Soft Comput
DOI 10.1007/s00500-014-1502-7

METHODOLOGIES AND APPLICATION

Hybridizing harmony search algorithm with cuckoo search for global numerical optimization

Gai-Ge Wang · Amir H. Gandomi · Xiangjun Zhao · Hai Cheng Eric Chu

Abstract For the purpose of enhancing the search ability of the cuckoo search (CS) algorithm, an improved robust approach, called HS/CS, is put forward to address the optimization problems. In HS/CS method, the pitch adjustment operation in harmony search (HS) that can be considered as a mutation operator is added to the process of the cuckoo updating so as to speed up convergence. Several benchmarks are applied to verify the proposed method and it is demonstrated that, in most cases, HS/CS performs better than the standard CS and other comparative methods. The parameters used in HS/CS are also investigated by various simulations.

Keywords Global optimization problem · Cuckoo search (CS) · Harmony search (HS) · Pitch adjustment operation

Communicated by V. Loia.

G.-G. Wang (✉) · X. Zhao
School of Computer Science and Technology,
Jiangsu Normal University, Xuzhou 221116, Jiangsu, China
e-mail: gaigewang@163.com; gaigewang@gmail.com

X. Zhao
e-mail: xjzhao@jsnu.edu.cn

A. H. Gandomi
Department of Civil Engineering,
The University of Akron,
Akron, OH 44325, USA
e-mail: a.h.gandomi@gmail.com; ag72@uakron.edu

H. C. E. Chu
National Taichung University of Education (NTCU),
140 MinSheng Rd., Taichung 40306, Taiwan, China
e-mail: ayura66@gmail.com

1 Introduction

Optimization is the process of searching for a vector in a given domain that makes the best solution among a large number of possible feasible solutions. The traditional optimization techniques hardly deal with modern complicated problems. Many scholars turn to learn from nature and various nature-inspired metaheuristic algorithms (Gandomi et al. 2013a) have been put forward and can often be applied to solve NP-hard problems, such as flow shop scheduling problem (Rahimi-Vahed and Mirzaei 2008; Li and Yin 2013a), parameter estimation (Li and Yin 2014), constrained optimization (Li and Yin 2012b), train neural networks (Mirjalili et al. 2012, 2014b), feature selection (Li and Yin 2013b) and frequency assignment problem (Luna et al. 2011). Metaheuristic algorithms are well capable to extract information from a set of solutions and will often generate the best solutions in practice. During the 1960s,a novel kind of optimization methods, called genetic algorithms (GAs) (Goldberg 1998; García-Martínez and Lozano 2010) was put forward by idealizing the evolution theory. Since then, many other metaheuristic algorithms have emerged, such as grey wolf optimizer (GSO) (Mirjalili et al. 2014a), particle swarm optimization (PSO) (Kennedy and Eberhart 1995; Gandomi et al. 2013b; Mirjalili and Lewis 2013), differential evolution (DE) (Storn and Price 1997; Gandomi et al. 2012; Li and Yin 2012a), interior search algorithm (ISA) (Gandomi 2014), bat algorithm (BA) (Gandomi et al. 2013c; Yang 2010a; Mirjalili et al. 2013), animal migration optimization (AMO) (Li et al. 2013d), krill herd (KH) (Gandomi and Alavi 2012; Wang et al. 2013a; Guo et al. 2014), biogeography-based optimization (BBO) (Simon 2008; Li et al. 2011; Wang et al. 2013b), and recently, the cuckoo search (CS) algorithm (Yang 2009; Yang and Deb 2010; Gandomi et al. 2013d) that is inspired by smart incubation behavior of cuckoos.

Published online: 03 December 2014

Developed by Yang and Deb in 2009, CS is a metaheuristic search technique (Yang 2009; Gandomi et al. 2013e), by simplifying and idealizing the brood parasitism behavior of certain kind of cuckoos. The aim of CS algorithm is to use the new and possibly better solutions to take the place of a relatively worse solution. When dealing with single-objective problems, each nest is only corresponding to one egg. One of the most important advantages of CS algorithm is its simplicity, therefore, it is very easy to implement. In principle, comparing with other metaheuristic algorithms such as PSO (Kennedy and Eberhart 1995) and HS (Geem et al. 2001), only discovery rate p_a needs to fine tune in CS method. On the other hand, by idealizing the musician' improvisation process, Geem et al. proposed harmony search (HS) (Geem et al. 2001) for addressing the optimization problems.

CS performs global search well in general cases, but at times it may still occasionally be trapped into some local optima. For CS, Lévy flight can fully determine the search, so it cannot always converge to the best solutions if the step sizes are too large. In the present work, to forbid premature convergence and increase the cuckoo population diversity, pitch adjustment operation in HS that can be a mutation operator is introduced into the CS method. Pitch adjustment is a banded local random walks, which focuses on a band or a local region of the feasible solutions. This may also be relevant in many practical applications because many design problems may already be near some optimal solutions if the designers can formulate the problem using their specific knowledge. Also, due to stringent design codes, new designs tend to be based on some existing designs by modifying some parameters locally. That is to say, the good features of HS and CS are fully exploited and combined into a new hybrid metaheuristic algorithm. Herein, an improved CS method is put forward and applied to search the optimal objective function value. The proposed approach is evaluated on 14 benchmark functions. Experimental results demonstrate that the HS/CS performs better than other nine methods for most benchmark cases.

The structure of this paper is organized as follows. Section 2 describes the HS algorithm, and basic CS in brief. The HS/CS approach is presented in Sect. 3. Subsequently, our method is evaluated through 14 benchmarks in Sect. 4 by comparing with eight other methods. Finally, a summarization of our present work is given in Sect. 5.

2 Preliminary

2.1 Harmony search

By simplifying and idealizing the natural musical improvisation processes, Geem et al. in 2001 put forward HS (Geem et al. 2001; Wang et al. 2014b) that is a relatively new metaheuristic search technique (Yang 2011). The basic HS algorithm includes the following operators: the harmony memory (HM) [see Eq. (1)], the harmony memory size (HMS), the harmony memory consideration rate (HMCR), the pitch adjustment rate (PAR) and the pitch adjustment bandwidth (bw).

$$\mathrm{HM} = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \ldots & x_D^1 & fitness(x^1) \\ x_1^2 & x_2^2 & \ldots & x_D^2 & fitness(x^2) \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \ldots & x_D^{\mathrm{HMS}} & fitness(x^{\mathrm{HMS}}) \end{array}\right] \quad (1)$$

In HS, there are three components: use of harmony memory, pitch adjusting, and randomization. In the HS optimization process, the value of each decision variable in HM can be determined by one of the above three rules. After the harmony updating, if the newly generated one has better fitness, it will be used to take place of the worst one in HM.This updating process is iterated until the satisfactory solution is found.

Like the selection of the optimal fit individuals in GA, the first part is significant in the whole HS process. This can guarantee that the best harmonies cannot be changed and make the HM always stay the best status. HMCR $\in$ [0, 1] should be carefully adjusted with the aim of using this memory more effectively. If it nears 1 (too high), almost all the harmonies in HM can be fully exploited, but the HS algorithm cannot search globally well, leading to potentially wrong solutions. In contrast, if it is very small (even 0), HS only takes use of few best harmonies, which may result in finding the best solutions slowly. Herein, generally, HMCR = 0.7–0.95.

For the second part, though the pitch can be slightly adjusted in the linear or nonlinear form theoretically, a linear adjustment is chosen in most cases. The pitch is updated as follows:

$$x_{\mathrm{new}} = x_{\mathrm{old}} + \mathrm{bw}(2\delta - 1) \quad (2)$$

where δ is a random number in [0,1], bw is the band width. x_{old} and x_{new} are the current and new pitch, respectively.

Pitch adjustment has the similarity with the mutation operator in evolutionary algorithms. Similarly, the PAR is also carefully adjusted with the aim of HS implementing in the best way. If PAR nears 1, the harmony in HM will sway even at the end of HS process, and HS is therefore hard to converge to the best solutions. Contrarily, if it is too low, then little change will be made for harmonies in HM and HS may converge prematurely. Therefore, we use PAR = 0.1–0.5 for most cases.

The third part is essentially a random process with the aim of adding harmony diversity. The random process makes the HS explore the whole search space well and this has a higher probability of finding the final optimal solutions. Due to its excellent performance, HS has been applied to deal with various optimization problems including linear antenna arrays, train neural network, flow shop scheduling, reliability problem, economic load dispatch, and others.

The detailed description of HS method can be found in Wang et al. (2013c).

2.2 Cuckoo search

By simplifying and idealizing the brood parasitic behavior of cuckoo individuals in combination with the Lévy flight, CS is put forward that is a novel metaheuristic search method for solving optimization problems. In the case of CS, how a cuckoo individual moves to next position is fully determined by the Lévy flights. More details about Lévy flights can be found in Yang (2010b).

To apply the cuckoo brood behavior for optimization problems, Yang and Deb idealized the brood parasitic behavior of some cuckoos and the following three rules have been put forward (Yang 2009).

1. In cuckoo population, each cuckoo puts one egg to a randomly selected nest;
2. the high-quality nests will not be changed, and this can guarantee the cuckoo population that includes the better solutions, not worse than before at least;
3. the nest number is unchanged, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0, 1]$.

In the simplest form, each nest is only responding to one egg. Accordingly, the CS method can be easily extended to deal with multi-objective optimization problems in which each nest includes more than one eggs/solutions. In our present study, we only consider that each nest has merely an egg. Therefore, in our study, we do not distinguish the difference among the nest, egg, and solution.

In HS method, the discovery rate p_a is considered as a switching parameter that is used to balance the random walk locally and globally. The local one can be given as

$$x_i^{t+1} = x_i^t + \beta s \otimes H(p_a - \varepsilon) \otimes (x_j^t - x_k^t) \tag{3}$$

where x_j^t and x_k^t are two different randomly selected solutions, $H(u)$ is a Heaviside function, ε is a random number, and s is the step size. While, the global one is performed using Lévy flights

$$\begin{aligned} x_i^{t+1} &= x_i^t + \beta L(s, \lambda), \\ L(s, \lambda) &= \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s, s_0 > 0) \end{aligned} \tag{4}$$

Here $\beta > 0$ is the step size scaling factor related to the scales of the problem of interest. More details about CS method can be found in Li et al. (2013a).

3 HS/CS

Based on the introduction of HS and CS in previous section, the detailed description of the proposed cuckoo search with harmony search (HS/CS) will be provided in this section.

In general, the standard CS algorithm explores the search space well and has a fast speed of finding the global optimal value, but it exploits solutions poorly due to occasionally large steps or moves. On the other hand, standard harmony search is well capable of exploiting solutions by carefully tuning HMCR and PAR. In the present work, by combination of CS and HS, a hybrid metaheuristic algorithm, so-called harmony search/cuckoo search (HS/CS), is therefore proposed for the purpose of optimizing the benchmark functions. In HS/CS method, the improvisation of harmony in HS is introduced into cuckoo search as mutation operator. In this way, this method can explore the new search space by the hybrid HS operator and exploit the population with CS, and therefore can fully exploit the advantages of the CS and HS.

The core idea of the HS/CS method is the introduction of the hybrid HS mutation operator. In this way, firstly presented in the current work, a main improvement of adding mutation operator is made to the CS including two minor improvements.

The first improvement is to add pitch adjustment operation in HS that can be considered as a mutation operator with the aim of adding increase population diversity. In the exploitation stage, once an individual is chosen among the current best individuals, a new cuckoo individual is generated globally using Lévy flights. And then, we fine tune every element in x_i using HS. When ξ is larger than HMCR, i.e., $\xi_1 \geq$ HMCR, the element j is updated randomly; while when $\xi_1 <$ HMCR, we update the element j in accordance with $x*$, moreover, pitch adjustment operation in HS that can be considered as a mutation operator is applied to update the element j when $\xi_2 <$ PAR in an attempt to add population diversity, as shown in Eq. (2), where ξ_1 and ξ_2 are two uniformly distributed random numbers in [0,1], $x*$ is the global best solution in the current generation. Through various experiments in Sect. 4.2, it was found that HMCR is set to 0.9 and PAR to 0.1 that can generate the optimal solutions.

The second improvement is the addition of elitism scheme into the HS/CS. As with other optimization algorithms, an

Algorithm 1 *The hybrid meta-heuristic algorithm of HS/CS*

```
Begin
    Step 1: Initialization. Set the generation counter $t = 1$; initialize the population $P$ of $NP$
            host nests randomly; set HMCR and PAR.
    Step 2: Evaluate the fitness $f$ for each cuckoo.
    Step 3: While $t < MaxGeneration$ do
              Sort the population as per fitness.
              Store the KEEP best cuckoos.
              for i=1:NP (all cuckoos) do
                  Get a cuckoo randomly (say, i) and replace its solution by Lévy flights.
                  $v = \lceil NP * rand \rceil$
                for j=1:D (all elements) do      // Mutate
                    if ($\xi_1 < HMCR$) then
                        $x_v(j) = x^*(j)$
                        if ($\xi_2 < PAR$) then
                            $x_v(j) = x_v(j) + bw \times (2 \times rand - 1)$
                        endif
                    else
                        $x_v(j) = x_{\min,j} + rand \times (x_{\max,j} - x_{\min-j})$
                    endif
                endfor j
                Evaluate the fitness for the offsprings $x_j^t, x_i^t, x_v^t$
                Select the best offspring $x_k^t$ among $x_j^t, x_i^t, x_v^t$.
                $x_j^t = x_k^t$
                Replace the KEEP worst cuckoos with the KEEP best cuckoos.
                Sort the population and find the current best.
                Pass the current best to the next generation.
              end for i
              $t = t+1$;
    Step 4: end while
End.
```

improved elitism scheme is incorporated into the HS/CS method to retain the best individuals in the cuckoo population. In the current work, we use a more focused elitism on the best solutions, which can prevent the best solutions from being corrupted by pitch adjustment operator. This can make the whole population always proceed to the better solutions.

As per the above description, the harmony search/cuckoo search (HS/CS) can be given in Algorithm 1 and its corresponding flowchart is shown in Fig. 1.

4 Simulations

In this section, the proposed metaheuristic HS/CS was tested through an array of experiments. To get a fair result, all the methods are implemented under the same conditions as shown in Wang et al. (2014a).

The benchmark functions described in Table 1 are standard testing functions. Further information of all the benchmark functions can be referred as Yao et al. (1999). In each table, the last row is the total number of functions on which the specified method performs the best under certain conditions. The best value achieved for each test problem is shown in bold.

4.1 General performance of HS/CS

For the purpose of verifying the benefits of HS/CS, we compared its performance on global optimization problem with nine other optimization methods, which are ACO (Dorigo

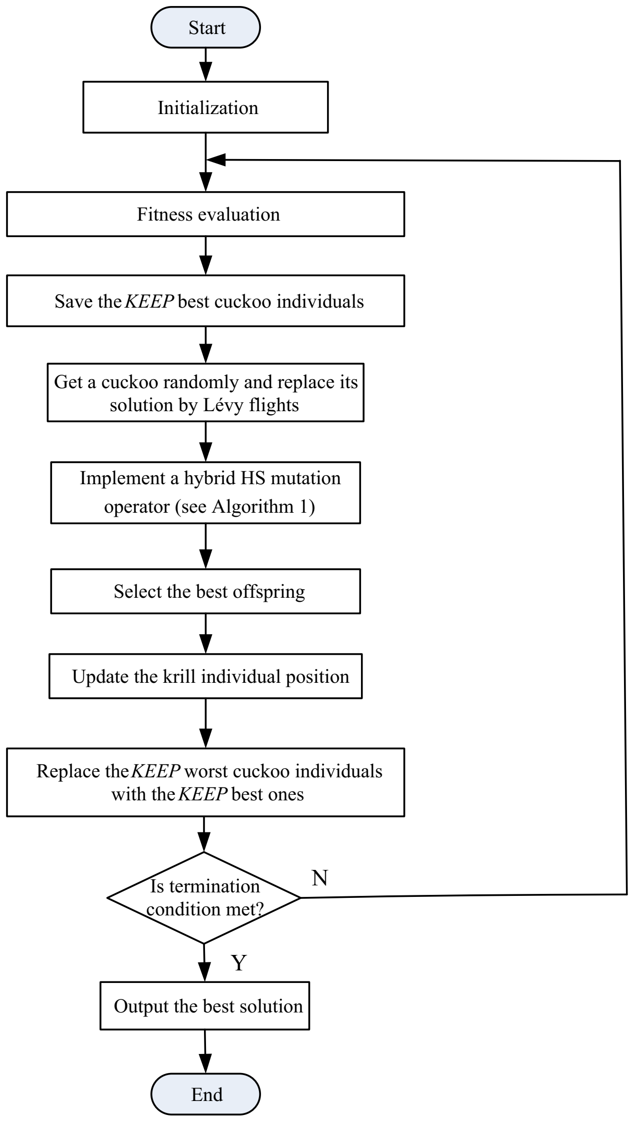
Start
Initialization
Fitness evaluation
Save the *KEEP* best cuckoo individuals
Get a cuckoo randomly and replace its solution by Lévy flights
Implement a hybrid HS mutation operator (see Algorithm 1)
Select the best offspring
Update the krill individual position
Replace the *KEEP* worst cuckoo individuals with the *KEEP* best ones
Is termination condition met?
N
Y
Output the best solution
End

Fig. 1 The flowchart of HS/CS algorithm

and Stutzle 2004), BBO (Simon 2008), CS (Yang 2009), DE (Storn and Price 1995), ES (Beyer 2001), GA (Goldberg 1998), HS (Geem et al. 2001), PSO (Kennedy and Eberhart 1995; Gandomi et al. 2012, 2013b, c, d, e; Mirjalili and Lewis 2013; Storn and Price 1997; Li and Yin 2012a; Gandomi 2014; Yang 2009, 2010a, b, 2011; Mirjalili et al. 2013; Li et al. 2013a, d; Gandomi and Alavi 2012; Wang et al. 2013a, b, c; Guo et al. 2014; Simon 2008; Li et al. 2011; Yang and Deb 2010; Geem et al. 2001; Wang et al. 2014a, b; Yao et al. 1999; Dorigo and Stutzle 2004; Storn and Price 1995; Beyer 2001; Doğan and Saka 2012), and SGA (Khatib and Fleming 1998).

The proposed HS/CS method in the present work is implemented by the authors in MATLAB. The ACO, BBO, DE, ES, GA, PSO and SGA are implemented by Simon (2008) (see http://academic.csuohio.edu/simond/bbo/). The CS and HS are implemented by Yang (2010b).

In all experiments, the same parameters for HS, CS and HS/CS are set to discovery rate $p_a = 0.25$, HMCR = 0.9, and PAR = 0.1. For ACO, BBO, DE, ES, GA, PSO and SGA, we set the parameters as shown in Wang et al. (2014c). For other parameters, population size NP, elitism number *Keep*, function dimension and maximum generation *Maxgen* are set 50, 2, 20 and 50, respectively. To decrease the influence of the randomness, 100 implementations are done for each algorithm on each benchmark function (see Tables 2, 3). To clarify the difference of the ten methods, different scales have been used to normalize the values in the tables.

From Table 2, we see that, on average, HS/CS has the strongest search ability of finding the function minimum on all the 14 benchmarks (F01–F14).

Table 3 shows that HS/CS performs the best on ten of the 14 benchmarks (F01–F03, F06–F09, F11, F13–F14). ACO ranks two and performs the best on F04 and F05. CS and SGA ranks three and performs the best on F10 and F12, respectively.

In addition, the running time is another important factor for the metaheuristic algorithms. The computational requirements are recorded in Table 2. BBO was the quickest method. HS/CS was the ninth fastest of the ten methods. However, in most practical applications, it is the fitness function evaluation that is the most time-consuming part of a population-based method.

Further, the most representative convergent curves are provided (see Figs. 2, 3, 4, 5, 6, 7, 8, 9, 10, 11). The values in the figures are the mean function optimum, which are the true values.

From Fig. 2, HSCS is the fastest method at finding the best solution, while BBO performs the second best for this case. Here, all the algorithms start optimization process at the beginning of almost same point, while HS/CS has a stably faster convergent speed than others. All methods clearly outperform the standard HS algorithm.

For this case, HS/CS is the most efficient and fastest method at finding the best global function values among ten methods.

From Fig. 4, apparently, HS/CS is well capable of finding the better solutions than all other methods. Here, PSO converges sharply at the first search stage, however, soon it gets trapped into the sub-minima and the global minimum decreases slightly. As the figures show, BBO performs the second best for this function. In addition, in this function both SGA and HS/CS have moved to the best solutions initially, while later HS/CS converges to the better minimum than SGA.

For this case, very similar to F04 (see Fig. 4), HS/CS significantly outperforms all other methods. By carefully looking at Fig. 5, in the beginning of the optimization process,

Table 1 Benchmark functions

No.	Name	Definition
F01	Ackley	$f(\vec{x}) = 20 + e - 20 \cdot e^{-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$
F02	Fletcher–Powell	$f(\vec{x}) = \sum_{i=1}^{n}(A_i - B_i)^2,\ A_i = \sum_{j=1}^{n}(a_{ij}\sin\alpha_j + b_{ij}\cos\alpha_j)$ $B_i = \sum_{j=1}^{n}(a_{ij}\sin x_j + b_{ij}\cos x_j)$
F03	Griewank	$f(\vec{x}) = \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$
F04	Penalty #1	$f(\vec{x}) = \frac{\pi}{30}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 \cdot \left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4),\ y_i = 1 + 0.25(x_i + 1)$
F05	Penalty #2	$f(\vec{x}) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2 \cdot \left[1 + \sin^2(3\pi x_{i+1})\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
F06	Quartic *with noise*	$f(\vec{x}) = \sum_{i=1}^{n}(i \cdot x_i^4 + U(0,1))$
F07	Rastrigin	$f(\vec{x}) = 10 \cdot n + \sum_{i=1}^{n}(x_i^2 - 10 \cdot \cos(2\pi x_i))$
F08	Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$
F09	Schwefel 2.26	$f(\vec{x}) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin(\lvert x_i\rvert^{1/2})$
F10	Schwefel 1.2	$f(\vec{x}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$
F11	Schwefel 2.22	$f(\vec{x}) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$
F12	Schwefel 2.21	$f(\vec{x}) = \max_i\{\lvert x_i\rvert, 1 \le i \le n\}$
F13	Sphere	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$
F14	Step	$f(\vec{x}) = 6 \cdot n + \sum_{i=1}^{n}\lfloor x_i \rfloor$

PSO converges faster than HS/CS, while HS/CS is well capable of improving its solution steadily for a long run.

Very clearly, for this case, HS/CS is the most efficient and fastest method at finding the best global function values among ten methods. BBO and SGA ranks two and three, respectively.

From Fig. 7, very apparently, though HS/CS performs equally with BBO between the generations 10 and 17. However, HS/CS converges in the much more stable state for this case. This demonstrates that, the combination of HS and CS can generate the good performance.

From Fig. 8, HS/CS performs far better than others for this unimodal case. PSO has a better function value than HS/CS; however, the function value obtained by HS/CS is better than PSO at the 11th generation.

From Fig. 9, very clearly, though HS/CS is outperformed by BBO between generation 6 and generation 26, it shows a much more stable convergent curve than others for this unimodal case. At last, HS/CS reaches the optimal solution significantly superiorly to other algorithms. BBO is only inferior to HS/CS, and can find the second best function value for this case.

Very clearly, HS/CS has the fastest convergence rate at finding the global minimum after 14 iterations. HS/CS is able to find the optimal solution significantly superiorly to others. BBO is only inferior to HS/CS, and can find the second best function values for this case.

Apparently, HS/CS is able to find the best solution with much more stable optimization process than others using the least time. Looking carefully at Fig. 11, for BBO and

Table 2 Mean optimization results in 14 benchmark functions

	ACO	BBO	CS	DE	ES	GA	HS	HSCS	PSO	SGA
F01	2.22	1.21	2.45	1.72	2.69	2.45	2.75	**1.00**	2.32	1.28
F02	14.23	1.43	9.55	5.35	13.37	5.25	12.87	**1.00**	10.65	1.66
F03	4.43	3.10	28.69	7.33	35.13	13.45	68.69	**1.00**	27.57	2.80
F04	1.1E7	5.3E3	7.8E5	4.1E4	5.9E6	6.6E4	9.6E6	**1.00**	9.7E5	1.70
F05	4.8E5	870.08	7.6E4	8.3E3	3.6E5	1.8E4	5.8E5	**1.00**	8.4E4	84.30
F06	218.45	20.13	588.25	86.62	2.9E3	234.36	3.0E3	**1.00**	679.08	8.00
F07	7.08	1.53	8.11	6.23	9.72	6.35	9.25	**1.00**	7.27	2.05
F08	24.16	1.57	7.06	3.67	32.66	6.33	21.80	**1.00**	7.52	1.52
F09	5.22	2.68	13.08	10.04	12.11	4.09	14.72	**1.00**	15.00	2.84
F10	2.15	1.25	1.31	2.84	3.33	2.48	3.06	**1.00**	2.21	1.84
F11	12.00	1.67	11.71	5.18	18.73	9.28	14.83	**1.00**	10.84	2.54
F12	1.66	1.73	1.87	2.04	2.48	2.09	2.53	**1.00**	2.09	1.51
F13	59.90	4.56	47.18	11.74	126.69	40.90	113.38	**1.00**	45.57	5.17
F14	7.81	3.74	33.09	8.78	58.35	18.14	85.79	**1.00**	34.44	2.89
Time	2.07	1.00	1.33	1.28	1.29	1.37	1.79	1.86	1.55	1.33

Table 3 Best optimization results in 14 benchmark functions

	ACO	BBO	CS	DE	ES	GA	HS	HSCS	PSO	SGA
F01	3.21	1.52	3.95	2.61	4.59	3.49	4.86	**1.00**	3.84	1.48
F02	34.38	2.35	16.08	13.38	25.15	7.47	27.70	**1.00**	23.22	2.69
F03	3.63	2.21	22.16	8.49	40.59	5.85	80.92	**1.00**	26.50	2.19
F04	**1.00**	3.1E32	2.4E36	3.4E35	5.638	3.7E32	1.1E39	6.1E31	4.2E36	9.9E31
F05	**1.00**	444.34	7.1E6	1.8E6	4.1E8	2.2E4	4.9E8	117.98	7.5E6	109.60
F06	825.75	34.65	1.8E3	274.10	2.4E4	350.89	3.0E4	**1.00**	4.5E3	4.73
F07	8.98	1.39	10.94	8.11	13.43	7.22	11.49	**1.00**	10.27	1.43
F08	91.75	3.09	14.08	14.66	120.72	13.37	56.02	**1.00**	18.21	2.97
F09	17.07	12.53	107.49	80.62	90.26	15.31	138.47	**1.00**	121.84	12.69
F10	1.89	1.72	**1.00**	4.85	4.53	2.31	4.81	1.33	3.02	1.46
F11	14.26	1.95	12.83	8.17	22.87	10.03	24.53	**1.00**	15.47	2.73
F12	1.24	1.91	2.03	2.85	3.45	1.87	3.43	1.12	2.17	**1.00**
F13	110.95	8.48	86.15	28.22	285.55	70.71	315.81	**1.00**	107.60	5.04
F14	10.67	3.21	70.83	23.87	118.67	20.19	267.54	**1.00**	104.87	2.33

SGA, they perform approximately equally and are inferior to HS/CS.

From above analyses about the Figs. 2, 3, 4, 5, 6, 7, 8, 9, 10 and 11, we draw the conclusion that our proposed HS/CS algorithm is able to find the best solutions among ten methods. In general, BBO and SGA are only inferior to the HS/CS. Further, benchmarks F04, F05, F06, F08, and F10 illustrate that PSO is well capable to find the best solutions initially, while later it may easily trap into the local values leading to fail to find the global function values. In addition, it should be noted that, in Simon (2008), BBO has been compared with seven EAs over 14 benchmarks and an engineering case. The results have shown the robustness of BBO method. This also indirectly demonstrated that our HS/CS method is a more powerful and efficient search technique than others.

4.2 Influence of control parameter

Like other metaheuristic methods, the parameter setting has an important factor on the performance of HS/CS. To compare the different effects on the parameter of HMCR and PAR, various simulations are conducted, and the results are recorded in Tables 4, 5, 6, and 7. Note that, all other parameter settings are set to the above experiments if there is no special notification. Tables 4, 6 and Tables 5, 7 have recorded the best and average performance of HS/CS algorithm, respectively.

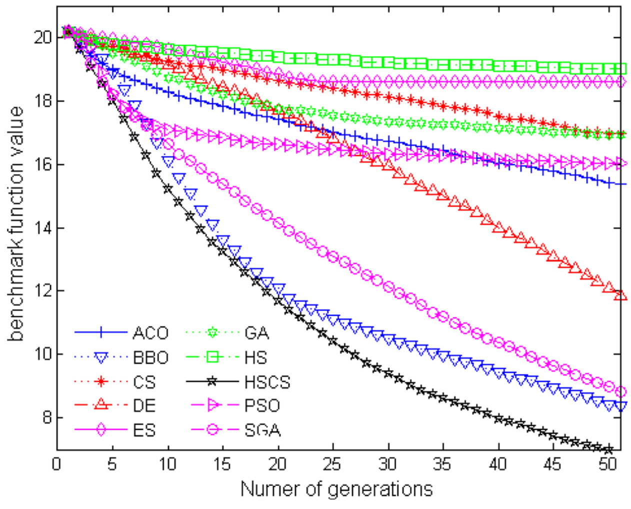

Fig. 2 Fitness curves of ten methods for the F01 function

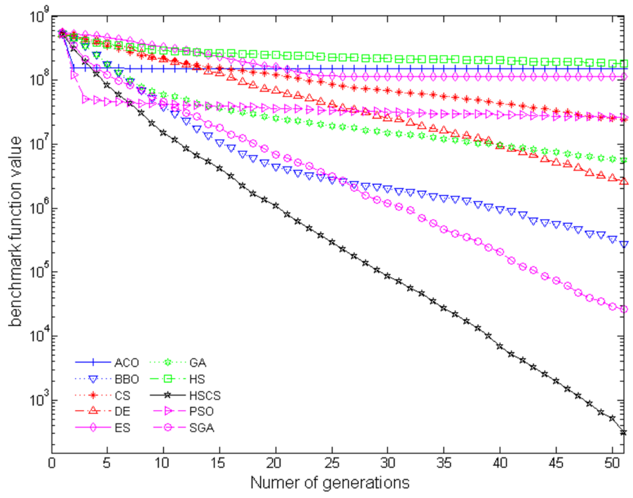

Fig. 5 Fitness curves of ten methods for the F05 function

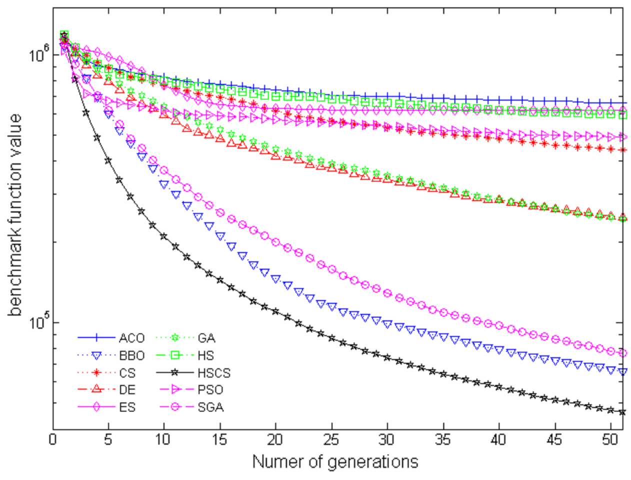

Fig. 3 Fitness curves of ten methods for the F02 function

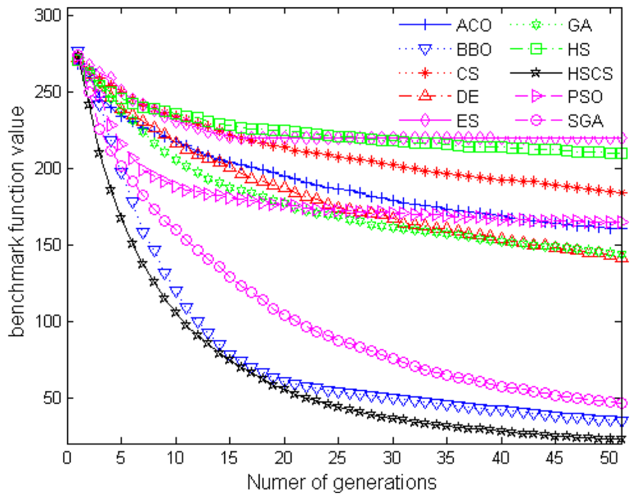

Fig. 6 Fitness curves of ten methods for the F07 function

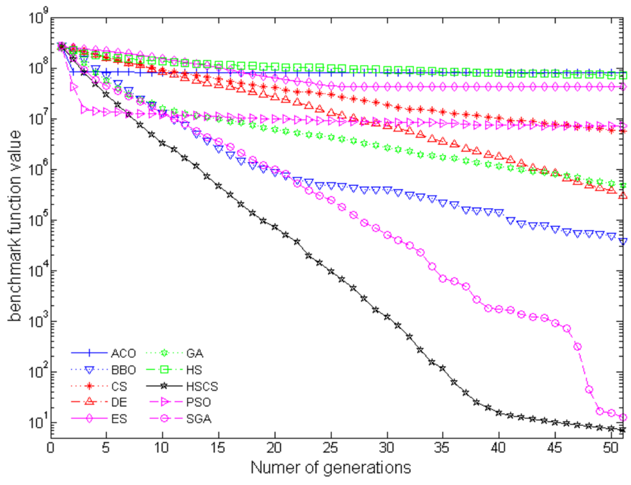

Fig. 4 Fitness curves of ten methods for the F04 function

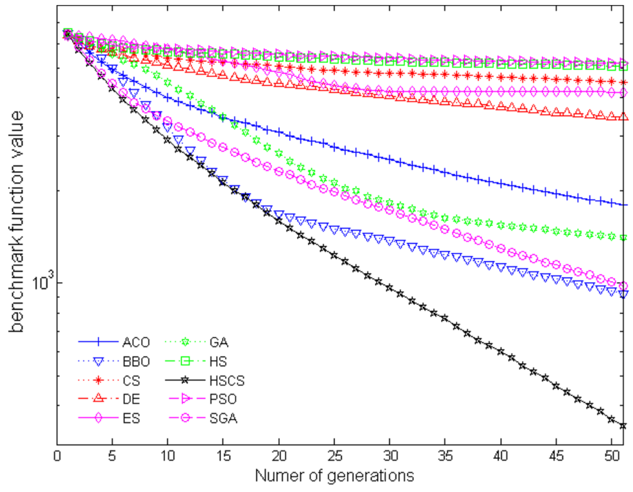

Fig. 7 Fitness curves of ten methods for the F09 function

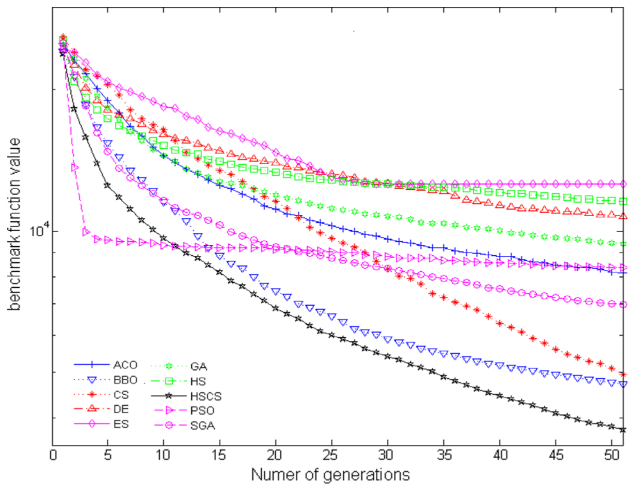

Fig. 8 Fitness curves of ten methods for the F10 function

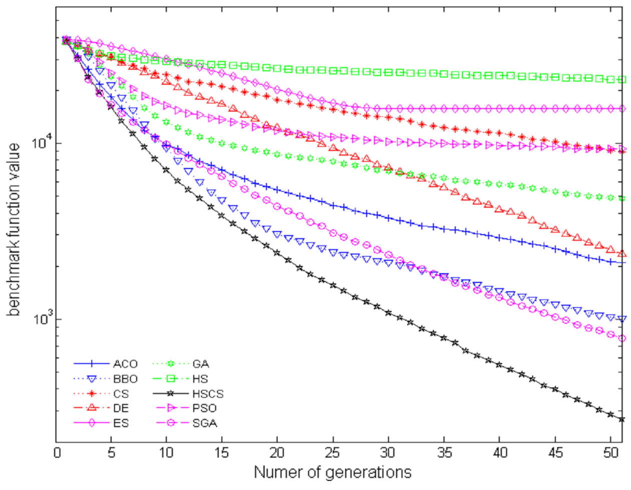

Fig. 11 Fitness curves of ten methods for the F14 function

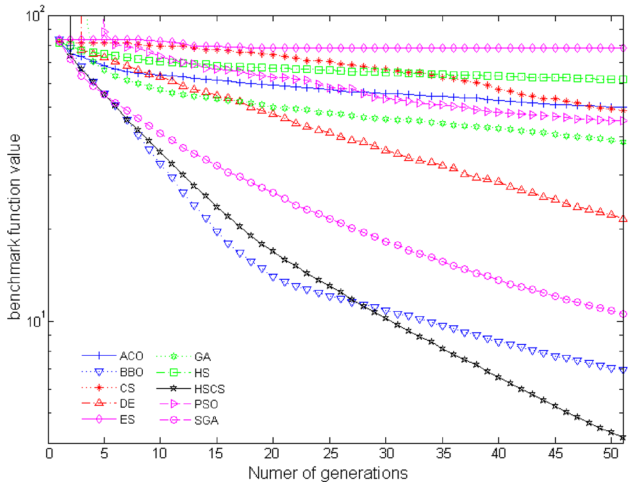

Fig. 9 Fitness curves of ten methods for the F11 function

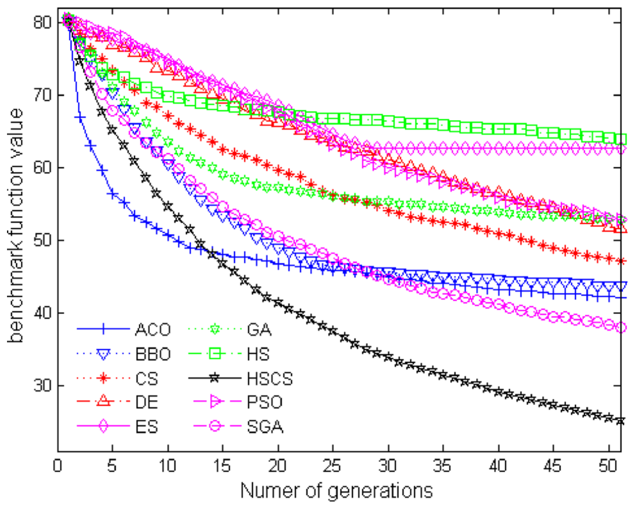

Fig. 10 Fitness curves of ten methods for the F12 function

4.2.1 HMCR

Firstly, the influence of HMCR is investigated through an array of simulations with HMCR = 0, 0.1, 0.2, . . ., 0.9, 1.0 and PAR = 0.1 (see Tables 4, 5). From Tables 4, 5, obviously, it can be seen that: (1) for the three benchmark functions F02, F05 and F14, HS/CS performs slightly differently, that is to say, these three benchmark functions are insensitive to the parameter HMCR. (2) For F12, HS/CS performs better on smaller HMCR (<0.5). (3) However, for other functions, HS/CS performs better on bigger HMCR (>0.5). In sum, HS/CS performs the best when HMCR is equal or very close to 0.9. So, we set HMCR = 0.9 in other experiments.

4.2.2 PAR

Secondly, the influence of PAR is investigated through an array of simulations with PAR = 0, 0.1, 0.2, $\cdots$, 0.9, 1.0 and HMCR = 0.9 (see Tables 6, 7). From Table 6, we can recognize that the function values for HS/CS vary little with the increase of PAR, and HS/CS reaches optimum/minimum in most benchmarks when PAR is equal or very close to 0.1. Whereas, looking at numbers in Table 7, the numbers are very complex and in disorder. Considering comprehensively, our aim is to get the best performance, so we set PAR = 0.1 in other experiments.

At last, we must point out that, the effects of the two main parameters of the HS namely HMCR and HMS were investigated by many researchers (Omran and Mahdavi 2008). As per the previous research, a large value for HMCR (i.e. approaching to 1.00) is generally used except for problems with a very low dimensionality for which a small value of HMCR is recommended. This conclusion coincides with the experimental results conducted in Sect. 4.2.

Table 4 Best optimization results with different HMCR

	HMCR										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F01	2.46	2.46	2.46	2.41	2.17	2.17	1.88	1.73	1.59	**1.00**	2.46
F02	1.50E4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F03	75.29	105.09	9.03	9.03	9.03	9.03	9.03	9.03	6.94	**1.00**	9.03
F04	4.73E6	3.04	4.56E4	2.98	2.98	2.98	2.98	2.98	2.95	**1.00**	2.98
F05	3.05E6	9.44	1.13	1.70E4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F06	2.56E6	4.31E3	5.70E3	2.40E3	3.05E3	1.02E3	534.30	118.16	17.00	**1.00**	3.08E3
F07	11.69	13.56	10.88	1.30	1.30	8.86	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F08	83.72	1.41	16.51	1.41	1.41	1.41	30.90	**1.00**	**1.00**	**1.00**	**1.00**
F09	361.80	391.00	1.54	12.86	1.54	1.54	1.54	189.91	**1.00**	**1.00**	**1.00**
F10	926.85	20.54	1.31E3	28.67	2.46	2.46	2.46	2.46	813.89	**1.00**	**1.00**
F11	10.10	1.29	3.76	3.76	8.17	3.76	3.76	3.76	1.68	**1.00**	3.76
F12	5.11	**1.00**	5.31	1.72	4.82	1.72	1.72	1.72	1.72	1.72	5.71
F13	168.46	221.09	165.85	146.03	66.16	84.06	39.90	26.77	14.45	**1.00**	66.16
F14	905.13	10.38	8.33	**1.00**	**1.00**	11.63	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	0	2	1	2	3	2	4	5	6	**13**	7

Table 5 Mean optimization results with different HMCR

	HMCR										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	**1.00**	**1.00**
F02	1.52E3	14.75	13.04	15.24	6.98	8.19	6.48	5.27	**1.00**	1.14	6.72
F03	11.44	3.19E4	1.10	1.05	1.06	1.05	1.01	1.01	**1.00**	**1.00**	1.07
F04	4.1E6	3.9E4	4.5E4	1.3E4	3.1E3	2.9E3	286.03	143.04	1.01	**1.00**	4.7E3
F05	1.0E7	8.7E4	8.0E4	5.5E4	6.5E4	2.1E4	1.2E4	3.1E3	4.37	**1.00**	3.3E4
F06	**1.00**	3.77E4	326.92	369.77	3.8E4	1.19	1.19	1.19	1.19	1.19	1.19
F07	10.80	4.04E6	11.34	271.72	307.32	3.16E4	1.05	1.02	**1.00**	**1.00**	1.04
F08	90.13	3.79E4	3.0E4	307.45	263.18	297.48	3.0E4	1.06	1.04	**1.00**	1.47
F09	217.94	8.4E6	2.75	11.11	261.52	223.64	252.80	2.59E4	1.22	**1.00**	2.32
F10	280.93	4.25E4	1.96E6	1.52E4	73.27	155.81	134.02	149.84	1.5E4	**1.00**	3.22
F11	3.77	**1.00**	4.58E4	1.29	13.17	169.97	371.14	317.58	359.20	3.7E4	1.18
F12	3.19	2.9E4	4.1E4	**1.00**	2.9E4	160.05	136.29	297.59	254.64	288.02	2.9E4
F13	3.11	9.71	9.3E6	3.6E6	**1.00**	10.19	154.67	131.71	287.60	246.10	278.38
F14	151.36	4.7E5	1.0E4	4.6E3	**1.00**	3.7E3	16.62	20.65	17.45	38.03	33.84
	1	1	0	1	2	0	0	0	3	**8**	1

5 Conclusions

We have improved the CS by combining cuckoo search with harmony search algorithm, and we also have evaluated the HS/CS on various benchmarks. A novel variant of CS algorithm has been presented, and an improvement is applied to the mutation between cuckoos using harmony search algorithm during the process of cuckoos updating. Using the original configuration of the cuckoo search algorithm, we have generated the new harmonies as per the newly generated cuckoo at each iteration after cuckoo's position updating. If a new harmony vector has the better finesse than before and it will be used to replace a newly generated cuckoo. By combination of HS and CS, the HS/CS is well capable to exploit their good feature and this can avoid all individuals getting trapped in inferior local optimal regions. As per various experiments in the present work, we have observed that the proposed HS/CS is able to exploit the useful information in population with the aim of generating better quality solutions when compared with the other search tech-

Table 6 Best optimization results in 14 benchmark functions with different PAR

	PAR										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F02	4.3E3	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F03	**1.00**	3.15	2.00	2.00	2.00	2.00	2.00	2.00	1.39	1.90	1.09
F04	**1.00**	3.84	243.63	3.84	3.84	3.84	3.84	3.84	3.84	3.84	3.84
F05	4.46	**1.00**	2.00	8.7E3	2.00	2.00	2.00	2.00	2.00	2.00	2.00
F06	**1.00**	11.94	23.57	37.16	6.29	25.36	18.77	35.89	4.23	8.10	14.05
F07	13.90	**1.00**	1.91	3.84	3.84	32.85	3.84	3.84	3.84	3.84	3.84
F08	6.24	**1.00**	1.57	**1.00**	**1.00**	**1.00**	13.92	**1.00**	**1.00**	**1.00**	**1.00**
F09	123.88	4.46	2.00	**1.00**	2.00	2.00	2.00	372.21	2.00	2.00	2.00
F10	1.1E3	1.91	**1.00**	6.05	3.84	3.84	3.84	3.84	5.3E3	3.84	3.84
F11	3.1E3	**1.00**	4.5E3	4.5E3	2.2E3	4.5E3	4.5E3	4.5E3	4.5E3	2.6E3	4.5E3
F12	1.4E3	**1.00**	3.5E3	378.40	595.74	378.40	378.40	378.40	378.40	378.40	2.7E3
F13	1.11	7.86	3.88	3.46	5.56	5.42	4.83	3.49	2.10	1.31	**1.00**
F14	131.63	**1.00**	1.91	3.84	3.84	6.05	3.84	3.84	3.84	3.84	3.84
	4	**8**	3	4	3	3	2	3	3	3	4

Table 7 Mean optimization results with different PAR

	PAR										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F01	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F02	436.42	2.79	4.71	5.18	4.40	9.82	6.22	2.98	1.12	2.04	**1.00**
F03	**1.00**	1.3E4	1.42	1.42	1.42	1.43	1.42	1.44	1.42	1.42	1.41
F04	2.30	58.17	9.0E3	1.02	**1.00**	1.01	1.03	1.02	**1.00**	**1.00**	**1.00**
F05	246.14	1.28	123.44	8.3E3	1.47	1.02	9.86	65.50	1.24	1.97	**1.00**
F06	**1.00**	5.1E6	5.6E4	3.3E4	5.2E6	572.00	571.99	572.01	571.98	571.98	571.99
F07	5.17	3.17	**1.00**	134.94	79.99	1.2E4	1.44	1.42	1.40	1.40	1.42
F08	11.63	56.16	8.75E3	105.22	95.67	56.70	8.8E3	1.09	1.04	**1.00**	1.13
F09	62.60	169.23	1.53	1.36	69.19	62.90	37.29	5.7E3	1.30	1.35	**1.00**
F10	122.02	1.05	2.34	1.8E3	19.25	23.58	21.86	14.00	1.8E3	**1.00**	1.58
F11	98.30	**1.00**	7.1E3	122.82	86.05	1.1E4	1.3E4	1.2E4	7.1E3	1.1E6	127.69
F12	3.76	8.4E3	8.5E3	**1.00**	8.5E3	194.28	87.13	102.55	93.26	55.26	8.6E3
F13	**1.00**	22.99	1.6E3	14.09	6.09	4.27	1.2E3	564.97	664.98	604.71	358.27
F14	35.09	1.88	1.63	34.39	**1.00**	5.2E3	76.18	120.01	54.08	63.60	57.59
	4	2	2	2	3	1	1	1	2	4	**5**

niques, such as ACO, BBO, CS, DE, ES, GA, HS, PSO and SGA.

In optimization field, there are many issues that are worthy of further study. Our future work will focus on the following issues. Firstly, aiming to the disadvantages of the benchmark evaluation, HS/CS will be further tested by comparing with more published results and applied to solve practical engineering optimization problems, and we believe it can be an effective method for addressing real-world problems. Secondly, more methods will be used to investigate the performance of HS/CS in various respects, such as fitness evaluations, statistical analyses (*T* test or *F* test). Thirdly, to make HS/CS method implement in the faster way, the research of speeding up its convergence is also an interesting problem, and this can be performed using nonlinear activation function to speed up convergence of dynamic neural networks (Li and Li 2014; Li et al. 2012, 2013b, c) and accelerated algorithm for data association (Li et al. 2014). Fourthly, for the purpose of making HS/CS method implement in the stable way, some theoretical analyses will be implemented using nonlinear dynamic system and Markov chain theory. Lastly, some other optimization strategies, such as quantum theory, orthogonal learning, incremental learning and opposition-based learning, can be combined with HS or CS method to develop new meta-hybrid approach to enhance the search ability of the basic method.

Acknowledgments This work was supported by Research Fund for the Doctoral Program of Jiangsu Normal University (no. 13XLR041) and National Natural Science Foundation of China (no. 61272297 and no. 61402207).

References

Beyer H (2001) The theory of evolution strategies. Springer, New York

Doğan E, Saka MP (2012) Optimum design of unbraced steel frames to LRFD-AISC using particle swarm optimization. Adv Eng Softw 46(1):27–34. doi:10.1016/j.advengsoft.2011.05.008

Dorigo M, Stutzle T (2004) Ant colony optimization. MIT Press, Cambridge

Gandomi AH (2014) Interior search algorithm (ISA): a novel approach for global optimization. ISA Trans 53(4):1168–1183. doi:10.1016/j.isatra.2014.03.018

Gandomi AH, Yang XS, Talatahari S, Alavi AH (2013a) Metaheuristic applications in structures and infrastructures. Elsevier, Waltham

Gandomi AH, Yun GJ, Yang X-S, Talatahari S (2013b) Chaos-enhanced accelerated particle swarm optimization. Commun Nonlinear Sci Numer Simulat 18(2):327–340. doi:10.1016/j.cnsns.2012.07.017

Gandomi AH, Yang X-S, Alavi AH, Talatahari S (2013c) Bat algorithm for constrained optimization tasks. Neural Comput Appl 22(6):1239–1255. doi:10.1007/s00521-012-1028-9

Gandomi AH, Yang X-S, Alavi AH (2013d) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35. doi:10.1007/s00366-011-0241-y

Gandomi AH, Talatahari S, Yang X-S, Deb S (2013e) Design optimization of truss structures using cuckoo search algorithm. Struct Des Tall Spec Build 22(17):1330–1349. doi:10.1002/tal.1033

Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simulat 17(12):4831–4845. doi:10.1016/j.cnsns.2012.05.010

Gandomi AH, Yang X-S, Talatahari S, Deb S (2012) Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization. Comput Math Appl 63(1):191–200. doi:10.1016/j.camwa.2011.11.010

García-Martínez C, Lozano M (2010) Evaluating a local genetic algorithm as context-independent local search operator for metaheuristics. Soft Comput 14(10):1117–1139

Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68. doi:10.1177/003754970107600201

Goldberg DE (1998) Genetic algorithms in search optimization and machine learning. Addison-Wesley, New York

Guo L, Wang G-G, Gandomi AH, Alavi AH, Duan H (2014) A new improved krill herd algorithm for global numerical optimization. Neurocomputing. doi:10.1016/j.neucom.2014.01.023

Kennedy J, Eberhart R (1995) Particle swarm optimization. Paper presented at the Proceeding of the IEEE international conference on neural networks, Perth, Australia, 27 Nov–1 Dec

Khatib W, Fleming P (1998) The stud GA: a mini revolution? In: Eiben A, Back T, Schoenauer M, Schwefel H (eds) Proceedings of the 5th international conference on parallel problem solving from nature, New York, USA. Parallel problem solving from nature. Springer, London, pp 683–691

Li X, Wang J, Zhou J, Yin M (2011) A perturb biogeography based optimization with mutation for global numerical optimization. Appl Math Comput 218(2):598–609. doi:10.1016/j.amc.2011.05.110

Li S, Chen S, Liu B (2012) Accelerating a recurrent neural network to finite-time convergence for solving time-varying Sylvester equation by Using a Sign-Bi-power Activation Function. Neural Process Lett 37(2):189–205. doi:10.1007/s11063-012-9241-1

Li X, Yin M (2014) Parameter estimation for chaotic systems by hybrid differential evolution algorithm and artificial bee colony algorithm. Nonlinear Dyn 77(1–2):61–71. doi:10.1007/s11071-014-1273-9

Li Y, Li S, Song Q, Liu H, Meng MQH (2014) Fast and robust data association using posterior based approximate joint compatibility test. IEEE Trans Ind Inform 10(1):331–339. doi:10.1109/TII.2013.2271506

Li S, Li Y (2014) Nonlinearly activated neural network for solving time-varying complex Sylvester equation. IEEE Trans Cybern 44(8):1397–1407. doi:10.1109/TCYB.2013.2285166

Li X, Wang J, Yin M (2013a) Enhancing the performance of cuckoo search algorithm using orthogonal learning method. Neural Comput Appl 24(6):1233–1247. doi:10.1007/s00521-013-1354-6

Li S, Liu B, Li Y (2013b) Selective positive–negative feedback produces the winner-take-all competition in recurrent neural networks. IEEE Trans Neural Netw Learn Syst 24(2):301–309. doi:10.1109/TNNLS.2012.2230451

Li S, Li Y, Wang Z (2013c) A class of finite-time dual neural networks for solving quadratic programming problems and its k-winners-take-all application. Neural Netw 39:27–39. doi:10.1016/j.neunet.2012.12.009

Li X, Zhang J, Yin M (2013d) Animal migration optimization: an optimization algorithm inspired by animal migration behavior. Neural Comput Appl 24(7–8):1867–1877. doi:10.1007/s00521-013-1433-8

Li X, Yin M (2012a) Application of differential evolution algorithm on self-potential data. PLoS ONE 7(12):e51199. doi:10.1371/journal.pone.0051199

Li X, Yin M (2012b) Self-adaptive constrained artificial bee colony for constrained numerical optimization. Neural Comput Appl 24(3–4):723–734. doi:10.1007/s00521-012-1285-7

Li X, Yin M (2013a) An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. Adv Eng Softw 55:10–31. doi:10.1016/j.advengsoft.2012.09.003

Li X, Yin M (2013b) Multiobjective binary biogeography based optimization for feature selection using gene expression data. IEEE Trans Nanobiosci 12(4):343–353. doi:10.1109/TNB.2013.2294716

Luna F, Estébanez C, León C, Chaves-González JM, Nebro AJ, Aler R, Segura C, Vega-Rodríguez MA, Alba E, Valls JM (2011) Optimization algorithms for large-scale real-world instances of the frequency assignment problem. Soft Comput 15(5):975–990. doi:10.1007/s00500-010-0653-4

Mirjalili S, Mohd Hashim SZ, Moradian Sardroudi H (2012) Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl Math Comput 218(22):11125–11137. doi:10.1016/j.amc.2012.04.069

Mirjalili S, Lewis A (2013) S-Shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evolut Comput 9:1–14. doi:10.1016/j.swevo.2012.09.002

Mirjalili S, Mirjalili SM, Yang X-S (2013) Binary bat algorithm. Neural Comput Appl. doi:10.1007/s00521-013-1525-5

Mirjalili S, Mirjalili SM, Lewis A (2014a) Grey wolf optimizer. Adv Eng Softw 69:46–61. doi:10.1016/j.advengsoft.2013.12.007

Mirjalili S, Mirjalili SM, Lewis A (2014b) Let a biogeography-based optimizer train your multi-layer perceptron. Inf Sci 269:188–209. doi:10.1016/j.ins.2014.01.038

Omran MGH, Mahdavi M (2008) Global-best harmony search. Appl Math Comput 198(2):643–656. doi:10.1016/j.amc.2007.09.004

Rahimi-Vahed A, Mirzaei A (2008) Solving a bi-criteria permutation flow-shop problem using shuffled frog-leaping algorithm. Soft Comput 12(5):435–452. doi:10.1007/s00500-007-0210-y

Simon D (2008) Biogeography-based optimization. IEEE Trans Evolut Comput 12(6):702–713. doi:10.1109/TEVC.2008.919004

Storn R, Price K (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. International Computer Science Institute, Berkley

Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359. doi:10.1023/A:1008202821328

Wang G-G, Gandomi AH, Alavi AH, Hao G-S (2013a) Hybrid krill herd algorithm with differential evolution for global numerical optimization. Neural Comput Appl 25(2):297–308. doi:10.1007/s00521-013-1485-9

Wang G-G, Gandomi AH, Alavi AH (2013b) An effective krill herd algorithm with migration operator in biogeography-based optimization. Appl Math Model 38(9–10):2454–2462. doi:10.1016/j.apm.2013.10.052

Wang G, Guo L, Duan H, Wang H, Liu L, Shao M (2013c) Hybridizing harmony search with biogeography based optimization for global numerical optimization. J Comput Theor Nanosci 10(10):2318–2328. doi:10.1166/jctn.2013.3207

Wang G-G, Guo L, Gandomi AH, Hao G-S, Wang H (2014a) Chaotic krill herd algorithm. Inf Sci 274:17–34. doi:10.1016/j.ins.2014.02.123

Wang G, Guo L, Wang H, Duan H, Liu L, Li J (2014b) Incorporating mutation scheme into krill herd algorithm for global numerical optimization. Neural Comput Appl 24(3–4):853–871. doi:10.1007/s00521-012-1304-8

Wang G-G, Gandomi AH, Alavi AH (2014c) Stud krill herd algorithm. Neurocomputing 128:363–370. doi:10.1016/j.neucom.2013.08.031

Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: Proceedings of world congress on nature and biologically inspired computing (NaBIC 2009), Coimbatore, India, December 2009. IEEE Publications, USA, pp 210–214

Yang XS (2010a) A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010), studies in computational intelligence, vol 284. Springer, pp 65–74. doi:10.1007/978-3-642-12538-6_6

Yang XS (2010b) Nature-inspired metaheuristic algorithms, 2nd edn. Luniver Press, Frome

Yang XS (2011) Optimization algorithms. In: Koziel S, Yang X-S (eds) Computational optimization, methods and algorithms. Studies in computational intelligence, vol 356. Springer, Berlin, Heidelberg, pp 13–31. doi:10.1007/978-3-642-20859-1_2

Yang XS, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1(4):330–343

Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evolut Comput 3(2):82–102