Received April 15, 2019, accepted May 12, 2019, date of publication May 16, 2019, date of current version June 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2917273

Solving Distributed Hybrid Flowshop Scheduling Problems by a Hybrid Brain Storm Optimization Algorithm

JIAN-HUA HAO[1], JUN-QING LI[1,2], YU DU[1], MEI-XIAN SONG[2], PENG DUAN[2], AND YING-YU ZHANG[2]

[1]School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
[2]School of Computer, Liaocheng University, Liaocheng 252059, China

Corresponding author: Jun-Qing Li (lijunqing@lcu-cs.com)

This work was supported in part by the National Science Foundation of China under Grant 61773192 and Grant 61803192, in part by the Shandong Province Higher Educational Science and Technology Program under Grant J17KZ005, in part by the State Key Laboratory of Synthetical Automation for Process Industries under Grant PAL-N201602, and in part by the Major Basic Research Projects in Shandong under Grant ZR2018ZB0419.

ABSTRACT With the trend of manufacturing globalization, distributed production has attracted wide attention from the industry and academia. Nevertheless, there has been little research on the distributed hybrid flowshop scheduling (DHFS) problem. To make up for the gap, this study aims to solve the DHFS problem, in which multiple factories with hybrid flowshop scheduling (HFS) problems are considered. This problem consists of two subproblems: 1) how to choose a factory for each job and 2) how to schedule all jobs within the assigned factories. To solve the DHFS problem, a mathematical model is formulated. Then, inspired by successful applications of brain storm optimization (BSO) algorithm in different fields, we try to solve the DHFS with a hybrid BSO (HBSO). In the proposed algorithm, firstly, a new approach to calculate the distance in the procedure of clustering is embedded. Then, a novel constructive heuristic based on the Nawaz–Enscore–Ham (NEH) method, called distributed NEH, is proposed. Moreover, an improved crossover operator based on the partial-mapped crossover (PMX) is designed for the distributed scheduling problem. Finally, the 20 large-scale instances based on the realistic production data are randomly generated to test the performance of the proposed algorithm. The experimental results verify that the proposed algorithm is efficient and effective for solving the considered DHFS problems in comparison with the other recently published efficient algorithms.

INDEX TERMS Distributed hybrid flowshop, hybrid brain storm optimization, distributed Nawaz-Enscore-Ham, K-means method.

I. INTRODUCTION

Production scheduling is one of the important problems which should be settled in the production management system of enterprises. The hybrid flow shop scheduling (HFS) problem has attracted the attention of many researchers since Salvador's innovative paper in the 1970's [1]. In the traditional hybrid flow shop scheduling problem, a series of jobs is processed by only one factory, which consists of a set of production stages and at least one stage has multiple identical machines. HFS system is widely applied in actual production, e.g., in glass-making, textile industries, paper and steel production. Therefore, a large number of researchers have conducted in-depth studies during the last decade and proposed many methods to solve the HFS problem such as the exact solution, heuristics and metaheuristics [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Azwirman Gusrialdi.

Nowadays, with the trend of manufacturing globalization, companies have already started to employ distributed production management system to reduce manufacture costs and mitigate management risks [3]. Unlike the classical flow shop scheduling problem within a single factory, the distributed flowshop scheduling problem (DFSP) with multiple factories has attracted attention by researchers in the last few years. Many different types of DFSP have arisen from actual manufacturing activities, such as distributed permutation flow shop scheduling problems (DPFSP) [4], distributed assembly permutation flowshop scheduling problems (DAPFSP) [5], distributed flexible job-shop scheduling

TABLE 1. The processing times of six jobs at each stage.

Stages	Jobs					
	1	2	3	4	5	6
1	2	3	6	5	8	6
2	8	3	3	2	4	7

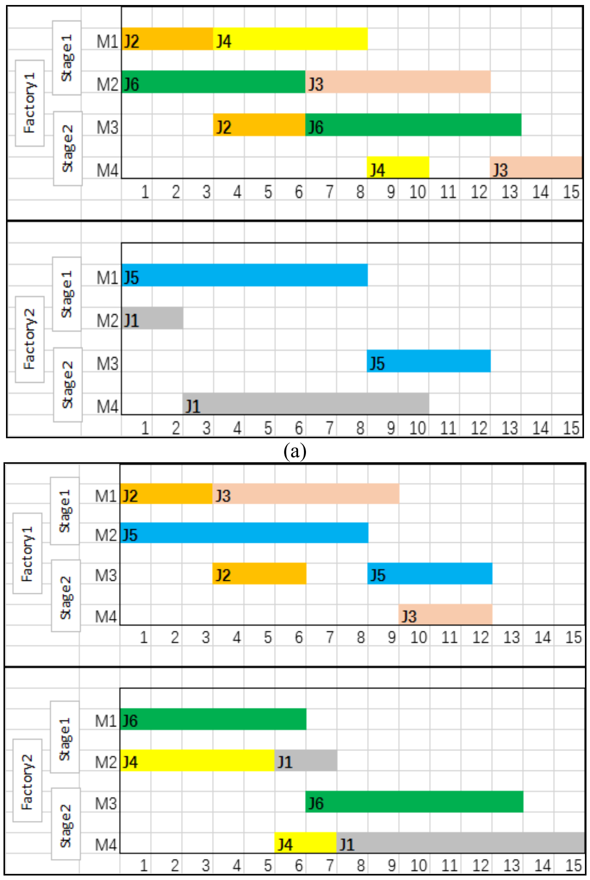

FIGURE 1. Gantt chart of the example. (a) Gantt chart of the example generated by rule 1. (b) Gantt chart of the example generated by Rule 2.

problems (DFJSP) [6], distributed no-wait flowshop scheduling problems (DNFSP) [7], distributed hybrid flowshop scheduling problem with multiprocessor tasks (DHFSPMT) [8], and distributed reentrant permutation flow shop scheduling problems (DRPFS) [9].

Due to the complexity and practicality of the DFSP, a number of literatures have tried to propose multiple models and algorithms to tackle them. Naderi and Ruiz characterized distributed permutation flow shop scheduling (DPFS) problem and proposed six different alternative mixed integer linear programming (MILP) models [4]. For the DPFS with makespan criterion, Gao and Chen proved that the tabu search (TS) and a hybrid GA_LS are efficient algorithms [10], [11]. Wang *et al.* developed an effective estimation of distribution algorithm (EDA) with some local search operators based on DPFSP characteristics [12]. Naderi and Ruiz presented a scatter search (SS) method for DPFSP to optimize the makespan [13]. Viagas *et al.* introduced a bounded-search iterated greedy algorithm for

Algorithm 1: Canonical BSO

```
1.  Randomly generate N individuals (solutions)
2.  Evaluate the fitness of N individuals
3.  Iteration = 1
4.  While Iteration <= max_iterations do
5.      Cluster N individuals into $K$ clusters with the $K$-means method
6.      Select the best individual in each cluster as the cluster center
7.      If random (0,1) < $P_{replace}$ then
8.          Randomly select a cluster center and generate $X_{new}$ to replace the selected cluster center
9.      End if
10.     While i < N do do
11.         If random (0,1) < $P_{one}$ then
12.             Randomly select a cluster with probability $P_r$
13.             If random (0,1) < $P_{oc}$ then
14.                 Select the cluster center and add random values to generate $X_{new}$
15.             Else
16.                 Randomly select an individual from this cluster and add random values to generate $X_{new}$
17.             End if
18.         Else
19.             Randomly select two clusters
20.             If random (0,1) < $P_{tc}$ then
21.                 Combine the two cluster centers and add random values to generate $X_{new}$
22.             Else
23.                 Randomly select two individuals from two clusters and add random values to generate $X_{new}$
24.             End if
25.         End if
26.         If $X_{new}$ is better than $X_{selected}$ then
27.             Replace $X_{selected}$ with $X_{new}$
28.         End if
29.     End while
30. End while
31. Output the best individual
```

the DPFSP and proposed eighteen constructive heuristics to obtain high-quality solutions [14], [15]. A novel chemical Reaction Optimization (CRO) is proposed lately by Bargaoui *et al.* to solve DPFSP with makespan criterion [16]. More recently, Ruiz *et al.* provided a simple Iterated Greedy (IG) algorithms to solve the DPFSP, in which modified initialization, construction and destruction methods, associate with a novel local search are presented [17]. For distributed flexible job-shop scheduling problems (DFJSP), Wu *et al.* briefly reviewed the impact of diverse chromosome representations and proposed an

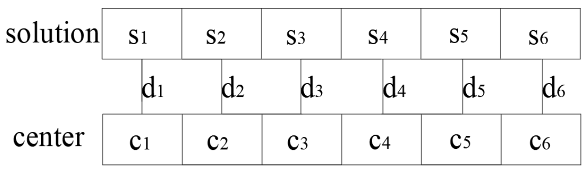

FIGURE 2. The distance of each dimension.

improved genetic algorithm [18]. Lin and Zhang worked out distributed assembly permutation flowshop scheduling problems (DAPFSP) with a hybrid biography-based optimization (HBBO) algorithm [19]. Lin *et al.* put forward ten heuristic construction rules in a backtracking search hyperheuristic (BS-HH) algorithm to settle DAPFSP [20]. Zhang and Xing combined memetic algorithm (MA) with social spider optimization (SSO) to solve DAPFSP for the first time [21]. Ying *et al*. developed an Iterated Reference Greedy (IRG) algorithm for distributed no-wait flowshop scheduling problems (DNFSP) with makespan criterion [22]. Unlike the DFSP with a single objective, the DFSP with multiple objectives has drawn attention from researchers recently. For example, Rifai *et al*. developed a novel multi-objective adaptive large neighborhood search (MOALNS) algorithm [9]. Deng and Wang proposed a competitive memetic algorithm (CMA) [23]. Furthermore, Jiang *et al.* presented an improved multi-objective evolutionary algorithm which gives much thought to low-carbon scheduling and energy-efficiency [24], [25].

Although, a significant number of studies have investigated the HFS and the DFSP, there is less research on the distributed hybrid flowshop scheduling (DHFS) problem. To make up for the gap, this study explores the DHFS problem. In the classical HFS, there is only one factory and each job is processed by a machine at each stage. However, in the DHFS, each job can be processed in any among a set of available factories. The two-stage HFS problem is already NP-hard on minimizing the makespan [26], while the DHFS is a much more complex version of the HFS. It follows that the DHFS problem is also NP-hard. To explore the DHFS problem, this study attempts to minimize the makespan for this problem. More specifically, there are two subproblems need to be solved: the one is the distribution subproblem that assigns each job to a factory selected from a set of suitable factories, and the other is the scheduling sub-problem that sequences jobs within each factory so as to yield a feasible schedule to minimize the makespan.

During recent years, many types of meta-heuristics have been developed to solve realistic optimization problems, such as artificial bee colony (ABC) [27]–[36], tabu search (TS) [37]–[41], genetic algorithm (GA) [42], [43], teaching-learning-based optimization (TLBO) [44], [45], invasive weed optimization (IWO) [46]–[48], particle swarm optimization (PSO) [49]–[51], and the fruit fly optimization algorithm (FOA) [52]–[54]. The other optimization algorithms such as the migrating bird optimization (MBO) [55], the harmony search (HS) [56], and the artificial fish swarm (AFS) algorithm [57] have also been researched. In addition, the multi-objective optimization algorithms have also been developed [58]–[64] and many typical applications including task assignment in a cooperative multi-agent design system [65], trust propagation and sequential behaviors [66], heterogeneous scheduling [67], interval data driven construction of shadowed sets with application to linguistic word modeling [68], cloud scheduling [69], and grid scheduling [70].

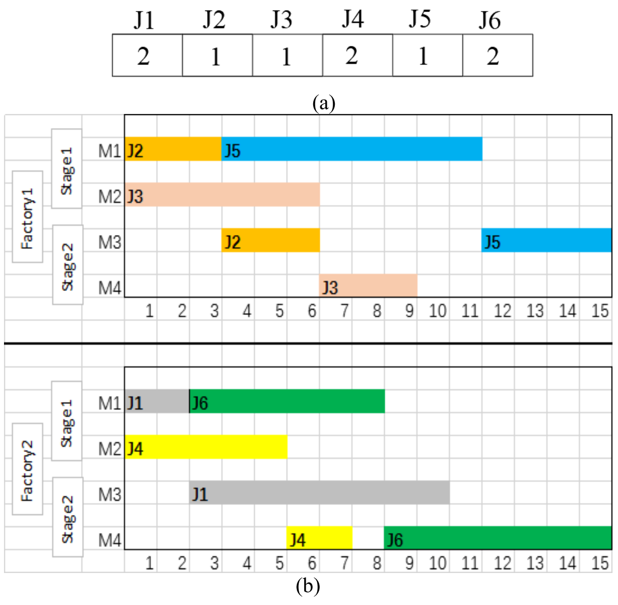

FIGURE 3. Process of encoding and decoding. (a) Process of encoding. (b) Process of decoding.

Algorithm 2: DNEH

```
1. Randomly assign jobs to factories (each factory is
   assigned at least one job)
2. f = 1
3. While f <= factory_num do
4.     While there are unscheduled jobs in f do
5.         Calculate the makespan for all possible
           positions of job j
6.         Find the best position to insert job j and store
           the makespan
7.     End while
8. End while
9. Return the maximum makespan of all factories
```

Very recently, the brain storm optimization (BSO) algorithm proposed by Shi has been extended greatly by researchers [71]. In [72], a random grouping method is introduced by Cao *et al.* to simplify the BSO. Jia *et al.* incorporated the BSO algorithm with the simulated annealing (SA) algorithm to solve continuous optimization problems [73]. Yu *et al*. combined the BSO algorithm with the chaotic local search (CLS) to alleviate stagnation during exploitation [74].

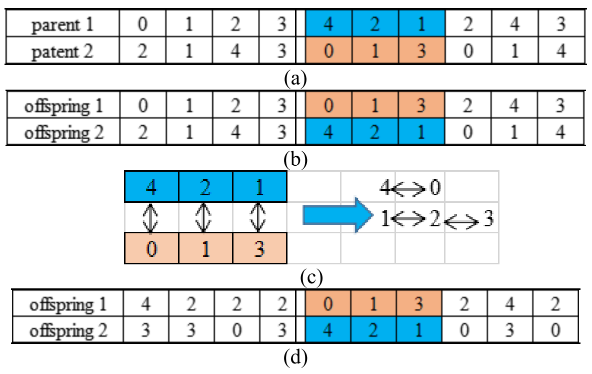
parent 1 | 0 1 2 3 4 2 1 2 4 3
patent 2 | 2 1 4 3 0 1 3 0 1 4
(a)
offspring 1 | 0 1 2 3 0 1 3 2 4 3
offspring 2 | 2 1 4 3 4 2 1 0 1 4
(b)
4 2 1
0 1 3
4<->0
1<->2<->3
(c)
offspring 1 | 4 2 2 2 0 1 3 2 4 2
offspring 2 | 3 3 0 3 4 2 1 0 3 0
(d)

FIGURE 4. Procedure of the proposed crossover operator. (a) Two selected blocks. (b) Swap two blocks. (c) The mapping relationship. (d) Two offspring.

TABLE 2. Key parameter levels.

Parameter	Values			
	1	2	3	4
P_s	50	100	150	200
P_c	0.10	0.30	0.50	0.70
C_n	0.05	0.10	0.20	0.30

Abd proposed the global-best BSO (GBSO), which improved the performance of the BSO algorithm [75]. In [76], Yu *et al.* introduced the flexible search length and memory-based selection to reinforce the BSO algorithm. More recently, it has been proved that the BSO algorithm is an effective method for many types of problems in practice, e.g., multi-objective optimization problems [77], optimization problems in the electromagnetic field [78], the formation reconfiguration of multiple satellites [79], the neural network [80], and vehicle routing problems [81].

The rest of the paper is organized as follows. Section 2 describes the DHFS problem and presents a mixed integer linear programming (MILP) formulation for the DHFS problem. Section 3 illustrates the procedure of BSO and proposes three methods to strengthen BSO. Section 4 presents experimental results and analyses. Finally, Section 5 details conclusions and future research works.

II. PROBLEM DEFINITION AND FORMULATION

A. PROBLEM DEFINITION

The DHFS can be defined in the following way. There is a set of n jobs that must be assigned to one of F identical factories and each job must be processed through a set of S production stages in their designated factory. The job cannot be transferred to another factory when it has been assigned to a certain factory. Each factory f $(f = 1, \ldots, F)$ has exactly the same stage and can process all the jobs. Every single stage s $(s = 1, \ldots, S)$ consists of M_s identical machines, and at least one stage has more than one machine. Every job $j\,(j = 1, \ldots, J)$ has to be processed by the only one machine at each stage. The buffer between two successive stages is infinite and no precedence is allowed.

The optimization objective is to choose a factory for each job and scheduling all jobs within the designated factories simultaneously, so as to minimize the maximum makespan of all factories. According to the classic notation of Graham *et al.* [82], the DHFS with makespan criterion of this paper can be denoted as $DHFF|P_{sjf} = P_{sj}|C_{\max}$, where $DHFF$ is a distributed hybrid flowshop with F distributed factories, $P_{sjf} = P_{sj}$ means the processing time of each job j is identical at stage s for all factories, and $C_{\max}$ denotes the objective is to minimize the maximum makespan of all factories.

TABLE 3. Response values.

P_s	P_c	C_n	Average Fitness
1	1	1	1250.27
1	2	2	1235.22
1	3	3	1231.57
1	4	4	1229.53
2	1	2	1225.00
2	2	1	1253.86
2	3	4	1232.94
2	4	3	1265.27
3	1	3	1228.52
3	2	4	1226.73
3	3	1	1256.69
3	4	2	1237.45
4	1	4	1252.61
4	2	3	1215.27
4	3	2	1223.42
4	4	1	1240.38

B. ILLUSTRTIVE EXAMPLE

In this section, we present a straightforward example to illustrate the DHFS, applying two factory assignment rules proposed by Naderi and Ruiz [4]. Here is an example consists of two factories, six jobs, two stages and two identical machines at each stage. The processing times of six jobs at each stage is shown in Table 1. We assume that there has been a sequence generated by a heuristic: $X = \{2, 5, 6, 4, 3, 1\}$. Fig. 1 (a) shows the Gantt chart of a feasible solution generated by Naderi's Rule 1, assigning the job j to the factory f with the current lowest makespan (without considering the job j). Fig. 1 (b) presents the Gantt chart of a possible solution produced by Naderi's Rule 2, assigning the job j to the factory f with the lowest makespan after inserting the job j.

C. PROBLEM FORMULATION

This section will introduce a MILP model for DHFS, expanding from an HFS model introduced by Li and Han [83].

The following assumptions are used to build the mathematical model:

- A job can only be processed in one factory and changing the factory during processing is not allowed.
- Each job can be processed at time 0.
- Every job must pass through all stages in a factory and is processed by one machine at each stage.
- All machines are in good condition and does not require maintenance during processing.
- All machines in a stage are identical.
- The machine can only process one job at a time.

- All factories are identical, and each factory has the ability to process every job.

Parameters and binary variables of the DHFS model are shown as follows:

Parameters

F :	Number of factories
S :	Identical number of stages in all factories
J :	Number of jobs
Ms :	Number of identical machines in each stage s
f :	Factory index
s :	Stage index
ij :	Job index
m :	Machine index
P_{js} :	Processing time of job j at stage s
B_{js} :	Beginning time of job j at stage s
C_{js} :	Completion time of job j at stages s
T :	a very large positive number

Binary variables

$$X_{i,j,m,f} = \begin{cases} 1 & \text{if job } j \text{ is the successor of job } i \text{ on} \\ & \text{machine } m \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{jf} = \begin{cases} 1 & \text{if job } j \text{ is processed in factory } f \\ 0 & \text{otherwise} \end{cases}$$

$$Z_{j,s,m} = \begin{cases} 1 & \text{if job } j \text{ is processed on} \\ & \text{machine } m \text{ at stage } s \\ 0 & \text{otherwise} \end{cases}$$

The mathematical model of the DHFS can be formulated as follows:

$$\min C\max \tag{1}$$

$$S.T \sum_{f=1}^{F} Y_{jf} = 1 \quad \forall j \in J \tag{2}$$

$$\sum_{m=1}^{Ms} Z_{j,s,m} = 1 \quad \forall j, s \tag{3}$$

$$\sum_{i=1, i\neq j}^{J} (X_{i,j,m,f} + X_{j,i,m,f}) \leq 2 \cdot Y_{jf} \quad \forall j, m, f \tag{4}$$

$$C_{js} \geq C_{is} + P_{js} + ((\sum_{f=1}^{F}\sum_{m=1}^{Ms} X_{i,j,m,f}) - 1) \cdot T \quad \forall i, j, m, f \tag{5}$$

$$C_{js} \geq C_{js} - 1 + P_{js} \quad s \in \{2, \ldots, S\} \quad \forall j \in J \tag{6}$$

$$C\max \geq C_{js} \quad \forall j \in J \tag{7}$$

$$C_{js} = B_{js} + P_{js} \quad \forall j \in J, \ \forall s \in S \tag{8}$$

$$X_{i,j,m,f} \in \{0, 1\} \quad \forall i, j, m, f \tag{9}$$

$$Y_{jf} \in \{0, 1\} \quad \forall j, f \tag{10}$$

$$Z_{j,s,m} \in \{0, 1\} \quad \forall j, s, m \tag{11}$$

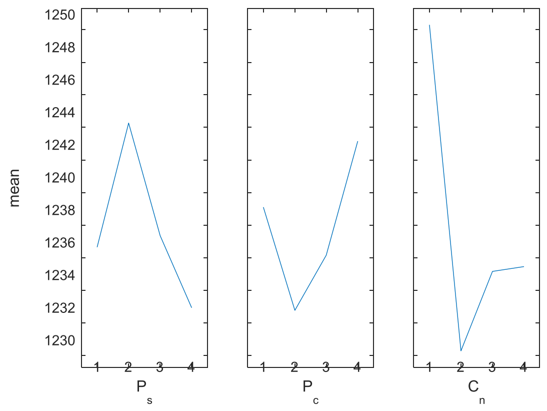

FIGURE 5. Factor level trend.

As is illustrated in the formula (1), the objective of the mathematical model is to minimize the maximum makespan of all factories. Constraint (2) mandates that a job can only be assigned to one factory, in other words, all the stages of an unprocessed job must be performed in the one factory. Constraint (3) ensures that each job passes through all stages in a factory and is processed by one machine at each stage. Constraint (4) guarantees that each job can serve as either an immediate predecessor activity or successor on the selected machine, in simple terms, a job can only be processed once by the selected machine. Constraint (5) ensures that one machine can only process one job at a time. Constraint (6) requires that a job cannot start until the previous stage is completed. Constraint sets (7) and (8) define the maximum makespan. Constraint sets (9) - (11) define all the binary variables.

Theorem 1: In the optimal solution of DHFS with general makespan criterion, each factory is assigned at least one job (if $J > F$).

Proof: Let $C_{\max}$ be the optimal solution of DHFS with makespan criteria.

$C_{\max} = \max\{C_1, C_2, \ldots, C_\alpha, \ldots C_\beta, \ldots C_F\}$, where C_f denotes the completion time of factory f. Suppose that no one job is assigned to factory α and let $C_{\max} = C_\beta$. Owning to $J > F$, at least one factory is assigned more than one job. Without loss of generality, let's consider two scenarios.

Scenario 1: Let factory β be the factory which is assigned more than one job. We send the last complete job in factory β to factory α. And now let $C^*_{\max}$ be the global makespan, C^*_α be the makespan of factory α and C^*_β be the makespan of factory β. After operating, $C^*_{\max} = \max\{C_1, C_2, \ldots, C^*_\alpha, \ldots C^*_\beta, \ldots C_F\}$, $C^*_\beta \leq C_\beta \leq C_{\max}$, $C^*_\alpha \leq C_\beta \leq C_{\max}$. Therefore, we derive $C^*_{\max} \leq C_{\max}$.

Scenario 2: Let factory β has only one job and factory k $(k \neq \alpha, \beta)$ has more than one job. If we move one job from factory k to factory α, the $C_{\max}$ won't increase.

III. ALGORITHM FOR THE DHFS

A. THE CANONICAL BSO

The concept and theory of the BSO algorithm originates from the human brainstorm conference, which is a way of

TABLE 4. Comparisons of HBSO_NO and HBSO_DNEH.

Ins	Scale	Best	Method		dev	
			HBSO_NO	HBSO_DNEH	HBSO_NO	HBSO_DNEH
Inst1	3_50_9	**257.00**	286.00	257.00	11.28	**0.00**
Inst2	5_50_22	**723.00**	757.00	723.00	4.70	**0.00**
Inst3	4_50_35	**516.00**	609.00	516.00	18.02	**0.00**
Inst4	2_50_16	**1036.00**	1114.00	1036.00	7.53	**0.00**
Inst5	3_50_55	**1147.00**	1147.00	1160.00	**0.00**	1.13
Inst6	2_100_10	**562.00**	588.00	562.00	4.63	**0.00**
Inst7	4_100_10	**1420.00**	1459.00	1420.00	2.75	**0.00**
Inst8	4_100_41	**1535.00**	1619.00	1535.00	5.47	**0.00**
Inst9	4_100_58	**760.00**	859.00	760.00	13.03	**0.00**
Inst10	2_100_68	**1249.00**	1312.00	1249.00	5.04	**0.00**
Inst11	3_150_10	**1335.00**	1348.00	1335.00	0.97	**0.00**
Inst12	3_150_18	**2796.00**	2796.00	2809.00	**0.00**	0.46
Inst13	3_150_12	**1586.00**	1750.00	1586.00	10.34	**0.00**
Inst14	5_150_46	**1634.00**	1804.00	1634.00	10.40	**0.00**
Inst15	3_150_47	**1791.00**	1890.00	1791.00	5.53	**0.00**
Inst16	5_200_6	**2082.00**	2082.00	2096.00	**0.00**	0.67
Inst17	5_200_23	**814.00**	861.00	814.00	5.77	**0.00**
Inst18	4_200_46	**706.00**	796.00	706.00	12.75	**0.00**
Inst19	5_200_51	**1235.00**	1341.00	1235.00	8.58	**0.00**
Inst20	2_200_55	**2181.00**	2237.00	2181.00	2.57	**0.00**
Mean		1268.25	1332.75	1270.25	6.47	0.11

stimulating thinking to inspire innovative ideas. The BSO algorithm applies clustering method to search for the local optimum and the global optimum is obtained by comparing the local optimum of each cluster. In order to avoid the algorithm falling into the local optimization, BSO adopts a mutation operation to keep the diversity of solutions.

Inspired by successful applications of BSO in different fields, we try to solve the DHFS problem based on the BSO algorithm. The specific procedure of the canonical BSO algorithm is shown in Algorithm 1. The canonical BSO algorithm involves five parameters: $P_{replace}$, P_{one}, P_{r}, P_{oc}, P_{tc}. $P_{replace}$ is a probability value that determines the probability of replacing the selected cluster center with a new individual. The probability value P_{one} determines whether to select one cluster or two clusters to generate new individuals. In the case of selecting one cluster, P_{oc} is a probability value which decides whether to select a cluster center or randomly select a cluster member to generate new individuals. In the case of selecting two clusters, the probability value P_{tc} determines whether to select two clusters' center or randomly select one member from each selected cluster to generate new individuals. The parameter P_{r} is calculated directly according to the function (12)

$$P_{\mathrm{r}} = M_K/N \tag{12}$$

where M_K is the number of individuals in the cluster K and N is the total number of individuals of all clusters. However, in

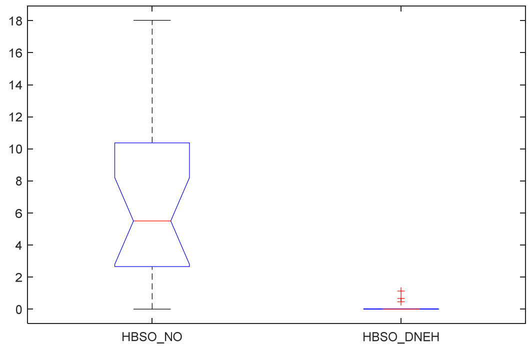

FIGURE 6. Means and 95% LSD intervals for HBSO_NO and HBSO_DNEH ($p = 1.47 \times 10^{-6}$).

the case of randomly selecting two clusters to generate new individuals, the function (12) is not adopted.

In the step of generate new individuals, individuals are generated according to the following function (13).

$$X_{new}^{d} = X_{selected}^{d} + \xi \times n(\mu, \sigma) \tag{13}$$

where $X_{selected}^{d}$ is the d dimension of the selected individual and X_{new}^{d} is the d dimension of the newly generated individual. $n(\mu, \sigma)$ is a gaussian function with a mean of μ and a variance of σ. The ξ is a weighting coefficient that making a difference to the contribution of the gaussian noise, which is generated according to function (14).

$$\xi = \log sig((0.5 \times \max\_T - current\_T)/k) \times rand() \tag{14}$$

where the $\log sig$ is log-sigmoid transfer function, the $\max\_T$ and the $current\_T$ represent the maximum iterations and the current number of iterations respectively. K can change the slope of the $\log sig$ function and the $rand()$ is the random value between (0,1).

B. K-MEANS DISCRETIZATION

In this subsection, the K-means clustering method is adapted to solve the DHFS problem. The K-means clustering method is an algorithm to classify or to group objects based on attributes or features, into K number of groups. K is a positive integer number. Since the K-means method is only applied to a given complete data set and it does not require any special training data, it is an unsupervised machine learning method. The procedure of the K-means clustering method is shown as follows:

Step 1: K solutions are randomly selected from N solutions as the initial center of K clusters.

Step 2: For each solution, calculate the distance to each center and classify it to the nearest center. The distance is calculated according to the Euclidean distance formula:

$$\mathrm{D}^{i} = \sqrt{(S^{i} - C^{i})^{2}} \tag{15}$$

where S^i and C^i represent the i dimension of each solution and each center respectively. D^i is the

TABLE 5. Comparisons of HBSO_D-I and HBSO_D-II.

Ins	Best	Method		dev	
		HBSO_ D-I	HBSO_ D-II	HBSO_ D-I	HBSO_ D-II
Inst1	**246.00**	269.00	246.00	9.35	**0.00**
Inst2	**713.00**	715.00	713.00	0.28	**0.00**
Inst3	**532.00**	532.00	544.00	**0.00**	2.26
Inst4	**1016.00**	1025.00	1016.00	0.89	**0.00**
Inst5	**1110.00**	1130.00	1110.00	1.80	**0.00**
Inst6	**538.00**	538.00	541.00	**0.00**	0.56
Inst7	**1428.00**	1428.00	1436.00	**0.00**	0.56
Inst8	**1487.00**	1499.00	1487.00	0.81	**0.00**
Inst9	**760.00**	789.00	760.00	3.82	**0.00**
Inst10	**1228.00**	1228.00	1264.00	**0.00**	2.93
Inst11	**1311.00**	1334.00	1311.00	1.75	**0.00**
Inst12	**2701.00**	2753.00	2701.00	1.93	**0.00**
Inst13	**1646.00**	1687.00	1646.00	2.49	**0.00**
Inst14	**1647.00**	1647.00	1668.00	**0.00**	1.28
Inst15	**1753.00**	1814.00	1753.00	3.48	**0.00**
Inst16	**2082.00**	2100.00	2082.00	0.86	**0.00**
Inst17	**779.00**	787.00	779.00	1.03	**0.00**
Inst18	**682.00**	686.00	682.00	0.59	**0.00**
Inst19	**1233.00**	1308.00	1233.00	6.08	**0.00**
Inst20	**2059.00**	2072.00	2059.00	0.63	**0.00**
Mean	1245.55	1267.05	1251.55	1.79	0.38

distance between the solution and the center of i dimension.

Step 3: Recalculate the center of each cluster by the average of the i dimension.

Step 4: Iterate 2 $\sim$ 3 steps until the new center is equal to the old center or the specified threshold is satisfied.

Furthermore, instead of using Euclidean distance of all dimensions to cluster solutions, we propose a novel approach to calculate the distance between the solution and the center. As shown in Fig. 2, in the first step, we calculate the distance of each dimension between the solution s and the center c. Then, the sum of the distances of all dimensions $D = \sum_{i=1}^{6} d_i$ is used to measure the distance from the solution s to the center c.

C. ENCODING/DECODING SCHEMES AND INITIALIZATION

As described in the above sections, for the DHFS problem, a solution consists of two parts, i.e., the distribution of each job to the factory, and the processing sequence of jobs within each factory. To solve the problem with the BSO algorithm, we propose an encoding scheme and two initialization strategies. The encoding scheme is illustrated as follows.

Let the vector A consists of J elements which are randomly generated from Unidrnd (1, F). To make the encoding scheme understood easily, we apply it to the previous example presented in Table 1.

Here is a simple DHFS problem consists of two factories, six jobs, two stages and two identical machines at each stage, one of the possible factory assignment of the problem can be denoted by $A = (2\ 1\ 1\ 2\ 1\ 2)$. As shown in Fig. 3 (a), the first element 2 means that the job 1 is assigned to the factory 2. Similarly, the second element 1 represents that the job 2 is assigned to the factory 1, and so on. To put it in a nutshell, the job set (2 3 5) is assigned to the factory 1, and the job set (1 4 6) is assigned to the factory 2.

It has been proved that each factory is assigned at least one job in the optimal solution in Theorem 1. In order to satisfy this condition, we adopt a method which combines the shuffling order and the random generation in the first initialization strategy. Steps of the first initialization strategy are shown as follows:

Step 1: Randomly disrupt the order of factories $(1,2,\ldots,F)$ as the first F elements of the vector A.

Step 2: Randomly generate $J - F$ elements from $(1,2,\ldots,F)$ as the remaining part of the vector A.

In the first step, we guarantee that each factory is assigned at least one job. To simplify the decoding scheme, let $A = (2\ 1\ 1\ 2\ 1\ 2)$.

In the decoding scheme, jobs are assigned to the first idle machine at each stage. Fig. 3 (b) illustrates the decoding process of the factory 1. Makespan is 15 in the factory 1. The same scheduling process is applied to the factory 2. Finally, the makespan of the DHFS problem is the maximum completion time of all factories.

Next, a simple example is given to illustrate the novel approach to calculate the distance between the solution and the center presented in Fig. 2. $c = (2, 1, 2, 2, 1, 1)$ is one of the centers, $s = (2, 2, 1, 2, 1, 2)$ is a solution, then the distance between the solution and the center $D = (2-2) + (2-1) + (1-2) + (2-2) + (1-1) + (2-1) = 1$.

D. DNEH CONSTRUCTIVE HEURISTIC

NEH is one of the most effective construction heuristic algorithms for flowshop scheduling problems, which is proposed by Nawaz *et al.* [84]. The procedure of NEH is shown as follows:

Step 1: Sum up the processing time of each job at all processing stages, and then sort the sum value from large to small.

Step 2: Select the first job (the job with the longest processing time) and insert the second job into the front and the back position of the first job, and then arrange the second job into the position with minimum makespan.

Step 3: Insert the next job into all possible positions of the previous processing sequence and calculate the makespan for each possible position, and then arrange it into the position with minimum makespan.

Step 4: Iterate step 3 until all of jobs are arranged.

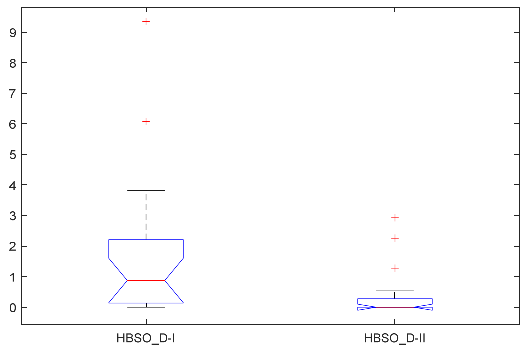

FIGURE 7. **Means and 95% LSD intervals for HBSO_D-I and HBSO_D-II ($p = 1.67 \times 10^{-2}$).**

To take advantage of good features of NEH, we develop a novel constructive heuristic based on NEH for distributed flowshop scheduling problems in this paper, which is called distributed NEH (DNEH). The main procedure of the proposed DNEH method is illustrated in Algorithm 2. Firstly, jobs are assigned to each factory randomly, and the final manufacturing scheduling must satisfy the Theorem 1. And then, in the process of scheduling within each factory, the job is scheduled according to the order in which it is assigned to the factory. Each job is inserted into all possible positions of the previous sequence and the makespan of each possible position is calculated. After that the position with the minimum makespan is selected as the final position. Finally, the maximum completion time of all factories is referred to as the makespan of the distributed flowshop scheduling problem.

E. CROSSOVER OPERATOR

Two-point crossover operator is used in the canonical BSO. To enhance the performance of the algorithm, we propose a new crossover operator for the BSO algorithm based on the Partial-Mapped Crossover (PMX). The best individual of each cluster is selected as parents. Here is a simple example with ten jobs, five factories. The crossover procedure is shown as follows:

Step 1: Randomly select two segments from two parents (two segments are chosen in the same place of two parents as is shown in Fig. 4 (a)).

Step 2: Swap two selected segments (see Fig. 4 (b)).

Step 3: Do conflict detection. Firstly, creating a mapping relationship based on selected segments of two parents (see Fig. 4 (c)). Next, all genes that conflicted with the selected segment are mapped to other genes (see Fig. 4 (d)).

F. PROCEDURE OF THE HBSO

With the above research and design, the procedure of the proposed HBSO algorithm is shown as follows:

Step 1: Initialize the population and calculate the fitness of each individual with the DNEH constructive heuristic.

TABLE 6. **Comparisons of HBSO_NCR and HBSO_CR.**

Ins	Best	Method		dev	
		HBSO_NCR	HBSO_CR	HBSO_NCR	HBSO_CR
Inst1	**244.00**	244.00	248.00	**0.00**	1.64
Inst2	**255.00**	263.00	255.00	3.14	**0.00**
Inst3	**250.00**	252.40	250.00	0.96	**0.00**
Inst4	**691.00**	691.00	714.00	**0.00**	3.33
Inst5	**743.00**	745.00	743.00	0.27	**0.00**
Inst6	**717.80**	717.80	730.20	**0.00**	1.73
Inst7	**523.00**	529.00	523.00	1.15	**0.00**
Inst8	**557.00**	596.00	557.00	7	**0.00**
Inst9	**535.60**	562.00	535.60	4.93	**0.00**
Inst10	**1001.00**	1004.00	1001.00	0.3	**0.00**
Inst11	**1014.00**	1062.00	1014.00	4.73	**0.00**
Inst12	**1008.00**	1028.60	1008.00	2.04	**0.00**
Inst13	**1048.00**	1098.00	1048.00	4.77	**0.00**
Inst14	**1074.00**	1145.00	1074.00	6.61	**0.00**
Inst15	**1062.80**	1120.60	1062.80	5.44	**0.00**
Inst16	**532.00**	532.00	551.00	**0.00**	3.57
Inst17	**563.00**	564.00	563.00	0.18	**0.00**
Inst18	**546.80**	546.80	556.40	**0.00**	1.76
Inst19	**1382.00**	1398.00	1382.00	1.16	**0.00**
Inst20	**1439.00**	1440.00	1439.00	0.07	**0.00**
Mean	759.35	776.96	762.75	2.14	0.6

Step 2: Cluster the population into K clusters with the K-means method and select the best individual in each cluster as the cluster center. The distance calculation method proposed in Section III Part B is involved in K-means method.

Step 3: Randomly select a cluster center by the formula (12) and replace it with a randomly generated individual.

Step 4: Update all individuals. The four ways to update individuals are detailed as follows:

(1) Randomly select a cluster center by the formula (12) and add a random perturbation to generate a new individual.

(2) Randomly select a cluster by the formula (12). In the selected cluster, an individual is randomly selected, and a random perturbation is added to generate a new individual.

(3) Randomly select two cluster centers. The crossover operator proposed in Section III Part E is adopted to generate new individuals.

(4) Randomly select two clusters. In each selected cluster, an individual is randomly selected to generate new individuals by the proposed crossover operator.

Step 5: Stop if the specified threshold is satisfied, otherwise go to step 2.

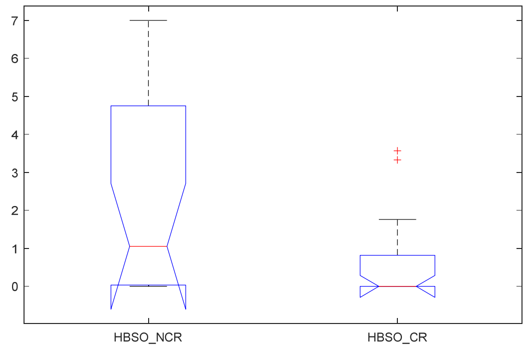

FIGURE 8. Means and 95% LSD intervals for HBSO_NCR and HBSO_CR ($p = 1.69 \times 10^{-2}$).

IV. EXPERIMENT ANALYSES

A. EXPERIMENTAL INSTANCES

To solve the DHFS problem and verify the effectiveness of the HBSO algorithm, we randomly generate twenty large-scale test instances for the DHFS problem based on the realistic production data. The number of the identical factory for each test problem is randomly generated from Unidrnd (2, 5). The test instances set are classified into four categories according to the number of jobs. In addition, to test the effectiveness of the HBSO algorithm in the environment of different complexity, each category is divided into five sub-problems with diffident number of stages. For instance, the notation instance_50_2 means a problem consists of 50 jobs and 2 stages.

B. EXPERIMENTAL PARAMETER ANALYSIS

In order to effectively set the system parameters, we carried out preliminary experiments. The key parameters included in the experiment are as follows. (1) The population size (P_s), which is the total number of individuals in an experiment. (2) The probability of crossover (P_c), which determines the possibility of crossover-operation for each individual. (3) The coefficient C_n. The number of clusters is equal to C_n multiplied by the population size. There is no fixed formula or method to select the optimal number of clusters, which needs to be specified manually. It should be noted that choosing a larger number of clusters can reduce the error of data but will increase the risk of over-fitting. According to preliminary experiments, the levels of three critical parameters are given in Table 2.

These three key parameters affecting the performance of the proposed algorithm are analyzed by using DOE Taguchi method. For three key parameters, select the orthogonal array $L_{16}(4^3)$. The proposed algorithm runs 30 times for each parameter combination independently. And then, we calculated the average RPI value obtained by the comparison algorithm as the response variable which is presented in Table 3. The factor level trend of the three key parameters is shown in Fig. 5.

As can be seen from Fig. 5, the proposed algorithm can achieve better performance by selecting the following three key parameter levels: (1) P_s with the level 4; (2) P_c with the level 2; and (3) C_n with the level 2. In addition, it can

TABLE 7. Comparisons with the canonical BSO.

Instance	CBSO			HBSO			dev	
	Min	Max	Avg	Min	Max	Avg	CBSO	HBSO
Inst1	259.00	269.00	263.00	242.00	263.00	**252.40**	4.20	**0.00**
Inst2	679.00	702.00	**690.60**	681.00	745.00	717.80	**0.00**	3.94
Inst3	562.00	582.00	569.20	514.00	596.00	**562.00**	1.28	**0.00**
Inst4	1061.00	1097.00	1074.60	1000.00	1062.00	**1028.60**	4.47	**0.00**
Inst5	1074.00	1108.00	**1095.60**	1064.00	1145.00	1120.60	0.00	2.28
Inst6	575.00	583.00	578.80	535.00	564.00	**546.80**	5.85	**0.00**
Inst7	1388.00	1419.00	**1400.80**	1360.00	1440.00	1417.00	**0.00**	1.16
Inst8	1542.00	1597.00	1559.60	1473.00	1562.00	**1546.00**	0.88	**0.00**
Inst9	795.00	830.00	812.60	735.00	771.00	**756.80**	7.37	**0.00**
Inst10	1271.00	1309.00	1287.20	1219.00	1287.00	**1255.80**	2.50	**0.00**
Inst11	1322.00	1331.00	1326.20	1285.00	1328.00	**1310.40**	1.21	**0.00**
Inst12	2753.00	2771.00	2758.60	2697.00	2796.00	**2758.00**	0.02	**0.00**
Inst13	1700.00	1754.00	1729.80	1569.00	1658.00	**1617.40**	6.95	**0.00**
Inst14	1684.00	1765.00	1734.00	1612.00	1735.00	**1701.40**	1.92	**0.00**
Inst15	1839.00	1874.00	1856.40	1741.00	1813.00	**1784.60**	4.02	**0.00**
Inst16	2071.00	2133.00	**2106.40**	2053.00	2146.00	2127.60	**0.00**	1.01
Inst17	828.00	855.00	841.80	760.00	836.00	**798.80**	5.38	**0.00**
Inst18	758.00	771.00	762.20	658.00	729.00	**705.00**	8.11	**0.00**
Inst19	1290.00	1311.00	1301.20	1221.00	1304.00	**1277.00**	1.90	**0.00**
Inst20	2201.00	2236.00	2218.20	2018.00	2116.00	**2067.40**	7.29	**0.00**
Mean	1282.60	1314.85	1298.34	1221.90	1294.80	1267.57	2.43	0.00

TABLE 8. Calibrated parameters of two comparison algorithms.

Algorithm	Parameters			
SS	$b = 10$	$l = 10$	$a = 40$	$ITRN = 450$
GA	$P_{size} = 110$	$P_c = 1$ $P_m = 0$	$G_r = 30$ $G_l = 10$	$int_{ls} = 1$ $ITRN = 450$

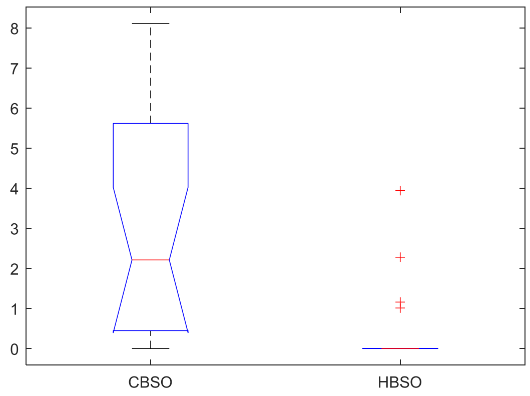

FIGURE 9. Means and 95% LSD intervals for CBSO and HBSO ($p = 2.42 \times 10^{-4}$).

be seen from Fig. 5 that the proposed algorithm can achieve better performance with the increase of the population size. However, the larger the P_s value, the more computing resources will be consumed during the exploration process and the exploitation capability of the proposed algorithm will decrease. Therefore, in order to balance the exploration and exploitation ability of the proposed algorithm, the upper bound of the P_s value is limited to 200. The suitable values for P_s, P_c, and C_n are set to 200, 0.3 and 0.1, respectively.

C. EFFECTIVENESS OF DNEH CONSTRUCTIVE HEURISTIC

In order to verify the effectiveness of the DNEH constructive heuristic, we present a detailed comparison of the two methods, i.e., the HBSO algorithm with the proposed DNEH method (HBSO_DNEH for short), and the HBSO algorithm without the DNEH method, hereafter denoted HBSO_NO. For experimental credibility, the two comparison algorithms are run independently 30 times with the same twenty test instances. The experimental results are shown in Table 4.

Table 4 consists of seven columns. The instance name is presented in the first column. The second column shows the size of the instance. The following column provides the optimal value of each instance. The fourth column presents the experimental results of the HBSO_NO algorithm, and the experimental results of the HBSO_DNEH algorithm are shown in the following column. In order to intuitively compare the quality of solutions obtained by two contrast algorithms, we calculate the percentage deviation with formula (16), and results are presented in the last two columns.

The following information can be obtained from Table 4: (1) the HBSO_DNEH algorithm obtains 17 optimal solutions of 20 test instances, while there are only 3 optimal solutions calculated by the HBSO_NO algorithm; (2) as shown in the last line, the average makespan and the average percentage deviation of HBSO_DNEH algorithm is much smaller than that of HBSO_NO algorithm.

Furthermore, to determine the statistical superiority of the BSO_DNEH algorithm, we conduct the Analysis of Variance (ANOVA) on experimental results. Fig. 6 presents the means and 95% intervals for HBSO_NO algorithm and HBSO_DNEH algorithm. It is obvious that there are significant differences between HBSO_NO algorithm and HBSO_DNEH algorithm. As mentioned above, it is obvious that HBSO_DNEH algorithm is superior to HBSO_NO algorithm.

$$dev = \frac{C_{\max} - C_{best}}{C_{best}} \times 100\% \qquad (16)$$

D. EFFECTIVENESS OF THE DISTANCE METHOD

To verify the validity of the distance calculation method proposed in Section III Part B, we compared the two methods of calculating distance in the HBSO algorithm, i.e., the HBSO algorithm with the Euclidean distance method (HBSO_D-I for short), and the HBSO algorithm with the proposed distance calculation method (HBSO_D-II for short). For the reliability of the experiment, the two comparison algorithms are run independently 30 times with the same twenty test instances. The experimental results are given in Table 5.

Table 5 consists of six columns. The instance name is shown in the first column. The second column presents the optimal value of each instance. The third column details the experimental results of HBSO_D-I, and the experimental results of HBSO_D-II is shown in the following column. In order to present the quality of the solutions of the two methods intuitively, we calculate the percentage deviation of solutions, and the results are presented in the last two columns.

Table 5 has shown that: (1) HBSO_D-II obtains 15 optimal solutions of 20 test instances, while there are only 5 optimal solutions calculated by HBSO_D-I; (2) as shown in the last line, HBSO_D-II obtains a smaller average maximum makespan and average percentage deviation than HBSO_D-I. Moreover, to investigate the statistical superiority of HBSO_D-II, we conduct the Analysis of Variance (ANOVA) on experimental results. Fig. 7 presents the means and 95% intervals for HBSO_D-I and HBSO_D-II. It can be observed from Fig. 7 that there are significant differences between HBSO_D-I and HBSO_D-II. In conclusion, it is safe to say that HBSO_D-II is superior to HBSO_D-I.

E. EFFECTIVENESS OF THE CROSSOVER OPERATOR

To verify the performance of the crossover operator proposed in Section III Part E, we conducted controlled experiments, i.e., the HBSO algorithm with the proposed crossover operator (HBSO_CR for short), and the HBSO algorithm without the proposed crossover operator (HBSO_NCR). Two-point

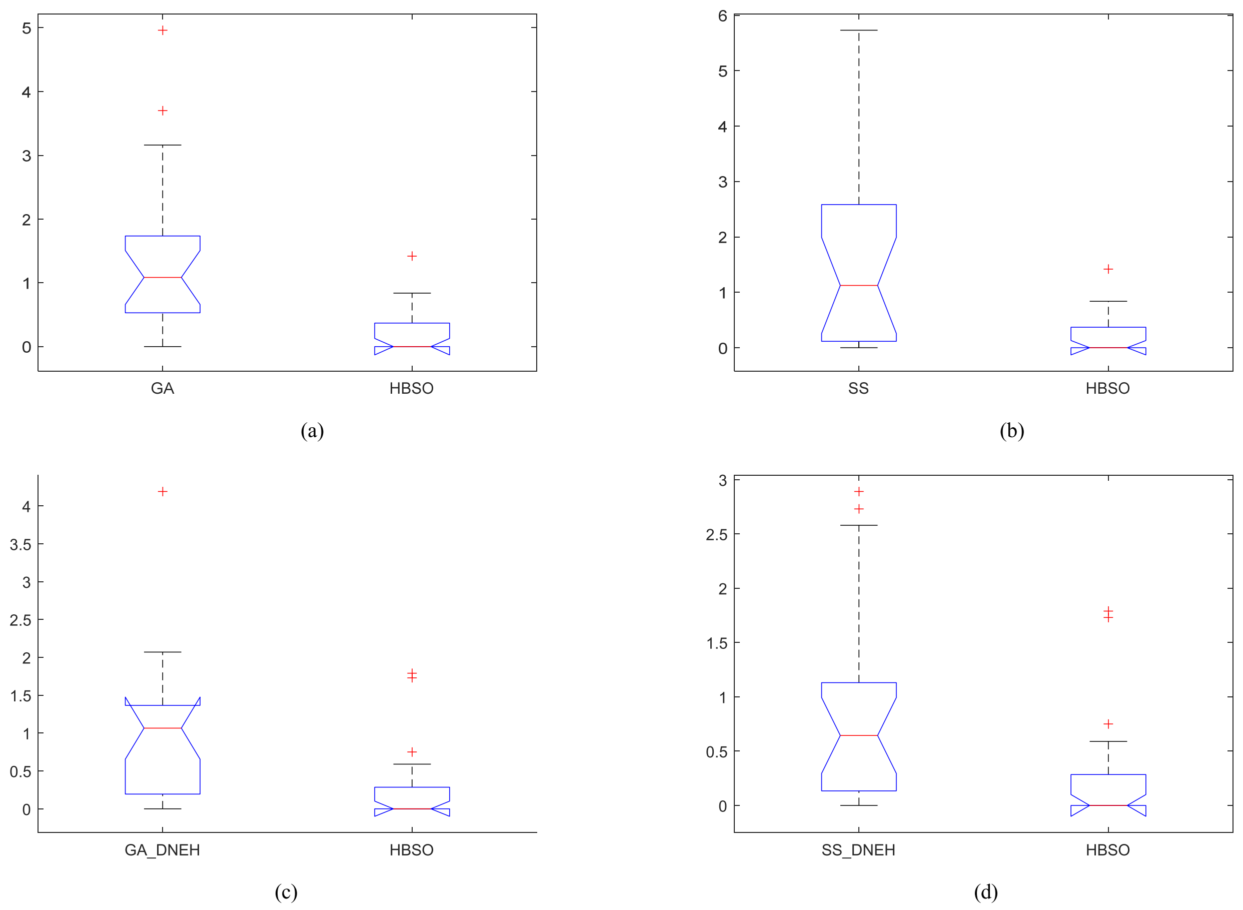

FIGURE 10. **Means and 95% LSD intervals for comparison algorithms. (a) Means and 95% LSD intervals for GA and HBSO ($p = 4.64 \times 10^{-4}$). (b) Means and 95% LSD intervals for SS and HBSO ($p = 7.82 \times 10^{-4}$). (c) Means and 95% LSD intervals for GA_DNEH and HBSO ($p = 1.01 \times 10^{-2}$). (d) Means and 95% LSD intervals for SS_DNEH and HBSO ($p = 2.07 \times 10^{-2}$).**

crossover operator is used in the HBSO-NCR. For the reliability of the experiment, the two comparison algorithms are run independently 30 times with the same twenty test instances. The experimental results are given in Table 6.

The instance name is shown in the first column of the Table 6. The second column presents the optimal value of each instance. The third column details the experimental results of HBSO_NCR, and the experimental results of HBSO_CR is shown in the following column. In order to present the quality of the solutions obtained by two contrast algorithms intuitively, we calculate the percentage deviation of solutions, and results are presented in the last two columns.

It can be observed from Table 6 that: (1) HBSO_NCR obtains 15 optimal solutions of 20 test instances, while there are only 5 optimal solutions calculated by HBSO_CR; (2) as shown in the last line, HBSO_CR obtains a smaller average maximum makespan and average percentage deviation than HBSO_NCR. In addition, to investigate the statistical superiority of HBSO_D-II, we conduct the Analysis of Variance (ANOVA) on experimental results. Fig. 8 presents the means and 95% intervals for HBSO_NCR and HBSO_CR. It can be observed from Fig. 8 that there are significant differences between HBSO_NCR and HBSO_CR. Therefore, the proposed crossover operator can improve the performance of the proposed algorithm.

F. COMPARISONS WITH THE CANONICAL BSO

To investigate the improvement of the HBSO algorithm proposed in this paper, we conduct a comparison with the canonical BSO algorithm, hereafter denoted CBSO. Each algorithm is run 30 times on the same computer with twenty test instances. The minimum, maximum and average values of each compared algorithm are shown in Table 7. The percentage deviation of each instance is presented in the last two columns of the Table 7.

As can be seen from the Table 7: (1) the HBSO algorithm works out 16 optimal solutions, while there are only 5 optimal solutions obtained by the CBSO algorithm; (2) as shown in the last line, the average makespan (the minimum, maximum and average values) and the average percentage deviation of the HBSO algorithm is smaller than the CBSO algorithm. In addition, to confirm the statistical superiority of the HBSO algorithm, we conduct the Analysis of Variance (ANOVA) on average experimental results. Fig. 9 presents the means

TABLE 9. Comparisons results of GA, SS and HBSO.

Instance	Best	Makespan			dev		
		GA	SS	HBSO	GA	SS	HBSO
Inst1	**242.00**	254.00	**242.00**	**242.00**	4.96	**0.00**	**0.00**
Inst2	**681.00**	689.00	720.00	**681.00**	1.17	5.73	**0.00**
Inst3	**514.00**	521.00	530.00	**514.00**	1.36	3.11	**0.00**
Inst4	**986.00**	1006.00	**986.00**	1000.00	2.03	**0.00**	1.42
Inst5	**1061.00**	**1061.00**	1107.00	1064.00	**0.00**	4.34	0.28
Inst6	**535.00**	536.00	539.00	**535.00**	0.19	0.75	**0.00**
Inst7	**1360.00**	1381.00	1387.00	**1360.00**	1.54	1.99	**0.00**
Inst8	**1473.00**	1486.00	1485.00	**1473.00**	0.88	0.81	**0.00**
Inst9	**729.00**	**729.00**	759.00	735.00	**0.00**	4.12	0.82
Inst10	**1219.00**	1231.00	1244.00	**1219.00**	0.98	2.05	**0.00**
Inst11	**1285.00**	1293.00	1288.00	**1285.00**	0.62	0.23	**0.00**
Inst12	**2697.00**	2726.00	2704.00	**2697.00**	1.08	0.26	**0.00**
Inst13	**1556.00**	1586.00	**1556.00**	1569.00	1.93	**0.00**	0.84
Inst14	**1612.00**	1663.00	1657.00	**1612.00**	3.16	2.79	**0.00**
Inst15	**1733.00**	1734.00	**1733.00**	1741.00	0.06	**0.00**	0.46
Inst16	**2053.00**	2062.00	2065.00	**2053.00**	0.44	0.58	**0.00**
Inst17	**756.00**	784.00	**756.00**	760.00	3.70	**0.00**	0.53
Inst18	**658.00**	664.00	673.00	**658.00**	0.91	2.28	**0.00**
Inst19	**1221.00**	1237.00	1250.00	**1221.00**	1.31	2.38	**0.00**
Inst20	**2018.00**	2040.00	2047.00	**2018.00**	1.09	1.44	**0.00**
Mean	1219.45	1234.15	1236.40	1221.90	1.37	1.64	0.24

and 95% intervals for the CBSO algorithm and the HBSO algorithm. It is evident from the Fig. 9 that there are obvious differences between the CBSO algorithm and the HBSO algorithm, with the p-value 2.24×10^{-4}. In conclusion, the proposed HBSO algorithm has statistical advantages over the CBSO algorithm.

G. COMPARISONS WITH THE OTHER EFFICIENT ALGORITHMS

To the best of our knowledge, this study is one of the first work to solve the DHFS problem. There is none published algorithms to make detail comparisons directly. In order to investigate the effectiveness of the proposed HBSO algorithm, we conduct a comparison with the scatter search (SS) algorithm [13] and genetic algorithm (GA) [85]. The reasons that the two algorithms SS and GA are selected as the compared algorithm are as follows: (1) SS is the algorithm for solving the distribute permutation flowshop scheduling problem (DPFSP), which has been verified as one of the efficient algorithms and be selected as the compared algorithm by many published literatures. The problem considered in this study is the distributed hybrid flow shop problem, which can also be considered as the special case of the DPFSP, with the special constraint that in each factory there are several parallel machines. Therefore, in this study, we recoded and modified the SS algorithm to solve the considered problem in this study; and (2) the GA algorithm proposed in [85] is to solve the hybrid flow shop scheduling problem. The problem in this study can also be considered as one special case of the hybrid flow shop problem, with the special constraint that there are several factories in the production horizon. Therefore, the two selected compared algorithms are taken from the very related literatures, and these two algorithms have also been verified as the efficient algorithms for solving the corresponding problems. Based on the analysis, we select and recode the two compared algorithms and make detailed comparisons to verify the efficiency of the proposed algorithm.

The time complexity of the proposed HBSO algorithm is $O(M \times N^2)$, where M and N are the number of iterations and the number of individuals, respectively. From the literature of the two compared algorithms, we found that the time complexity of GA is $O(M \times N^2)$, where M and N are the number of iterations and the number of individuals, respectively, and the time complexity of SS is $O(M \times b \times l \times n)$, where M, b, l and n are the number of iterations, the number of best solutions, the number of vectors and the number of jobs, respectively. However, these two algorithms should be recoded to solve the DHFS problem, and the time complexity of the two compared algorithms for solving the considered problem in this study are all of $O(M \times N^2)$. It should be noted that the time complexity of SS has changed. The main reason is that in solving the distributed flowshop, the proposed rapid fitness computation method in the SS can be used, but it cannot be used for solving the DHFS problem directly.

To enable GA and SS to solve DHFS problems, we have modified the two algorithms and adopted strategies proposed in the two papers. In order to confirm that the parameters in these papers are suitable for DHFS problems, we carried out calibration experiments. These numerical levels parameters affecting the performance of the two comparison algorithms are analyzed by DOE Taguchi method with the factors and levels taken from the two original papers. The calibrated parameters of the two comparison algorithms are presented in Table 8. In SS, the controlled factors b, l, a and the number of iterations (*ITRN*) are set to 10, 10, 40 and 450 respectively. In GA, the population size (P_{size}) is set to 110, the crossover probability (P_c) is set to 1, and the mutation probability (P_m) is set to 0, the restart level (G_r) is set to 30, the local search

TABLE 10. Comparisons results of GA_DNEH, SS_DNEH and HBSO.

Instance	Best	Makespan			dev		
		GA_DNEH	SS_DNEH	HBSO	GA_DNEH	SS_DNEH	HBSO
Inst1	**242.00**	247.00	249.00	**242.00**	2.07	2.89	**0.00**
Inst2	**669.00**	697.00	**669.00**	681.00	4.19	**0.00**	1.79
Inst3	**511.00**	516.00	**511.00**	514.00	0.98	**0.00**	0.59
Inst4	**983.00**	994.00	**983.00**	1000.00	1.12	**0.00**	1.73
Inst5	**1064.00**	1067.00	1093.00	**1064.00**	0.28	2.73	**0.00**
Inst6	**534.00**	**534.00**	536.00	535.00	**0.00**	0.37	0.19
Inst7	**1360.00**	1380.00	1380.00	**1360.00**	1.47	1.47	**0.00**
Inst8	**1473.00**	1475.00	1477.00	**1473.00**	0.14	0.27	**0.00**
Inst9	**735.00**	743.00	741.00	**735.00**	1.09	0.82	**0.00**
Inst10	**1219.00**	1232.00	1224.00	**1219.00**	1.07	0.41	**0.00**
Inst11	**1285.00**	1289.00	1290.00	**1285.00**	0.31	0.39	**0.00**
Inst12	**2696.00**	**2696.00**	2709.00	2697.00	**0.00**	0.48	0.04
Inst13	**1564.00**	**1564.00**	1578.00	1569.00	**0.00**	0.90	0.32
Inst14	**1610.00**	1614.00	**1610.00**	1612.00	0.25	**0.00**	0.12
Inst15	**1728.00**	**1728.00**	1742.00	1741.00	**0.00**	0.81	0.75
Inst16	**2053.00**	2082.00	2072.00	**2053.00**	1.41	0.93	**0.00**
Inst17	**760.00**	770.00	770.00	**760.00**	1.32	1.32	**0.00**
Inst18	**658.00**	665.00	675.00	**658.00**	1.06	2.58	**0.00**
Inst19	**1218.00**	1231.00	**1218.00**	1221.00	1.07	**0.00**	0.25
Inst20	**2018.00**	2050.00	2037.00	**2018.00**	1.59	0.94	**0.00**
Mean	1219.05	1228.70	1228.20	1221.90	0.95	0.84	0.29

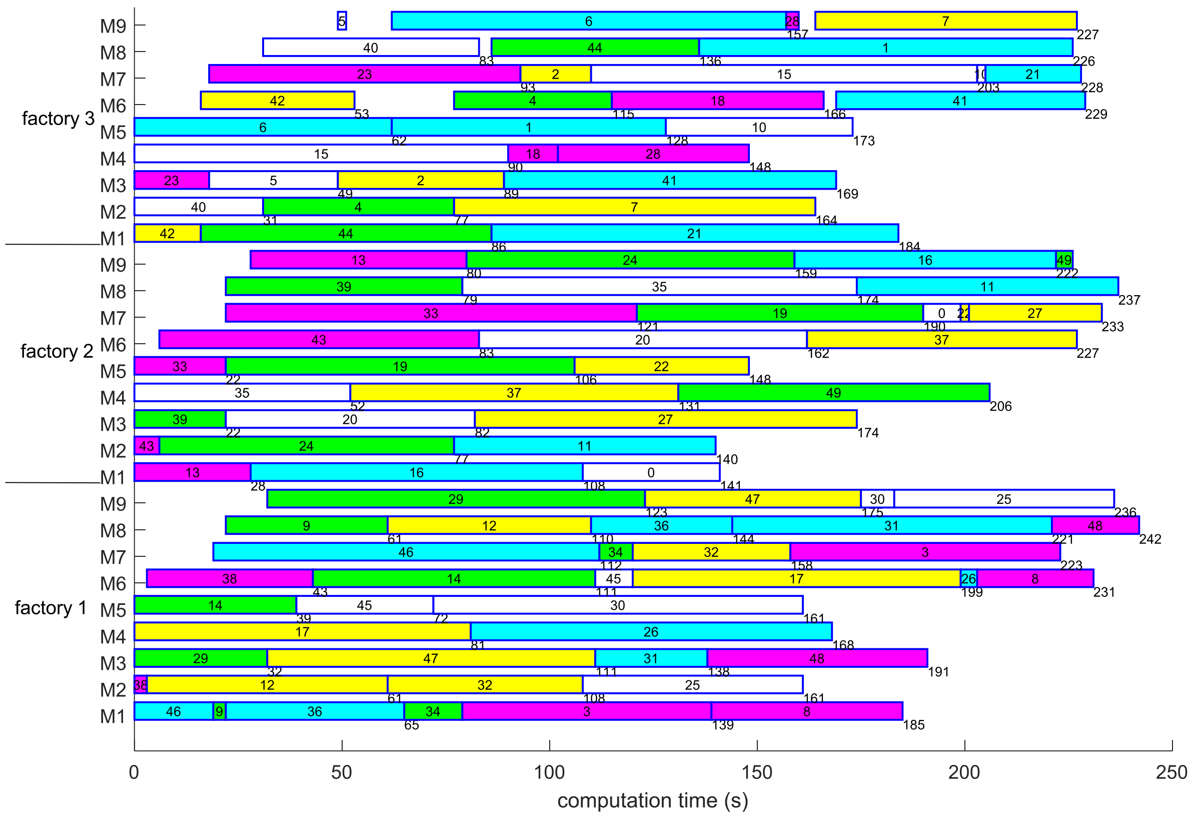

FIGURE 11. The Gantt chart for "Inst1".

frequency (G_l) is set to 10 and the local search intensity factor (int_{ls}) is set to 1.

Every algorithm runs 30 times on the same computer with twenty test instances. The comparison experiment results are shown in Table 9. The instance name is shown in the first column. The second column presents the best value of each instance. The following three columns detail the best values obtained by each comparison algorithm. To intuitively compare the quality of solutions obtained by three contrast algorithms, we calculate the percentage deviation of solutions, and the results are given in the last four columns. It can be seen from Table 9 that: (1) the HBSO algorithm obtains 13 optimal solutions, while there are only 2 and 5 optimal solutions found out by the GA and SS, respectively; (2) as shown in the last line, the average makespan and the average percentage deviation of the HBSO algorithm is the smallest of three comparison algorithms. Moreover, in order to confirm the statistical superiority of the HBSO algorithm, we conduct the Analysis of Variance (ANOVA) on experimental results. Fig. 10 (a) and (b) present the means and 95% intervals for the comparison algorithms. It is evident from Fig. 10 (a) and (b) that the HBSO is superior to GA and SS.

In addition, we combine the DNEH method proposed in this paper with two comparison algorithms, i.e., GA with the DNEH method (GA_DNEH for short), and SS with the DNEH method (SS_DNEH for short). Every algorithm runs 30 times on the same computer with twenty test instances. The comparison experiment results are shown in Table 10. We conduct the Analysis of Variance (ANOVA) on experimental results. Fig. 10 (c) and (d) present the xmeans and 95% intervals for the comparison algorithms. It is obvious that HBSO is superior to GA_DNEH and SS_DNEH. In conclusion, the above experimental results and analyses prove that the HBSO algorithm is effective for solving DHFS problems.

Fig. 11 presents the Gantt chart for the best solution of instance "Inst1", which illustrates the effectiveness of the proposed HBSO algorithm.

V. CONCLUSIONS

With the trend of manufacturing globalization, distributed multi-factories are becoming more and more common in various industries. In this paper, the distributed hybrid flowshop scheduling (DHFS) problem is proposed, with the objective to minimize the maximum makespan of all factories. Then, a hybrid brain storm optimization (HBSO) algorithm is applied to solve the DHFS problem for the first time. In addition, a DNEH constructive heuristic is proposed for distributed flowshop scheduling problems. Moreover, we design a new approach to calculate the distance in the procedure of K-means discretization and propose a new crossover operator to strengthen the BSO algorithm. Furthermore, to evaluate the effectiveness of the proposed HBSO algorithm, a set of test instances is generated. This paper broadens the scope of research on distributed flow shop scheduling problems and enables the BSO algorithm to be used for combinatorial optimization problems successfully.

For future research works, the HBSO algorithm can be adapted to other distributed flowshop scheduling problems or other combinatorial optimization problems, such as the crowd evacuation simulation system [86] and the optimization problem in wireless networks [87]. Besides, future studies can consider more realistic factors for the DHFS, such as heterogeneous factories. Finally, energy consumption can be considered as a future research direction for DHFS problems.

REFERENCES

[1] M. S. Salvador, "A solution to a special class of flow shop scheduling problems," in *Proc. Symp. Theory Scheduling Appl.*, 1973, pp. 83–91.
[2] C.-J. Liao, E. Tjandradjaja, and T.-P. Chung, "An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem," *Appl. Soft. Comput.*, vol. 12, no. 6, pp. 1755–1764, 2012.
[3] H.-C. Chang and T.-K. Liu, "Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms," *J. Intell. Manuf.*, vol. 28, no. 8, pp. 1973–1986, 2015.
[4] B. Naderi and R. Ruiz, "The distributed permutation flowshop scheduling problem," *Comput. Oper. Res.*, vol. 37, no. 4, pp. 754–768, 2010.
[5] S. Hatami, R. Ruiz, and C. Andrés-Romano, "The distributed assembly permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 17, pp. 5292–5308, 2013.
[6] T.-K. Liu, Y.-P. Chen, and J.-H. Chou, "Solving distributed and flexible job-shop scheduling problems for a real-world fastener manufacturer," *IEEE Access*, vol. 2, pp. 1598–1606, 2014.
[7] S.-W. Lin and K.-C. Ying, "Minimizing makespan for solving the distributed no-wait flowshop scheduling problem," *Comput. Ind. Eng.*, vol. 99, pp. 202–209, Sep. 2016.
[8] K.-C. Ying and S.-W. Lin, "Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks," *Expert. Syst. Appl.*, vol. 92, pp. 132–141, Feb. 2018.
[9] A. P. Rifai, H.-T. Nguyen, and S. Z. M. Dawal, "Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling," *Appl. Soft Comput.*, vol. 40, pp. 42–57, Mar. 2016.
[10] J. Gao, R. Chen, and W. Deng, "An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 3, pp. 641–651, 2013.
[11] J. Gao and R. Chen, "A hybrid genetic algorithm for the distributed permutation flowshop scheduling problem," *Int. J. Comput. Intell. Syst.*, vol. 4, no. 4, pp. 497–508, 2011.
[12] S.-Y. Wang, L. Wang, M. Liu, and Y. Xu, "An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem," *Int. J. Prod. Econ.*, vol. 145, no. 1, pp. 387–396, 2013.
[13] B. Naderi and R. Ruiz, "A scatter search algorithm for the distributed permutation flowshop scheduling problem," *Eur. J. Oper. Res.*, vol. 239, no. 2, pp. 323–334, 2014.
[14] V. Fernandez-Viagas and J. M. Framinan, "A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 53, no. 4, pp. 1111–1123, 2014.
[15] V. Fernandez-Viagas, P. Perez-Gonzalez, and J. M. Framinan, "The distributed permutation flow shop to minimise the total flowtime," *Comput. Ind. Eng.*, vol. 118, pp. 464–477, Apr. 2018.
[16] H. Bargaoui, O. B. Driss, and K. Ghédira, "A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion," *Comput. Ind. Eng.*, vol. 111, pp. 239–250, Sep. 2017.
[17] R. Ruiz, Q.-K. Pan, and B. Naderi, "Iterated Greedy methods for the distributed permutation flowshop scheduling problem," *Omega*, vol. 83, pp. 213–222, Mar. 2019.
[18] M.-C. Wu, C.-S. Lin, C.-H. Lin, and C.-F. Chen, "Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems," *Comput. Oper. Res.*, vol. 80, pp. 101–112, Apr. 2017.
[19] J. Lin and S. Zhang, "An effective hybrid biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem," *Comput. Ind. Eng.*, vol. 97, pp. 128–136, Jul. 2016.
[20] J. Lin, Z.-J. Wang, and X. Li, "A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem," *Swarm Evol. Comput.*, vol. 36, pp. 124–135, Oct. 2017.
[21] G. Zhang and K. Xing, "Memetic social spider optimization algorithm for scheduling two-stage assembly flowshop in a distributed environment," *Comput. Ind. Eng.*, vol. 125, pp. 423–433, Nov. 2018.
[22] K.-C. Ying, S.-W. Lin, C.-Y. Cheng, and C.-D. He, "Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems," *Comput. Ind. Eng.*, vol. 110, pp. 413–423, Aug. 2017.
[23] J. Deng and L. Wang, "A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem," *Swarm Evol. Comput.*, vol. 32, pp. 121–131, Feb. 2017.
[24] E. D. Jiang, L. Wang, and J. Lu, "Modified multi-objective evolutionary algorithm based on decomposition for low-carbon scheduling of distributed permutation flow-shop," in *Proc. IEEE Symp. Ser. Comput. Intell.*, 2017.
[25] E. D. Jiang and L. Wang, "An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time," *Int. J. Prod. Res.*, vol. 57, no. 6, pp. 1756–1771, 2019.

[26] J. N. D. Gupta, "Two-stage, hybrid flowshop scheduling problem," *J. Oper. Res. Soc.*, vol. 39, no. 4, pp. 359–364, 1988.

[27] J.-Q. Li, S.-C. Bai, P.-Y. Duan, H.-Y. Sang, Y.-Y. Han, and Z.-X. Zheng, "An improved artificial bee colony algorithm for addressing distributed flow shop with distance coefficient in a prefabricated system," *Int. J. Prod. Res.*, to be published. doi: 10.1080/00207543.2019.1571687.

[28] J. Bai and H. Liu, "Multi-objective artificial bee algorithm based on decomposition by PBI method," *Appl. Intell.*, vol. 45, no. 4, pp. 976–991, 2016.

[29] J. Li, P. Duan, H. Sang, S. Wang, Z. Liu, and P. Duan, "An efficient optimization algorithm for resource-constrained steelmaking scheduling problems," *IEEE Access*, vol. 6, pp. 33883–33894, 2018.

[30] L. Wang, G. Zhou, Y. Xu, and M. Liu, "An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 60, pp. 1111–1123, Jun. 2012.

[31] J.-Q. Li, Q.-K. Pan, and M. F. Tasgetiren, "A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities," *Appl. Math. Model.*, vol. 38, no. 3, pp. 1111–1132, Feb. 2014.

[32] Y.-Y. Han, J. Liang, Q.-K. Pan, J.-Q. Li, H.-Y. Sang, and N. N. Cao, "Effective hybrid discrete artificial bee colony algorithms for the total flowtime minimization in the blocking flowshop problem," *Int. J. Adv. Manuf. Technol.*, vol. 67, nos. 1–4, pp. 397–414, 2013.

[33] J.-Q. Li, Q.-K. Pan, and K.-Z. Gao, "Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 55, pp. 1159–1169, Aug. 2011.

[34] Y.-Y. Han, D. Gong, and X. Sun, "A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking," *Eng. Optim.*, vol. 47, no. 7, pp. 927–946, Jul. 2015.

[35] J.-Q. Li, Q.-K. Pan, and P.-Y. Duan, "An improved artificial bee colony algorithm for solving hybrid flexible flowshop with dynamic operation skipping," *IEEE Trans. Cybern.*, vol. 46, no. 6, pp. 1311–1324, Jun. 2016.

[36] L. Wang, G. Zhou, Y. Xu, and M. Liu, "A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3593–3608, 2013.

[37] J.-Q. Li, P. Duan, J. Cao, X.-P. Lin, and Y.-Y. Han, "A hybrid Pareto-based tabu search for the distributed flexible job shop scheduling problem with E/T criteria," *IEEE Access*, vol. 6, pp. 58883–58897, 2018.

[38] Y. Xia and Z. Fu, "Improved tabu search algorithm for the open vehicle routing problem with soft time windows and satisfaction rate," *Cluster Comput.*, to be published. doi: 10.1007/s10586-018-1957-x.

[39] J.-Q. Li, Q.-K. Pan, P. N. Suganthan, and T. J. Chua, "A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 52, nos. 5–8, pp. 683–697, Feb. 2011.

[40] Q.-K. Pan and R. Ruiz, "Local search methods for the flowshop scheduling problem with flowtime minimization," *Eur. J. Oper. Res.*, vol. 222, no. 1, pp. 31–43, 2012.

[41] Q.-K. Pan, P. N. Suganthan, J. J. Liang, and M. F. Tasgetiren, "A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem," *Expert. Syst. Appl.*, vol. 38, pp. 3252–3259, Apr. 2011.

[42] A. Baniamerian, M. Bashiri, and F. Zabihi, "Two phase genetic algorithm for vehicle routing and scheduling problem with cross-docking and time windows considering customer satisfaction," *J. Ind. Eng. Int.*, vol. 14, no. 1, pp. 15–30, 2018.

[43] K. Ghoseiri and S. F. Ghannadpour, "Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm," *Appl. Soft. Comput.*, vol. 10, no. 4, pp. 1096–1107, 2010.

[44] P.-Y. Duan, J.-Q. Li, Y. Wang, H.-Y. Sang, and B.-X. Jia, "Solving chiller loading optimization problems using an improved teaching-learning-based optimization algorithm," *Optim. Control Appl. Methods*, vol. 39, no. 1, pp. 65–77, 2018.

[45] J.-Q. Li, Q.-K. Pan, and K. Mao, "A discrete teaching-learning-based optimisation algorithm for realistic flowshop rescheduling problems," *Eng. Appl. Artif. Intell.*, vol. 37, pp. 279–292, Jan. 2015.

[46] Z. Zheng and J. Li, "Optimal chiller loading by improved invasive weed optimization algorithm for reducing energy consumption," *Energy Buildings*, vol. 161, pp. 80–88, Feb. 2018.

[47] Y. Liu, Y. C. Jiao, Y. M. Zhang, and Y. Y. Tan, "Synthesis of phase-only reconfigurable linear arrays using multiobjective invasive weed optimization based on decomposition," *Int. J. Antennas Propag.*, 2014. doi: 10.1155/2014/630529.

[48] H.-Y. Sang, Q.-K. Pan, P.-Y. Duan, and J.-Q. Li, "An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems," *J. Intell. Manuf.*, vol. 29, no. 6, pp. 1337–1349, 2018.

[49] X. Zhao, "A perturbed particle swarm algorithm for numerical optimization," *Appl. Soft. Comput.*, vol. 10, no. 1, pp. 119–124, 2010.

[50] H. Liu, L. Gao, and Q. Pan, "A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4348–4360, Apr. 2011.

[51] C.-C. Wu, J.-Y. Chen, W.-C. Lin, K. Lai, S.-C. Liu, and P.-W. Yu, "A two-stage three-machine assembly flow shop scheduling with learning consideration to minimize the flowtime by six hybrids of particle swarm optimization," *Swarm Evol. Comput.*, vol. 41, pp. 97–110, Aug. 2018.

[52] J.-Q. Li, Q.-K. Pan, and K. Mao, "A hybrid fruit fly optimization algorithm for the realistic hybrid flowshop rescheduling problem in steelmaking systems," *IEEE Trans. Automat. Sci. Eng.*, vol. 13, no. 2, pp. 932–949, Apr. 2016.

[53] E. Osaba, X.-S. Yang, F. Diaz, E. Onieva, A. D. Masegosa, and A. Perallos, "A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy," *Soft Comput.*, vol. 21, no. 18, pp. 5295–5308, 2017.

[54] Q.-K. Pan, H.-Y. Sang, J.-H. Duan, and L. Gao, "An improved fruit fly optimization algorithm for continuous function optimization problems," *Knowl.-Based Syst.*, vol. 62, pp. 69–83, May 2014.

[55] T. Meng, Q.-K. Pan, J.-Q. Li, and H.-Y. Sang, "An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem," *Swarm Evol. Comput.*, vol. 38, pp. 64–78, Feb. 2018.

[56] B. Zhang, Q.-K. Pan, X.-L. Zhang, and P.-Y. Duan, "An effective hybrid harmony search-based algorithm for solving multidimensional knapsack problems," *Appl. Soft Comput.*, vol. 29, pp. 288–297, Apr. 2015.

[57] Z.-X. Zheng, J.-Q. Li, and P.-Y. Duan, "Optimal chiller loading by improved artificial fish swarm algorithm for energy saving," *Math. Comput. Simul.*, vol. 155, pp. 227–243, Jan. 2019.

[58] W. Zheng, Y. Tan, L. Meng, and H. Zhang, "An improved MOEA/D design for many-objective optimization problems," *Appl. Intell.*, vol. 48, no. 10, pp. 3839–3861, 2018.

[59] Q. Lin and J. Chen, "A novel micro-population immune multiobjective optimization algorithm," *Comput. Oper. Res.*, vol. 40, no. 6, pp. 1590–1601, 2013.

[60] R. Han, Y. Gao, C. Wu, and D. Lu, "An effective multi-objective optimization algorithm for spectrum allocations in the cognitive-radio-based Internet of Things," *IEEE Access*, vol. 6, pp. 12858–12867, 2018.

[61] J.-Q. Li, H.-Y. Sang, Y.-Y. Han, C.-Q. Wang, and K.-Z. Gao, "Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions," *J. Clean. Prod.*, vol. 181, pp. 584–598, Apr. 2018.

[62] W. Gong, Z. Cai, and D. Liang, "Adaptive ranking mutation operator based differential evolution for constrained optimization," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 716–727, Apr. 2015.

[63] J.-Q. Li, Q.-K. Pan, and S.-X. Xie, "An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems," *Appl. Math. Comput.*, vol. 218, no. 18, pp. 9353–9371, 2012.

[64] P. Zhang, H. Liu, and Y. Ding, "Dynamic bee colony algorithm based on multi-species co-evolution," *Appl. Intell.*, vol. 40, no. 3, pp. 427–440, 2014.

[65] H. Liu, P. Zhang, B. Hu, and P. Moore, "A novel approach to task assignment in a cooperative multi-agent design system," *Appl. Intell.*, vol. 43, no. 1, pp. 162–175, 2015.

[66] Z. Zhang and H. Liu, "Social recommendation model combining trust propagation and sequential behaviors," *Appl. Intell.*, vol. 43, no. 3, pp. 695–706, 2015.

[67] J. Wang, B. Gong, H. Liu, and S. Li, "Model and algorithm for heterogeneous scheduling integrated with energy-efficiency awareness," *Trans. Inst. Meas. Control*, vol. 38, no. 4, pp. 452–462, 2015.

[68] C. Li, J. Yi, H. Wang, G. Zhang, and J. Li, "Interval data driven construction of shadowed sets with application to linguistic word modelling," *Inf. Sci.*, to be published. doi: 10.1016/j.ins.2018.11.018.

[69] J. Wang, B. Gong, H. Liu, and S. Li, "Multidisciplinary approaches to artificial swarm intelligence for heterogeneous computing and cloud scheduling," *Appl. Intell.*, vol. 43, no. 3, pp. 662–675, Oct. 2015.

[70] J. Wang, B. Gong, H. Liu, S. Li, and J. Yi, "Heterogeneous computing and grid scheduling with parallel biologically inspired hybrid heuristics," *Trans. Inst. Meas. Control*, vol. 36, no. 6, pp. 805–814, 2014.

[71] Y. H. Shi, "Brain storm optimization algorithm," in *Proc. Int. Conf. Swarm. Int.*, vol. 1, 2011, pp. 303–309.
[72] Z. Cao, Y. H. Shi, X. F. Rong, B. L. Liu, Z. Q. Du, and B. Yang, "Random grouping brain storm optimization algorithm with a new dynamically changing step size," *Adv. Swarm. Comput. Int.*, pp. 357–364, 2015.
[73] Z. Jia, H. Duan, and Y. Shi, "Hybrid brain storm optimisation and simulated annealing algorithm for continuous optimisation problems," *Int. J. Bio-Inspired Comput.*, vol. 8, no. 2, pp. 109–121, 2016.
[74] Y. Yu, S. Gao, S. Cheng, Y. Wang, S. Song, and F. Yuan, "CBSO: A memetic brain storm optimization with chaotic local search," *Memetic Comput.*, vol. 10, no. 4, pp. 353–367, 2017.
[75] M. El-Abd, "Global-best brain storm optimization algorithm," *Swarm Evol. Comput.*, vol. 37, pp. 27–44, Dec. 2017.
[76] Y. Yu, S. Gao, Y. Wang, J. Cheng, and Y. Todo, "ASBSO: An improved brain storm optimization with flexible search length and memory-based selection," *IEEE Access*, vol. 6, pp. 36977–36994, 2018.
[77] Y. Shi, J. Xue, and Y. Wu, "Multi-objective optimization based on brain storm optimization algorithm," *Int. J. Swarm Intell. Res.*, vol. 4, no. 3, pp. 1–21, 2013.
[78] H. Duan, S. Li, and Y. Shi, "Predator–prey brain storm optimization for DC brushless motor," *IEEE Trans. Magn.*, vol. 49, no. 10, pp. 5336–5340, Oct. 2013.
[79] C. Sun, H. Duan, and Y. Shi, "Optimal satellite formation reconfiguration based on closed-loop brain storm optimization," *IEEE Comput. Intell. Mag.*, vol. 8, no. 4, pp. 39–51, Nov. 2013.
[80] X. Ma, Y. Jin, and Q. Dong, "A generalized dynamic fuzzy neural network based on singular spectrum analysis optimized by brain storm optimization for short-term wind speed forecasting," *Appl. Soft. Comput.*, vol. 54, pp. 296–312, May 2017.
[81] L. Ke, "A brain storm optimization approach for the cumulative capacitated vehicle routing problem," *Memetic Comput.*, vol. 10, no. 4, pp. 411–421, 2018.
[82] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Ann. Math.*, vol. 5, pp. 287–326, 1979.
[83] J. Q. Li and Y. Han, "A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system," *Cluster Comput.*, to be published.
[84] M. Nawaz, Jr., 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.
[85] C. Yu, Q. Semeraro, and A. Matta, "A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility," *Comput. Oper. Res.*, vol. 100, pp. 211–229, Dec. 2018.
[86] H. Liu, B. Liu, H. Zhang, L. Li, X. Qin, and G. Zhang, "Crowd evacuation simulation approach based on navigation knowledge and two-layer control mechanism," *Inf. Sci.*, vols. 436–437, no. 88, pp. 247–267, Apr. 2018.
[87] J. Tian, H. Zhang, D. Wu, and D. Yuan, "QoS-constrained medium access probability optimization in wireless interference-limited networks," *IEEE Trans. Commun.*, vol. 66, no. 3, pp. 1064–1077, Mar. 2018.

JIAN-HUA HAO received the B.Sc. degree from Shandong Normal University, in 2018, where he is currently pursuing the master's degree. His current research interests include discrete optimization and scheduling.

JUN-QING LI received the master's degree of computer science and technology from Shandong Economic University, Shandong, China, in 2004, and the Ph.D. degree from Northeastern University, Shenyang, China, in 2016. Since 2017, he has been with the School of Information Science and Engineering, Shandong Normal University, where he became a Professor, in 2017. He is also with the School of Computer Science, Liaocheng University. He has authored more than 30 refereed papers. His current research interests include intelligent optimization and scheduling.

YU DU received the B.Sc. and master's degrees from Sichuan University. He is currently pursuing the Ph.D. with the School of Information Science and Engineering, Shandong Normal University. His current research interests include discrete optimization and scheduling.

MEI-XIAN SONG received the B.Sc. degree from Qufu Normal University, in 2017. She is currently pursuing the master's degree with Liaocheng University. Her current research interests include discrete optimization and scheduling.

PENG DUAN received the Ph.D. degree from Shandong University, Jinan, China, in 2015. Since 2015, he has been with the School of Computer, Liaocheng University, where he was appointed as a Lecturer. He has authored more than 10 refereed papers. His current research interests include discrete optimization and scheduling.

YING-YU ZHANG received the Ph.D. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2009. Since 2009, he has been with the School of Computer, Liaocheng University, where he was appointed as a Lecturer. He has authored more than 10 refereed papers. His current research interests include discrete optimization and scheduling.