Neural Comput & Applic (2019) 31 (Suppl 1):S653–S670
DOI 10.1007/s00521-017-3012-x

ORIGINAL ARTICLE

A hybridization of cuckoo search and particle swarm optimization for solving optimization problems

Rui Chi[1] · **Yi-xin Su**[1] · **Dan-hong Zhang**[1] · **Xue-xin Chi**[1] · **Hua-jun Zhang**[1]

Received: 8 November 2016 / Accepted: 11 April 2017 / Published online: 11 May 2017

Abstract A new hybrid optimization algorithm, a hybridization of cuckoo search and particle swarm optimization (CSPSO), is proposed in this paper for the optimization of continuous functions and engineering design problems. This algorithm can be regarded as some modifications of the recently developed cuckoo search (CS). These modifications involve the construction of initial population, the dynamic adjustment of the parameter of the cuckoo search, and the incorporation of the particle swarm optimization (PSO). To cover search space with balance dispersion and neat comparability, the initial positions of cuckoo nests are constructed by using the principle of orthogonal Lation squares. To reduce the influence of fixed step size of the CS, the step size is dynamically adjusted according to the evolutionary generations. To increase the diversity of the solutions, PSO is incorporated into CS using a hybrid strategy. The proposed algorithm is tested on 20 standard benchmarking functions and 2 engineering optimization problems. The performance of the CSPSO is compared with that of several meta-heuristic algorithms based on the best solution, worst solution, average solution, standard deviation, and convergence rate. Results show that in most cases, the proposed hybrid optimization algorithm performs better than, or as well as CS, PSO, and some other exiting meta-heuristic algorithms. That means that the proposed hybrid optimization algorithm is competitive to other optimization algorithms.

Keywords Cuckoo search · Particle swarm optimization · Hybrid optimization · Orthogonal Lation squares · Step size

1 Introduction

A general optimization problem with equality, inequality, and upper bound and lower bound constraints is stated as follows:

$$\min \quad f\left(\overrightarrow{x}\right)$$
$$s.t. \quad \begin{cases} g_s\left(\overrightarrow{x}\right) \le 0, \quad s = 1, 2, \ldots, n_g, \\ h_t\left(\overrightarrow{x}\right) = 0, \quad t = 1, 2, \ldots, n_h, \\ \overrightarrow{x} = \left\{(x_1, x_2, \ldots, x_D)^T \in R^D | l_i \le x_i \le u_i, i = 1, \ldots, D\right\}, \\ \overrightarrow{l} = (l_1, l_2, \ldots, l_D)^T, \overrightarrow{u} = (u_1, u_2, \ldots, u_D)^T. \end{cases} \tag{1}$$

where $\overrightarrow{x} = (x_1, x_2, \ldots, x_D)^T$ is the decision vector, R^D is the D-dimensional Euclidean space, $f\left(\overrightarrow{x}\right)$ is the objective function, $g_s\left(\overrightarrow{x}\right) \le 0$ are the inequality constraints, $h_t\left(\overrightarrow{x}\right) = 0$ are the equality constraints, n_g is the number of inequality constraints and n_h is the number of equality constraints, and $\overrightarrow{l}$ and $\overrightarrow{u}$ are the lower and upper bounds of the decision variables, respectively. Without the loss of generality, the present paper aims at the sets of minimization problems.

The above general optimization problem (1) is the most general and important type of design optimization problems in engineering. Many real-world engineering optimization problems are often highly nonlinear and involve a number of different design variables under complex constraints [1]. The classical optimizations such as Newton method [2], the gradient technique [3], and the interior method [4] have been developed to solve the engineering optimization problems.

✉ Yi-xin Su
suyixin@whut.edu.cn

[1] School of Automation, Wuhan University of Technology, Wuhan 430070, China

However, the classical techniques suffer from various drawbacks. The Newton method is limited to continuity of the optimization problem. The practicality of gradient-based techniques has been reduced by the derivatives for nonlinearity of the engineering problems and the difficulty of generating automatically objective functions [5]. The interior method is also not very suitable to obtain the optima of engineering problems because it is more likely to return a local minimum and is always time consuming [6].

In view of these shortcomings of the classical techniques, for solving the optimization problem (1) efficiently, some researchers have focused on the application of heuristic algorithms. The heuristic algorithms can be broadly classified into three categories [7, 8]: constructive heuristic algorithms, improvement heuristic algorithms, and hybrid heuristic algorithms. In general, constructive heuristics can find a solution in a reasonable time but the quality of the solution is not very satisfactory [7]. The improvement heuristics are mainly the meta-heuristic evolutionary algorithms. In the last decade, meta-heuristic evolutionary algorithms such as genetic algorithm (GA) [9], simulated annealing (SA) [10], particle swarm optimization (PSO) [11, 12], differential evolution (DE) [13], ant colony optimization (ACO) [14], bat-inspired algorithm (BA) [15], harmony search (HS) [16], dragonfly algorithm (DA) [17], and more recently, cuckoo search (CS) [18] have been developed to solve the engineering optimization problems. Improvement heuristic algorithms can normally converge to better-quality minimums, but they are timely and laborious processes.

Considering the restrictiveness of the constructive heuristic algorithms and the improvement heuristic algorithms, recently, the research concentration has expanded to hybrid meta-heuristic algorithms instead of a sole meta-heuristic algorithm. It has become evident that a combination of two or more meta-heuristic algorithms, called hybrid heuristic algorithms, is mostly efficient and can receive good application in dealing with the real-world engineering problems. For example, Nearchou [19] proposed a hybrid SA algorithm which integrates the basic structure of a SA algorithm together with features borrowed from the fields of GA and local search techniques for solving the flow shop scheduling problem, Alikhani et al. [20] hybridized electromagnetism-like mechanism algorithm and Solis and Wets local search method for continuous optimization problems, Costa et al. [21] used a hybridization of genetic algorithm and pattern search method for constrained global optimization problems, Yildiz [22] hybridized an artificial immune algorithm with a hill climbing local search algorithm for optimization problems, Melo et al. [23] proposed a multi-view differential evolution for constrained engineering design problems, Su et al. [24] hybridized the particle swarm algorithm with the differential evolution algorithm to solve the multi-objective optimization problems, and Kanagaraj et al. [25, 26] hybridized genetic algorithm and cuckoo search algorithm to solve several constrained optimization problems. They have obtained good effects for the engineering optimization problems.

Although some improved techniques for the engineering optimization problems have been achieved, the further investigation is also needed due to the complexity of conflicting objectives and various constraints [27]. As one of the existing meta-heuristic algorithms, an evolution technique, called CS [1, 6, 7, 18, 28–30], is a population-based heuristic evolutionary algorithm that is simple to implement and has few parameters to be tuned [7]. This algorithm is based on the interesting breeding behavior such as brood parasitism of certain species of cuckoos. And it has gained promising importance and applicability in many fields like shop scheduling, parameter optimization in structure designing and machining processes, networking, and so on [6]. CS and its application in many fields have been discussed in [28].

Along with the development trend of hybrid meta-heuristic algorithms discussed above, this paper proposes a new hybrid algorithm combining cuckoo search and particle swarm optimization, called CSPSO in the following sections. The crucial idea behind CSPSO can be summarized as follows. Population of the algorithm is initialized by using the principle of orthogonal Lation squares, to cover search space with balance dispersion and neat comparability. A dynamically variable step size is proposed as a possible improvement over the original CS. PSO is incorporated to increase the diversity of population. The proposed CSPSO is used to optimize benchmark functions and solve engineering design problems. Result evaluation assures that the proposed algorithm provides better performance than other approaches.

The remainder of the paper is organized as follows. Section 2 briefly reviews the fundamentals of CS and PSO. Section 3 describes the proposed CSPSO algorithm. Sections 4–5 evaluate CSPSO using 20 standard benchmarking functions and 2 engineering optimization problems, compare the performance of CSPSO with other relative algorithms, and discuss the strengths and weaknesses of the CSPSO algorithm. Finally, the concluding remarks and some directions for future research are provided in Section 6.

2 The original PSO and original CS

In this section, the features of the original PSO, the original CS, and a hybridization of CS and PSO are discussed.

2.1 The original PSO

PSO is a meta-heuristic evolutionary technique proposed by Kennedy and Eberhart [11, 12]. The original intent of particle swarm theory is to graphically mimic the swarm

characteristics of a bird flock which is characterized by graceful but unpredictable choreography. Like other evolutionary techniques, PSO uses the common evolutionary computation method [31] as follows.

(1) It is initialized with a random population called swarm. Each individual of the swarm can be written as a particle, and each particle's position resembles a solution.
(2) It finds out the best solution by updating generations consecutively.
(3) The reproduction operation is on the basis of the old generation. A particle modifies its memorized values including velocity, position, personal best position, and global best position to achieve better position. The whole particles search the problem space for the best solution by the current optimal solutions.

Defined that D is the problem space dimension, and N is the number of particles. The position of particle i is noted as $\overrightarrow{y}_i = \left(\overrightarrow{y}_{i1}, \overrightarrow{y}_{i2}, \ldots, \overrightarrow{y}_{iD}\right)^T$, its velocity is defined as $\overrightarrow{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})^T$, and its best position (best previous performance) in history is called the personal best ($\overrightarrow{p}_i$). The best position found previously by all particles in the group is called the global best ($\overrightarrow{g}$). The updates of the swarm particles are accomplished using the following equations.

$$v_{ij}(k+1) = v_{ij}(k) + c_1 r_1 \left(p_{ij}(k) - y_{ij}(k)\right) + c_2 r_2 \left(g_j(k) - y_{ij}(k)\right) \quad (2)$$

$$y_{ij}(k+1) = y_{ij}(k) + v_{ij}(k+1) \quad (3)$$

where $i = 1, 2, \ldots, N, j = 1, 2, \ldots, D.k$ is the current generation. c_1 is the cognitive memory factor, c_2 is the social memory factor, and they are two constants. r_1 and r_2 (r_1, $r_2 \in [0, 1]$) are random numbers. Equation (2) calculates a new velocity for each swarm particle on the basis of its previous velocity $\overrightarrow{v}_i(k)$, its personal best position $\overrightarrow{p}_i(k)$, and the global best position $\overrightarrow{g}(k)$. Equation (3) updates the position of each swarm particle in problem space.

Performance evolution of the particles depends on the fitness of an evolution function that is predefined and related to the objective functions of the optimization problem [32]. In the process of optimizing minimization problem, when the fitness of the improved solution is lower than that of the previous solutions, the improved solution is closer to the best solution. $\overrightarrow{p}_i(k)$ of each swarm particle is updated only if its current position's fitness value is lower than the previous best value, otherwise $\overrightarrow{p}_i(k)$ will be unchanged. $\overrightarrow{g}(k)$ is always the best position at which the best fitness so far has been achieved. Mathematically, this can be formulated as Eqs. (4–5).

$$\overrightarrow{p}_i(k+1) = \begin{cases} \overrightarrow{y}_i(k+1), & \text{if } f\left(\overrightarrow{y}_i(k+1)\right) < f\left(\overrightarrow{p}_i(k)\right); \\ \overrightarrow{p}_i(k), & \text{otherwise.} \end{cases} \quad (4)$$

$$\overrightarrow{g}(k+1) = \operatorname{argmin} f\left(\overrightarrow{p}_i(k+1)\right) \quad (5)$$

where $f\left(\overrightarrow{y}_i(k+1)\right)$ is the fitness value of the current position of particle i. The pseudo code of the original PSO is given in Fig. 1.

2.2 The original CS

2.2.1 Cuckoo breeding behavior and Lévy flight

The concept of meta-heuristic technique known as cuckoo search algorithm [1, 18] was first proposed by Yang and Deb in the year 2009. It is a novel algorithm based on the breeding behavior of cuckoo species. Cuckoo hens lay their eggs in other birds' nests or the communal nests. While laying their eggs, they may remove the host birds' eggs from the nest to increase the hatching probability of their own eggs. In general, eggs of cuckoo hens hatch slightly earlier than the host birds' eggs, and the cuckoo chick grows faster. Once the first cuckoo egg hatches, the foremost action of the chick is to blindly roll the host eggs out of the nest, which can increase the cuckoo chicks' share of food provided by its host bird [28]. Apart from the propelling action of the cuckoo chick, it also mimics the host chicks' call in order to access more opportunity of feeding [29].

In the original CS, Yang and Deb also introduced the concept of Lévy flight. A cuckoo performs Lévy flight to search for a new nest. The Lévy flight process, named by the French mathematician Paul Lévy, is essentially a model of random walks that is characterized by random step lengths drawn from a power law distribution [30]. The foraging behavior of many animals and insects has the typical characteristics of Lévy flight [34, 35]. This process has previously been widely used to solve the problems in optimization and other fields of sciences [18, 33–35].

2.2.2 Cuckoo search implementation

As proposed by Yang, three idealized rules are suggested as follows [18].

(1) Each cuckoo lays one egg at a time, and a random nest is chosen to dump the egg.
(2) The best nest with high quality of eggs will pass onto the next generations.

Fig. 1 Pseudo code of PSO algorithm

Algorithm 1 Particle Swarm Optimization

Initialize a population of N swarm particles $\vec{y}_i(i=1,2,\ldots,N)$, the velocity of the swarm particles $\vec{v}_i(i=1,2,\ldots,N)$, $\vec{p}_i=\vec{y}_i(i=1,2,\ldots,N)$
Evaluate the quality/fitness $f(\vec{y}_i)$
$\vec{g}_i=\arg\min f(\vec{p}_i)$
  While ($k <$ Max Generation) and (the stop criterion is not met) do
    For each particle $i\in 1\rightarrow N$ do
      Update velocity using (2)
      Update position using (3)
       Evaluate the quality/fitness $f(\vec{y}_i(k+1))$
       If ($f(\vec{y}_i(k+1))<f(\vec{p}_i(k))$)
         Replace $\vec{p}_i(k)$ using the equation (4)
       End if
    End for
    Update $\vec{g}(k)$ using equation (5)
  End while
Output: the global best position $\vec{g}$

(3) The number of host nests is fixed and there is a probability $p_a\in[0,1]$ with which a host bird discovers an alien. In this case, the host bird can either discard the egg or the nest so as to build a completely new nest in a new location.

Based on the above-mentioned rules, the original CS is developed as follows.

In the CS technique, one egg in a nest resembles a solution and a cuckoo egg resembles a new solution. The algorithm begins with an initial population generated randomly. The population uses D-dimension parameter vector restricted by the upper and lower bounds as given in Eqs. (6–7).

$$\overrightarrow{l}=(l_1,l_2,\ldots,l_D)^T \tag{6}$$

$$\overrightarrow{u}=(u_1,u_2,\ldots,u_D)^T \tag{7}$$

During the generation of a new solution $\overrightarrow{x}_i(k+1)$, a Lévy flight is carried out as in Eq. (8).

$$\overrightarrow{x_i}(k+1)=\overrightarrow{x_i}(k)+\alpha\oplus \mathrm{Le}'\mathrm{vy}(s,\lambda) \tag{8}$$

where $\overrightarrow{x}_i(k)$ represents the current solution, k is the current generation, and $\alpha(\alpha>0)$ represents the step size. In general, α should be proportional to the scale of the studied problem, even though $\alpha=1$ can be used in common [18]. The product $\oplus$ means entry-wise multiplications. Lévy flights essentially resemble random walks, and their random steps follow the Lévy distribution as given in Eq. (9).

$$\mathrm{Le}'\mathrm{vy}(s,\lambda)\sim s^{-\lambda},(1<\lambda\leq 3) \tag{9}$$

Equation (9) has an infinite variance with an infinite mean [1]. Accordingly, the consecutive steps/jumps of a cuckoo have formed a process with random walks, and the process obeys a power law distribution with a heavy tail [1]. In fact, the process of generating a new solution can be seen as a stochastic equation for a random walk, and this random walk forms a Markov chain whose next location only depends on the current position and the transition probability [7]. In this way, the evolution process is to use the new and potentially better solutions to continually replace worse solutions. The final goal is to gain the best solution. The pseudo code of the original CS is presented in Fig. 2.

3 The hybrid CSPSO algorithm

3.1 Selection method of initial population

CS has succeeded in proving its superior performance, compared with PSO and GA [30]. But there is still some room for improvement in CS, we can control the intensification and diversification through the cuckoo's mobility in the search space to help the individual find much better solutions [30]. It is a serious issue that how to use a minimum number of

Fig. 2 Pseudo code of CS algorithm

Algorithm 2 Cuckoo Search Algorithm

Initialize objective function $f(\vec{x})$, $\vec{x} = (x_1, \ldots, x_D)^T$
  Generate an initial population of N host nests $\vec{x}_i (i = 1, 2, \ldots, N)$
  While ($k <$ Max Generation) and (the stop criterion is not met) do
    For each nest $i \in 1 \rightarrow N$ do
      Generate a new solution $\vec{x}_i(k+1)$ using (8-9)
      Evaluate the quality/fitness $f(\vec{x}_i(k))$
      If ($f(\vec{x}_i(k+1)) < f(\vec{x}_i(k))$)
        Replace i by the new solution
      End if
    End for
    Abandon worse nests with a fraction p_a and rebuild the corresponding new nests
    Keep the best solutions (or nests with quality solutions)
    Rank the solutions and find the current best
  End while
Post-process results and visualization

initial solutions to sample the distribution characteristic of the search space. A selection method of initial population presented here, called orthogonal arrays, is expounded in detail [36]. The key aim of using orthogonal arrays is to draw its advantage of balance dispersion and neat comparability.

Following the rule of probability and statistics, orthogonal arrays are generated based on the principle of orthogonal Lation squares. Here the orthogonal arrays, which include N initial solutions, can be constructed by the following two steps.

Step: 1 Calculate the dispersed coordinate points a_{ij} of each variable by using D-dimension vector in the problem space restricted by the upper and lower bounds.

$$a_{ij} = l_j + (i-1) \times \left(u_j - l_j\right) \Big/ (N-1) \quad (i = 1, 2, \ldots, N; j = 1, 2, \ldots, D) \tag{10}$$

Step: 2 Generate the initial solutions by using the principle of orthogonal Lation squares.

$$b_{ij} = a_{hj} \quad (i = 1, 2, \ldots, N; j = 1, 2, \ldots, D) \tag{11}$$

where $h = (i + j - 1) \bmod N$, and if $h = 0$, then $h = N$.

Hence, an initial individual of the population $\overrightarrow{b}_i = (b_{i1}, b_{i2}, \ldots, b_{iD})^T, (i = 1, 2, \ldots, N)$ is generated via the above mentioned steps. In the proposed CSPSO, the initial individual can also be denoted as: $\overrightarrow{x}_i$, $\overrightarrow{y}_i$ and $\overrightarrow{z}_i$, $\overrightarrow{x}_i = \overrightarrow{b}_i$, $\overrightarrow{y}_i = \overrightarrow{b}_i$, $\overrightarrow{z}_i = \overrightarrow{b}_i (i = 1, 2, \ldots, N)$..

3.2 Dynamic step size

The step size α is an important parameter in fine-tuning of improved solutions and can potentially affect the convergence rate of algorithm [37–39]. The performance of CS has been proved to be superior, compared with PSO and GA [1]. This relies on the introduction of Lévy flight process. However, a fast convergence of CS algorithm cannot be guaranteed, since the process is essentially a model of random walk. In the original CS, α is a fixed constant and $\alpha = 1$ is used in [18]. To enhance the convergence rate, the original CS is slightly modified with respect to the step size α in this paper. $\alpha(k)$ is decreased with increasing iteration number. This is done similar to the principle that the inertia weight is reduced with increasing iteration number in PSO [11]. In the initial generations, $\alpha(k)$ is large enough to enforce the cuckoos to search more various solutions, which guarantee the diversity of the new solution vectors. In the final generations, $\alpha(k)$ is decreased to result in a better fine-tuning of improved solution vectors. In this paper, $\alpha(k)$ is dynamically adjusted with the generation number given below.

$$\alpha(k) = \alpha_{\max} - \frac{(\alpha_{\max} - \alpha_{\min})}{k_{\max}} \times k \tag{12}$$

where k is the current iteration, $k_{\max}$ is the total number of iterations. $\alpha_{\min}$ and $\alpha_{\max}$ are the lower and upper bounds of the parameter α, respectively, and their values are taken as $\alpha_{\min} = 0.01$, $\alpha_{\max} = 0.5$ in a hit and trial method.

3.3 The hybrid CSPSO procedure

In the original CS, there is no information exchange between each cuckoo, and actually, the search process of each cuckoo is performed independently [40]. Here, we will combine the good search ability of CS and the global search advantage of PSO to enhance the population diversity and the convergence rate of the proposed hybrid algorithm. In this case, instead of employing a single pattern which is called Lévy flight to generate new solutions in CS, we use an integration of two different ways to generate the solutions in CSPSO. The first way is the classical pattern of Lévy flight in CS, and the second is the updating ways as Eqs. (2–3) in PSO. The detailed hybrid process is described below.

Each cuckoo performs Lévy flight to generate a new solution $\overrightarrow{x}_i(k+1)$ and follows the updating ways based on PSO to generate a new solution $\overrightarrow{y}_i(k+1)$. A new solution of the CSPSO is generated with the combination of $\overrightarrow{x}_i(k+1)$ and $\overrightarrow{y}_i(k+1)$, and the updating formula of the new solutions is proposed in Eq. (13).

$$\overrightarrow{z}_i(k+1) = d \times \overrightarrow{x}_i(k+1) + (1-d) \times \overrightarrow{y}_i(k+1) \quad (13)$$

where $\overrightarrow{z}_i(k+1)$ is the new solution of the CSPSO, d ($d \in [0, 1]$) is a random number.

The steps involved in the hybrid CSPSO can be shown in detail in Fig. 3. As is shown in Fig. 3, there are three improvements in the proposed hybrid algorithm. The first improvement is that initial individuals of the hybrid algorithm are generated by using orthogonal Lation squares, which is described in Subsection 3.1. The second improvement is that the step size of the CS is dynamically adjusted instead of a fixed value which is discussed in subsection 3.2. The third improvement is that the CS is hybridized with the PSO to form a new hybrid optimization algorithm using the hybrid strategy which is described in this section. The proposed CSPSO can be summarized as given in Fig. 3.

Fig. 3 Pseudo code of the proposed algorithm

Algorithm 3 A hybridization of cuckoo search and particle swarm optimization (CSPSO)

Initialize objective function $f(\vec{b})$, $\vec{b} = (b_1, \ldots, b_D)^T$ $\vec{x} = \vec{b}, \vec{y} = \vec{b}, \vec{z} = \vec{b}$

Initialize a population of N individuals $\vec{z}_i (i = 1, 2, \ldots, N)$ by using Eqs. (10-11)

For all $\vec{z}_i (i = 1, 2, \ldots, N)$ do

Evaluate the fitness F_i

$\vec{p}_i = \vec{z}_i$, $\vec{x}_i = \vec{z}_i$, $\vec{y}_i = \vec{z}_i$

$\vec{g} = \arg\min f(\vec{p}_i)$

End for

While ($k <$ Max Generation) and (the stop criterion is not met) do

For each individual $i \in 1 \rightarrow N$ do

Calculate the step size of CS by using (12)

$\vec{x}_i(k+1)$ =new solutions updated using (8-9)

$\vec{y}_i(k+1)$ =new solutions updated using (2-3)

Generate a new solution $\vec{z}_i(k+1) = d \times \vec{x}_i(k+1) + (1-d) \times \vec{y}_i(k+1)$

Evaluate its quality/fitness F_i

If ($f(\vec{z}_i(k+1)) < f(\vec{z}_i(k))$)

Replace i by the new solution

End if

End for

Choose a nest among N (say j) randomly

Abandon worse nests with a fraction p_a and rebuild the corresponding new nests

$\vec{x}_i(k+1) = \vec{z}_i(k+1)$, $\vec{y}_i(k+1) = \vec{z}_i(k+1)$

Update $\vec{p}_i(k+1)$ and $\vec{g}(k+1)$ using (4-5)

Keep the best solutions (or nests with quality solutions)

Rank the solutions and find the current best

End while

Post-process results and visualization

4 Experimental studies on benchmark functions

4.1 Benchmark functions

To evaluate the performance of the proposed CSPSO and conduct a further comparative study, a set of well-known test functions [41–44] are used as benchmark problems. The dimension of these test functions can be fixed or unfixed. The number of dimensions is set as 10 in this work for unfixed dimension functions. But for a function with fixed number of dimension, the dimension number has been preset by the literature and cannot be changed. Both unimodal and multimodal functions are selected to evaluate the robustness of the proposed CSPSO algorithm. Functions $F_1 - F_{14}$ are unimodal functions and $F_{15} - F_{20}$ are multimodal functions. The name (Function), the formula (Formula), the range of dimension (D), the searching range (Range), and the known optimal value (Optima) of these problems are listed in Table 1.

4.2 Algorithms used for comparison and parametric studies

To test the effectiveness of CSPSO and to conduct an exhaustive comparison, the proposed CSPSO and other six algorithms PSO [12], DE [13], BA [15], HS [16], CS [18], and ICS [37] are tested on the series of test functions above. The detailed computational data of all test functions for all these algorithms will be presented. The successful rate, the best fitness, the worst fitness, mean value, and standard deviation of all algorithms are given.

In all experiments, the values of the common parameters used in each algorithm such as the population size and the total iteration number are chosen to be the same. For all algorithms, the population size is set as $N = 20$, and the total number of iterations is set as $k_{\max} = 2000$. To reduce the random error of the simulation, all experiments on each test function is repeated 50 times. All computational experiences for the benchmark problems are implemented using Matlab2013b on a PC with a Intel core i5-4460 3.20 GHz processor and 8.0 GB memory.

The algorithms and other specific parameters settings are given below:

(1) CSPSO: $p_a = 0.25$, $\alpha_{\min} = 0.01$, $\alpha_{\max} = 0.5$;
(2) PSO [12]: $c_1 = c_2 = 2$, $w_{\min} = 0.4$, $w_{\max} = 0.9$;
(3) DE [13]: $F = 0.6$, $CR = 0.1$;
(4) BA [15]: $A = 0.25$, $r = 1$, $Q_{\min} = 0$, $Q_{\max} = 2$;
(5) HS [16]: $M = 30$, $HMCR = 0.95$, $PAR = 0.3$, $BW = 0.06$;
(6) CS [18]: $p_a = 0.25$, $\alpha = 1$;
(7) (7)ICS [37]: $p_{\min} = 0.005$, $p_{\max} = 0.5$, $\alpha_{\min} = 0.01$, $\alpha(\max) = 0.5$.

4.3 Simulation and comparison

We begin with a discussion on the success rate obtained for 20 benchmark problems. Table 2 reports the success rate by applying the seven algorithms to optimize the benchmark problems $F_1 - F_{20}$, respectively. The success of an algorithm means that this algorithm can result in a function value less than the pre-specified optimal value, i.e., 1.0E−08, for all problems with the number of iterations less than the pre-specified maximum number. The successful rate is calculated as the number of successful runs divided by the total number of runs. In this paper, on one function, 0 success rate means that in all iterations the algorithm cannot obtain a fitness value less than 1.0E−08.

From Table 2, it can be seen that CSPSO achieves best success rate on 16 functions, and on 15 of these 16 functions it achieves 100% success rate. Both CS and ICS achieve 100% success rate on 15 functions. DE achieves 100% success rate on 14 functions. On nine functions, PSO gets 100% success rate. On five functions, BA and HS all have 100% success rate. Only on function Griewank, DE has better performance than CSPSO. In summary, the CSPSO has a very good success rate in testing the majority of the benchmark problems.

Tables 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12 present the computational results obtained for 20 functions, respectively. The best results among the seven algorithms are shown in italics. “Best” represents the best fitness, “Worst” represents the worst fitness, “Mean” represents the mean value of the fitness, and “Std” represents the standard deviation of the fitness value.

Sphere function and Schwefel 2.22 function are two unimodal functions. They are mainly used to test the accuracy of optimization algorithms. Here, they are easily optimized by the seven algorithms. From the results of Table 3, it can be observed that CSPSO shows the highest search performance in terms of the solution qualities. In comparison with PSO, DE, BA, HS, CS, and ICS algorithms, the proposed CSPSO gets the smallest best, worst, mean, and standard deviation values.

From Table 4, on Eason function, CSPSO, PSO, BA, CS, and ICS algorithms show the same performances in terms of the solution results. Except for DE and HS, the other five algorithms all could approximate to global optimum. Step function has one minimum and it is a discontinuous function. The same performances are obtained by CSPSO, DE, CS, and ICS algorithms. Except for PSO, BA, and HS, other four algorithms all approximate to the global optimum for the Step function.

Table 5 shows good capability of the proposed CSPSO algorithm on Sum square function and Quadric function. For Sum square function, CSPSO performs the highest

Table 1 The benchmark test functions

Function	Formula	D	Range	Optima
Sphere	$F_1(x) = \sum_{i=1}^{D} x_i^2$	10	[−100,100]	0
Schwefel 2.22	$F_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	10	[−10,10]	0
Eason	$F_3(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$	2	[−100,100]	0
Step	$F_4(x) = \sum_{i=1}^{D} \left(\lfloor x_i + {}^{i}0.5 \rfloor\right)^2$	10	[−100,100]	0
Sum square	$F_5(x) = \sum_{i=1}^{D} i x_i^2$	10	[−100,100]	0
Quadric	$F_6(x) = \sum_{i=1}^{D} i x_i^4 + \text{random}(0,1)$	10	[−1.28,1.28]	0
Dixon-Price	$F_7(x) = (x_1-1)^2 + \sum_{i=2}^{D} i\left(2x_i^2 - x_{i-1}\right)^2$	10	[−10,10]	0
Schwefel 2.21	$F_8(x) = \max\{\lvert x_i \rvert, 1 \le i \le D\}$	10	[−100,100]	0
Zakharov	$F_9(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5 i x_i\right)^4$	10	[−5,10]	0
Matyas	$F_{10}(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48 x_1 x_2$	2	[−10,10]	0
Trid6	$F_{11}(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$	6	$[-D^2, D^2]$	−50
Powell	$F_{12}(x) = \sum_{i=1}^{n/k} \left[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4\right]$	24	[−4,5]	0
Beale	$F_{13}(x) = (1.5 - x_1 + x_1 x_2)^2 + \left(2.25 - x_1 + x_1 x_2^2\right) + \left(2.625 - x_1 + x_1 x_2^3\right)^2$	2	[−4.5,4.5]	0
Trid10	$F_{14}(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$	10	$[-D^2, D^2]$	−210
Rastrigin	$F_{15}(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	10	[−5.12,5.12]	0
Griewank	$F_{16}(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	0	[−600,600]	0
Weierstrass	$F_{17}(x) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k\max} \left[a^k \cos\left(2\pi b^k (x_i + 0.5)\right)\right] - D \sum_{k=0}^{k\max} \left[a^k \cos\left(2\pi b^{k*} 0.5\right)\right]\right.$	2	[−0.5,0.5]	0
Bohachevsky1	$F_{18}(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	2	[−100,100]	0
Bohachevsky2	$F_{19}(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\ {}^{*}0.4\cos(4\pi x_2) + 0.3$	2	[−100,100]	0
Bohachevsky3	$F_{20}(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$	2	[−100,100]	0

performance and ICS shows a moderate performance. Quartic function is padded with random noise. It is hard to be optimized because every evaluation for function depends on the random noise generator (uniform or gaussian). From the results of Table 5, on Quadric function, it can be observed that the performance of CSPSO and HS are better than other five algorithms. HS gets the best "Best" result, and CSPSO achieves the best "Worst" "Mean" and "Std" results. Overall, CSPSO is also competitive and effective for solving Quartic function.

For Dixon-Price function in Table 6, CSPSO finds the best "Best" result in comparison with other six algorithms and it

Table 2 The successful rate of all algorithms for test functions

Function	Threshold value	Successful rate (%)						
		PSO [12]	DE [13]	BA [15]	HS [16]	CS [18]	ICS [37]	CSPSO
Sphere	1.0E−08	0	100	0	0	100	100	100
Schwefel 2.22	1.0E−08	0	100	0	0	100	100	100
Eason	1.0E−08	100	100	100	100	100	100	100
Step	1.0E−08	50	100	18	98	100	100	100
Sum square	1.0E−08	0	100	0	0	100	100	100
Quadric	1.0E−08	0	0	0	0	0	0	0
Dixon-Price	1.0E−08	0	8	0	0	2	0	8
Schwefel 2.21	1.0E−08	100	100	0	0	100	100	100
Zakharov	1.0E−08	0	0	0	0	100	0	100
Matyas	1.0E−08	100	100	100	0	100	100	100
Trid6	1.0E−08	100	100	100	100	100	100	100
Powell	1.0E−08	0	0	0	0	0	0	0
Beale	1.0E−08	86	100	36	0	100	100	100
Trid10	1.0E−08	100	100	100	100	100	100	100
Rastrigin	1.0E−08	0	76	0	0	0	100	78
Weierstrass	1.0E−08	100	100	100	100	100	100	100
Griewank	1.0E−08	0	82	0	0	0	0	2
Bohachevsky1	1.0E−08	100	100	4	100	100	100	100
Bohachevsky2	1.0E−08	100	100	4	0	100	100	100
Bohachevsky3	1.0E−08	100	100	12	0	100	100	100

Table 3 Experimental results of Sphere function and Schwefel 2.22 function

Function Criteria	Sphere				Schwefel 2.22			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO [12]	5.14E−06	1.23E−02	1.50E−03	2.20E−03	4.50E−03	1.206E−01	2.87E−02	2.52E−02
DE [13]	1.00E−79	4.22E−76	4.04E−77	8.03E−77	7.80E−44	7.85E−42	9.61E−43	1.44E−42
BA [15]	3.27E−05	1.24E−04	8.11E−05	1.85E−05	1.39E−02	2.77E−02	2.33E−02	2.70E−03
HS [16]	5.90E−03	3.29E−01	1.09E−01	6.35E−02	3.09E−02	1.67E−01	7.78E−02	2.85E−02
CS [18]	1.04E−10	8.53E−09	1.13E−09	1.38E−09	1.70E−17	1.39E−15	3.68E−16	3.48E−16
ICS [37]	1.00E−48	2.56E−45	1.23E−46	3.64E−46	1.25E−27	3.87E−26	1.03E−26	8.45E−27
CSPSO	*2.07E−124*	*2.70E−103*	*5.30E−105*	*3.78E−105*	*7.04E−76*	*9.01E—69*	*2.40E−70*	*1.31E−69*

Italic emphasis indicate the best result

Table 4 Experimental results of Eason function and Step function

Function Criteria	Eason				Step			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO [12]	*−1.00E+00*	*−1.00E+00*	*−1.00E+00*	*0*	3.00E+00	6.40E−01	7.49E−01	7.05E−01
DE [13]	−1.00E+00	−9.98E−01	−1.00E+00	3.00E−04	*0*	*0*	*0*	*0*
BA [15]	*−1.00E+00*	*−1.00E+00*	*−1.00E+00*	*0*	4.00E+00	1.68E+00	1.10E+00	1.07E+00
HS [16]	−1.00E+00	−1.00–04	−5.29–01	4.93E−01	1.00E+00	2.00E−02	1.41E−01	1.41E−01
CS [18]	*−1.00E+00*	*−1.00E+00*	*−1.00E+00*	*0*	*0*	*0*	*0*	*0*
ICS [37]	*−1.00E+00*	*−1.00E+00*	*−1.00E+00*	*0*	*0*	*0*	*0*	*0*
CSPSO	*−1.00E+00*	*−1.00E+00*	*−1.00E+00*	*0*	*0*	*0*	*0*	*0*

Italic emphasis indicate the best result

Table 5 Experimental results of Sum square function and Quadric function

Function Criteria	Sum square				Quadric			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO [12]	1.00E−04	1.86E−01	2.02E−02	3.53E−02	6.70E−03	4.27E−02	2.48E−02	8.60E−03
DE [13]	4.69E−80	5.13E−75	2.08E−76	7.27E−76	2.50E−03	2.27E−02	1.23E−02	4.80E−03
BA [15]	2.32E−04	6.52E−04	4.02E−04	8.96E−05	1.64E−02	2.38E−01	1.06E−01	4.93E−02
HS [16]	2.10E−03	3.10E + 00	6.41E−01	5.94E−01	*1.00E−04*	2.09E−02	3.90E−03	4.30E−03
CS [18]	1.63E−35	1.38E−31	3.60E−33	1.95E−32	4.00E−04	7.50E−03	2.90E−03	1.40E−03
ICS [37]	1.20E−47	3.50E−45	3.43E−46	5.78E−46	9.00E−04	5.60E−03	2.80E−03	1.10E−03
CSPSO	*4.14E−125*	*4.90E−104*	*1.11E−105*	*6.97E−105*	3.00E−04	*4.00E−03*	*1.00E−03*	*8.00E−04*

Italic emphasis indicate the best result

Table 6 Experimental results of Dixon-Price function and Schwefel2.21 function

Function Criteria	Dixon-Price				Schwefel 2.21			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO [12]	0.80E−03	7.68E−01	6.15E−01	1.81E−01	0	4.51E−93	2.13E−94	7.77E−94
DE [13]	5.10E−20	*6.67E−01*	*7.47E−02*	1.63E−01	4.49E−89	1.25E−56	2.50E−58	1.77E−57
BA [15]	2.50E−03	6.68E−01	6.54E−01	*9.40E−02*	1.70E−08	4.05E−06	7.98E−07	8.87E−07
HS [16]	2.11E−02	1.08E + 00	4.63E−01	3.08E−01	8.00E−04	1.34E−01	3.21E−02	3.37E−02
CS [18]	4.85E−14	*6.67E−01*	3.45E−01	3.33E−01	1.32E−55	2.39E−47	1.01E−48	3.93E−48
ICS [37]	1.05E−11	*6.67E−01*	2.48E−01	3.17E−01	2.57E−76	9.49E−67	2.82E−68	1.39E−67
CSPSO	*4.41E−18*	*6.67E−01*	6.13E−01	1.83E−01	*3.47E−247*	*1.69E−236*	*4.70E−238*	*0*

Italic emphasis indicate the best result

Table 7 Experimental results of Zakharov function and Matyas function

Function Criteria	Zakharov				Matyas			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO [12]	1.94E−05	3.37E+01	6.74E−01	4.76E+00	2.37E−177	3.68E−163	1.50E−164	0
DE [13]	5.70E−05	1.43E−02	2.00E−03	2.70E−03	7.79E−93	1.91E−78	4.80E−80	2.71E−79
BA [15]	7.43E-05	2.24E−04	1.41E−04	3.87E−05	3.10E−11	5.56E−09	8.97E−10	1.01E−09
HS [16]	2.12E−02	4.39E+00	7.82E−01	8.41E−01	4.56E−07	9.10E−03	1.50E−03	1.90E−03
CS [18]	1.60E−22	6.29E−20	6.27E−21	1.19E−20	1.98E−90	1.10E−70	3.30E−72	1.63E−71
ICS [37]	1.00E−04	4.76E−02	7.00E−03	9.30E−03	3.33E−48	5.11E−41	1.31E−42	7.35E−42
CSPSO	*2.30E−35*	*3.09E−21*	*7.60E−23*	*4.44E−22*	*1.13E−304*	*5.83E−287*	*1.42E−288*	*0*

Italic emphasis indicate the best result

Table 8 Experimental results of Trid6 function and Powell function

Function Criteria	Trid6				Powell			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO [12]	*−5.00E+01*	*−5.00E+01*	*−5.00E+01*	*0*	2.28E−01	1.56E+02	5.21E+00	2.19E+01
DE [13]	*−5.00E+01*	*−5.00E+01*	*−5.00E+01*	*0*	7.99E−02	8.31E−01	2.66E−01	1.61E−01
BA [15]	*−5.00E+01*	*−5.00E+01*	*−5.00E+01*	*0*	1.88E−02	2.12E−01	9.93E−02	5.07E−02
HS [16]	−4.99E+01	−4.89E+01	−4.96E+01	2.67E−01	8.00E−04	9.04E−01	1.72E−01	1.73E−01
CS [18]	*−5.00E+01*	*−5.00E+01*	*−5.00E+01*	*0*	*3.00E−04*	1.95E−02	*4.20E−03*	4.00E−03
ICS [37]	*−5.00E+01*	*−5.00E+01*	*−5.00E+01*	*0*	2.30E−03	2.43E−02	9.10E−03	4.50E−03
CSPSO	*−5.00E+01*	*−5.00E+01*	*−5.00E+01*	*0*	1.60E−03	*1.20E−02*	6.90E−03	*2.30E−03*

Italic emphasis indicate the best result

Table 9 Experimental results of Beale function and Trid10 function

Function Criteria	Beale				Trid10			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO [12]	*0*	7.62E−01	1.07E−01	2.67E−01	*−2.10E+02*	−2.09E+02	*−2.10E+02*	1.51E−01
DE [13]	*0*	1.41E−09	2.80E−11	1.99E−10	*−2.10E+02*	−2.05E+02	−2.09E+02	1.20E+00
BA [15]	3.40E−10	7.62E−01	1.22E−01	2.82E−01	*−2.10E+02*	−1.98E+02	−2.09E+02	2.55E+00
HS [16]	7.07E−06	7.93E−02	1.23E−02	1.48E−02	−2.09E+02	−8.57E+01	−1.76E+02	3.61E+01
CS [18]	*0*	*0*	*0*	*0*	*−2.10E+02*	*−2.10E+02*	*−2.10E+02*	*3.67E−13*
ICS [37]	*0*	2.92E−30	1.36E−31	4.73E−31	*−2.10E+02*	−2.09E+02	*−2.10E+02*	1.37E−05
CSPSO	*0*	*0*	*0*	*0*	*−2.10E+02*	−2.09E+02	−2.09E+02	1.90E−03

Italic emphasis indicate the best result

Table 10 Experimental results of Rastrigin function and Weierstrass function

Function Criteria	Rastrigin				Weierstrass			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO [12]	1.67E−02	9.61E+00	6.70E+00	2.08E+00	−8.83E+01	−6.70E+01	−8.16E+01	4.79E+00
DE [13]	*0*	1.10E+00	2.79E−01	5.33E−01	−1.20E+02	*−1.20E+02*	*−1.20E+02*	*0*
BA [15]	2.01E+00	2.09E+01	9.29E+00	3.81E+00	−5.38E+01	−2.48E+01	−4.00E+01	7.04E+00
HS [16]	8.20E−03	2.68E−01	7.14E−02	5.55E−02	*−1.19E+02*	−1.19E+02	−1.17+02	4.27E−01
CS [18]	1.73E−02	4.85E+00	1.87E+00	1.15E+00	*−1.19E+02*	−1.19E+02	−1.19E+02	1.00E−04
ICS [37]	*0*	*0*	*0*	*0*	−9.0E+01	−9.0E+01	−9.0E+01	*0*
CSPSO	*0*	9.95E−01	5.14E−02	2.09E−01	*−1.19E+02*	−1.16E+02	−1.19E+02	6.18E−01

Italic emphasis indicate the best result

achieves the same best "Worst" result as DE, CS, and ICS. Regarding the "Mean" and "Std" results, ICS provides the best "Mean" result and "BA" provides the best "Std" result. In Table 6 and Table 7, it is obvious that CSPSO achieves the best results for Schwefel 2.21 function, Zakharov function, and Matyas function. Compared with other six algorithms, CSPSO performs the best and HS shows the worst performance.

The results of Trid 6 function and Powell function are presented in Table 8. For Trid 6 function, CSPSO shows the same best performance as PSO, DE, BA, CS, and ICS, but the performance of HS gets worse. On Powell function, CS achieves the best "Best" and "Mean" results. CSPSO gets the best "Worst" and "Std" results. Overall, CSPSO is very competitive and effective for solving Powell function.

From the results in Table 9, it can be clearly seen that CSPSO and CS show the same performance and they are much better than other five algorithms on Beale function. They both could approximate to the global optimum successfully. On Trid 10 function, all

Table 11 Experimental results of Griewank function and Bohachevsky1 function

Function Criteria	Griewank				Bohachevsky1			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO [12]	3.82E−02	3.91E−01	1.54E−01	8.49E−02	*0*	*0*	*0*	*0*
DE [13]	*0*	1.47E−02	1.70E−03	3.80E−03	*0*	*0*	*0*	*0*
BA [15]	3.32E−06	*1.57E−05*	*9.69E−06*	*2.62E−06*	5.06E−09	4.70E−01	1.77E−01	2.12E−01
HS [16]	3.51E−02	5.23E−01	2.25E−01	8.47E−02	−8.00E+00	−7.83+00	−7.92E+00	4.60E−02
CS [18]	3.20E−03	6.37E−02	2.86E−02	1.56E−02	*0*	*0*	*0*	*0*
ICS [37]	1.85E−07	3.30E−02	1.26E−02	7.70E−03	*0*	*0*	*0*	*0*
CSPSO	1.10E−14	6.58E−02	2.00E−02	1.64E−02	*0*	*0*	*0*	*0*

Italic emphasis indicate the best result

Table 12 Experimental results of Bohachevsky2 function and Bohachevsky3 function

Function Criteria	Bohachevsky2				Bohachevsky3			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
PSO [12]	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
DE [13]	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
BA [15]	3.47E−09	2.18E−01	9.18E−02	1.06E−01	3.91E−09	2.26E−01	3.88E−02	8.46E−02
HS [16]	5.00E−04	5.85E−01	2.01E−01	1.09E−01	2.30E−03	3.64E−01	1.11E−01	1.07E−01
CS [18]	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
ICS [37]	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
CSPSO	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*

Italic emphasis indicate the best result

algorithms get the best "Best" result, CS achieves the best "Worst" and "Std" results, and PSO gets the best "Mean" result as well as CS and ICS. In general, CS performs the best and CSPSO shows a moderate performance on Trid 10 function.

Table 10 gives the results of all algorithms for Rastrigin function and Weierstrass function. Rastrigin function is a complex multimodal function and based on the above-mentioned Sphere function. It uses cosine function to consistently produce many local optimal points. So it could easily fall into the local optimum when the algorithm finds out the global optimum. In Table 10, for Rastrigin function, it is remarkable that ICS performs better than others and it approximates to the global optimum. The performance of CSPSO is also promising but slightly worse than ICS. For Weierstrass function, DE gets the best "Best", "Worst", "Mean", and "Std" results and its performance is the best. CSPSO shows a moderate performance.

The results of Griewank function and Bohachevsky1 function are presented in Table 11, and the results of Bohachevsky2 function and Bohachevsky3 function are presented in Table 12. For Griewank function, DE achieves the best "Best" results, BA gets the best "Worst" "Mean" and "Std" results. In comparison with other six algorithms, CSPSO shows a moderate performance. For Bohachevsky1 function, Bohachevsky2 function, and Bohachevsky3 function, CSPSO, PSO, DE, CS, and ICS all could approximate to global optimum and they perform much better than BA and HS.

To give a visualized and detailed comparison, Figs. 4, 5, 6, 7, 8, 9, 10, 11 gives the convergence curves of the proposed CSPSO algorithm and other six algorithms PSO [12], DE [13], BA [15], HS [16], CS [18], and ICS [37]. This paper only presents eight representative convergence curves on eight functions (Sphere, Schwefel 2.22, Sum square, Schwefel 2.21, Zakharov, Matyas, Powell, and Bohachevsky3). The plot depicts convergence trends of the considered seven algorithms in a random run. To clearly compare the performance of each algorithm, the log-based 10 of the obtained results is made in plotting figures. In a figure, the labels of X-axis and Y-axis represent the algorithm iteration numbers and the function value, respectively.

From Figs. 4, 5, 6, 7, 8, 9, 10, 11, we could observe that CSPSO converges to global optimum quickly. CSPSO has much faster convergence compared with other six algorithms on Sphere function, Schwefel 2.22 function, Sum square function, Schwefel 2.21 function, Zakharov function, Matyas function, and Bohachevsky3 function. From Fig. 10, for Powell function, CSPSO shows higher convergence rate than CS and they both converge faster than other five algorithms. What is more, the optimum obtained by CSPSO is better than that of other six algorithms. Through above analysis and discussion, it can be illustrated that the proposed CSPSO is efficient for solving these benchmark problems, while DE and ICS show a moderate performance in the comparison.

5 Application studies on two engineering cases

Several cases taken from the optimization literatures have been previously solved by using a variety of other methods [1, 45], which is useful to show the performance of the algorithms. In this section, to validate the accuracy and effectiveness of the proposed CSPSO, we use two typical design cases [1] as applications.

5.1 Constraint handing technique

For a constrained problem, it is an important aspect to choose a suitable method which is employed to handle constraints. The method should help the algorithm search in the feasible regions and be able to arrive at the bounds of the search space. An unfeasible solution may become a feasible solution in the population. A common method is the use of penalty function [46] (see Eqs. (14–15)) and this method is chosen for the proposed CSPSO in this paper. The idea of the penalty method is to transform

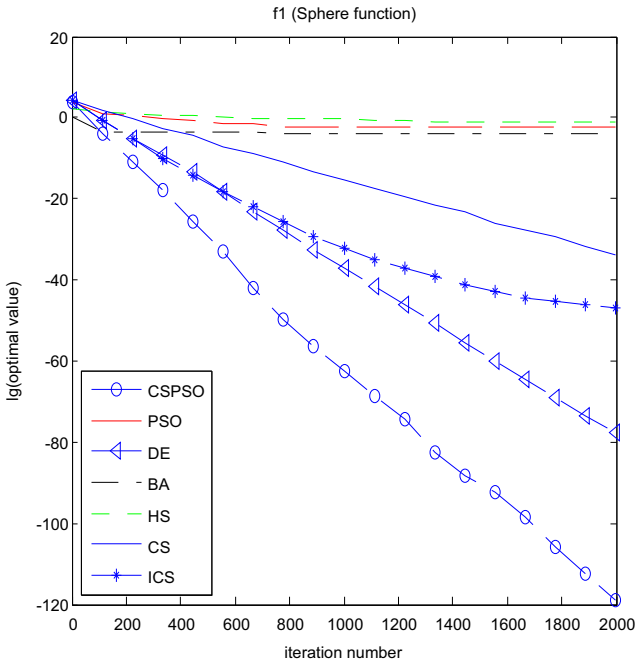

Fig. 4 The convergence curve of Sphere function>

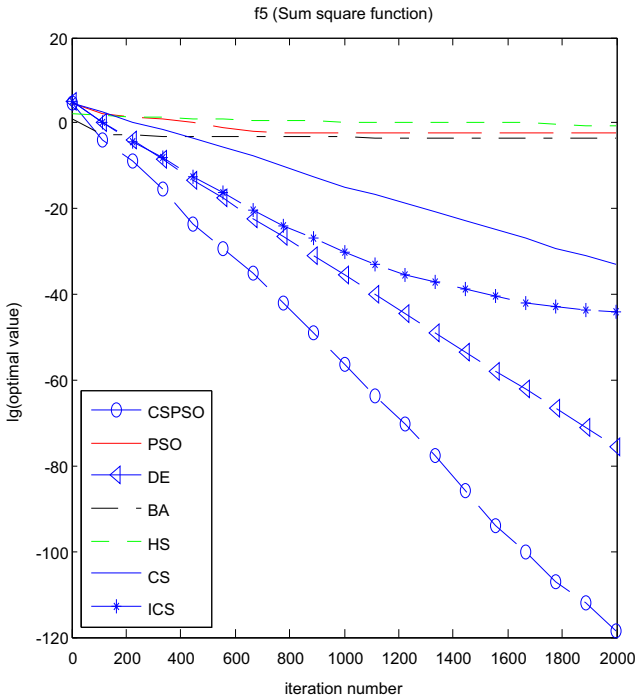

Fig. 6 The convergence curve of Sum square function

the constrained problem into an unconstrained one by introducing a penalty factor.

The constrained violation degree is stated as:

$$G\left(\vec{x}\right) = \sum_{i=1}^{n_g} \max\left\{0, g_i\left(\vec{x}\right)\right\} \tag{14}$$

The penalty function of the constrained problem (1) can be defined as follow:

$$T\left(\vec{x}, \sigma\right) = f\left(\vec{x}\right) + \sigma G\left(\vec{x}\right) \tag{15}$$

where $f\left(\vec{x}\right)$ is the objective function, σ is the penalty factor [48].

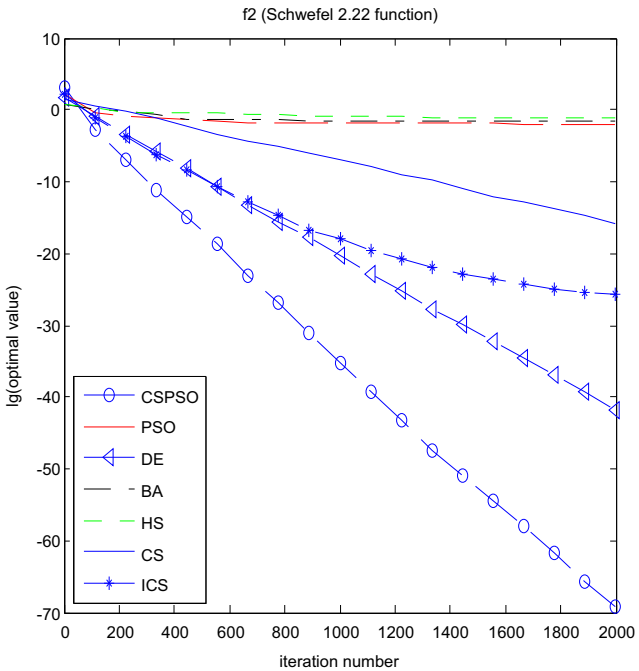

Fig. 5 The convergence curve of Schwefel 2.22 function

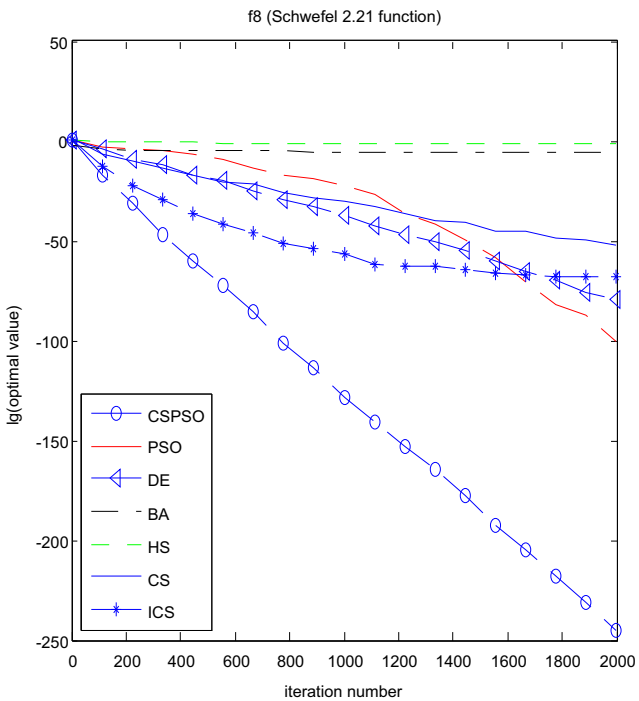

Fig. 7 Pseudo code of PSO algorithm

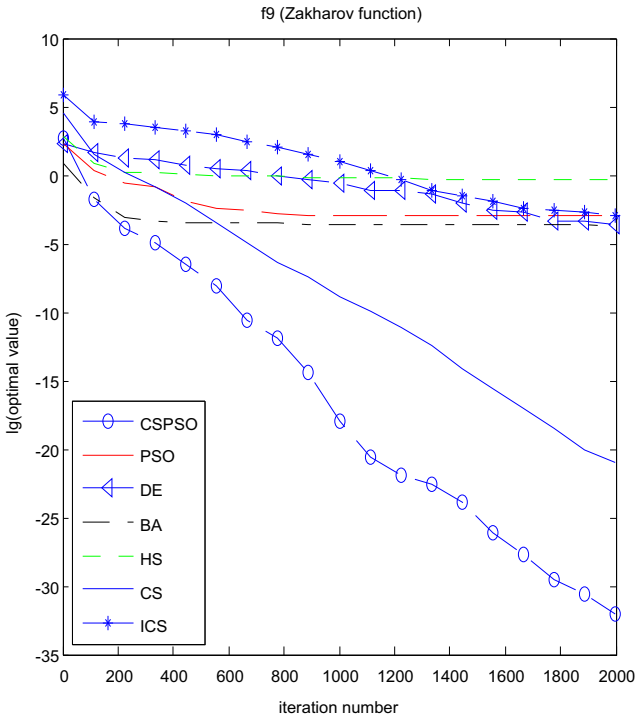

Fig. 8 The convergence curve of Zakharov function

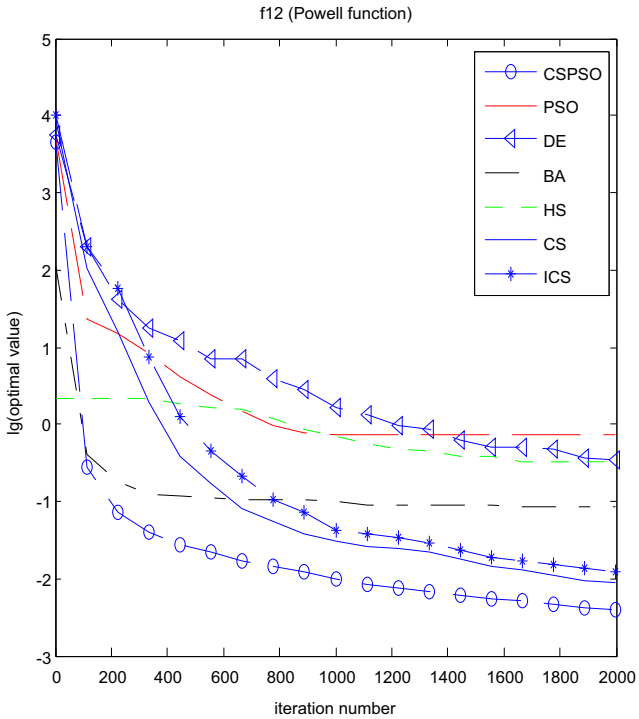

Fig. 10 The convergence curve of Powell function

5.2 Tension/compression spring design problem

The tension/compression springs are often used in the field of engineering. A standard spring design problem has three design variables: the wire diameter w ($=x_1$), the mean coil diameter d($=x_2$), and the number of coils l ($=x_3$). The objective of this problem is to minimize the weight of spring. It also has four nonlinear inequality constraints.

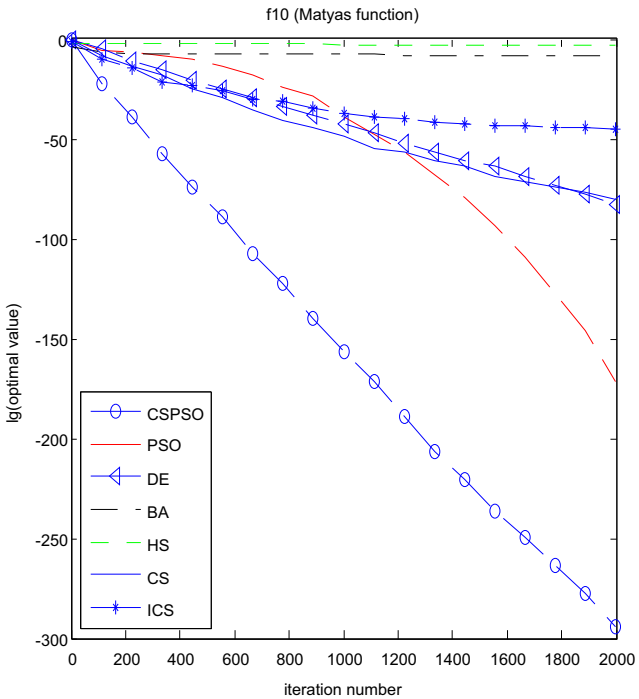

Fig. 9 The convergence curve of Matyas function

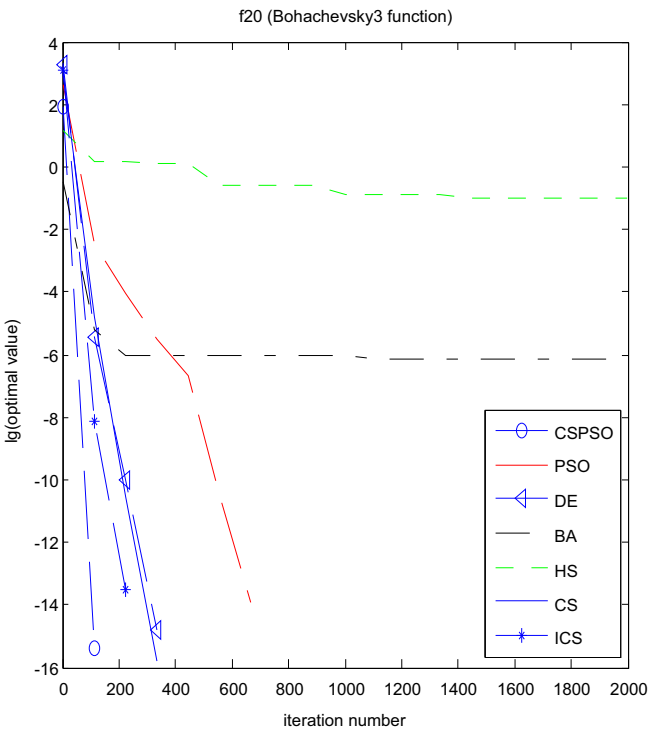

Fig. 11 The convergence curve of Bohachevsky3 function

Table 13 Optimal results for minimization of the weight of spring

Methods	Optimal design variables (x)			Cost
	w	d	l	
PSO [12]	0.052560919	0.378056872	10.139443232	0.012678906
DE [13]	0.051997147	0.364174266	10.864896900	0.012667024
BA [15]	0.050000000	0.311156617	14.914377401	0.012699285
HS [16]	0.050000000	0.315233688	14.322644706	0.012863618
DA [17]	0.054277415	0.422242014	8.3756900062	0.012906745
CS [18]	0.057840785	0.523525808	5.5999560338	0.012665799
HCSGA [25, 26]	0.283000000	1.400032000	6.0000000000	2.213301000
ICS [37, 38]	0.053945073	0.413394932	8.6154155079	0.012665743
CSPSO	0.054719001	0.434025369	7.8714408840	0.012665256

Italic emphasis indicate the best result

The mathematical model of this problem can be written compactly as:

$$\min \quad f\left(\overrightarrow{x}\right) = (x_3+2)x_2x_1^2$$

$$s.t. \quad \begin{cases} g_1\left(\overrightarrow{x}\right) = 1-\dfrac{x_2^3x_3}{71785x_1^4} \le 0, \\ g_2\left(\overrightarrow{x}\right) = \dfrac{4x_2^2-x_1x_2}{12566\left(x_2x_1^3-x_1^4\right)} + \dfrac{1}{5108x_1^2} - 1 \le 0, \\ g_3\left(\overrightarrow{x}\right) = 1-\dfrac{140.45x_1}{x_2^2x_3} \le 0, \\ g_4\left(\overrightarrow{x}\right) = \dfrac{x_2+x_1}{1.5} - 1 \le 0, \end{cases}$$

where

$$0.05 \le x_1 \le 2.0,\ 0.25 \le x_2 \le 1.3,\ 2.0 \le x_3 \le 15.0.$$

This problem has been studied previously by several researchers from earlier studies. Belegundu [47] used eight different mathematical programming methods to solve this problem. Coello [48] also solved this problem by using GA algorithm. Additionally, Eskandar [49] solved this problem using a water cycle algorithm. In this work, the proposed CSPSO algorithm and other six referred algorithms: PSO [12], DE [13], BA [15], HS [16], DA [17], CS [18], HCSGA [25, 26], and ICS [37, 38] are all employed to solve the tension/compression spring design problem. The parameters of these algorithms for this problem are set as the same as subsection 4.2. The optimal solution is obtained at $\overrightarrow{x}^* = (0.054719001, 0.434025369, 7.8714408840)$ with corresponding function value equal to $f^*\left(\overrightarrow{x}\right) = 0.012665256$. The best result obtained by the proposed CSPSO is compared with that gained by other eight algorithms and the comparisons are presented in Table 13. One important thing to note about this table is that the results of HCSGA are from [25, 26].

As it can be seen from Table 13, the searching quality of CSPSO is better than all other six algorithms, that is, the cost is the lowest.

5.3 Welded beam design optimization problem

The so-called welded beam design is another practical problem that has been often used as a benchmark for testing the performance of different optimization methods. The objective is to find the minimum of the overall fabrication cost which consists of the set-up, weld labor, and material costs, under the

Table 14 Optimal results for welded beam design

Methods	Optimal design variables (x)				Cost
	w	l	d	h	
PSO [12]	0.205730228	3.470480974	9.036610984	0.205730228	1.7248544245
DE [13]	0.321076220	2.471791858	7.233530741	0.321076575	2.1219959265
BA [15]	0.147793791	5.411485988	9.038818192	0.205769634	1.8675292043
HS [16]	0.363918170	2.282509819	6.692758675	0.375095249	2.3004829282
DA [17]	0.191845186	3.797668365	9.037082880	0.205727870	1.7463220380
CS [18]	0.205730228	3.470480979	9.036610990	0.205730228	1.7248544246
HCSGA [25, 26]	0.205752962	3.470488920	9.036624340	0.205729620	1.7248522000
ICS [37, 38]	0.205496531	3.475776552	9.036073079	0.205756271	1.7253114787
CSPSO	0.219211677	3.354013957	8.634284996	0.225366761	1.7248544245

Italic emphasis indicate the best result

appropriate constraints of shear stress τ, bending stress in the beam σ, buckling load on the bar P, and maximum end deflection δ. The problem has four design variables: the width $w(=x_1)$, the length of the welded area $l(=x_2)$, the depth of the main beam $d(=x_3)$, and the thickness of the main beam $h(=x_4)$.

The mathematical formulation of this problem can be written as:

$$\min \quad f\left(\vec{x}\right) = 1.1047x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

$$s.t. \quad \begin{cases} g_1\left(\vec{x}\right) = x_1 - x_4 \le 0, \\ g_2\left(\vec{x}\right) = \delta\left(\vec{x}\right) - 0.25 \le 0, \\ g_3\left(\vec{x}\right) = \tau\left(\vec{x}\right) - 13600 \le 0, \\ g_4\left(\vec{x}\right) = \sigma\left(\vec{x}\right) - 30000 \le 0, \\ g_5\left(\vec{x}\right) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5.0 \le 0, \\ g_6\left(\vec{x}\right) = 0.0125 - x_1 \le 0, \\ g_7\left(\vec{x}\right) = 6000 - P\left(\vec{x}\right) \le 0, \end{cases}$$

where

$$\begin{cases} \sigma\left(\vec{x}\right) = \dfrac{5040000}{x_3^2x_4}, \\ Q = 6000\left(14 + \dfrac{x_2}{2}\right), \\ D = \dfrac{1}{2}\sqrt{x_2^2 + (x_1 + x_3)^2}, \\ J = \sqrt{2}x_1x_2\left[\dfrac{x_2^2}{6} + \dfrac{(x_1 + x_3)^2}{2}\right], \\ \delta\left(\vec{x}\right) = \dfrac{65856}{30000x_3^3x_4}, \\ \beta = \dfrac{QD}{J}, \\ \alpha = \dfrac{6000}{\sqrt{2}x_1x_2}, \\ \tau\left(\vec{x}\right) = \sqrt{a^2 + \dfrac{\alpha\beta x_2}{D} + \beta^2}, \\ P\left(\vec{x}\right) = 0.61423 \times 10^6 \dfrac{x_3x_4^3}{6}\left(1 - \dfrac{x_3\sqrt{30/48}}{28}\right), \\ 0.1 \le x_2, x_3 \le 10, 0.1 \le x_1, x_4 \le 2.0. \end{cases}$$

The proposed CSPSO algorithm is applied to the welded beam design optimization problem and the optimal result is compared with that obtained by earlier algorithms: PSO [12], DE [13], BA [15], HS [16], DA [17], CS [18], HCSGA [25, 26], and ICS [37, 38]. The parameters set of these algorithms are the same as subsection 4.2. Their comparison results are listed in Table 14, where the results of HCSGA are from [25, 26]. Using the proposed CSPSO, we have the following optimal result $\vec{x}^* = (0.219211677, 3.354013957, 8.634284996, 0.225366761)$ with corresponding function value equal to $f^*\left(\vec{x}\right) = 1.7248544245$.

From Table 14, HCSGA achieves the best result, and the proposed CSPSO and PSO provide the better results for the welded beam design optimization problem. Additionally, except for HCSGA, the better function value 1.7248544245 obtained by the proposed CSPSO is the same as the result by PSO.

6 Discussion and conclusion

In the present study, we have formulated a hybrid algorithm called CSPSO to solve continuous optimization problems. From the formulation of CSPSO to its implementation and comparison, we can see that it is a promising algorithm. The proposed CSPSO algorithm is potentially more promising than PSO [12], DE [13], BA [15], HS [16], DA [17], CS [18], HCSGA [25, 26], and ICS [37, 38] for most of the test problems. The primary reason is that CSPSO takes the search advantage of PSO and combines it to CS. From the framework of the CSPSO, it can be observed that the population individuals in CSPSO evolve with two different mechanisms and then exchange their information with each other. This can be viewed as a kind of co-evolutionary to some extent. According to the interactive iterations of the CS and PSO, the diversity of the optimal solutions and the convergence rate of the CSPSO are both enhanced. This superior performance of CSPSO is guaranteed by the co-evolutionary mechanism.

Moreover, the population of the proposed CSPSO is initialized by using the principle of orthogonal Lation squares and a dynamic step size is employed in CS instead of the original fixed constant. The effectiveness of these improvements is checked for several different performance criteria, such as success rate, best solution, worst solution, and mean solution and the original deviation, convergence rate, etc.

Unlike the problem-dependent algorithms which are on the basis of specific assumptions, such as the gradient information of the optimization objective, the convexity of constraint regions, and so on [50], CSPSO has solved different types of problem directly. The results on 20 benchmark test functions and 2 typical engineering design cases validate the effectiveness of the proposed CSPSO. Moreover, the extensive conducted comparative study shows that CSPSO outperforms the other algorithms in the literature for most cases. However, the present algorithm is only suitable for single objective optimization problems. Application of the proposed algorithm to multi-objective optimization problems is the future work.

Acknowledgments The authors would like to thank the Associate Editor and the reviewers for their detailed comments and valuable suggestions which considerably improved the presentation of the paper. The work is supported by the Natural Science Foundation of Hubei Province, China (#2015CFB586 and #2016CFB502), and the Fundamental Research Funds for the Central Universities (#163111005).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Yang XS, Deb S (2010) Engineering optimisation by cuckoo search. International Journal of Mathematical Modelling and Numerical Optimisation 1(4):330–343
2. Sun DI, Ashley B, Brewer B et al (1984) Optimal power flow by Newton approach. IEEE Transactions on Power Apparatus and Systems 103(10):2864–2880
3. Dommel HW, Tinney WF (1968) Optimal power flow solutions. IEEE Transactions on Power Apparatus and Systems 87(10):1866–1876
4. Capitanescu F, Wehenkel L (2013) Experiments with the interior-point method for solving large scale optimal power flow problems. Electric Power Systems Research, vol 95:276–283
5. Diez M, Peri D (2010) Robust optimization for ship conceptual design. Ocean Eng 37(11–12):966–977
6. Sekhar P, Mohanty S (2016) An enhanced cuckoo search algorithm based contingency constrained economic load dispatch for security enhancement. Int J Electr Power Energy Syst 75:303–310
7. Li X, Yin M (2013) A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem. Int J Prod Res 51(16):4732–4754
8. Nagano MS, Moccellin JV (2002) A high quality solution constructive heuristic for flow shop sequencing. J Oper Res Soc 53(12):1374–1379
9. Mitchell M (1998) An introduction to genetic algorithms. MIT press, London
10. Tang O (2004) Simulated annealing in lot sizing problems. Int J Prod Econ 88(2):173–181
11. Bratton D, Kennedy J (2007) Defining a standard for particle swarm optimization. In: Proceedings of the 2007 I.E. Swarm Intelligence Symposium, pp 120–127
12. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the IEEE International joint conference on neural networks, pp 1942–1948
13. 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
14. Dorigo M, Di Caro G (1999) The ant colony optimization meta-heuristic. In: New ideas in optimization, pp 11–32
15. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. SIMULATION 76(2):60–68
16. Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization, studies in computational intelligence, pp 65–74
17. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput & Applic 27(4):1053–1073
18. Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: World Congress on Nature and Biologically Inspired Computing, pp 210–214
19. Nearchou AC (2004) A novel metaheuristic approach for the flow shop scheduling problem. Eng Appl Artif Intell 17(3):289–300
20. Alikhani MG, Javadian N, Tavakkoli-Moghaddam R (2009) A novel hybrid approach combining electromagnetism-like method with Solis and wets local search for continuous optimization problems. J Glob Optim 44(2):227–234
21. Costa L, Santo I, Fernandes E (2012) A hybrid genetic pattern search augmented Lagrangian method for constrained global optimization. Appl Math Comput 218(18):9415–9426
22. Yildiz AR (2009) A novel hybrid immune algorithm for optimization of machining parameters in milling operations. Robot Comput Integr Manuf 25(2):261–270
23. De Melo VCV, Carosio GLC (2013) Investigating multi-view differential evolution for solving constrained engineering design problems. Expert Syst Appl 40(9):3370–3377
24. Su Y, Chi R (2017) Multi-objective particle swarm-differential evolution algorithm. Neural Comput & Applic 28(2):407–418
25. Kanagaraj G, Ponnambalam SG, Jawahar N et al (2013) An effective hybrid cuckoo search and genetic algorithm for constrained engineering design optimization. Eng Optim 46(10):1331–1351
26. Kanagaraj G, Ponnambalam SG, Gandomi AH (2016) Hybridizing cuckoo search with bio-inspired algorithms for constrained optimization problems. International Conference on Swarm, Evolutionary, and Memetic Computing, pp260–273
27. Huang J, Gao L, Li X (2015) An effective teaching-learning-based cuckoo search algorithm for parameter optimization problems in structure designing and machining processes. Appl Soft Comput 36:349–356
28. Mohamad AB, Zain AM, Bazin NEN (2014) Cuckoo search algorithm for optimization problems—a literature review and its applications. Appl Artif Intell 28(5):419–448
29. Mohapatra P, Chakravarty S, Dash PK (2015) An improved cuckoo search based extreme learning machine for medical data classification. Swarm and Evolutionary Computation 24:25–49
30. Ouaarab A, Ahiod B, Yang XS (2014) Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput & Applic 24(7–8):1659–1669
31. Hu X, Eberhart R (2002) Multiobjective optimization using dynamic neighborhood particle swarm optimization. In: Congress on Evolutionary Computation, pp1677–1681
32. Shokrian M, High KA (2014) Application of a multi objective multi-leader particle swarm optimization algorithm on NLP and MINLP problems. Comput Chem Eng 60:57–75
33. Shlesinger MF, Zaslavsky GM, Frisch U (1995) Lévy flights and related topics in physics. Lecture Notes in Physics, Berlin
34. Brown CT, Liebovitch LS, Glendon R (2007) Lévy flights in dobe ju/'hoansi foraging patterns. Hum Ecol 35(1):129–138
35. Pavlyukevich I (2007) Lévy flights, non-local search and simulated annealing. J Comput Phys 226(1):1830–1844
36. Chen K, Zhang Y, Chen G et al (2016) Further results on mutually nearly orthogonal Latin squares. Acta Mathematicae Applicatae Sinica, English Series 32(1):209–220
37. Valian E, Tavakoli S, Mohanna S et al (2013) Improved cuckoo search for reliability optimization problems. Comput Ind Eng 64(1):459–468
38. Valian E, Mohanna S, Tavakoli S (2011) Improved cuckoo search algorithm for feed-forward neural network training. International Journal of Artificial Intelligence & Applications 2(3):36–43
39. Bulatović RR, Bošković G, Savković MM et al (2014) Improved cuckoo search (ICS) algorithm for constrained optimization problems. Latin American Journal of Solids and Structures 11(8):1349–1362
40. Walton S, Hassan O, Morgan K et al (2011) Modified cuckoo search: a new gradient free optimisation algorithm. Chaos, Solitons Fractals 44(9):710–718
41. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(1):108–132
42. Hedar AR, Fukushima M (2006) Tabu search directed by direct search methods for nonlinear global optimization. Eur J Oper Res 170(2):329–349
43. Wang L, Zou F, Hei X et al (2014) A hybridization of teaching-learning-based optimization and differential evolution for chaotic

time series prediction. Neural Computing and Application 25(6): 1407–1422
44. Suganthan PN, Hansen N, Liang JJ, et al (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. Technical Report, Nanyang Technological University, Singapore and KanGAL Report Number 2005005
45. Cagnina LC, Esquivel SC, Coello CAC (2008) Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica 32(3):319–326
46. Bazaraa MS, Sherali HD, Shetty CM (1979) Nonlinear programming, theory and algorithm. Academic Press, New York
47. Belegundu AD (1985) A study of mathematical programming methods for structural optimization, PhD thesis, Department of Civil and Environmental Engineering, University of Iowa, Iowa
48. Coello CAC, Montes EM (2002) Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inform 16(3):193–203
49. Eskandar H, Sadollah A, Bahreininejad A et al (2012) Water cycle algorithm–a novel metaheuristic optimization method for solving constrained engineering optimization problems. Computers & Structures, vol 110-111:151–166
50. Ma W, Wang M, Zhu X (2014) Improved particle swarm optimization based approach for bilevel programming problem—an application on supply chain model. Int J Mach Learn Cybern 5(2):281–292