Neurocomputing 188 (2016) 294–310

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Elite opposition-based flower pollination algorithm

Yongquan Zhou [a,b,*], Rui Wang [a], Qifang Luo [a]

[a] College of Information Science and Engineering, Guangxi University for Nationalities, Nanning 530006, China
[b] Key Laboratory of Guangxi High Schools Complex System and Intelligent Computing, Nanning 530006, China

ARTICLE INFO

Article history:
Received 27 October 2014
Received in revised form
2 January 2015
Accepted 9 January 2015
Available online 17 December 2015

Keywords:
Flower pollination algorithm
Elite opposition-based learning
Metaheuristic optimization algorithm
Function optimization
Structure engineering design

ABSTRACT

Flower pollination algorithm (FPA) is a novel metaheuristic optimization algorithm with quick convergence, but its population diversity and convergence precision can be limited in some applications. In order to enhance its exploitation and exploration abilities, in this paper, an elite opposition-based flower pollination algorithm (EOFPA) has been applied to functions optimization and structure engineering design problems. The improvement involves two major optimization strategies. Global elite opposition-based learning enhances the diversity of the population, and the local self-adaptive greedy strategy enhances its exploitation ability. An elite opposition-based flower pollination algorithm is validated by 18 benchmark functions and two structure engineering design problems. The results show that the proposed algorithm is able to obtained accurate solution, and it also has a fast convergence speed and a high degree of stability.

1. Introduction

Swarm intelligence optimization algorithm originates from the simulation of various types of biological behavior in nature and has characteristics of simple operation, good optimization performance and strong robustness. Inspired by this idea, there are many bio-inspired swarm intelligent optimization algorithms are proposed, such as, ant colony optimization (ACO) [1], differential evolution (DE) [2], particle swarm optimization(PSO) [3], firefly algorithm (FA) [4], glowworm swarm optimization (GSO) [5], monkey search (MS) [6], harmony search (HS) [7], cuckoo search (CS) [8], bat algorithm (BA) [9], et al. Swarm intelligence optimization algorithm can solve problems which traditional methods cannot handle effectively and have shown excellent performance in many respects, and its application scope has been greatly expanded.

Flower pollination algorithm is proposed by Xin-She Yang in 2012 [10], it is a novel metaheuristic optimization algorithm by simulating flower pollination behavior. Flower pollination behavior stems from the purpose of reproduction. From the biological evolution point of view, the objective of flower pollination is the survival of the fittest and the optimal reproduction of plant species. All these factors and processes of flower pollination interact so as to achieve optimal reproduction of the flowering plants. Self-pollination and cross-pollination are two different ways of pollination [11]. Cross-pollination means pollination can occur from pollen of a flower of a different plant, and self-pollination is just the opposite. Biotic, cross-pollination can occur at long distance, the pollinators of these pollinations such as bees, bats,birds can fly a long distance, thus they can considered as the global pollination. And these pollinators can fly as Lévy flight behavior [12], with fly distance steps obey a Lévy distribution. Thus, this can inspire to design new optimization algorithm. Flower pollination algorithm is an optimization algorithm which simulates the flower pollination behavior above-mentioned, flower pollination algorithm can also be divided into global pollination process and local pollination process. And it has been extensively researched in last two years by scholars. Huang and Yu have proposed a novel alignment-free sequence comparison method based on the numbers of adjacent amino acids based on Normalized feature vectors [13]. Yang and He have used FPA to solve multi-objective engineer optimization problems in 2013 [14,55]; Marwa Sharawi has applied FPA for wireless sensor network lifetime global optimization in 2014 [15]; Osama Abdel-Raouf has used an improved FPA to solve Sudoku Puzzles in 2014 [16]; FPA has also been applied to solve large integer programming problems by Ibrahim EI-henawy in 2014 [17]. In this paper, an elite opposition-based flower pollination algorithm (EOFPA) has been applied to functions optimization and structure engineering design problems. The improvement involves two major optimization strategies. Global elite opposition-based learning enhances the diversity of the population, and the local self-adaptive greedy strategy enhances its exploitation ability. An elite opposition-based flower pollination algorithm is validated by 18 benchmark functions and two structure engineering design problems. The results show that the proposed algorithm is able to obtained accurate solution,

* Corresponding author.
E-mail address: yongquanzhou@126.com (Y. Zhou).

http://dx.doi.org/10.1016/j.neucom.2015.01.110
0925-2312/© 2015 Elsevier B.V. All rights reserved.

and it also has a fast convergence speed and a high calculation precision.

The remainder of the paper is organized as follows: Section 2 briefly introduces the original flower pollination algorithm; this is followed in Section 3 by new elite opposition-based flower pollination algorithm (EOFPA); simulation experiments and results analysis are described in Section 4. Finally, conclusion and future works can be found and discussed in Section 5.

2. Flower pollination algorithm (FPA)

Flower pollination algorithm (FPA) is inspired by the flow pollination process of flowering plants are the following rules [10,18]:

Rule 1**:** biotic and cross-pollination can be considered as a process of global pollination process, and pollen-carrying pollinators move in a way that obeys Lévy flights.
Rule 2: for local pollination, a biotic and self-pollination are used.
Rule 3: pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved.
Rule 4**:** the interaction or switching of local pollination and global pollination can be controlled by a switch probability, with a slight bias toward local pollination.

In order to formulate updating formulas, we have to convert the aforementioned rules into updating equations. For example, in the global pollination step, flower pollen gametes are carried by pollinators such as insects, and pollen can travel over a long distance because insects can often fly and move in a much longer range. Therefore, Rule 1 and flower constancy can be represented mathematically as:

$$x_i^{t+1} = x_i^t + \gamma L(\lambda)(x_i^t - B) \tag{1}$$

where x_i^t is pollen i or solution vector x_i at iteration t, and B is the current best solution found among all solutions at the current generation/iteration. Here γ is a scaling factor to control the step size. In addition, $L(\lambda)$ is the parameter that corresponds to the strength of the pollination, which essentially is also the step size. Since insects may move over a long distance with various distance steps, we can use a Lévy flight to imitate this characteristic efficiently. That is, we draw $L > 0$ from a Lévy distribution:

$$L \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{S^{1+\lambda}}, (S >> S_0 > 0) \tag{2}$$

Here, $\Gamma(\lambda)$ is the standard gamma function, and this distribution is valid for large steps $s > 0$. Then, to model the local pollination, both Rule 2 and Rule 3 can be represented as

$$x_i^{t+1} = x_i^t + U(x_j^t - x_k^t) \tag{3}$$

where x_j^t and x_k^t are pollen from different flowers of the same plant species. This essentially imitates the flower constancy in a limited neighborhood. Mathematically, if x_j^t and x_k^t comes from the same species or selected from the same population, this equivalently becomes a local random walk if we draw U from a uniform distribution in [0, 1]. Though flower pollination activities can occur at all scales, both local and global, adjacent flower patches or flowers in the not-so-far-away neighborhood are more likely to be pollinated by local flower pollen than those faraway. In order to imitate this, we can effectively use the switch probability like in Rule 4 or the proximity probability p to switch between common global pollination to intensive local pollination. To begin with, we can use a naïve value of $p = 0.5$ as an initially value. A preliminary parametric showed that $p = 0.8$ might work better for most applications [10]. Specific implementation steps of the Standard Flower Pollination Algorithm (FPA) can be summarized in the pseudo code shown in Algorithm 1.

Algorithm 1. Flower pollination algorithm

```
Define objective function $f(x), x = (x_1, x_2, ..., x_d)$
Initialize a population of $n$ flowers/pollen gametes with random solutions;
Find the best solution $B$ in the initial population;
Define a switch probability $p \in (0, 1)$;
Define a stopping criterion (either a fixed number of generations/iterations or accuracy)
while ($t < MaxGeneration$)
    for $i = 1 : n$ (all $n$ flowers in the population)
        if $rand < p$
            Draw a ($d$-dimensional) step vector L which obeys a Lévy distribution;
            Global pollination via Eq. (1) and get new solution $x_i$;
        else
            Draw $U$ form a uniform distribution in (0,1);
            Do local pollination via Eq. (3) and get new solution $x_i$ ;
        endif
        Evaluate the new solutions;
        If new solutions are better, update them in the population;
    endfor
        Find the current best solution $B$;
end while
Output the best solution found
```

3. Elite opposition-based flower pollination algorithm (EOFPA)

Flower pollination algorithm can easily solve low-dimensional unimodal optimization problems. Whereas when handling the high-dimensional and multi-modal optimization problems, we can clearly discover that the solutions obtained by FPA are not good enough. In order to enhance the global searching and local searching abilities, we append three optimization strategies to basic flower pollination algorithm (FPA). There are global elite opposition-based learning strategy (GEOLS), local self-adaptive greedy strategy (LSGS) and dynamic switching probability strategy (DSPS).

3.1. Global elite opposition-based learning strategy (GEOLS)

Basic flower pollination algorithm use Lévy flight in global search process. It is simulated by Lévy distribution. As we know that it is a stochastic process, the probability of getting a good solution is relatively low. For increasing the probability of obtained a better solution to the problem in global search process and expand the searching space, this strategy is applied to the proposed EOFPA. In essence, it is a greedy strategy.

Elite opposition-based Learning is a new technique in the field of intelligence computation. Its main ideology is: for a feasible solution, calculate and evaluate the opposite solution at the same time, and choose the better one as the individual of next generation. In this paper, individual with the best fitness value in the population is viewed as the elite individual. For explaining the definition of elite opposition-based solution, an example should be exhibited. If we suppose that the elite individual of the population is $X_e = (x_{e,1} x_{e,2}, ..., x_{e,D})$. For the individual $X_i = (x_{i,1} x_{i,2}, ..., x_{i,D})$, the elite opposition-based solution of X_i can be defined as $X_i' =$

$(x'_{i,1}x'_{i,2},...,x'_{i,D})$. And it can be obtained by following equation:

$$x'_{i,j} = k\bullet(da_j + db_j) - x_{e,j}, i = 1,2,...,n; j = 1,2,...,D. \tag{4}$$

where n is the population size, D is the dimension of X, $k \in U(0,1)$, $(da_j db_j)$ is the dynamic bound of jth decision variable. da_j, db_j can be obtained by following equation:

$$da_j = \min(x_{i,j}), db_j = \max(x_{i,j}) \tag{5}$$

As we know that the shrink of searching space may cause algorithm stuck in local minimal. Thus, in this proposed algorithm, we will update da_j and db_j every 50 generations. Dynamic bound is good at restoring searching experience. But it can make $x'_{i,j}$ jump out of $(da_j db_j)$, if that happened, equation below should be used to reset $x'_{i,j}$.

$$x'_{i,j} = rand(da_j, db_j), \quad \text{if} \quad x'_{i,j} < da_j \quad \text{or} \quad db_j > x'_{i,j} \tag{6}$$

In global searching process, this strategy expands the searching space of algorithm and it can strengthen the diversity of the population, thus the proposed global searching ability can be enhanced by this optimization strategy.

3.2. Local self-adaptive greedy strategy (LSGS)

Standard Flower Pollination Algorithm use DE to conduct local search. And it is known to us that the searching ability of standard DE is not good enough to solve high-dimensional optimization problems. For improving the local search, this local self-adaptive greedy strategy is added to the proposed algorithm. In local searching process, if $X = (x_{i,1}x_{i,2},...,x_{i,D})$ is the ith individual in the population, we can find a greedy solution (X') in the neighbor of it by following equation:

$$x'_{i,j} = x_{i,j} + \varphi_{i,j}C_{i,j}x_{i,j} \tag{7}$$

Table 1
Benchmark test functions.

Benchmark test functions	Dimension	Range	Optimum	Itera- tions
$f_{01}(x) = \sum_{i=1}^{n} x_i^2$	30	[−100,100]	0	1500
$f_{02}(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$	30	[−10,10]	0	2000
$f_{03}(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_i\right)^2$	30	[−100,100]	0	5000
$f_{04}(x) = \max_i\{\lvert x_i\rvert, 1 \le i \le D\}$	30	[−100,100]	0	5000
$f_{05}(x) = \sum_{i=1}^{D-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[−30,30]	0	5000
$f_{06}(x) = \sum_{i=1}^{n} \lfloor x_i + 0.5 \rfloor$	30	[−100,100]	0	1500
$f_{07}(x) = \sum_{i=1}^{D}[x_i^2 - 10\ \cos(2\pi x_i + 10]$	30	[−5.12,5.12]	0	3000
$f_{08}(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp\left(\frac{1}{D}\sum_{I=1}^{D}\cos\ 2\pi x_i\right) + 20 + e$	30	[−32,32]	0	1500
$f_{09}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{2}}) + 1$	30	[−600,600]	0	2000
$f_{10}(x) = \frac{\pi}{D}\left\{\begin{array}{l} 10\sin^2(\pi y_1) \\ + \sum_{i=1}^{D-1}(y-1)^2[1 + 10\sin^2(\pi y_1)] + (y_D - 1)^2 \end{array}\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $y_1 = 1 + \frac{x_i + 1}{4}$; $u(x_i, a, k, m) = \begin{cases} k\ (x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k\ (-x_i - a)^m, x_i < a \end{cases}$	30	[−50,50]	0	1500
$f_{11}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	[−5,5]	0.0003075	400
$f_{12}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[−5,5]	−1.0316285	100
$f_{13}(x) + \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos\ x_1 + 10$	2	[−5,10]	0.398	100
$f_{14}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times[30 + (2x_1 - 3x_2)^2\ (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[−5,5]	3	30
$f_{15}(x) = -\sum_{i=1}^{4} c_i\exp\left[\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right]$	3	[0,1]	−3.8628	100
$f_{16}(x) = -\sum_{i=1}^{5}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0,10]	−10.1532	100
$f_{17}(x) = -\sum_{i=1}^{7}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0,10]	−10.4029	100
$f_{18}(x) = -\sum_{i=1}^{10}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0,10]	−10.5364	100

Table 2
Sinulation results for test functions *fi*, $i = 1, 2, 3, 4, 5, 6$.

Benchmark functions	Methods	Results			
		Best	Mean	Worst	Std.
*f*1 ($D=30$)	DE	1.50710E-45	6.88386E-08	1.36624E-06	3.05381E-07
	CS	3.06E-06	6.67E-06	9.43E-06	1.75E-06
	PSO	5.43245E-16	1.09881E-05	8.6001E-05	2.88213E-05
	ABC	1.35672E-16	2.88952E-16	4.8451E-16	9.24626E-17
	FPA	260.8704	460.1893	695.6179	144.2958
	EOFPA	**0**	**0**	**0**	**0**
*f*2 ($D=30$)	DE	1.3138E-12	7.7508E-06	7.90987E-05	1.99829E-05
	CS	0.001115	0.002437	0.004629	0.00106
	PSO	8.75E-16	3.30E-06	2.65E-05	7.23E-06
	ABC	3.48052E-16	6.56687E-16	9.67058E-16	1.57079E-16
	FPA	1.927108	3.419715	7.835688	1.325707
	EOFPA	**0**	**0**	**0**	**0**
*f*3 ($D=30$)	DE	6.37486E-30	1.13403E-06	1.44558E-05	3.31491E-06
	CS	0.000825	0.001513	0.003018	0.000573
	PSO	0.01299	4.21307	28.32982	7.205002
	ABC	3831.116	5775.815	7684.872	1371.431
	FPA	1003.061	2371.397	4148.677	730.4888
	EOFPA	**0**	**0**	**0**	**0**
*f*4 ($D=30$)	DE	1.25E-10	0.420039	3.001909	0.796228
	CS	0.00103	0.042196	0.242109	0.056944
	PSO	0.000449	5.15502	100	22.32554
	ABC	10.19505	21.007	29.46092	5.285541
	FPA	13.6454	17.42714	26.01978	2.777382
	EOFPA	**0**	**0**	**0**	**0**
*f*5 ($D=30$)	DE	2.25E-05	0.881922	9.123986	2.131354
	CS	5.047136	8.771959	12.96618	1.856324
	PSO	0.09073	14.60663	185.3633	40.35036
	ABC	0.006156	0.089377	0.239864	0.064914
	FPA	75.80602	239.229	484.8164	108.4824
	EOFPA	**2.36E-14**	**1.91E-11**	**1.9E-10**	**4.85E-11**
*f*6 ($D=30$)	DE	0	13.35	256	57.14918
	CS	0	0	0	0
	PSO	0	1	7	1.622214
	ABC	0	0	0	0
	FPA	146	494.25	1471	282.9205
	EOFPA	**0**	**0**	**0**	**0**

Table 3
Sinulation results for test functions *fi*, $i=7, 8, 9, 10$.

Benchmark functions	Methods	Results			
		Best	Mean	Worst	Std.
*f*7 ($D=30$)	DE	1.005301	5.465793	13.47421	3.101458
	CS	37.77611	51.01831	63.75098	7.626058
	PSO	0	12.83494	32.83339	8.454177
	ABC	0	3.97904E-14	3.83693E-13	8.3832E-14
	FPA	94.28336	187.0876	231.9585	34.05495
	EOFPA	**0**	**0**	**0**	**0**
*f*8 ($D=30$)	DE	1.51E-14	5.523946	19.78685	7.378242
	CS	0.046082	0.651707	1.862639	0.566777
	PSO	9.53E-05	4.746604	19.86863	5.386355
	ABC	6.42292E-10	1.85671E-09	6.39743E-09	1.67887E-09
	FPA	4.120131	5.271212	6.584102	0.614404
	EOFPA	**8.88E-16**	**8.88E-16**	**8.88E-16**	**0**
*f*9 ($D=30$)	DE	0	0.015813	0.121311	0.038879
	CS	8.62E-09	4.68E-08	1.66E-07	4.37E-08
	PSO	5.76E-07	0.297691	1.250082	0.284139
	ABC	4.44089E-16	4.47791E-09	3.9685E-08	9.64744E-09
	FPA	3.889933	11.48897	20.02404	4.290201
	EOFPA	**0**	**0**	**0**	**0**
*f*10 ($D=30$)	DE	3.62E-29	0.031101	0.207338	0.075958
	CS	0.024484	0.638515	1.257338	0.33893
	PSO	0.04879	2.318881	5.413524	1.722417
	ABC	2.50144E-16	1.41355E-15	6.43332E-15	1.40579E-15
	FPA	33.14828	13173.05	69071.29	22204.86
	EOFPA	**3.31E-31**	**4.15E-29**	**2.06E-28**	**6.26E-29**

$$C_{i,j} = c_{min} + \exp\left[-c\left(\frac{iter}{Max\_iter}\right)^d\right](c_{max} - c_{min})$$

where $x'_{i,j}$ is the jth decision variable of greedy solution, $\varphi_{i,j}$ is a random number in $(-1,1)$. *iter* is the number of current iteration, *Max_iter* is the maximum iteration. c_{min}, c_{max}, c and $d$are related parameters. In this paper, we set $c=60, d=5, c_{min}=0.1, c_{max}=0.5$.

By using this strategy, the adjustment of greedy solution is bigger at the beginning of the iterations. It helps to expand the searching space, enhance the ability of jump out of the local minima. And the adjustment of greedy solution is smaller in the end, which helps to enhance the local search and speed up finding the best solution.

3.3. Dynamic switching probability strategy (DSPS)

In FPA, local search and global search is controlled by a switch probability $p \in [0,1]$, and it is a constant value. We suppose that a reasonable algorithm should do more global search at the beginning of searching process and global search should be less in the end. Thus, we apply dynamic switching probability strategy (DSPS) to adjust the proportion of two kinds of search. Switch probability p can alter according to following formula

$$p = 0.6 - 0.1 \times (Max\_iter - t)/Max\_iter \tag{8}$$

where *Max_iter* is the maximum iterations, and t is current iteration. Specific implementation steps of the elite opposition-based flower pollination algorithm (EOFPA) can be summarized in the pseudo code shown in Algorithm 2.

Algorithm 2. Elite opposition-based flower pollination algorithm

```
Define Objective function $f(x), x=(x_1, x_2, ..., x_d)$;
Initialize a population of $n$ flowers/pollen gametes with random solutions;
Find the best solution $B$ in the initial population;
Define a switch probability $p \in (0,1)$;
Define a stopping criterion (either a fixed number of generations/iterations or accuracy)
while ($t < MaxGeneration$)
    for i=1:$n$ (all $n$ flowers in the population)
        if $rand < p$
```

Table 4
Sinulation results for test functions *fi*, i =11, 12, 13, 14, 15, 16, 17, 18.

Benchmark functions	Methods	Results			
		Best	Mean	Worst	Std.
$f11$ (D=4)	DE	0.000424	0.001142	0.007075	0.001444
	CS	0.000661	0.000876	0.001032	0.000114
	PSO	0.000307	0.002477	0.020363	0.004262
	ABC	0.000797	0.001166	0.002117	0.000361
	FPA	0.000521	0.000744	0.000892	7.49E-05
	EOFPA	***0.000307***	***0.00054***	0.001594	0.000488
$f12$ (D=2)	DE	−1.03163	−1.03158	−1.03124	0.000102
	CS	−1.03163	−1.03163	−1.03163	6.29E-08
	PSO	−1.03163	−1.03163	−1.03163	1.08E-11
	ABC	−1.03163	−1.03163	−1.03163	8.65E-08
	FPA	−1.03163	−1.03159	−1.03131	7.96E-05
	EOFPA	**−1.03163**	**−1.03163**	**−1.03163**	**0**
$f13$ (D=2)	DE	0.39789	0.39803	0.398502	0.000177
	CS	0.397887	0.397891	0.397902	4.59E-06
	PSO	0.397887	0.397887	0.397887	8.57E-09
	ABC	0.397887	0.397902	0.397994	2.82E-05
	FPA	0.397888	0.3979	0.397963	1.8E-05
	EOFPA	**0.397887**	**0.397887**	**0.397887**	**0**
$f14$ (D=2)	DE	3.016266	14.28008	70.04086	15.70908
	CS	3.010708	3.609433	5.990295	0.671218
	PSO	3.000115	3.003209	3.013157	0.003618
	ABC	3.131013	12.54678	33.29449	9.773685
	FPA	3.070886	4.913745	14.63766	2.953379
	EOFPA	**3**	**3**	**3**	**2.65E-14**
$f15$ (D=3)	DE	−3.86278	−3.86277	−3.86274	1.38E-05
	CS	−3.86278	−3.86278	−3.86276	6.61E-06
	PSO	−3.84438	−3.31273	−2.15055	0.464373
	ABC	−3.86278	−3.86278	−3.86271	1.61E-05
	FPA	−3.86278	−3.86272	−3.86255	6.23E-05
	EOFPA	**−3.86278**	**−3.86278**	**−3.86278**	**1.84E-15**
$F16$ (D=4)	DE	−10.1446	−5.75769	−2.57559	3.684511
	CS	−10.1196	−9.80342	−9.29294	0.236496
	PSO	−1.99188	−0.58988	−0.16733	0.412888
	ABC	−10.1388	−9.93101	−9.28958	0.24291
	FPA	−9.84067	−6.90573	−4.5189	2.013148
	EOFPA	**−10.1532**	**−9.31182**	**−2.63047**	**2.267192**
$f17$ (D=4)	DE	−10.4024	−8.62774	−2.74173	2.760935
	CS	−10.3832	−9.97411	−9.06904	0.355134
	PSO	−1.52176	−0.58021	−0.30157	0.280967
	ABC	−10.3856	−9.68857	−3.7176	1.478628
	FPA	−10.0753	−7.67807	−3.72647	2.11409
	EOFPA	**−10.4029**	**−7.5398**	**−2.7659**	**3.64322**
$f18$ (D=4)	DE	−10.5353	−7.91452	−2.4148	3.446094
	CS	−10.4302	−9.72198	−8.49279	0.576931
	PSO	−4.51532	−0.98403	−0.32645	0.951639
	ABC	−10.5305	−9.66973	−6.05163	1.249731
	FPA	−10.455	−7.90547	−3.11414	2.518221
	EOFPA	**−10.5364**	**−8.38128**	**−2.42734**	**3.211434**

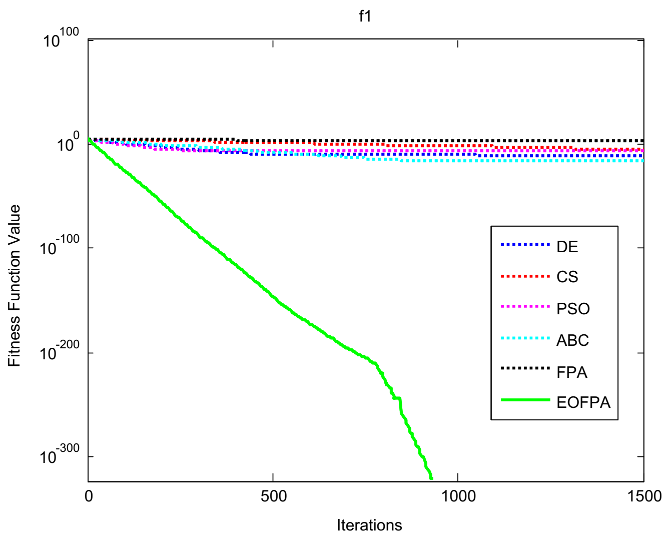

Fig. 1. D=30, evolution curves of fitness value for f_{01}.

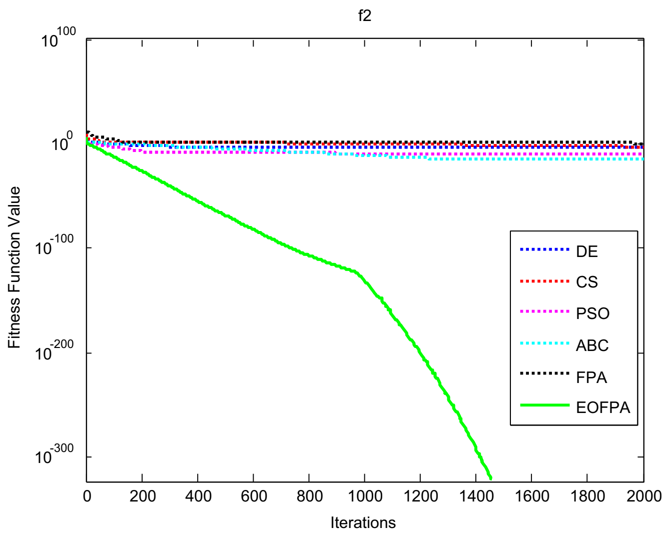

Fig. 2. D=30, evolution curves of fitness value for f_{02}.

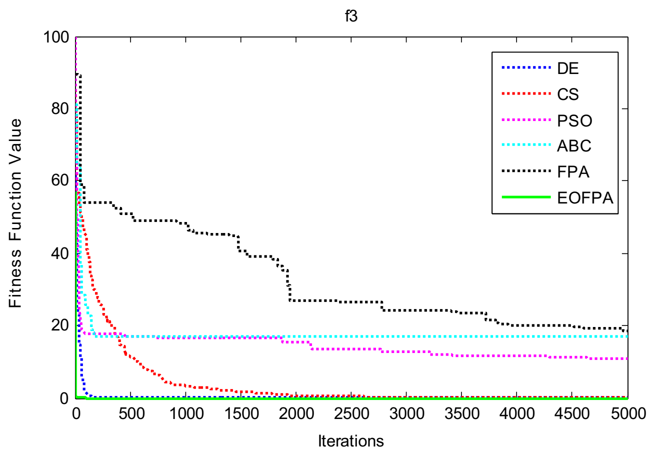

Fig. 3. $D=30$, evolution curves of fitness value for f_{03}.

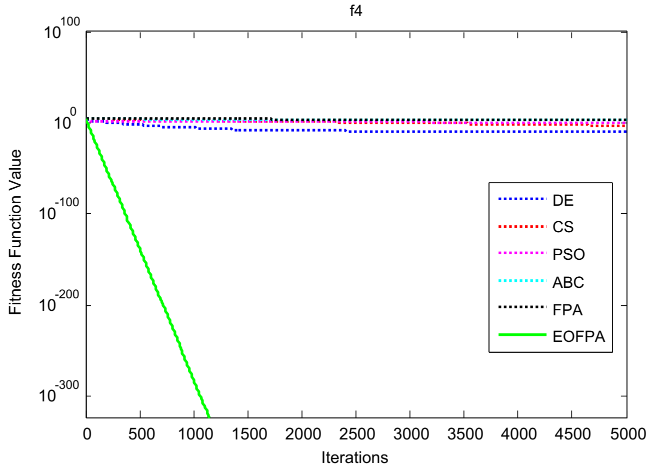

Fig. 4. $D=30$, evolution curves of fitness value for f_{04}.

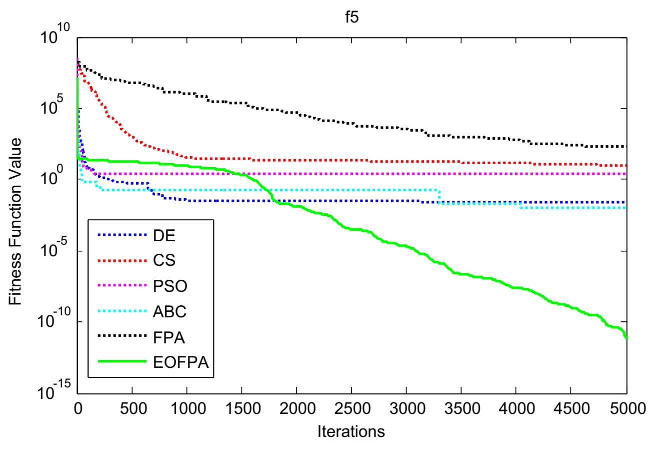

Fig. 5. $D=30$, evolution curves of fitness value for f_{05}.

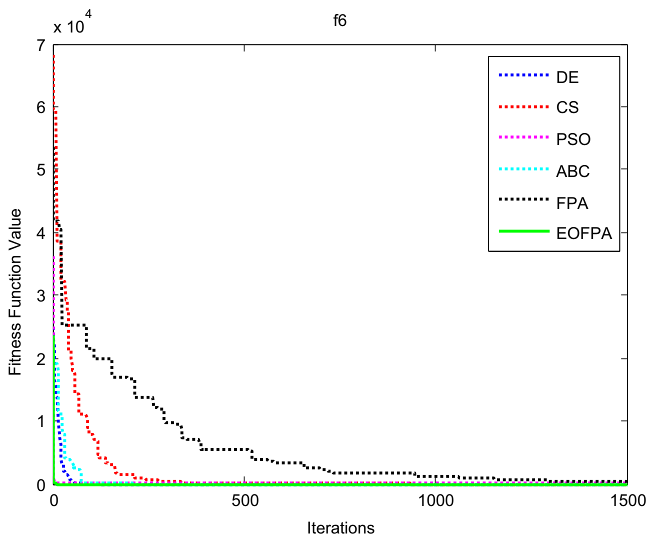

Fig. 6. $D=30$, evolution curves of fitness value for f_{06}.

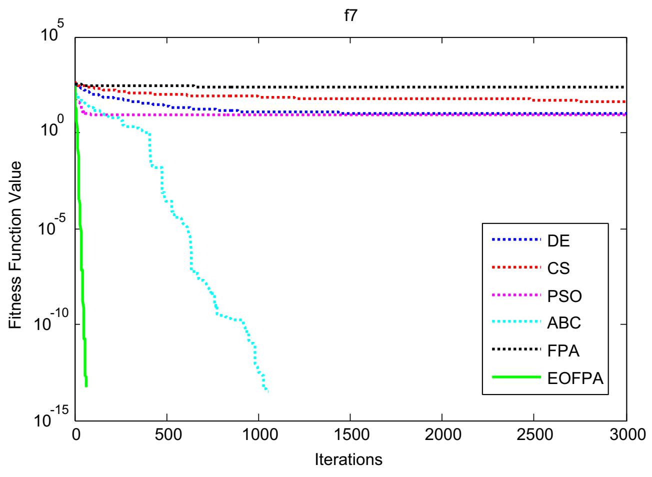

Fig. 7. $D=30$, evolution curves of fitness value for f_{07}.

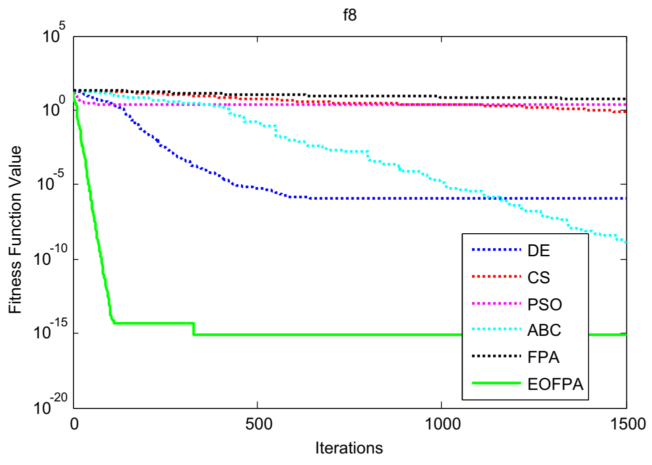

Fig. 8. $D=30$, evolution curves of fitness value for f_{08}.

Draw a (d-dimensional) step vector L which obeys a Lévy distribution;
Global pollination via Eq. (1) and get new solution x_i;
Use GEOLS to generate the elite opposition-based solution x_i' via (4);
Choose the better one between x_i and x_i' as the individual of next generation.
else

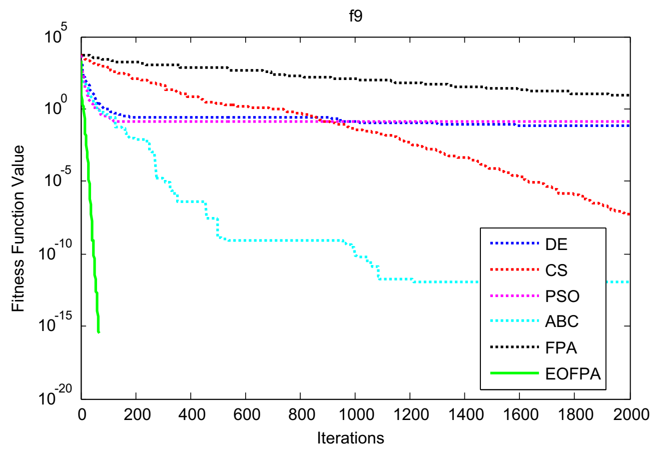

Fig. 9. $D=30$, evolution curves of fitness value for f_{09}.

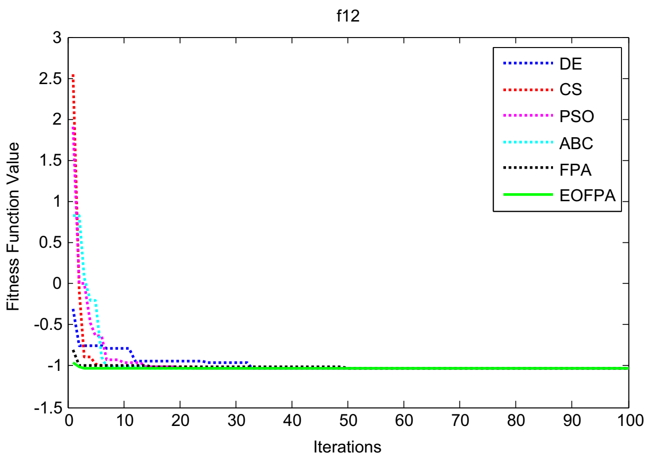

Fig. 12. $D=2$, evolution curves of fitness value for f_{12}.

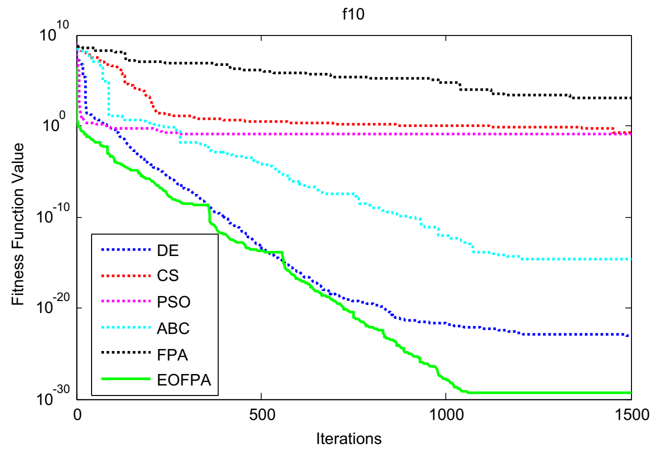

Fig. 10. $D=30$, evolution curves of fitness value for f_{10}.

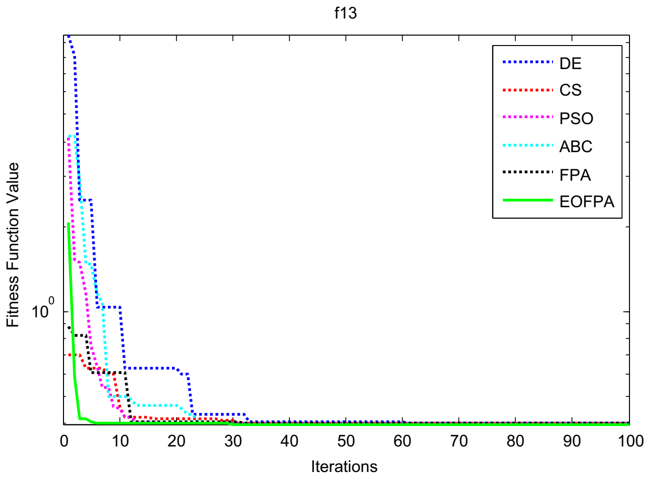

Fig. 13. $D=2$, evolution curves of fitness value for f_{13}.

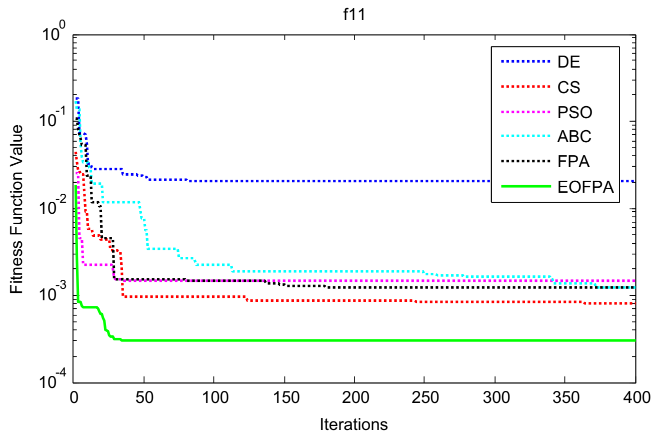

Fig. 11. $D=4$, evolution curves of fitness value for f_{11}.

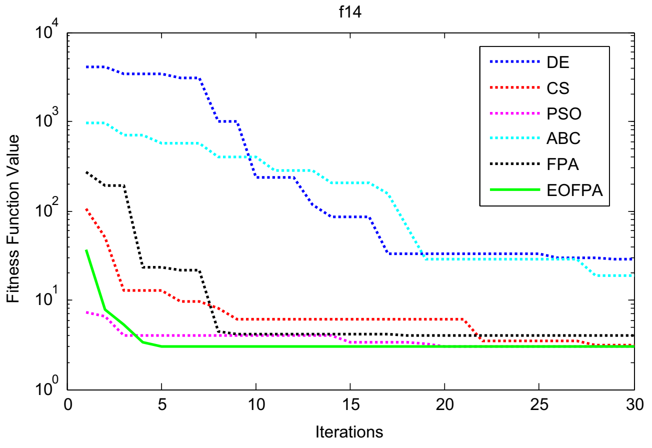

Fig. 14. $D=2$, evolution curves of fitness value for f_{14}.

```
            Draw U form a uniform distribution in (0,1);
            Do local pollination via Eq. (3) and get new solution $x_i$;
            Use LSGS to generate greedy solution $x_i'$ via Eq. (7).;
            Choose the better one between $x_i$ and$x_i'$ as the individual of next generation.
        endif
            Evaluate the new solutions;
            If new solutions are better, update them in the population;
    endfor
    Find the current best solution B;
    Update the switch probability p via Eq. (8);
end while
Output the best solution found
```

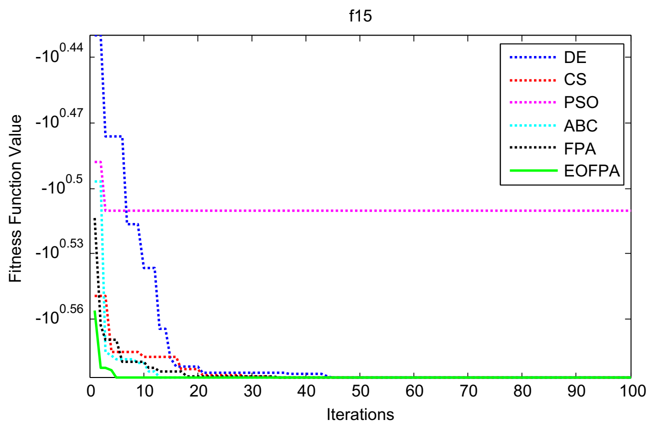

Fig. 15. $D=3$, evolution curves of fitness value for f_{15}.

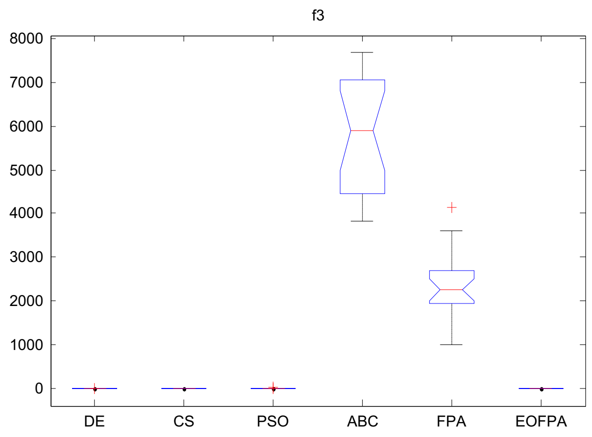

Fig. 18. $D=30$, anova test of global minimum for f_{03}.

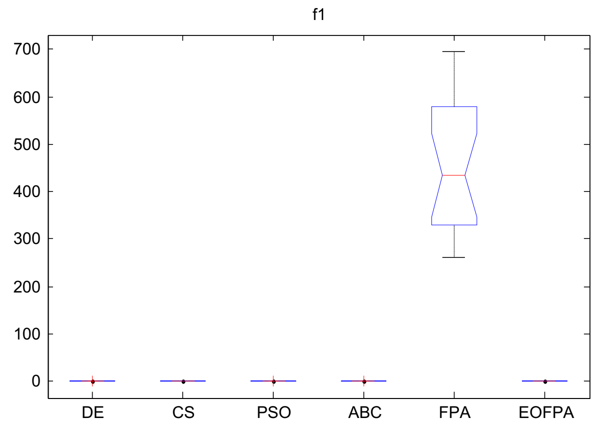

Fig. 16. $D=30$, anova test of global minimum for f_{01}.

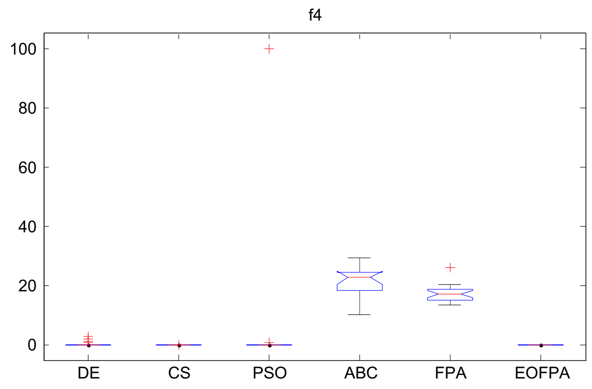

Fig. 19. $D=30$, anova test of global minimum for f_{04}.

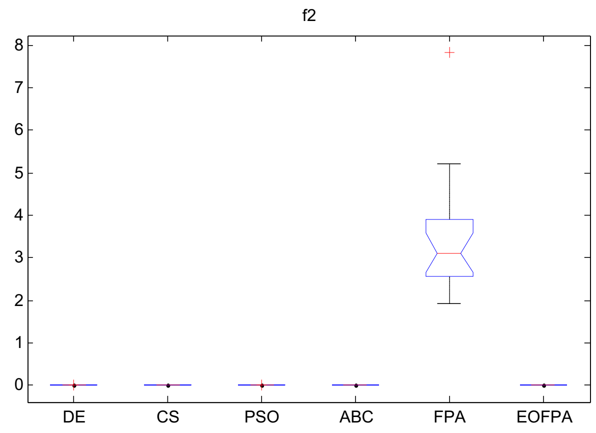

Fig. 17. $D=30$, anova test of global minimum for f_{02}.

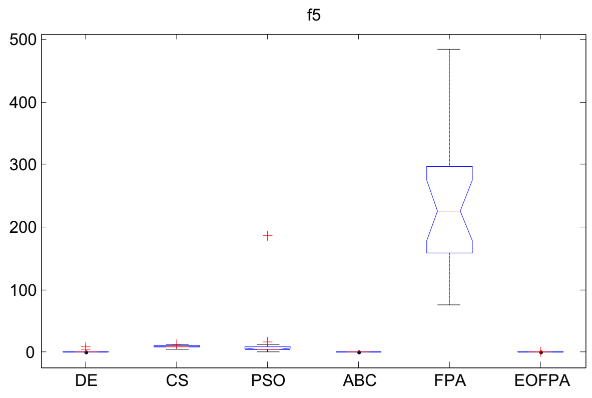

Fig. 20. $D=30$, anova test of global minimum for f_{05}.

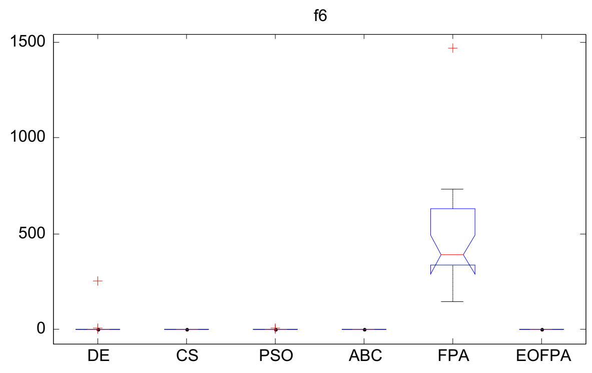

Fig. 21. $D=30$, anova test of global minimum for f_{06}.

4. Simulation experiments and result analysis

In this section, 18 standard test functions [19,20] are applied to evaluate the optimal performance of EOFPA. The space dimension, scope, optimal value and the iterations of 18 functions are shown in Table 1. The rest of this section is organized as follows: experimental setup is given in Section 4.1; the comparison of each algorithm performance is shown in Section 4.2; in Section 4.3, seven functions are selected to test the searching performance of EOFPA in high dimensional cases; and two structural design examples (welded beam design [21] and compression spring

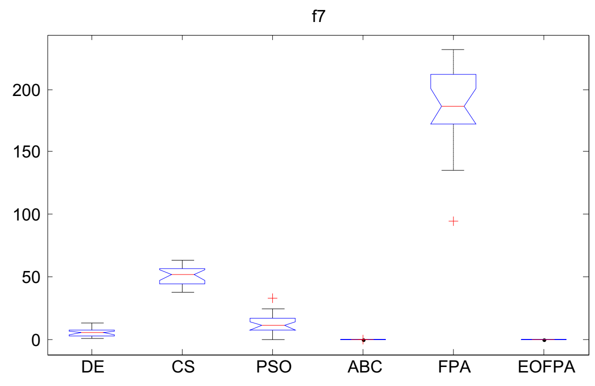

Fig. 22. $D=30$, anova test of global minimum for f_{07}.

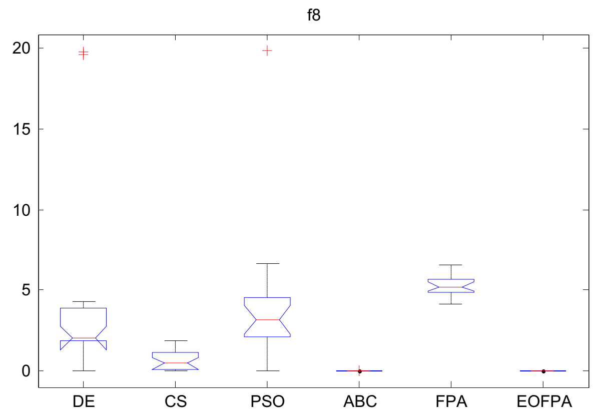

Fig. 23. $D=30$, anova test of global minimum for f_{08}.

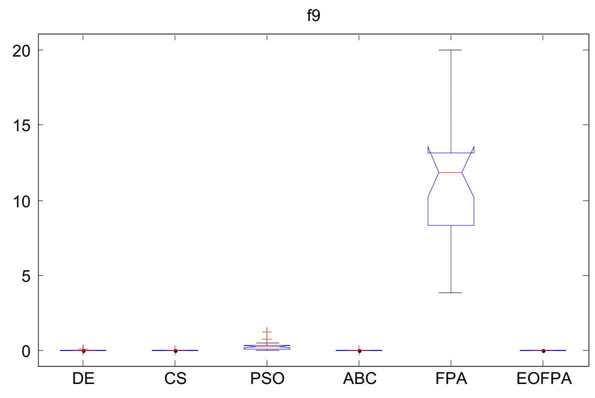

Fig. 24. $D=30$, anova test of global minimum for f_{09}.

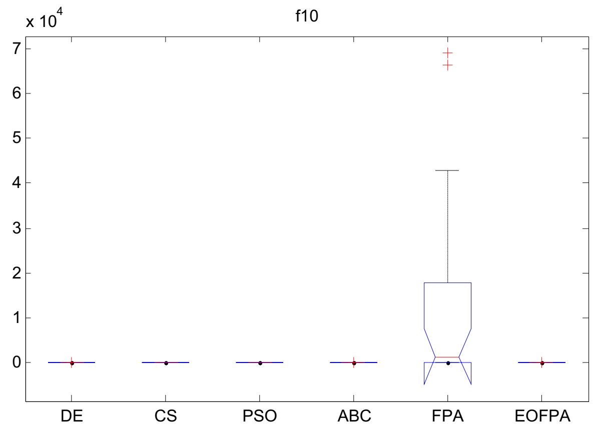

Fig. 25. $D=30$, anova test of global minimum for f_{10}.

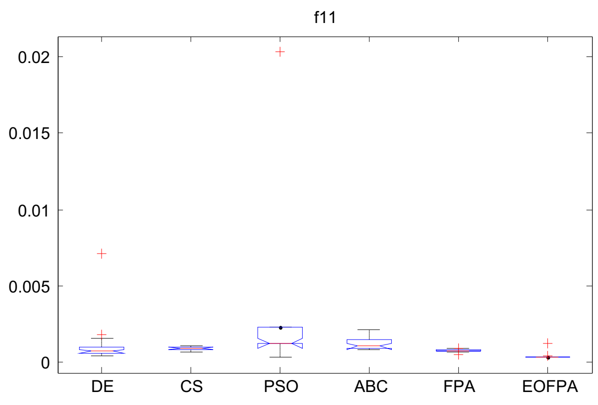

Fig. 26. $D=4$, anova test of global minimum for f_{11}.

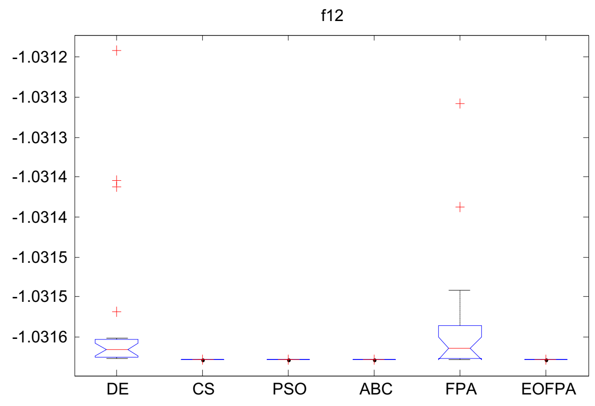

Fig. 27. $D=2$, anova test of global minimum for f_{12}.

design) [22] are used to evaluated its ability for solving constrained function optimization problem.

The benchmark functions selected can be divided into three categories (i.e., high-dimensional unimodal functions, high-dimensional multimodal functions and low-dimensional functions). They are f_{01},f_{02},f_{03},f_{04},f_{05} and f_{06} for category I, f_{07},f_{08}, f_{09} and f_{10} for category II and $f_{11}-f_{18}$ for category III. In high-dimensional functions, f_{05} is a classical test function. Its global minimum is in a parabolic valley, and function values change little in the valley. So, it is very difficult to find the global minimum. There are a large number of local minima in the solution space of f_{07} and in low-dimensional function, most functions have the characteristic of string shocks. As the global minimum of most benchmark functions in zero, in order to verify the searching capability of the algorithm effectively, we select some functions with nonzero global minimum.

4.1. Experimental setup

All of the algorithm was programmed in MATLAB R2012a, numerical experiment was set up on AMD Athlont (tm) II*4640 processor and 2 GB memory.

4.2. Comparison of each algorithm performance

The proposed elite opposition-based flower pollination algorithm compared with swarm intelligence optimization algorithms DE [23], CS [8], PSO [3], ABC [24], and FPA [10], respectively using the mean and standard deviation to compare their optimal performance. The setting values of algorithm control parameters of the mentioned algorithms are given below.

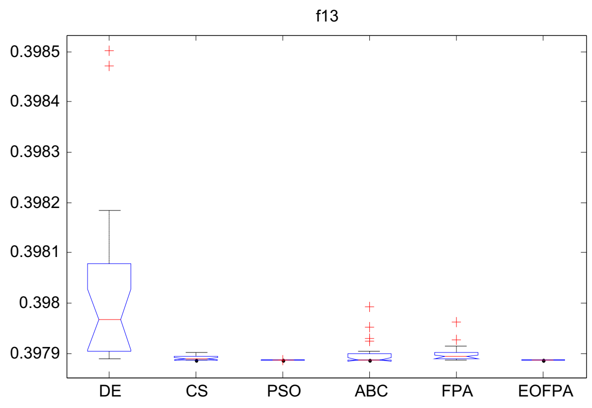

Fig. 28. $D=2$, anova test of global minimum for f_{13}.

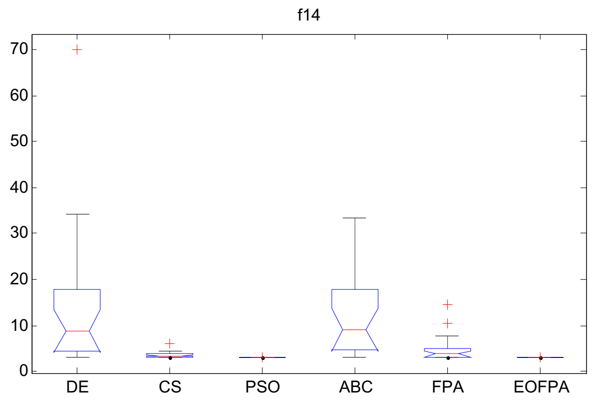

Fig. 29. $D=2$, anova test of global minimum for f_{14}.

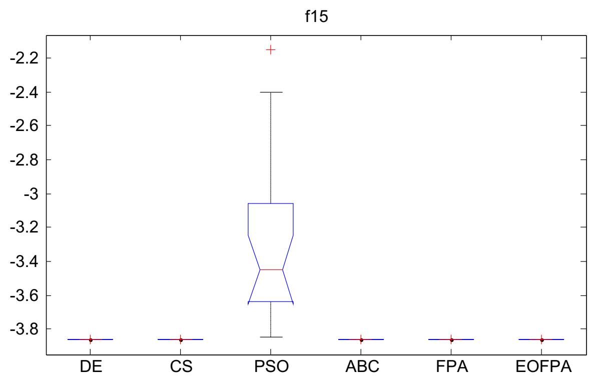

Fig. 30. $D=3$, anova test of global minimum for f_{15}.

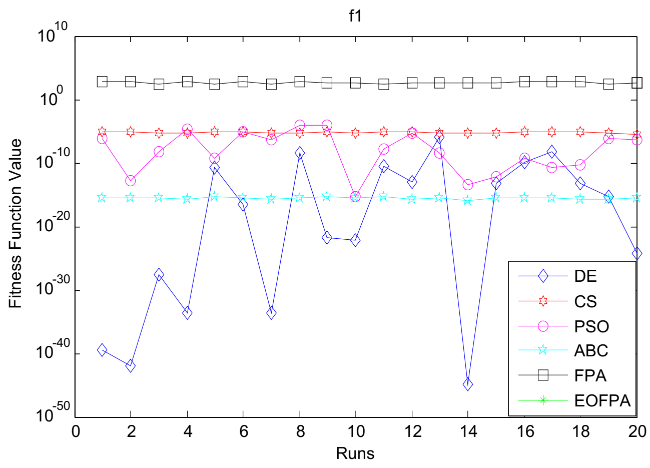

Fig. 31. $D=30$, comparison of optimal fitness value for f_{01}.

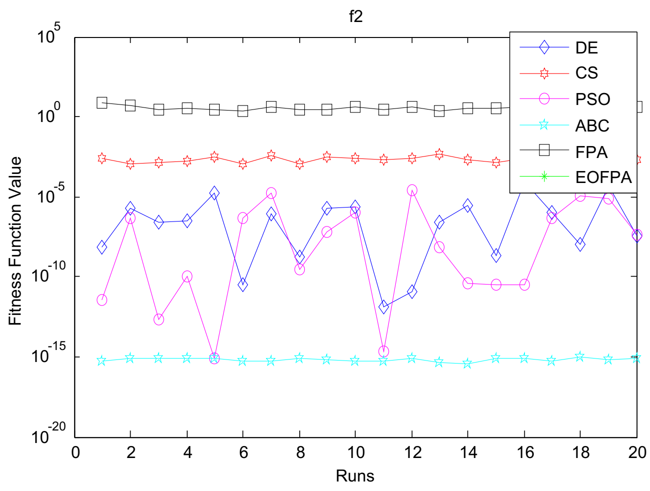

Fig. 32. $D=30$, comparison of optimal fitness value for f_{02}.

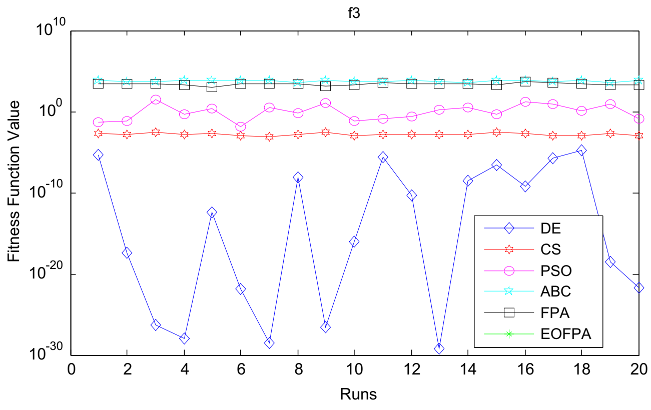

Fig. 33. $D=30$, comparison of optimal fitness value for f_{03}.

DE parameter setting: $F=0.5, CR=0.9$ in accordance with the suggestions given in [23], the population size is 50.

CS parameter setting: $\beta=1.5, \rho_0=1.5$ have been used as recommended in [8], the population size is 50.

PSO parameter setting: weight factor $\omega=0.6$, $c_1=c_2=2$ [3], the population size is 50.

ABC parameter setting: limit $=5D$ has been used as recommended in [24], the population size is 50.

FPA parameter setting: switch probability $\rho=0.8$ in accordance with the suggestions given in [10], the population size is 50.

For standard benchmark functions in Table 1, the comparison of test results is shown in Tables 2–4. In this paper, the results are obtained in 20 trials. The Best, Mean, Worst and Std. represent the optimal fitness value, mean fitness value, worst fitness value and standard deviation, respectively.

Bold and italicized results mean that EOFPA is better, while underlined results mean that other algorithm is better.

Seen from Table 2, in category I, EOFPA can find the optimal solution for f_{01},f_{02},f_{03},f_{04} and f_{06} and has a very strong robustness. For f_{05}, the precision of optimal fitness value and mean fitness value are higher than other population based algorithms. For the five functions in category I, standard deviation of EOFPA is less than that of other algorithm. And this means that in the optimization of high-dimensional unimodal function, EOFPA has better stability.

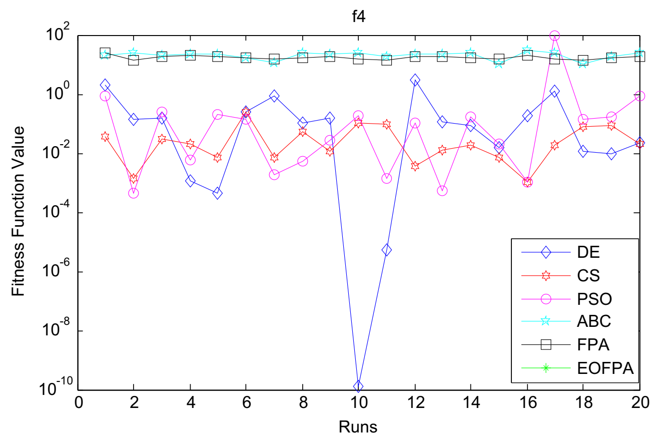

Fig. 34. $D=30$, comparison of optimal fitness value for f_{04}.

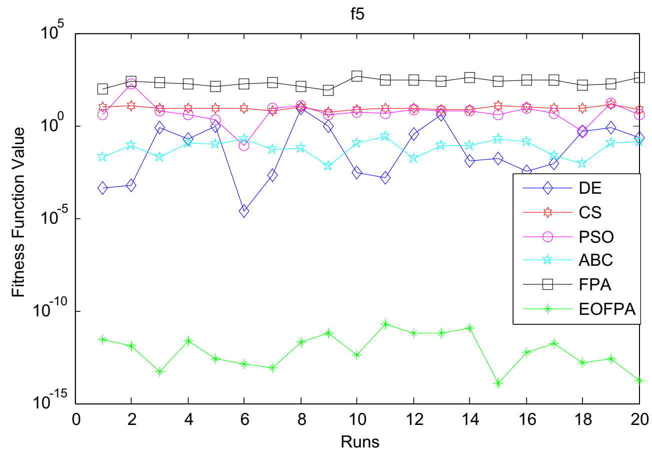

Fig. 35. $D=30$, comparison of optimal fitness value for f_{05}.

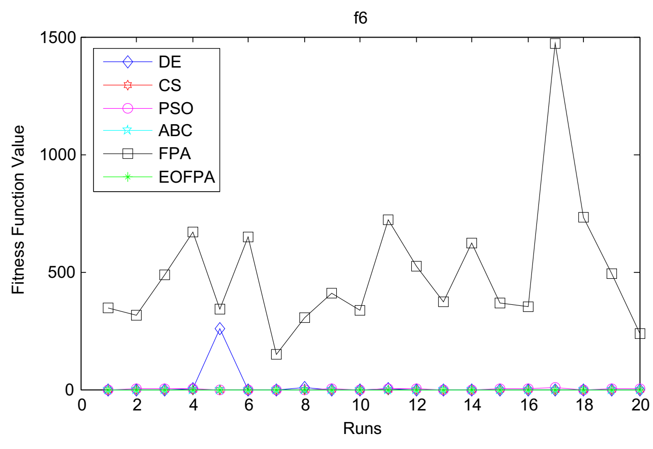

Fig. 36. $D=30$, comparison of optimal fitness value for f_{06}.

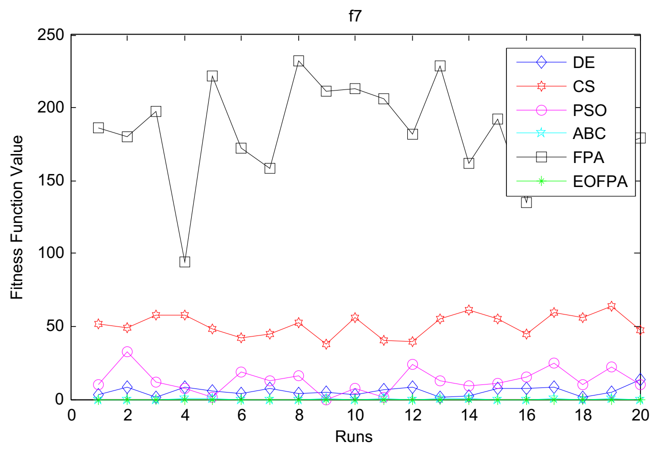

Fig.37. $D=30$, comparison of optimal fitness value for f_{07}.

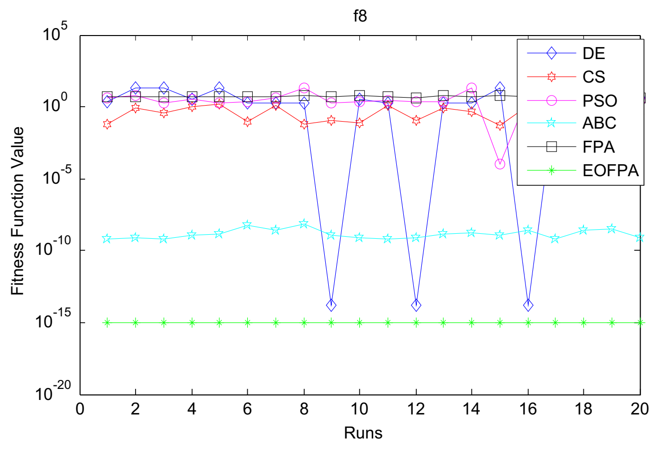

Fig. 38. $D=30$, comparison of optimal fitness value for f_{08}.

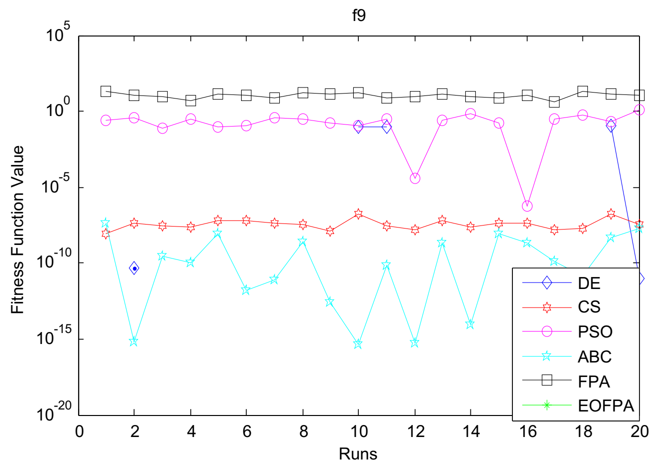

Fig. 39. $D=30$, comparison of optimal fitness value for f_{09}.

Similarly, seen from Table 3, EOFPA can find the optimal solution for f_{07} and f_{09}. And the standard deviations are zeros. For f_{08} and f_{10}, the precision of mean fitness value, best fitness value, worst fitness value and standard deviation of EOFPA are better than other algorithms. These results mean that EOFPA has a strong searching ability and a great stability for solving high-dimensional multimodal function optimization.

For f_{12},f_{13},f_{14} and f_{15}, we can see from Table 4 above that the best fitness value, the worst fitness value, mean fitness value and standard deviation produced by EOFPA are better than other algorithms. And for f_{11}, EOFPA can get better best fitness value and mean fitness value, but the worst fitness value and standard deviation are worse than that of FPA. We can also discover that EOFPA can find the optimal solution for f_{16},f_{17} and f_{18}. For f_{16}, mean fitness value and the worst fitness value of CS are better than that of EOFPA, and the standard deviation of FPA is the best. For f_{17}, mean fitness value and the worst fitness value are the best, and the stability of PSO is higher. And for f_{18}, we can easily discover that the worst fitness value, mean fitness value and the standard deviation of CS are better. After analyzing Table 4, a conclusion can be easily drawn that EOFPA has a great ability for solving low dimensional optimization problems according to the experimental results.

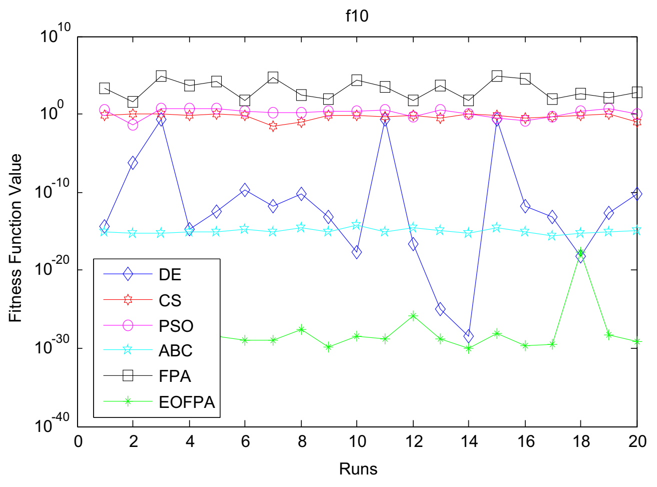

Fig. 40. $D=30$, comparison of optimal fitness value for f_{10}.

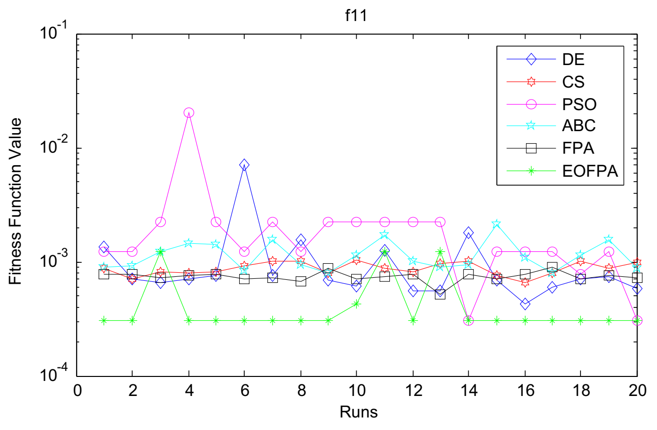

Fig. 41. $D=4$, comparison of optimal fitness value for f_{11}.

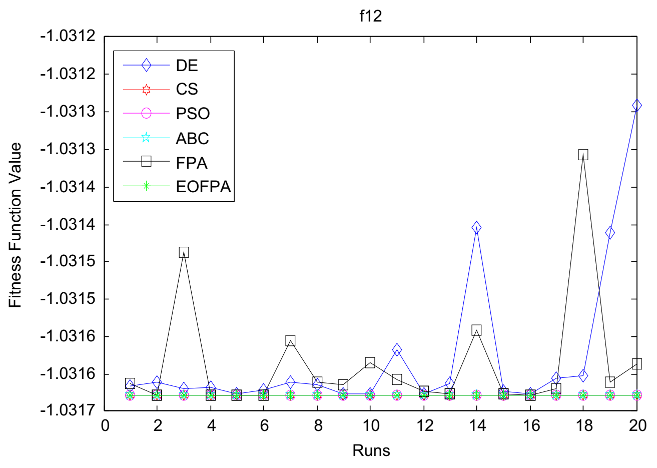

Fig. 42. $D=2$, comparison of optimal fitness value for f_{12}.

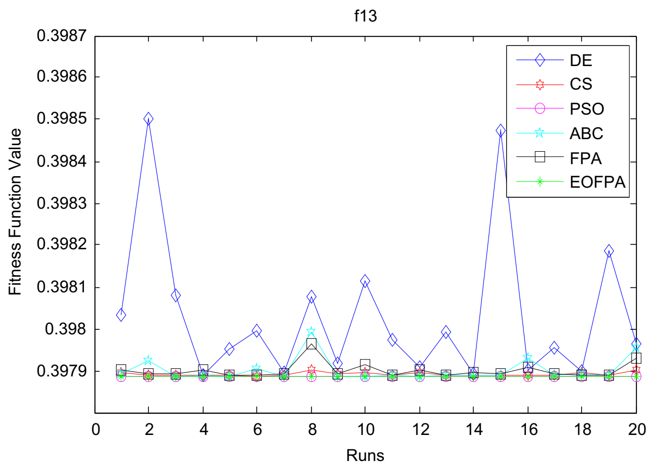

Fig. 43. $D=2$, comparison of optimal fitness value for f_{13}.

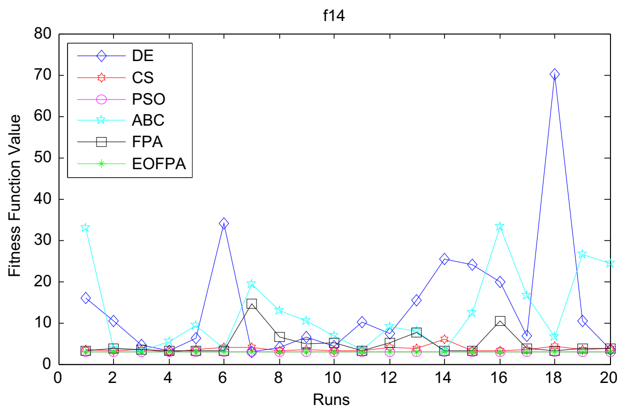

Fig. 44. $D=2$, comparison of optimal fitness value for f_{14}.

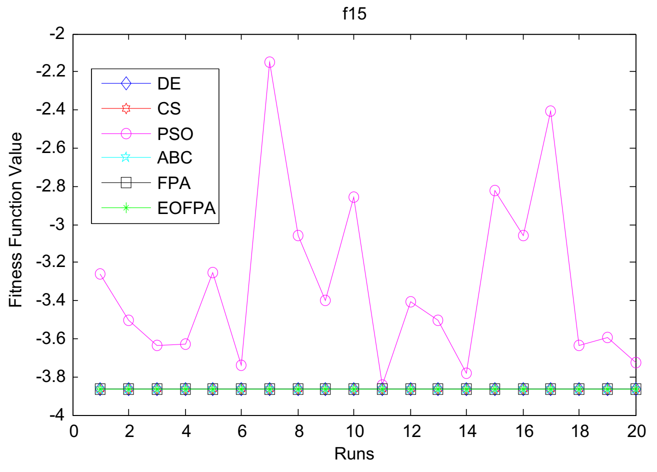

Fig. 45. $D=3$, comparison of optimal fitness value for f_{15}.

For functions $f_{01}-f_{15}$, Figs. 1–15 are the fitness evolution curves, Figs. 16–30 are the anova tests of the global minimum and Figs. 31–45 are the comparisons of optimal fitness value.

Figs. 1–15 show the evolution curves of fitness value for $f_{01}-f_{15}$. Blue dotted curves are plotted by DE [23]; red dotted curves are produced by CS [8]; PSO [3] draw pink dotted curves, sapphire dotted curves are drawn by ABC [24]; the evolution curves produced by FPA [10] are the dotted ones colored black; and the green solid curves are drawn by proposed EOFPA. From these Figs, we can easily find that EOFPA converge faster than

Table 5
High-dimensional functions test results.

Functions	Dimensions	Means	Std.	Best	Worst
*f*1	10,000	0	0	0	0
*f*2	10,000	0	0	0	0
*f*3	1000	0	0	0	0
*f*4	10,000	0	0	0	0
*f*6	10,000	0	0	0	0
*f*7	10,000	0	0	0	0
*f*9	10,000	0	0	0	0

other population based algorithms mentioned above, and the values obtained by EOFPA are closer to the optimal value of benchmark functions. These show that EOFPA has a faster convergence speed and a better precision than FPA and other population based algorithms. Figs. 16–30 show the anova test of global minimum for $f_{01}-f_{15}$. From these figs, we can discover that the standard deviation of EOFPA is much smaller, and the number of outlier is less than other algorithms. These imply that EOFPA has a great performance with a high degree of stability. Figs. 31–45 show comparisons of optimal fitness value for $f_{01}-f_{15}$. After analyzing this group of figs. The conclusion can be drawn that the stability of EOFPA is higher than that of others. In sum, proposed EOFPA is an algorithm with fast convergence speed, high level of precision and a great performance of stability.

4.3. *High-dimensional functions test*

In above section, 18 standard benchmark functions are applied to evaluate the optimal performance of the elite opposition-based flower pollination algorithm (EOFPA) in the case of normal dimension. In order to evaluate the performance of EOFPA comprehensively, we also do some high-dimensional test in f_1,f_2,f_3, f_4,f_6,f_7 and f_9. The test results are shown in Table 5. As can be seen in Table 5, EOFPA can also solve high-dimensional function optimization problems efficiently and stably.

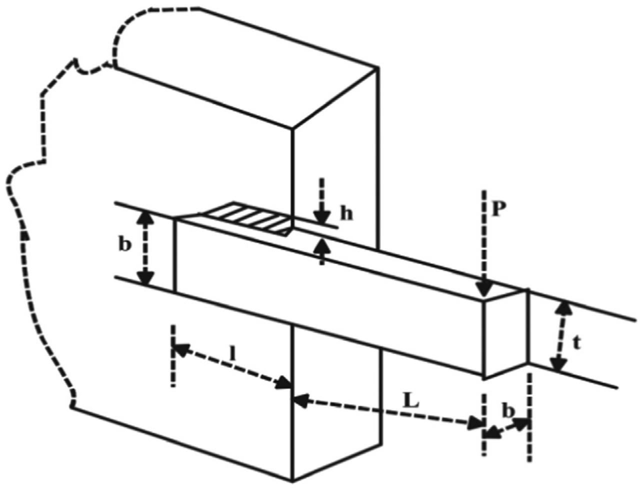

Fig. 46. Design of welded beam problem.

4.4. *Structural design examples*

Design optimization, especially the design of structures, has many applications in engineering and industry. As a result, there are many different benchmarks with detailed studies in the literature [19]. In this subsection, EOFPA will be used to solve two design case studies: design of a welded beam and a compression spring.

4.4.1. *Welded beam design*

The welded beam structure, shown in Fig. 46 taken from Rao [25], is a practical design problem that has been often used as a benchmark problem. The objective is to find the minimum fabricating cost of the welded beam subject to constraints on shear stress (τ), bending stress in beam (θ), buckling load on the bar (P_c), end deflection of the beam (δ), and side constraints. There are four design variables

associated with this problem namely, thickness of the beam $h=x_1$, length of the welded joint $l=x_2$, width of the beam $t=x_3$ and thickness of the beam $b=x_4$ i.e. the decision vector is $X=(h,l,t,b)=(x_1,x_2,x_3,x_4)$. For this particular problem, there are several modals available in the literature, all of them are vary with respect to number of constraints and the way in which constraints are defined.

In this paper, the mathematical formulation of the objective function $f(X)$ which is the total fabricating cost mainly comprised of the set-up, welding labor, and material cost, is as follows:

Minimize $f(X)=1.10471x_1^2x_2+0.04811x_3x_4(14+x_2)$

$$s.t. g_1(X)=\tau(X)-\tau_{max}\le 0;$$

$$g_2(X)=\sigma(X)-\sigma_{max}\le 0;$$

$$g_3(X)=x_1-x_4\le 0$$

$$g_4(X)=0.125-x_1\le 0;$$

$$g_5(X)=\delta(X)-0.25\le 0;$$

$$g_6(X)=P-P_c(X)\le 0;$$

Table 6
Comparison of the best solution for welded beam found by different methods.

Method	Design variables				*f(X)*
	w	L	D	h	
Coello [26]	0.208800	3.420500	8.997500	0.210000	1.748309
Coello and Montes [27]	0.205986	3.471328	9.020224	0.206480	1.728226
Hu et al. [28]	0.20573	3.47049	9.03662	0.20573	1.72485084
Hedar and Fukushima [29]	0.205644	3.4725787	9.03662391	0.2057296	1.7250022
He and Wang [30]	0.202369	3.544214	9.048210	0.205723	1.728024
Dimopoulos [31]	0.2015	3.5620	9.041398	0.205706	1.731186
Mahdavi et al. [32]	0.20573	3.47049	9.03662	0.20573	1.7248
Montes et al. [33]	0.205730	3.470489	9.036624	0.205730	1.724852
Montes and Coello [34]	0.199742	3.612060	9.037500	0.206082	1.73730
Cagnina et al. [35]	0.205729	3.470488	9.036624	0.205729	1.724852
Fesanghary et al. [36]	0.20572	3.47060	9.03682	0.20572	1.7248
Kaveh and Talatahari[37]	0.205729	3.469875	9.036805	0.205765	1.724849
Kaveh and Talatahari[38]	0.205700	3.471131	9.036683	0.205731	1.724918
Gandomi et al. [39]	0.2015	3.562	9.0414	0.2057	1.73121
Mehta and Dasgupta[40]	0.205728	3.4705056	9.03662392	0.2057296	1.724855
Akay and Karaboga [41]	0.205730	3.470489	9.036624	0.205730	1.724852
Our results	**0.205729**	**3.2531200**	**9.036623910**	**0.2057296**	**1.69524716**

Table 7
Statistical results of different methods for welded beam design problem (NA means not available).

Method	Best	Mean	Worst	Std-dev	Median
Coello [26]	1.748309	1.771973	1.785835	0.011220	NA
Coello and Montes [27]	1.728226	1.792654	1.993408	0.07471	NA
Dimopoulos [31]	1.731186	NA	NA	NA	NA
He and Wang [30]	1.728024	1.748831	1.782143	0.012926	NA
Hedar and Fukushima [29]	1.7250022	1.7564428	1.8843960	0.0424175	NA
Montes et al. [33]	1.724852	1.725	NA	1E-15	NA
Montes and Coello [34]	1.737300	1.813290	1.994651	0.070500	NA
Cagnina et al. [35]	1.724852	2.0574	NA	0.2154	NA
Kaveh and Talatahari [38]	1.724918	1.729752	1.775961	0.009200	NA
Kaveh and Talatahari [37]	1.724849	1.727564	1.759522	0.008254	NA
Gandomi et al. [39]	1.7312065	1.8786560	2.3455793	0.2677989	NA
Mehta and Dasgupta [40]	1.724855	1.724865	1.72489	NA	1.724861
Akay and Karaboga [41]	1.724852	1.741913	NA	0.031	NA
Our results	**1.6952472**	**1.6952472**	**1.6952472**	**2.13967E-16**	**1.6952472**

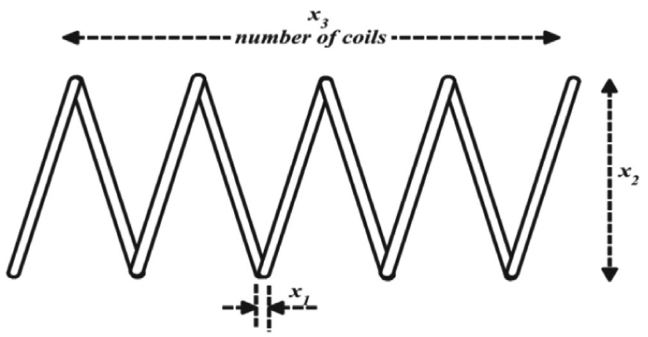

Fig. 47. Design of compression spring problem.

$$g_7(X) = 0.10471x_1^2 + 0.04811x_3x_4(14+x_2) - 5 \le 0;$$

$$0.1 \le x_1 \le 2; 0.1 \le x_2 \le 10; 0.1 \le x_3 \le 10; 0.1 \le x_4 \le 2$$

where τ is the shear stress in the weld, $\tau_{\max}$ is the allowable shear stress of the weld ($=$ 13,600 psi), $\sigma$the normal stress in the beam, $\sigma_{\max}$ is the allowable normal stress for the beam material ($=$ 30,000 psi), P_c the bar buckling load, P the load ($=$6000 lb), and δ the beam end deflection.

The shear stress τ has two components namely primary stress (τ_1) and secondary stress (τ_2) given as

$$\tau(X) = \sqrt{\tau_1^2 + 2\tau_1\tau_2(\frac{x_2}{2R}) + \tau_2^2}; \quad \tau_1 = \frac{P}{\sqrt{2}x_1x_2}; \quad \tau_2 = \frac{MR}{J};$$

where

$$M = P\left(L + \frac{x_2}{2}\right); \quad J(X) = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + (\frac{x_1+x_2}{2})^2\right]\right\}$$

are known as moments and polar moment of inertia respectively while the other terms associated with the modal are as follows

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}; \quad \sigma(X) = \frac{6PL}{x_4x_3^2}$$

$$\delta(X) = \frac{6PL^3}{Ex_3^3x_4}; \quad P_c = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

$$G = 12 \times 10^6 \text{ psi}, E = 30 \times 10^6 \text{ psi}, P = 6000 \text{ lb}, L = 13 \text{ in}$$

A comparison of the present work with the previous studies is presented in Table 6. From the table, it has been seen that the proposed EOFPA performs much better in comparison to other algorithms as optimal function value is lower than the previous studies. The statistically result, after 20 independent runs, in terms of the best, median, mean, worst and the standard deviation obtained for the best objective value by EOFPA are given in Table 7. It shows that mean from the 20 runs performed is $f(X) = 1.6953472$ with a standard deviation of $2.13967E-16$. Also the worst solution found in this version is better than any of the solutions produced by any other techniques. The best solution reported by EOFPA is $f(X) = 1.6952472$ corresponding to decision variable $x_1 = 0.205729, x_2 = 3.2531200, x_3 = 9.036623910, x_4 = 0.2057296$. Thus the proposed EOFPA provides the best results.

4.4.2. *Compression spring design*

This problem is described by Arora [43] and Belegundu [42] and it consists of minimizing the weight of a compression spring (as shown in Fig. 47) subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The design variables are the mean coil diameter (x_1), the wire diameter (x_2)and the number of active coil (x_3). The mathematical formulation of this problem can be described as follow:

Minimize $f(X) = (x_3+2)x_2x_1^2$

$$s.t. g_1(X) = 1 - \frac{x_2^3x_3}{71785x_1^4} \le 0$$

$$g_2(X) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5180x_1^2} - 1 \le 0$$

$$g_3(X) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0$$

$$g_4(X) = \frac{x_1+x_2}{1.5} \le 0$$

$$0.05 \le x_1 \le 2; 0.25 \le x_2 \le 1.3; 2 \le x_3 \le 15$$

This problem has been solved by Belegundu [42] using eight different mathematical optimization techniques (only the best results are shown). Arora [43] solved this problem using a numerical optimization technique called a constraint correction at the constant cost. Coello [26] and Coello and Montes [44] solved this problem using GA-based method. Additionally, He and Wang [30] utilized a co-evolutionary particle swarm optimization (CPSO). Montes and Coello [34] used various evolution strategies to solve this problem. Table 8 presents the best solution of this problem obtained using the proposed EOFPA and compares with the solutions reported by other researchers, and correspondingly statistical simulation results are

Table 8
Comparison of the best solution for tension/compression string design problem by different methods.

Method	Design variables			*f*(*X*)
	d	D	N	
Belegundu [42]	0.05	0.315900	14.25000	0.0128334
Arora [43]	0.053396	0.399180	9.185400	0.0127303
Coello [26]	0.051480	0.351661	11.632201	0.01270478
Ray and Saini [45]	0.050417	0.321532	13.979915	0.013060
Coello and Montes [44]	0.051989	0.363965	10.890522	0.0126810
Ray and Liew [46]	0.0521602170	0.368158695	10.6484422590	0.01266924934
Hu et al. [28]	0.051466369	0.351383949	11.60865920	0.0126661409
Hedar and Fukushima [29]	0.051742503409	0.358004783455	11.2139073627	0.012665285
He et al. [47]	0.05169040	0.35674999	11.28712599	0.0126652812
Raj et al. [48]	0.05386200	0.41128365	8.68437980	0.01274840
Tsai [49]	0.05168906	0.3567178	11.28896	0.01266523
Mahdavi et al. [32]	0.05115438	0.34987116	12.0764321	0.0126706
Montes et al. [33]	0.051688	0.356692	11.290483	0.012665
He and Wang [30]	0.051728	0.357644	11.244543	0.0126747
Cagnina et al. [35]	0.051583	0.354190	11.438675	0.012665
Zhang et al. [50]	0.0516890614	0.3567177469	11.2889653382	0.012665233
Montes and Coello [34]	0.051643	0.355360	11.397926	0.012698
Omran and Salman [51]	0.0516837458	0.3565898352	11.2964717107	0.0126652375
Keveh and Talatahari [38]	0.051865	0.361500	11.00000	0.0126432
Coelho [52]	0.051515	0.352529	11.538862	0.012665
Akay and Karaboga [41]	0.051749	0.358179	11.203763	0.012665
Our results	**0.05168942400**	**0.35672647054**	**11.2884539259**	**0.01266523280**

Table 9
Statistical results of different methods for tension/compression string (NA means not available).

Method	Best	Mean	Worst	Std-Dev	Median
Belegundu [42]	0.0128334	NA	NA	NA	NA
Arora [43]	0.0127303	NA	NA	NA	NA
Coello [26]	0.0127047	0.0127692	0.01282208	3.9390E − 5	0.01275576
Ray and Saini [45]	0.0130600	0.015526	0.018992	NA	NA
Coello and Montes [44]	0.0126810	0.012742	0.012973	5.9000E − 5	NA
Ray and Liew [46]	0.0126692	0.0129226	0.01671727	5.92E-4	0.0129226
Hu et al. [28]	0.0126661	0.0127189	NA	6.446E-5	NA
He et al. [47]	0.0126652	0.0127023	NA	4.12439E-5	NA
He and Wang [30]	0.0126747	0.012730	0.012924	5.1985E-5	NA
Zhang et al. [50]	0.0126652	0.01266936	0.01273826	1.25E-5	NA
Hedar and Fukushima [29]	0.0126652	0.012665299	0.012665338	2.2E-8	NA
Montes et al. [33]	0.012665	0.012666	NA	2.0E-6	NA
Montes and Coello [34]	0.012698	0.013461	0.164850	9.6600E-4	NA
Cagnina et al. [35]	0.012665	0.0131	NA	4.1E-4	NA
Kaveh and Talatahari [38]	0.0126432	0.012720	0.012884	3.4888E-5	NA
Omran and Salman [51]	0.0126652	0.012665264	NA	NA	NA
Coelho [52]	0.012665	0.013524	0.017759	0.001268	0.012957
Akay and Karaboga [41]	0.012665	0.012709	NA	0.012813	NA
Our results	**0.0126652**	**0.0126652**	**0.0126652**	**4.68127E-10**	**0.0126652**

shown in Table 9. The best results obtained by EOFPA is

$$f(X) = 0.012665232809742$$

corresponding to $X = [0.051689424008163, 0.356726470546953, 11.288453925993009]$.

4.5. Result analysis

Sufficient simulation experiment has been conducted in Sections 4.2–4.4. In Section 4.2, 18 standard benchmark functions are selected to evaluate the performance of elite opposition based flower pollination algorithm (EOFPA). $f_{01} - f_{06}$ are unimodal functions, $f_{07} - f_{10}$ are multimodal functions, $f_{11} - f_{18}$ are low-dimensional functions. Experiment results are listed in Tables 2–4. Figs. 1–45 are evaluation curves of fitness values, anova test of global minimum, comparison of optimal fitness value for benchmark functions $f_{01} - f_{15}$. The results in tables show that a more precise solution can be found by EOFPA. Figures listed in paper reflect a fact that EOFPA has a faster convergence speed and a higher stability. In Section 4.3, seven benchmarks are selected to test the performance of EOFPA in high-dimensional cases. And the data listed in Table 5 shows that EOFPA has a remarkable ability for solving high-dimensional function optimization problem. In Section 4.4, two structure design problem (welded beam design and compression spring design) are chosen to test the proposed EOFPA. The remarkable achievements show that EOFPA has an outstanding ability for solving constrained optimization problems.

5. Conclusions and future works

In order to overcome the disadvantage of Standard Flower Pollination Algorithm, three optimization strategies have been incorporated into the FPA to generate an elite opposition-based flower pollination algorithm for function optimization and structure engineering design problems. The global elite opposition-based learning enhances the diversity of the population, which helps to improve its exploration ability. And local self-adaptive greedy strategy changes the bound dynamically, which is good for local search. By using the dynamic switching probability strategy, EOFPA can balance exploration and exploitation and cope with multi-modal benchmarks and some structure engineering design problems. From the results of the 18 benchmark functions and two engineering design problems, the performance of EOFPA is better than, or at least comparable with other population-based algorithm mentioned in this paper. EOFPA has a fast convergence speed, a relatively high degree of stability. And it is also much more accurate in precision.

For EOFPA, there are various issues that still deserve further study. Firstly, multi-objective optimization problems can be seen here and there in real world. Compared with single objective optimization problems, it is often very challenging to obtain high-equality solution. The proposed elite opposition-based flower pollination algorithm should be used to solve these multi-objective optimization problems in the future to validate the performance its performance. Secondly, there exists many NP- hard problems in literature, such as traveling salesman problem, graph coloring problem, radial basis probabilistic neural networks [53], finder of polynomials based on root moments [54] and knapsack problem. In order to test performance of EOFPA comprehensively, it should be used to solve these NP-hard problems in the future.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grants nos. 61165015, 61463007. Key Project of Guangxi Science Foundation under Grant no. 2012GXNSFDA053028, Key Project of Guangxi High School Science Foundation under Grant no. 20121ZD008. Key Project of Guangxi University for Nationalities Science Foundation under Grant no. 2012MDZD037.

References

[1] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, Eur. J. Op. Res. 185 (3) (2008) 1155–1173.
[2] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341–359.
[3] J. Kennedy, R. Eberhart, Particle swarm optimization, In: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 1995, IV, pp. 1942–1948.
[4] Yang Xinshe, Multiobjective firefly algorithm for continuous optimization, Eng. Comput. 29 (2) (2013) 175–184.
[5] Liu Zhou Yongquan, Zhao Guangwei Jiakun, Leader glowworm swarm optimization algorithm for solving nonlinear equations systems, Electr. Rev. 88 (1b) (2012) 101–106.
[6] A. Mucherino, O. Seref, Monkey search: a novel metaheuristic search for global optimization, In: Proceedings of the American Institute of Physics Conference Gainesville, USA, pp. 162–173.
[7] B. Alatas, Chaotic harmony search algorithms, Appl. Math. Comput. 216 (9) (2010) 2687–2699.
[8] X.S. Yang, S. Deb, Cuckoo search via Levy flights, In: Proceedings of the World Congress on Nature and Biologically Inspired Computing, NaBIC 2009, IEEE Publication, USA, 2009, pp. 210–214.
[9] X.S. Yang, A new metaheuristic bat-inspired algorithm, in: J.R. Gonzalez, D. A. Pelta, C. Cruz (Eds.), Nature Inspired Cooperative Strategies for Optimization, Springer-Ver-lag, Berlin, Germany, 2010, pp. 65–74.
[10] X.S. Yang, Flower pollination algorithm for global optimization, In: Proceedings of the Unconventional Computation and Natural Computation, Lecture Notes Computer Science, 2012, 7445, pp. 240–249.
[11] X.S. Yang, S. Deb, Eagle strategy using Levy walk and firefly algorithm for stochastic optimization, in: J.R. Gonzalez, et al., (Eds.), Nature Inspired Cooperative Strategies for Optimization, 284, NICSO, 2010, pp. 101–111 2010, SCI.
[12] I. Pavlyukevich, Lévy flights, non-local search and simulated annealing, J. Comput. Phys. 226 (2007) 1830–1844.
[13] D.S. Huang, Yu Hong-Jie, Normalized feature vectors: a novel alignment-free sequence comparison method based on the numbers of adjacent amino acids, IEEE/ACM Trans. Comput. Biol. Bioinform. 10 (2) (2013) 457–467.
[14] X.S. Yang, M. Karamanoglu, X.S. He, Multiobjective flower algorithm for optimization, Proc. Comput. Sci. 18 (2013) 861–868.
[15] E. Marwa Sharawi, Imane Aly Emary, Hesham El-Mahdy Saroit, Flower pollination optimization algorithm for wireless sensor network lifetime global optimization, Int. J. Soft Comput. Eng. 4 (2014) 3.
[16] Osama Abdel Raouf, Ibrahim El-henawy, A. Mohamed Abdel-Baset., Novel hybrid flower pollination algorithm with chaotic harmony search for solving sudoku puzzles, Int. J. Mod. Educ. Comput. Sci. 3 (2014) 38–44, http://dx.doi.org/10.5815/ijmecs.2014.03.05 http://www.mecs-press.org/.
[17] Mahmoud Ismail Ibrahim El-henawy, An Improved chaotic flower pollination algorithm for solving large integer programming problems, Int. J. Digit. Content Technol. Appl. 8 (3) (2014) 72–81.
[18] X.S. Yang, Engineering Optimization: An Introduction with Metaheuristic Applications, John Wiley and Sons, USA, 2010.
[19] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, Z. Yang, Benchmark Functions for the CEC 2008 Special Session and Competition on Large Scale Global Optimization, University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL), Hefei, Anhui, China, 2007, Tech. Rep. http://nical.ustc.edu.cn/cec08ss.php.
[20] N. Hansen, A. Auger, S. Finck, R. Ros, Real-Parameter Black-Box Optimization Benchmarking 2009 Experimental Setup, Institute National de Recherche en Informatique et en Automatique (INRIA), 2009, Rapports de Recherche RR-6828, 〈http://hal.inria.fr/inria-00362649/en/〉.
[21] K. Deb, Optimal design of a welded beam via genetic algorithms, AIAA J. 29 (11) (1991) 2013–2015.
[22] J.K., S.S. Kuang, Li Chen, Taguchi-aided search method for design optimization of engineering systems, Eng. Optim. 30 (1998) 1–23.
[23] X. Li, M. Yin, An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure, Adv. Eng. Softw. 55 (2013) 10–31.
[24] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Glob. Optim. 39 (3) (2007) 459–471.
[25] S.S. Rao, Engineering Optimization: Theory and Practice, 3rd edition, John Wiley & Sons, Chichester, 1996.
[26] C.A.C. Collo, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2000) 113–127.
[27] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Inform. 16 (2002) 193–203.
[28] X.H. Hu, R.C. Eberhart, Y.H. Shi, Engineering optimization with particle swarm, In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003, pp. 53–57.
[29] A.R. Hedar, M. Fukushima, Derivative-free filter simulated annealing method for constrained continuous global optimization, J. Glob. Optim. 35 (2006) 521–549.
[30] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Eng. Appl. Artif. Intell. 20 (2007) 89–99.
[31] G.G. Dimopoulos, Mixed-variable engineering optimization based on evolutionary and social metaphors, Comput. Methods Appl. Mech. Eng. 196 (2007) 803–817.
[32] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2007) 1567–1579.
[33] E.M. Montes, C.A.C. Coello, J.V. Reyes, L.M. Davila, Multiple trial vectors in differential evolution for engineering design, Eng. Optim. 39 (2007) 567–589.
[34] E.M. Montes, C.A.C. Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, Int. J. Gen. Syst. 37 (2008) 443–473.
[35] L.C. Cagnina, S.C. Esquivel, C.A.C. Coello, Solving engineering optimization problems with the simple constrained particle swarm optimizer, Information 32 (2008) 319–326.
[36] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, Y. Alizadeh, Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems, Comput. Methods Appl. Mech. Eng. 197 (2008) 3080–3091.
[37] A. Kaveh, S. Talatahari, Engineering optimization with hybrid particle swarm and ant colony optimization, Asian J. Civ. Eng. (Build. Hous.) 10 (2009) 611–628.
[38] A. Kaveh, S. Talatahari, An improved ant colony optimization for constrained engineering design problems, Eng. Comput. 27 (2010) 155–182.
[39] A.H. Gandomi, X.S. Yang, A.H. Alavi, Mixed variable structural optimization using firefly algorithm, Comput. Struct. 89 (2011) 2325–2336.
[40] V.K. Mehta, B. Dasgupta, A constrained optimization algorithm based on the simplex search method, Eng. Optim. 44 (2012) 537–550.
[41] B. Akay, D. Karaboga, Artificial bee colony algorithm for large-scale problems and engineering design optimization, J. Intell. Manuf. 23 (2012) 1001–1014.
[42] A.D. Belegundu, A Study of Mathematical Programming Methods for Structural Optimization (PhD thesis), Department of Civil and Environmental Engineering, University of Iowa, Iowa, USA, 1982.
[43] J.S. Arora, Introduction to Optimum Design, McGraw-Hill, New York, 1989.

[44] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Inform. 16 (2002) 193–203.
[45] T. Ray, P. Saini, Engineering design optimization using a swarm with an intelligent information sharing among individuals, Eng. Optim. 33 (2001) 735–748.
[46] T. Ray, K.M. Liew, Society and civilization: an optimization algorithm based on the simulation of social behavior, IEEE Trans. Evol. Comput. 7 (2003) 386–396.
[47] S. He, E. Prempain, Q.H. Wu, An improved particle swarm optimizer for mechanical design optimization problems, Eng. Appl. Artif. Intell. 20 (2007) 89–99.
[48] K.H. Raj, R.S. Sharma, G.S. Mishra, A. Dua, C. Patvardhan, An evolutionary computational technique for constrained optimization in eigineering design, J. Inst. Eng. India Part Mech. Eng. Div. 86 (2005) 121–128.
[49] J. Tsai, Global optimization of nonlinear fractional programming problems in engineering design, Eng. Optim. 37 (2005) 399–409.
[50] M. Zhang, W. Luo, X. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, Inf. Sci. 178 (2008) 3043–3074.
[51] M.G.H. Omran, A. Salman, Constrained optimization using CODEQ, Chaos Solitons Fractals 42 (2009) 662–668.
[52] L.S. Coelho, Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems, Expert Syst. Appl. 37 (2010) 1676–1683.
[53] D.S. Huang, Ji-Xiang Du., A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks, IEEE Trans. Neural Netw. 19 (12) (2008) 2099–2115.
[54] D.S. Huang, H.S. Ip Horace, Zheru Chi, A neural root finder of polynomials based on root moments, Neural Comput. 16 (8) (2004) 1721–1762.
[55] X.S. Yang, K. Mehmet, X.S. He, Flower pollination algorithm: a novel approach for multiobjective optimization, Eng. Optim. 46 (9) (2014) 1222–1237, http://dx.doi.org/10.1080/0305215X.2013.832237.

Yongquan Zhou, Ph.D and Prof. He received the MS degree in computer science from Lanzhou University, Lanzhou, China, in 1993 and the Ph.D degree in pattern recognition and intelligence system from the Xiandian University, Xi'an, China, in 2006. He is currently a professor in Guangxi University for Nationalities. His research interests include computation intelligence, neural networks, and intelligence information processing et al. He has published 1 book, and more than 150 research papers in journals.

Rui Wang, M.S., received his BS degree from Anqing Normal School of Computer Science and Technology, Anhui, China, in 2011. His current research interest is in computation intelligence, intelligence optimization.

Qifang Luo, Associate Prof. received his BS degree from Guangxi University of School of Computer and Electronis Information, Guangxi, China, in 1993. His current research interest is in computation intelligence, functional network.