An improved self-adaptive harmony search algorithm for joint replenishment problems

Lin Wang

School of Management, Huazhong University of Science & Technology
zhoulearner@gmail.com

Xiaojian Zhou

School of Management, Huazhong University of Science & Technology

Abstract
To solve joint replenishment problems (JRPs) effectively and efficiently which are typical NP-hard problems, an improved self-adaptive harmony search algorithm (ISHS) is designed. The proposed algorithm applies differential evolution mutation strategies to HS. Experimental results show that ISHS outperforms other start-of-art algorithms on both tested benchmark functions and typical JRPs.

Keywords: Joint Replenishment Problems, Harmony Search, Function Optimization

INTRODUCTION

Complex optimization problems are frequently confronted by researchers and practitioners in various fields. Researchers are looking for effective methods to solve difficult optimization problems. Harmony search (HS) is a relatively new meta-heuristic algorithm, which is inspired by the improvisation process of music players, proposed by Geem et al. (2001). Thanks to the advantages of HS and its variants, they have been wildly applied to various practical fields, including medicine, engineering, job scheduling and many others.

In recent years, there has been an increasing interest in improving HS algorithms. Mahdavi et al. (2007) introduced an improved HS (IHS) to overcome the defect of fixed parameters by dynamically changing the key parameters of the algorithms. Omran and Mahdavi (2008) designed a pitch adjustment rule to utilize the best solution in the harmony memory (HM) and introduced a global-best harmony search (GHS). Pan et al. (2010) developed a self-adaptive global-best harmony search (SGHS) which has a new improvisation rule and in which the parameters are dynamically changed. To further overcome the weakness of choosing suitable parameters for HS, Geem and Sim (2010) improved HS by integrating a parameter-setting-free (PSF) technique and proposed PSF-HS. Yadav et al. (2012) introduced an intelligent tuned HS algorithm which employed several concepts from the decision making theory. El-Adb (2013) combined a new improvisation scheme with a mechanism for adjusting key parameters for HS and proposed an improved global-best harmony search (IGHS). Khalili et al. (2014) managed to eliminate the trouble of selecting value of any parameter for HS and introduced the global dynamic HS (GDHS). Though improvements are made, the previous variants of HS still have some weakness, such as unsatisfactory solution quality and not being good at optimization complex problems. Also, there are seldom research about hybridizing differential evolution (DE) and HS.

To overcome the previous weakness and investigate the potential of hybridizing DE and HS, this paper introduces an improved self-adaptive HS (ISHS in short). The proposed ISHS hybridizes DE and HS by applying a DE mutation strategy to its improvisation procedure, and enhances its convergence speed by making use of best solution found. Besides, we design a learning mechanism to adjust the scale factor of DE mutation operator to overcome the weakness of a fixed scale factor. The ISHS also dynamically changes its key parameters in the search process to balance its exploitation power and exploration power.

The remainder of this paper is arranged as follows: Section 2 describes the principles and procedure of the basic harmony search. Section 3 introduces the proposed ISHS in detail. Section 4 presents a test to evaluate the performance of ISHS on benchmark functions while compared with state-of-art algorithms. Section 5 provides the application of ISHS to joint-replenishment problems (JRPs) and its results and analysis. Finally, conclusions are shown in the last section.

ORIGINAL HARMONY SEARCH

In this section, we will introduce the basic HS in detail. The procedure of the original HS are as follows (Lee and Geem, 2005):

Step 1. Initialize the optimization problem and algorithm parameters. The first step to solve a problem is to define the problem. The optimization problems are often defined as follows:

$$Minimize\ f(x)\quad s.t.\ x_i \in X_i,\quad i = 1,2,\dots,N \tag{1}$$

where f(x) is the objective function; x is the best solution of the problem; X is the possible range of x; and N is the number of dimensions of design variable x. Besides, the parameters for the algorithm should also be set in this step, including the harmony memory size (HMS), harmony memory consideration rate (HMCR), pitch adjustment rate (PAR), bandwidth (BW), and the termination criterion.

Step 2. Initialize the harmony memory (HM). In this step, the HM is randomly initialized in the original HS algorithm. HM is a matrix as shown in Eq. 2.

$$HM = \begin{bmatrix} x^1 \\ x^2 \\ \dots \\ x^{HMS} \end{bmatrix} \tag{2}$$

Step 3. Improvise a new harmony from the HM. In this step, a new harmony $x' = \{x'_1, x'_2, \dots, x'_N,\}$ is improvised. There are three key operators in the improvisation procedure, which are pitch selection (Operator 1 in Algorithm 1), pitch adjustment (Operator 2 in Algorithm 1) and random selection (Operator 3 in Algorithm 1). The improvisation procedure is shown in Algorithm 1.

Algorithm 1. Improvisation procedure of the original HS

```
1: for j = 1:D
2:     if rand()≤HMCR
3:         $x'_j = HM^i_j$;   // Opaerator 1
4:         if rand()≤PAR
```

```
5:            $x_j' = x_j' + rand() \times BW$;   //  Operator 2
6:         end
7:     else
8:         $x_j^i = (ub_j - lb_j) \times rand() + lb_j$;   //  Operator 3
9:     end
10: end
i is a randomly selected number in $[1,2,\ldots,HMS]$
```

Step 4. Update the HM. The improvised x' is evaluated by corresponding value of the objective function. If x' is better than the worst harmony in the HM, the new harmony x' replaces the worst harmony in the HM.

Step 5. Repeat Steps 3 and 4 until the termination criterion is satisfied. If the termination criterion is satisfied, the algorithm terminates. If not, go back to Step 3.

THE PROPOSED ISHS

In this section, we introduced the proposed ISHS algorithm.

Hybrid Improvisation Scheme

The improvisation procedure is shown in algorithm 2.

Algorithm 2. Improvisation procedure of ISHS

```
1:  for j = 1:$D$
2:     if rand()≤HMCR
3:         $x_j' = HM_j^{best} + F1 \times (HM_j^a + HM_j^b - HM_j^c - HM_j^d)$;   // Operator 1
4:         if rand()≤PAR
5:            $x_j' = HM_j^e$;   //  Operator 2
6:         end
7:     else
8:         $x_j' = HM_k^{best}$   //  Operator 3
9:     end
10: end
$a, b, c, d, e$ are randomly selected numbers in $[1,2,\ldots,HMS]$
$k$ is a randomly selection number in $[1,2,\ldots,N]$
```

To enhance the convergence ability of HS, we design a novel improvisation scheme that applies a popular DE mutation strategy. As described in the "Original Harmony Search" section, none of the three key operators of the original HS makes use of the best solution specifically in the HM. On the other hand, DE algorithm often makes use of the best solution found so far and shows a fast convergence (Storn & Price, 1997). Among the DE mutation strategies, DE/best/2/bin is one of the most commonly used one because it has been proven to have a good convergence capacity (Qin et al., 2009). Consequently, we hybridize DE and HS by applying this mutation strategy to the improvisation scheme of ISHS as shown in Operator 1 in Algorithm 2.

In GHS proposed by Omran and Mahdavi (2008), a novel operator, which assigns the value of a random dimension of best solution in the HM to a specific dimension of the new harmony, improves the optimization ability of HS. This operator (Operator 3 in Algorithm 2) is thus integrated into our novel improvisation scheme.

As both two key operators mentioned above utilize the best solution in the HM, which are focusing on improving exploitation capacity, we design an operator (Operator 2 in Algorithm 2) that emphasizes the exploration capacity of the algorithm.

Self-Adaptive Mechanism

In most of research about DE algorithms that utilizes DE/best/2/bin mutation strategy, the scale factor F1 is fixed or dynamically adjusted according to preset equations. Those means fail to benefit from the information gained throughout the evolution process. To overcome this weakness, we design a self-adaptive learning mechanism for ISHS. We assume that the F1 is normally distributed with mean F1m and standard deviation 0.1. F1m is initialized as 0. During each learning period, which is 100 generations in ISHS, the generated F1 is recorded if the improvised new harmony enters HM. In the first generation of each learning period, F1m is updated as the average of all the recorded F1 in the last learning period.

Dynamic Changing HMCR and PAR

To balance exploitation power and exploration power through the search process, the key parameters, i.e. HMCR and PAR, are dynamic changing according to Eqs. 3 and 4.

$$HMCR(t) = HMCR_{min} + (HMCR_{max} - HMCR_{min})\times t/NI \tag{3}$$

$$PAR(t) = PAR_{max} + (PAR_{max} - PAR_{min})\times t/NI \tag{4}$$

TEST BY BENCHMARK FUNCTIONS AND ANALYSIS

To fully evaluation the performance of the proposed ISHS, it is applied to 27 widely-used benchmark functions (shown in Table 1), which are extracted from previous research (Gao and Liu, 2007; Wang et al., 2015). ISHS is compared with 6 algorithms: the basic HS, IHS, GHS, SGHS, IGHS, and GDHS. The parameters for each algorithm is set as recommended by the original author, which are shown in Table 2. For a fair comparison, each algorithm is run with maximum number of iteration (NI) as 50000. The benchmark functions are tested with dimensions of 30 and 50.

Table 1 – Benchmark functions

No.	Functions	x_i	$f(x^*)$
F1	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(cx_i)\right) + 20 + \exp(1)$	(-32.768, 32.768)	0
F2	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{D} i\left(2x_i^2 - x_{i-1}\right)^2$	(-10, 10)	0
F3	$f(x) = \sum_{i=1}^{D} x_i^2/4000 - \prod_{i=1}^{D}\cos\left(x_i^2/\sqrt{i}\right) + 1$	(-600, 600)	0

F4	$f(x) = sin^2(\pi w_i) + \sum_{i=1}^{D-1}(w_i - 1)^2[1 + sin^2(\pi w_i + 1)] + (w_D - 1)^2[1 + sin^2(\pi w_D)], \quad where\ w_i = 1 + {x_i - 1}/_4$	(-10, 10)	0
F5	$f(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^4$	(-5, 10)	0
F6	$f(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{D}(j + \beta)\left(x_j^i - \frac{1}{j^i}\right)\right)^2$	(-D, D)	0
F7	$f(x) = \sum_{i=1}^{D-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	(-5, 10)	0
F8	$f(x) = \sum_{i=1}^{D}\sum_{j=1}^{i} x_j^2$	(-65.536, 65.536)	0
F9	$f(x) = \sum_{i=1}^{D} x_i^2$	(-5.12, 5.12)	0
F10	$f(x) = \frac{1}{2}\sum_{i=1}^{D}\left(x_i^4 - 16x_i^2 + 5x_i\right)$	(-5, 5)	-39.16599D
F11	$f(x) = \sum_{i=1}^{D} \lvert x_i \rvert^{i-1}$	(-1, 1)	0
F12	$f(x) = \sum_{i=1}^{D} ix_i^2$	(-5.12, 5.12)	0
F13	$f(x) = \sum_{i=2}^{D} ix_i^2$	(-5.12, 5.12)	0
F14	$f(x) = -\exp\left(-0.5\sum_{i=1}^{D} x_i^2\right)$	(-1, 1)	-1
F15	$f(x) = \sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}} x_i^2$	(-100, 100)	0
F16	$f(x) = \sum_{i=1}^{D-1}\left(\sum_{j=1}^{i} x_j\right)^2$	(-100, 100)	0
F17	$f(x) = \max(\lvert x_i \rvert)$	(-100, 100)	0
F18	$f(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	(-10, 10)	0
F19	$f(x) = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	(-10, 10)	0
F20	$f(x) = 1 - \cos\left(2\pi\left(\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}\right)$	(-5.12, 5.12)	0
F21	$f(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	(-3, 3)	-1.0316
F22	$f(x) = \sum_{i=1}^{D} x_i^2 + bias$	(-65.536, 65.536)	0
F23	$f(x) = \sum_{i=1}^{D} ix_i^4 + randdom[0, 1)$	(-1.28, 1.28)	0

F24	$f(x) = \frac{1}{10}\{sin^2(\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2[1 + sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + sin^2(2\pi x_{i+1})]\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	(-50, 50)	0
F25	$f(x) = \sum_{i=1}^{D-1}(x_i - 1)^2[1 + sin^2(3\pi x_{i+1})] + sin^2(3\pi x_1) + \lvert x_n - 1\rvert[1 + sin^2(3\pi x_n)]$		
F26	$f(x) = \frac{1}{n}\sum_{i=1}^{D}\left(x_i^4 - 16x_i^2 + 5x_i\right)$	(-5, 5)	-78.33236
F27	$f(x) = \sum_{i=1}^{D}\lvert z_i \times \sin(z_i) + 0.1 \times z_i\rvert,\ \ z = x - o$	(-10, 10)	0

Table 2 – Parameters for tested algorithms

Algorithms	HMS	HMCR	PAR	BW	Other	NI
HS	5	0.9	0.3	0.001	-	50000
IHS	5	0.9	PAR_{min}=0.01 PAR_{max}=0.99	BW_{min}=0.0001 BW_{max}=(UB-LB)/20	-	50000
GHS	5	0.9	PAR_{min}=0.01 PAR_{max}=0.99	-	-	50000
SGHS	5	$HMCR_m$=0.99	PAR_m=0.9	BW_{min}=0.0005 BW_{max}=(UB-LB)/10	LP=100	50000
IGHS	5	0.99	PAR_{min}=0.01 PAR_{max}=0.99	BW_{min}=0.0001 BW_{max}=(UB-LB)/20	-	50000
GDHS	5	-	-	-	-	50000
IDHS	50	$HMCR_{min}$=0.8 $HMCR_{max}$=0.99	PAR_{min}=0.01 PAR_{max}=0.1	-	-	50000

Each algorithm on each benchmark function is repeated for 50 times, and the mean and standard deviation is calculated. We employ a Wilcoxon rank sum test at a significance level of 5% to distinguish whether the difference between ISHS and other algorithm is significant or not. The test results of 1 and -1 mean that ISHS performs significantly better and worst than corresponding algorithm, while 0 means that there is no significant difference between them. For concision, the mean errors of the benchmark function optimization results are left out, and the comparison results are shown in Table 3.

Table 3 – results of Wilcoxon rank sum tests

	D = 30						D = 50					
Function	**HS**	**IHS**	**GHS**	**SGHS**	**IGHS**	**GDHS**	**HS**	**IHS**	**GHS**	**SGHS**	**IGHS**	**GDHS**
F1	1	1	1	1	1	1	1	1	1	1	1	1
F2	1	1	1	1	0	0	1	1	1	1	1	1
F3	1	1	1	1	1	1	1	1	1	1	1	0
F4	1	1	1	1	1	0	1	1	1	1	1	0
F5	1	1	1	1	1	0	1	1	1	1	1	1
F6	1	1	1	1	1	0	1	1	1	0	1	0

F7	1	1	1	1	0	0	1	1	1	1	1	1
F8	1	1	1	1	1	1	1	1	1	1	1	1
F9	1	1	1	1	1	1	1	1	1	1	1	1
F10	1	0	1	1	0	0	1	1	1	1	1	0
F11	1	1	1	1	1	1	1	1	1	1	1	1
F12	1	1	1	1	1	1	1	1	1	1	1	1
F13	1	1	1	1	1	1	1	1	1	1	1	1
F14	1	1	1	1	1	1	1	1	1	1	1	1
F15	1	1	1	1	1	1	1	1	1	1	1	1
F16	1	1	1	1	1	1	1	1	1	1	1	1
F17	0	0	0	0	0	-1	1	1	1	0	1	-1
F18	1	1	1	1	1	1	1	1	1	1	1	1
F19	1	1	1	1	1	1	1	1	-1	-1	1	1
F20	1	1	0	1	1	0	1	1	1	1	1	1
F21	0	0	1	0	0	0	1	0	1	0	0	0
F22	1	1	1	1	1	1	1	1	1	1	1	1
F23	0	0	0	0	0	0	1	1	0	0	0	0
F24	1	1	1	1	1	0	1	1	1	1	1	0
F25	1	1	1	1	1	0	1	1	1	1	1	1
F26	1	1	1	1	1	0	1	1	1	1	1	0
F27	1	1	1	1	1	0	1	1	1	0	1	1
appearing times of 1	24	23	24	24	21	13	27	26	25	21	25	18
appearing times of 0	3	4	3	3	6	13	0	1	1	5	2	8
appearing times of -1	0	0	0	0	0	1	0	0	1	1	0	1

Base on the results in Tables 3, the following conclusions can be drawn:

(a) ISHS significantly outperforms tested state-of-art HS variants. It can be observed from Table 3 that ISHS significantly outperforms HS/IHS/GHS/SGHS/IGHS/GDHS on 24/23/24/24/21/13 problems out of 27 30-dimensional problems, and on 27/26/25/21/25/18 problems out of 27 50-dimensional problems. On 30-dimensional problems, ISHS is never significantly worst than other algorithm on any problem, except for comparison with GDHS on benchmark function 17. On 50-dimensioanl problems, ISHS gets significantly worst results on only 3 problems in total compared with other 6 algorithms. It is safe to say that ISHS is significantly better than the tested state-of-art HS variants.

(b) The advantages of ISHS become clearer as the problems gets more complex. As the problems become more difficult with increasing number of dimensions, the number of benchmark functions on which ISHS outperforms HS, IHS, SGHS, IGHS, and GDHS increases by 3, 3, 1, 4 and 5. It is to say, ISHS has good ability to deal with complex problems.

(c) The proposed ISHS gets accurate optimization results. It can be observed from the mean error results (which are left out) that the errors gained by ISHS are smaller than those gained by other algorithms in most of the situations.

APPLICATION TO JRPS

Overview of JRPs

Joint replenishment problems (JRPs) are multi-item inventory problems of coordinating the replenishment of a group of items that could be jointly ordered from a single supplier (Wang et al., 2012). JRP models are also wildly applied to problems in manufacturing areas (Khouja and Goyal, 2008). This test applies the propose ISHS to JRPs under direct grouping strategy (DGS), the test problems are gathered directly from Wang et al. (2015). The JRP under DGS can be concisely modeled as a problem to minimize the total cost (TC) shown in Eq. 5.

$$TC(T_1, T_2, \dots, T_m) = \sum_{j=1}^{m} \left[(S + \sum_{i=1}^{n} s_i)/T_j + \frac{1}{2} T_j \sum_{i=1}^{n} D_i h_i \right] \tag{5}$$

T_j^*, the optimal value of T_j, can be gained by taking the deviation of TC with respect to T_j, as shown in Eq. 6.

$$T_j^* = \left[2 \left(S + \sum_{i \in G_j} s_i \right) / \sum_{i \in G_j} D_i h_i \right]^{1/2} \tag{6}$$

The meaning of notions in Eqs. 5 and 6 are as follows:

i index of item, $i = 1,2,\dots,n$

n number of items

D_i demand rate for item i (units/year)

S the major ordering cost

s_i the minor ordering cost of item i

h_i inventory holding cost of item i per year

T basic cycle time (years), time between ordering items, **decision variable**

T_i time between successive replenishment of item i in year, cycle time for item i

Experimental Settings and Results

Other than ISHS, the basic HS, GHS, IHS, SGHS, IGHS and GDHS are also applied to this problem. The parameters of algorithms in HS family are the same as those in Table 2, except that NI = 5000. In this experiment, we randomly generate 10 problems based on each of the 24 combinations from Wang et al. (2015), and then each of the algorithms are executed on the generated problems. The mean value of total costs is calculated, and then the average cost saving of ISHS against other algorithm is calculated by Eq. 7.

$$ACS = \frac{TC(average)_{other\ algorithm} - TC(average)_{ISHS}}{TC(average)_{other\ algorithm}} \times 100\% \tag{7}$$

For concision, the average total costs are left out, and the ACS results are shown in Tables 5.

Table 5 – ACS of ISHS against other algorithm

Combination	n	S	s_i	D_i	h_i	HS	GHS	IHS	SGHS	IGHS	GDHS
1	10	5	[3,10]	[500,1000]	[2,5]	1.28%	0.00%	1.28%	1.59%	1.56%	0.80%
2	10	5	[3,10]	[500,1000]	[5,10]	0.00%	0.00%	0.00%	0.05%	1.22%	0.00%
3	30	5	[3,10]	[500,1000]	[2,5]	2.85%	0.48%	2.96%	4.62%	3.28%	2.45%

4	30	5	[3,10]	[500,1000]	[5,10]	2.85%	0.91%	2.78%	4.60%	3.30%	2.02%
5	30	5	[3,10]	[300,1500]	[2,5]	2.83%	1.68%	2.94%	4.94%	3.35%	1.86%
6	30	5	[3,10]	[300,1500]	[5,10]	2.62%	1.87%	2.79%	4.50%	3.42%	1.95%
7	30	5	[3,10]	[200,2000]	[2,5]	2.99%	2.51%	3.33%	5.51%	3.71%	2.30%
8	30	5	[3,10]	[200,2000]	[5,10]	2.85%	2.40%	3.00%	5.02%	3.51%	2.17%
9	30	5	[11,20]	[500,1000]	[2,5]	1.45%	1.13%	1.60%	2.47%	1.90%	1.26%
10	30	5	[11,20]	[500,1000]	[5,10]	1.28%	0.24%	1.38%	2.03%	1.56%	1.10%
11	30	5	[11,20]	[300,1500]	[2,5]	1.31%	1.20%	1.30%	2.30%	1.74%	0.90%
12	30	5	[11,20]	[300,1500]	[5,10]	1.30%	1.00%	1.44%	2.56%	1.56%	0.98%
13	30	5	[11,20]	[200,2000]	[2,5]	1.17%	1.78%	1.20%	2.51%	1.74%	0.92%
14	30	5	[11,20]	[200,2000]	[5,10]	1.12%	1.58%	1.36%	2.32%	1.57%	0.98%
15	50	5	[3,10]	[500,1000]	[2,5]	4.77%	1.14%	4.63%	5.08%	3.14%	4.42%
16	50	5	[3,10]	[500,1000]	[5,10]	4.19%	0.47%	4.13%	4.64%	2.70%	3.74%
17	50	5	[3,10]	[300,1500]	[2,5]	4.39%	2.88%	4.52%	5.19%	3.18%	4.17%
18	50	5	[3,10]	[300,1500]	[5,10]	3.82%	1.82%	4.11%	4.65%	2.99%	3.93%
19	50	5	[11,20]	[500,1000]	[2,5]	2.03%	0.59%	1.98%	2.39%	1.44%	1.78%
20	50	5	[11,20]	[500,1000]	[5,10]	1.93%	0.12%	1.89%	2.18%	1.40%	1.88%
21	50	20	[11,20]	[500,1000]	[2,5]	7.09%	0.51%	7.26%	7.71%	5.27%	6.55%
22	50	20	[11,20]	[500,1000]	[5,10]	7.38%	0.97%	7.49%	7.91%	5.30%	7.19%
23	50	20	[11,20]	[200,2000]	[2,5]	6.30%	1.41%	6.66%	6.89%	4.24%	5.96%
24	50	20	[11,20]	[200,2000]	[5,10]	6.91%	0.88%	6.93%	7.91%	4.95%	6.84%

Experimental Analysis

The following conclusions can be drawn from Tables 5:

(a) ISHS always gets smaller average total cost than other tested algorithms. It can be observed from Table 5 that ISHS always gets smaller average total cost than HS, GHS, HIS, SGHS, IGHS and GDHS. The optimization results gained by ISHS is significantly better than other tested algorithms, which can lead to better decisions and save costs in practical situations.

(b) The advantages of ISHS tend to become greater the number of items gets larger. The optimization becomes more difficult as the the number of items increases. At the same time, the advantages of ISHS become clearer when the problems get more complex, especially compared with HS, GHS, SGHS, IGHS and GDHS.

CONCLUSIONS AND FUTURE RESEARCH

This paper introduces a novel variant of harmony search, called improved self-adaptive harmony search (ISHS). The proposed ISHS has a newly-designed improvisation scheme that integrates a DE mutation strategy, utilizes a learning mechanism for scale factor, and balances it capacity of exploration and capacity of exploitation by dynamically changing it key parameters.

The proposed ISHS is tested by 27 benchmark functions in comparison with six other state-of-art algorithms. The test shows that ISHS has excellent optimization ability and significantly outperforms other algorithms on benchmark functions. Beside, ISHS is successfully applied to a joint replenishment problem. ISHS gets better results than other six tested algorithms which can save a significant number of cost when applied to practical problems. Both of the experiments show that ISHS is especially good at solving complex optimization problems. Our future work will adapt the proposed ISHS to solve discrete optimization problems.

BIBLIOGRAPHY

El-Abd, M. (2013). An improved global-best harmony search algorithm. *Applied Mathematics and Computation*, *222*, 94–106.

Gao, W. F., & Liu, S. Y. (2012). A modified artificial bee colony algorithm. *Computers & Operations Research*, *39*(3), 687–697.

Geem, Z. W., Kim, J., & Loganathan, G. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, *76*(2), 60–68.

Geem, Z. W., & Sim, K. B. (2010). Parameter-setting-free harmony search algorithm. *Applied Mathematics and Computation*, *217*(8), 3881–3889.

Khalili, M., Kharrat, R., Salahshoor, K., & Sefat, M. H. (2014). Global dynamic harmony search algorithm: gdhs. *Applied Mathematics & Computation*, *228*(9), 195–219.

Khouja, M., & Goyal, S. (2008). A review of the joint replenishment problem literature: 1989–2005. *European Journal of Operational Research, 186*(1), 1-16.

Mahdavi, M., Fesanghary, M., & Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, *188*(2), 1567–1579.

Omran, M. G. H., & Mahdavi, M. (2008). Global-best harmony search. *Applied Mathematics and Computation*, *198*(2), 643–656.

Pan, Q. K., Suganthan, P. N., Tasgetiren, M. F., & Liang, J. J. (2010). A self-adaptive global best harmony search algorithm for continuous optimization problems. *Applied Mathematics and Computation*, *216*(3), 830–848.

Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *Evolutionary Computation, IEEE Transactions on*, *13*(2), 398–417.

Storn, R., & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*, 341–359.

Wang, L., He, J., & Zeng, Y. R. (2012). A differential evolution algorithm for joint replenishment problem using direct grouping and its application. *Expert Systems*, *29*(5), 429–441.

Wang, L., Shi, Y., & Liu, S. (2015). An improved fruit fly optimization algorithm and its application to joint replenishment problems. *Expert Systems with Applications*, *42*(9), 4310–4323.

Yadav, P., Kumar, R., Panda, S. K., & Chang, C. S. (2012). An intelligent tuned harmony search algorithm for optimisation. *Information Sciences*, *196*, 47-72.