IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON RECENT COMPUTATIONAL METHODS IN KNOWLEDGE ENGINEERING AND INTELLIGENCE COMPUTATION

Received October 5, 2017, accepted October 30, 2017, date of publication November 2, 2017, date of current version December 5, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2769100

A Hybrid Chaotic Biogeography Based Optimization for the Sequence Dependent Setup Times Flowshop Scheduling Problem With Weighted Tardiness Objective

YUNHE WANG AND XIANGTAO LI, (Member, IEEE)
School of Information Science and Technology, Northeast Normal University, Changchun 130117, China

Corresponding author: Xiangtao Li (lixt314@nenu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61603087, in part by the Natural Science Foundation of Jilin Province under Grant 20160101253JC and in part by the Fundamental Research Funds for the Central Universities under Grant 2412017FZ026.

ABSTRACT This paper proposes a hybrid chaotic biogeography-based optimization (HCBBO) for solving the sequence dependent setup times flowshop scheduling problem with the objective of minimizing the total weighted tardiness. First of all, a largest-order-value rule is employed to transform continuous vectors into discrete job permutations. Second, the chaotic theory and the problem-specific Nawaz-Enscore-Ham heuristic are applied to compose the initial population with the property of intensification and diversification. Third, an improved biogeography-based optimization is introduced to improve the global search ability by designing new migration and mutation schemes. Meanwhile, a further local search is proposed and embedded in HCBBO to enhance the quality of the elite habitats. In addition, an effective perturbation is applied to avoid the solutions getting trapped in the local optima. Computation comparison experiments of seven benchmark algorithms on the Taillard benchmark problems are provided to verify the efficiency of the proposed algorithm. From the experiment results, it can conclude that HCBBO beats other compared algorithms effectively with higher quality and robustness solutions.

INDEX TERMS Flowshop scheduling, biogeography based optimization, sequence dependent setup times, weighted tardiness.

I. INTRODUCTION

Scheduling problems consist of a series of classical combinational optimization problems to allocate the shared resources and sort the production tasks within a period of time. Therein the flowshop scheduling problem is a famous combinational optimization problem which is not only a hotspot issue in the theoretical research area but also taking necessary effect in manufacturing [1]–[5]. According to the process constraints and production conditions, it is divided into different kinds of problems with practical meanings such as the permutation flowshop scheduling problem (PFSSP), the no-wait flowshop scheduling problem (NWFSSP), and the flowshop scheduling problem with sequence dependent setup times (SDST-FSSP). Particularly, SDST-FSSP has attracted more and more attentions from ambitious researchers in the industrial process and mathematics. Moreover, the sequence dependent setup time is not only dependent with the previous job but also with the job to be processed which is not ignored and part of the processing time [6], [7] including the releasing time or the fixing time. As it is important desperately for the in-time or the make-to-order environment to take into account the large number of jobs waiting for completing after the due dates in the job permutation, the weighted tardiness objective is proposed to find the best sequence with the minimized total weighted tardiness for saving the cost. In addition, for each job, a weight is given as a priority to indicate the relative importance to other jobs.

Since SDST-FSSP is a NP-hard problem regarding the computational complexity in the combinational optimization [8], many different computational methods have been proposed to solve this problem. However, because the computation space and time increase exponentially with the scale,

exact methods can only handle scheduling problems with small scales [9], [10]. In order to reduce time in getting appropriate solutions for FSSP, heuristic algorithms are developed which may result in suboptimal solutions of poor quality. It is divided into the constructive algorithm and the improved algorithm [1]. For the constructive algorithm it was first proposed by Johnson [11] which can solve FSSP for minimizing makespan with the time complexity of O(nlogn). After that, Campbell-Dudek-Smith (CDS) algorithm [12] was developed to split FSSP into some FSSPs with small scales and resolve them by Johnson algorithm respectively. In particular, Nawaz-Enscore-Ham heuristic (NEH) [4] is an excellent constructive algorithm in solving FSSP effectively. For the improved heuristic algorithm, it employs some effective strategies on the algorithm for increasing the efficiency of it such as, Suliman algorithm initialized the solutions by the CDS algorithm and enhanced the quality of solutions by swapping two jobs chosen randomly [5]. Furthermore, the iterated greedy (IG_RS) algorithm with a prominent performance compared with many fine-tuned algorithms [1] on FSSP based on a highly effective and improved construction scheme was raised in [13].

Recently, evolutionary algorithms (EA) are built to achieve better solutions for complex optimization problems with the characteristics of self-organizing, self-adaptive and self-learning. Following this, two algorithms namely HA and HA_IS2 were proposed [14], [15] to enhance the performance of algorithms. Moreover, two improved memetic algorithms on the base of the classical GA were proposed by Ruiz *et al.* [16] with a better performance than those in [17]. Except for the above three types of algorithms for dealing with FSSP, the hybrid algorithms are raised for increasing the performance of existing algorithms [18]. For example, an iterated hill climbing method was used to improve the performance of GA [19]. It is noteworthy to mention that two well-performing algorithms (IG_RS and MA) were combined with an effective local search on the destruction and construction aspects to generate IG_RS$_{LS}$ and MA$_{LS}$ respectively. By computational experiments, they outperformed other heuristics in obtaining better results [15].

In this paper, a hybrid chaotic biogeography based optimization (HCBBO) is proposed with the goal of minimizing the total weighted tardiness. Firstly, a Nawaz-Enscore-Ham heuristic combined with the chaotic sequence is developed to initialize the habitat population with high quality and diversity based on a largest-order-value rule. Secondly, regarding the migration and mutation operators, the improved methods are proposed to enhance the diversity as well as the quality. Moreover, a further local search is employed on the elite population to converge the algorithm more quickly and look for better habitats in the search area. However, in order to explore better solutions in expansive regions under the condition that the algorithm is trapped in the local optima, an efficient perturbation is applied to interfere the elite solutions and guide them to the neighbour area. To validate the performance of the proposed algorithm, seven outstanding algorithms including GA, MA, MA$_{LS}$, IG_RS, IG_RS$_{LS}$, SA_PR and HA_IS2 referred above [8] are compared on the 480 benchmark instances of SDST-FSSP. In conclusion, the statistical results indicate that HCBBO behaves prominently with the best solutions among all compared and excellent algorithms.

The rest of this paper is organized as follows: Section II reviews the flowshop scheduling problem with sequence dependent setup times and the objective of total weighted tardiness. The detail of the proposed algorithm is described in Section III. In Section IV, we conduct the computational experiments. Finally, the conclusion and the future work are summarized in Section V.

II. THE FLOWSHOP SCHEDULING PROBLEM WITH SEQUENCE DEPENDENT SETUP TIMES

A. SDST-FSSP

For the flowshop scheduling problem with sequence dependent setup times (SDST-FSSP), a set of m machines comprising of $M = \{1, 2, 3, \ldots, m\}$ has to process a set of jobs including $J = \{1, 2, 3, \ldots, n\}$ sequentially, such as the processing sequence will be first processed on Machine 1, then the same one will be processed on Machine 2, until in the end it will be handled on Machine m. It means that the operating order of each processing sequence is identical for all the machines. In this problem, each job can be processed on one machine at each time while each machine can process one job. None of the $n \times m$ operations has the priority and all of them are not allowed to preempt. The description of the flowshop scheduling problem with sequence dependent setup times can be stated in the following formulas:

$$\begin{aligned} C(\pi_1, 1) &= p_{\pi_1,1} \\ C(\pi_i, 1) &= C(\pi_{i-1}, 1) + p_{\pi_i,1} + S_{1,\pi_{i-1},\pi_i}, i = 2, \ldots, n \\ C(\pi_1, j) &= C(\pi_1, j-1) + p_{\pi_1,j}, \quad j = 2, \ldots, m \\ C(\pi_i, j) &= \max(C(\pi_{i-1}, j) + S_{j,\pi_{i-1},\pi_i}, C(\pi_i, j-1)) \\ &\quad + p_{\pi_i,j}, i = 2, \ldots, n; \quad j = 2, \ldots, m \\ C_{max}(\pi) &= C(\pi_n, m) \end{aligned} \tag{1}$$

where π is a job permutation, $p_{\pi_i,j}$ denotes the processing time of job π_i on machine j. S_{j,π_i,π_k} represents the setup time when producing job π_k on the machine j, after having processed job π_i. Besides, $C(\pi_i, m)$ is the completion time of π_i on machine m. Here it is assumed that $S_{j,\pi_0,\pi_i} = 0$, $C_{\pi_i,0} = 0$ and $C_{\pi_0,j} = 0$. ($\pi_i, \pi_k \in J, i \neq k, j \in M$)

B. TOTAL WEIGHTED TARDINESS

In this scheduling problem, the objective that total weighted tardiness has yet to be addressed. For each job π_i, the weighted tardiness is given by the following step:

$$T_{\pi_i} = max\{C(\pi_i, j) - D_{\pi_i}; 0\} \times W_{\pi_i} \tag{2}$$

where $C(\pi_i, j)$ is provided as the completion time of job π_i on machine j. D_{π_i} denotes the due date for job π_i and W_{π_i} gives the weight for job π_i. Besides, the weight for each job is drawn from the distribution range [1,10] [6]. What's more,

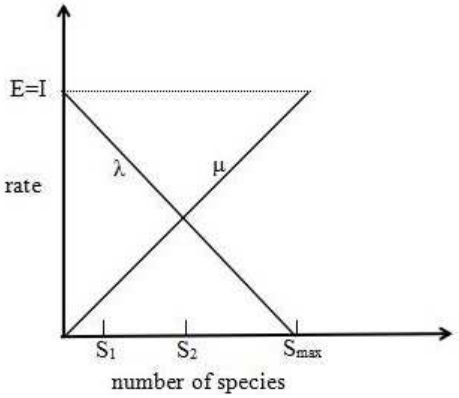

FIGURE 1. The linear migration model.

the due dates of SDST-FSSP for each job π_i are determined by the following formulas [8], [20] ($\pi_i \in J, j \in M$):

$$Sum_{P_{\pi_i}} = \sum_{j=1}^{m} p_{\pi_i,j} \tag{3}$$

$$Aver_{S_{\pi_i}} = \sum_{j=1}^{m} \frac{\sum_{k=1,k\neq i}^{n} S_{j,\pi_i,\pi_k}}{n-1} \tag{4}$$

$$D_{\pi_i} = (Sum_{P_{\pi_i}} + Aver_{S_{\pi_i}}) \times (1 + random \times 3). \tag{5}$$

For all the n jobs, the total weighted tardiness is described by the following formula:

$$TWT = \sum_{i=1}^{n} T_{\pi_i}. \tag{6}$$

III. A HYBRID CHAOTIC BIOGEOGRAPHY BASED OPTIMIZATION

A. BIOGEOGRAPHY BASED OPTIMIZATION

In the biogeography based optimization (BBO) [21]–[23], a habitat is used to represent an individual of the habitat population or a solution, the performance of a habitat called the habitat suitability index (HSI) indicates the fitness of a solution. Besides, HSI is a vector of suitability index variables (SIVs) which denote by real numbers. A solution with good quality has a high HSI while a habitat with a low HSI represents that the solution is poor in the population. Then the evolution of each habitat depends on the migration among different habitats and the mutation in the habitats internally.

Here, a simple linear migration model illustrated as Figure 1 is used to calculate the immigration and emigration rate concerning HSI of each habitat with different number of species [21]. λ represents the immigration rate while the emigration rate is denoted by μ. Based on both, the migration operator changes SIVs in each HSI to implement the exploitation part in BBO. They can be defined as

$$\begin{aligned} \lambda_k &= I(1 - \frac{k}{S_{max}}) \\ \mu_k &= \frac{E}{S_{max}} k \end{aligned} \tag{7}$$

where S_{max} represents the maximum number of species, $E \in (0, 1]$ is the maximum immigration rate and $I \in (0, 1]$ is the maximum emigration rate. k indicates the number of species.

Based on the above analyses, the scheme of the migration operator is obtained as shown in Algorithm 1.

Algorithm 1 The Migration Operator

Input:
The number of the habitats in the population NP.
The immigration rate of the i-th habitat λ_i.
The emigration rate of the i-th habitat μ_i.
The i-th habitat in the habitat population H_i.
Output:
The migrated habitat population H_{mi}.

```
for i = 1 to NP do
    if rand(0, 1) < λ_i then
        for j = 1 to NP do
            if rand(0, 1) < μ_j then
                Randomly select an SIV from H_j to replace a random SIV in H_i;
            end if
        end for
    end if
end for
H_mi = H;
return H_mi
```

In the mutation phase, it performs an adaptive process namely the exploration procedure based on a mutation probability to modify the feature of existing habitats randomly. The mutation probability of the habitat h can be obtained as

$$m_h = m_{max}(1 - \frac{P_h}{P_{max}}) \tag{8}$$

where m_{max} is the maximum mutation probability, P_h is the probability of the h-th species, P_{max} is the maximum probability of P_h ($h \in 1, 2, \ldots, NP$) and NP is the habitat population size. Furthermore, P_h can be calculated as

$$\dot{P}_h = \begin{cases} -(\lambda_h + \mu_h)P_h + \mu_{h+1}P_{h+1}, & h = 0. \\ -(\lambda_h + \mu_h)P_h + \lambda_{h-1}P_{h-1} + \mu_{h+1}P_{h+1}, & \\ \qquad\qquad 1 \le h \le S_{max} - 1. & \\ -(\lambda_h + \mu_h)P_h + \lambda_{h-1}P_{h-1}, & h = S_{max}. \end{cases} \tag{9}$$

The pseudo code of the mutation operator can be presented as shown in Algorithm 2.

In the above scheme, a strategy which uses the first NE best habitats among the current habitat population to replace with the last NE worse habitats is employed. It is called the elitist selection strategy for keeping the habitats with good quality in the evolutionary process.

Algorithm 2 The Mutation Operator

Input:
The number of the habitats in the population NP.
The number of the dimensions in the solution vector n.
The mutation probability of the i-th habitat m_i.
The j-th job of the i-th habitat $H_i(j)$.
The i-th habitat in the habitat population H_i.
Output:
The mutated habitat population H_{mu}.

```
for i = 1 to NP do
    for j = 1 to n do
        if rand(0, 1) < m_i then
            Replace the chosen H_i(j) with a randomly generated SIV;
        end if
    end for
end for
H_mu = H;
return H_mu
```

B. OVERALL FRAMEWORK

A hybrid chaotic biogeography based optimization (HCBBO) for SDST-FSSP-TWT is proposed in this section based on the framework of BBO. It is obvious that each method of HCBBO plays an important role in increasing the effectiveness and efficiency of the algorithm.

Firstly, the current habitat population and the best habitat are initialized. The population initialization method is a problem-specific NEH algorithm embedded with an effective chaotic sequence generator which benefits from the ergodicity and pseudo-randomness.

Following this, the main iteration of HCBBO is derived from the migration, mutation and the elite strategy of BBO. Hence, an improved migration operator to migrate some habitats and an improved mutation operator to mutate the partial habitats are employed on the current sorted habitat population. Thirdly, for each habitat of the elite population, an effective insert_based local search for searching better solutions deeply is implemented. However, if the HSI of the best habitat is not enhanced after some iterations, a perturbation method on each elite habitat to let the algorithm jump out of the local optimum and guide the search to the neighbour regions is carried out. The framework of HCBBO can be briefly summarized in Algorithm 3.

C. INITIALIZATION

1) SOLUTION REPRESENTATION

In HCBBO, a continuous encoding scheme is employed by a n-dimension real vector. It is not appropriate for solving discrete optimization problems directly. In order to apply the continuous encoding mode to SDST-FSSP-TWT, a relationship between the job permutation and the vector of habitat in the population should be constructed. Therefore, a largest-order-value rule [24] based on the random position value is

Algorithm 3 HCBBO

Input:
The habitat population size NP.
The number of jobs n.
The number of elite habitats NE.
The maximum number of the reiteration $MaxReiterationNum$.
The maximum number of Insert_based Local Search $MaxLSNum$.
The user_defined maximum mutation probability $MaxMP$.
Output:
The best habitat h_{best}.

```
Initialize the current habitat population H_c = {h0_1, h0_2, ..., h0_NP};
Carry out the problem-specific NEH heuristic;
SortPopulation(H_c):Sort the habitat population in the descending order of fitness (HSI) to generate H_c = {h_1, h_2, ..., h_NP};
Initialize ElitePop = {h_1, h_2};
Initialize h_best = h_1;
Initialize NoImproved = 0;
for i = 1 to NP do
    Initialize m(i), λ(i), μ(i);
end for
while the stopping criterion is not met do
    ImprovedMigrationOperator(H_c);
    ImprovedMutationOperator(H_c);
    SortPopulation(H_c);
    UpdateElitePop(H_c, ElitePop);
    for i = 1 to NE do
        iterNum = 0;
        while iterNum ≤ MaxLSNum do
            random (r) ∈ (0, n];
            h'_i = Insert_basedLocalSearch(r, h_i);
            if TWT(h'_i) < TWT(h_i) then
                h_i = h'_i; break;
            else
                iterNum = iterNum + 1;
            end if
        end while
    end for
    tempH = h_1;
    if TWT(tempH) < TWT(h_best) then
        h_best = tempH;
        NoImproved = 0;
    else
        NoImproved = NoImproved + 1;
    end if
    if NoImproved > MaxReiterationNum then
        for i = 1 to NE do
            h_i = Perturbation(h_i);
        end for
        NoImproved = 0;
    end if
    for i = 1 to NE do
        UpdateCurrentPop(ElitePop);
    end for
end while
return h_best
```

TABLE 1. Solution representation by LOV rule.

Initial Job	1	2	3	4	5	6
Position Value	1.36	3.85	2.55	0.63	2.68	0.82
Job Permutation	4	1	3	6	2	5

used to convert the solution vector to the job sequence. In this rule, each job is assigned into the scheduling sequence by the descending order of random real numbers in the vector. A simple example is introduced in Table I.

The following formula is used to generate each dimension of a random real number vector:

$$x(i) = x_{min} + (x_{max} - x_{min}) \times r \tag{10}$$

where r represents a uniform number generated randomly between 0 and 1. $x_{min} = 0.0$ and $x_{max} = 4.0$ are used in HCBBO.

When generating the vector, a vector with the identical position values may occur. It might lead to a different permutation from the pervious one generated by the local search methods. To overcome this drawback, the revision process [25] is adopted in our algorithm. The pseudo code is provided in Algorithm 4.

Algorithm 4 The Revision Process

Input:
The initial job sequence x.
The i-th job of the job sequence $x(i)$.
The number of jobs n.
Output:
The revised individual x_r.

```
for i = 1 to n − 1 do
    if x(i) == x(i + 1) then
        if i ≠ 1 then
            x(i) = x(i) + (x(i − 1) − x(i))/n;
        else
            x(i) = x(i) + σ;
        end if
    end if
end for
x_r = x;
return x_r
```

2) CHAOTIC SEQUENCE

In HCBBO, a chaotic sequence is generated based on the logistic map [26]. It has the characteristics of uniformity, randomicity and regularity that are very effective for avoiding the heuristic optimization getting trapped in the local optimum [27]–[29]. The simple equation is shown as follows:

$$X_i^{t+1} = \eta X_i^t (1 - X_i^t) \tag{11}$$

where X_i^t indicates the t-th chaotic number in the i-th dimension of the solution vector and t is the iteration number. η is the growth rate for the next iteration relative to the current iteration. Besides, $X_i^t \in (0, 1)$ is under conditions that $X_i^0 \in (0, 1)$ and $X_i^0 \neq \{0.0, 0.25, 0.5, 0.75, 1.0\}, \forall i \in [1, n]$. Furthermore, $\eta = 4$ has been applied in HCBBO. It should be noted that a vector of random real numbers in the range (0,4) is applied to represent the job permutation. The equation which generates the real numbers between 0 and 4 using $X_i^t, \forall i \in [1, n]$ can be described as below:

$$x_i^t = x_{min} + X_i^t (x_{max} - x_{min}) \tag{12}$$

In this equation, $i \in [1, n]$ denotes the dimension number of the vector, X_i^t is the i-th chaotic number for the t-th iteration. To conclude, at the beginning of the proposed algorithm, a chaotic generator is applied to produce a random and unpredictable real sequence. Then we use the LOV rule to convert the habitat represented by the real number vector to the job sequence. Lastly, the fitness (TWT) of the job sequence is calculated to evaluate a solution.

3) INITIALIZATION METHOD

At the beginning of HCBBO, a problem-specific NEH heuristic is applied to initialize the habitat population which consists of NP habitats. It is important in enhancing HSI of the habitat which has been proved effective in [30]. In the problem-specific NEH method, only a part of habitats have undergone the NEH method. The process of NEH can be described in the following steps.

1) Select a habitat h from H_c randomly, extract the first two jobs and evaluate the possible job schedules with them. The better one with less TWT is selected as the current permutation.

2) For each unscheduled job j in h, put it in all the possible positions of the current scheduled permutation, then all the possible partial permutations are produced. The best one of them is chosen as the current sequence to schedule the next job.

3) A new habitat h_{new} is generated after arranging all jobs, if the HSI of h_{new} is better than h, then h will be replaced by h_{new}.

In regard to this problem-specific NEH method, a random habitat is selected to execute this method. Because not all the habitats carry out the NEH heuristic, the diversity of the initial habitat population is guaranteed. As a result, a habitat population H_c is formed with the property of good quality and diversity. Afterwards, the best habitat h_{best} is initialized by the first habitat selected from the sorted habitat population. At the last of HCBBO, h_{best} will be outputted as the final result with the objective of total weighted tardiness.

D. MIGRATION AND MUTATION METHOD

In this section, an improved migration operator combined with the roulette strategy which helps to choose the habitat randomly is adopted based on the migration operator of BBO. In addition, an improved mutation method is proposed embedded with two ways of local search. It plays a significant role in enhancing the diversity of the habitat population.

Algorithm 5 ImprovedMigrationOperator()

Input:
The habitat population size NP.
The current population H_c including h_i.
The number of jobs n.
The migration probability λ and μ.
Output:
The population after migration H_c.

for $i = 1$ to NP **do**
  $r = \text{random}(0,1)$;
  for $j = 1$ to n **do**
    if $r < \lambda(i)$ **then**
      Employ the roulette strategy using $\mu(i)$ to choose the habitat h_t to be migrated;
      $h_i(j) = h_t(j)$;
    end if
  end for
end for
return H_c

Algorithm 6 ImprovedMutationOperator()

Input:
The habitat population size NP.
The current population H_c including h_i.
The number of jobs n.
The mutation probability m.
Output:
The population after mutation H_c.

for $i = 1$ to NP **do**
  $r = random(0, 1)$;
  if $r < m(i)$ **then**
    $h_i = Inverse\_basedLocalSearch(h_i)$;
  else
    $p = random(0, n)$;
    $h_i = Insert\_basedLocalSearch(h_i, p)$;
  end if
end for
return H_c

The improved migration and mutation method of HCBBO is shown in Algorithm 5 and Algorithm 6 respectively.

1) LOCAL SEARCH METHOD

In the scheme of HCBBO, an insert_based local search has been employed in the multiple further search and the mutation operator. In [30]–[37], the insert_based local search has demonstrated its high efficiency. The detail process is given as follows.

(1) Input a habitat h and a given position p.

(2) Let job j be a job in the position p of h, place j on each of the left potential positions of h to result $n - 1$ neighbor habitats.

(3) Let h_{best} be the best habitat based on the minimal total weighted tardiness among the $n - 1$ neighbor habitats.

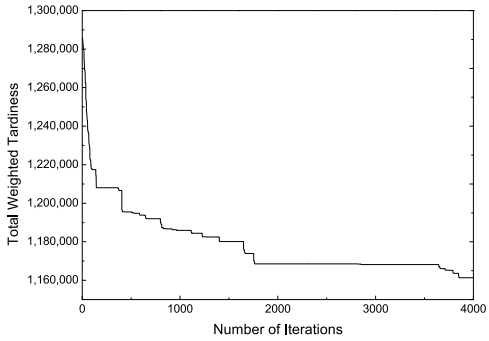

FIGURE 2. The convergence rate of HCBBO on Ta071 with 4000 iterations.

(4) Output h_{best}.

In addition, another local search method which is called the inverse_based local search has been applied in the mutation operator [23]. The scheme of it details as below.

(1)Input a habitat h and let the counter of iteration be zero.

(2)Choose two positions p_1 and p_2 randomly ($p_1 < p_2$). Inverse the partial permutation between p_1 and p_2 to generate a new habitat h_{new}.

(3)If the fitness of h_{new} is better than h, replace h with h_{new}.

(4)Add one to the counter.

(5)Repeat (2)(3)(4) until the counter reaches $n \times (n - 1)$.

(6)Output h.

After that, if the fitness of the new habitat is better than the old one, the new one will replace the old habitat to prepare for the next iteration to search for the better results.

E. PERTURBATION METHOD

The pitfall of the iterative further search is easily falling into a local optimum. In order to escape from this drawback and explore new regions of the unexplored search space, the perturbation methods are applied to the current elite habitats which would explore the neighbourhood of the local optima. What's more, selecting an appropriate perturbation is a key operation to change the current habitat. If the perturbation is too strong, it may perform as a restart operator randomly. On the contrary, it has little effect on the further local search to avoid getting trapped in the local optimum.

In HCBBO, the appropriate perturbation referred to *Point_Insert* behaves on each of the elite habitats when the fitness of the current best habitat has not be improved until the counter of reiteration (*MaxReiterationNum*) has reached. The steps of this perturbation method can be described as follows.

(1) Input a habitat h.

(2) Select two positions p_1, p_2 ($p_1 < p_2$) randomly.

(3) Let h_p be the partial permutation between p_1 and p_2 which is not including p_1. Shift the job in position p_1 behind the partial sequence h_p to form a new habitat h_{new}.

(4) Output h_{new}.

F. UPDATE METHOD

In this part, two update frames are shown in Algorithm 7 and Algorithm 8 respectively. In Algorithm 7, the habitats in the

current population and the elite population are used to update the elite population for maintaining the elite habitats to be the first two best habitats in the population to carry out the rest operators. They are sorted as the descending order of fitness and the first two are chosen to generate a new elite population. In addition, the update method for the current population details in Algorithm 8. Based on this algorithm, the current habitat population with some good quality habitats are acquired to get ready for exploring the global optimum in the next iteration.

Algorithm 7 UpdateElitePop()

Input:
The current habitat population H_c.
The elite habitat population *ElitePop*.
Output:
The updated elite population *ElitePop*.

1: $tempElitePop = \emptyset$;
2: **for** $i = 1$ to NE **do**
3: $\quad tempElitePop(i) = H_c(i)$;
4: **end for**
5: **for** $j = 1$ to NE **do**
6: $\quad tempElitePop(NE + j) = ElitePop(j)$;
7: **end for**
8: Sort *tempElitePop* with the descending order of fitness which is also the ascending order of TWT;
9: **for** $i = 1$ to NE **do**
10: $\quad ElitePop(i) = tempElitePop(i)$;
11: **end for**
12: **return** *ElitePop*

Algorithm 8 UpdateCurrentPop()

Input:
The habitat h_e in the elite habitat population *ElitePop*.
Output:
The current population H_c.

1: $TempH_c = H_c$;
2: **while** $TempH_c \neq \emptyset$ **do**
3: $\quad$ Randomly select an individual h from H_c;
4: $\quad$ **if** $TWT(h_e) < TWT(h)$ **then**
5: $\quad\quad h = h_e$; break;
6: $\quad$ **else**
7: $\quad\quad$ remove the solution h in the $TempH_c$;
8: $\quad$ **end if**
9: **end while**
10: **return** H_c

G. CONVERGENCE OF HCBBO

For heuristic algorithms, the diversity and convergence are the main issues to settle NP-hard problems effectively. Except for the advantages inherited from the BBO framework with a balance between the exploration and the exploitation [21], HCBBO also benefits from a further local search on the elite habitats in order to increase the rate of convergence in the exploitation process efficiently. However, if the speed of convergence is too fast to lead the algorithm to a local optimum prematurely, a good perturbation *Point_Insert* is executed to compensate for the shortcomings. Considering the diversity namely the exploration section of HCBBO, the initialization method which is a problem-specific NEH method combined with a chaotic strategy is applied to keep the diversity of the habitat population. From Figure 2 to Figure 11, there is a distinctly view of the convergence trend on the instances from Ta071 to Ta080 respectively in DD_SDST50 when solving SDST-FSSP with the goal of obtaining the minimum total weighted tardiness.

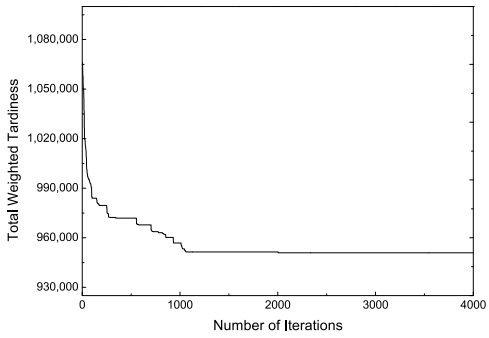

FIGURE 3. The convergence rate of HCBBO on Ta072 with 4000 iterations.

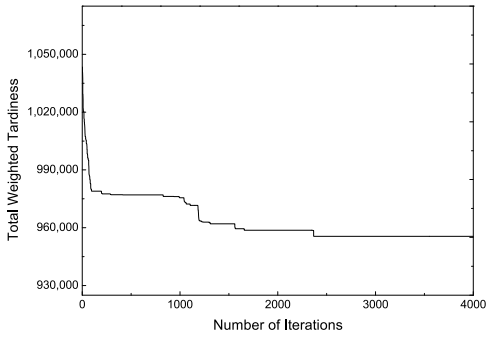

FIGURE 4. The convergence rate of HCBBO on Ta073 with 4000 iterations.

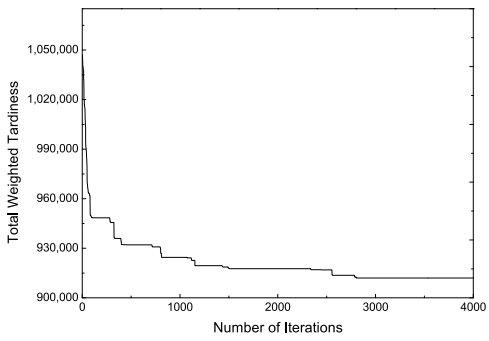

FIGURE 5. The convergence rate of HCBBO on Ta074 with 4000 iterations.

IV. COMPUTATIONAL EXPERIMENT AND RESULTS

A. ENVIRONMENTAL SETUP

HCBBO algorithm is coded in C++ by Microsoft Visual Studio 2013. It is performed on a PC with 3 GB memory

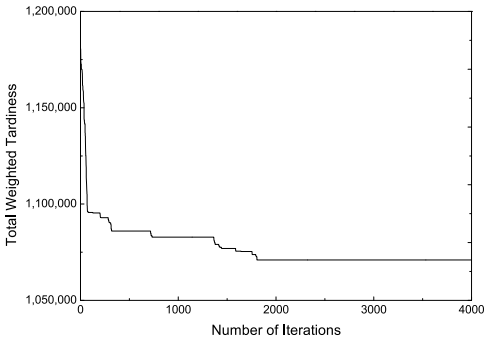

FIGURE 6. The convergence rate of HCBBO on Ta075 with 4000 iterations.

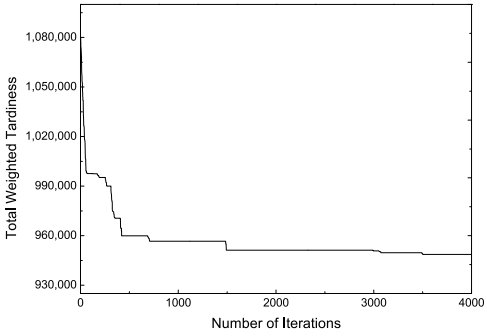

FIGURE 7. The convergence rate of HCBBO on Ta076 with 4000 iterations.

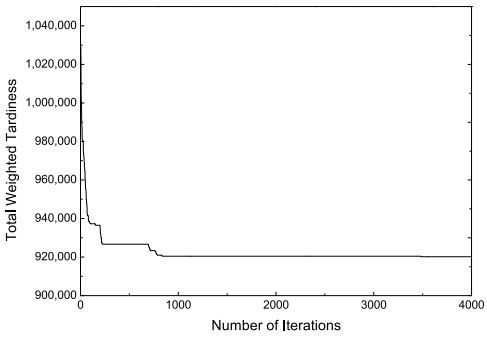

FIGURE 8. The convergence rate of HCBBO on Ta077 with 4000 iterations.

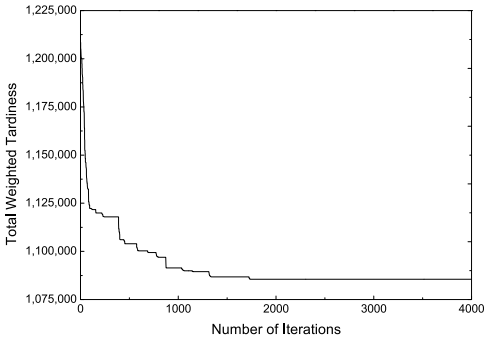

FIGURE 9. The convergence rate of HCBBO on Ta078 with 4000 iterations.

and a CPU of 3.40 GHz on the OS of Windows 7. The stop criterion of this algorithm is CPU time limit given by $(n \times m/2) \times 180$ milliseconds as the same as [8]. In the following

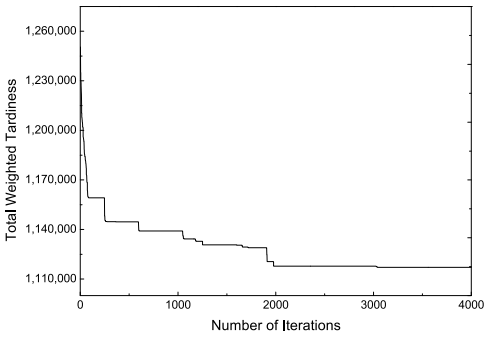

FIGURE 10. The convergence rate of HCBBO on Ta079 with 4000 iterations.

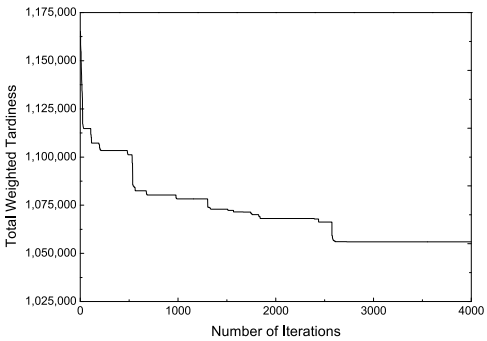

FIGURE 11. The convergence rate of HCBBO on Ta080 with 4000 iterations.

experiments, for each benchmark instance, HCBBO algorithm runs 10 times independently.

B. BENCHMARK PROBLEM INSTANCES AND BENCHMARK ALGORITHMS

The algorithm is applied to four sets of benchmark instances which are based on 120 classical Taillard's original instances for SDST-FSSP in which the processing times ($p_{\pi_i,j}$) are generated from the range [1,99] uniformly. Besides, the sets of the extended instances are grouped by the sequence dependent setup times which are 10%, 50%, 100% and 125% of $p_{\pi_i,j}$. They are considered as DD_SDST10, DD_SDST50, DD_SDST100 and DD_SDST125 that augmented by the due dates and the weight of each job for SDST-FSSP-TWT. In each set, the examples are divided into different combinations of the number of jobs n and the number of machines m including $\{20,50,100\} \times \{5,10,20\}$, $200 \times \{10,20\}$ and 500×20 which are 120 instances in total. The 480 instances in the four sets and the best known solutions can be obtained from http://soa.iti.es/rruiz [8]. To verify the effectiveness of our proposed algorithm, seven effective compared algorithms which contain GA, MA, MA_{LS}, IG_RS, IG_RS_{LS}, SA_PR and HA_IS2 are used. The assessment variable called the relative percentage deviation (RPD) [18] is employed to present the increase percentage of the solution generated from a certain algorithm over the best known bound. For statistics, it is convenient to

apply the average relative percentage deviation (ARPD) as an overall mean shown as below on each instance to evaluate the performance of each algorithm:

$$RPD = \frac{SOME_{sol} - BEST_{sol}}{BEST_{sol}} \times 100(\%) \quad (13)$$

$$ARPD = \sum_{i=1}^{Q} \left(\frac{SOME_{sol} - BEST_{sol}}{BEST_{sol}} \times 100 \right) \times \frac{1}{Q}(\%) \quad (14)$$

where $BEST_{sol}$ is the total weighted tardiness of the best known solution gained from http://soa.iti.es/rruiz for each instance, $SOME_{sol}$ is the total weighted tardiness of the final result obtained by the considered algorithm for the same instance. And Q is the running times of the specific algorithm.

C. EXPERIMENTAL PARAMETER SETTINGS

There are four control parameters in the overall frame of HCBBO to be discussed, including *NP*, *MaxLSNum*, *MaxReiterationNum*, *MaxMP*. It is worth mentioning that the parameters are vital for enhancing the effectiveness of the algorithm. In regard to other parameters used in HCBBO, the number of the elite habitats *NE* is set to 2 as [23] defines and the rest parameters employed in the calculation formulas (8)(10) are the same as BBO [21]. Instances Ta071-Ta080 (100×10) in DD_SDST50 are used to calibrate the parameters as the base cases due to space limitations in the following experiments. Each experiment is ran ten times independently with CPU time equaling to $(n \times m/2) \times 180$. The good results relatively are marked in bold. Besides, the solution better than the best known bound is shown as a negative number.

1) THE INFLUENCE OF THE HABITAT QUANTITY NP

In all the population optimization algorithms, the number of the individuals in the population is an indispensable parameter which exerts significant influence on the efficiency of the algorithm. In HCBBO, it is important to select the appropriate number of individuals to form a habitat population. One issue is that if the number is too small, it is easy for this algorithm to converge prematurely and result in getting trapped in local optima fast. On the other issue, if the size of it is too large, additional and unnecessary computational costs will be produced. In order to get the effect of the habitat quantity *NP* on the search performance directly, *NP* has varied from 10 to 40 with the RPD and ARPD results summing up in Table II on instances from Ta071 to Ta080.

From the results of ARPD shown in Table II, it is obvious to aware that $NP = 20$ gains better RPD in most of the instances including Ta071, Ta074, Ta076 and Ta079 and provides the best ARPD as 1.16. In Ta072, Ta075, Ta078 and Ta080, $NP = 10$ gives less RPD than other settings with an ARPD being 1.31. Furthermore, *NP* of 20 and 40 can produce the same least RPD 1.13 in Ta071. Besides HCBBO with a habitat quantity 30 can gain a better RPD in Ta073 than other sizes. $NP = 40$ presents the least RPD on the rest instances including Ta071 and Ta077 with the ARPD equaling to 1.40.

TABLE 2. ARPD of different numbers of *NP*.

DataNo. \ *NP*	10	**20**	30	40
Ta071	1.67	**1.13**	1.31	**1.13**
Ta072	**0.63**	1.01	1.27	1.27
Ta073	1.99	1.57	**1.32**	2.56
Ta074	1.32	**1.09**	1.44	1.51
Ta075	**0.94**	1.47	1.83	1.75
Ta076	1.79	**1.35**	2.02	1.77
Ta077	1.30	1.19	1.76	**0.87**
Ta078	**0.60**	0.99	0.97	0.68
Ta079	1.81	**0.30**	1.28	0.97
Ta080	**1.00**	1.54	1.03	1.51
Average	1.31	**1.16**	1.42	1.40

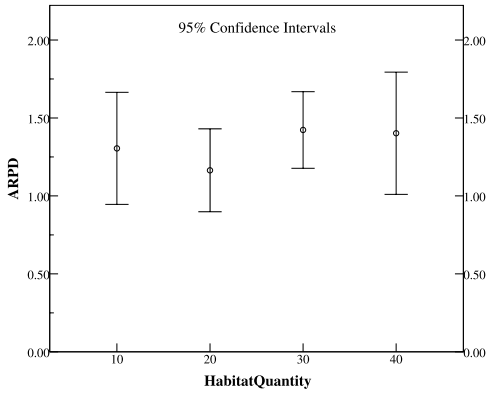

FIGURE 12. 95% CI plot of ARPD for Ta071-Ta080 with different numbers of *NP*.

Figure 12 provides an empirical insight of the effect on each size of *NP* for Ta071-Ta080 using the ARPD evaluation of a confidence interval at the 95% confidence level. As a result it is easily observed that $NP = 20$ outperforms other sizes with the best ARPD. To conclude, the habitat population size *NP* of 20 which results in better performance with the less ARPD than the other settings is the best choice.

2) THE INFLUENCE OF THE MAXIMAL NUMBER OF LOCAL SEARCH MaxLSNum

By this experiment the influence of the maximal number of the insert_based local search *MaxLSNum* of HCBBO is discussed. Through the further exploitation in HCBBO, there exists much better solutions that can be further interfered to the unexplored regions. In order to prove the effect of *MaxLSNum* on HCBBO, we apply the examples from Ta071 to Ta080 on different *MaxLSNum* selecting from the set {15,20,25,30} with the measurement of ARPD. Other parameters are set as the previous experiment temporarily.

What summaries in Table III is the RPD results of each *MaxLSNum*. From Table III, it can be easily found that $MaxLSNum = 15$ achieves best result in Ta073 only and $MaxLSNum = 20$ performs best in Ta074 only. However,

TABLE 3. ARPD of different numbers of *MaxLSNum*.

DataNo. \ MLSN	15	20	**25**	30
Ta071	1.13	1.49	1.29	**0.90**
Ta072	1.01	1.11	**0.27**	1.02
Ta073	**1.57**	1.58	2.03	1.72
Ta074	1.09	**0.37**	1.00	1.52
Ta075	1.47	1.35	**1.13**	1.82
Ta076	1.35	1.30	1.40	**1.08**
Ta077	1.19	1.08	1.20	**0.79**
Ta078	0.99	0.70	0.84	**0.60**
Ta079	0.30	1.00	**-0.20**	1.25
Ta080	1.54	1.64	**0.45**	1.11
Average	1.16	1.16	**0.94**	1.18

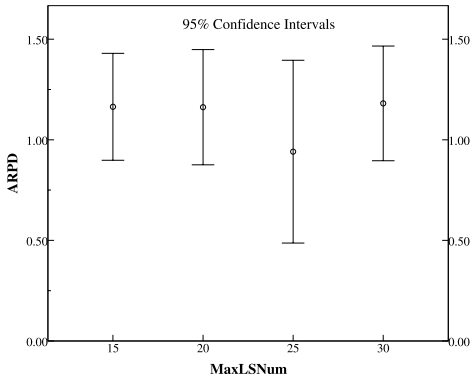

FIGURE 13. 95% CI plot of ARPD for Ta071-Ta080 with different numbers of *MaxLSNum*.

MaxLSNum = 25 can provide better RPD in the instances including Ta072, Ta075, Ta079 and Ta080 with the best ARPD equaling to 0.94. In particular, the negative number -0.20 on Ta079 indicates this result is better than the best known bound of Ta079. Moreover, *MaxLSNum* = 30 gains best RPD results on the rest instances including Ta071, Ta076, Ta077 and Ta078. According to ARPD, *MaxLSNum* = 25 beats other settings on all the ten instances with the ARPD of 0.94.

Figure 13 shown below illustrates the 95% confidence interval of ARPD for different numbers of *MaxLSNum*. It is observed directly that *MaxLSNum* = 25 is adaptive and appropriate in solving SDST-FSSP-TWT with the least ARPD. By means of the above analyses, it is the best choice to set *MaxLSNum* at 25 obviously since *MaxLSNum* = 25 beats other settings in achieving best ARPD of all the ten test instances.

3) THE INFLUENCE OF THE MAXIMAL ITERATION TIMES TO RESTART MaxReiterationNum

In this experiment, the effect of *MaxReiterationNum* that indicates the maximal iteration times to restart a new iteration is considered. By the further local search, this algorithm may be trapped into a local optimum when the fitness of the best habitat is not improved after a certain number of iterations. Therefore, the suitable restart is essential for the algorithm to search in other solution areas effectively in the next iteration.

TABLE 4. ARPD of different numbers of *MaxReiterationNum*.

DataNo. \ MRN	20	25	**30**	35
Ta071	1.54	1.38	1.33	**0.44**
Ta072	1.22	1.09	**0.31**	1.29
Ta073	2.14	2.24	**1.80**	1.95
Ta074	0.93	**0.83**	1.11	1.20
Ta075	1.96	2.09	2.00	**1.17**
Ta076	1.04	**0.30**	0.90	1.51
Ta077	1.45	0.96	**0.89**	1.21
Ta078	1.68	1.10	**0.69**	1.64
Ta079	**1.05**	1.39	1.08	0.78
Ta080	1.02	1.50	**0.63**	1.44
Average	1.40	1.29	**1.07**	1.26

Taking advantage of Ta071-Ta080 evaluated as RPD running for ten times, Table IV illustrates the influence of different *MaxReiterationNum*s which chosen from the set {20,25,30,35}. Other parameter settings are temporarily like the previous experiments.

From Table IV, *MaxReiterationNum* = 20 gives the best RPD on Ta079 only, but for Ta074 and Ta076 *MaxReiterationNum* = 25 performs better RPD than other settings. To most of the instances selected from the set {Ta072, Ta073, Ta077, Ta078, Ta080}, *MaxReiterationNum* = 30 gains the best RPD. Besides *MaxReiterationNum* = 30 achieves the best ARPD 1.07 among all the settings of the ten instances. For the rest instances including Ta071 and Ta075, *MaxReiterationNum* = 35 beats others with the best RPD. It is easily observed that *MaxReiterationNum* = 30 is superior to the other settings on ARPD of all the instances. In addition, Figure 14 as below is shown ARPD with 95% confidence intervals of the instances from Ta071 to Ta080. What can be easily found is that *MaxReiterationNum* = 30 is the best setting for solving SDST-FSSP with the objective of TWT in HCBBO.

4) THE EFFECT OF THE MAXIMUM MUTATION PROBABILITY MaxMP

In the frame of BBO, the maximum mutation probability is a user-defined parameter as m_{max} in formula (8) which is vital for the mutation operator. Moreover, it is a control parameter in the mutation operator to increase the diversity of the habitat population. Selecting an appropriate number of *MaxMP* can not only make the habitat with low HSI mutate to a better habitat but also make the habitat with high HSI have a chance to improve the HSI more than it already has. If *MaxMP* is a small value then the search will be guided to the parallel axis of search space. However, if it is too large, the search space

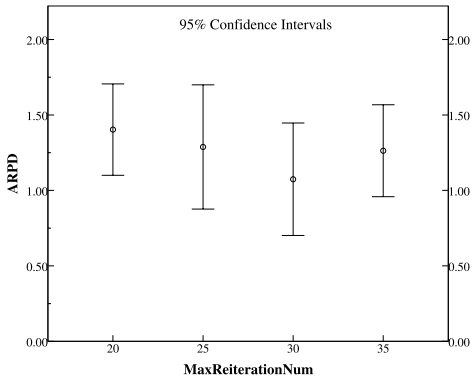

FIGURE 14. 95% CI plot of ARPD for Ta071-Ta080 with different numbers of *MaxReiterationNum*.

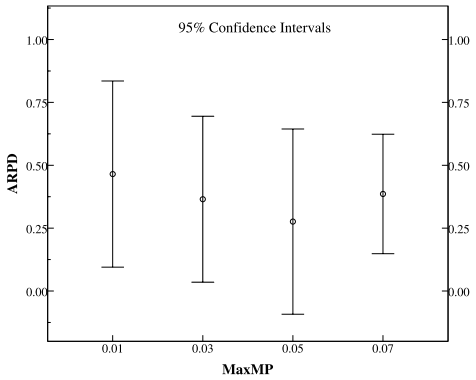

FIGURE 15. 95% CI plot of ARPD for Ta071-Ta080 with different numbers of *MaxMP*.

TABLE 5. ARPD of different numbers of *MaxMP*.

DataNo. \ *MaxMP*	0.01	0.03	**0.05**	0.07
Ta071	0.01	0.17	**-0.39**	0.20
Ta072	0.22	0.31	**-0.24**	-0.03
Ta073	1.49	1.19	1.07	**0.92**
Ta074	**-0.10**	0.09	0.46	0.17
Ta075	**0.43**	0.66	0.69	0.45
Ta076	0.98	0.87	**-0.22**	0.47
Ta077	0.72	**-0.42**	0.55	0.44
Ta078	0.18	0.24	**-0.13**	0.00
Ta079	**-0.05**	0.02	0.12	0.32
Ta080	0.77	**0.52**	0.85	0.92
Average	0.46	0.36	**0.28**	0.39

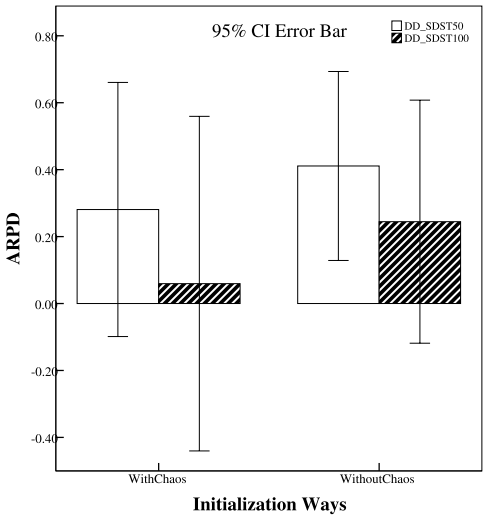

FIGURE 16. 95% CI error bar of ARPD for Ta071-Ta080 with different ways of initialization in DD_SDST50 and DD_SDST100.

is shifted at angles for searching better habitats. Herein as the scheme of HCBBO shows, *MaxMP* chosen from the set {0,01,0.03,0.05,0.07} is discussed. Table V provides the RPD results on Ta071-Ta080 running for ten times with different values of *MaxMP*.

It is demonstrated in Table V that $MaxMP = 0.01$ can achieve best RPD in Ta074, Ta075 and Ta079, while $MaxMP = 0.03$ gains better RPD in instances Ta077 and Ta080. Furthermore, $MaxMP = 0.05$ can provide better results in most of the ten instances including Ta071, Ta072, Ta076 and Ta078 with the least ARPD 0.28. However, $MaxMP = 0.07$ gives the best RPD result on Ta073 only. Figure 15 concludes the ARPD of all the instances with each value of *MaxMP* which is presented in 95% CI. It is obvious to see that $MaxMP = 0.05$ is the best setting with the least ARPD among all the settings. In conclusion, this experiment demonstrates that $MaxMP = 0.05$ is the best setting among all the compared settings for HCBBO.

D. EFFECT OF THE CHAOTIC GENERATOR

Two initialization methods are generated when the problem-specific NEH heuristic is embedded with the chaotic generator and the random generator respectively. These two generators can be shown as formula (10) and (12) refer above. In order to compare the efficiency of them, instances Ta071-Ta080 in DD_SDST50 and DD_SDST100 are used as the based test cases. As Figure 16 illustrates, the initialization way benefiting from the chaotic generator which has less ARPD in both DD_SDST50 and DD_SDST100 is proved to be more effective with the feature of randomicity, ergodicity and regularity in the initialization section of HCBBO.

E. EFFECTIVENESS EVALUATION OF DIFFERENT MUTATION OPERATORS

It is important for selecting an appropriate local search operator which intensifies the search in gaining better results and drives the search into new regions in the mutation method of HCBBO. Except for the efficient insert_based local search, there is another local search method which is the swap_based local search [23] used to compare the efficiency with the inverse_based local search.

Via the experiments on instances Ta071-Ta080 in DD_SDST50 and DD_SDST125, it is easily known from

TABLE 6. ARPD for each algorithm in DD_SDST10 and DD_SDST50.

	Scale	DataName	GA	MA	MA_{LS}	IG_RS	IG_RS_{LS}	SA_PR	HA_IS2	$HCBBO$
DD_SDST10	20×5	Ta001-Ta010	0.92	2.16	0.00	0.00	0.00	3.53	4.61	**0.00**
	20×10	Ta011-Ta020	1.60	8.34	0.00	0.00	0.00	9.90	6.43	**0.00**
	20×20	Ta021-Ta030	1.88	0.00	0.00	0.00	0.00	22.45	8.09	**0.00**
	50×5	Ta031-Ta040	2.83	1.44	1.17	2.70	0.77	5.17	4.56	**0.28**
	50×10	Ta041-Ta050	5.39	1.99	1.63	3.62	1.22	8.13	7.88	**0.23**
	50×20	Ta051-Ta060	10.80	4.28	2.60	8.54	1.51	16.37	20.79	**0.23**
	100×5	Ta061-Ta070	2.83	2.40	1.95	3.52	1.39	4.37	3.04	**0.69**
	100×10	Ta071-Ta080	3.91	2.80	2.50	4.17	2.12	5.53	4.93	**0.58**
	100×20	Ta081-Ta090	6.44	4.23	3.84	5.95	3.41	7.72	8.38	**0.29**
	200×10	Ta091-Ta100	1.94	2.61	1.86	2.29	1.35	3.50	1.35	**0.32**
	200×20	Ta101-Ta110	2.06	4.90	3.47	2.29	2.30	3.44	2.34	**0.06**
	500×20	Ta111-Ta120	0.78	1.37	0.80	0.58	0.66	2.76	**0.11**	1.15
	Average		3.45	3.04	1.65	2.80	1.23	7.74	6.04	**0.32**
DD_SDST50	20×5	Ta001-Ta010	1.45	6.96	0.00	0.02	0.00	5.92	7.20	**0.00**
	20×10	Ta011-Ta020	5.28	13.87	0.09	0.15	0.15	12.58	24.90	**0.00**
	20×20	Ta021-Ta030	0.43	30.78	0.00	0.00	0.00	4.79	1.68	**0.00**
	50×5	Ta031-Ta040	5.42	2.19	2.00	3.77	1.79	7.59	7.31	**0.54**
	50×10	Ta041-Ta050	5.83	2.02	2.24	3.79	2.41	8.42	9.13	**0.54**
	50×20	Ta051-Ta060	6.75	2.53	2.08	4.69	2.80	10.95	12.97	**0.20**
	100×5	Ta061-Ta070	4.47	3.14	2.66	4.82	2.14	6.70	4.19	**0.61**
	100×10	Ta071-Ta080	4.65	3.09	2.61	4.59	2.51	6.65	6.25	**0.28**
	100×20	Ta081-Ta090	5.42	3.39	3.15	5.22	3.29	7.12	6.06	**0.49**
	200×10	Ta091-Ta100	1.95	2.56	1.71	2.37	1.07	3.65	1.12	**-0.28**
	200×20	Ta101-Ta110	1.61	4.10	2.70	1.95	1.61	3.10	1.51	**-0.73**
	500×20	Ta111-Ta120	1.43	1.93	1.42	1.04	0.95	3.76	**0.00**	0.56
	Average		3.72	6.38	1.72	2.70	1.56	6.77	6.86	**0.18**

TABLE 7. ARPD for each algorithm in DD_SDST100 and DD_SDST125.

	Scale	DataName	GA	MA	MA_{LS}	IG_RS	IG_RS_{LS}	SA_PR	HA_IS2	$HCBBO$
DD_SDST100	20×5	Ta001-Ta010	3.99	2.89	0.07	0.07	0.40	8.24	9.77	**0.00**
	20×10	Ta011-Ta020	2.67	6.53	0.09	0.11	0.10	13.21	18.28	**0.00**
	20×20	Ta021-Ta030	2.80	2.49	0.00	0.00	0.00	18.17	40.94	**0.00**
	50×5	Ta031-Ta040	7.57	3.89	3.38	5.83	3.46	11.46	10.75	**1.07**
	50×10	Ta041-Ta050	7.80	3.23	3.00	5.34	2.93	10.77	9.76	**0.61**
	50×20	Ta051-Ta060	8.08	2.78	2.57	5.06	2.36	11.34	13.15	**0.28**
	100×5	Ta061-Ta070	5.22	3.71	3.24	5.91	2.76	8.40	5.44	**0.16**
	100×10	Ta071-Ta080	4.76	3.52	2.80	5.45	2.57	7.58	5.17	**0.06**
	100×20	Ta081-Ta090	5.35	3.39	3.13	5.27	3.29	7.72	6.43	**-0.01**
	200×10	Ta091-Ta100	2.60	3.55	2.34	2.60	1.27	4.78	1.63	**-0.42**
	200×20	Ta101-Ta110	1.96	4.42	3.20	2.33	1.60	3.77	1.61	**-0.43**
	500×20	Ta111-Ta120	2.38	2.89	2.37	2.26	0.63	6.38	**0.14**	1.81
	Average		4.60	3.61	2.18	3.35	1.78	9.32	10.26	**0.26**
DD_SDST125	20×5	Ta001-Ta010	3.68	5.54	0.41	0.15	0.40	10.22	14.82	**0.00**
	20×10	Ta011-Ta020	3.91	6.28	0.10	0.00	0.04	10.80	15.12	**0.00**
	20×20	Ta021-Ta030	3.01	11.57	0.00	0.00	0.00	42.38	14.08	**0.00**
	50×5	Ta031-Ta040	8.36	4.07	3.75	7.20	3.62	12.67	11.43	**1.73**
	50×10	Ta041-Ta050	8.01	3.63	3.49	6.25	3.79	11.16	10.50	**0.98**
	50×20	Ta051-Ta060	7.83	2.63	2.75	5.56	2.74	11.45	10.70	**0.81**
	100×5	Ta061-Ta070	5.01	3.73	2.87	5.58	2.01	9.27	4.78	**0.06**
	100×10	Ta071-Ta080	4.69	3.38	2.81	5.30	2.52	7.88	5.35	**0.42**
	100×20	Ta081-Ta090	4.68	3.07	2.61	4.64	2.79	6.97	5.97	**-0.05**
	200×10	Ta091-Ta100	3.10	4.38	2.97	3.46	0.99	6.00	1.23	**0.41**
	200×20	Ta101-Ta110	2.08	3.48	3.10	1.84	1.17	4.10	1.35	**-0.44**
	500×20	Ta111-Ta120	3.20	3.86	3.16	3.13	0.45	7.43	**0.32**	2.22
	Average		4.80	4.63	2.34	3.59	1.71	11.69	7.97	**0.51**

Figure 17 that the inverse_based local search has more powerful efficiency than the swap_based local search in the mutation section of HCBBO for achieving less ARPD.

F. EFFECT OF DIFFERENT PERTURBATION OPERATORS

Different perturbations have different influence on the solutions gained by the further local search. A random move on the neighborhoods of the elite habitats may lead to a satisfactory result. Besides, taking into account the property of SDST-FSSP and matching the specified local search can achieve much better results. Therefore the detail schemes of three simple but frequently-used perturbation operators, including *2Point_Exchange*, *Segment_Insert* and *Self_Inverse* operators [23], [30], [31] are employed to

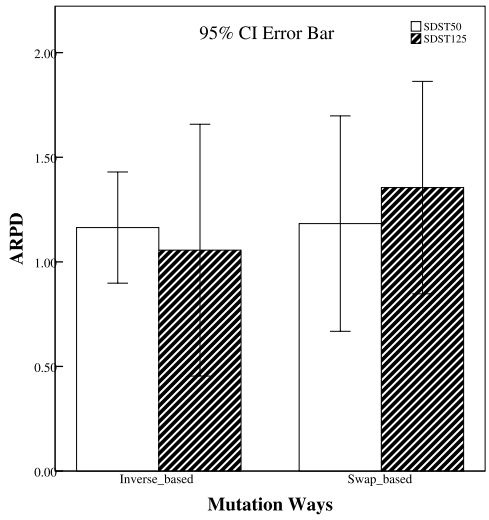

FIGURE 17. 95% CI error bar of ARPD for Ta071-Ta080 with different ways of mutation in DD_SDST50 and DD_SDST125.

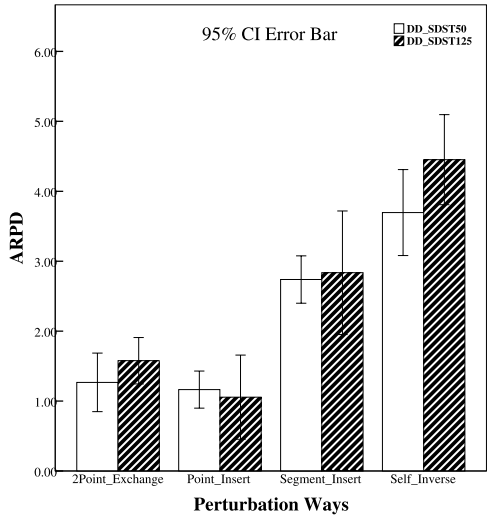

FIGURE 18. 95% CI error bar of ARPD for Ta071-Ta080 with different ways of perturbation in DD_SDST50 and DD_SDST125.

compare with the used operator *Point_Insert* in HCBBO. In addition, instances Ta071-Ta080 in DD_SDST50 and DD_SDST125 are similarly chosen for verifying the efficiency of *Point_Insert* perturbation by computational experiments. As a result, from Figure 18 *Point_Insert* perturbation with the least ARPD is the best operator fitting with the insert_based local search among all the compared perturbation methods.

G. COMPARISON RESULTS WITH SOME STATE-OF-THE-ART APPROACHES

This subsection analyses the comparison results between HCBBO with the best algorithm settings obtained by previous experiments and seven outstanding algorithms contain GA, MA, MA_{LS}, IG_RS, IG_RS_{LS}, SA_PR and HA_IS2. For ease of reference, HCBBO runs for the same computational time with the CPU time executing on experiments in [8]. For the seven compared algorithms, it has concluded that IG_RS_{LS} algorithms beats other six algorithms with higher performance [8]. And the details of comparative results are presented in the following tables and figures. Moreover, we calculate RPD for each instance of SDST-FSSP in order to

TABLE 8. NewBounds for instances in DD_SDST10.

	Instances	NewBound	Best-knownBound	Difference
DD_SDST10	50×10			
	Ta047	116710	116773	-63
	Ta048	145428	145985	-557
	100×20			
	Ta082	487232	488290	-1058
	Ta090	511122	516771	-5649
	200×10			
	Ta091	3679743	3679800	-57
	Ta093	3490573	3492533	-1960
	Ta097	3993343	3995468	-2125
	200×20			
	Ta102	3350143	3351969	-1826
	Ta103	3248533	3253371	-4838
	Ta104	3265568	3265717	-149
	Ta105	3395552	3405413	-9861
	Ta106	3168099	3182036	-13937

TABLE 9. NewBounds for instances in DD_SDST50.

	Instances	NewBound	Best-knownBound	Difference
DD_SDST50	100×5			
	Ta062	980012	981348	-1336
	100×10			
	Ta072	938927	940299	-1372
	Ta078	1076355	1081217	-4862
	Ta079	1107072	1109912	-2840
	100×20			
	Ta081	729123	729978	-855
	Ta083	911531	912617	-1086
	Ta085	894966	898116	-3150
	200×10			
	Ta091	4681565	4695272	-13707
	Ta092	4944803	4946439	-1636
	Ta093	4848174	4887054	-38880
	Ta094	4518497	4534210	-15713
	Ta095	5133525	5156724	-23199
	Ta098	4765380	4797137	-31757
	Ta100	4811607	4852486	-40879
	200×20			
	Ta101	4727534	4781801	-54267
	Ta102	4408960	4426225	-17265
	Ta103	3948489	3951252	-2763
	Ta104	4952967	4991684	-38717
	Ta105	5196391	5239776	-43385
	Ta106	4790760	4817883	-27123
	Ta107	5249674	5319119	-69445
	Ta108	5219755	5272903	-53148
	Ta109	4289223	4320966	-31743
	Ta110	5051256	5077798	-26542
	500×20			
	Ta113	33437269	33470508	-33239

TABLE 10. NewBounds for instances in DD_SDST100.

	Instances	NewBound	Best-knownBound	Difference
DD_SDST100	50×10			
	Ta050	281097	283260	-2163
	50×20			
	Ta052	122242	122375	-133
	100×5			
	Ta061	1305248	1307077	-1829
	Ta062	1546738	1553024	-6286
	Ta064	1210883	1216076	-5193
	Ta066	1330462	1331983	-1521
	Ta069	1289524	1292359	-2835
	Ta070	1303042	1308470	-5428
	100×10			
	Ta071	1403664	1415331	-11667
	Ta075	1444882	1449131	-4249
	Ta076	1589092	1596542	-7450
	Ta079	1561761	1564155	-2394
	Ta080	1374184	1386611	-12427
	100×20			
	Ta081	1328439	1334768	-6329
	Ta083	1420458	1432103	-11645
	Ta085	1196669	1211298	-14629
	Ta086	1197505	1199265	-1760
	Ta088	1280081	1284208	-4127
	Ta089	1290832	1291136	-304
	200×10			
	Ta091	6455575	6514797	-59222
	Ta092	6463677	6469831	-6154
	Ta093	6179320	6191207	-11887
	Ta094	5878596	5957765	-79169
	Ta095	5865950	5867475	-1525
	Ta096	6075277	6103884	-28607
	Ta097	6359018	6397552	-38534
	Ta098	6397816	6430234	-32418
	Ta099	5987555	6023261	-35706
	200×20			
	Ta101	6609236	6680259	-71023
	Ta103	5767075	5795394	-28319
	Ta104	6295370	6346074	-50704
	Ta105	6396909	6426679	-29770
	Ta107	6170062	6205583	-35521
	Ta108	6538078	6636735	-98657

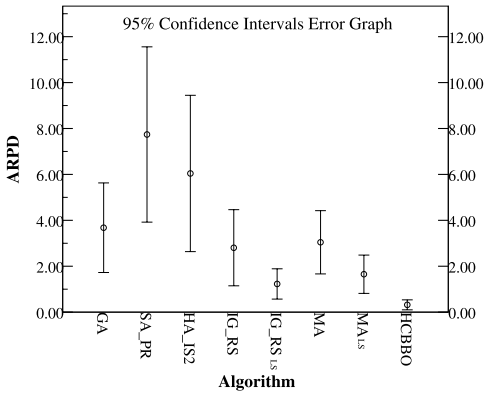

FIGURE 19. ARPD in 95% CI for each algorithm in DD_SDST10.

compare the performance of each algorithm. Owing to the space limitation, only the comparative ARPD results of every data set are shown in Table VI and Table VII in detail

TABLE 11. NewBounds for instances in DD_SDST125.

	Instances	NewBound	Best-knownBound	Difference
DD_SDST125	50×10			
	Ta042	332549	333688	-1139
	50×20			
	Ta053	200318	200350	-32
	100×5			
	Ta061	1387901	1391094	-3193
	Ta062	1356350	1359669	-3319
	Ta063	1487292	1490691	-3399
	Ta065	1316366	1326726	-10360
	Ta066	1634501	1645175	-10674
	100×10			
	Ta074	1616450	1625362	-8912
	100×20			
	Ta081	1331201	1341244	-10043
	Ta085	1588194	1592282	-4088
	Ta086	1357130	1360487	-3357
	Ta088	1923868	1932614	-8746
	Ta089	1691956	1709193	-17237
	Ta090	1629982	1643533	-13551
	200×10			
	Ta095	6629803	6648769	-18966
	200×20			
	Ta101	7626412	7650844	-24432
	Ta102	7238555	7267353	-28798
	Ta103	7641169	7724161	-82992
	Ta104	7722746	7785086	-62340
	Ta105	7331725	7335148	-3423
	Ta107	7339743	7376093	-36350
	Ta108	8399891	8412652	-12761
	Ta109	7445244	7513506	-68262
	Ta110	6813571	6835923	-22352

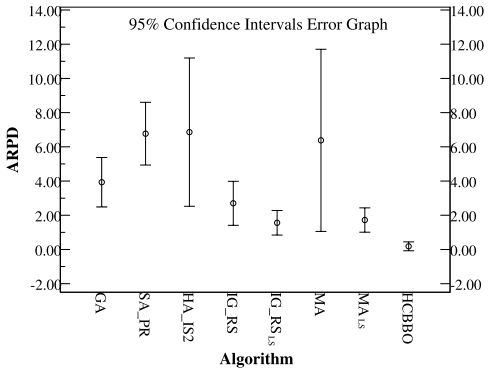

FIGURE 20. ARPD in 95% CI for each algorithm in DD_SDST50.

classified by the type and size of instances. The better results in tables are marked in bold. In particular, if the solution is better than the best-known bound statistically, ARPD is shown as a negative number. From Table VI, it implies that HCBBO beats other algorithms obviously with the least overall ARPD 0.32 and 0.18 in each scale of DD_SDST10 and DD_SDST50 respectively except for the large scale of 500×20. Due to the ability to obtain good results of HCBBO, it can be found in Table VII

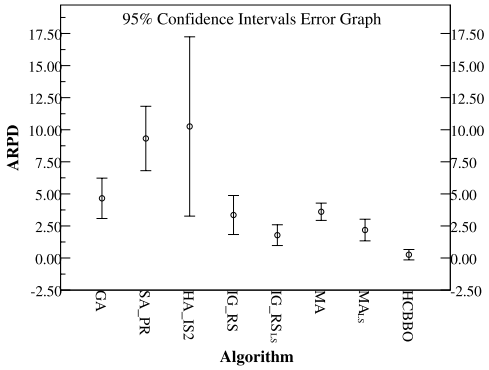

FIGURE 21. ARPD in 95% CI for each algorithm in DD_SDST100.

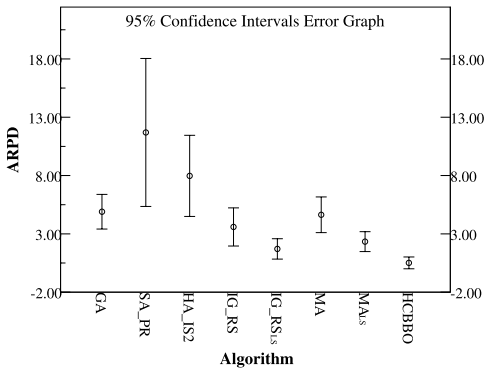

FIGURE 22. ARPD in 95% CI for each algorithm in DD_SDST125.

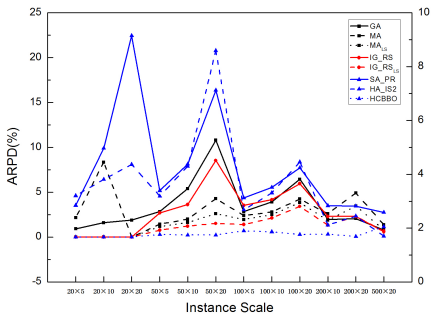

FIGURE 23. ARPD of each algorithm for all instances in DD_SDST10.

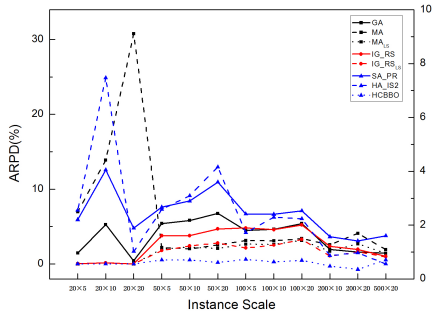

FIGURE 24. ARPD of each algorithm for all instances in DD_SDST50.

that HCBBO is superior to other algorithms with the least ARPD 0.26 and 0.51 respectively on all the instances in DD_SDST100 and DD_SDST125. Regarding ARPD on each scale of DD_SDST100 and DD_SDST125 despite of the

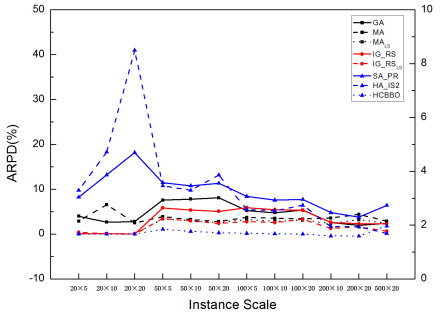

FIGURE 25. ARPD of each algorithm for all instances in DD_SDST100.

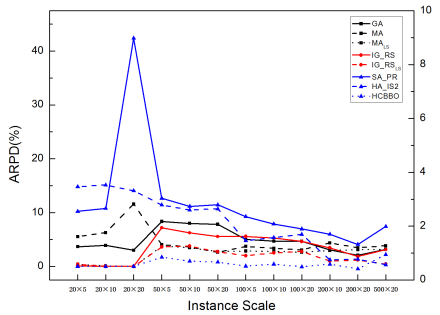

FIGURE 26. ARPD of each algorithm for all instances in DD_SDST125.

scale 500×20, HCBBO can provide best ARPD among all the algorithms. It is necessary to note that for obtaining solutions with better quality in the large scale 500×20, HA_IS2 provides better solutions than HCBBO mainly due to the slow initialization of the proposed algorithm. However from the overall mean of ARPD on all the instances, HCBBO is the best algorithm in getting solutions with high quality.

Besides, from Figure 19 to Figure 22, it is evident that HCBBO has the best performance with the highest robustness represented by the shortest and lowest error bar among all the compared algorithms using ARPD at the 95% confidence level of confidence intervals. Hence, via the figures it is concluded that HCBBO excels in the aspect of robustness among all the compared algorithms on each type of all instances in DD_SDST10, DD_SDST50, DD_SDST100 and DD_SDST125 of SDST-FSSP. To get a clear insight into the performance of HCBBO, Figure 23,24,25,26 are drawn to demonstrate the rank of each algorithm. It can be obviously shown that HCBBO is the best algorithm on a comprehensive consideration of both quality and robustness.

For the illustration purpose, from Table VIII to Table XI they present the new bounds yielded by HCBBO better than the best-known ones and the difference between them in each data set to prove the high performance of HCBBO empirically. Based on these results, it is summarized that HCBBO surpasses by other seven algorithms evidently resulting in good solutions with high quality and robustness.

V. CONCLUSION AND FUTURE WORK

In this paper, the flowshop scheduling problem with sequence dependent setup times is addressed for minimizing the total weighted tardiness. We have explained the issues about the overall framework, the strategies between the exploration and

exploitation and the problem-specific tunings in the above sections. One of the key advantages of HCBBO is the initialization method which is a problem-specific NEH embedded with a chaotic sequence generator. Besides, in this proposed method it is important for the largest-order-value rule to make HCBBO suitable to this scheduling problem. It has to be mentioned that there are other reasons why HCBBO has high efficiency and robustness. First, the improved migration and mutation operators referred above are developed to keep the diversity and quality of the population to a certain extent. Second, in order to intensify the quality of solutions, a further local search namely the insert_based local search is applied to the elite population for exploiting the improved solutions. Third, due to some cases that the deep search would lead the solutions to be trapped in the local optima, an effective perturbation *Point_Insert* is employed for accelerating the solutions to overstep the local extremum and pursuing for the better global optimum. At last, the current population is updated by the specific method duly to be intended for the next iteration of HCBBO. By computational experiments on the benchmark instances, it is concluded that HCBBO is a robust and highly efficient algorithm for solving SDST-FSSP to minimize weighted tardiness which is superior to other compared effective and prominent algorithms.

Looking forward to increase the future research, HCBBO is expected to be extended to solve other kinds of combinatorial optimization problems. Such as the blocking flowshop scheduling problem (BFSSP), the no-idle flowshop scheduling problem (NIFSSP) and the multi-objective flowshop scheduling problems which is much closer to real problems. In addition, we need to do more work to understand the interactions between the intensification and the diversification strategies explicitly in HCBBO for enhancing the performance of HCBBO.

Acknowledgment

The authors would like to thank all anonymous reviewers for their constructive comments, which have helped improve the study in numerous ways.

REFERENCES

[1] R. Ruiz and C. Maroto, "A comprehensive review and evaluation of permutation flowshop heuristics," *Eur. J. Oper. Res.*, vol. 40, no. 2, pp. 479–494, 2005.

[2] J. N. D. Gupta, "A functional heuristic algorithm for the flowshop scheduling problem," *J. Oper. Res. Soc.*, vol. 22, no. 1, pp. 39–47, 1971.

[3] M. C. Bonney and S. W. Gundry, "Solutions to the constrained flowshop sequencing problem," *J. Oper. Res. Soc.*, vol. 27, no. 4, pp. 869–883, 1976.

[4] M. Nawaz, E. E. Enscore, Jr., and I. Ham, "A heuristic algorithm for the m -machine, n -job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983. [Online]. Available: https://scholar.google.com/scholar?hl=zh-CN&as_sdt=0%2C5&q=A+heuristic+algorithm+for+the+m-machine%2Cn-job+flow-shop+sequencing+problem%2C%27%27&btnG=

[5] S. M. A. Suliman, "A two-phase heuristic approach to the permutation flow-shop scheduling problem," *Int. J. Prod. Econ.*, vol. 64, nos. 1–3, pp. 143–152, 2000.

[6] R. K. Congram, C. N. Potts, S. L. Van, and D. Velde, "An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem," *Inf. J. Comput.*, vol. 14, no. 1, pp. 52–67, 2002.

[7] S. A. Mansouri, S. H. Hendizadeh, and N. Salmasi, "Bicriteria scheduling of a two-machine flowshop with sequence-dependent setup times," *Int. J. Adv. Manuf. Technol.*, vol. 40, nos. 11–12, pp. 1216–1226, 2009.

[8] R. Ruizab and T. Stützle, "An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 1143–1159, 2008.

[9] R. Z. Ríos-Mercado and J. F. Bard, "A branch-and-bound algorithm for permutation flow shops with sequence-dependent setup times," *IIE Trans.*, vol. 31, no. 8, pp. 721–731, 1999.

[10] B. N. Srikar and S. Ghosh, "A MILP model for the n-job, M-stage flowshop with sequence dependent set-up times," *Int. J. Prod. Res.*, vol. 24, no. 6, pp. 1459–1474, 1986.

[11] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Res. Logistics Quart.*, vol. 1, no. 1, pp. 61–68, 1954.

[12] H. G. Campbell, R. A. Dudek, and M. L. Smith, "A heuristic algorithm for the n Job, m machine sequencing problem," *INFORMS*, vol. 16, no. 10, pp. B-630–B-637, 1970.

[13] R. Ruizab, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *Eur. J. Oper. Res.*, vol. 177, no. 3, pp. 2033–2049, 2007.

[14] C. Rajendran and H. Ziegler, "Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times," *Eur. J. Oper. Res.*, vol. 149, no. 3, pp. 513–522, 2003.

[15] C. Rajendran and H. Ziegler, "A heuristic for scheduling to minimize the sum of weighted flowtime of jobs in a flowshop with sequence-dependent setup times of jobs," *Comput. Ind. Eng.*, vol. 33, nos. 1–2, pp. 281–284, 1997.

[16] R. Ruiz, C. Maroto, and J. Alcaraz, "Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics," *Eur. J. Oper. Res.*, vol. 165, no. 1, pp. 34–54, 2005.

[17] R. Z. Ríos-Mercado and J. F. Bard, "An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup times," *J. Heurist.*, vol. 5, no. 1, pp. 53–70, 1999.

[18] R. Ruiz, C. Maroto, and J. Alcaraz, "Two new robust genetic algorithms for the flowshop scheduling problem," *Omega*, vol. 34, no. 5, pp. 461–476, 2006.

[19] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 1061–1071, 1996.

[20] S. Hasija and C. Rajendran, "Scheduling in flowshops to minimize total tardiness of jobs," *Int. J. Prod. Res.*, vol. 42, no. 11, pp. 2289–2301, 2004.

[21] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.

[22] J. Lin, "A hybrid discrete biogeography-based optimization for the permutation flow shop scheduling problem," *Int. J. Prod. Res.*, vol. 54, no. 16, pp. 1–10, 2017.

[23] M. Yin and X. Li, "A hybrid bio-geography based optimization for permutation flow shop scheduling," *Sci. Res. Essays*, vol. 6, no. 10, pp. 2078–2100, 2011.

[24] B. Qian, L. Wang, R. Hu, W. L. Wang, D. X. Huang, and X. Wang, "A hybrid differential evolution method for permutation flow-shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 38, nos. 7–8, pp. 757–777, 2008.

[25] L. Wang, Q. K. Pan, and M. F. Tasgetiren, "A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem," *Comput. Ind. Eng.*, vol. 61, no. 1, pp. 76–83, 2011.

[26] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, no. 5560, p. 459, 1976.

[27] B. Alatas, E. Akin, and A. B. Ozer, "Chaos embedded particle swarm optimization algorithms," *Chaos Solitons Fractals*, vol. 40, no. 4, pp. 1715–1734, 2009.

[28] J. Wang, Z. Wu, and H. Wang, "Hybrid differential evolution algorithm with chaos and generalized opposition-based learning," in *Proc. Int. Symp. Intell. Comput. Appl.*, 2010, pp. 103–111.

[29] L. Wang, D. Z. Zheng, and Q.-S. Lin, "Survey on chaotic optimization methods," *Comput. Technol. Autom.*, vol. 1, pp. 1–5, Jan. 2001.

[30] X. Li and M. Li, "Multiobjective local search algorithm-based decomposition for multiobjective permutation flow shop scheduling problem," *IEEE Trans. Eng. Manag.*, vol. 62, no. 4, pp. 544–557, Apr. 2015.
[31] X. Li and M. Yin, "A discrete artificial bee colony algorithm with composite mutation strategies for permutation flow shop scheduling problem," *Sci. Iranica*, vol. 19, no. 6, pp. 1921–1935, 2012.
[32] X. Li and M. Yin, "A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 16, pp. 4732–4754, 2013.
[33] X. Li and M. Yin, "An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure," *Adv. Eng. Softw.*, vol. 55, no. 8, pp. 10–31, 2013.
[34] X. Li and S. Ma, "Multi-objective memetic search algorithm for multi-objective permutation flow shop scheduling problem," *IEEE Access*, vol. 4, pp. 2154–2165, 2017.
[35] C. Fang and L. Wang, "An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem," *Inf. Sci.*, vol. 181, no. 20, pp. 4804–4822, 2011.
[36] Q.-K. Pan and L. Wang, "Effective heuristics for the blocking flowshop scheduling problem with makespan minimization," *Omega*, vol. 40, no. 2, pp. 218–229, 2012.
[37] X. Li and S. Ma, "Multiobjective discrete artificial bee colony algorithm for multiobjective permutation flow shop scheduling problem with sequence dependent setup times," *IEEE Trans. Eng. Manag.*, vol. 64, no. 2, pp. 149–165, May 2017.

YUNHE WANG is a currently pursuing the Ph.D. degree with the School of Information Science and Technology, Northeast Normal University, Changchun, China. Her current research interests include intelligent computation and machine learning.

XIANGTAO LI (M'15) received the B.Eng., M.Eng., and Ph.D. degrees in computer science from Northeast Normal University, Changchun, China, in 2009, 2012, and 2015, respectively. He is currently an Associate Professor with the Department of Information Science and Technology, Northeast Normal University. He has authored over 40 research papers. His research interests include intelligent computation, evolutionary data mining, constrained optimization, multi-objective optimization, and their applications.