Soft Comput (2016) 20:3349–3362
DOI 10.1007/s00500-015-1726-1

FOCUS

Chaotic cuckoo search

Gai-Ge Wang[1,2,3] · **Suash Deb**[4] · **Amir H. Gandomi**[5] · **Zhaojun Zhang**[6] · **Amir H. Alavi**[7]

Published online: 27 May 2015

Abstract This study proposes a novel chaotic cuckoo search (CCS) optimization method by incorporating chaotic theory into cuckoo search (CS) algorithm. In CCS, chaos characteristics are combined with the CS with the intention of further enhancing its performance. Further, the elitism scheme is incorporated into CCS to preserve the best cuckoos. In CCS method, 12 chaotic maps are applied to tune the step size of the cuckoos used in the original CS method. Twenty-seven benchmark functions and an engineering case are utilized to investigate the efficiency of CCS. The results clearly demonstrate that the performance of CCS together with a suitable chaotic map is comparable as well as superior to that of the CS and other metaheuristic algorithms.

Keywords Global optimization · Cuckoo search · Chaotic maps · Multimodal function

Communicated by S. Deb, T. Hanne and S. Fong.

S. Deb is pioneer of cuckoo search algorithm.

✉ Gai-Ge Wang
gaigewang@163.com; gaigewang@gmail.com

Suash Deb
suashdeb@gmail.com

Amir H. Gandomi
a.h.gandomi@gmail.com

Zhaojun Zhang
zzj921@163.com

Amir H. Alavi
ah_alavi@hotmail.com; alavi@msu.edu

1 School of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, Jiangsu, China

2 Institute of Algorithm and Big Data Analysis, Northeast Normal University, Changchun 130117, China

3 School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China

4 Cambridge Institute of Technology, Cambridge Village, Tatisilwai, Ranchi 835103, Jharkhand, India

5 BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI 48824, USA

6 School of Electrical Engineering and Automation, Jiangsu Normal University, Xuzhou 221116, Jiangsu, China

7 Department of Civil and Environmental Engineering, Michigan State University, East Lansing, MI 48824, USA

1 Introduction

By observing the great laws of nature, several modern metaheuristic algorithms (Gandomi et al. 2013a; Yang et al. 2013) are generally applied to address myriads of complex optimization problems (Yang 2010b). Some of these optimization problems include feature selection (Li and Yin 2013a), image segmentation (Zhang et al. 2011), flow shop scheduling (Li and Yin 2013b), reliability problem (Zou et al. 2010, 2011) and knapsack problem (Zou et al. 2011). These kinds of metaheuristic algorithms are well capable of finding the best solutions by extracting the useful information from a group of individuals. Since the genetic algorithms (GAs) (Goldberg 1998) were put forward during the 1950s and 1960s, several metaheuristic algorithms have been proposed such as artificial plant optimization algorithm (APOA) (Cai et al. 2012), ant colony optimization (ACO) (Zhang and Feng 2012; Zhang et al. 2014; Dorigo et al. 1996), differential evolution (DE) (Storn and Price 1997; Li and Yin 2012), bat algorithm (BA) (Gandomi et al. 2013b; Yang and Gandomi 2012; Yang 2010a; Mirjalili et al. 2013), artificial physics optimization (Xie et al. 2012), biogeography-based optimization (BBO) (Simon 2008; Wang et al. 2013a; Mirjalili et al.

2014a), krill herd (KH) (Gandomi and Alavi 2012; Wang et al. 2014a), harmony search (HS) (Geem et al. 2001; Wang et al. 2014b), monarch butterfly optimization (MBO) (Wang et al. 2015), flower pollination algorithm (FPA) (Yang et al. 2014), animal migration optimization (AMO) (Li et al. 2014), particle swarm optimization (PSO) (Kennedy and Eberhart 1995; Mirjalili and Lewis 2013; Talatahari et al. 2013; Zhao et al. 2014a, b), grey wolf optimizer (GWO) (Mirjalili et al. 2014b) and firefly algorithm (FA) (Gandomi et al. 2011; Yang et al. 2012; Wang et al. 2014c).

Recently, Yang and Deb (2010) proposed a new metaheuristic optimization algorithm, called CS method. CS is inspired by smart incubation behavior of a type of birds called cuckoos in nature. For the single-object case, in CS, the number of eggs, cuckoos and nests are equal, and each one represents an available solution. CS is well capable of finding the best solutions by continuously using new and potentially better solution to replace a not-so-good cuckoo in the population. Conceptually, extremely simple, CS is very easy to implement.

In most cases, CS performs local search well, but sometimes it may have no ability of escaping from local optima which restricts its ability to carry out full search globally. In order to overcome this and enhance the searching ability of the CS method, a strategy has been provided (Li and Yin 2015), which uses self-adaptive method towards adjusting parameters.

On the other hand, several scholars have paid more attention in nonlinear dynamics, especially of chaos. And recently, chaotic theory has found applications in the area of metaheuristics. Up to now, chaotic sequences have been combined with several metaheuristic algorithms, such as imperialist competitive algorithm (Talatahari et al. 2012), FA (Gandomi et al. 2013c), charged system search (Nouhi et al. 2013), chaotic swarming of particles (CSP) (Kaveh et al. 2014), BA (Gandomi and Yang 2014), KH algorithm (Wang et al. 2014d), memetic DE algorithm (Jia et al. 2011) and PSO (Gandomi et al. 2013d).

The present manuscript introduces a chaotic CS-based method, intended for accelerating convergence. In serials of chaotic CS-based algorithms, various one-dimensional chaotic maps are utilized in place of the step size of CS. Therefore, various approaches that use chaotic maps as efficient alternatives to pseudorandom sequences have been put forward. The chaotic CS-based methods are experimented on 27 benchmark functions and an engineering case. Performance comparison with the other approaches demonstrated the superiority of the proposed strategy. Series of the simulation show that CCS performs more efficiently and accurately than the basic CS and other metaheuristic methods. The enhancement of the new methods are revealed due to the usage of determinate chaotic signals substitute for original step size.

The paper is organized into different sections: Sect. 2 describes the basic CS algorithm and 12 chaotic maps. The proposed CCS approach is described in Sect. 3. Subsequently, the tuning of the step size α and finding the best chaotic CS are discussed in Sect. 4. In addition, comparing with eight other methods, the CCS algorithm is also evaluated through 27 functions and an engineering case. Finally, Sect. 5 provides a summary of our work.

2 Preliminary

First and foremost, we will provide a brief preliminary on the CS algorithm and 12 chaotic maps.

2.1 The CS algorithm

By idealizing and simplifying the brood behavior of some cuckoo species, CS is put forward in combination with the Levy flight, which is a swarm intelligence optimization method.

In order to describe the CS method more easily, Yang and Deb have proposed the following three hypothetical rules:

1. each cuckoo lays and places one egg in a randomly chosen nest at a time;
2. the optimal nests cannot be corrupted;
3. the number of host nests is fixed and the egg can be found by the host bird with a fixed probability $p_{\mathrm{a}} \in [0, 1]$.

Based on three rules, the pseudo-code of the CS can be represented as shown in Fig. 1.

For cuckoo i, when a new cuckoo $x^{(t+1)}$ is generated, a Levy flight is implemented

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus \hat{\mathrm{Le}}\mathrm{vy}(\lambda) \tag{1}$$

where $\alpha > 0$ is the step size. In most cases, α is simply set to 1. The product $\oplus$ means entrywise multiplications.

2.2 Chaotic maps

In optimization field, chaotic optimization algorithm (COA) is a kind of random-based methods that use chaotic variables. In COA, because the chaos has the property of the non-repetition and ergodicity, full search can be implemented at higher speeds than stochastic searches that rely on probabilities. In order to serve for this objective, 12 one-dimensional non-invertible maps are applied to generate chaotic sets (see Table 1). Note that M1, M2, ..., M12 in Tables 3 and 4 are short for the 12 responding chaotic maps. More details about COA method and 12 chaotic maps can be found in Wang et al. (2013b).

Fig. 1 Pseudo-code of the CS algorithm

```
Cuckoo search via Lévy flights
Begin
    Step 1: Initialization. Set the generation counter $G = 1$; initialize the population $P$ of $n$ host
            nests randomly; set the discovery rate $p_a$.
    Step 2: While $G$ < MaxGeneration do
            Sort the population as per their fitness.
            Get a cuckoo randomly (say, i) and replace its solution by performing Lévy flights.
            Evaluate its fitness $F_i$.
            Choose a nest among n (say, j) randomly.
            if ($F_i < F_j$)
                Replace j by the new solution.
            end if
            A fraction ($p_a$) of the worse nests is abandoned and new ones are built.
            Keep the best solutions/nests.
            Sort the population and find the current best.
            Pass the current best to the next generation.
            $G = G+1$.
    Step 3: end while
End.
```

Table 1 Twelve different chaotic maps

No.	Name	Definition
M1	Chebyshev map	$x_{k+1} = \cos(k \cos^{-1}(x_k))$
M2	Circle map[a]	$x_{k+1} = x_k + b - (a/2\pi)\sin(2\pi k) \bmod (1)$
M3	Gaussian map	$x_{k+1} = \begin{cases} 0 & x_k = 0 \\ 1/x_k \bmod (1) & \text{otherwise} \end{cases}, \quad 1/x_k \bmod (1) = \frac{1}{x_k} - \left[\frac{1}{x_k}\right]$
M4	Intermittency map	$x_{k+1} = \begin{cases} \varepsilon + x_k + c x_k^n & 0 < x_k \le P \\ \frac{x_k - P}{1-P} & P < x_k < 1 \end{cases}$
M5	Iterative map	$x_{k+1} = \sin\left(\frac{a\pi}{x_k}\right), \quad a \in (0, 1)$
M6	Liebovitch map	$x_{k+1} = \begin{cases} \alpha x_k & 0 < x_k \le P \\ \frac{P - x_k}{P_2 - P_1} & P_1 < x_k \le P_2 \\ 1 - \beta(1 - x_k) & P_2 < x_k \le 1 \end{cases},$
M7	Logistic map	$x_{k+1} = a x_k (1 - x_k)$
M8	Piecewise map	$x_{k+1} = \begin{cases} \frac{x_k}{P} & 0 \le x_k < P \\ \frac{x_k - P}{0.5 - P} & P \le x_k < \frac{1}{2} \\ \frac{1 - P - x_k}{0.5 - P} & \frac{1}{2} \le x_k < 1 - P \\ \frac{1 - x_k}{P} & 1 - P \le x_k < 1 \end{cases}$
M9	Sine map	$x_{k+1} = \frac{a}{4}\sin(\pi x_k), \quad 0 < a \le 4$
M10	Singer map	$x_{k+1} = \mu(7.86 x_k - 23.31 x_k^2 + 28.75 x_k^3 - 13.302875 x_k^4)$
M11	Sinusoidal map	$x_{k+1} = a x_k^2 \sin(\pi x_k)$
M12	Tent map	$x_{k+1} = \begin{cases} \frac{x_k}{0.7} & x_k < 0.7 \\ \frac{10}{3} & x_k \ge 0.7 \end{cases}$

[a] With $a = 0.5$ and $b = 0.2$, it generates chaotic sequence in (0, 1)

3 Chaotic CS

As presented in Eq. (1), the main parameters of CS are the step size α and discovery rate p_{a} that characterizes the variations of the global best step size, and their values have a great influence on the convergent speed and how CS performs.

The basic CS is well capable of finding the best solutions, but the solutions are still swinging slightly around the optima. As per the previous literature (Yang 2010b), we used $\alpha = O(L/10)$ and $p_{\mathrm{a}} = 0.25$ for the standard CS, where L is the characteristic scale of the problem of interest.

The step size used in CS remains unchanged. The improved CS method with a chaotic varying step size α may be better than the basic one, which may also accelerate its convergence. By normalizing all chaotic maps, their variations are always in [0, 2]. After normalization, chaotic maps

Fig. 2 Pseudo-code of the CCS algorithm

```
CCS algorithm
Begin
    Step 1: Initialization. Set the generation counter $t = 1$; initialize the population P randomly;
            set $p_a$, initial value of the chaotic map $c_0$ randomly, and elitism parameter KEEP.
    Step 2: While $t <$ MaxGeneration do
            Sort the population as per their fitness.
            Store the KEEP best cuckoos.
            Update the step size using chaotic maps ($\alpha = c_{t+1}$).
            Get a cuckoo randomly (say, i) and replace its solution by performing Lévy flights.
            Evaluate its fitness $F_i$.
            Choose a nest among n (say, j) randomly.
            if ($F_i < F_j$)
                Replace j by the new solution.
            end if
            A fraction ($p_a$) of the worse nests is abandoned and new ones are built.
            Replace the KEEP worst cuckoo with the KEEP best cuckoo.
            Sort the population and find the current best.
            $t = t+1$.
    Step 3: end while
End.
```

are able to tune step size α, and this improved CS method is referred as the chaotic CS.

The simple description of pseudo-code of CCS algorithm can be provided as shown in Fig. 2.

In addition, to protect the best cuckoos, elitism strategy is introduced into CCS. This forbids the best cuckoos from being corrupted by cuckoo updating operator. Note that an elitism strategy is used to save the property of the best cuckoos in the CCS process, so even if cuckoo updating operation destroys its corresponding cuckoo, the best cuckoos can be reverted back if needed.

4 Simulation experiments

In order to illustrate the benefits of the CCS method, it is tested by means of an array of experiments implemented on benchmark functions and an engineering case. In order to obtain the real and fair results, we did all the implementations under the same conditions as Wang et al. (2014e) and Guo et al. (2014). The benchmark functions as shown in Table 2 are standard testing functions, and their detailed properties can be found in Wang et al. (2014f, g). In the following experiments, the best value for each function is shown in bold. The dimension of the function is 20 in the present work.

4.1 CCS with different chaotic maps

Different chaotic CS variants were benchmarked using 14 high-dimensional distinguished numerical examples (see Table 2). In this subsection, tuning the step size α is carried out. Here, the value of α is replaced with 12 different chaotic maps (see Sect. 2.2).

For CCS algorithm, we set population size, elitism parameter and maximum generation to 50, 2 and 50, respectively. 1000 independent runs are implemented to get typical performances. The results are illustrated in Tables 3 and 4.

From Tables 3 and 4, it can be seen that the CCS performs more effectively with the M9 (Sine map) and M11 (Sinusoidal map) than others. For these two maps, on average, the M11 significantly outperforms the M9 on most benchmarks. Further, from Table 4, we can see that the M11 yields an output, having only slight difference with the optimal value when multiple runs are made. Comprehensive consideration suggests that we select M11 as the final optimal map to be used for chaotic CS (CCS). In addition, from the numerical simulations, the obtained results indicate that choosing an appropriate chaotic map is crucial in leading to making full use of the advantage of the chaotic CS. M11 can provide more information for CS method to guide its search while other chaotic maps cannot.

4.2 General performance of CCS

In order to investigate the benefits of CCS, it was compared with nine other metaheuristic algorithms, which are ACO (Dorigo and Stutzle 2004), DE (Storn and Price 1997), ES (Beyer 2001), GA (Goldberg 1998), HS (Geem et al. 2001), PBIL (Shumeet 1994) and PSO (Kennedy and Eberhart 1995).

In the following experiments, for CS and CCS, we will use the same parameters that is discovery rate $p_a = 0.25$.

Table 2 Benchmark functions

No.	Name	Definition
F01	Ackley	$f(\overrightarrow{x}) = 20 + e - 20\mathrm{e}^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - \mathrm{e}^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$
F02	Alpine	$f(\overrightarrow{x}) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1 x_i \rvert$
F03	Brown	$f(\overrightarrow{x}) = \sum_{i=1}^{n-1} [(x_i^2)^{x_{i+1}^2+1} + (x_{i+1}^2)^{x_i^2+1}]$
F04	Csendes	$f(\overrightarrow{x}) = \sum_{i=1}^{n} x_i^6 \left(2 + \sin \frac{1}{x_i}\right)$
F05	Dixon & Price	$f(\overrightarrow{x}) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$
F06	Fletcher–Powell	$f(\overrightarrow{x}) = \sum_{i=1}^{n} (A_i - B_i)^2,\ A_i = \sum_{j=1}^{n} (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j)$ $B_i = \sum_{j=1}^{n} (a_{ij} \sin x_j + b_{ij} \cos x_j)$
F07	Griewank	$f(\overrightarrow{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
F08	Holzman 2 function	$f(\overrightarrow{x}) = \sum_{i=1}^{n} i x_i^4$
F09	Levy	$f(\overrightarrow{x}) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1} [(y_i - 1)^2(1 + 10\sin^2(\pi y_i + 1))]$ $+ (y_n - 1)^2(1 + 10\sin^2(2\pi y_n)),\ y_i = 1 + (x_i - 1)/4$
F10	Pathological function	$f(\overrightarrow{x}) = \sum_{i=1}^{n} \left(0.5 + \frac{\sin^2\sqrt{100x_i^2 + x_{i+1}^2} - 0.5}{1 + 0.001(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2)^2}\right)$
F11	Penalty #1	$f(\overrightarrow{x}) = \frac{\pi}{30}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})]\right.$ $\left. + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4),\ \ y_i = 1 + 0.25(x_i + 1)$
F12	Penalty #2	$f(\overrightarrow{x}) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})]\right.$ $\left. + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
F13	Perm #1	$f(\overrightarrow{x}) = \sum_{k=1}^{n} \left[\sum_{i=1}^{n} (i^k + 0.5)\left(\left(\frac{x_i}{i}\right)^k - 1\right)\right]^2$
F14	Perm #2	$f(\overrightarrow{x}) = \sum_{k=1}^{n} \left[\sum_{i=1}^{n} (i + 10)\left(x_i^k - \left(\frac{1}{i}\right)^k\right)\right]^2$
F15	Powell	$f(\overrightarrow{x}) = \sum_{i=1}^{n/4} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2$ $+ (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$
F16	Quartic with noise	$f(\overrightarrow{x}) = \sum_{i=1}^{n} (i \cdot x_i^4 + U(0, 1))$
F17	Rastrigin	$f(\overrightarrow{x}) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi x_i))$
F18	Rosenbrock	$f(\overrightarrow{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
F19	Schwefel 2.26	$f(\overrightarrow{x}) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin(\lvert x_i \rvert^{1/2})$
F20	Schwefel 1.2	$f(\overrightarrow{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$
F21	Schwefel 2.22	$f(\overrightarrow{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$

Table 2 continued

F22	Schwefel 2.21	$f(\vec{x}) = \max_i \{\lvert x_i \rvert, \quad 1 \le i \le n\}$
F23	Sphere	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$
F24	Step	$f(\vec{x}) = 6 \cdot n + \sum_{i=1}^{n} \lfloor x_i \rfloor$
F25	Sum function	$f(\vec{x}) = \sum_{i=1}^{n} i x_i^2$
F26	Zakharov	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$
F27	Wavy1	$f(\vec{x}) = \sum_{i=1}^{n} \lvert 2(x_i - 24) + (x_i - 24)\sin(x_i - 24)\rvert$

In benchmark function F06, the matrix elements $\mathbf{a}_{n\times n}, \mathbf{b}_{n\times n} \in (-100, 100)$, $\alpha_{n\times 1} \in (-\pi, \pi)$ are drawn from uniform distribution. In benchmark functions F11 and F12, the definition of the function $u(x_i, a, k, m)$ is as follows:

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$$

Table 3 Minimum values obtained by CCS algorithm with different chaotic maps

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
F01	**1.00**	1.03	1.03	1.95	1.03	1.94	1.02	1.01	1.02	1.03	**1.00**	**1.00**
F06	1.11	1.06	1.05	5.96	1.03	5.93	1.06	1.09	1.08	1.04	**1.00**	1.02
F07	**1.00**	1.10	1.10	20.88	1.11	20.69	1.03	1.02	1.10	1.05	1.12	1.13
F11	5.27	4.47	**1.00**	1.2E6	3.81	1.1E6	2.61	3.51	1.04	10.56	1.81	6.16
F12	1.07	1.74	1.56	4.3E3	1.34	4.1E3	1.49	1.01	**1.00**	1.65	1.03	1.31
F16	1.07	1.37	1.19	592.57	1.03	582.85	1.02	1.27	1.34	1.10	**1.00**	1.24
F17	1.09	**1.00**	1.04	2.52	1.12	2.45	1.07	1.07	1.16	1.14	1.08	1.09
F18	1.33	**1.00**	1.20	33.53	1.30	34.64	1.25	1.29	1.44	1.44	1.18	1.26
F19	1.03	1.01	1.01	1.16	1.01	1.13	1.01	1.03	1.02	1.03	1.02	**1.00**
F20	1.03	1.08	1.11	29.01	1.08	28.78	1.11	1.04	**1.00**	1.07	1.06	1.06
F21	1.07	1.06	1.03	4.68	1.07	4.76	1.07	1.04	1.03	1.04	**1.00**	1.05
F22	**1.00**	1.01	1.03	3.71	1.04	3.69	1.01	**1.00**	1.03	1.03	1.02	1.01
F23	1.07	1.04	1.02	30.64	1.09	30.01	**1.00**	1.07	1.21	1.01	1.02	1.03
F24	1.12	1.08	1.07	21.86	1.10	22.23	1.01	1.10	1.08	**1.00**	1.04	1.06

In addition, the settings of population size, elitism parameter and maximum generation are the same as Sect. 4.1. For other methods, their parameters are set as follows (Wang et al. 2014g): For ACO, initial pheromone value $\tau_0 = 1\text{E}{-6}$, pheromone update constant $Q = 20$, exploration constant $q_0 = 1$, global pheromone decay rate $\rho_g = 0.9$, local pheromone decay rate $\rho_l = 0.5$, pheromone sensitivity $\alpha = 1$ and visibility sensitivity $\beta = 5$; for DE, a weighting factor $F = 0.5$ and a crossover constant $\text{CR} = 0.5$; For ES, the number of offspring $\lambda = 10$ produced in each generation, and standard deviation $\sigma = 1$ for changing solutions. For GA, we used roulette wheel selection, single-point crossover with a crossover probability of 1 and a mutation probability of 0.01. For HS, we set HM accepting rate = 0.75, and pitch adjusting rate = 0.7. For KH, we used the foraging speed $V_f = 0.02$, the maximum diffusion speed $D^{\max} = 0.005$, the maximum induced speed $N^{\max} = 0.01$. For PBIL, we used a learning rate of 0.05, 1 good population member and 0 bad population members to use to update the probability vector each generation, an elitism parameter of 1 and a 0 probability vector mutation rate. For PSO, an inertial constant = 0.3, a cognitive constant = 1 and a social constant for swarm interaction = 1.

It is well known that the metaheuristic methods are generally based on random distribution. In order to remove the influence of the randomness, 1000 independent runs are implemented for each method. Tables 5, 6 and 7 illustrate the results of the simulations.

From Table 5, we see that, on average, CCS is well capable of finding the best values except F03, F04, F07, F10, F19 and F27. ACO performs best on F07, F19, F24 and F27 function. Table 6 shows that CCS performs the best except

Table 4 Best values obtained by CCS algorithm with different chaotic maps

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
F01	1.05	1.10	**1.00**	2.77	1.17	2.75	1.07	1.10	1.09	1.03	**1.00**	1.10
F06	1.86	1.64	2.33	9.63	1.72	12.01	1.64	**1.00**	1.74	2.10	2.06	1.49
F07	1.52	1.30	1.15	38.93	1.42	36.35	1.33	1.20	**1.00**	1.33	1.55	1.53
F11	1.32	2.12	1.97	1.5E7	1.50	8.8E6	2.21	1.64	**1.00**	1.48	1.12	2.06
F12	7.12	12.33	1.55	1.5E7	18.33	1.0E7	1.28	6.16	2.37	5.50	16.73	**1.00**
F16	1.61	3.18	2.21	6.2E3	**1.00**	4.0E3	2.54	3.24	2.92	1.06	2.79	3.29
F17	1.77	1.46	1.35	4.00	1.41	3.59	1.68	**1.00**	1.45	1.65	1.63	1.52
F18	1.67	1.25	1.58	40.58	1.69	41.14	1.46	1.52	**1.00**	1.58	1.43	1.57
F19	1.05	1.08	1.14	1.15	1.09	1.41	**1.00**	1.30	1.15	1.04	1.19	1.02
F20	1.37	1.14	1.12	68.30	1.87	46.42	1.64	2.00	1.79	**1.00**	1.26	1.37
F21	1.62	1.12	1.38	7.94	1.06	8.98	1.38	1.61	1.31	**1.00**	1.19	1.36
F22	1.07	1.07	1.12	5.77	1.27	5.38	1.20	1.24	1.23	**1.00**	1.02	1.09
F23	**1.00**	1.30	1.91	91.52	2.23	47.86	1.64	1.15	1.19	2.11	1.53	1.27
F24	1.42	1.19	**1.00**	49.85	1.01	47.10	1.65	1.61	1.42	1.01	1.16	1.16

Table 5 Mean optimization results

	ACO	CCS	CS	DE	ES	GA	HS	PBIL	PSO
F01	1.52	**1.00**	1.68	1.22	1.87	1.68	1.91	1.95	1.62
F02	1.47	**1.00**	2.35	1.85	3.13	1.47	2.88	3.39	2.04
F03	5.17	4.12	8.81	**1.00**	48.37	2.74	20.01	9.02	12.44
F04	5.8E4	172.39	444.93	6.2E3	1.9E4	**1.00**	1.3E4	1.6E4	2.1E3
F05	18.61	**1.00**	52.91	6.28	248.81	15.09	235.33	351.76	22.45
F06	4.93	**1.00**	3.27	2.21	3.93	1.83	4.31	4.64	3.81
F07	**1.00**	1.93	7.62	2.17	11.29	3.49	21.05	24.34	8.24
F08	31.82	**1.00**	25.43	4.75	152.46	10.76	179.22	202.42	19.44
F09	2.08	**1.00**	4.78	2.06	11.24	3.18	9.45	12.13	4.71
F10	3.22	1.70	1.89	1.92	**1.00**	2.48	3.78	3.29	2.42
F11	9.5E6	**1.00**	2.7E5	3.2E4	1.8E6	9.1E3	3.6E6	4.5E6	8.9E4
F12	2.0E3	**1.00**	481.49	50.05	1.2E3	28.25	1.8E3	3.0E3	309.76
F13	1.1E6	**1.00**	62.46	203.61	7.7E3	1.8E6	1.1E3	1.1E6	4.6E4
F14	5.6E5	**1.00**	3.42	8.40	4.0E3	4.0E5	2.4E3	4.0E5	1.1E4
F15	34.82	**1.00**	11.37	20.35	73.10	8.64	48.11	49.74	20.91
F16	13.34	**1.00**	45.03	3.61	179.24	15.29	160.69	194.49	33.68
F17	1.51	**1.00**	1.69	1.28	1.99	1.36	1.84	2.05	1.49
F18	13.71	**1.00**	3.39	1.93	17.48	2.52	9.93	12.17	3.04
F19	**1.00**	2.37	2.82	2.13	2.65	1.04	3.17	3.22	3.09
F20	11.70	**1.00**	6.43	15.58	17.92	13.40	18.25	18.74	10.30
F21	3.06	**1.00**	2.96	1.34	4.20	2.17	3.38	3.39	2.70
F22	1.79	**1.00**	2.26	2.55	2.83	2.77	3.06	3.24	2.61
F23	10.67	**1.00**	7.98	2.23	20.38	5.99	20.75	22.07	7.61
F24	**1.00**	**1.00**	4.94	1.11	8.09	2.24	9.63	12.78	3.92
F25	10.46	**1.00**	8.80	2.16	23.51	6.68	25.89	25.89	9.63
F26	1.89	**1.00**	2.53	2.77	2.69	2.67	3.11	2.57	2.17
F27	**1.00**	1.38	2.76	1.43	2.45	1.62	3.54	3.73	2.46

F03, F04, F06, F07, F09, F10, F12, F19 and F27. GA and ACO can find the best values on F04, F06, F09, F19 and F07, F12, F27, respectively. Moreover, for the worst values as shown in Table 7, CCS is able to search for the best values on 23 of the 27 benchmarks (F01, F03–F06, F08–F18 and F20–F26). ACO performs best on F01, F02, F07, F10,

Table 6 Best optimization results

	ACO	CCS	CS	DE	ES	GA	HS	PBIL	PSO
F01	1.73	**1.00**	1.90	1.39	2.24	1.82	2.28	2.36	1.90
F02	1.24	**1.00**	2.98	2.25	3.90	1.47	3.39	4.14	2.50
F03	4.62	1.74	5.97	**1.00**	36.04	2.42	18.21	9.00	9.52
F04	9.3E4	53.24	73.18	3.2E3	2.4E4	**1.00**	1.6E4	1.0E4	2.6E3
F05	103.89	**1.00**	319.30	42.90	1.3E3	17.39	1.7E3	2.7E3	134.02
F06	3.49	1.10	3.60	2.59	5.27	**1.00**	5.52	6.45	4.75
F07	**1.00**	2.05	7.53	2.37	10.22	2.84	25.81	31.19	9.77
F08	44.46	**1.00**	39.82	7.11	213.98	3.85	352.59	435.85	27.01
F09	2.58	1.01	4.02	2.17	12.35	**1.00**	11.62	15.24	4.90
F10	4.18	1.32	1.97	1.97	**1.00**	2.99	5.15	4.14	2.30
F11	73.23	**1.00**	8.4E4	3.4E3	2.0E6	5.0E3	2.3E6	4.8E6	1.7E4
F12	**1.00**	297.20	7.2E5	9.2E4	2.1E6	2.9E3	3.7E6	6.3E6	5.1E5
F13	7.6E7	**1.00**	12.46	3.67	1.1E3	7.6E7	6.4E3	7.6E7	1.1E5
F14	1.2E7	**1.00**	16.74	57.26	9.1E3	1.2E7	7.20	1.2E7	8.7E3
F15	34.88	**1.00**	12.06	30.85	82.33	5.53	69.47	55.96	17.29
F16	23.95	**1.00**	92.83	8.68	386.11	16.11	454.32	483.72	44.17
F17	1.24	**1.00**	1.81	1.36	2.10	1.44	1.88	1.99	1.61
F18	9.12	**1.00**	3.08	1.41	15.73	1.89	9.69	9.42	2.95
F19	1.61	4.41	5.33	3.69	4.93	**1.00**	5.71	6.07	4.05
F20	15.04	**1.00**	9.94	15.36	23.09	15.66	24.41	27.59	13.44
F21	2.52	**1.00**	2.49	1.39	4.63	2.40	3.67	3.68	2.30
F22	1.38	**1.00**	2.10	2.67	2.98	2.51	3.20	3.63	2.75
F23	11.25	**1.00**	11.57	3.41	31.44	7.96	33.94	28.98	9.61
F24	1.41	**1.00**	4.61	1.46	9.92	2.16	12.19	17.75	4.42
F25	9.11	**1.00**	9.27	2.51	33.85	5.27	35.62	35.42	11.48
F26	1.99	**1.00**	2.84	2.82	3.03	2.66	3.60	3.27	2.41
F27	**1.00**	1.17	2.78	1.48	2.58	1.58	3.85	4.36	2.60

F19 and F27 function. As is evident, replacing step size α with chaotic maps can definitely improve the performance of the CS.

Moreover, by carefully looking at the Tables 5, 6 and 7, we can see, the obtained results indicate that incorporating chaotic maps in the CS model would lead to a significant increase in the performance of the CS. It is apparent that when chaotic map is taken into account, the performance is improved as compared to the other methods. This verifies the effectiveness and the ability of the proposed CCS algorithm in solving the global numerical optimization problem.

Further, to prove the superiority of the CCS more clearly, several representative convergence results for CCS and CS are shown in Figs. 3, 4, 5, 6 and 7. The values shown in Figs. 3, 4, 5, 6 and 7 are the average objective function optimum.

From Fig. 3, we can draw the conclusion that CCS is significantly superior to the CS method, especially F01.

Figure 4 illustrates the function values for F07–F10 function. For this case, the figure clearly shows that CCS performs significantly better than CS method.

From Fig. 5, CCS is significantly superior as compared to the CS algorithm, though both CCS and CS have similar performance on F11 at the end of optimization.

Figure 6 shows the results for F18–F20 and F22 function. From Fig. 6, apparently, CCS outperforms CS method in this example.

Figure 7 shows the performance achieved for F23–F25, F05 and F27 function. It can be seen that CCS has a faster speed of convergence than CS method.

From above analyses about the Figs. 3, 4, 5, 6 and 7, we conclude that the substitution of step size with an appropriate chaotic map demonstrates superiority in solving global optimization problems.

4.3 Sensor selection problem

Except the standard functions discussed in the section above, one more engineering optimization problem is also used to validate the CCS method. The sensor selection problem can be considered as a test problem to validate the CCS method.

Table 7 Worst optimization results

	ACO	CCS	CS	DE	ES	GA	HS	PBIL	PSO
F01	**1.00**	**1.00**	1.03	**1.00**	1.11	1.01	1.15	1.17	**1.00**
F02	**1.00**	1.10	1.48	1.10	1.74	1.10	1.58	1.85	1.21
F03	1.53	**1.00**	1.73	1.63	8.35	1.63	7.73	2.74	2.20
F04	1.5E3	**1.00**	27.04	7.3E3	7.3E3	7.3E3	7.3E3	7.3E3	7.3E3
F05	27.44	**1.00**	46.11	5.59	385.02	13.21	197.31	521.87	20.18
F06	5.67	**1.00**	3.88	2.81	3.95	1.98	4.14	5.25	4.69
F07	**1.00**	3.15	10.78	2.73	8.71	3.11	27.27	35.04	10.85
F08	44.46	**1.00**	88.51	15.53	246.28	12.20	632.31	442.70	27.01
F09	1.48	**1.00**	3.73	2.09	10.74	3.41	8.43	11.35	5.37
F10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
F11	2.0E7	**1.00**	4.1E5	3.3E4	3.3E6	3.5E3	3.2E6	6.9E6	9.9E4
F12	6.1E3	**1.00**	402.38	79.81	1.1E3	17.30	4.9E3	6.0E3	332.33
F13	2.4E5	**1.00**	4.89	12.74	6.2E3	1.0E6	28.12	2.4E5	1.6E3
F14	3.5E5	**1.00**	1.39	23.42	2.3E5	3.5E5	2.3E5	3.5E5	2.3E5
F15	34.57	**1.00**	16.38	20.12	117.31	7.59	53.40	78.99	19.22
F16	20.28	**1.00**	75.73	3.37	150.16	27.01	365.69	365.69	365.69
F17	1.24	**1.00**	1.81	1.60	2.36	1.48	2.03	2.63	1.86
F18	24.52	**1.00**	2.95	2.04	17.41	4.25	9.28	16.46	2.87
F19	**1.00**	2.37	2.58	1.93	2.73	1.01	3.13	3.00	2.90
F20	8.54	**1.00**	6.57	14.68	18.88	15.29	13.86	18.21	9.83
F21	3.75	**1.00**	3.85	1.31	4.37	2.16	3.83	3.22	2.16
F22	1.12	**1.00**	2.46	2.17	2.42	3.02	2.60	3.19	2.39
F23	6.37	**1.00**	6.35	1.96	27.27	27.27	27.27	27.27	27.27
F24	1.15	**1.00**	3.60	1.48	10.50	2.40	12.85	16.80	3.68
F25	8.30	**1.00**	8.45	2.65	32.34	9.66	34.10	32.29	16.44
F26	1.38	**1.00**	2.19	2.75	2.88	2.85	3.81	2.26	2.18
F27	**1.00**	1.78	2.78	1.65	3.50	2.31	4.14	4.36	2.99

The Modular Aero Propulsion System Simulation (MAPSS) (Simon 2008) is used as the engine simulation in sensor selection problem. The model of the turbofan engine can be represented in the form of the discretized time invariant equations shown as follows:

$$\begin{aligned} x(k+1) &= f[x(k), u(k), p(k)] + w_x(k) \\ p(k+1) &= p(k) + w_p(k) \\ y(k) &= g[x(k), u(k), p(k)] + e(k), \end{aligned} \tag{2}$$

where k, x, u, p and y are the time index, three-element state vector, three-element control vector, ten-element health parameter vector and measurement vector, respectively. Between measurement times their deviations can be approximated by the zero mean noise $w_p(k)$. The $w_x(k)$ and $e(k)$ mean inaccuracies in the system model and measurement noise, respectively.

The state vector x and the health parameter vector p in Eq. (2) can be estimated by a Kalman filter, and its uncertainty can be given by the error covariance $\sum$. Here, the covariance $\sum$ is a 13×13 matrix, because this problem includes three states and ten health parameters. In the sensor selection problem, only the health parameter estimation errors should be taken into account, so close attention can be simply paid to the diagonal elements $\sum(i,i)$ $(i = 4, 5, \ldots, 13)$.

Based on the above analyses, the object function of the health estimation problem can be expressed as

$$J = \sum_{i=4}^{13} \sqrt{\frac{\sum(i,i)}{\sum_0(i,i)}} + \frac{\alpha C}{C_0}, \tag{3}$$

where $\sum_0$ and C_0 are used for normalization. $\sum_0$is the covariance, and C_0 is the cost of setting the aircraft engine with all 11 sensors. α is a scale factor that can balance the importance of financial cost and estimation accuracy.

It is clear that the selection of sensors to generate the minimum for J is essentially an optimization problem.

In fact, optimization methods can be used to solve the sensor selection problem. Here, nine optimization methods are used to search for a sub-optimal sensor set. The results on the sensor selection problem are recorded in Table 8. It can

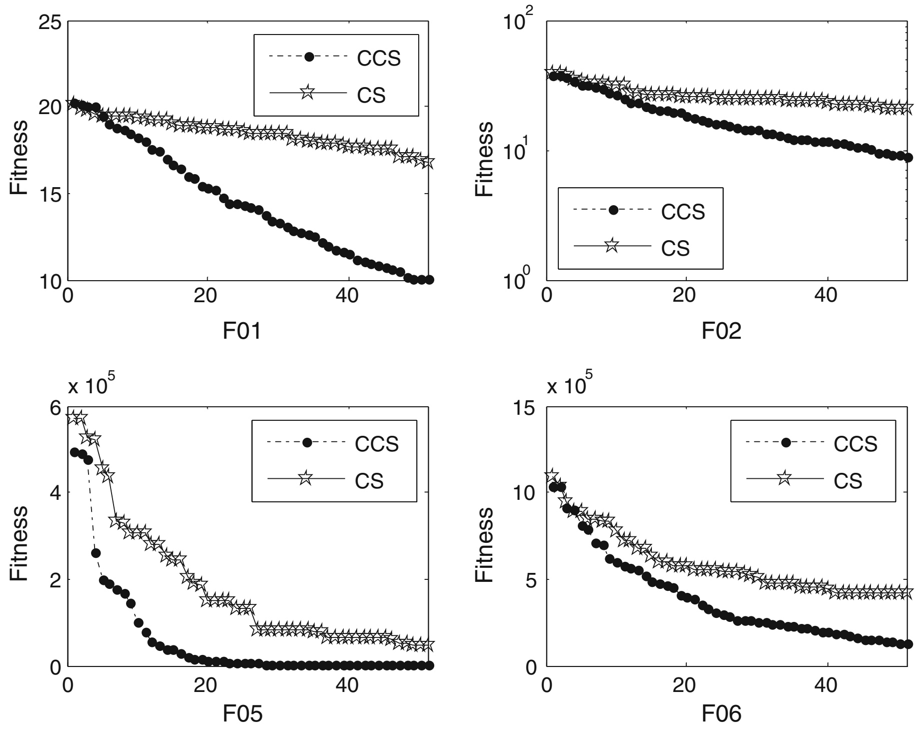

Fig. 3 Convergent curves of the F01, F02, F05 and F06 function

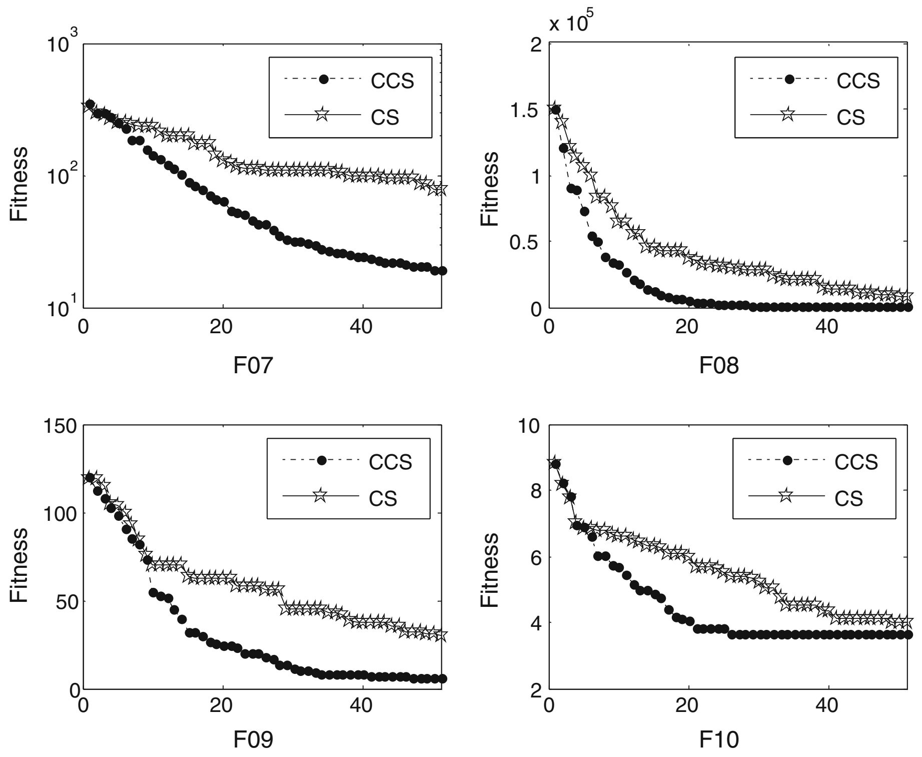

Fig. 4 Convergent curves of the F07–F10 function

be seen that CCS yields better performance than the other eight methods in terms of average, best and worst performance. Furthermore, the Std of CCS is much smaller than other methods. That is, CCS has a relatively high possibility of finding the satisfactory sensor set.

5 Conclusion

In the proposed study, chaos has been combined with the standard CS which yields a new improved version of the CS algorithm, namely CCS algorithm. Twelve chaotic maps

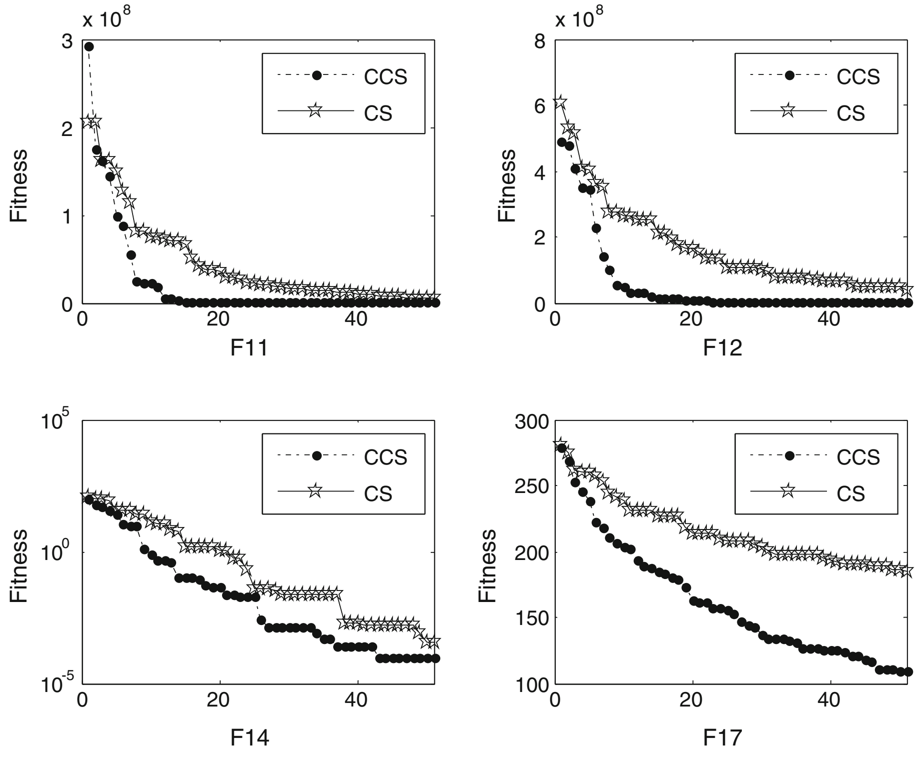

Fig. 5 Convergent curves of the F11, F12, F14 and F17 function

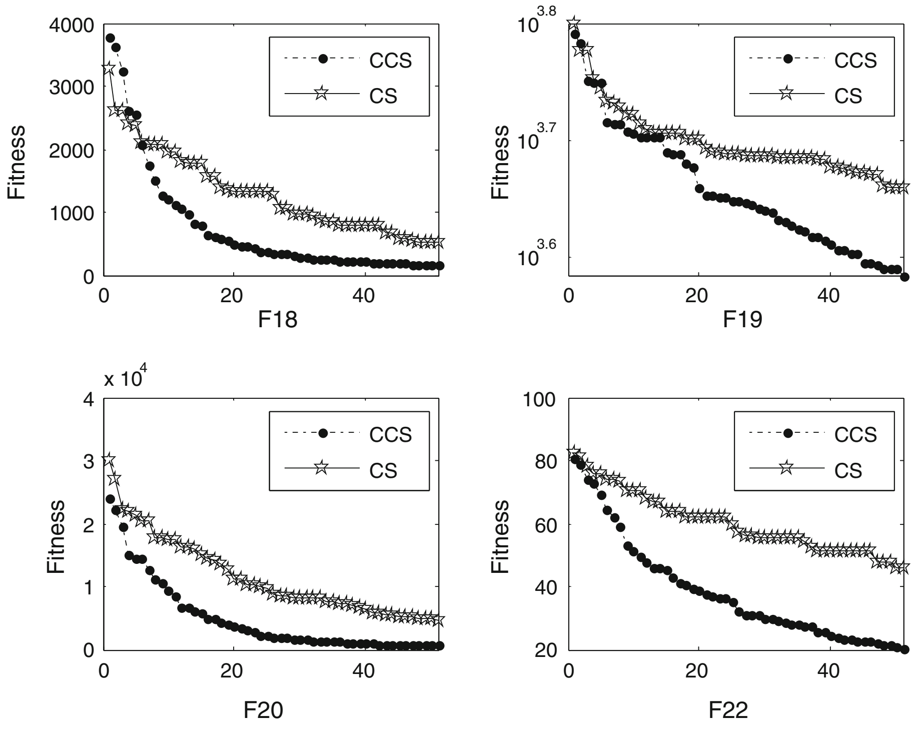

Fig. 6 Convergent curves of the F18–F20 and F22 function

are used to tune the step size, α, of the CS. By series of simulations on various chaotic CS variants, the algorithm in combination of Sinusoidal map in place of α is the best chaotic CS. From the experimental results, we observe that the tuned CS significantly enhances the ability of the global search as well as the quality of the results. As per the results of the nine approaches on the benchmarks and sensor selection problem, it can be seen that the CCS significantly enhances the search ability of the CS on most benchmark problems and engineering problem.

Various issues are worthy of further study in optimization problems, and some more efficient methods may be put

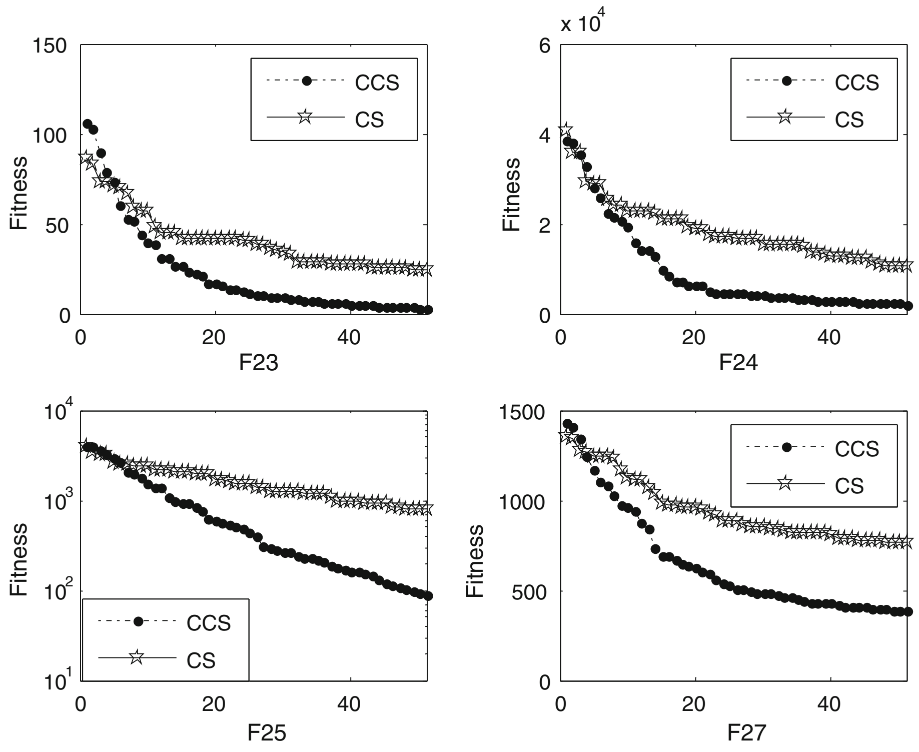

Fig. 7 Convergent curves of the F23–F25, F05 and F27 function

Table 8 The final values for the sensor selection problem

	Best	Mean	Worst	Std
ACO	8.19	8.24	8.30	0.029
CCS	**8.02**	**8.08**	**8.11**	**0.019**
CS	8.11	8.16	8.22	0.034
DE	8.06	8.10	8.12	0.020
ES	8.13	8.18	8.17	0.039
GA	8.03	8.09	8.16	0.047
HS	8.07	8.16	8.20	0.045
PBIL	8.10	8.19	8.25	0.044
PSO	8.09	8.15	8.15	0.041

forward, relying on the specific engineering problems. In future, we will focus on such challenging issues. On the one hand, the CCS method would be utilized to solve other practical engineering problems. We strongly believe that CCS can become an efficient and promising method for solving other real-world engineering problems. At the same time, some new meta-hybrid approaches would also be put forward to address optimization problems, based on the current studies.

Acknowledgments This work was supported by Research Fund for the Doctoral Program of Jiangsu Normal University (No. 9213614102) and National Natural Science Foundation of China (No. 61305149).

References

Beyer H (2001) The theory of evolution strategies. Springer, New York
Cai X, Fan S, Tan Y (2012) Light responsive curve selection for photosynthesis operator of APOA. Int J Bio-Inspir Comput 4(6):373–379
Dorigo M, Stutzle T (2004) Ant colony optimization. MIT Press, Cambridge
Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B Cybern 26(1):29–41. doi:10.1109/3477.484436
Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845. doi:10.1016/j.cnsns.2012.05.010
Gandomi AH, Yang X-S (2014) Chaotic bat algorithm. J Comput Sci 5(2):224–232. doi:10.1016/j.jocs.2013.10.002
Gandomi AH, Yang X-S, Alavi AH (2011) Mixed variable structural optimization using firefly algorithm. Comput Struct 89(23–24):2325–2336. doi:10.1016/j.compstruc.2011.08.002
Gandomi AH, Yang XS, Talatahari S, Alavi AH (2013a) Metaheuristic applications in structures and infrastructures. Elsevier, Waltham
Gandomi AH, Yang X-S, Alavi AH, Talatahari S (2013b) Bat algorithm for constrained optimization tasks. Neural Comput Appl 22(6):1239–1255. doi:10.1007/s00521-012-1028-9
Gandomi AH, Yang XS, Talatahari S, Alavi AH (2013c) Firefly algorithm with chaos. Commun Nonlinear Sci Numer Simulat 18(1):89–98. doi:10.1016/j.cnsns.2012.06.009
Gandomi AH, Yun GJ, Yang X-S, Talatahari S (2013d) Chaos-enhanced accelerated particle swarm optimization. Commun Nonlinear Sci Numer Simulat 18(2):327–340. doi:10.1016/j.cnsns.2012.07.017
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 138:392–402. doi:10.1016/j.neucom.2014.01.023

Jia D, Zheng G, Khurram Khan M (2011) An effective memetic differential evolution algorithm based on chaotic local search. Inf Sci 181(15):3175–3187. doi:10.1016/j.ins.2011.03.018

Kaveh A, Sheikholeslami R, Talatahari S, Keshvari-Ilkhichi M (2014) Chaotic swarming of particles: a new method for size optimization of truss structures. Adv Eng Softw 67:136–147. doi:10.1016/j.advengsoft.2013.09.006

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Paper presented at the proceeding of the IEEE international conference on neural networks, Perth, 27 November 1995–1 December 1995

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

Li X, Yin M (2013a) 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

Li X, Yin M (2013b) 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 (2015) Modified cuckoo search algorithm with self adaptive parameter method. Inf Sci 298:80–97. doi:10.1016/j.ins.2014.11.042

Li X, Zhang J, Yin M (2014) 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

Mirjalili S, Lewis A (2013) S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evol 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 25(3–4):663–681. doi:10.1007/s00521-013-1525-5

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

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

Nouhi B, Talatahari S, Kheiri H, Cattani C (2013) Chaotic charged system search with a feasible-based method for constraint optimization problems. Math Probl Eng 2013:1–8. doi:10.1155/2013/391765

Shumeet B (1994) Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Carnegie Mellon University, Pittsburgh, PA

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

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

Talatahari S, Farahmand Azar B, Sheikholeslami R, Gandomi AH (2012) Imperialist competitive algorithm combined with chaos for global optimization. Commun Nonlinear Sci Numer Simulat 17(3):1312–1319. doi:10.1016/j.cnsns.2011.08.021

Talatahari S, Kheirollahi M, Farahmandpour C, Gandomi AH (2013) A multi-stage particle swarm for optimum design of truss structures. Neural Comput Appl 23(5):1297–1309. doi:10.1007/s00521-012-1072-5

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

Wang G-G, Gandomi AH, Alavi AH (2013b) A chaotic particle-swarm krill herd algorithm for global numerical optimization. Kybernetes 42(6):962–978. doi:10.1108/K-11-2012-0108

Wang G, Guo L, Wang H, Duan H, Liu L, Li J (2014a) 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, Zhao X, Chu HE (2014b) Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. Soft Comput. doi:10.1007/s00500-014-1502-7

Wang G-G, Guo L, Duan H, Wang H (2014c) A new improved firefly algorithm for global numerical optimization. J Comput Theor Nanos 11(2):477–485. doi:10.1166/jctn.2014.3383

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

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

Wang G-G, Gandomi AH, Alavi AH, Hao G-S (2014f) 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 (2014g) 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-G, Deb S, Cui Z (2015) Monarch butterfly optimization. Neural Comput Appl. doi:10.1007/s00521-015-1923-y

Xie L, Zeng J, Formato RA (2012) Selection strategies for gravitational constant G in artificial physics optimisation based on analysis of convergence properties. Int J Bio-Inspir Comput 4(6):380–391

Yang XS (2010a) A new metaheuristic bat-inspired algorithm. In: González JR, Pelta DA, Cruz C, Terrazas G, Krasnogor N (eds) Nature inspired cooperative strategies for optimization (NICSO 2010), vol 284. Studies in computational intelligence. Springer, Heidelberg, 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, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1(4):330–343. doi:10.1504/IJMMNO.2010.03543

Yang XS, Gandomi AH (2012) Bat algorithm: a novel approach for global engineering optimization. Eng Comput 29(5):464–483. doi:10.1108/02644401211235834

Yang X-S, Hosseini SSS, Gandomi AH (2012) Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. Appl Soft Compt 12(3):1180–1186. doi:10.1016/j.asoc.2011.09.017

Yang XS, Gandomi AH, Talatahari S, Alavi AH (2013) Metaheuristics in water. Geotechnical and transport engineering. Elsevier, Waltham

Yang X-S, Karamanoglu M, He X (2014) Flower pollination algorithm: a novel approach for multiobjective optimization. Eng Optim 46(9):1222–1237. doi:10.1080/0305215X.2013.832237

Zhang Z, Feng Z (2012) Two-stage updating pheromone for invariant ant colony optimization algorithm. Expert Syst Appl 39(1):706–712. doi:10.1016/j.eswa.2011.07.062

Zhang Y, Huang D, Ji M, Xie F (2011) Image segmentation using PSO and PCM with Mahalanobis distance. Expert Syst Appl 38(7):9036–9040. doi:10.1016/j.eswa.2011.01.041

Zhang Z, Zhang N, Feng Z (2014) Multi-satellite control resource scheduling based on ant colony optimization. Expert Syst Appl 41(6):2816–2823. doi:10.1016/j.eswa.2013.10.014

Zou D, Gao L, Li S, Wu J (2011) An effective global harmony search algorithm for reliability problems. Expert Syst Appl 38(4):4642–4648. doi:10.1016/j.eswa.2010.09.120

Zhao X, Lin W, Zhang Q (2014a) Enhanced particle swarm optimization based on principal component analysis and line search. Appl Math Comput 229:440–456. doi:10.1016/j.amc.2013.12.068

Zhao X, Liu Z, Yang X (2014b) A multi-swarm cooperative multistage perturbation guiding particle swarm optimizer. Appl Soft Compt 22:77–93. doi:10.1016/j.asoc.2014.04.042

Zou D, Gao L, Wu J, Li S, Li Y (2010) A novel global harmony search algorithm for reliability problems. Comput Ind Eng 58(2):307–316. doi:10.1016/j.cie.2009.11.003

Zou D, Gao L, Li S, Wu J (2011) Solving 0–1 knapsack problem by a novel global harmony search algorithm. Appl Soft Compt 11(2):1556–1564. doi:10.1016/j.asoc.2010.07.019