Neural Comput & Applic (2014) 25:1369–1381
DOI 10.1007/s00521-014-1624-y

ORIGINAL ARTICLE

A novel complex-valued bat algorithm

Liangliang Li · Yongquan Zhou

Received: 5 November 2013 / Accepted: 15 May 2014 / Published online: 5 June 2014

Abstract Bat algorithm is a recent optimization algorithm with quick convergence, but its population diversity can be limited in some applications. This paper presents a new bat algorithm based on complex-valued encoding where the real part and the imaginary part will be updated separately. This approach can increase the diversity of the population and expands the dimensions for denoting. The simulation results of fourteen benchmark test functions show that the proposed algorithm is effective and feasible. Compared to the real-valued bat algorithm or particle swarm optimization, the proposed algorithm can get high precision and can almost reach the theoretical value.

Keywords Complex-valued encoding · Bat algorithm · Diploid · Test functions

1 Introduction

Swarm intelligence optimization algorithm originates from the simulation of various types of biological behavior in nature and has the characteristics of simple operation, strong parallelism, good optimization performance, etc. Inspired by this idea, the genetic algorithm (GA) [1], ant colony optimization (ACO) [2, 3], particle swarm optimization (PSO) [4] are proposed and are applied widely. In recent years, some new swarm intelligence algorithms were also proposed, such as the shuffled frog leaping algorithm (SFLA) [5], artificial bee colony optimization (ABC) [6], artificial fish swarm algorithm (AFSA) [7], cuckoo search (CS) [8], monkey algorithm (MA) [9] and firefly algorithm (FA), Glowworm swarm optimization algorithm (GSO) [10–13]. Swarm intelligence optimization algorithm can effectively solve some problems which traditional methods cannot solve and have shown excellent performance in many respects. So, its application scope has been greatly expanded.

L. Li · Y. Zhou (✉)
College of Information Science and Engineering, Guangxi University for Nationalities, Nanning 530006, Guangxi, China
e-mail: yongquanzhou@126.com

Y. Zhou
Guangxi Key Laboratory of Hybrid Computation and Integrated Circuit Design Analysis, Nanning 530006, Guangxi, China

The real-valued bat algorithm (BA) was proposed by Yang in 2010 [14, 15], which originated from the simulation of echolocation behavior in bats. Bats use a type of sonar called echolocation to detect prey, avoid obstacles in the dark. When searching their prey, the bats emit ultrasonic pulses. The loudness at this time is the maximum, which can help lengthen the ultrasonic propagation distance. During flight to the prey, loudness will decrease while the pulse emission will gradually increase, which can make the bat locate the prey more accurately. But the basic bat algorithm uses real number encoding method, and the application range is limited in the real number. So, the population diversity is limited, and the algorithm is easy to fall into the local optimum. In the low dimensional case, optimization performance is good [16–18], engineering optimization [19, 20], multi-objective optimization [21], and hybrid bat algorithm [22], but in the high dimensional case, optimization performance cannot be satisfactory. In order to solve the high space optimization problems in the basic bat algorithm based on the idea of complex diploid encoding [23–25], we present a bat algorithm based on the complex-valued encoding (CBA) in this paper. The idea of complex-valued encoding uses two parameters (i.e., the real part and the imaginary part) to represent a variable, and the real and imaginary parts can be updated in parallel.

The independent variables of the objective function are determined by the modules and angles of their corresponding complex numbers. So, the diversity of population is greatly enriched, the proposed CBA algorithm expands the dimensions for denoting and the performance of the algorithm is improved and CBA can expand the scope of application and basic theory of bat algorithm to certain extent and also provides a new way for bat algorithm to solve the practical problems. This paper is organized as follows. In the Sect. 2, the basic bat algorithm is described. Section 3 gives the CBA algorithm. The simulation and comparison of this proposed algorithm are presented in Sect. 4. Finally, some remarks and conclusions are provided in Sect. 5.

2 Bat algorithm

2.1 The velocity updating and position updating of the bat

Firstly, initialize the bat population randomly. Supposed the dimension of search space is n, the position of the bat i at time t is x_i^t and the velocity is v_i^t. Therefore, the position x_i^{t+1} and velocity v_i^{t+1} at time $t+1$ are updated by the following formulas:

$$Q_i^t = Q_{\min} + (Q_{\max} - Q_{\min})\beta \tag{1}$$

$$v_i^{t+1} = v_i^t + (x_i^t - \text{best})Q_i^t \tag{2}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{3}$$

where Q_i represents the pulse frequency emitted by bat i at the current moment. $Q_{\max}$ and $Q_{\min}$ represent the maximum and minimum values of pulse frequency, respectively, β is a random number in $[0, 1]$ and *best* represents the current global optimal solution.

Select a bat from the bat population randomly and update the corresponding position of the bat according to Eq. (4). This random walk can be understood as a process of local search, which produces a new solution by the chosen solution.

$$x_{\text{new}}(i) = x_{\text{old}} + \varepsilon A^t \tag{4}$$

where x_{old} represents a random solution selected from the current optimal solution, A^t is the loudness, ε is a random vector and its arrays are random values in $[-1,\ 1]$.

2.2 Loudness and pulse emission

Usually, at the beginning of the search, loudness is strong and pulse emission is small. When a bat has found its prey, the loudness decreases while pulse emission gradually increases. Loudness $A(i)$ and pulse emission $r(i)$ are updated according to Eqs. (5) and (6):

$$r^{t+1}(i) = r^0(i) \times [1 - \exp(-\gamma t)] \tag{5}$$

$$A^{t+1}(i) = \alpha A^t(i) \tag{6}$$

where α and γ are constants. For any $0 < \alpha < 1$, $\gamma > 0$, $A(i) = 0$ means that a bat has just found its prey and temporarily stop emitting any sound. It is not hard to find that as $t \to \infty$, we can get $A^t(i) \to 0$, $r^t(i) = r^0(i)$.

2.3 The implementation steps of bat algorithm

Generally speaking, the implementation steps of bat algorithm as follows:

1. Initialize the basic parameters: population size N, attenuation coefficient of loudness α, increasing coefficient of pulse emission γ, the maximum loudness A^0 and maximum pulse emission r^0 and the maximum number of iterations *iterMax*;
2. Define pulse frequency $Q_i \in [Q_{\min},\ Q_{\max}]$;
3. Initialize the bat population x_i and v;
4. Enter the main loop. If rand $< r_i$, update the velocity and the current position of the bat according to Eqs. (2) and (3). Otherwise, make a random disturbance for position of the bat and go to step 5;
5. If (rand$<A_i$,$f(x_i)<f(x)$), accept the new solutions and fly to the new position;
6. If $f(x_i)<f_{\min}$, replace the best bat and adjust the loudness and pulse emission according to Eqs. (5) and (6);
7. Evaluate the bat population and find out the best bat and its position;
8. If termination condition is met (i.e., reach maximum number of iterations or satisfy the search accuracy), go to step 9; otherwise, go to step 4 and execute the next search.
9. Output the best fitness values and global optimal solution.

3 Complex-valued bat algorithm (CBA)

3.1 The complex-valued encoding method

Compared with the traditional real number encoding method, complex-valued encoding has many advantages. It maps one-dimensional expression space with two-dimensional coding space. For each individual bat, the real and imaginary part of complex are updated separately which leads to an inherent parallelism and increases the diversity of individuals in the intangible. So, to some extent, the CBA has higher population diversity and overcomes the disadvantage of bat algorithm, which is easy to fall into local optimum. What's more, the application range of the bat algorithm is expanded to complex range. Because of the two-dimensional properties of complex number, CBA can express a higher dimension space.

3.1.1 Initialize the complex-valued encoding population

Based on the definition interval of the problem $[A_k, B_k]$, $k = 1, 2, \infty\ldots, 2M$, generate $2M$ complex modulus and $2M$ phase angle randomly.

$$\rho_k \in \left[0, \frac{A_k - B_k}{2}\right], \quad k = 1, 2, \ldots, 2M \tag{7}$$

$$\theta_k \in [-2\pi, 2\pi], \quad k = 1, 2, \ldots, 2M \tag{8}$$

According to the Eq. (9), get $2M$ complex number:

$$X_{Rk} + iX_{Ik} = \rho_k(\cos\theta_k + i\sin\theta_k), \quad k = 1, 2, \ldots, 2M \tag{9}$$

Thus, we obtain $2M$ real parts and $2M$ imaginary parts, and the real and imaginary parts are updated according to the following way.

3.1.2 The updating method of CBA

1. Update the real parts

$$V_R(t+1) = V_R(t) + (X_R(t) - best_1) * Q_1(t) \tag{10}$$

$$X_R(t+1) = X_R(t) + V_R(t+1) \tag{11}$$

2. Update the imaginary parts

$$V_I(t+1) = V_I(t) + (X_I(t) - \mathrm{best}_2) * Q_2(t) \tag{12}$$

$$X_I(t+1) = X_I(t) + V_I(t+1) \tag{13}$$

where the $V_R(t), V_I(t)$ are the bat speed of real and imaginary, $X_R(t), X_I(t)$ are the bat current position of real and imaginary, best_1 is the best solution of real parts and best_2 is the best solution of imaginary parts. $Q_1(t)$ and $Q_2(t)$ are the pulse frequency.

3.1.3 The calculation method of fitness value

Because the complex domain has two parts (i.e., the real and imaginary parts), when calculating the fitness value, the complex number needs to be converted into real number firstly and then calculates its fitness value. Specific practices are as follows:1. Take complex modulus as the value of the real number:

$$\rho_k = \sqrt{X_{Rk}^2 + X_{Ik}^2}, k = 1, 2, \ldots, M \tag{14}$$

2. The sign is determined by phase angle:

$$X_k = \rho_k \mathrm{sgn}\left(\sin\left(\frac{X_{Ik}}{\rho_k}\right)\right) + \frac{B_k + A_k}{2}, k = 1, 2, \ldots, M \tag{15}$$

where X_n represents the converted real variables.

3.2 CBA algorithm

The complex-valued encoding idea which can be considered as an efficient global optimization strategy is introduced to the bat algorithm. Based on the two-dimensional properties of the complex number, the real and imaginary parts of complex number are updated separately. This strategy can greatly enrich the diversity of population and enhance the global search ability of individual bat. Thus, the performance of the algorithm is improved greatly. When updating two new parameters, we also introduce the differential evolution strategy "DE/best/2/bin" [26, 27] to improve the local search ability of the algorithm. In this case, CBA can balance global and local search and cope with multimodal benchmarks. The pseudo code of CBA is as follows:

Pseudo code of CBA

```
1:  BEGIN
2:    Initialize the bat population: Let $\rho_k \in [0, \frac{A_k - B_k}{2}]$, $\theta_k \in [-2\pi, 2\pi]$;
3:      Get the real and imaginary part of the complex number [Eq.(9) ];
4:      Convert to real variables [Eq.(14) and Eq.(15) ];
5:      For all $X_i$ do
6:          Calculate fitness $F(X_i)$;
7:      End for
8:          Get the best solution;
9:          $iter \leftarrow 1$;
10:     While ( $iter < iterMax$ )
11:         Update the real part [Eq.(10) and Eq.(11) ];
12:         Update the imaginary part [Eq.(12) and Eq.(13) ];
13:         If ( $rand > r_i$ )
14:             Adjust the real and imaginary parts randomly;
15:         End If
16:             Reduce $A_i$ and increase $r_i$;
17:             Modify the real and imaginary part by "DE/best/2/bin";
18:             Select randomly $r_1 \neq r_2 \neq i$;
19:         If $rand < p_m$ then
20:             $X_{new} = X_{best} + F * (X_{r_1} - X_{r_2})$;
21:         End if
22:             Convert to real variables [Eq.(14) and Eq.(15) ];
23:             Calculate fitness $F(X_i)$;
24:             Get the best solution;
25:             $iter \leftarrow iter + 1$;
26:     End While
27:     Memorize the best solution achieved
28: END
```

4 Simulation experiments and results analysis

4.1 Simulation platform

The proposed algorithm is implemented in MATLAB. Operating system: Windows XP; CPU: AMD Athlon (tm) II X4 640 Processor, 3.01 GHz; RAM: 3 GB; Programming language: Matlab R2012 (a).

4.2 Benchmark functions

In order to verify the effectiveness of the proposed algorithm, we select fourteen standard benchmark functions [28, 29] to detect the searching capability of the proposed algorithm. Each function has its own characteristics, and a single algorithm cannot apply to every benchmark function. Therefore, the experimental results can fully reflect the performance of the algorithm.

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_1, F_2, F_3, F_4 and F_5 for category I, F_6, F_7, F_8 and F_9 for category II and F_{10}, F_{11}, F_{12}, F_{13} and F_{14} for category III. In high-dimensional functions, F_2 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_6. And in low-dimensional functions, most functions have the characteristic of strong shocks. As the global minimum of most benchmark functions is zero, in order to verify the searching capability of the algorithm effectively, we select some functions with nonzero global minimum.

4.3 Parameter setting

Generally, the choice of parameters requires some experimenting. In this paper, after a lot of experimental comparison, the parameters of the algorithm are set as follows.

In BA, the parameters are generally set as follows: Pulse frequency range is $Q_i \in [0,\ 2]$, the maximum loudness is $A^0 = 0.5$, maximum pulse emission is $r^0 = 0.5$, attenuation coefficient of loudness is $\alpha = 0.95$, increasing coefficient of pulse emission is $\gamma = 0.05$ and population size is $N = 40$.

In PSO, we use linear decreasing inertia weight that is $\omega_{\max} = 0.9$, $\omega_{\min} = 0.4$, and learning factor is $C_1 = C_2 = 1.4962$.

In CBA, the basic parameters are the same with BA. The range of complex modulus is $\rho_k \in [0, \frac{A_k - B_k}{2}]$, the range of phase angle is $\theta_k \in [-2\pi, 2\pi]$, where $[A_k, B_k]$ is the range of variables.

In the tests, the maximum number of iterations of each algorithm is $iterMax = 500$.

4.4 Comparison of experiment results

For standard benchmark functions in Table 1, the comparison of test results is shown in Tables 2, 3 and 4, while

Table 1 Benchmark functions

Category	No.	Name	Benchmark functions	D	Scope	$f_{\min}$
I	F_1	Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	30	$[-5.12, 5.12]$	0
	F_2	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} [(x_i - 1)^2 + 100(x_i^2 - x_{i+1})^2]$	30	$[-2.048, 2.048]$	0
	F_3	Step	$f(x) = \sum_{i=1}^{n-1} (\lfloor xi + 0.5 \rfloor)^2$	30	$[-100, 100]$	0
	F_4	Quartic	$f(x) = \sum_{i=1}^{n} x_i^4 + \text{random}[0, 1)$	30	$[-1.28, 1.28]$	0
	F_5	X. S. Yang-7	$f(x) = \sum_{i=1}^{n} \varepsilon_j \lvert x_i - 1/i \rvert,\ \varepsilon_i \in U[0, 1]$	30	$[-5, 5]$	0
II	F_6	Rastrigin	$f(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	0
	F_7	Ackley	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)\right) + 20 + e$	30	$[-32.768, 32.768]$	0
	F_8	Griewank	$f(x) = \frac{1}{4{,}000}\sum_{i=1}^{n} (x_i^2) - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]$	0
	F_9	Alpine	$f(x) = \sum_{i=1}^{n} x_i \sin(x_i) + 0.1x_i$	30	$[-10, 10]$	0
III	F_{10}	Schaffer's F6	$f(x) = \frac{\sin^2\sqrt{(x_1^2 + x_2^2)} - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2} - 0.5$	2	$[-100, 100]$	−1
	F_{11}	Drop Wave	$f(x) = -\frac{1 + \cos(12\sqrt{x_1^2 + x_2^2})}{{}^1\!/\!{}_2(x_1^2 + x_2^2) + 2}$	2	$[-5.12, 5.12]$	−1
	F_{12}	Shekel 1	$f(x) = -\sum_{i=1}^{5} ((x - a_i)(x - a_i)^T + c_i)^{-1})$	4	$[0, 10]$	10.1532
	F_{13}	Shekel 2	$f(x) = -\sum_{i=1}^{7} ((x - a_i)(x - a_i)^T + c_i)^{-1}$	4	$[0, 10]$	−10.4029
	F_{14}	Shekel 3	$f(x) = -\sum_{i=1}^{10} ((x - a_i)(x - a_i)^T + c_i)^{-1}$	4	$[0, 10]$	−10.5364

Table 2 Simulation results for test functions $F_i, i = 1, 2, 3, 4, 5$

Benchmark functions	Method	Results			
		Best	Mean	Worst	Std.
$F_1(D = 30)$	PSO	0.848903879	2.019502887	3.543298331	0.882332983
	BA	0.001241624	0.001441276	0.001680063	1.13E−04
	CBA	***0***	***0***	***0***	***0***
$F_2(D = 30)$	PSO	36.63927586	70.69199924	1.14E+02	19.0223137
	BA	23.72699513	28.05953865	29.51861835	1.476970852
	CBA	***0.001219204***	***0.191232376***	***3.765215505***	***0.841231155***
$F_3(D = 30)$	PSO	345	8.60E+02	1775	3.72E+02
	BA	49575	6.27E+04	71257	5.96E+03
	CBA	***0***	***0***	***0***	***0***
$F_4(D = 30)$	PSO	0.034431114	0.120536812	0.262638954	0.061560069
	BA	0.08291352	0.149113591	0.217150154	0.032129523
	CBA	***2.41E−06***	***6.18E−05***	***1.68E−04***	***5.31E−05***
$F_5(D = 30)$	PSO	4.189661137	7.345693455	11.87075461	2.309022597
	BA	0.561140026	0.71553531	2.715762867	0.66129969
	CBA	***0.438096172***	***0.691105745***	***0.831076592***	***0.104928892***

Table 3 Simulation results for test functions $F_i, i = 6, 7, 8, 9$

Benchmark functions	Method	Results			
		Best	Mean	Worst	Std
$F_6(D = 30)$	PSO	59.31426674	1.07E+02	1.72E+02	23.13294355
	BA	1.46E+02	1.99E+02	2.63E+02	30.87488147
	CBA	***0***	***0***	***0***	***0***
$F_7(D = 30)$	PSO	4.315029312	6.569731545	8.927045465	1.302036553
	BA	18.80506661	19.13267851	19.62574603	0.220545362
	CBA	***8.88E−16***	***8.88E−16***	***8.88E−16***	***0***
$F_8(D = 30)$	PSO	1.696523568	2.829749864	5.047220552	1.090576403
	BA	3.53E+02	5.48E+02	6.66E+02	76.99297871
	CBA	***0***	***0***	***0***	***0***
$F_9(D = 30)$	PSO	0	1.49E−41	1.43E−40	3.69E−41
	BA	3.68E−10	8.13E−07	5.56E−06	1.64E−06
	CBA	0	***0***	***0***	***0***

comparison of searching success rate is shown in Table 5. In this paper, the results are obtained in twenty 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 CBA is better, while underlined results mean that other algorithm is better.

Seen from Table 2, in category I, CBA can find the optimal solution for F_1 and F_3 and has a very strong robustness. For other three functions, the precision of optimal fitness value and mean fitness value is higher than those of PSO and BA. For the five functions in category I, standard deviation of CBA is less than that of PSO and BA. And this means that in the optimization of high-dimensional unimodal function, CBA has better stability. Similarly, seen from Table 3, besides F_7, for other functions in category II, CBA can find out the optimal solution, and all the standard deviation is 0. Even for F_7, CBA has a higher precision of optimization. The accuracy of CBA can be higher than that of PSO and BA for 16, 17 orders of magnitude, respectively. Overall, the optimization performance of CBA is superior to PSO and BA in the high-dimensional case.

The optimal value of functions in category III is non-zero. Seen from the results in Table 4, although in functions F_{10}, F_{11} and F_{12}, both PSO and CBA can find out the optimal solution, but the mean optimal value and standard deviation of CBA are smaller than those of the PSO. Only for F_{14}, the optimal value of CBA is slightly worse than PSO, while standard deviation of CBA is slightly larger

Table 4 Simulation results for test functions $F_i, i = 10, 11, 12, 13, 14$

Benchmark functions	Method	Results			
		Best	Mean	Worst	Std.
$F_{10}(D = 2)$	PSO	−1	−0.996599432	−0.99028409	0.004754582
	BA	−0.962775925	−0.71578852	−0.528385341	0.131553488
	CBA	−1	***−1***	***−1***	***0***
$F_{11}(D = 2)$	PSO	−1	−0.996812266	−0.936245328	0.014255978
	BA	−0.999999932	−0.732945647	−0.477784428	0.179212441
	CBA	−1	***−1***	***−1***	***0***
$F_{12}(D = 4)$	PSO	−10.15319968	−8.267267062	−2.630471668	3.033997517
	BA	−10.15311639	−4.895061995	−2.630453789	3.232005155
	CBA	−10.15319968	**−10.15319964**	**−10.15319953**	**3.86E−08**
$F_{13}(D = 4)$	PSO	−10.40294057	−9.491619079	−2.765897328	2.268580502
	BA	−10.40286341	−5.861470277	−1.837589128	3.552600389
	CBA	***−10.4029401***	***−10.40293146***	***−10.40289599***	***1.19E−05***
$F_{14}(D = 4)$	PSO	−10.53640982	−9.003684191	−2.421734027	3.156219888
	BA	−10.53626178	−4.232217299	−1.676545005	2.852811947
	CBA	***−10.53640972***	***−9.112674411***	−2.421733874	2.933412724

Table 5 Comparison of the optimization success rate

Category	Benchmark functions	D	PSO	BA	CBA
I	F_1	10	85.345 %	95.8 %	***1***
	F_2	10	0	0	***67.5 %***
	F_3	10	18.935 %	0	***1***
	F_4	10	0.085 %	2.09 %	***99.9 %***
	F_5	10	0	0	0
II	F_6	10	0	0	***1***
	F_7	10	0	0	***1***
	F_8	10	0	0	***1***
	F_9	10	99.885 %	99.975 %	***1***
III	F_{10}	2	98.03 %	9.975 %	***1***
	F_{11}	2	0.94 %	5 %	***99.99 %***
	F_{12}	4	22.39 %	19.73 %	***99.1 %***
	F_{13}	4	58.905 %	29.62 %	***99.9 %***
	F_{14}	4	36.16 %	24.7 %	***83.8 %***

than the BA. But the mean optimal value of CBA is better than PSO and BA. Obviously, for functions in category III, the optimization performance of CBA is still better than PSO and BA.

For functions in three categories, Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 and 14 are the fitness evolution curve, Figs. 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27 and 28 are the anova tests of the global minimum and Figs. 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41 and 42 are the comparisons of optimal fitness value.

If the error between actual optimization and theoretical optimal value is less than 1 %, count as a successful search

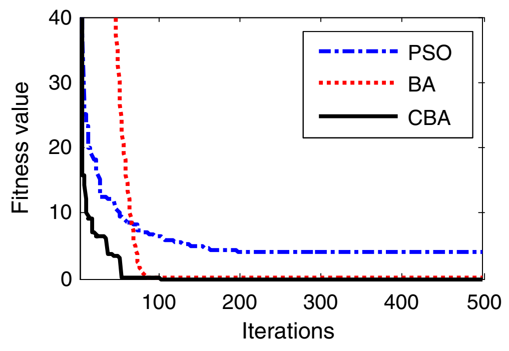

Fig. 1 $D = 30$, evolution curves of fitness value for F_1

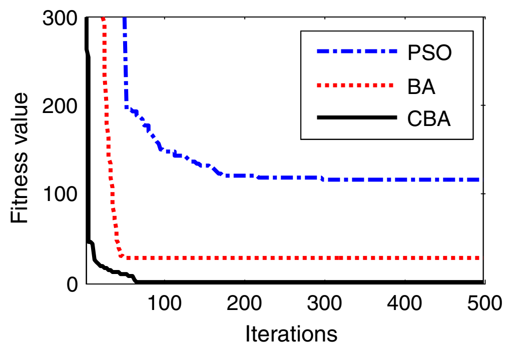

Fig. 2 $D = 30$, evolution curves of fitness value for F_2

to the optimal value. In order to fully validate the effectiveness of the algorithm, we made a statistical analysis of the optimization success rate of each algorithm. Table 5 shows that optimization success rate of each algorithm, where dimension of functions in category I and category II

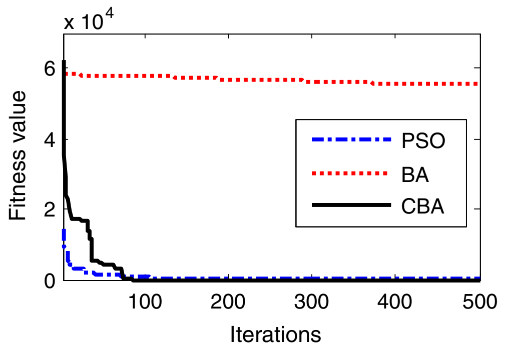

Fig. 3 $D = 30$, evolution curves of fitness value for F_3

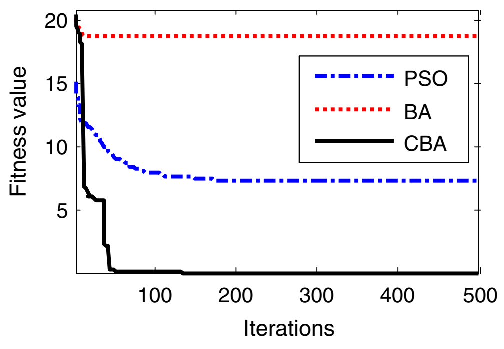

Fig. 7 $D = 30$, evolution curves of fitness value for F_7

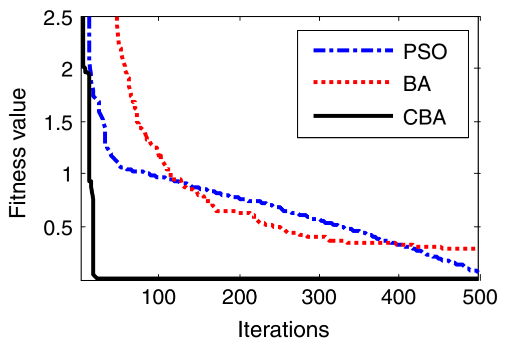

Fig. 4 $D = 30$, evolution curves of fitness value for F_4

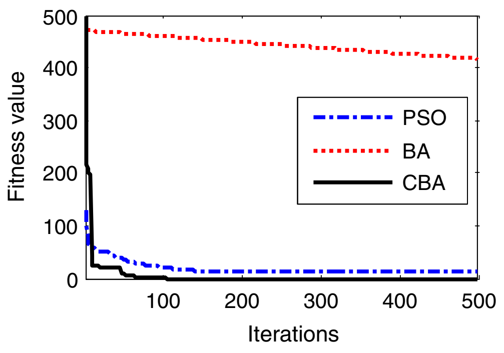

Fig. 8 $D = 30$, evolution curves of fitness value for F_8

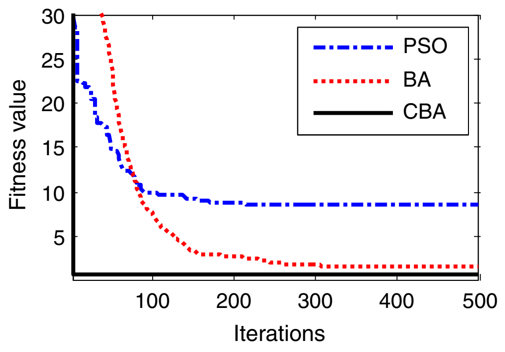

Fig. 5 $D = 30$, evolution curves of fitness value for F_5

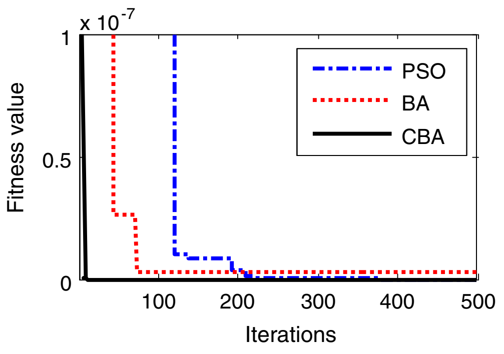

Fig. 9 $D = 30$, evolution curves of fitness value for F_9

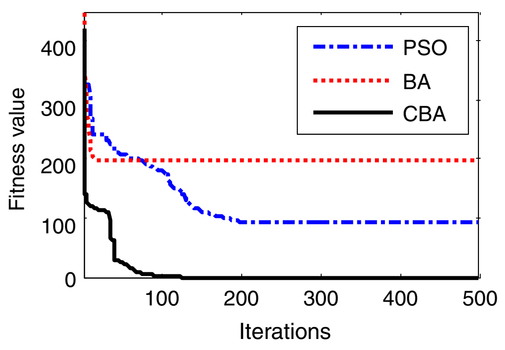

Fig. 6 $D = 30$, evolution curves of fitness value for F_6

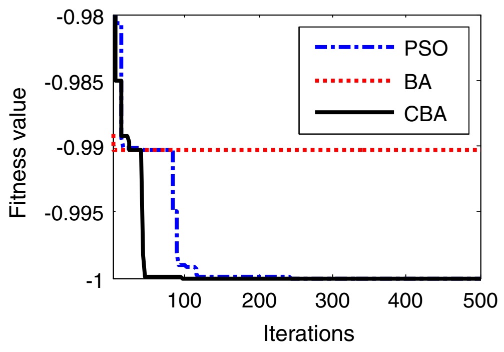

Fig. 10 $D = 2$, evolution curves of fitness value for F_{10}

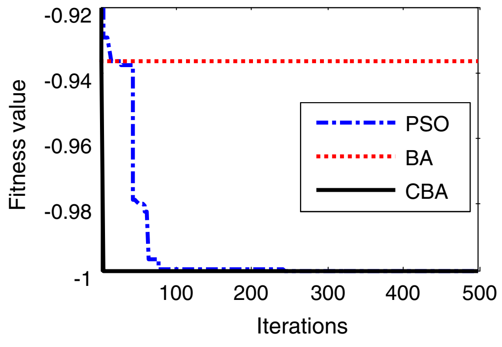

Fig. 11 $D = 2$, evolution curves of fitness value for F_{11}

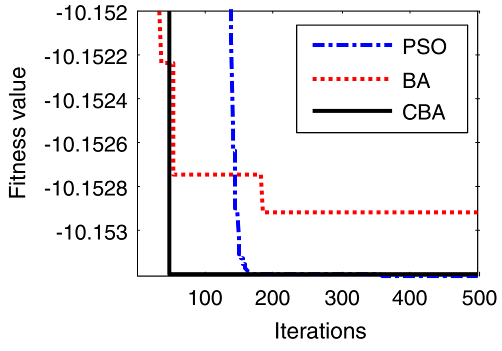

Fig. 12 $D = 4$, evolution curves of fitness value for F_{12}

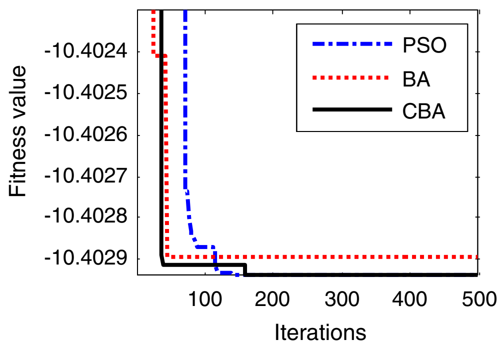

Fig. 13 $D = 4$, evolution curves of fitness value for F_{13}

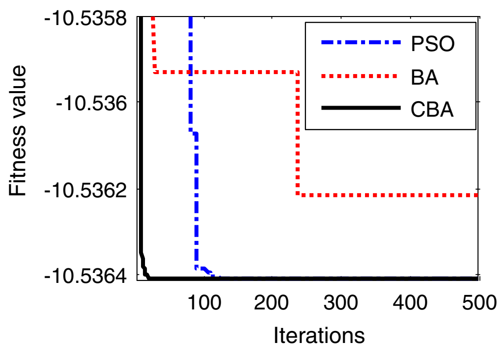

Fig. 14 $D = 4$, evolution curves of fitness value for F_{14}

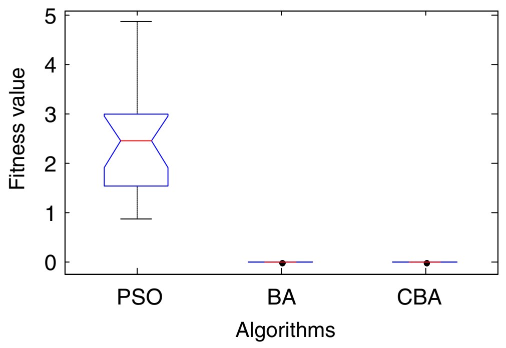

Fig. 15 $D = 30$, anova tests of the global minimum for F_1

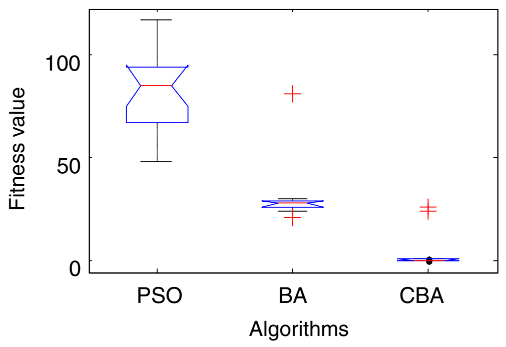

Fig. 16 $D = 30$, anova tests of the global minimum for F_2

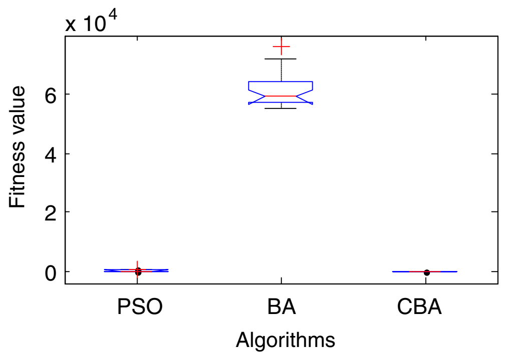

Fig. 17 $D = 30$, anova tests of the global minimum for F_3

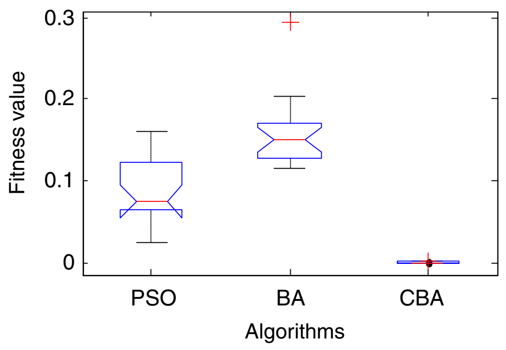

Fig. 18 $D = 30$, anova tests of the global minimum for F_4

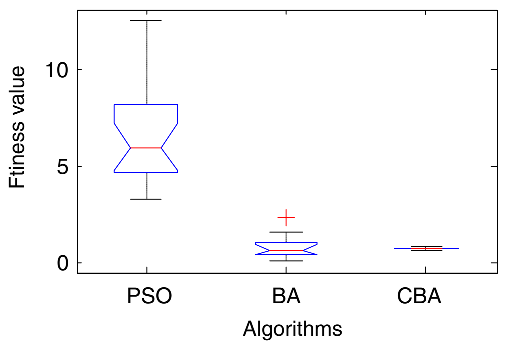

Fig. 19 $D = 30$, anova tests of the global minimum for F_5

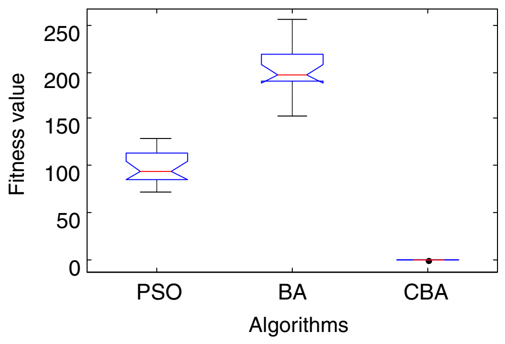

Fig. 20 $D = 30$, anova tests of the global minimum for F_6

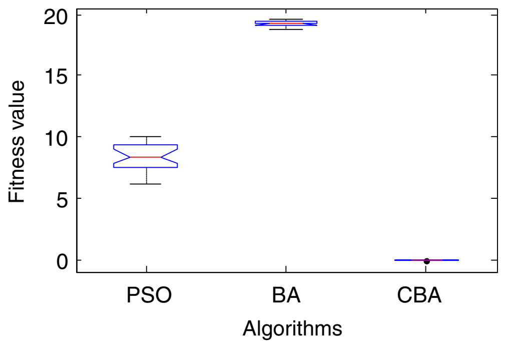

Fig. 21 $D = 30$, anova tests of the global minimum for F_7

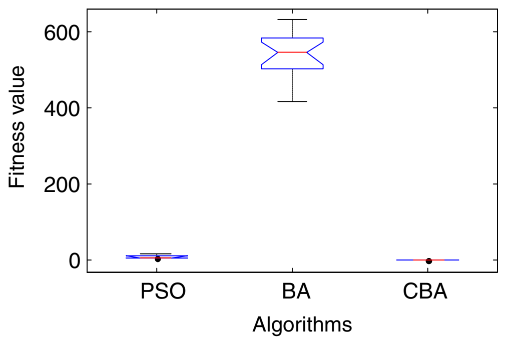

Fig. 22 $D = 30$, anova tests of the global minimum for F_8

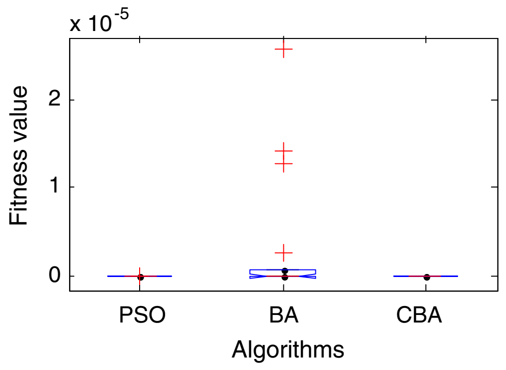

Fig. 23 $D = 30$, anova tests of the global minimum for F_9

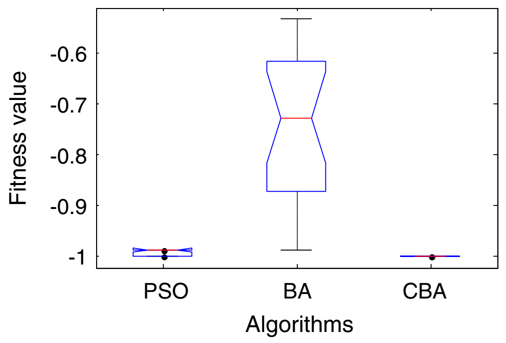

Fig. 24 $D = 2$, anova tests of the global minimum for F_{10}

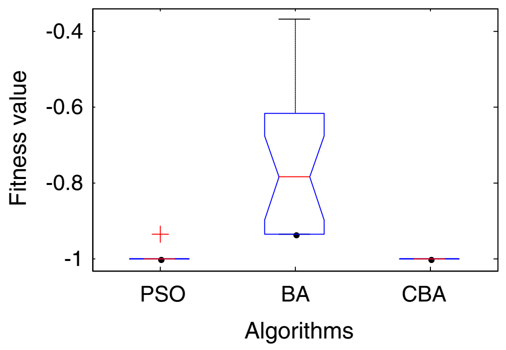

Fig. 25 $D = 2$, anova tests of the global minimum for F_{11}

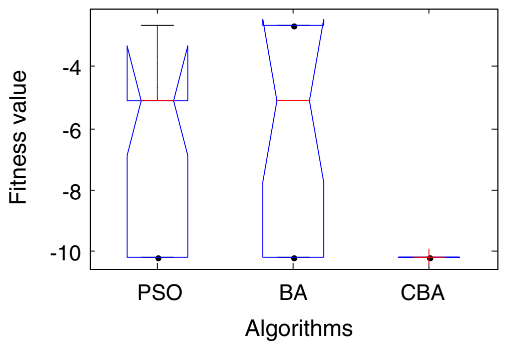

Fig. 26 $D = 4$, anova tests of the global minimum for F_{12}

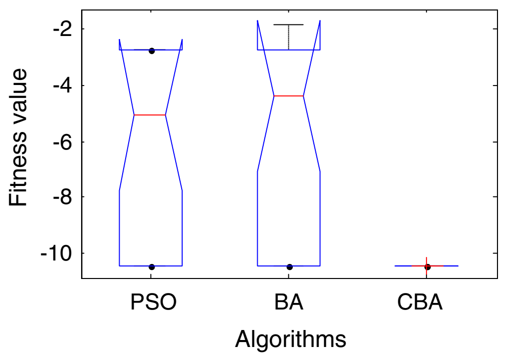

Fig. 27 $D = 4$, anova tests of the global minimum for F_{13}

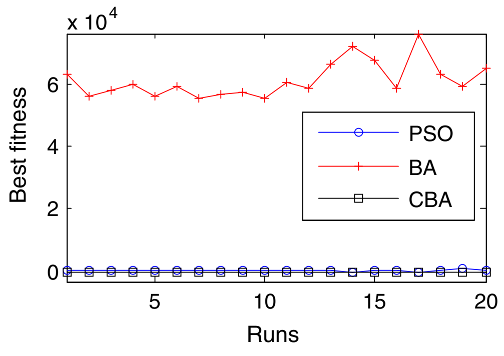

Fig. 31 $D = 30$, comparison of optimal fitness value for F_3

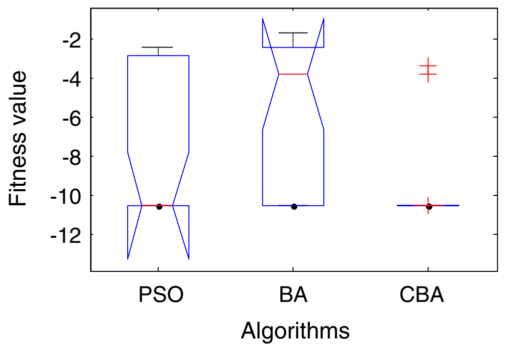

Fig. 28 $D = 4$, anova tests of the global minimum for F_{14}

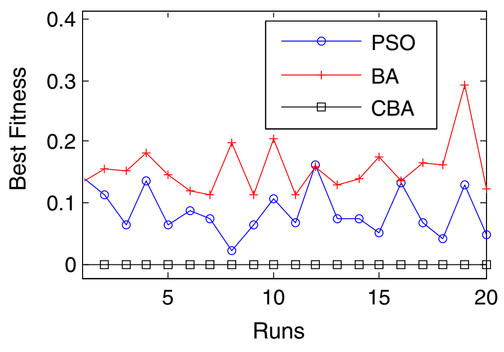

Fig. 32 $D = 30$, comparison of optimal fitness value for F_4

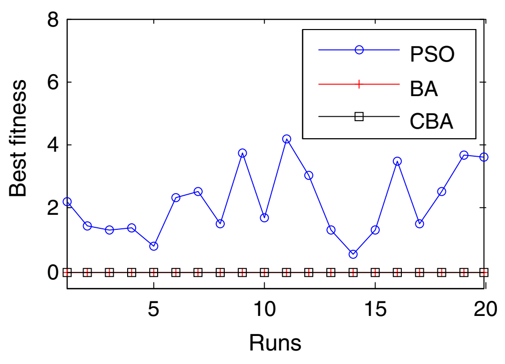

Fig. 29 $D = 30$, comparison of optimal fitness value for F_1

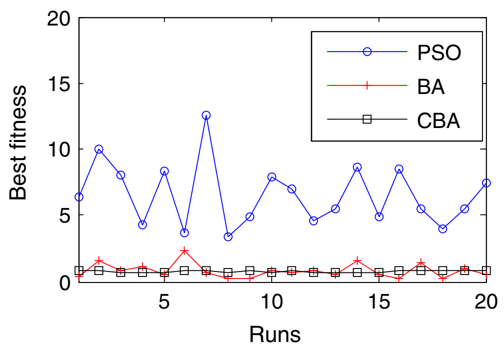

Fig. 33 $D = 30$, comparison of optimal fitness value for F_5

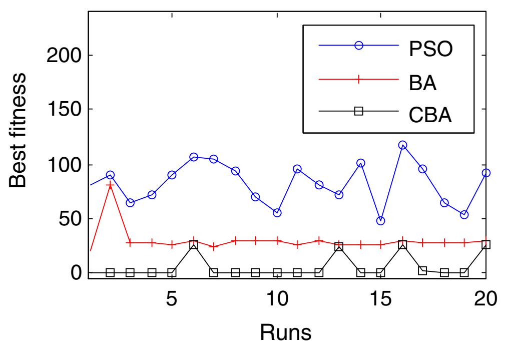

Fig. 30 $D = 30$, comparison of optimal fitness value for F_2

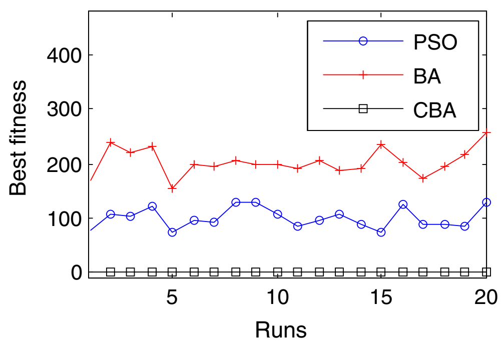

Fig. 34 $D = 30$, comparison of optimal fitness value for F_6

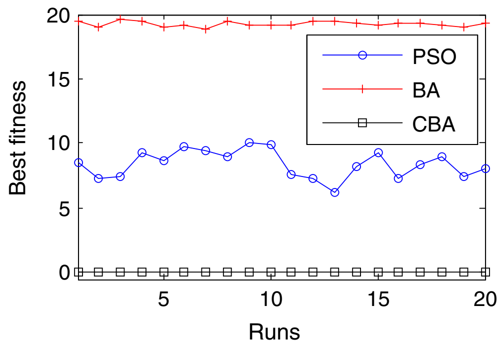

Fig. 35 $D = 30$, comparison of optimal fitness value for F_7

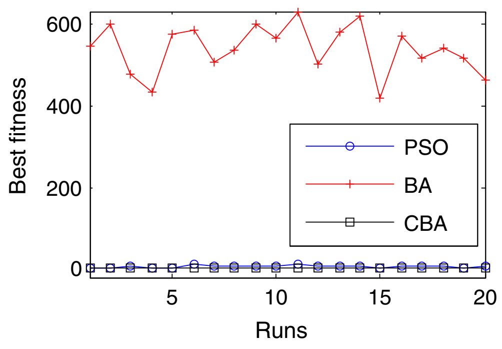

Fig. 36 $D = 30$, comparison of optimal fitness value for F_8

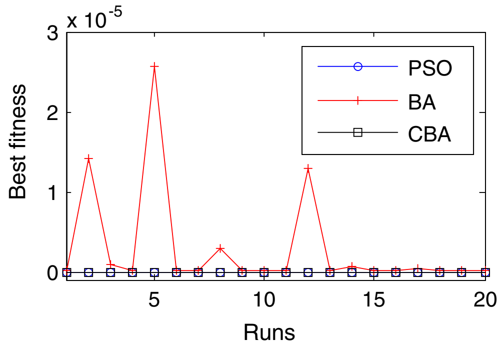

Fig. 37 $D = 30$, comparison of optimal fitness value for F_9

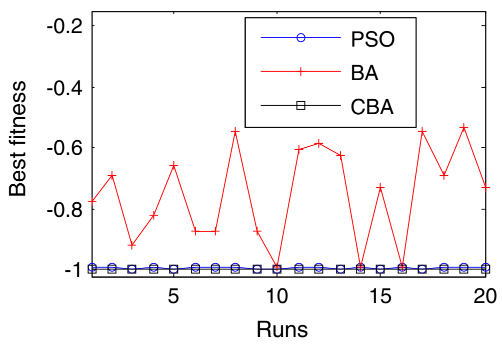

Fig. 38 $D = 2$, comparison of optimal fitness value for F_{10}

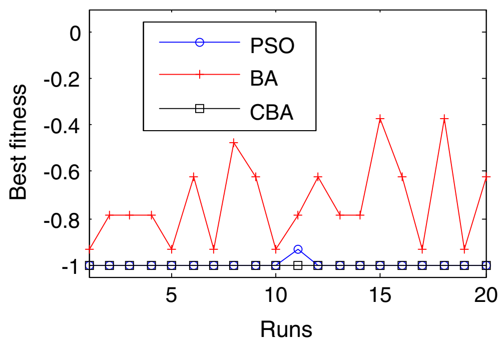

Fig. 39 $D = 2$, comparison of optimal fitness value for F_{11}

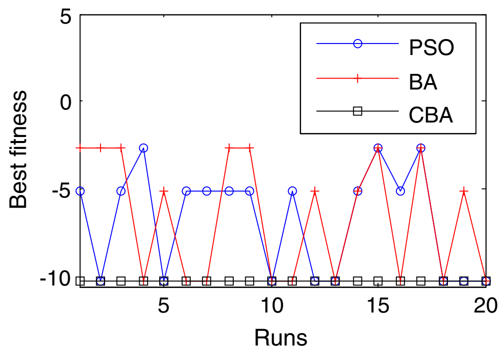

Fig. 40 $D = 4$, comparison of optimal fitness value for F_{12}

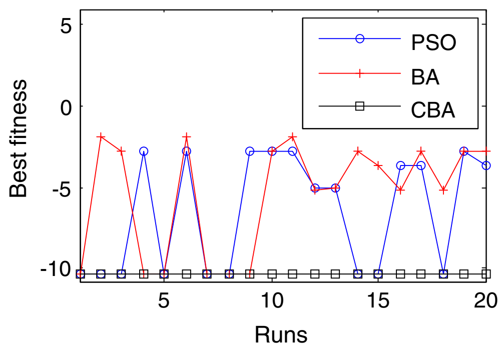

Fig. 41 $D = 4$, comparison of optimal fitness value for F_{13}

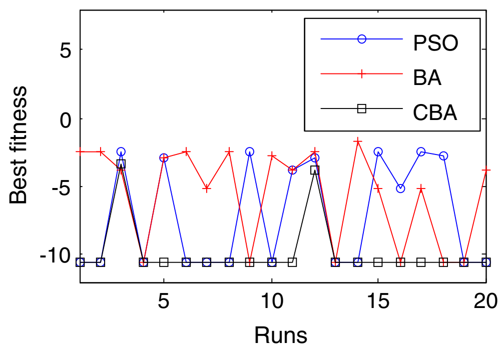

Fig. 42 $D = 4$, comparison of optimal fitness value for F_{14}

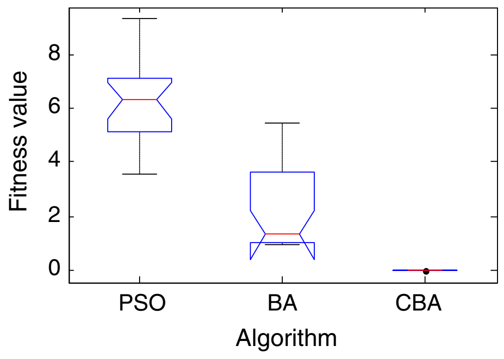

Fig. 43 $D = 10$, anova tests of the global minimum for F_2

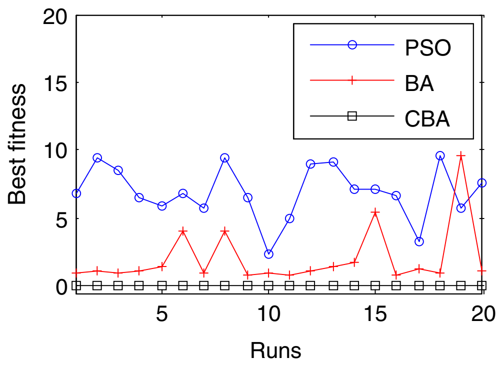

Fig. 45 $D = 10$, comparison of optimal fitness value for F_2

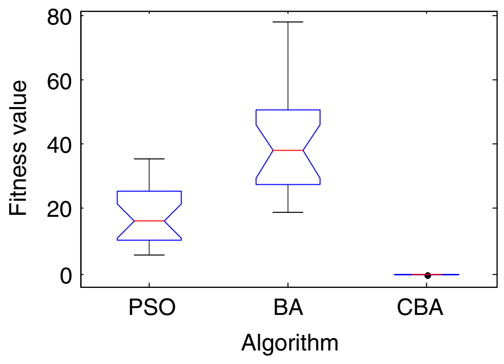

Fig. 44 $D = 10$, anova tests of the global minimum for F_6

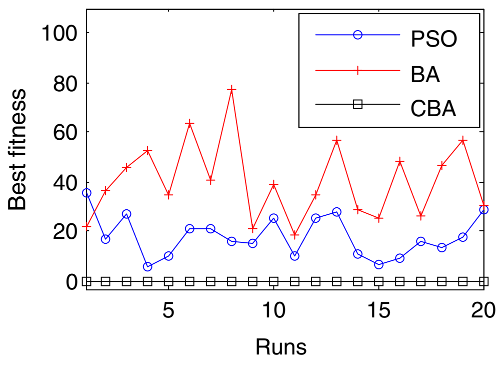

Fig. 46 $D = 10$, comparison of optimal fitness value for F_6

is 10. Figures 43 and 44 are the anova tests of the global minimum for F_2 and F_6 in the 10 dimensional case, and Fig. 45, Fig. 46 is comparison of optimal fitness value for F_2 and F_6 in the 10 dimensional case.

Because the optimization success rate of PSO and BA in high-dimensional search is very low, so the dimension for functions in category I and category II is set to 10. It can be seen from Table 5, except that the optimization success rate of CBA for F_5 is 0, the optimization success rate of CBA for other functions is higher than those of PSO and BA. In particular, for functions F_6, F_7 and F_8, CBA can certainly find out the optimal solution, while optimization success rate of PSO and BA is 0 and cannot find a satisfactory result. The anova function tests of the global minimum in Figs. 43 and 44 and comparison of optimal fitness value in Fig. 45, Fig. 46 also show that the CBA has better stability and higher precision of optimization.

5 Conclusions

There are many shortcomings of bat algorithm, such as poor population diversity, low precision of optimization, easy to fall into local optimum and poor optimization performance in high-dimensional case. This paper introduces the idea of complex-valued encoding into bat algorithm and proposes a novel bat algorithm based on complex-valued encoding (CBA). With the unique two-dimensional characteristics of complex number, the proposed algorithm increases the diversity of population and improves the optimization performance of the algorithm. In this article, we tested fourteen typical benchmark functions. The results of comparison with the PSO and BA show that precision of optimization, convergence speed and robustness of CBA are all better than PSO and BA. The results of simulation test show that the proposed algorithm is effective and feasible.

Acknowledgments This work is supported by National Science Foundation of China under Grant No. 61165015. Key Project of Guangxi Science Foundation under Grant No. 2012GXNSF DA053028, Key Project of Guangxi High School Science Foundation under Grant Nos. 20121ZD008, 201203YB072, funded by Open Research Fund Program of Key Lab of Intelligent Perception and Image Understanding of Ministry of Education of China under Grant No. IPIU01201100 and the Innovation Project of Guangxi Graduate Education under Grant No. gxun-chx2012103.

References

1. Srinivas M, Patnaik LM (1994) Adaptive probabilities of crossover and mutation in genetic algorithm. IEEE Trans Syst Man Cybern 24(4):656–667

2. Kennedy J, Eberhart RC, Shi Y (2001) Swarm intelligence. Morgan Kaufman, San Francisco
3. Coloni A, Dorigo M, Maniezzo V (1996) Ant system: optimization by a colony of cooperating agent. IEEE Trans Syst Man Cybern Part B Cybern 26(1):29–41
4. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks. IEEE Press, Piscataway, NJ, pp 1942–1948
5. Wen-hua Cui, Xiao-bing Liu, Wei Wang, Jie-sheng Wang (2012) Survey on shuffled frog leaping algorithm. Control Decis 27(4):481–486
6. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459–471
7. Xiao-lei Li, Zhi-jiang Shao, Ji-xin Qian (2002) An optimizing method based on autonomous animals: fish-swarm algorithm. Syst Eng Theory Pract 22(11):32–38 (in Chinese)
8. Yang XS, Deb S (2009) Cuckoo search via Lévy Flights. In: Proceedings of world congress on nature & biologically inspired computing. IEEE Press, Coimbatore, pp 210–214
9. Rui-qing Zhao, Wan-sheng Tang (2008) Monkey algorithm for global numerical optimization. J Uncertain Syst 2(3):164–175
10. Yang XS (2009) Firefly algorithms for multimodal optimization. Stoch Algorithms Found Appl Lect Notes Comput Sci 5792:169–178
11. Krishnanand KN, Ghose D (2009) Glowworm swarm optimization: a new method for optimizing multi-modal functions. Int J Comput Intell Stud 1(1):93–119
12. Zhou Y, Zhou G, Zhang J (2013) A hybrid glowworm swarm optimization algorithm to solve constrained multimodal functions optimization. Optimization 1–24 (published online)
13. Zhou Y, Luo Q, Liu J (2013) Glowworm swarm optimization for dispatching system of public transit vehicles. Neural Process Lett 1–9 (published online)
14. Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: Gonzalez JR et al (eds) Nature inspired cooperative strategies for optimization, NICSO 2010, SCI:284, pp 65–74
15. Yang XS (2011) Nature-inspired metaheuristic algorithms. Luniver Press, Frome
16. Zhou Y, Xie J, Zheng H (2013) A hybrid bat algorithm with path relinking for capacitated vehicle routing Problem. Math Probl Eng 2013:2013
17. Xie J, Zhou Y, Zheng H (2013) A hybrid metaheuristic for multiple runways aircraft landing problem based on bat algorithm. J Appl Math 2013:2013
18. Gandomi AmirHossein, Yang Xin-She, Alavi AmirHossein, Talatahari Siamak (2013) Bat algorithm for constrained optimization tasks. Neural Comput Appl 22:1239–1255
19. Yang XS, Gandomi AH (2013) Bat algorithm: a novel approach for global engineering optimization. Eng Comput 29(5):464–483
20. Kaveh A, Zakian P (2014) Enhanced bat algorithm for optimal design of skeletal structures. Asian J Civial Eng 15(2):179–212
21. Yang XS (2011) Bat algorithm for multi-objective optimisation. Int J Bio-Inspired Comput 3(5):267–274
22. He X-s, Ding W-j, Yang X-s (2013) Bat algorithm based on simulated annealing and gaussian perturbations. Neural Comput Appl (published online)
23. Casasent D, Natarajan S (1995) A classifier neural network with complex-valued weights and square-law nonlinearities. Neural Netw 8(6):989–998
24. De-bao Chen, Huai-jiang Li, Zheng Li (2009) Particle swarm optimization based on complex-valued encoding and application in function optimization. Comput Eng Appl 45(10):59–61 (in Chinese)
25. Zhao-hui Zheng, Yan Zhang, Yu-huang Qiu (2003) Genetic algorithm based on complex-valued encoding. Control Theory Appl 20(1):97–100 (in Chinese)
26. Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31
27. Pei-chong Wang, Qian Xu, Yue W (2009) Overview of differential evolution algorithm. Comput Eng Appl 45(28):13–16 (in Chinese)
28. Yang XS (2010) Appendix a: Test problems in optimization. In: Yang XS (ed) Engineering optimization. John, Hoboken, pp 261–266
29. Tang K, Yao X, Suganthan PN et al (2007) Benchmark functions for the CEC' 2008 special session and competition on large scale global optimization. University of Science and Technology of China, Hefei