An Improved Harmony Search Algorithm for the Distributed Two Machine Flow-Shop Scheduling Problem

Jin Deng, Ling Wang, Jingnan Shen and Xiaolong Zheng

Abstract In this paper, an improved harmony search (IHS) algorithm is proposed to solve the distributed two machine flow-shop scheduling problem (DTMFSP) with makespan criterion. First, a two-stage decoding rule is developed for the decimal vector based representation. At the first stage, a job-to-factory assignment method is designed to transform a continuous harmony vector to a factory assignment. At the second stage, the Johnson's method is applied to provide a job sequence in each factory. Second, a new pitch adjustment rule is developed to adjust factory assignment effectively. The influence of parameter setting on the IHS is investigated based on the Taguchi method of design of experiments, and numerical experiments are carried out. The comparisons with the global-best harmony search and the original harmony search demonstrate the effectiveness of the IHS in solving the DTMFSP.

Keywords Harmony search · Distributed flow-shop scheduling · Decoding rule · Pitch adjustment

1 Introduction

The two machine flow-shop scheduling problem has been widely studied during the past few decades [1-10]. Recently, the distributed shop scheduling has attracted more and more attention [11-18], which is considered under the globalization environment to improve the production efficiency and economic benefits in the multi-plant companies. However, to the best of our knowledge, there is no

J. Deng(✉) · L. Wang · J. Shen · X. Zheng
Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, People's Republic of China
e-mail: {dengj13,chenjn12,zhengxl11}@mails.tsinghua.edu.cn, wangling@tsinghua.edu.cn

J.H. Kim and Z.W. Geem (eds.), *Harmony Search Algorithm*,
Advances in Intelligent Systems and Computing 382,
DOI: 10.1007/978-3-662-47926-1_11

published work that directly addresses the distributed two machine flow-shop scheduling problem (DTMFSP). The DTMFSP is to allocate jobs to suitable factories and to determine a reasonable processing sequence in each factory so as to optimize certain objectives, such as the makespan criterion.

The classical two machine flow-shop scheduling problem with makespan criterion can be solved by Johnson's algorithm [1] in polynomial time. Nevertheless, the DTMFSP is more complex than the distributed single machine scheduling problem (DSMSP) that is a special version of the parallel machine scheduling problem (PMSP). The DTMFSP is NP-hard, as the PMSP is NP-hard [19]. Therefore, it is significant to develop effective and efficient approaches for solving the DTMFSP.

Inspired by the search procedure for better harmonies in the musical performance, the harmony search (HS) algorithm [20] is one of the population-based meta-heuristics. Different from the genetic algorithm (GA) that utilizes only two parents for generating the offspring, the HS generates a new harmony vector by considering all of the vectors in the harmony memory. Due to its simplicity and easy implementation, the HS algorithm has been applied to many optimization problems, such as engineering optimization [21], vehicle routing [22], truss structures design [23], water network design [24], electrical distribution network reconfiguration [25], and shop scheduling [26]. Numerical comparisons showed that the HS algorithm was faster than the GA [27, 28]. In this paper, an improved HS (IHS) algorithm will be proposed to solve the DTMFSP. To be specific, a harmony is represented as a real vector, and a two-stage decoding rule is developed to convert the continuous vector to a feasible schedule, and a new pitch adjustment rule is designed to adjust factory assignment effectively. The influence of parameter setting is investigated and numerical results are provided. The comparative results demonstrate the effectiveness of the IHS.

The remainder of the paper is organized as follows. The DTMFSP is described in Section 2. The IHS algorithm for the DTMFSP is introduced in Section 3. In Section 4, the influence of parameter setting is investigated, and numerical results and comparisons are provided. Finally, we end the paper with some conclusions and future work in Section 5.

2 Problem Description

Notions:

n: the total number of jobs.
f: the total number of factories.
n_k: the number of jobs in the factory k.
$O_{i,j}$: the j-th operation of job i.
$p_{i,j}$: the processing time of $O_{i,j}$.
$C_{i,j}$: the completion time of $O_{i,j}$.
π^k: the processing sequence in factory k.
$\pi = \{\pi^1, \pi^2, \ldots, \pi^f\}$: a certain schedule.

The DTMFSP can be described as follows. There are n jobs to be processed in f identical factories, where each factory has two machines. Job i has two operations $\{O_{i,1}, O_{i,2}\}$ to be processed one after another. Operation $O_{i,j}$ is executed on machine j with processing time $p_{i,j}$. Once a job is assigned to a factory, it cannot be transferred to other factories.

For the DTMFSP, the makespan $C_{\max}$ of a certain schedule can be calculated in the following way:

$$C_{\pi^k(1),1} = p_{\pi^k(1),1}, k = 1, 2, \dots, f \tag{1}$$

$$C_{\pi^k(i),1} = C_{\pi^k(i-1),1} + p_{\pi^k(i),1}, k = 1, 2, \dots, f; i = 2, 3, \dots, n_k \tag{2}$$

$$C_{\pi^k(1),2} = C_{\pi^k(1),1} + p_{\pi^k(1),2}, k = 1, 2, \dots, f \tag{3}$$

$$C_{\pi^k(i),2} = \max\{C_{\pi^k(i),1}, C_{\pi^k(i-1),2}\} + p_{\pi^k(i),2}, k = 1,2,\dots, f; i = 2, 3, \dots, n_k \tag{4}$$

$$C_{\max} = \max C_{\pi^k(n_k),2}, k = 1, 2, ..., f \tag{5}$$

3 IHS for DTMFSP

3.1 Original HS

In the original HS algorithm, each harmony denotes a solution, represented by a D-dimension real vector $X_i = \{x_i(1), x_i(2), ..., x_i(D)\}$. The HS algorithm contains four parameters: the harmony memory size (MS), the harmony memory consideration rate (P_{CR}), the pitch adjusting rate (P_{AR}) and the distance bandwidth (dB). The harmony vectors are stored in the harmony memory (HM) as $\{X_1, X_2, ..., X_{MS}\}$ to generate new harmony vectors. According to [22], P_{CR} and P_{AR} help the algorithm find globally and locally improved solutions, respectively. P_{AR} and dB are important in fine-tuning the solution vectors and in adjusting convergence rate.

The procedure of the HS algorithm can be simplely described as follows:

Step 1. Set parameters MS, P_{CR}, P_{AR}, dB and the stopping criterion.

Step 2. Initialize the HM and calculate the objective function value $F(X_i)$ for each harmony X_i.

Step 3. Improvise a new harmony X_{new} from the HM as Fig. 1. When improvising a new harmony, three rules are applied: a memory consideration, a pitch adjustment and a random selection.

Step 4. Update the HM as $X_w = X_{new}$ if X_{new} is better, where X_w represents the worst harmony in the HM.

Step 5. If stopping criteron is not satisfied, go to Step 3.

For $j = 1 : D$
  If ($rand_1 < P_{CR}$) /* memory consideration */
    $x_{new}(j) = x_a(j)$, $a = \{1, 2, \ldots, MS\}$
    If ($rand_2 < P_{AR}$) /* pitch adjustment */
      $x_{new}(j) = x_{new}(j) + \mathrm{sgn}(rand_3 - 0.5) \times dB$
    End if
  Else /* random selection */
    $x_{new}(j) = x_{min}(j) + (x_{max}(j) - x_{min}(j)) \times rand_4$
  End if
End for
Note:
1. $rand_1$–$rand_4$ are random numbers generated uniformly between 0 and 1;
2. sgn(.) is a sign function that returns -1, 0 or 1.

Fig. 1 Improvising a new harmony in HS

3.2 Encoding and Decoding

To solve the DTMFSP, a harmony in the IHS is represented by an n-dimension decimal vector. To determine the factory assignment for each job and the processing sequence in each factory, a two-stage decoding rule is developed.

At the first stage, a job-to-factory assignment method is designed to map a continuous harmony vector to a factory assignment. To be specific, the search range (0, 1] is partitioned into f intervals $(0, 1/f]$, $(1/f, 2/f]$ … $((f–1)/f, 1]$. If the value of the j-th dimension belongs to $((k–1)/f, k/f]$, job j is assigned to factory k. For an example with 2 factories and 5 jobs, suppose the harmony vector is {0.54, 0.22, 0.78, 0.16, 0.93}. Job 1 is assigned to factory 2 because 0.54 belongs to (1/2, 1]. Similarly, job 2 and job 4 are assigned to factory 1, while job 3 and job 5 are assigned to factory 2.

Since the two machine flow-shop scheduling problem with makespan criterion can be solved by the Johnson's algorithm [1], the Johnson's algorithm is applied at the second stage to provide a job sequence for each factory.

With such a decoding rule, a real harmony vector can be converted to a feasible schedule, and then its objective value can be calculated.

3.3 Initial Harmony Memory

To generate a harmony memory with enough diversity, all MS initial harmony vectors are randomly generated.

For each harmony vector, the value of each dimension is a random number that is uniformly generated between 0 and 1.

3.4 Improvise New Harmony

To adjust the factory assignment effectively, a new pitch adjustment is developed as follows.

When adjusting the value of $x_{new}(j)$, it first allocates job j to the position that is determined by the Johnson's algorithm in every factory. Then, it chooses factory k with the earliest completion time. The value of $x_{new}(j)$ is calculated as follows:

$$x_{new}(j) = \quad (k - rand) \,/\, f \tag{6}$$

For an example with 2 factories and 5 jobs, suppose that $x_{new}(3)$ is undergoing the pitch adjustment with the precondition of $x_{new}(1) = 0.54$ and $x_{new}(2) = 0.22$. First, suppose that job 3 is allocated to factory 1 and factory 2, respectively. Assume that the processing sequences obtained by Johnson's algorithm of the two factories are $\pi^1 = \{2, 3\}$ and $\pi^2 = \{3, 1\}$. Then, job 3 will be assigned to factory 1 if the completion time of π^1 is smaller than that of π^2; otherwise, job 3 will be assigned to factory 2. Finally, the value of $x_{new}(3)$ is calculated by formula (6).

The pseudo-code of improvising a new harmony in the IHS is illustrated in Fig. 2.

```
For j = 1 : D
   If (rand1 < PCR) /* memory consideration */
      xnew(j) = xa(j), a = {1, 2, …, MS}
      If (rand2 < PAR) /*new pitch adjustment */
         Find the factory k that can process job j with the earliest completion time;
         xnew(j) = (k – rand3) / f
      End if
   Else /* random selection */
      xnew(j) = xmin(j) + (xmax(j) – xmin(j)) × rand4
   End if
End for
Note: rand1–rand4 are uniform random number between 0 and 1.
```

Fig. 2 Improvising a new harmony in HIS for DTMFSP

3.5 Update Harmony Memory

After a new harmony vector X_{new} is generated, it will be compared with the worst harmony vector X_w in the HM, and then a greedy selection is employed.

That is, if X_{new} is better than X_w, then X_{new} will replace X_w and become a new member of the HM.

4 Numerical Results and Comparisons

4.1 Experimental Setup

To evaluate the performance of the IHS, 100 random instances are generated. Table 1 lists the ranges of parameters for generating the instances. The IHS is coded in C language, and all the tests are run on a PC with an Intel(R) core(TM) i5-3470 CPU @ 3.2GHz / 8GB RAM under Microsoft Windows 7. The stopping criterion is set as $0.1 \times n$ seconds CPU time.

Table 1 Ranges of Parameters for Instances

Parameter	Range
f	{2, 3, 4, 5, 6}
n	{20, 50, 100, 200}
$p_{i,j}$	U(1, 100)

4.2 Parameter Setting

The IHS contains three parameters: MS, P_{CR} and P_{AR}. To investigate their influence on the performance of the IHS, the Taguchi method of design of experiments (DOE) is carried out with a moderate-sized instance F4_11 (i.e. $f = 4$, $n = 100$). Four levels are considered for each parameter as in Table 2.

For each parameter combination, the IHS is run 10 times independently. With the orthogonal array $L_{16}(4^3)$, Table 3 lists the resulted average makespan value as response value (RV). Then, the response value and the rank of each parameter are calculated and listed in Table 4, and the main effect plots are shown in Fig. 3.

Table 2 Factor Levels

Factor	Factor levels			
	1	2	3	4
MS	5	10	15	20
P_{CR}	0.7	0.75	0.8	0.85
P_{AR}	0.7	0.75	0.8	0.85

Table 3 Orthogonal Array and RV Values

Experiment number	Factor levels			RV
	MS	P_{CR}	P_{AR}	
1	1	1	1	1200.0
2	1	2	2	1200.0
3	1	3	3	1200.0
4	1	4	4	1200.3
5	2	1	2	1200.0
6	2	2	1	1200.1
7	2	3	4	1200.0
8	2	4	3	1199.9
9	3	1	3	1200.0
10	3	2	4	1200.2
11	3	3	1	1200.1
12	3	4	2	1200.3
13	4	1	4	1200.4
14	4	2	3	1200.3
15	4	3	2	1200.0
16	4	4	1	1200.1

Table 4 Response Value and Rank of Each Parameter

Level	*MS*	P_{CR}	P_{AR}
1	1200.075	1200.100	1200.075
2	1200.000	1200.150	1200.075
3	1200.150	1200.025	1200.050
4	1200.200	1200.150	1200.225
Delta	0.200	0.125	0.175
Rank	1	3	2

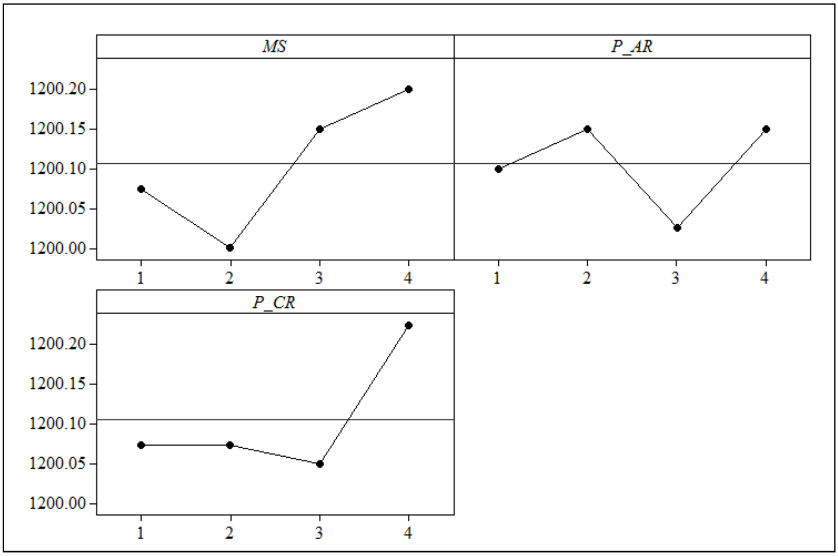

Fig. 3 Main Effect Plot

From Table 4, it can be seen that *MS* is the most significant to the IHS. A too small value of *MS* is harmful to the diversity of the harmony memory, while a too large value may slow down the convergence. The influence of P_{AR} ranks the second. Although a large value of P_{AR} is beneficial to find a proper factory for a job, it costs more computational time due to the greedy way of the pitch adjustment rule. Besides, a large value of P_{CR} is helpful to use the information of the explored solutions, but it may lead to a premature convergence. According to the above analysis, a good choice of parameter combination is recommended as *MS* = 10, P_{CR} = 0.8 and P_{AR} = 0.8, which will be used in the following tests.

4.3 Comparative Results

To the best of our knowledge, there is no published work addressing the DTMFSP. Therefore, we compare the IHS with the original HS and the global-best HS (GHS) [30]. For the HS, we set $MS = 5$, $P_{CR} = 0.9$, $P_{AR} = 0.3$ and $dB = 1/f$. For the GHS, we set $MS = 5$, $P_{CR} = 0.9$, ${P_{AR}}^{\min} = 0.01$ and ${P_{AR}}^{\max} = 0.99$ as suggested in [30].

For each instance, the above algorithms are run 10 times independently. The Best, average (Ave) and standard deviation (SD) values obtained by the algorithms are listed in Tables 5-9 in solving the problems with different number of factories.

From Tables 5-9, it can be seen that the IHS outperforms the HS and the GHS, especially when $f > 3$. As f increases, the factory assignment becomes more complex and it is more difficult to find proper factories for jobs. Since the IHS can obtain better Best and Ave results, it can be concluded that the proposed new pitch adjustment rule is effective to adjust the factory assignment in solving the DTMFSP.

Besides, the SD values of the IHS are also better than those of the HS and the GHS, which implies that the IHS is more robust than the HS and GHS.

Table 5 Results of the Algorithms ($f = 2$)

Instance		IHS			HS			GHS		
No.	***n***	**Best**	**Ave**	**SD**	**Best**	**Ave**	**SD**	**Best**	**Ave**	**SD**
1	20	582	582	0.00	582	582	0.00	582	582	0.00
2	20	564	564	0.00	564	564	0.00	564	564	0.00
3	20	529	529	0.00	529	529	0.00	529	529	0.00
4	20	529	529	0.00	529	529	0.00	529	529	0.00
5	20	497	497	0.00	497	497	0.00	497	497	0.00
6	50	979	979	0.00	979	979	0.00	979	979	0.00
7	50	1319	1319	0.00	1319	1319	0.00	1319	1319	0.00
8	50	1356	1356	0.00	1356	1356	0.00	1356	1356	0.00
9	50	1267	1267	0.00	1267	1267	0.00	1267	1267	0.00
10	50	1238	1238	0.00	1238	1238	0.00	1238	1238	0.00
11	100	1519	1519	0.00	1519	1519	0.00	1519	1519	0.00
12	100	2389	2389	0.00	2389	2389	0.00	2389	2389	0.00
13	100	2721	2721	0.00	2721	2721	0.00	2721	2721	0.00
14	100	2616	2616	0.00	2616	2616	0.00	2616	2616	0.00
15	100	2614	2614	0.00	2614	2614	0.00	2614	2614	0.00
16	200	2688	2688	0.00	2688	2688	0.00	2688	2688	0.00
17	200	5312	5312	0.00	5312	5312	0.00	5312	5312	0.00
18	200	5257	5257	0.00	5257	5257	0.00	5257	5257	0.00
19	200	5023	5023	0.00	5023	5023	0.00	5023	5023	0.00
20	200	5296	5296	0.00	5296	5296	0.00	5296	5296	0.00

Table 6 Results of the Algorithms (*f* = 3)

Instance		IHS			HS			GHS		
No	***n***	**Best**	**Ave**	**SD**	**Best**	**Ave**	**SD**	**Best**	**Ave**	**SD**
1	20	393	393	0.00	393	393.5	0.50	393	393.5	0.50
2	20	360	360	0.00	360	360	0.00	360	360	0.00
3	20	362	362	0.00	362	362.1	0.30	362	362.1	0.30
4	20	401	401	0.00	401	401	0.00	401	401	0.00
5	20	342	342	0.00	342	342.8	0.75	342	342.9	1.14
6	50	854	854	0.00	854	854.4	0.49	854	854.3	0.64
7	50	817	817	0.00	817	817.9	0.30	817	817.7	0.46
8	50	924	924	0.00	924	925.1	0.54	924	925.1	0.70
9	50	908	908	0.00	908	908	0.00	908	908	0.00
10	50	775	775	0.00	775	775.8	0.60	775	775.4	0.49
11	100	1700	1700	0.00	1700	1700	0.00	1700	1700	0.00
12	100	1873	1873	0.00	1873	1873.	0.49	1873	1873.8	0.60
13	100	1664	1664	0.00	1664	1664.	0.40	1664	1664.2	0.40
14	100	1678	1678	0.00	1678	1680.	1.36	1678	1679.4	1.02
15	100	1682	1682	0.00	1682	1682.	0.46	1682	1682.4	0.49
16	200	3575	3575	0.00	3575	3575	0.00	3575	3575	0.00
17	200	3614	3614	0.00	3614	3614.	0.40	3614	3614.2	0.40
18	200	3468	3468	0.00	3469	3469.	0.83	3468	3469	0.63
19	200	3559	3559	0.00	3559	3560.	0.54	3559	3559.8	0.40
20	200	3449	3449	0.00	3449	3450	0.63	3449	3450.1	0.54

Table 7 Results of the Algorithms (*f* = 4)

Instance		IHS			HS			GHS		
No	***n***	**Best**	**Ave**	**SD**	**Best**	**Ave**	**SD**	**Best**	**Ave**	**SD**
1	20	238	238.2	0.40	239	242.2	2.23	240	242.1	1.37
2	20	259	259.8	0.40	260	263.5	3.07	263	264.9	1.04
3	20	303	303	0.00	303	304.8	1.40	304	305.8	1.40
4	20	313	313.3	0.46	313	313.4	0.66	313	313.4	0.49
5	20	291	291	0.00	291	292.3	0.90	291	293.3	2.61
6	50	735	735	0.00	737	738.4	0.80	737	737.6	0.80
7	50	630	630	0.00	634	638.5	2.33	633	636.7	2.53
8	50	583	583	0.00	585	589.1	2.26	585	588.2	1.78
9	50	610	610	0.00	612	613	0.77	611	612.9	1.04
10	50	666	666	0.00	669	671.8	1.54	670	671.5	1.28
11	100	1200	1200.	0.40	1201	1202.	1.17	1200	1202.8	1.66
12	100	1293	1293	0.00	1295	1296.	1.30	1295	1296.9	1.14
13	100	1304	1304.	0.50	1305	1307.	1.42	1305	1307.5	1.28
14	100	1305	1305	0.00	1309	1310.	1.00	1306	1309.4	2.15
15	100	1277	1277	0.00	1281	1281.	0.75	1279	1281	0.89
16	200	2439	2439	0.00	2443	2449.	3.45	2443	2449.4	4.20
17	200	2456	2456.	0.49	2460	2465.	3.64	2458	2465.3	4.71
18	200	2529	2529.	0.54	2532	2536.	2.79	2533	2535.9	2.12
19	200	2448	2448	0.00	2449	2451.	1.19	2450	2452.3	1.49
20	200	2542	2542.	0.40	2544	2547.	2.06	2544	2547.5	1.50

Table 8 Results of the Algorithms ($f = 5$)

Instance		IHS			HS			GHS		
No	***n***	**Best**	**Ave**	**SD**	**Best**	**Ave**	**SD**	**Best**	**Ave**	**SD**
1	20	227	227.3	0.46	229	232	2.14	231	233.4	2.24
2	20	234	235	0.45	236	239.3	1.55	235	240.7	4.12
3	20	244	244	0.00	244	246.6	2.62	244	247.2	2.71
4	20	272	22.2	0.40	275	277.9	2.47	275	277.8	2.32
5	20	234	234.9	0.30	236	240.4	2.65	238	239.9	1.22
6	50	545	545.8	0.60	547	552.4	2.33	546	551.4	2.06
7	50	537	537.8	0.40	547	553.8	3.19	544	548.6	3.67
8	50	519	519.9	0.54	525	525.5	0.67	522	524.2	1.94
9	50	534	534.6	0.49	542	548.7	3.85	542	548.5	3.64
10	50	465	465	0.00	473	478.9	3.11	469	477	4.27
11	100	1007	1007	0.00	1024	1030.	4.40	1029	1031.4	2.11
12	100	1044	1044	0.00	1057	1064.	4.17	1060	1063.3	2.37
13	100	982	982	0.00	988	989.9	1.30	985	990.1	3.18
14	100	988	988.8	0.75	1004	1009.	2.97	1005	1011	3.85
15	100	1092	1092.	0.66	1096	1100.	2.47	1097	1100.3	2.37
16	200	2079	2079	0.00	2086	2090.	2.99	2089	2093.4	3.47
17	200	2131	2132	0.45	2140	2151.	5.63	2143	2153.1	5.91
18	200	2087	2087	0.00	2094	2098.	3.99	2095	2098.4	2.69
19	200	2050	2050	0.00	2073	2083.	5.16	2069	2082.7	6.94
20	200	2017	2017	0.00	2029	2037	4.15	2029	2039.2	5.53

Table 9 Results of the Algorithms ($f = 6$)

Instance		IHS			HS			GHS		
No	***n***	**Best**	**Ave**	**SD**	**Best**	**Ave**	**SD**	**Best**	**Ave**	**SD**
1	20	173	173	0.00	173	177.5	3.29	174	177.8	3.22
2	20	235	236.4	0.66	235	238.2	3.49	235	239.4	2.73
3	20	194	194	0.00	194	197.5	3.91	194	197.6	2.65
4	20	218	220.5	1.20	219	224.5	3.04	219	223.4	2.58
5	20	232	232.1	0.30	234	237.3	2.15	236	237.9	1.76
6	50	438	439	0.77	449	453.1	2.62	443	449.6	2.84
7	50	426	427.6	1.02	433	439.3	3.00	428	435.5	4.84
8	50	391	391.7	0.64	406	414	3.49	405	410.1	4.01
9	50	442	442	0.00	462	465	1.61	459	461.8	1.94
10	50	459	461.5	1.28	474	477.9	3.27	465	472.3	4.05
11	100	904	904.6	0.49	925	936.1	4.89	926	934.9	4.93
12	100	787	787	0.00	798	806.9	5.59	796	807.7	5.59
13	100	867	867	0.00	880	884.9	3.42	882	886.9	2.51
14	100	874	874.1	0.30	901	911	5.78	900	910.1	4.68
15	100	878	879.1	0.70	915	916.7	1.19	904	916.9	5.80
16	200	1786	1786.	0.49	1801	1805.	5.17	1799	1805.8	4.19
17	200	1724	1724	0.00	1740	1744.	3.11	1739	1745.3	4.05
18	200	1662	1662	0.00	1698	1703.	3.46	1689	1700.2	6.65
19	200	1707	1709.	1.02	1724	1737.	8.75	1731	1739.7	5.51
20	200	1677	1677.	0.50	1717	1728	5.23	1715	1728	6.84

5 Conclusions

This paper proposed an improved harmony search algorithm for solving the distributed two machine flow-shop scheduling problem. According to the characteristics of the problem, a two-stage decoding rule for the decimal vector based representation and a new pitch adjustment rule for the HS were designed. The influence of parameter setting was investigated, and the effectiveness of the IHS was demonstrated by numerical comparisons. Future work could focus on generalizing the harmony search algorithm for other types of the distributed shop scheduling problems and developing multi-objective HS algorithms for the problems with multiple scheduling criteria.

Acknowledgment This research is partially supported by the National Key Basic Research and Development Program of China (No. 2013CB329503), the National Science Foundation of China (No. 61174189), and the Doctoral Program Foundation of Institutions of Higher Education of China (20130002110057).

References

1. Johnson, S.M.: Optimal two- and three-stage production schedules with setup times included. Nav. Res. Logist. Quart. **1**, 61–68 (1954)
2. Ku, P.S., Niu, S.C.: On Johnson's two-machine flow shop with random processing times. Oper. Res. **34**, 130–136 (1986)
3. The two-machine total completion time flow shop problem: Della Croce, F., Narayan, V., Tadei, R. Eur. J. Oper. Res. **90**, 227–237 (1996)
4. Sayin, S., Karabati, S.: Theory and Methodology: a bicriteria approach to the two-machine flow shop scheduling problem. Eur. J. Oper. Res. **113**, 435–449 (1999)
5. Croce, Della: F., Ghirardi, M., Tadei, R.: An improved branch-and-bound algorithm for the two machine total completion time flow shop problem. Eur. J. Oper. Res. **139**, 293–301 (2002)
6. Pan, J.C.H., Chen, J.S., Chao, C.M.: Minimizing tardiness in a two-machine flow-shop. Comput. Oper. Res. **29**, 869–885 (2002)
7. Toktaş, B., Azizoğlu, M., Köksalan, S.K.: Two-machine flow shop scheduling with two criteria: maximum earliness and makespan. Eur. J. Oper. Res. **157**, 286–295 (2004)
8. Blazewicz, J., Pesch, E., Sterna, M., Werner, F.: Metaheuristic approaches for the two-machine flow-shop problem with weighted late work criterion and common due date. Comput. Oper. Res. **35**, 574–599 (2008)
9. Ng, C.T., Wang, J.B., Cheng, T.C.E., Liu, L.L.: A branch-and-bound algorithm for solving a two-machine flow shop problem with deteriorating jobs. Comput. Oper. Res. **37**, 83–90 (2010)
10. Kasperski, A., Kurpisz, A., Zieliński, P.: Approximating a two-machine flow shop scheduling under discrete scenario uncertainty. Eur. J. Oper. Res. **217**, 36–43 (2012)
11. Jia, H.Z., Nee, A.Y.C., Fuh, J.Y.H., Zhang, Y.F.: A modified genetic algorithm for distributed scheduling problems. J. Intell. Manuf. **14**, 351–362 (2003)

12. Moon, C., Seo, Y.: Evolutionary algorithm for advanced process planning and scheduling in a multi-plant. Comput. Ind. Eng. **48**, 311–325 (2005)
13. Chan, F.T.S., Chung, S.H., Chan, P.L.Y.: Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems. Int. J. Prod. Res. **44**, 523–543 (2006)
14. Naderi, B., Ruiz, R.: The distributed permutation flowshop scheduling problem. Comput. Oper. Res. **37**, 754–768 (2010)
15. Behnamian, J.: Fatemi Ghomi, S.M.T.: The heterogeneous multi-factory production network scheduling with adaptive communication policy and parallel machine. Inf. Sci. **219**, 181–196 (2013)
16. Hatami, S., Ruiz, R., Andrés-Romano, C.: The distributed assembly permutation flowshop scheduling problem. Int. J. Prod. Res. **51**, 5292–5308 (2013)
17. Wang, S.Y., Wang, L., Liu, M., Xu, Y.: An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. Int. J. Prod. Econ. **145**, 387–396 (2013)
18. Naderi, B., Azab, A.: Modeling and heuristics for scheduling of distributed job shops. Expert Syst. Appl. **41**, 7754–7763 (2014)
19. Garey, M.R., Johnson, D.S.: "Strong" NP-completeness results: motivation, examples, and implications. J. ACM **25**, 499–508 (1978)
20. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization algorithm: harmony search. Simul. **76**, 60–68 (2001)
21. Lee, K.S., Geem, Z.W.: A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput. Methods Appl. Mech. Eng. **194**, 3902–3933 (2005)
22. Geem, Z.W., Lee, K.S., Park, Y.: Application of harmony search to vehicle routing. Am. J. Appl. Sci. **2**, 1552 (2005)
23. Kaveh, A., Talatahari, S.: Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput. Struct. **87**, 267–283 (2009)
24. Geem, Z.W.: Particle-swarm harmony search for water network design. Eng. Optim. **41**, 297–311 (2009)
25. Rao, Srinivasa: R., Narasimham, S.V.L., Ramalinga Raju, M., Srinivasa Rao, A.: Optimal network reconfiguration of large-scale distribution system using harmony search algorithm. IEEE Trans. Power Syst. **26**, 1080–1088 (2011)
26. Wang, L., Pan, Q.K., Tasgetiren, M.F.: A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. Comput. Ind. Eng. **61**, 76–83 (2011)
27. Lee, K.S., Geem, Z.W., Lee, S.H., Bae, K.W.: The harmony search heuristic algorithm for discrete structural optimization. Eng. Optim. **37**, 663–684 (2005)
28. Mahdavi, M., Fesanghary, M., Damangir, E.: An improved harmony search algorithm for solving optimization problems. Appl. Math. Comput. **188**, 1567–1579 (2007)
29. Baker, K.R.: Introduction to Sequencing and Scheduling. Wiley, New York (1974)
30. Omran, M.G.H., Mahdavi, M.: Global-best harmony search. Appl. Math. Comput. **198**, 643–656 (2008)