Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 2734362, 20 pages
https://doi.org/10.1155/2017/2734362

Research Article

Chicken Swarm Optimization Based on Elite Opposition-Based Learning

Chiwen Qu,[1] Shi'an Zhao,[2] Yanming Fu,[3] and Wei He[1]

[1] *School of Information Engineering, Baise University, Baise 533000, China*
[2] *School of Mathematics and Statistics, Baise University, Baise 533000, China*
[3] *Computer and Electronic Information College, Guangxi University, Nanning 530004, China*

Correspondence should be addressed to Chiwen Qu; quchiwen@163.com

Received 14 October 2016; Revised 13 February 2017; Accepted 1 March 2017; Published 26 March 2017

Academic Editor: George S. Dulikravich

Chicken swarm optimization is a new intelligent bionic algorithm, simulating the chicken swarm searching for food in nature. Basic algorithm is likely to fall into a local optimum and has a slow convergence rate. Aiming at these deficiencies, an improved chicken swarm optimization algorithm based on elite opposition-based learning is proposed. In cock swarm, random search based on adaptive t distribution is adopted to replace that based on Gaussian distribution so as to balance the global exploitation ability and local development ability of the algorithm. In hen swarm, elite opposition-based learning is introduced to promote the population diversity. Dimension-by-dimension greedy search mode is used to do local search for individual of optimal chicken swarm in order to improve optimization precision. According to the test results of 18 standard test functions and 2 engineering structure optimization problems, this algorithm has better effect on optimization precision and speed compared with basic chicken algorithm and other intelligent optimization algorithms.

1. Introduction

Many problems in areas of scientific computing, engineering science, and business management can be concluded as global optimum problems. The consumption time of traditional accurate computation approaches for solving large-scale optimization problems increases exponentially. So this approach cannot meet the real requirements. To solve these problems, many scholars, by simulating life habits of creatures in the nature, presented intelligent swarm optimization algorithms including particle swarm optimization (PSO) [1], ant colony optimization (ACO) [2], artificial bee colony (ABC) [3], invasive weed colonization optimization (IWO) [4], firefly algorithm (FA) [5, 6], cuckoo search (CS) algorithm [7, 8], fish swarm algorithm (FSA) [9], bat algorithm (BA) [10, 11], monkey algorithm (MA) [12], krill herd (KH) algorithm, and flower pollination algorithm (FPA) [13]; all these have gained favorable results. As a new kind of burgeoning metaheuristic algorithm, intelligent swarm optimization algorithms have characteristics of high precision, fast convergence rate, and good stability and can obtain the exact solution or approximate solution of large-scale optimum problems within limited time.

Chicken swarm optimization (CSO) is a new intelligent bionic algorithm proposed by Meng et al. [14] in 2014, which simulates chickens swarm hierarchy and their food search behavior. The whole chicken swarm is divided into cock swarm, hen swarm, and chick swarm. Chickens with highest fitness values and lowest fitness values are taken as cock swarm and chick swarm, respectively, and the rest are taken as hen swarm. When solving optimization problems, each chicken in the swarm corresponds to a solution. Different search strategies are adopted for different subswarm according to different population. In contrast with standard particle swarm optimization, differential evolution, and bat algorithm, chicken swarm has advantage in either searching precision or convergence rate [14, 15]. In basic chicken swarm algorithm, a random search strategy based on Gaussian distribution is adopted for particles of cock swarm. This search strategy has strong ability of local development, but its global development ability is weak to make it liable to fall into a local optimum. For hen swarm particles, the searching is done by common guidance of cocks of their own population and other populations, which helps hen particles close to global optimum. However, the population diversity will lack and the hen particles will fall into a local optimum when most cock particles (with cock particles of own race and other races) are in a local optimum. Consequently, the global optimal solution cannot be obtained and its search performance will be affected for basic chicken swarm algorithm.

In this paper, a chicken swarm optimization algorithm on the basis of elite opposition-based learning (EOCSO) is presented to solve global optimum problems. A random search strategy based on dynamic adaptive t distribution is adopted in this algorithm for cock swarm to replace the random search based on Gaussian distribution. The local exploitation ability and global development ability are balanced. In order to improve the optimization precision and convergence rate of the algorithm, an opposition-based learning method is used to improve population diversity for hen swarm and a greedy dimension-by-dimension search mode is applied to individual of optimal chicken swarm for local search. Through experiments of 18 basic test functions and 2 engineering structure optimization measurement problems, the comparison among improved chicken swarm algorithm, basic chicken swarm algorithm, and other typical intelligent algorithms is conducted to show that improved chicken swarm algorithm has more excellent optimization precision, convergence rate, and robustness.

2. Chicken Swarm Algorithm

Chicken swarm optimization is a new intelligent bionic algorithm proposed according to various behaviors of cocks, hens, and chicks in the process of searching food. In this algorithm, chicken swarm in searching space is mapped as specific particle individual. Cock particle swarm, hen particle swarm, and chicken particle swarm are sorted according to fitness value of particle, and each subswarm uses different searching mode [16, 17].

In this algorithm, several particles with best fitness are selected as cock particle swarm, which is given by

$$x_{i,j}^{t+1} = x_{i,j}^{t} + \operatorname{randn}\left(0, \sigma^2\right) \cdot x_{i,j}^{t}, \tag{1}$$

where $x_{i,j}^{t+1}$ and $x_{i,j}^{t}$ are the position of jth dimension of particle i in $t+1$ and t iterations, respectively, and $\operatorname{randn}(0,\sigma^2)$ is a random number of Gaussian distribution whose variance is σ^2. The parameter σ^2 can be calculated by

$$\sigma^2 = \begin{cases} 1, & \text{fit}_i < \text{fit}_k \\ \exp\left(\dfrac{(\text{fit}_k - \text{fit}_i)}{(|\text{fit}_i| + \xi)}\right), & \text{fit}_i \geq \text{fit}_k, \end{cases} \tag{2}$$

where $i, k \in [1, r\text{size}]$ and $i \neq k$. $r$size represents the number of cock swarms. fit_i and fit_k are the fitness values of cock particle i and k, respectively; ξ represents a number which is small enough.

Moreover, most particles with good fitness are selected as hen swarm. Its random search is done via cocks of population of hen and that of others, which can be expressed as

$$\begin{aligned} x_{i,j}^{t+1} &= x_{i,j}^{t} + S1 \cdot \text{rand} \cdot \left(x_{r1,j}^{t} - x_{i,j}^{t}\right) + S2 \cdot \text{rand} \\ &\quad \cdot \left(x_{r2,j}^{t} - x_{i,j}^{t}\right), \end{aligned} \tag{3}$$

where $x_{r1,j}^{t}$ and $x_{r2,j}^{t}$ are the position of cock individual $r1$ in the population of hen x_i and cock individual $r2$ in the other population, respectively. rand is an uniform random number over $[0, 1]$. $S1$ and $S2$ denote the weight calculated by

$$\begin{aligned} S1 &= \exp\left(\frac{(\text{fit}_i - \text{fit}_{r1})}{(|\text{fit}_i| + \xi)}\right), \\ S2 &= \exp\left(\text{fit}_{r2} - \text{fit}_i\right), \end{aligned} \tag{4}$$

where fit_{r1} and fit_{r2} are, respectively, the fitness value of cock individual $r1$ in the population of hen x_i and cock individual $r2$ in the other population.

All individuals, except for cock swarm and hen swarm, are defined as chick swarm. Its search mode follows that of hen swarm, which is given by

$$x_{i,j}^{t+1} = x_{i,j}^{t} + \text{FL} \cdot \left(x_{m,j}^{t} - x_{i,j}^{t}\right), \quad \text{FL} \in [0, 2], \tag{5}$$

where FL stands for a parameter, meaning that the chick would follow its mother to forage for food. $x_{m,j}^{t}$ represents the position of the ith chick's mother ($m \in [1, N]$).

3. Chicken Swarm Optimization Based on Elite Opposition-Based Learning

3.1. Random Search Strategy Based on Dynamic Adaptive t Distribution. In chicken swarm optimization, a random search mode on the basis of Gaussian distribution is adopted for cock particle searching. This search mode has strong ability in local development, but it is weak in global exploitation. Therefore, enlargement of search space for cock particle will effectively reduce the risk to fall into a local optimum. Thus, the search mode with t distribution is used for cock particle in order to balance the global development ability and local exploitation ability of the algorithm.

The t distribution, also called student distribution [18], owns degree of freedom parameter n; its probability density function is as follows:

$$\begin{aligned} p_t(x) &= \frac{\Gamma((n+1)/2)}{\sqrt{n\pi} \cdot \Gamma(n/2)} \cdot \left(1 + \frac{x^2}{n}\right)^{-((n+1)(n+1)/2)}, \\ &\qquad -\infty < x < +\infty, \end{aligned} \tag{6}$$

where Γ is the Gamma function and $\Gamma(n + 1/2) = (2n)!\sqrt{\pi}/(n)!4^n$.

Apparently, the t distribution is defined as Cauchy distribution density function ($t(n = 1) = C(0, 1)$) when degree of freedom parameter $n = 1$ and is defined as

Gaussian distribution density function when $n = \infty$ ($t(n \to \infty) \to N(0,1)$). Thus, Cauchy distribution and Gaussian distribution are two special boundary cases of t distribution. The characteristics of Cauchy distribution and Gaussian distribution are integrated into t distribution, and different mutation range can be obtained via changing degree of freedom parameter n.

In the algorithm, random search based on dynamic adaptive t distribution is selected for the position of cock particles $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$, which is expressed as

$$x'_{i,j} = x_{i,j} + \psi_{i,j} \cdot t(n) \cdot x_{i,j}, \tag{7}$$

where $x'_{i,j}$ is the position after random search of dynamic adaptive t distribution used by jth dimension of cock particle i, $x_{i,j}$ is the position of jth dimension of ith cock particle, $t(n)$ is random function of t distribution with iteration n as its degree of freedom, and ψ is mutation control factor. Each dimension value performs differently in searching process of multidimensional objective function. Thus, the mutation control factor ψ has the same control strategy to each dimension, and it cannot meet the requirement of algorithm obviously. In this paper, dynamic adaptive mutation control factor is used to adjust mutation range, as given by

$$\psi_{i,j} = \beta \cdot \left| \frac{1}{n} \cdot \sum_{k=1}^{n} x_{k,j} - x_{\text{best},j} \right|, \tag{8}$$

where β is proportionality coefficient, n is the number of cock members, $(1/n) \cdot \sum_{k=1}^{n} x_{k,j}$ is mean value of jth dimension of cock particle in the current iteration, and $x_{\text{best},j}$ is a mutated value of jth dimension of cock particle in optimal position of the current iteration.

In random search of dynamic adaptive t distribution, iteration n is taken as degree of freedom parameter, and dynamic adaptive mutation control factor is used to adjust searching range. In the early stage of evolution, the search is conducted by t distribution (which is similar to that via Cauchy distribution) and maintains population diversity due to small n. Thus, the algorithm has good ability of global exploitation. With increase of n, random search of t distribution transits gradually from that of Cauchy distribution to that of Gaussian distribution, and random search of t distribution is similar to that of Gaussian distribution with good ability of local development in the later stage.

3.2. Elite Opposition-Based Learning. Elite opposition-based learning proposed by Tizhoosh [19] is a new technology applied to intelligent computing area, and it has been successfully applied in many intelligent algorithm optimizations [20–22]. Theoretically verified by Zhong and others [23], opposition-based learning can acquire a solution that closes to global optimal solution with a higher probability. In the paper, elite opposition-based learning is adopted to enlarge search range for hen swarm and enhance population diversity so as to improve searching performance of the algorithm. The basic thoughts are as follows.

Set the optimal cock position and its opposite position in tth iteration as $x^t_{\text{best}} = (x^t_{\text{best},1}, x^t_{\text{best},2}, \ldots, x^t_{\text{best},d})$ and $\text{op}^t_{\text{best}} = (\text{op}^t_{\text{best},1}, \text{op}^t_{\text{best},2}, \ldots, \text{op}^t_{\text{best},d})$, respectively; then

$$\text{op}^t_{\text{best},j} = k \cdot \left(la_j + ub_j\right) - x^t_{\text{best},j}, \tag{9}$$

where $x^t_{\text{best},j} \in [la_j, ub_j]$, k is a coefficient which denotes $(0,1)$ uniformly distributed random variable, and $[la_j, ub_j]$ is the boundary of jth dimension of search space in the current population. The result is obtained by

$$\begin{aligned} la_j &= \min\left(x_{i,j}\right), \\ ub_j &= \max\left(x_{i,j}\right), \end{aligned} \tag{10}$$

where $i \in (1, 2, \ldots, \text{size})$ and $j \in (1, 2, \ldots, d)$. Equation (10) is applied to replace fixed boundary, which ensures that cocks generate opposite solution in decreased space range. Besides, threshold value of the opposite solution may go beyond the range of $[la_j, ub_j]$ to form a nonfeasible solution. Traditional handling is to set the searching particle which goes beyond borders as boundary value. As a result, numerous searchers gather on the border. To avoid above problem, border buffering wall is used for handling, and its basic thoughts are as follows.

Suppose that a_j is upper bound of jth dimension of search range in the population and b_j is the lower bound. If opposite solution of the optimal individual in the population is $\text{op}'_{\text{best},j}$, the value after border buffering wall handling will be

$$\text{op}^t_{\text{best},j} = \begin{cases} a_j \cdot \left(\left(1 - \text{sgn}\left(a_j \cdot L\right)\right) + \text{sgn}\left(a_j \cdot L \cdot \left(\dfrac{\left|\alpha_{i,j} \cdot \text{op}^t_{\text{best},j}\right|}{a_j - b_j} \right) \right) \cdot \text{rand} \right), & \text{op}^t_{\text{best},j} > a_j \\ b_j \cdot \left(\left(1 + \text{sgn}\left(b_j \cdot L\right)\right) - \text{sgn}\left(b_j \cdot L \cdot \left(\dfrac{\left|\alpha_{i,j} \cdot \text{op}^t_{\text{best},j}\right|}{a_j - b_j} \right) \right) \cdot \text{rand} \right), & \text{op}^t_{\text{best},j} < b_j, \end{cases} \tag{11}$$

where sgn is sign function and $L \in [0,1]$ is a proper constant, which relates to the thickness of buffering wall.

With border buffering wall to process off normal individual, search individuals are able to decide their values

```
(1) Set the generation counter $t$, the maximum generation max_gen, dynamic adaptive
    change step step, max step step_max, the chicken individual $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,j}, \ldots, x_d)$.
(2) step = $\frac{(\text{max\_gen} - t)}{\text{max\_gen}} \cdot$ step_max    (*)
(3) for $k = 1 : d$
(4)     temp = $x_i$;
(5)     temp($k$) = temp($k$) + (2 * rand − 1) * step;
(6)     if $f(\text{temp}) < f(x_i)$
(7)         $x_i(k)$ = temp($k$);
(8)     end
(9) end
```

Algorithm 1

```
(1) Set the initial parameters, including the total population size popsize, the roosters accounts $Nr$,
    the hens accounts $Nh$, the mother hens accounts $Nm$, updating frequency of the
    chicken swarm $G$ and the maximum number of generations itermax et al.
(2) Generate a population $x = (x_1, x_2, \ldots, x_i, \ldots, x_{\text{popsize}})$ of popsize chickens with random solutions.
(3) Calculate the fitness fitness($x_i$) and find the best solution $x_{\text{best}}$ of the population.
(4) for iter = 1 : itermax
(5)     if iter% $G$ == 1 or iter == 1
(6)         Sort all population individuals according to their fitness.
(7)         Divide total population individuals into three subpopulations (called rooster population, hen
            population, and chickens population) according to their sort criteria, and establish
            the relationship between the chickens and its mother (hens).
(8)     end
(9)     Update the rooster population individuals according to Eq. (7) and calculate their fitness.
(10)    Update the hen population individuals according to Eq. (12) and calculate their fitness.
(11)    Update the chicken population individuals according to Eq. (5) and calculate their fitness.
(12)    Update the personal best position $x_i^*$ and the global optimal position $x_{\text{best}}$.
(13)    Perform local search for the global optimal individual according to Section 3.3.
(14) end
(15) Output the best solution $x_{\text{best}}$
```

Algorithm 2

dynamically according to the practical searching conditions and can better overcome deficiencies brought by traditional handling. The search mode of hen swarm using elite opposition-based learning can be expressed as follows:

$$x_{i,j}^{t+1} = x_{i,j}^{t} + S1 \cdot \text{rand} \cdot \left(x_{r1,j}^{t} - x_{i,j}^{t}\right) + S2 \cdot \text{rand} \cdot \left(x_{r2,j}^{t} - x_{i,j}^{t}\right) + S3 \cdot \text{rand} \cdot \left(\text{op}_{\text{best},j}^{t} - x_{i,j}^{t}\right), \quad (12)$$

where $S1$, $S2$, and $S3$ are weights, $S3$ and $S2$ have the same value, rand is an uniform random number over $[0, 1]$, and $\text{op}_{\text{best},j}^{t}$ is the opposite solution of the optimal particle of the population of chick swarm particle i in tth iteration.

3.3. Local Search via Greedy Dimension-by-Dimension Search. In basic chicken swarm optimization algorithm, the cock swarm, the hen swarm, and the chick swarm use different random search modes to acquire comparatively high-quality solution and convergence rate. But the three subswarms are all assessed by overall upgrading assessment method. For high-dimensional global optimization problem, overall upgrading assessment method will affect the quality of solution and convergence rate due to the interference among each dimension [24]. Suppose that the objective global optimization function is $f(x_i) = \sum_{j=1}^{3} x_{i,j}^2$ (sphere test function) and independent variable equals $x_i = (0.2, 0.2, 0.2)$. Then, $f(x_i) = 0.12$. In iterations, the independent variable is updated as $x_i' = (0.1, 0.3, 0.4)$ after overall upgrading. Then, $f(x_i') = 0.26$, and $x_i' = (0.1, 0.3, 0.4)$ will be abandoned in chicken swarm optimization as $f(x_i') > f(x_i)$. The overall upgrading assessment method will cause a slow convergence rate for the algorithm as the first-dimensional contribution is ignored due to deterioration of the second- and third-dimensional contributions.

The greedy dimension-by-dimension search mode in this paper makes full use of updated achievement of each dimension. The basic idea is as follows. (1) Suppose that $x_{i,j}'$ is the value after updating of the value of jth dimension of chicken swarm particle $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,j}, \ldots, x_{i,d})$ (fitness value

Table 1: The parameters set of all other algorithms.

Algorithms	Parameters
BA	Qmin = 0, Qmax = 2, $R^0 = 0.1$, $A = 0.9$, $\alpha = 0.95$, $\gamma = 0.9$
PSO	$c1 = 1.49445$, $c2 = 1.49445$, $\varpi = 0.729$
DE	pCR = 0.2, $\beta_{\min} = 0.2$, $\beta_{\max} = 0.8$
ACO	The intensification factor $q = 0.5$, the deviation-distance ratio zeta = 1
CS	pa = 0.25, $\alpha = 1$
FPA	$p = 0.8$
CSO	$Nr = 0.15$popsize, $Nh = 0.7$popsize, $Nm = 0.15$popsize, $G = 10$

is fit(x_i)). Then, $x'_{i,j}$ and the values of other dimensions of chicken swarm particle x_i before updating are integrated into a new chicken swarm particle $y = (x_{i,1}, x_{i,2}, \ldots, x'_{i,j}, \ldots, x_{i,d})$. (2) Calculate its fitness value fit(y). If fit(y) < fit(x_i), then the updated result will be saved. Otherwise, it will be abandoned and the updating of $(j+1)$th dimension will be conducted. The procedures are shown in Algorithm 1.

To ensure that each dimension of chicken particles in initial period of iteration updates in a relatively big step-length and in a small step-length in later period of iteration, dynamic adaptive step-length updating as (*) in Algorithm 1 is adopted to further improve the algorithm's convergence rate and quality of solution.

3.4. Algorithm Flow. Based on Sections 3.1–3.3, the procedure of the chicken swarm optimization algorithm is shown in Algorithm 2.

4. Results and Discussions

4.1. Parameter Setting. To test the algorithm's performance, seven typical intelligent algorithms, bat algorithm (BA) [10, 11], particle swarm optimization (PSO) [1], differential evolution (DE) [25], ant colony optimization (ACO) [2], cuckoo search (CS) algorithm [7], flower pollination algorithm (FPA) [13], and chick swarm algorithm (CSO) [14], are selected to do contrast experiments with EOCSO algorithm. The parameters of EOCSO algorithm are set as follows. Population size is set as 100, and each of the scale of cock swarm and that of chick swarm occupies 15%, while the scale of hen swarm takes up 70%. All of the common parameters of these algorithms (including the population size, dimensions, and maximum number of generations) are set to be the same for a fair comparison. Other parameters are described in Section 3. The parameters set of all other five algorithms is shown in Table 1.

4.2. Test Function. To verify the optimization precision and convergence rate of the proposed algorithm, 18 standard test functions in [26, 27] are chosen for contrast experiments. $f_1 \sim f_7$ are high-dimensional unimodal functions and are extremely hard to converge to a global optimum, which are used to inspect the searching precision. $f_8 \sim f_{13}$ are high-dimensional multimodal functions with several local extreme points. So they are used to test the global searching performance and avoid premature of the algorithms [28–30]. $f_{14} \sim f_{18}$ are low-dimensional multimodal functions. The standard test functions are seen in Table 2.

4.3. Influence of Dynamic Adaptive t Distribution Search Strategy on Bird Swarm Algorithm. Influence of dynamic adaptive t distribution search strategy on bird swarm algorithm contains two parts. The first part is the influence of different parameter values of variable control factor ψ on Gaussian distribution algorithm. The second part is the empirical comparison and analysis of CSO under dynamic adaptive t distribution (ATD-CSO) and dynamic adaptive Gaussian distribution (AGD-CSO) as well as the original algorithm (O-CSO). High-dimensional unimodal functions (f_3, f_5), high-dimensional multimodal functions (f_{11}, f_{13}), and low-dimensional multimodal functions (f_{15}, f_{17}) are selected as test functions.

4.3.1. Impact Analysis of Control Factor ψ. Other parameters remain invariant, 0.2, 0.5, and 0.8, and dynamic adaptive change is used as control factor ψ for Gaussian distribution algorithm. Test function parameters are set the same as Section 4.1, and test result is assessed from optimal value, mean value, worst value, and standard variance. Table 3 shows the statistical result of different ψ on the test functions.

As can be seen from Table 3, search effect is better when ψ is smaller. But it is not quite ideal for Gaussian distribution with fixed ψ to improve algorithm performance. Through dynamic adaptive parameter setting, the algorithm can be ensured to maintain a large search step size in initial search stage. In later stage, dynamic adaptive step size of control factor ψ is reduced, while the local search ability is enhanced. The test result proved that dynamic adaptive control factor has more advantages than single fixed control factor.

4.3.2. Comparison of ATD-CSO, AGD-CSO, and O-CSO. Table 4 indicates the statistical result of dynamic adaptive t distribution, dynamic adaptive Gaussian distribution, and original CSO algorithm on three different kinds of test function. As the data shows, in terms of optimal values, 6 and 4 orders of magnitude are improved for f_3 and f_{13} separately by ATD-CSO compared with AGD-CSO. For function f_5, searching precision of ATD-CSO is higher than that of AGD-CSO. For f_{11}, f_{15}, and f_{17}, optimal solution is searched by

Table 2: Benchmark test functions.

Number	Name	Benchmark test functions	Dimension	Scope	Optimum
$f_1(x)$	Sphere model	$f(x)=\sum_{i=1}^{D} x_i^2$	30	$[-100,100]$	0
$f_2(x)$	Schwefel's problem 2.22	$f(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	30	$[-10,10]$	0
$f_3(x)$	Schwefel's problem 1.2	$f(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_i\right)^2$	30	$[-100,100]$	0
$f_4(x)$	Schwefel's problem 2.21	$f(x)=\max_{i=1}^{D}\{\lvert x_i\rvert\}$	30	$[-100,100]$	0
$f_5(x)$	Generalized Rosen-brock's function	$f(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(1-x_i)^2\right]$	30	$[-30,30]$	0
$f_6(x)$	Step function	$f(x)=\sum_{i=1}^{D}\lfloor x_i+0.5\rfloor$	30	$[-100,100]$	0
$f_7(x)$	Quartic function, that is, noise	$f(x)=\sum_{i=1}^{D} i\cdot x_i^4+\text{random}(0,1)$	30	$[-1.28,1.28]$	0
$f_8(x)$	Generalized Schwefel's problem 2.26	$f(x)=\sum_{i=1}^{D}-x_i\cdot\sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	$[-500,500]$	$-418.9829*n$
$f_9(x)$	Generalized Rastrigin's function	$f(x)=\sum_{i=1}^{n}\left(x_i^2-10\cdot\cos(2\cdot\pi\cdot x_i)+10\right)$	30	$[-5.12,5.12]$	0
$f_{10}(x)$	Ackley's function	$f(x)=-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right)-\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right)+20+e$	30	$[-32,32]$	0
$f_{11}(x)$	Generalized Griewank function	$f(x)=\frac{1}{4000}\cdot\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\frac{x_i}{\sqrt{i}}+1$	30	$[-600,600]$	0
$f_{12}(x)$	Generalized penalized function	$f(x)=\frac{\pi}{D}\left\{10\sin^2(\pi\cdot y_1)+\sum_{i=1}^{D-1}(y_i-1)^2\left[1+10\sin^2(\pi\cdot y_{i+1})+(y_n-1)^2\right]\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$; $y_i=1+\frac{x_i+1}{4}$; $u(x_i,\alpha,k,m)=\begin{cases}k(x_i-\alpha)^m, & x_i>\alpha\\ 0, & -\alpha\le x_i\le\alpha\\ k(-x_i-\alpha)^m, & x_i\le\alpha\end{cases}$	30	$[-50,50]$	0
$f_{13}(x)$	Generalized penalized Function	$f(x)=0.1\left\{10\sin^2(3\pi\cdot x_1)+\sum_{i=1}^{D-1}(x_i-1)^2\left[1+10\sin^2(3\pi\cdot x_{i+1})\right]+(x_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	30	$[-50,50]$	0
$f_{14}(x)$	Shekel's foxholes function	$f(x)=\left[\frac{1}{500}+\sum_{j=25}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$	2	$[-65.56,65.56]$	0.9980
$f_{15}(x)$	Kowalik's function	$f(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	$[-5,5]$	0.0003075
$f_{16}(x)$	Hartman's function	$f(x)=-\sum_{i=1}^{4}c_i\cdot\exp\left[-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right]$	3	$[0,1]$	-3.8628

Table 2: Continued.

Number	Name	Benchmark test functions	Dimension	Scope	Optimum
$f_{17}(x)$	Hartman's function	$f(x) = -\sum_{i=1}^{4} c_i \cdot \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]$	6	$[0,1]$	−3.32
$f_{18}(x)$	Shekel's family	$f(x) = -\sum_{i=1}^{10}\left[(x-a_i)(x-a_i)^T + C_i\right]^{-1}$	10	$[0,10]$	−10.5364

Table 3: Influence of various ψ on the test functions.

Benchmark function		$f_3(x)$	$f_5(x)$	$f_{11}(x)$	$f_{13}(x)$	$f_{15}(x)$	$f_{17}(x)$
$\psi = 0.2$	Best	0.01631162	25.28779662	0	0.608008583	0.000307777	−3.321995171
	Mean	5.826567112	26.47886719	0	2.347145417	0.000428489	−3.308813113
	Worst	25.52896184	28.91557367	0	4.237078833	0.001223174	−3.202409203
	Std.	7.284604701	0.4586344	0	0.92771315	0.000277897	0.036625398
$\psi = 0.5$	Best	0.530928436	25.99904272	0	0.69896625	0.00031048	−3.321877282
	Mean	132.4241294	27.1885086	0	1.34834884	0.00044856	−3.315628525
	Worst	729.2974459	28.55843288	0	1.96348532	0.001223763	−3.201008218
	Std.	197.5724792	0.696629183	0	0.287524396	0.000224541	0.026987801
$\psi = 0.8$	Best	0.018825799	26.28800133	0	0.717391541	0.000315938	−3.321995171
	Mean	500.9268589	27.17671059	0	1.083153802	0.000534308	−3.301356117
	Worst	1228.354641	28.77558754	0	1.33573296	0.00071324	−3.199239821
	Std.	400.9524735	0.60589184	0	0.219814939	0.000140829	0.044548317
Original strategy	Best	$7.00E+01$	26.50669099	0	0.809497151	0.000330135	−3.321995172
	Mean	$8.06E+02$	27.28932938	0	1.07318734	0.000615881	−3.291752652
	Worst	$1.67E+03$	28.77578053	0	1.574427431	0.001262543	−3.200111233
	Std.	430.7691621	0.537375926	0	0.193250844	0.000208736	0.053478449
Dynamic adaptive	Best	$1.455387E-22$	0.09980032	0	$1.90772E-12$	0.000307486	−3.321995172
	Mean	$2.932745E-20$	0.57800657	0	$9.09553E-12$	0.000307486	−3.321995172
	Worst	$9.453261E-20$	1.08886543	0	$6.57632E-10$	0.000307486	−3.321995172
	Std.	$5.187532E-21$	0.19345322	0	$2.42964E-12$	0	$5.264821E-12$

Table 4: Influence of different distributions on test functions.

Benchmark function		$f_3(x)$	$f_5(x)$	$f_{11}(x)$	$f_{13}(x)$	$f_{15}(x)$	$f_{17}(x)$
Original CSO	Best	$7.00E+01$	26.50669099	0	0.809497151	0.000330135	−3.321995172
	Mean	$8.06E+02$	27.28932938	0	1.07318734	0.000615881	−3.291752652
	Worst	$1.67E+03$	28.77578053	0	1.574427431	0.001262543	−3.200111233
	Std.	430.7691621	0.537375926	0	0.193250844	0.000208736	0.053478449
AGD-CSO	Best	$1.455387E-22$	0.09980032	0	$1.90772E-12$	0.000307486	−3.321995172
	Mean	$2.932745E-20$	0.57800657	0	$9.09553E-12$	0.000307486	−3.321995172
	Worst	$9.453261E-20$	1.08886543	0	$6.57632E-10$	0.000307486	−3.321995172
	Std.	$5.187532E-21$	0.19345322	0	$2.42964E-12$	0	$5.264821E-12$
ATD-CSO	Best	$3.81028E-28$	0.050669099	0	$5.43358E-16$	0.000307486	−3.321995172
	Mean	$1.24542E-27$	0.18932938	0	$5.16541E-15$	0.000307486	−3.321995172
	Worst	$1.06346E-27$	0.67578053	0	$4.81746E-13$	0.000307486	−3.321995172
	Std.	$9.19088E-28$	0.037375926	0	$4.52782E-15$	0	$9.858438E-14$

both ATD-CSO and AGD-CSO. Also, for mean value, worst value, and standard variance, f_{11} and f_{15} obtain the same through these two algorithms. But, for the standard variance for f_{17}, ATD-CSO is better than AGD-CSO. Compared with O-CSO, ATD-CSO produces optimal mean value, worst value, and variance. It can be seen from Figures 1–6 that, in early evolution period, convergence rate of ATD-CSO is faster than that of AGD-CSO when calculating the above 6 functions. In later period, the convergence rate is similar, but the overall convergence rate of ATD-CSO is more optimal.

Table 5: Statistical results of the influence of dynamic adaptive t distribution and opposition-based learning on CSO.

Benchmark function		$f_3(x)$	$f_5(x)$	$f_{11}(x)$	$f_{13}(x)$	$f_{15}(x)$	$f_{17}(x)$
Original CSO	Best	$7.00E+01$	26.50669099	0	0.809497151	0.000330135	−3.321995172
	Mean	$8.06E+02$	27.28932938	0	1.07318734	0.000615881	−3.291752652
	Worst	$1.67E+03$	28.77578053	0	1.574427431	0.001262543	−3.200111233
	Std.	430.7691621	0.537375926	0	0.193250844	0.000208736	0.053478449
ATD-EO-CSO	Best	$7.108233E-42$	$2.844564E-6$	0	$2.095481E-25$	0.000307486	−3.321995172
	Mean	$1.039602E-40$	$9.654632E-6$	0	$2.006629E-24$	0.000307486	−3.321995172
	Worst	$4.540331E-39$	$3.239232E-5$	0	$9.910062E-24$	0.000307486	−3.321995172
	Std.	$3.919585E-41$	$3.6784872E-7$	0	$1.887675E-25$	0	$6.455396E-14$

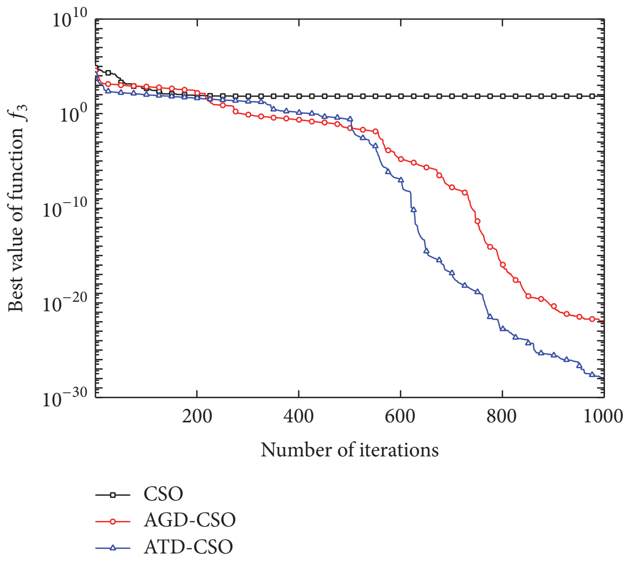


Figure 1: Convergence rates for $f_3(x)$.

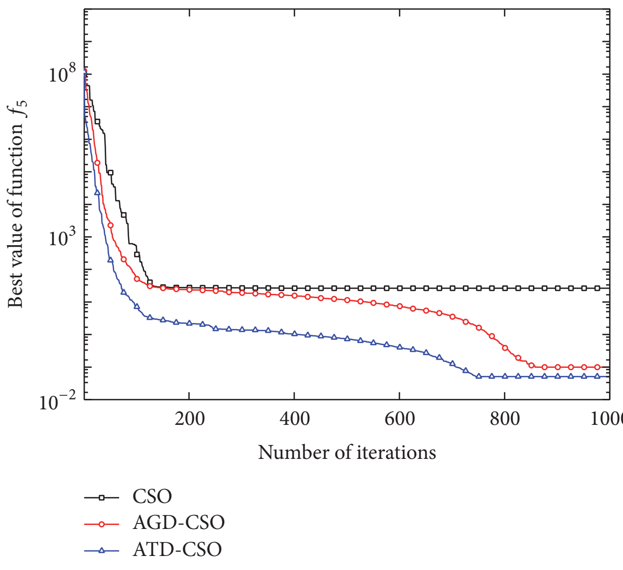


Figure 2: Convergence rates for $f_5(x)$.

However, the convergence rate of O-CSO is slower than that of ATD-CSO and AGD-CSO. Therefore, ATD-CSO has some advantages in terms of solution precision, robustness, and convergence rate generally.

4.4. The Influence of Dynamic Adaptive t Distribution and Opposition-Based Learning on CSO (ATD-EO-CSO). As demonstrated in Table 5, the best value, mean value, worst value, and variance of f_3, f_5, f_{13}, f_{15} obtained by dynamic adaptive t distribution and elite opposition-based learning are apparently better than those obtained by original CSO. Target value can be searched on function f_{11} by both ATD-EO-CSO and O-CSO. Therefore, the O-CSO is suitable to solve this function, and the improvement of the solution after optimization is not obvious. For function f_{17}, the best value can be searched by both. However, in terms of mean value, worst value, and variance, the results will be better based on cowork of both t distribution and elite opposition-based learning. Since searching can be done more widely via elite opposition-based learning, boundary buffering wall is adopted to deal with individuals crossing the border, avoiding the deficiency that individuals gather on boundary value to improve the diversity of population. As a result, ATD-EO-CSO is superior in solution precision.

4.5. Influence of Local Search via Greedy Dimension-by-Dimension Search on Bird Swarm Algorithm. Influence of local search via greedy dimension-by-dimension search on bird swarm algorithm contains two parts. The first part is the influence of fixed step size step = 0.1, step = 0.3, step = 0.6, and the adaptive step on algorithm. The second part is the comparison between adaptive step strategy and original CSO. Table 6 shows the statistical result by means of various step values. According to Table 6, when step value is small, higher search precision can be obtained, but it is liable to get into premature convergence. In case of larger step value, the range of search step size is larger, and it is liable to skip optimal solution range, which causes lower searching effect in later stage of evolution and reduced local search ability. Through adaptive step, the algorithm's global research can be ensured to conduct in long step size at initial stage and in short step size at later stage, which improves the search efficiency of dimension-by-dimension search. Hence, adaptive step dimension-by-dimension search is superior to the fixed step size strategy in terms of search precision.

TABLE 6: Influence of various step values on test functions.

Benchmark function		$f_3(x)$	$f_5(x)$	$f_{11}(x)$	$f_{13}(x)$	$f_{15}(x)$	$f_{17}(x)$
step = 0.1	Best	$1.1009139E-30$	$2.5022290E-4$	0	$7.1919782E-22$	0.000307486	−3.321995172
	Mean	$2.5675751E-26$	$7.1700102E-4$	0	$2.0929358E-20$	0.000307486	−3.321995163
	Worst	$5.9338088E-25$	$5.0003902E-3$	0	$9.0510335E-20$	0.000307486	−3.321995061
	Std.	$5.4742928E-26$	$4.8413835E-5$	0	$6.9769787E-22$	$7.537751E-10$	$9.6237934E-7$
step = 0.3	Best	$3.0856694E-28$	$4.3623649E-5$	0	$4.9527803E-18$	0.000307486	−3.321995172
	Mean	$6.4211983E-25$	$3.3217367E-4$	0	$5.8674385E-15$	0.000307486	−3.321995172
	Worst	$5.9480356E-24$	$7.1641617E-4$	0	$6.7902546E-12$	0.000307486	−3.321995163
	Std.	$1.2289958E-26$	$8.0512434E-5$	0	$2.6579955E-16$	$7.967684E-10$	$5.616744E-8$
step = 0.6	Best	$4.0232351E-26$	$9.7209661E-4$	0	$5.2857901E-25$	0.000307486	−3.321995172
	Mean	$1.3561952E-22$	$3.5667080E-3$	0	$9.2285451E-20$	0.000307499	−3.321956549
	Worst	$7.4188975E-21$	$8.5082852E-3$	0	$2.1028715E-18$	0.000307495	−3.321906894
	Std.	$6.2712862E-22$	$2.4668710E-4$	0	$7.0434808E-20$	$4.460765E-8$	$8.512414E-5$
Original CSO	Best	$7.00E+01$	26.50669099	0	0.809497151	0.000330135	−3.321995172
	Mean	$8.06E+02$	27.28932938	0	1.07318734	0.000615881	−3.291752652
	Worst	$1.67E+03$	28.77578053	0	1.574427431	0.001262543	−3.200111233
	Std.	430.7691621	0.537375926	0	0.193250844	0.000208736	0.053478449
Adaptive step	Best	$9.5838158E-40$	$8.9257907E-3$	0	$9.4224685E-25$	0.000307486	−3.321995172
	Mean	$1.8439439E-36$	$3.7218918E-2$	0	$5.3242987E-23$	0.000307486	−3.321995172
	Worst	$8.5706881E-36$	$7.5851102E-2$	0	$8.1986156E-22$	0.000307486	−3.321995172
	Std.	$1.0682863E-37$	$2.3586933E-2$	0	$2.1204222E-23$	$9.124585E-18$	$8.884563E-15$

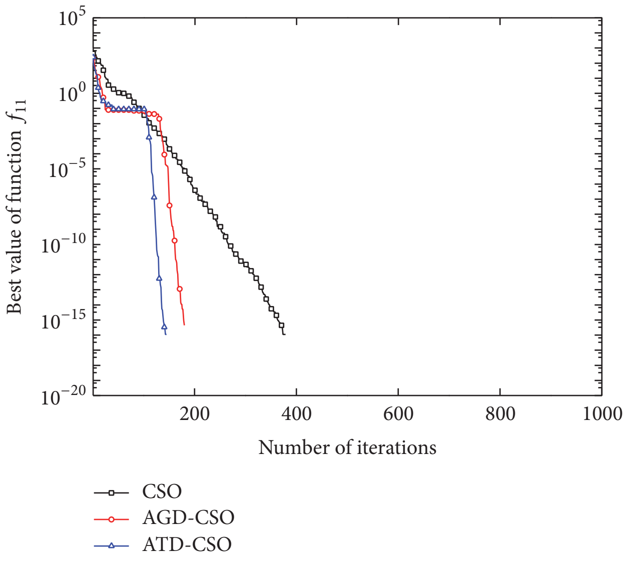


FIGURE 3: Convergence rates for $f_{11}(x)$.

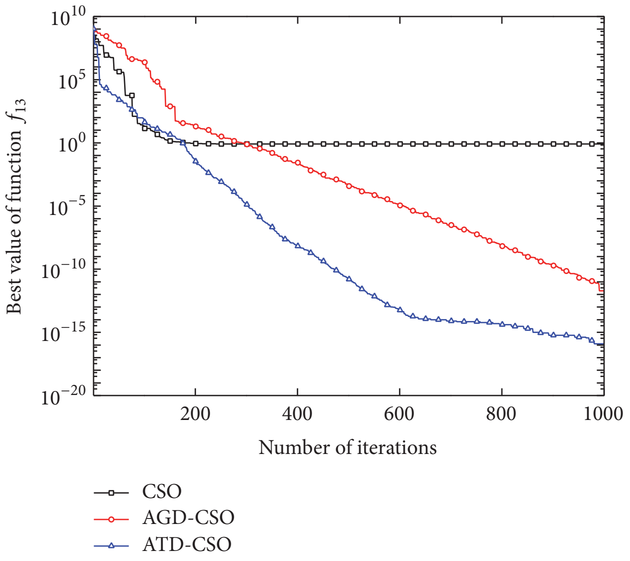


FIGURE 4: Convergence rates for $f_{13}(x)$.

4.6. Analysis of Experimental Results. In order to avoid being influenced by random factors, the experimental test for each case is conducted by 20 trials independently. The algorithm's searching performance is assessed according to the best value, mean value, standard deviation, and the worst value produced by the test result. The number of iterations of $f_{14} \sim f_{18}$ and $f_1 \sim f_{13}$ is 50 and 1000 in each trial, respectively. The algorithm is examined by Matlab 2012a on the platform with Win 8 OS, Intel Core i5-4210U 2.4 GHZ CPU, and 4 GB memory. The test statistical results are shown in Tables 7, 8, and 9.

The test statistical results of functions $f_1 \sim f_7$ are shown in Table 7. As can be seen from Table 7, the best value, mean value, standard variance, and the worst value of EOCSO are all superior to those of other intelligent algorithms (including BA, PSO, DE, ACO, CS, FPA, and CSO). Particularly, global minimum of the test function $f_5(x)$ lies in the bottom of parabola, which is quite reliable to fall into a local optimum in the searching process. The optimal result of EOCSO is $9.53224E-09$, which is 8, 11, 10, 10, 7, 3, and 10 orders of magnitude higher than that of BA, PSO, DE, ACO, CS,

TABLE 7: Test statistical results of functions $f_1 \sim f_7$.

Benchmark function		BA	PSO	DE	ACO	CS	FPA	CSO	EOCSO
$f_1(x)$	Best	$9.41672E-06$	6.774921774	$3.036519E-12$	$4.98686E-08$	0.040549145	$6.34628E-06$	$1.1893E-55$	$1.19E-201$
	Mean	$1.20474E-05$	20.80098597	$5.713553E-12$	$1.8224E-07$	0.08183638	0.000732691	$3.08046E-52$	$7.0639E-198$
	Worst	0.000014224	31.55593153	$9.811158E-12$	$3.76894E-07$	0.120709249	0.009162134	$3.16469E-51$	$7.2029E-197$
	Std.	$1.40714E-06$	9.102140636	$2.040785E-12$	$8.35921E-08$	0.021679322	0.002091063	$8.83269E-52$	0
$f_2(x)$	Best	0.012593909	6.594522786	$3.042127E-08$	$4.92993E-05$	1.479134511	$1.97228E-07$	$1.91405E-43$	$6.5805E-116$
	Mean	31.7282206	15.68696047	$4.943835E-08$	0.000507759	2.180229997	$7.11331E-06$	$6.5601E-41$	$2.13E-112$
	Worst	131.069574	35.29663724	$7.117019E-08$	0.00191464	3.029290054	$1.51764E-05$	$8.03444E-40$	$9.5751E-112$
	Std.	48.01800899	11.87133052	$1.339655E-08$	0.000549138	0.403631239	$4.85023E-06$	$1.78694E-40$	$2.8682E-112$
$f_3(x)$	Best	$2.27E-05$	$4.13E+02$	15255.144282	11951.29445	$5.99E+02$	$2.57E-02$	$7.00E+01$	$4.49E-51$
	Mean	$3.94E+02$	$9.17E+02$	24696.821583	16566.43552	$8.59E+02$	$6.70E-02$	$8.06E+02$	$9.55E-49$
	Worst	$4.77E+03$	$1.75E+03$	29125.784537	22644.44883	$1.10E+03$	$1.10E-01$	$1.67E+03$	$1.28E-47$
	Std.	1081.678915	411.6200929	3515.8752513	3287.33213	122.9938329	0.023267666	430.7691621	$2.85739E-48$
$f_4(x)$	Best	6.573087982	5.90965909	1.2612253015	3.058888831	5.273452649	12.24579118	0.000935694	$1.41944E-17$
	Mean	12.41978556	9.027042499	2.0432637030	5.055981092	6.116103354	15.33881514	3.027225289	$2.20256E-16$
	Worst	22.65352526	12.9169796	2.5042954408	7.296151756	7.215730774	19.26838273	10.74127991	$7.30302E-16$
	Std.	3.804254343	2.121495266	0.3514754143	0.953357307	0.52178179	2.184822605	3.480451504	$1.96521E-16$
$f_5(x)$	Best	0.196158304	343.6027646	25.837681251	17.1433613	0.013792405	$2.12808E-06$	26.50669099	$9.53224E-09$
	Mean	1.836717634	530.442542	39.933721918	17.86118196	0.59160704	0.398901513	27.28932938	0.000161047
	Worst	20.41255589	1427.570578	70.708954223	19.24540999	1.749078665	3.986765311	28.77578053	0.001214443
	Std.	4.605435894	299.0234027	13.215780402	0.617903448	0.57410589	1.227005992	0.537375926	0.000359017
$f_6(x)$	Best	$7.69118E-06$	13.55667233	$1.888545E-12$	$5.85581E-08$	0	$5.47537E-06$	1.671870558	0
	Mean	$9.79475E-06$	19.24687067	$5.975184E-12$	$1.68807E-07$	$3.77265E-10$	0.000265874	2.125011694	0
	Worst	$1.13079E-05$	25.44278333	$1.122444E-11$	$4.86966E-07$	$1.85832E-09$	0.000603906	2.710216862	0
	Std.	$9.19879E-07$	4.68251131	$2.832052E-12$	$9.4266E-08$	$4.55015E-10$	0.000212809	0.301038821	0
$f_7(x)$	Best	0.021225243	0.198906064	0.0171466731	0.01217843	0.001333226	0.029750255	0.000690044	0.001333226
	Mean	0.039176522	0.294912344	0.0240711071	0.024336378	0.003336074	0.108161359	0.002471843	0.003336074
	Worst	0.0606201	0.372732671	0.0301412708	0.034138264	0.004764046	0.160329105	0.005679536	0.004764046
	Std.	0.011065883	0.061817637	0.0041777349	0.007196828	0.000889975	0.044585434	0.001443844	0.000889975

FPA, and CSO, respectively. The mean value of EOCSO is $1.61047E-5$, which is 5, 7, 4, 4, 4, 4, and 6 orders of magnitude higher than that of the compared algorithms, respectively. Besides, EOCSO algorithm is also superior to seven other intelligent algorithms in terms of the worst value and standard variance. It comes to a conclusion that EOCSO algorithm has high precision and good robustness in searching high-dimensional unimodal function.

The test statistical result of functions $f_8 \sim f_{13}$ is demonstrated in Table 8, from which we can find that EOSCO algorithm can obtain global optimal extreme value for f_8, f_7, and f_{11}, which is free from interference of local extreme value. For other algorithms, only CSO can obtain global optimum in searching of f_9 and f_{11}. For f_{10}, the result is almost equivalent to that of EOCSO, CSO, and CS. Therefore, the original CSO is suitable to solve this function; the improvement of the solution after optimization is not obvious. With regard to $f_{12} \sim f_{13}$, EOSCO does better than five other algorithms in precision and standard variance of its solution, which reflects the advantage of EOCSO.

Table 8: Test statistical results of functions $f_8 \sim f_{13}$.

Benchmark function		BA	PSO	DE	ACO	CS	FPA	CSO	EOCSO
$f_8(x)$	Best	−8502.897133	−7963.943031	−12569.479219	−6090.034663	−9009.250145	−10694.82285	−9133.943944	−12569.48662
	Mean	−7060.92851	−6682.009356	−12546.705396	−8060.242464	−8449.968964	−10097.82587	−8186.965487	−12474.61802
	Worst	−5969.754314	−5622.614213	−12369.793372	−351.75903	−7986.143265	−9466.344181	−7087.508251	−12405.53547
	Std.	779.3469624	760.027236	53.131673	569.1183589	345.8135405	423.049839	596.4376456	51.6979597
$f_9(x)$	Best	30.84598172	71.56741319	47.02959855	108.4758259	6.691249469	15.91943794	0	0
	Mean	75.66883074	134.1736791	60.42457292	124.3756751	13.42928534	22.83514081	0	0
	Worst	146.2602458	177.1407232	71.54930622	138.2030082	20.70524753	27.467636	0	0
	Std.	32.32801039	35.8886515	7.058839227	7.737668073	4.409105246	3.170192621	0	0
$f_{10}(x)$	Best	11.60989873	4.914039621	$4.5281E-07$	$6.54539E-05$	$4.44089E-15$	2.013315503	$4.44089E-15$	$4.44089E-15$
	Mean	13.64745922	5.590179783	$6.07484E-07$	0.00013903	$4.74731E-15$	2.981722685	$5.50671E-15$	$4.70507E-15$
	Worst	15.09860799	6.074485544	$7.84382E-07$	0.00022114	$5.09361E-15$	4.298104668	$7.99361E-15$	$7.99361E-15$
	Std.	0.891126943	0.311563718	$1.11025E-07$	$3.80777E-05$	$2.25609E-16$	0.762341832	$1.67035E-15$	$7.99919E-16$
$f_{11}(x)$	Best	44.79000242	0.997496962	$6.55065E-12$	0.192490967	0	0.000152641	0	0
	Mean	79.03694557	1.203784605	$1.42400E-10$	0.360243829	0	0.016635909	0	0
	Worst	136.5821255	1.333771315	$5.74106E-10$	0.540731319	0	0.029692265	0	0
	Std.	23.79302299	0.092691068	$1.63674E-10$	0.092281983	0	0.009192596	0	0
$f_{12}(x)$	Best	5.506766196	2.320391097	$1.49415E-13$	$1.11888E-06$	$1.02781E-09$	$2.91491E-06$	0.072486935	$1.57054E-32$
	Mean	2371.120996	5.817999746	$5.97228E-13$	$8.9491E-06$	0.024315241	0.288022193	0.127682199	$8.79042E-30$
	Worst	23844.45981	9.370164248	$2.12638E-12$	$2.61683E-05$	0.103669022	5.455285401	0.294365491	$5.22E-29$
	Std.	6390.937613	2.137993761	$4.2672E-13$	$7.29249E-06$	0.033905983	1.217902188	0.063713849	$1.37918E-29$
$f_{13}(x)$	Best	64.83329341	7.038580766	$9.44877E-13$	$9.89913E-06$	$4.1161E-08$	0.153346238	0.809497151	$1.34978E-32$
	Mean	571341.8789	13.34788267	$3.57122E-12$	$6.3923E-05$	0.056026931	11.23615156	1.07318734	$4.63024E-28$
	Worst	5468139.556	20.53950899	$1.56888E-11$	0.000309668	0.393967438	24.58405568	1.574427431	$9.1725E-27$
	Std.	1242194.366	4.368603518	$3.0416E-12$	$6.7996E-05$	0.097188974	8.579601122	0.193250844	$2.05001E-27$

Table 9: Test statistical results of functions $f_{14} \sim f_{18}$.

Benchmark function		BA	PSO	DE	ACO	CS	FPA	CSO	EOCSO
$f_{14}(x)$	Best	0.998003838	0.998003838	0.998003838	1.000647439	0.998003838	0.998003838	0.998003838	0.998003838
	Mean	6.991491327	0.998003839	0.998003838	1.937812339	0.998003838	0.998003838	0.998003838	0.998003838
	Worst	20.15348696	0.99800384	0.998003838	3.969893097	0.998003838	0.998003838	0.998003838	0.998003838
	Std.	5.441619771	$5.35015E-10$	0	0.73957407	0	0	0	0
$f_{15}(x)$	Best	0.000307834	0.000307498	0.000368258	0.001325092	0.000483732	0.000307486	0.000330135	0.000307486
	Mean	0.001478932	0.000510263	0.000629779	0.002483753	0.000707371	0.000307486	0.000615881	0.000307486
	Worst	0.007775518	0.000780325	0.000764669	0.004487698	0.001227785	0.000307486	0.001262543	0.000307486
	Std.	0.001522132	0.000197339	0.000107155	0.000907677	0.000195684	$1.0842E-19$	0.000208736	0
$f_{16}(x)$	Best	−3.862782147	−3.862781836	−3.862782148	−3.862782111	−3.862782148	−3.862782148	−3.862782148	−3.862782148
	Mean	−3.862782125	−3.862776258	−3.862782148	−3.862781771	−3.862782148	−3.862782148	−3.862782148	−3.862782148
	Worst	−3.862782102	−3.862763876	−3.862782148	−3.862780215	−3.862782148	−3.862782148	−3.862782146	−3.862782148
	Std.	$1.44804E-08$	$4.86058E-06$	$2.27813E-15$	$4.04355E-07$	$2.27813E-15$	$2.27813E-15$	$4.9224E-10$	$2.27813E-15$
$f_{17}(x)$	Best	−3.321994333	−3.321950303	−3.321995172	−3.317012353	−3.32199515	−3.321995172	−3.321995172	−3.321995172
	Mean	−3.262546901	−3.228484991	−3.321440415	−3.24287723	−3.286327027	−3.321995172	−3.291752652	−3.321995172
	Worst	−3.203098591	−3.132218821	−3.311000677	−2.949849459	−3.203099953	−3.321995172	−3.200111233	−3.321995172
	Std.	0.060990736	0.081326677	0.002457311	0.08989745	0.055899237	$4.55626E-16$	0.053478449	$9.11252E-16$
$f_{18}(x)$	Best	−10.5364095	−10.53508305	−10.53640982	−10.43395927	−10.53640982	−10.53640982	−10.53640982	−10.53640981
	Mean	−6.282649281	−9.273746833	−10.53640982	−9.464159333	−10.53640982	−10.53640982	−10.53536203	−10.53636531
	Worst	−2.427335106	−2.425929198	−10.53640982	−5.139776016	−10.53640982	−10.53640982	−10.52081005	−10.53595233
	Std.	3.666609304	2.64050751	$3.99158E-13$	1.221326502	$3.64501E-15$	0.00010276	0.003534247	$1.8225E-15$

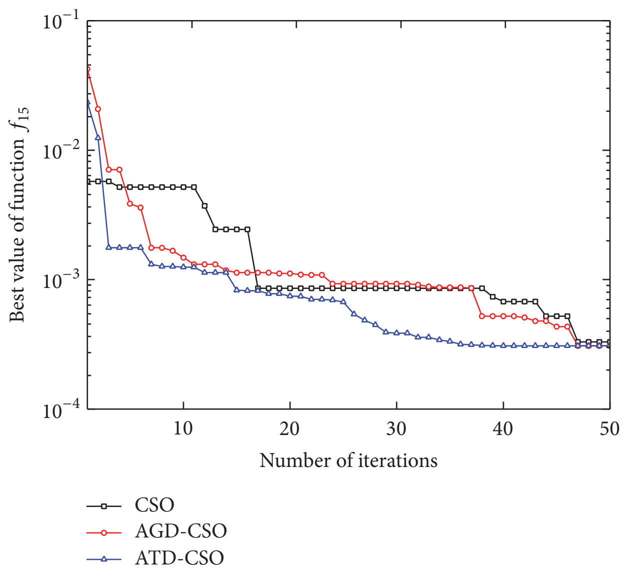


Figure 5: Convergence rates for $f_{15}(x)$.

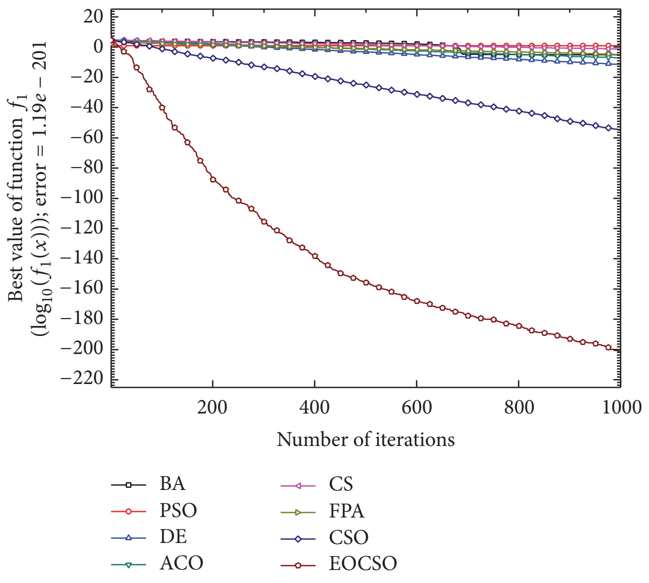


Figure 7: Convergence rates for $f_1(x)$.

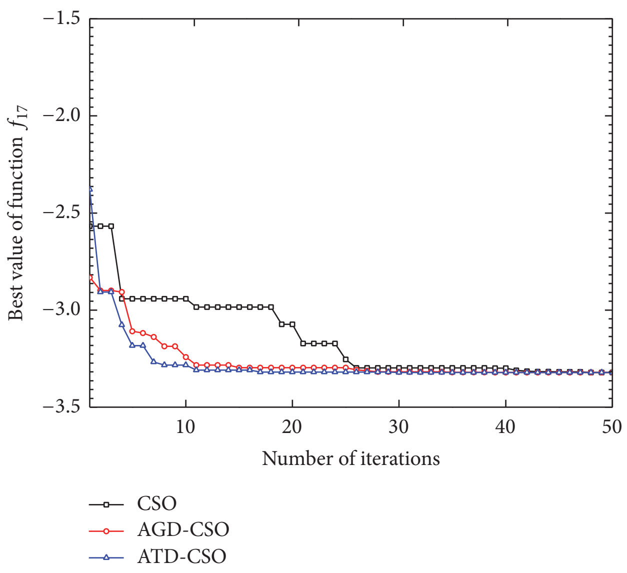


Figure 6: Convergence rates for $f_{17}(x)$.

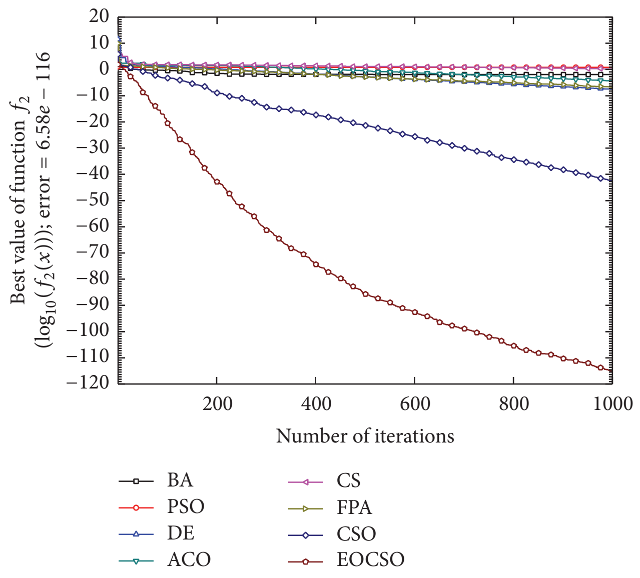


Figure 8: Convergence rates for $f_2(x)$.

In Table 9, the test statistical result of low-dimension multimodal functions $f_{14} \sim f_{18}$ is described. EOCSO and seven other algorithms can be seen to be able to get the global optimum in searching process, but EOCSO has fixed advantages in its mean value, worst value, and standard variance. "Error" on y-axis is the error between actual global minimum and converged minimum. The convergence graphs of the 6 algorithms are expressed in Figures 7–24. In EOCSO algorithm, its convergence rate of all 18 functions is more excellent than that in the other five algorithms, especially for $f_1 \sim f_4$, f_6, and $f_8 \sim f_{13}$. The convergence curve is quite smooth and drops rapidly, reflecting its good convergence rate.

4.7. Structural Design Examples. In order to validate the performance of proposed method for constraint problems, EOCSO is examined by solving constrained engineering design problems, such as speed reducer design problem and pressure vessel design problem.

4.7.1. Speed Reducer Design Problem. Speed reducer design problem is proposed by the famous scholar Mezura-Montes, which is a classic constrained optimization and is used to verify the design engineering constrained optimization algorithm performance, as shown in Figure 25. There are 7

Table 10: Comparison of the best solutions obtained by different methods for speed reducer design problem.

	EOCSO	PSO-DE [31]	MBA [32]	HEAA [33]	HGA [34]
x_1	3.500000000000003	3.5000000	3.500000	3.5000228993	3.500000
x_2	0.700000000000000	0.700000	0.700000	0.7000003924	0.700000
x_3	17.00000000000000	17.000000	17.000000	17.0000128592	17
x_4	7.300000000000000	7.300000	7.300033	7.3004277414	7.300000
x_5	7.715319911478278	7.800000	7.715772	7.7153774494	7.71533234
x_6	3.350214666096451	3.350214	3.350218	3.3502309666	3.35021511
x_7	5.286654464980222	5.2866832	5.286654	5.2866636970	5.28666404
$g_1(x)$	−0.073915280397874	−0.07391528	−0.07391528	−0.07392283	−0.07391528
$g_2(x)$	−0.197998527141950	−0.19799853	−0.19799853	−0.19800568	−0.19799853
$g_3(x)$	−0.499172248102422	−0.49917185	−0.49916745	−0.49909455	−0.49917251
$g_4(x)$	−0.904643904556068	−0.90147170	−0.90462711	−0.90464255	−0.90464413
$g_5(x)$	$-3.33066907388e-15$	$5.96466298e-07$	$-2.93020139e-06$	$-1.3895524e-05$	$-3.97499993e-07$
$g_6(x)$	0	$1.68865328e-08$	$3.505341306e-07$	$-5.2295669e-06$	$-5.43110185e-06$
$g_7(x)$	−0.702500000000000	−0.7025000	−0.70250000	−0.7024996	−0.70250000
$g_8(x)$	$-8.881784197001e-16$	0	0	$-6.1140457e-06$	0
$g_9(x)$	−0.583333333333333	−0.58333333	−0.58333333	−0.58333078	−0.58333333
$g_{10}(x)$	−0.051325753541825	−0.05132589	−0.05132936	−0.05137799	−0.05132566
$g_{11}(x)$	$-4.21884749358e-15$	−0.01085237	$-5.86590687e-005$	$-6.14133272e-06$	$-2.45744437e-07$
$f(x)$	2994.341315684048	2996.348167	2994.482453	2994.499107	2994.47

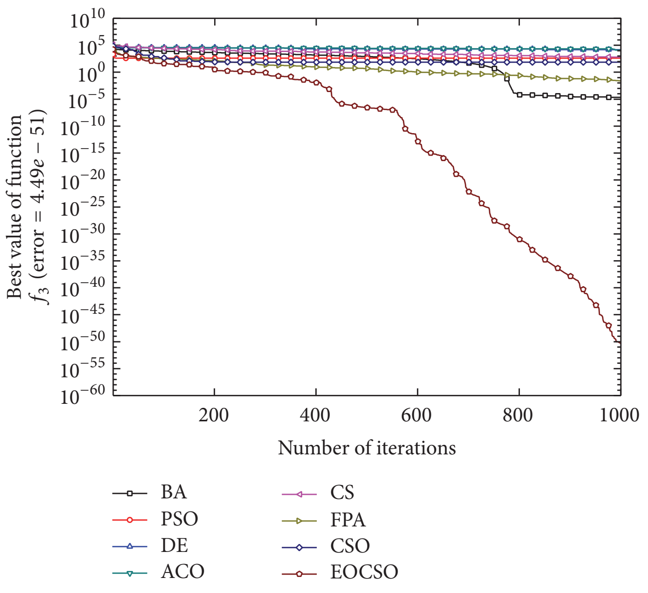


Figure 9: Convergence rates for $f_3(x)$.

variables and 11 inequality constraints of the problem. The mathematical model is represented as follows:

$$\begin{aligned}
\min \quad & f(x) \\
& = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) \\
& \quad - 1.508x_1\left(x_6^2 + x_7^2\right) + 7.477\left(x_6^3 + x_7^3\right) \\
& \quad + 0.7854\left(x_4x_6^2 + x_5x_7^2\right) \\
\text{s.t.} \quad & g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0, \\
& g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0, \\
& g_3(x) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0, \\
& g_4(x) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0, \\
& g_5(x) = \frac{\left[\left(745x_4/x_2x_3\right)^2 + 16.9\times 10^6\right]^{1/2}}{110x_6^3} - 1 \leq 0, \\
& g_6(x) = \frac{\left[\left(745x_5/x_2x_3\right)^2 + 157.5\times 10^6\right]^{1/2}}{85x_7^3} - 1 \\
& \leq 0, \\
& g_7(x) = \frac{x_2x_3}{40} - 1 \leq 0, \\
& g_8(x) = \frac{5x_2}{x_1} - 1 \leq 0, \\
& g_9(x) = \frac{x_1}{12x_2} - 1 \leq 0, \\
& g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0, \\
& g_{11}(x) = \frac{1.5x_6 + 1.7}{x_5} - 1 \leq 0,
\end{aligned} \tag{13}$$

where $2.6 \leq x_1 \leq 3.6$, $0.7 \leq x_2 \leq 0.8$, $17 \leq x_3 \leq 28$, $7.3 \leq x_4, x_5 \leq 8.3$, $2.9 \leq x_6 \leq 3.9$, and $5.0 \leq x_7 \leq 5.5$. This case study has been previously solved using other methods such as PSO-DE [31], MBA [32], HEAA [33], and HGA [34]. The best results of the various methods for solving the speed reducer design problem are shown in Table 7. Table 8 shows the statistical results of the different methods.

It can be seen from Table 10 that the optimal solution of the EOCSO algorithm is better than the other 4 kinds

Table 11: Comparison of statistical results for speed reducer design problem by the various algorithms.

Algorithm	Best	Mean	Worst	Std.
EOCSO	2994.3413156840	2994.3413169852	2994.3413180138	$1.2012791E-06$
PSO-DE [29]	2996.348167	2996.348174	2996.348204	$6.4E-06$
MBA [34]	2994.482453	2996.769019	2999.652444	1.56
HEAA [30]	2994.499107	2994.613368	2994.752311	$7.0E-02$
HGA [25]	2994.47	NA	NA	NA

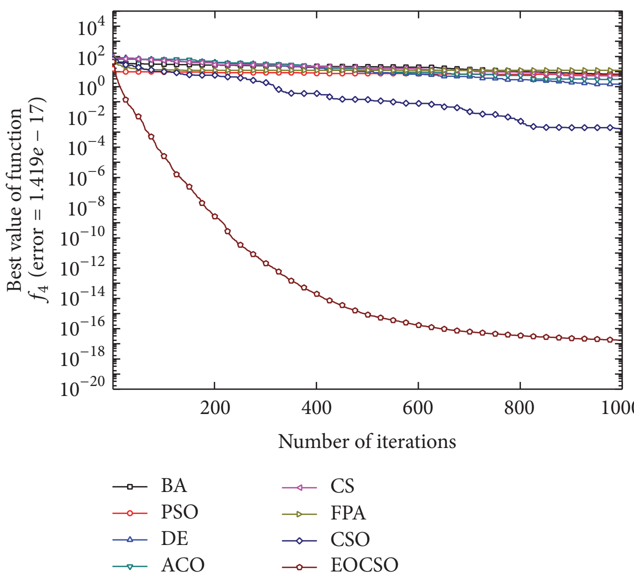


Figure 10: Convergence rates for $f_4(x)$.

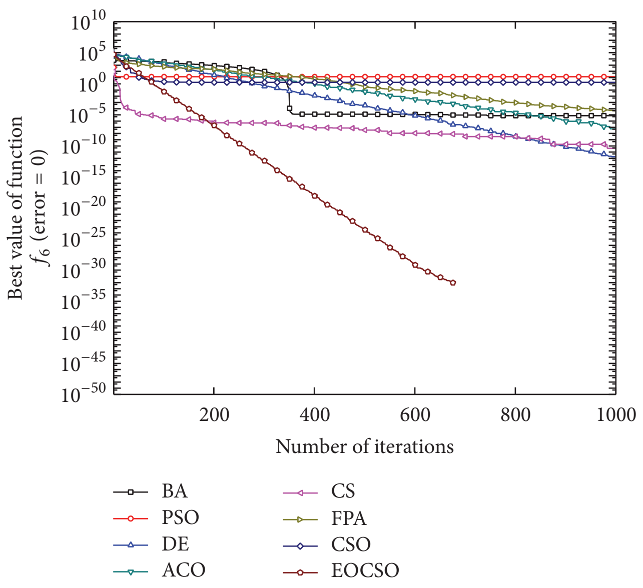


Figure 12: Convergence rates for $f_6(x)$.

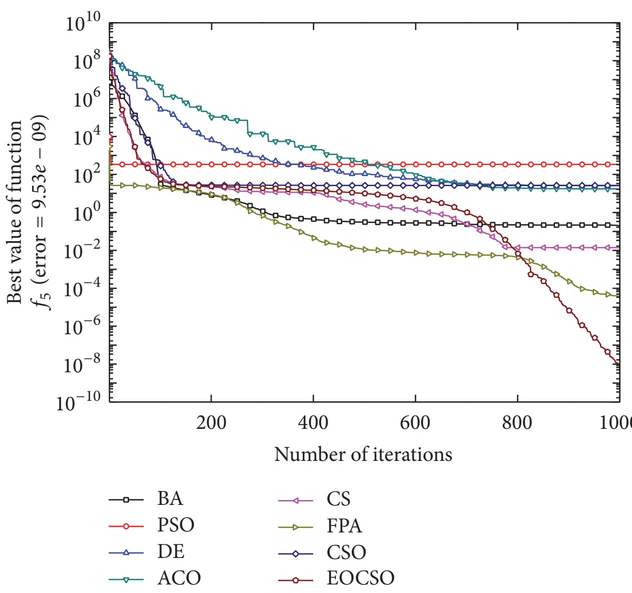


Figure 11: Convergence rates for $f_5(x)$.

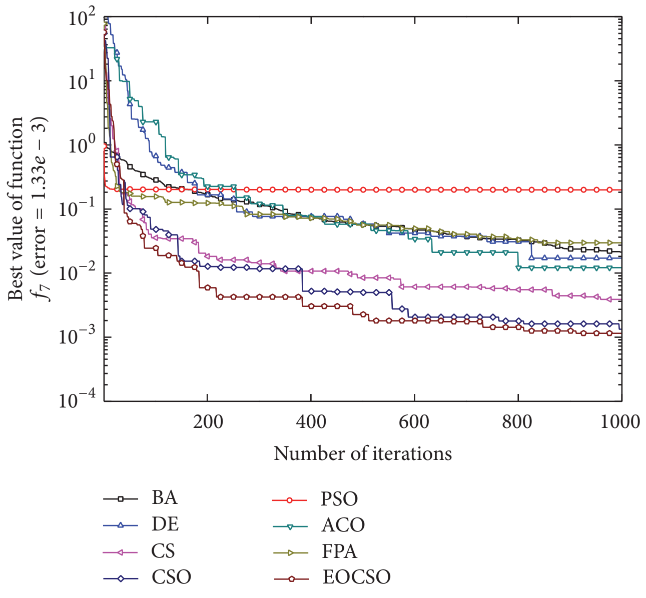


Figure 13: Convergence rates for $f_7(x)$.

of algorithms. From Table 11, the optimal value, the average value, and the worst value of EOCSO are better than those of the other 4 algorithms. In terms of stability, the standard deviation of EOCSO is smaller than those of MBA and HEAA algorithms by 6 and 4 orders of magnitude, respectively.

4.7.2. Pressure Vessel Design Problem. Another benchmark structural optimization problem is the pressure vessel design problem proposed by Kannan and Kramer. Figure 26 shows a pressure vessel design problem which has four variables (x_1, x_2, x_3, x_4) and four nonlinear inequality constraints

TABLE 12: Comparison of the best solutions for pressure vessel design problem by different algorithms.

Algorithm	x_1	x_2	x_3	x_4	$g_1(x)$	$g_2(x)$	$g_3(x)$	$g_4(x)$	$f(x)$
EOCSO	0.778169	0.384649	40.31962	200	$-7.20E-11$	$-1.12E-09$	$-4.29E-04$	-40	5885.3328
CS [7]	0.8125	0.4375	42.098446	176.636596	—	—	—	—	6059.714
MBA [32]	0.7802	0.3856	40.4292	198.4964	0	0	-86.3645	-41.5035	5889.3216
HCS-LSAL [38]	0.8125	0.4375	42.09844	176.6366	$-2.01E-09$	-0.0331588	-0.002495	-63.3634	6059.7143
NM-PSO [39]	0.8036	0.3972	41.6392	182.412	$3.65E-05$	$3.79E-05$	-1.5914	-57.5879	5930.3137
CSA [41]	0.8125	0.4375	42.098445	176.636598	$-4.02E+09$	-0.0358808	$-7.12E+04$	-63.3634	6059.7144

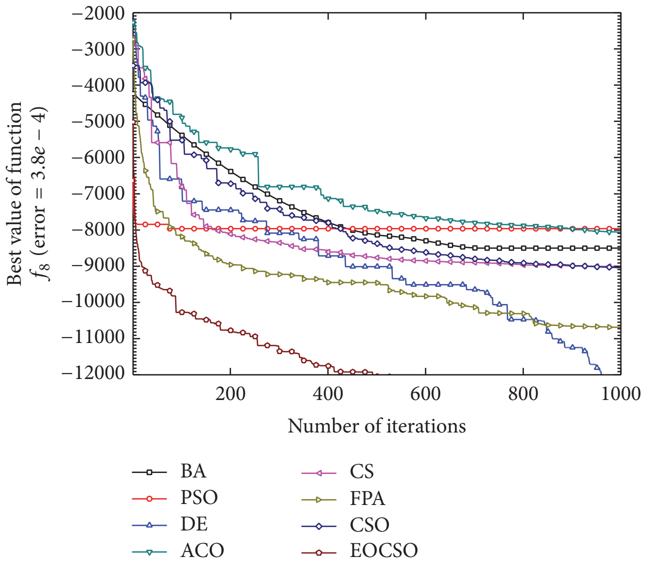


FIGURE 14: Convergence rates for $f_8(x)$.

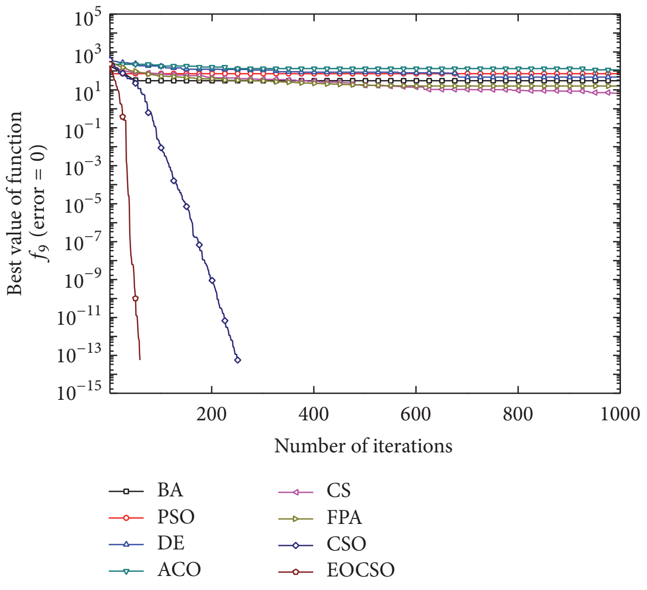


FIGURE 15: Convergence rates for $f_9(x)$.

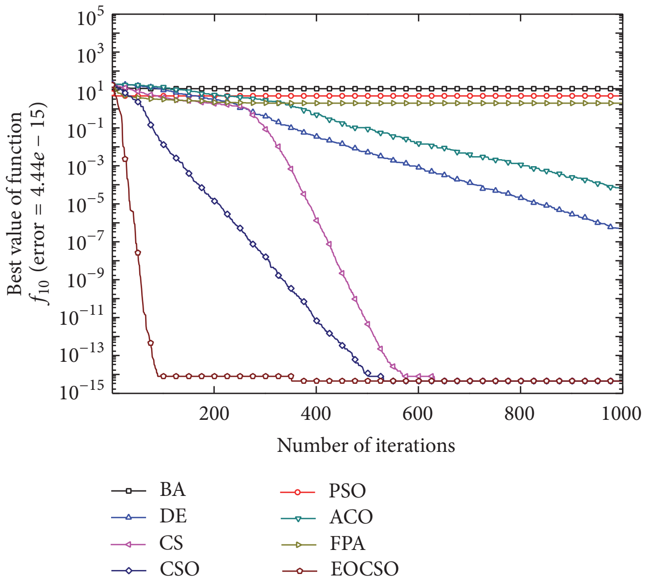


FIGURE 16: Convergence rates for $f_{10}(x)$.

(g_1, g_2, g_3, g_4). The objective function of the problem can be expressed as follows:

$$\begin{aligned} \min \quad & f(x) \\ & = 0.6224x_1x_3x_4 + 1.7881x_2x_3^2 + 3.1661x_1^2x_4 \\ & \quad + 19.84x_1^2x_3 \\ \text{s.t.} \quad & g_1(x) = -x_1 + 0.0193x_3 \le 0 \\ & g_2(x) = -x_2 + 0.0095x_3 \le 0 \\ & g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 129000 \le 0 \\ & g_4(x) = x_4 - 240 \le 0, \end{aligned} \tag{14}$$

where $0 \le x_1, x_2 \le 100$ and $10 \le x_3, x_4 \le 200$. The approaches have previously been applied to solve this problem including many different numerical optimization techniques, such as the new modification approach on bat algorithm (EBA) [35], the cuckoo search (CS) algorithm [7], interior search algorithm (ISA) [36], the mine blast algorithm (MBA) [32], the improved ant colony optimization (IACO) [37], an effective hybrid cuckoo search algorithm for constrained global optimization (HCS-LSAL) [38], the hybrid Nelder-Mead simplex search and particle swarm optimization (NM-PSO) [39], an improved accelerated PSO algorithm [40], and crow search algorithm (CSA) [41].

The best solutions obtained by the various methods are reported in Table 12. Table 13 shows the statistical results. It

Table 13: Statistical results of different approaches for pressure vessel design problem.

Algorithm	Best	Mean	Worst	Std.
EOCSO	5885.3328	5885.3328	5885.33279	$5.14E-06$
EBA [35]	6059.71	6173.67	6370.77	142.33
CS [7]	6059.714	6447.736	6495.347	502.693
ISA [36]	6059.71	6410.08	7332.84	384.6
MBA [32]	5889.32	6200.64	6392.5	160.34
IACO [37]	6059.73	6081.78	6150.13	67.2418
HCS-LSAL [38]	5930.3137	5946.7901	5960.0557	9.1614
NM-PSO [39]	6059.7143	6087.3225	6137.4069	$2.21E-02$
IAPSO [40]	6059.7143	6068.7539	6090.5314	14.0057
CSA [41]	6059.714363	6342.499106	7332.841621	384.9454163

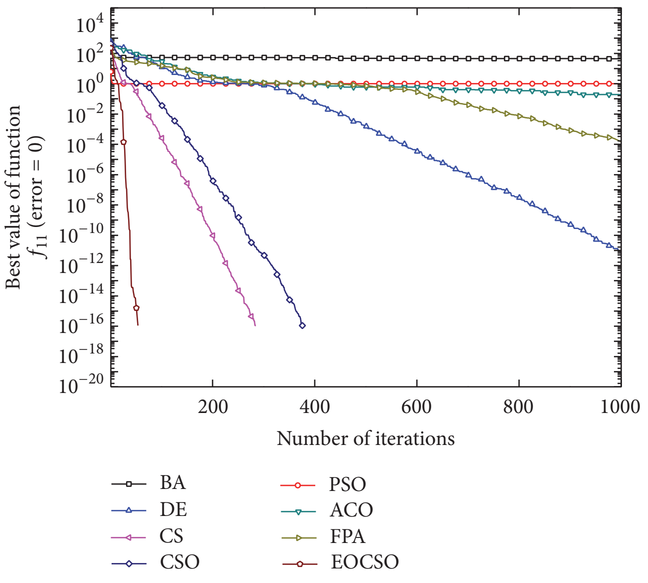


Figure 17: Convergence rates for $f_{11}(x)$.

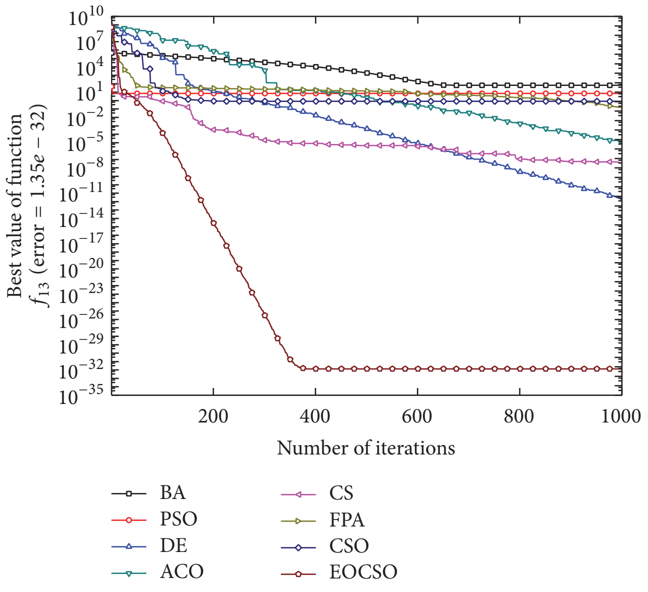


Figure 19: Convergence rates for $f_{13}(x)$.

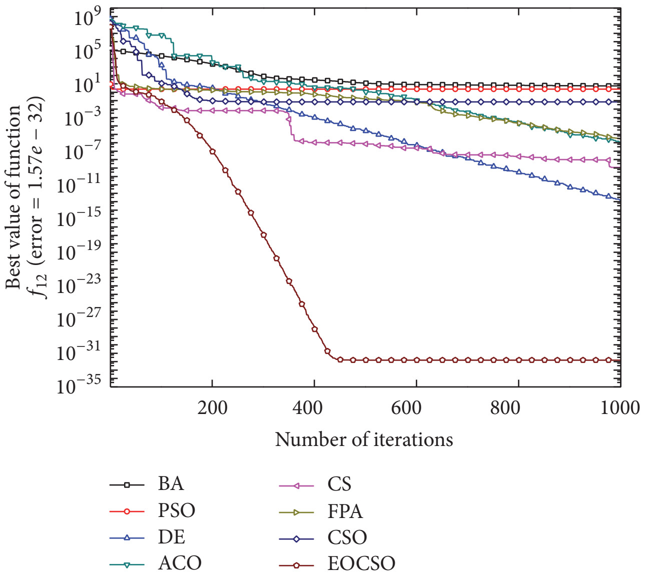


Figure 18: Convergence rates for $f_{12}(x)$.

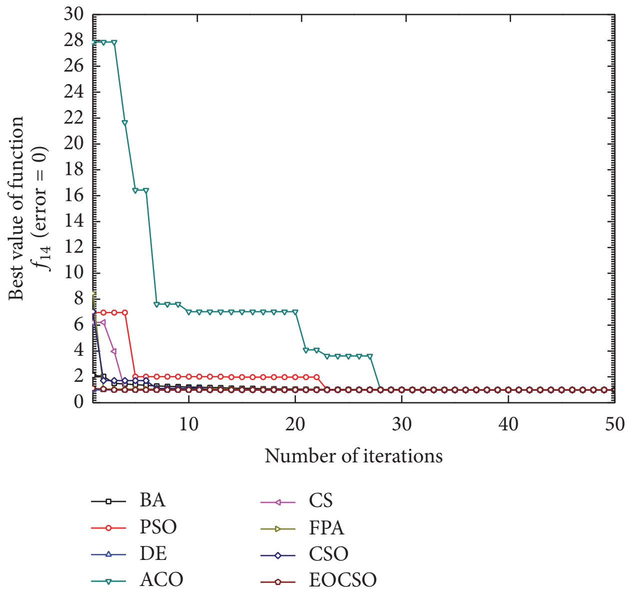


Figure 20: Convergence rates for $f_{14}(x)$.

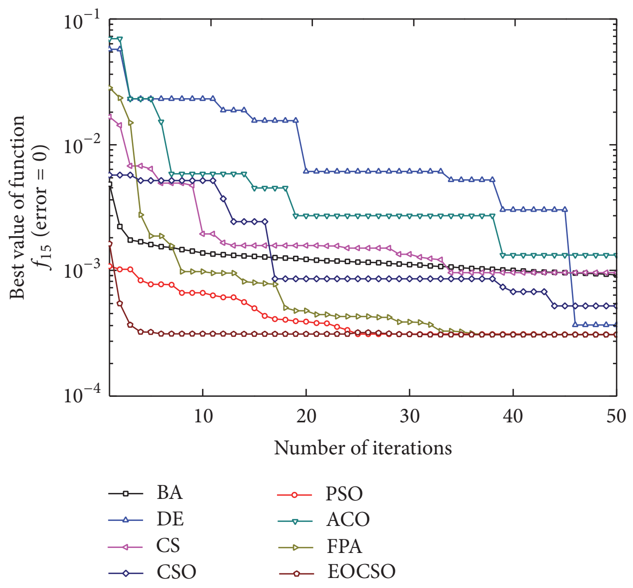


FIGURE 21: Convergence rates for $f_{15}(x)$.

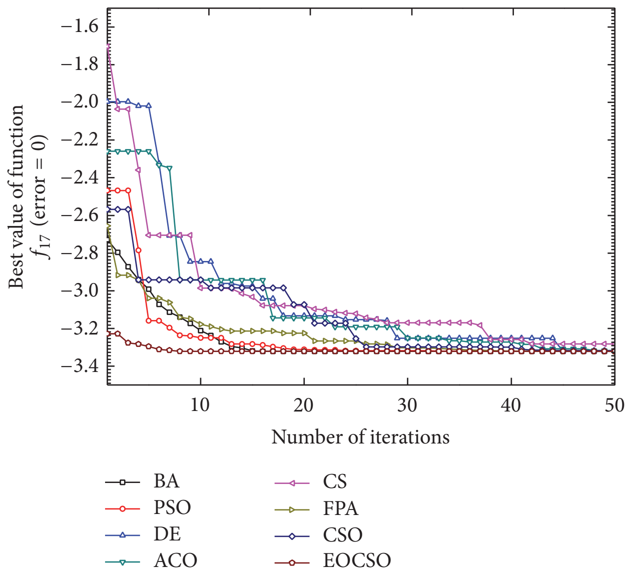


FIGURE 23: Convergence rates for $f_{17}(x)$.

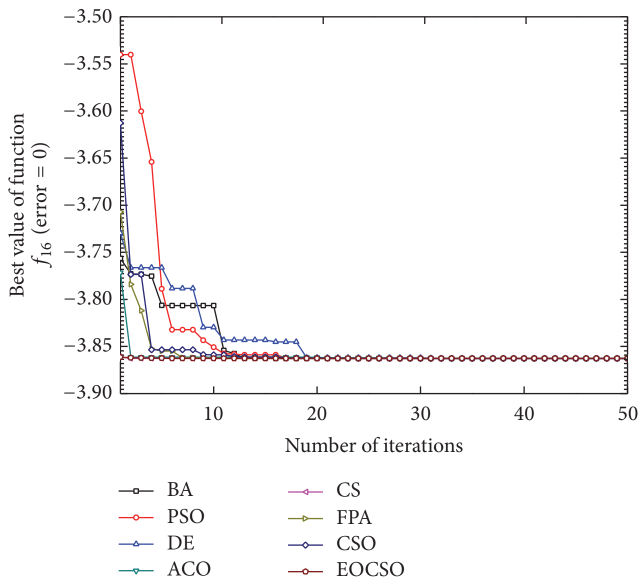


FIGURE 22: Convergence rates for $f_{16}(x)$.

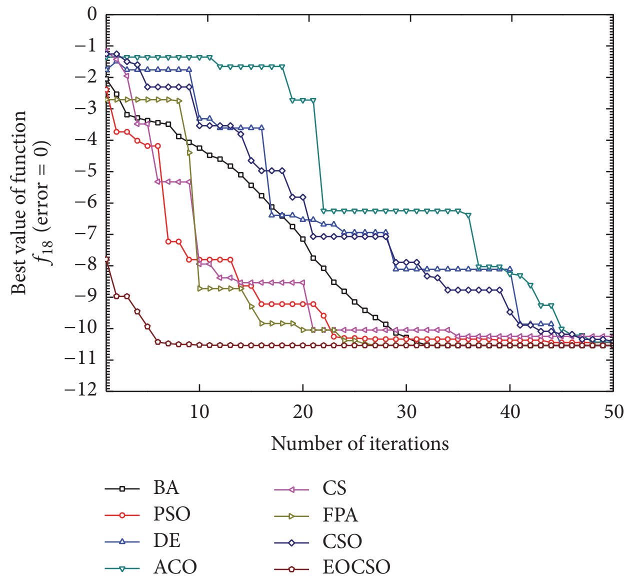


FIGURE 24: Convergence rates for $f_{18}(x)$.

is shown in Tables 12 and 13 that EOCSO performance for the pressure vessel design problem surpassed the other 10 methods in terms of the minimum obtained value, solution average, and the standard deviation.

5. Conclusion

In this paper, an improved chicken swarm algorithm based on elite opposition-based learning is proposed to overcome the disadvantages of the deficiencies of lack of population diversity, being easy to stick to "premature," and low searching precision in later stage of evolution in chicken swarm algorithm. Search mode of dynamic adaptive t distribution is applied for cock swarm to balance the global development

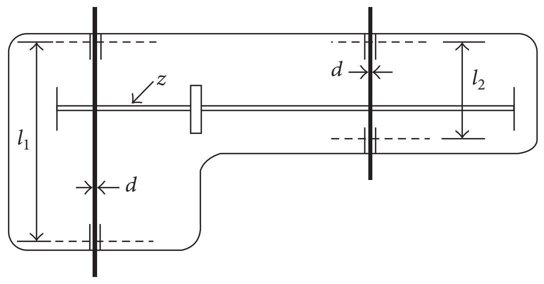


FIGURE 25: Speed reducer design problem.

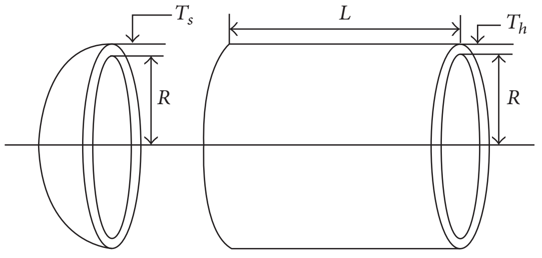


Figure 26: The pressure vessel design problem.

ability and local exploitation ability of the algorithm. Search mode of elite opposition-based learning is used to enrich population diversity for hen swarm. Greedy dimension-by-dimension searching is used for individual of optimal chicken swarm to do local search, which improves the search precision and convergence rate of the algorithm. Numerical experiments of 18 standard test functions and 2 engineering structure optimization problems are conducted to verify the availability and feasibility of the algorithm. The search precision, convergence rate, and robustness are all better than those in other typical intelligent algorithms.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is financially supported by the Natural Science Foundation of Guangxi Province (Grant no. 2014GXNSFBA118283) and the Higher School Scientific Research Project of Guangxi Province (Grant no. 2013YB247).

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[2] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[3] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[4] A. R. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecological Informatics*, vol. 1, no. 4, pp. 355–366, 2006.

[5] X. S. Yang, "Firefly algorithm," in *Engineering Optimization*, pp. 221–230, John Wiley & Sons, Hoboken, NJ, USA, 2010.

[6] X. S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications: 5th International Symposium, SAGA 2009, Sapporo, Japan, October 26–28, 2009. Proceedings*, vol. 5792 of *Lecture Notes in Computer Science*, pp. 169–178, Springer, Berlin, Germany, 2009.

[7] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, no. 1, pp. 17–35, 2013.

[8] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[9] X.-L. Li, Z.-J. Shao, and J.-X. Qian, "Optimizing method based on autonomous animats: fish-swarm Algorithm," *System Engineering Theory and Practice*, vol. 22, no. 11, pp. 32–38, 2002.

[10] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pp. 65–74, Springer, Berlin, Germany, 2010.

[11] X.-S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.

[12] R. Zhao and W. Tang, "Monkey algorithm for global numerical optimization," *Journal of Uncertain Systems*, vol. 2, no. 3, pp. 165–176, 2008.

[13] X. Yang, "Flower pollination algorithm for global optimization," in *Unconventional Computation and Natural Computation*, vol. 7445 of *Lecture Notes in Computer Science*, pp. 240–249, Springer, Berlin, Germany, 2012.

[14] X. Meng, Y. Liu, X. Gao et al., "A new bio-inspired algorithm: chicken swarm optimization," in *Advances in Swarm Intelligence*, vol. 8794 of *Lecture Notes in Computer Science*, pp. 86–94, Springer, 2014.

[15] Y. Chen, P. He, and Y. Zhang, "Combining penalty function with modified chicken swarm optimization for constrained optimization," in *Proceedings of the 1st International Conference on Information Sciences, Machinery, Materials and Energy*, pp. 1899–1907, Atlantis Press, Chongqing, China, April 2015.

[16] S. Liang, T. Feng, and G. Sun, "Sidelobe-level suppression for linear and circular antenna arrays via the cuckoo search–chicken swarm optimisation algorithm," *IET Microwaves, Antennas & Propagation*, vol. 11, no. 2, pp. 209–218, 2017.

[17] Y. L. Chen, P. L. He, and Y. H. Zhang, "Combining penalty function with modified chicken swarm optimization for constrained optimization," *Advances in Intelligent Systems Research*, vol. 126, pp. 1899–1907, 2015.

[18] F.-J. Zhou, X.-J. Wang, and M. Zhang, "Evolutionary programming using mutations based on the t probability distribution," *Acta Electronica Sinica*, vol. 36, no. 4, pp. 667–671, 2008.

[19] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA '05) and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC '05)*, pp. 695–701, November 2005.

[20] R. S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[21] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Information Sciences*, vol. 181, no. 20, pp. 4699–4714, 2011.

[22] X.-W. Xia, J.-N. Liu, K.-F. Gao, Y. Li, and H. Zeng, "Particle swarm optimization algorithm with reverse-learning and local-learning behavior," *Chinese Journal of Computers*, vol. 38, no. 7, pp. 1397–1407, 2015.

[23] Y. Zhong, X. Liu, L. Wang, and C. Wang, "Particle swarm optimisation algorithm with iterative improvement strategy for multi-dimensional function optimisation problems," *International Journal of Innovative Computing and Applications*, vol. 4, no. 3-4, pp. 223–232, 2012.

[24] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition versus randomness in soft computing techniques," *Applied Soft Computing Journal*, vol. 8, no. 2, pp. 906–918, 2008.

[25] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[26] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[27] L. Li, Y. Zhou, and J. Xie, "A free search krill herd algorithm for functions optimization," *Mathematical Problems in Engineering*, vol. 2014, Article ID 936374, 21 pages, 2014.

[28] X. Li, J. Zhang, and M. Yin, "Animal migration optimization: an optimization algorithm inspired by animal migration behavior," *Neural Computing and Applications*, vol. 24, no. 7-8, pp. 1867–1877, 2014.

[29] P. Civicioglu and E. Besdok, "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 315–346, 2013.

[30] E. Nabil, "A modified flower pollination algorithm for global optimization," *Expert Systems with Applications*, vol. 57, pp. 192–203, 2016.

[31] H. Liu, Z. Cai, and Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," *Applied Soft Computing*, vol. 10, no. 2, pp. 629–640, 2010.

[32] A. Sadollah, A. Bahreininejad, H. Eskandar, and M. Hamdi, "Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems," *Applied Soft Computing*, vol. 13, no. 5, pp. 2592–2612, 2013.

[33] Y. Wang, Z. Cai, Y. Zhou, and Z. Fan, "Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique," *Structural and Multidisciplinary Optimization*, vol. 37, no. 4, pp. 395–413, 2009.

[34] K. Deep and K. N. Das, "A novel hybrid genetic algorithm for constrained optimization," *International Journal of System Assurance Engineering and Management*, vol. 4, no. 1, pp. 86–93, 2013.

[35] S. Yılmaz and E. U. Küçüksille, "A new modification approach on bat algorithm for solving optimization problems," *Applied Soft Computing*, vol. 28, pp. 259–275, 2015.

[36] A. H. Gandomi, "Interior search algorithm (ISA): a novel approach for global optimization," *ISA Transactions*, vol. 53, no. 4, pp. 1168–1183, 2014.

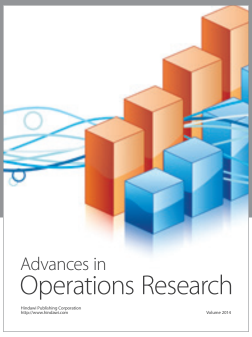
[37] A. Kaveh and S. Talatahari, "An improved ant colony optimization for constrained engineering design problems," *Engineering Computations*, vol. 27, no. 1, pp. 155–182, 2010.

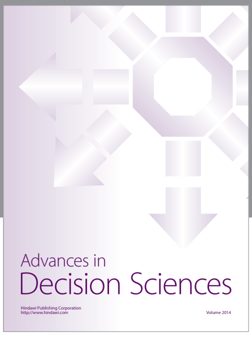
[38] W. Long, X. Liang, Y. Huang, and Y. Chen, "An effective hybrid cuckoo search algorithm for constrained global optimization," *Neural Computing and Applications*, vol. 25, no. 3-4, pp. 911–926, 2014.

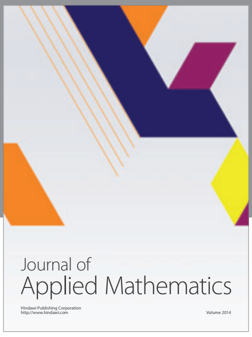
[39] E. Zahara and Y.-T. Kao, "Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3880–3886, 2009.

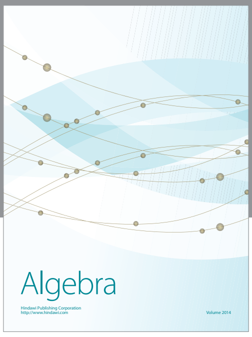
[40] N. Ben Guedria, "Improved accelerated PSO algorithm for mechanical engineering optimization problems," *Applied Soft Computing Journal*, vol. 40, pp. 455–467, 2016.

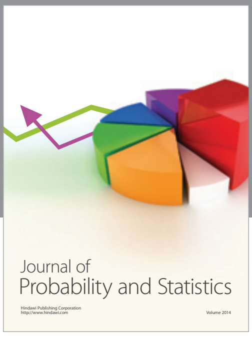
[41] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm," *Computers & Structures*, vol. 169, pp. 1–12, 2016.


Advances in Operations Research

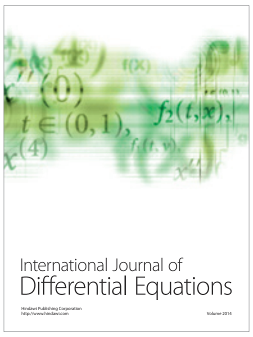

Advances in Decision Sciences


Journal of Applied Mathematics

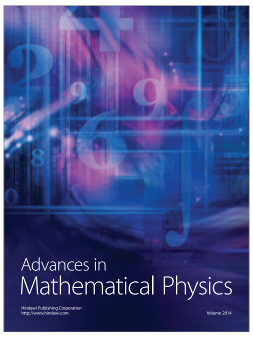

Algebra

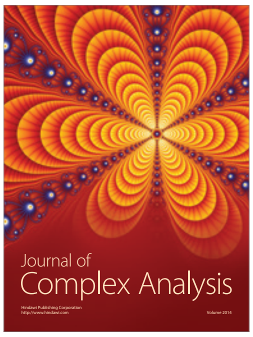

Journal of Probability and Statistics

The Scientific World Journal

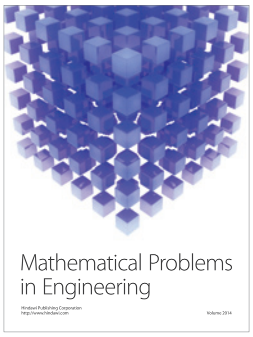

International Journal of Differential Equations

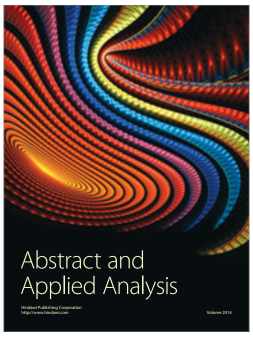
International Journal of Combinatorics

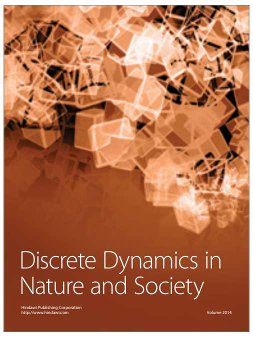

Advances in Mathematical Physics

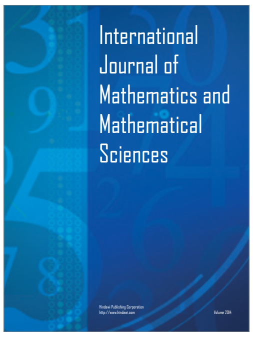

Journal of Complex Analysis

Journal of Mathematics


Mathematical Problems in Engineering

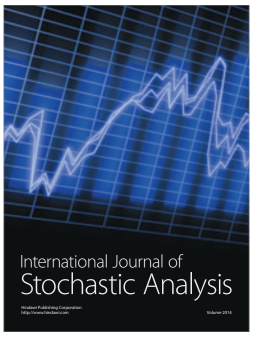

Abstract and Applied Analysis


Discrete Dynamics in Nature and Society


International Journal of Mathematics and Mathematical Sciences

Journal of Discrete Mathematics

Journal of Function Spaces


International Journal of Stochastic Analysis

Journal of Optimization