP with QDEA

For the maximum lateness criterion in PFSP, There are two algorithms: PSOVNS (Tasgetiren et al., 2004) and QDEA (Zheng and Yamashiro, 2010). However, the PSOVNS do not report their best solutions in their paper. Therefore, in order to show the effectiveness of HBBO, we carry out a simulation to compare our HBBO algorithms for the minimizing maximum lateness with QDEA. The 160 DMU benchmark problems (Demirkol et al., 1998) (available from http://cobweb.ecn.purdue.edu/~uzsoy2/benchmark/flmax.txt) are used to demonstrate the algorithm. The experimental solutions are listed in Table 9. We found that PSOVNS can obtain 157 out of 160 upper bounds where 156 of them were improved. However, our HBBO algorithm and QDEA can also obtain 157 out of 160 upper bounds where 156 of them were improved. But our algorithm can find 137 new upper bounds for the DWU problems. For the ARE, the HBBO are all better than the QDEA. The HBBO algorithm provided 137 new upper bounds for future research to provide new algorithms and to compare their results with our solutions.

Effects of the parameter P

Here, we will discuss the effects of the parameter of *P*. The value of *P* is very important for the BBO algorithm (Simon, 2008). Each individual has an associated probability, which indicates the likelihood that it was expected a priori to exist as a solution to the given problem. If a given problem S has a low probability Ps, then it is surprising that it exists as a solution. It is likely tomutate other solutions. In contrast, a solution with a high probability is likely to mutate to a different solution. Based on it, we also use Car problem and Rec problem as examples to analyze the value of *P*. $P \in \{0.005, 0.01, 0.02, 0.04, 0.06, 0.08, 0.1\}$. The experimental results are listed in Table 10. As can be seen in Table 4, the solution qualities of HBBO vary with *P*, whose value changes from 0.005 to 0.1.

Figure 8 shows the means plot with LSD intervals for the variant P. From Figure 8, we can clearly see that the

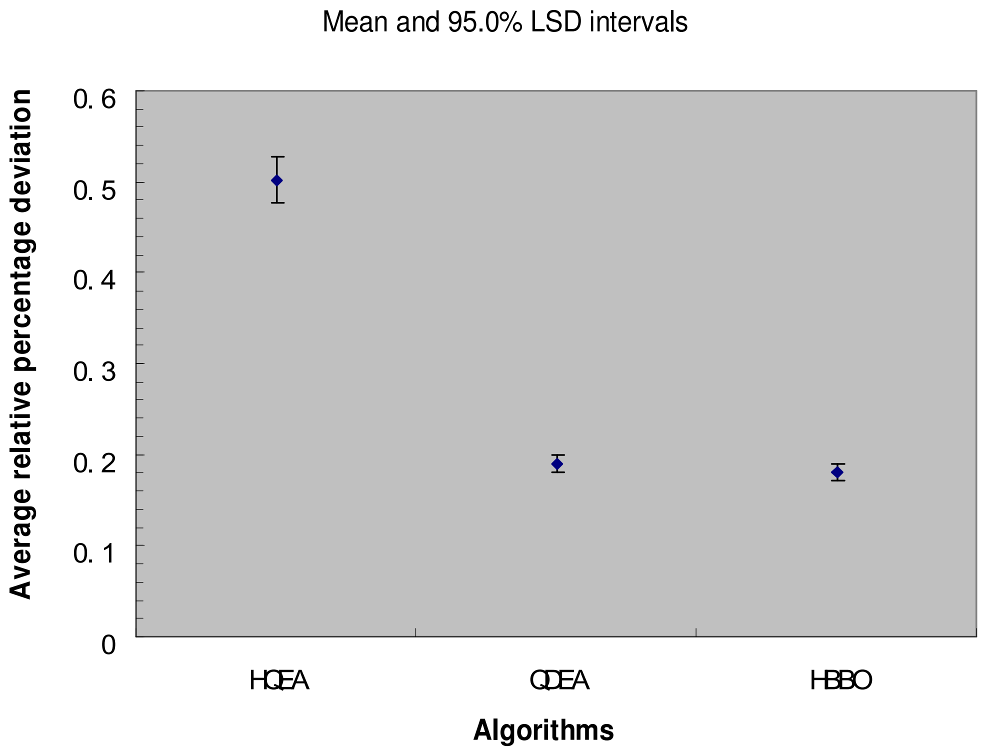

Figure 7. The means plot with LSD intervals for six algorithms.(HQEA, QDEA, HBBO).

Table 9. The maximum lateness obtained by HBBO.

P	UB	QDEA	HBBO	P	UB	QDEA	HBBO	P	UB	QDEA	HBBO
20,15				20,20				30,15			
1	2833	2468	2431	21	3437	3024	2962	41	2837	2291	2228
2	2322	2087	2059	22	3127	2752	2748	42	3088	2629	2568
3	2370	2112	2081	23	2906	2745	2741	43	2733	2346	2270
4	2554	2275	2255	24	3197	2995	2979	44	3054	2689	2563
5	2699	2330	2321	25	3069	2748	2715	45	3074	2636	2595
6	2239	2307	2307	26	2594	2579	2579	46	2158	2026	2007
7	1722	1712	1712	27	3388	3294	3294	47	1875	1748	1740
8	2526	2508	2508	28	2978	2947	2947	48	2637	2591	2575
9	2165	2132	2132	29	2271	2210	2210	49	2366	2333	2288
10	2292	2345	2340	30	2836	2740	2740	50	2381	2368	2360
11	3360	3042	3012	31	3878	3652	3631	51	4465	3750	3689
12	3651	3212	3182	32	3914	3564	3552	52	4197	3583	3501
13	3318	2908	2890	33	4076	3683	3638	53	3810	3236	3189
14	3347	3092	3049	34	4276	3931	3901	54	4472	3865	3814
15	3251	3049	3017	35	3853	3580	3545	55	4270	3631	3562
16	3009	2589	2563	36	3231	3115	3086	56	3221	2792	2669
17	2892	2627	2600	37	3279	3095	3053	57	2983	2679	2629
18	2462	2330	2302	38	3514	3364	3354	58	3279	2876	2815
19	2635	2531	2518	39	2998	2975	2975	59	3433	3053	2983
20	2533	2457	2436	40	3370	3188	3188	60	3252	2976	2911
APRD	-	-6.953	-7.6500	APRD	-	-5.710	-6.2833	APRD	-	-11.032	-12.7970
30,20				40,15				40,20			

Table 9. Cont'd.

61	3737	3207	3091	81	3530	2619	2452	101	4336	3495	3388
62	3592	3065	3002	82	3355	2662	2543	102	4278	3713	3564
63	4115	3496	3368	83	3312	2728	2588	103	4216	3610	3481
64	3731	3414	3333	84	3060	2552	2461	104	4139	3682	3516
65	3254	2894	2821	85	3159	2691	2618	105	4078	3612	3474
66	3296	3191	3191	86	2584	2370	2337	106	3379	3132	3132
67	3057	2934	2934	87	2343	2112	2064	107	3236	3212	3085
68	3158	3137	3137	88	2364	2364	2364	108	2891	2801	2779
69	3134	3166	3166	89	2364	2375	2375	109	3627	3339	3303
70	1994	1941	1893	90	2503	2419	2419	110	2610	2505	2505
71	4472	4007	3982	91	5152	4426	4317	111	5438	4842	4678
72	4603	4199	4087	92	4859	3932	3850	112	5640	4943	4818
73	4884	4430	4332	93	4969	4441	4273	113	5873	5066	4999
74	4628	4332	4216	94	4854	4123	4089	114	5560	4977	4885
75	4678	4117	4078	95	5133	4391	4297	115	5536	4954	4813
76	3997	3708	3657	96	3596	3198	3128	116	4177	3643	3536
77	3721	3461	3366	97	3470	3245	3016	117	4066	3707	3608
78	3591	3370	3316	98	3464	3145	2970	118	4590	4030	3880
79	4178	3877	3843	99	3479	3209	3088	119	3953	3757	3498
80	4111	3936	3854	100	3021	3021	2865	120	4320	3946	3795
APRD	-	-7.583	-9.1465	APRD	-	-11.640	-13.8724	APRD	-	-9.908	-12.4515

Table 9. Continue.

50,15				50,20			
P	UB	QDEA	HBBO	P	UB	QDEA	HBBO
121	4016	2976	2803	41	4495	3659	3524
122	3821	2935	2782	42	4713	3755	3614
123	3745	2774	2624	43	4262	3544	3475
124	3631	2774	2655	44	4922	3983	3834
125	3769	2942	2852	45	4380	3608	3435
126	2771	2602	2615	46	3654	3629	3536
127	2979	2972	2972	47	2816	2779	2734
128	3276	3064	3064	48	3593	3502	3487
129	2615	2606	2581	49	3812	3632	3611
130	3211	3190	3190	50	3596	3572	3506
131	5364	4554	4384	51	6224	5462	5349
132	5944	4765	4649	52	6582	5749	5591
133	5294	4540	4444	53	6462	5752	5593
134	5538	4659	4469	54	6074	5576	5368
135	5226	4516	4431	55	6166	5285	5179
136	3817	3433	3291	56	4472	4098	3941
137	3866	3473	3285	57	4438	4123	3986
138	3843	3483	3340	58	4461	4193	4002
139	4007	3233	3116	59	4259	3958	3839
40	3997	3806	3633	60	4521	4110	4009
APRD	-	-13.186	-15.8349	APRD	-	-9.910	-12.3830

Table 10. The ARE of variant for each algorithm.

Problem	P=0.005		P=0.01		P=0.02		P=0.04		P=0.06		P=0.08		P=0.1	
	BRE	ARE	BRE	ARE	BRE	ARE	BRE	ARE	BRE	ARE	BRE	ARE	BRE	ARE
Car1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Car2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Car3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Car4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Car5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Car6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Car7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Car8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rec01	0	0.0926	0	0.016	0	0.0962	0	0.0642	0	0.0642	0	0.0321	0	0.0642
Rec03	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rec05	0.2415	0.2415	0.2415	0.2415	0.2415	0.2415	0.2415	0.2415	0.2415	0.2415	0.2415	0.2415	0.2415	0.2415
Rec07	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rec09	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rec11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rec13	0	0.2902	0	0.2383	0.1036	0.3005	0.1036	0.2073	0	0.1710	0	0.2850	0.1036	0.3902
Rec15	0	0.4256	0	0.1744	0	0.0308	0	0.0462	0	0.2154	0	0.1641	0	0.1385
Rec17	0	0.2366	0	0.2261	0	0.2944	0	0.3891	0	0.3891	0	0.2050	0	0.5100
Rec19	0.2867	0.4682	0.2867	0.4013	0.2867	0.5208	0.2867	0.5399	0.2867	0.5495	0.2867	0.4634	0.2867	0.6737
Rec21	0.1487	1.2147	0.1487	1.2692	1.4378	1.4576	0.5454	1.3485	0.1487	1.1304	1.1899	1.4328	0.8428	1.4378
Rec23	0.4475	0.4923	0.3987	0.4525	0.1492	0.4426	0.4475	0.4873	0.4475	0.5669	0.2486	0.4326	0.3481	0.5619
Rec25	0.4437	0.9471	0	0.5412	0.3581	1.0306	0.4775	0.8715	0.2388	0.9272	0.2388	0.7839	0.2786	0.9073
Rec27	0.6743	0.8512	0.2107	0.4256	0.8007	0.9987	0.2528	0.7290	0.2528	0.9819	0.2528	0.8133	0.2528	0.7965
Rec29	0.5684	0.9969	0	0.7346	0	0.7433	0.4373	0.8920	0.3498	0.9095	0.3061	0.7871	0	0.7346
Rec31	0.2956	0.4893	0.2627	0.4499	0.3941	0.5681	0.2956	0.4729	0.3284	0.5583	0.2627	0.4335	0.2627	0.4532
Rec33	0.1285	0.6294	0	0.4110	0	0.3597	0	0.5267	0	0.5010	0	0.4624	0	0.5267
Rec35	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rec37	1.9996	2.4157	1.3533	1.7592	1.9390	2.1733	1.9996	2.4157	2.0400	2.3308	1.8178	2.5207	2.0602	2.3106
Rec39	1.4547	1.6906	0.8649	1.2168	1.1205	1.4429	1.4547	1.6906	0.9632	1.3839	1.4940	1.7299	1.0812	1.2227
Rec41	2.0565	2.2742	1.5121	1.9516	1.4516	2.2661	1.4718	2.1573	2.1976	2.3710	1.9758	2.2621	2.2177	2.9960
Average	0.3016	0.4743	0.1820	0.3624	0.2856	0.4471	0.2763	0.4510	0.2584	0.4583	0.2867	0.4500	0.2750	0.4816

value of P plays an important role on HBBO, and that when P equals 0.01, the algorithms can produce statistically better results. Thus, we choose P=0.01 in our algorithm HBBO.

Comparisons of HBBO, BEST (LR), M-MAMAC, PACO and PSO$_{VNS}$

In order to evaluate the performance of the HBBO algorithm with total flowtime criterion, this algorithm is compared with the performance of these methods proposed by Liu and Reeves, Rajendran, and Tasgetiren. Liu and Reeves (2001)

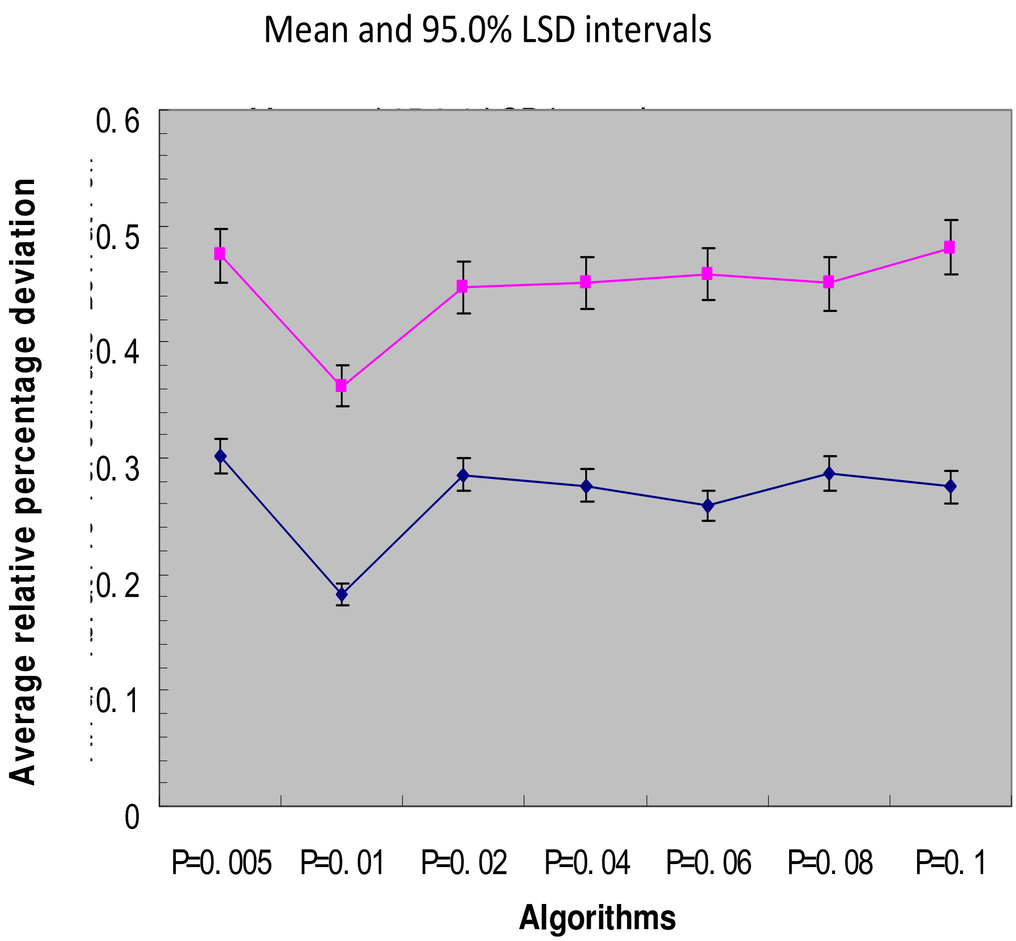

Figure 8. The means plot with LSD intervals for the variant P.

developed a couple of new heuristics. This heuristic is one of the best heuristic. Rajendran and Ziegler (2004) developed two new algorithms called M-MMAS and PACO.PSO$_{VNS}$ (Tasgetiren et al., 2007) presented by the Tasgetiren. The comparative experiments will be carried out on the benchmark problems of Taillard (1993)].

We list the results in Table 11. From Table 11, for the 90 instances considered in this experiment, the HBBO algorithm improves 49 current best solutions. However, the main reason for the success was due to the extensive use of the fast local search and pairwise-based local search in the BBO algorithm where we found that in the first 30 instance, we have 12 instances which have the same solution with other algorithms. For the next 60 instances, PSO$_{VNS}$ have 10 instances better than the HBBO algorithm. For the other 50 instances in middle and large scale, the comparison results are all better than the other algorithms which show the global search ability of our method. As the experimental solution shown in Table12, HBBO can provide the better search ability for the large scale problem with the total flow time of jobs.

The experimental results are reported in Table 12. Further analysis was carried out to see how these algorithms react to the problems. BEST (LR), M-MMAS, PACO, PSO$_{VNS}$, HBBO are denoted as P_1, P_2, P_3, P_4. The APRD is calculated as:

$$APRD = \sum_{i=1}^{R}\left(\frac{H_i - \min(P_k, k=1,2,3,4)\times 100}{\min(P_k, k=1,2,3,4)}\right) \Big/ R$$

The experimental results are listed in Table 12. From the table, it is obvious that the HBBO can provide better solution than the BEST (LR), M-MMAS, PACO, PSOVNS, HBBO, Therefore, the HBBO algorithm provided the new upper bounds which can be used for future research to provide new algorithms; and their results will becompared with our solutions.

Conclusion

In this paper, a promising hybrid biogeography based optimization is proposed to solve PFSSP. BBO is made

Table 11. The best solution by HBBO and other four algorithms.

n*m	Total flowtime of jobs					n*m	Total flowtime of jobs				
	BEST(LR)	**M-MMAS**	**PACO**	**PSO_{VNS}**	**HBBO**		**BEST(LR)**	**M-MMAS**	**PACO**	**PSO_{VNS}**	**HBBO**
20*5	14222	14056	14056	14033	14033	50*10	88770	89599	88942	88031	87654
	15446	15151	15214	15151	15151		85600	83612	84549	83624	83555
	13676	13416	13403	13301	13301		82456	81655	81338	80609	80322
	15750	15486	15505	15447	15447		89356	87924	88014	87053	86921
	13633	13529	13529	13529	13529		88482	88826	87801	87263	86984
	13265	13139	13123	13123	13123		89602	88394	88269	87255	86816
	13774	13559	13674	13548	13548		91422	90686	89984	89259	89517
	13968	13968	14042	13948	13948		89549	88595	88281	87192	87607
	14456	14317	14383	14295	14295		88230	86975	86995	86102	86015
	13036	12968	13021	12943	12943		90787	89470	89238	88631	88783
20*10	21207	20980	20958	20911	20911	50*20	129095	127348	126962	128622	126719
	22927	22440	22591	22440	22440		122094	121208	121098	122173	119600
	20072	19833	19968	19833	19833		121379	118051	117524	118719	117241
	18857	18724	18769	18710	18710		124083	123061	122807	123028	121319
	18939	18644	18749	18641	18641		122158	119920	119221	121202	119099
	19608	19245	19245	19249	19245		124061	122369	122262	123217	121343
	18723	18376	18377	18363	18363		126363	125609	125351	125586	123807
	20504	20241	20377	20241	20241		126317	124543	124374	125714	123054
	20561	20330	20330	20330	20330		125318	124059	123646	124932	122765
	21506	21320	21323	21320	21320		127823	126582	125767	126311	124970
20*20	34119	33623	33623	34975	33623	100*5	256789	257025	257886	254762	254931
	31918	31604	31597	32659	31587		245609	246612	246326	245315	244066
	34552	33920	34130	34594	33920		241013	240537	241271	239777	239152
	32159	31698	31753	32716	31661		231365	230480	230376	228872	228655
	34990	34593	34642	35455	34557		244016	243013	243457	242245	241678
	32734	32637	32594	33530	32564		235793	236225	236409	234082	233800
	33449	33038	32922	33733	32922		243741	243935	243854	242122	241440
	32611	32444	32533	33008	32412		235171	234813	234579	232755	232375
	34084	33625	33623	34446	33600		251291	252384	253325	249959	249056
	32537	32317	32317	33281	32262		247491	246261	246750	244275	244503
50*5	65663	65768	65546	65058	64848	100*10	306375	305004	305376	303142	301927
	68664	68828	68485	68298	68159		280928	279094	278921	277109	277952
	64378	64166	64149	63577	63331		296927	297177	294239	292465	291839
	69795	69113	69359	68571	68458		309607	306994	306739	304676	304591
	70841	70331	70154	69698	69691		291731	290493	289676	288242	288676
	68084	67563	67664	67138	67081		276751	276449	275932	272790	273031
	67186	67014	66600	66338	66470		288199	286545	284846	282440	282252
	65582	64863	65123	64638	64620		296130	297454	297400	293572	293951
	63968	63735	63483	63227	63170		312175	309664	307043	305605	305348
	70273	70256	69831	69195	69213		298901	296869	297182	295173	295098

suitable for permutation flow shop scheduling by using the LRV rule. This is proposed to convert the continuous encoding in BBO to a discrete job permutation. For initialing the population, The NEH heuristic was combined with the random initialization to initialize the population with certain quality and diversity. In BBO-local search,

Table 11. Contnd.

n*m	Total flowtime of jobs				
	BEST(LR)	M-MMAS	PACO	PSOVNS	HBBO
100*20	383865	373756	372630	374351	371828
	383976	383614	381124	379792	377692
	383779	380112	379135	378174	376161
	384854	380201	380765	380899	378623
	383802	377268	379064	376187	375685
	387962	381510	380464	379248	378616
	384839	381963	382015	380912	378543
	397264	393617	393075	392315	389896
	387831	385478	380359	382212	380256
	394861	387948	388060	386013	384096

Table 12. Relative performance of five heuristic for mean percent relative increase in total flowtime with respect to the best heuristic solution.

	BES(LR)	M-MAAS	PACO	PSOVNS	HBBO
20*5	1.36	0.20	0.45	0.00	0.00
20*10	1.43	0.05	0.32	0.002	0.00
20*20	1.22	0.12	0.19	2.83	0.00
50*5	1.43	1.01	0.83	0.13	0.02
50*10	2.42	1.43	1.17	0.19	0.09
50*20	2.37	1.05	0.74	1.60	0.00
100*5	0.96	0.91	1.03	0.20	0.02
100*10	1.54	1.13	0.85	0.08	0.06
100*20	2.15	0.90	0.67	0.49	0.00
Average	1.6533	0.7556	0.6944	0.6136	0.0211

BBO's migration and mutation can perform a wide global search in the whole solution space. This means that BBO-local search has the ability of obtaining enough sub-regions over the whole solution space. Then, the fast local search is proposed to enhance the individual of the BBO with a certain probability. Finally, the pairwise based local search is used to enhance the global optimal solution and help the algorithm to escape from local minimum. Experimental results and comparisons show the effectiveness of the proposed HBBO for PFSSP. Moreover, the further work is to study the theoretical aspects as well as the performance of the technique. The other problem is to extend the algorithm to solve other combination problem such as job shop scheduling.

ACKNOWLEDGMENTS

This research is fully supported by the National Natural Science Foundation of China under Grant Nos.60473042, 60573067, 60803102, and also funded by NSFC Major Research Program 60496321: Basic Theory and Core Techniques of Non Canonical Knowledge.

REFERENCES

Bansal SP (1977). Minimizing the sum of completion times of n-jobs over M-machines in a flowshop---a branch and bound approach, AIIE Trans., 9: 306-311.

Bassem J, Eddaly M (2009). An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems, Comput. Operat. Res., 36: 2638-2646.

Bhattacharya A, Chattopadhyay P (2010). Biogeography-based optimization for different economic load dispatch problems, IEEE Trans Power Syst., 25: 1064-1077.

Bhattacharya A, Chattopadhyay P (2010). Solving complex economic load dispatch problems using biogeography-based optimization, Expert Syst. Appl., 37: 3605-3615.

Campbell HG, Dudek RA, Smith ML (1970). A heuristic algorithm for the n job, m machine sequencing problem, Manage. Sci.,16:630-637.

Cariler J (1978). Ordonnancements a contraintes disjonctives, R.A.I.R.O. Recherche Operationelle/Oper .12: 333-351.

Croce FD, Ghirardi M, Tadei R (2002). An improved branch-and-bound algorithm for the two machine total completion time flow shop problem, Eur. J. Operat. Res., 139: 293-301.

Croce FD, Narayan V, Tadei R (1996). The two-machine total completion time flow shop problem, Eur. J. Operat. Res., 90 :227-237.
Demirkol E, Mehta S, Uzsoy R (1998), Benchmarks for shop scheduling problems. Eur. J. Operat. Res., 109: 137-141.
Dipak L, Uday K, Chakraborty (2007). An efficient stochastic hybrid heuristic for flowshop scheduling, Eng. Appl. Artif. intel., 20:851-856.
Dong XY, Huang HK (2009), An iterated local search algorithm for the permutation flowshop problem with total flowshop crition, Comput. Operat. Res., 36: 1664-1669.
Framinan JM, Leisten R (2003). An efficient constructive heuristic for flowtime minimization in permutation flow shops, OMEGA. 31: 311-317.
Garey MR, Johnson DS (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness, San Francisco, CA: Freeman, pp. x+338. ISBN 0-7167-1045-5.
Gong W, Cai Z, Ling C, Li H (2010). A real-coded biogeography-based optimization with mutation, Appl. Math. Comput., 216: 2749-2758.
Grabowski J, Wodecki M (2004). A very fast tabu search algorithm for the permutation flowshop problem with makespan criterion, Comput. Operat. Res., 31: 1891-1909.
Gupta JND (1972). Heuristic algorithms for multistage flowshop scheduling problem, AIIE Transactions. 4: 11-18.
Ho JC, Chang YL(1991). A new heuristic for the n-job, M-machine flow-shop problem, Eur. J. Operat. Res., 52: 194-202.
Ignall E, Schrage L (1965). Application of the branch and bound technique to some flowshop scheduling problem, Operat. Res., 13: 400-412.
Johnson SM (1954). Optimal two-and three-stage production schedules, Naval Res. Logist. Quart., 1: 61-68.
Liu B, Wang L (2007). An Effective PSO-Based Memetic Algorithm for flow shop scheduling, IEEE Transaction Systems, Man, Cybernetics-Part B. 37: 18-27.
Liu J, Reeves CR (2001). Constructive and composite heuristic solutions to $P\|\sum C_i$ scheduling problem, Eur. J. Operat. Reas., 132: 439-452.
Ma H (2010). An analysis of the equilibrium of migration models for biogeography-based optimization, Inf. Sci., 180: 3444-3464.
Ma H, Simon D (2010). Biogeography-based optimization with blended migration for constrained optimization problems, Gen. Evol. Comput. Conf. Portland, Oregon., 6: 417-418.
Ma H, Simon D (2010). Blended biogeography-based optimization for constrained optimization, Engineering Applications of Artificial Intelligence, 24(3): 517-525.
Murata T, Ishibuchi H, Tanaka H (1996). Genetic algorithms for flowshop scheduling problems, Comput. Operat. Res., 30: 1061-1071.
Nawaz M, Enscore Jr EE, Ham I (1983). A heuristic algorithm for the m-machine, n-job flow shop sequencing problem, OMEGA. 11: 91-95.
Nearchou AC (2004). A novel metaheuristic approach for the flow shop scheduling problem, Eng. Appl. Artif. Intel. 17: 289-300.
Nowicki E, Smutnicki C (1996). A fast tabu search algorithm for the permutation flowshop problem, Eur. J. Operat. Res., 91:160-175.
Ogbu F, Smith D (1990). The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem, Comput. Operat. Res., 17: 243-253.
Orhan E, Alper D (2004). A new approach to solve hybrid flow shop scheduling problems by artificial immune system, Future Generat. Comput. Systs., 20: 1083-1095.
Osman I, Potts C (1989). Simulated annealing for permutation flow shop scheduling, OMEGA. 17: 551-557.
Palmer DS (1965). Sequencing jobs through a multistage process in the minimum total time: A quick method of obtaining a near-optimum, Operat. Res. Quart., 16:101-107.
Qian B, wang L (2008).A hybrid differential evolution method for permutation flow-shop scheduling, The International J. Adv. Man. Technol., 38: 757-777.
Rajendran C (1993), Heuristic algorithm for scheduling in a flowshop to minimize total flowtime, Int. J. Prod. Econ., 29: 65-73.
Rajendran C, Ziegler H (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs, Eur. J. Operat. Res., 155: 426-438.
Reeves C (1995). A genetic algorithm for flowshop sequencing, Comput. Operat. Res., 22: 5-13.
Reeves C, Yamada T (1998). Genetic algorithms, path relinking and the flowshop sequencing problem, Evol. Comput., 6:45-60.
Revees C (1993). Improving the efficiency of tabu search for machine sequencing problem, J. Operat. Res. Soc., 44: 375-382.
Rinnooy KAHG (1976). Machine Scheduling Problems: Classification, Complexity, and Computations, Nijhoff, The Hague, pp.130-194.
Roy P, Ghoshal S, Thakur S (2010). Biogeography-based optimization for economic load dispatch problems, Electric Power Compon. Syst., 38:166-181.
Simon D (2008). Biogeography-Based Optimization. IEEE Trans. Evol. Comput., 12: 702-713.
Stadtler H (2005). Supply chain management and advanced planning basics, overview and challenges, Eur. J. Operat. Res., 163: 575-588.
Stafford EF (1988). On the development of a mixed integer linear programming model for the flowshop sequencing problem, J. Operat. Res. Soc., 39: 1163-1174.
Stützle T (1998). Applying iterated local search to the permutation flowshop problem, Technical Report, AIDA-98-04, Darmstad University of Technology, Computer Science Department, Intellctics Group, Darmstad, Germany.
Stützle T (1998). An ant approach to the flowshop problem, In: Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT'98), Verlag Mainz, Aachen, Germany, pp. 1560-1564.
Taillard E (1990), Some efficient heuristic methods for the flowshop sequencing problems, Eur. J. Operat. Res., 47:65-74.
Taillard E (1993). Benchmarks for basic scheduling problems. Eur. J. Operat. Res., 64: 278-285.
Tasgetiren MF, Liang YC, Sevkli M, Gencyilmaz G (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, Eur. J. Operational Res., 177: 1930-1947.
Tasgetiren MF, Liang YC, Sevkli M, Gencyilmaz G (2004). Particle swarm optimization algorithm for makespan and maximum lateness minimization in permutation flowshop sequencing problem. In: Proceedings of the fourth international symposium on intelligent manufacturing systems. Turkey: Sakarya, pp. 431-41.
Tseng LY, Lin YT (2009). A hybrid genetic local search algorithm for the permutation flowshop scheduling problem, Eur. J. operat. Res., 198: 84-92.
Wang L, Wu H, Tang F, Zheng DZ (2005). A hybrid quantum inspired genetic algorithm for flow shop scheduling, Lect Notes Comput. Sci., 3645: 636-644.
Wang L, Zheng DZ (2003). An effective hybrid heuristic for flow shop scheduling, The Int. J. Adv. Man. Technol., 21: 38-44.
Zhang Y, Li XP, Wang Q (2009). Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization, Eur. J. Operat. Res.,196: 869-876.
Zheng TM, Yamashiro MS (2010). Solving flow shop scheduling problems by quantum differential evolutionary algorithm, Int J Adv Manuf Technol., DOI 10.1007/s00170-009-2438-4.