Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 3017608, 13 pages
https://doi.org/10.1155/2017/3017608

Research Article

Improved Backtracking Search Algorithm Based on Population Control Factor and Optimal Learning Strategy

Lei Zhao,[1] Zhicheng Jia,[1] Lei Chen,[2,3] and Yanju Guo[1]

[1]*School of Electronic Information Engineering, Hebei University of Technology, Tianjin 300401, China*
[2]*School of Information Engineering, Tianjin University of Commerce, Tianjin 300134, China*
[3]*School of Precision Instrument and Opto-Electronics Engineering, Tianjin University, Tianjin 300072, China*

Correspondence should be addressed to Lei Chen; chenlei@tjcu.edu.cn

Received 3 January 2017; Revised 10 May 2017; Accepted 12 June 2017; Published 24 July 2017

Academic Editor: Erik Cuevas

Backtracking search algorithm (BSA) is a relatively new evolutionary algorithm, which has a good optimization performance just like other population-based algorithms. However, there is also an insufficiency in BSA regarding its convergence speed and convergence precision. For solving the problem shown in BSA, this article proposes an improved BSA named COBSA. Enlightened by particle swarm optimization (PSO) algorithm, population control factor is added to the variation equation aiming to improve the convergence speed of BSA, so as to make algorithm have a better ability of escaping the local optimum. In addition, enlightened by differential evolution (DE) algorithm, this article proposes a novel evolutionary equation based on the fact that the disadvantaged group will search just around the best individual chosen from previous iteration to enhance the ability of local search. Simulation experiments based on a set of 18 benchmark functions show that, in general, COBSA displays obvious superiority in convergence speed and convergence precision when compared with BSA and the comparison algorithms.

1. Introduction

In the rapid development of science and technology today, people set a high demand for optimization algorithms, and evolutionary algorithms are popular with people owing to its simple structure and less solving parameters, such as particle swarm optimization (PSO) algorithm [1], ant colony optimization (ACO) algorithm [2], and genetic algorithm (GA) [3]. However, a higher request is set for optimization algorithms when the multidimensional, multimodal, and nonlinear problems arise in academic research. So many scholars presented some modified evolutionary algorithms aiming to find the best values for system's parameters in different circumstances in recent decades. By taking advantage of the optimum solution, literature [4] mends the search equation when PSO algorithm operates aiming to accelerate convergence speed. Enlightened by biological evolution, literature [5] developed differential evolution (DE) algorithm, and an adaptive evolutionary strategy based on DE is proposed in literature [6] to raise efficiency of DE algorithm. Differential search algorithm (DSA), equipping the new crossover and mutation strategy, is revealed in the literature [7]. Learning from natural system, literatures [8–10] put forward artificial bee colony (ABC) algorithm enlightened by the natural behavior of bee, cuckoo search (CS) algorithm enlightened by flight behavior of cuckoo, and bat algorithm (BA) enlightened by foraging behavior of bat. Besides, these evolutionary algorithms have found increasingly wide utilization in many fields successfully, such as blind source separation [11], hyperspectral unmixing [12], and image processing [13].

BSA is such a novel computation method evolved by Civicioglu in 2013 [14]. BSA employs fewer parameters and has a simpler process that is efficient, fast, and able to handle various conditions and that makes it more easily to accept different complex problems. Literature [14] demonstrates that BSA has similar superiority to other evolutionary algorithms with strength of global search. Owing to its simplicity and high-efficiency, BSA has caught many scholars' attention [15, 16].

However, there are also some challenging complications arisen in BSA. For example, the convergence speed of BSA

is typically slower than these typical evolutionary algorithms (DE and PSO) in the early stage as a result of ignoring the importance of optimal individual that makes BSA difficult to achieve the satisfactory outcome when handling some complex problems. Therefore, accelerating convergence speed and improving convergence precision have become two relatively valuable and significative targets in BSA's study. A number of modified BSA aiming to reach above two targets have, hence, been put forward since its appearance. For example, literature [17] focuses on improving convergence precision by proposing the optimal evolution equation and optimal search equation but brings extra evaluations. Literatures [18, 19] focus on convergence precision and globe convergence by employing variation coefficient and selection mechanism but sacrifice iteration numbers as a result. An improved BSA is proposed in this article following the purpose of accelerating convergence speed and improving convergence precision when iteration numbers are lower. Firstly, to accelerate the convergence speed of population in the early stage, population control factor is added to the evolution equation. What is more, an evolutionary equation based on the fact that the disadvantaged group will search just around the best individual chosen from previous iteration is proposed to enhance the ability of local search. Simulation experiments based on 18 benchmark functions show that the improved algorithm can improve performance and productivity of BSA effectively and is a feasible evolutionary algorithm.

The rest of this article is arranged as follows. This article will describe BSA briefly in the second section; the improved algorithm will be introduced in the third section; in the fourth section, the simulation results will be presented; the fifth section is the discussion about the variance q appearing in the first strategy; in the last, this article will draw a conclusion.

2. Backtracking Search Algorithm

Similar to other evolutionary algorithms, BSA is also a population-based evolutionary algorithm designed to find a global optimum; the solving process can be described by dividing its operation into five steps as is done in other evolutionary algorithms: population initialization, historical population setting, population evolution, population crossover, and greedy selection. Specific steps of the basic BSA can be described as follows.

2.1. Population Initialization. N numbers of randomly emerged D-dimensional vectors make up the initial solution of BSA. And the random initialization method can be summarized as follows:

$$\mathrm{Pop}_{i,j} \sim U\left(\mathrm{low}_j, \mathrm{up}_j\right), \tag{1}$$

where $i \in [1, 2, 3, \ldots, N]$ and $j \in [1, 2, 3, \ldots, D]$; low and up stand for the lower and upper bound of search space, respectively; U is the uniform distribution function.

2.2. Historical Population Setting. BSA chooses the previous population randomly by appointing the Oldpop as history population employed with the intent of controlling the search space and remembers the position until the Oldpop gets changed. What makes BSA have memory function is two numbers a and b generated randomly when the cycle starts. If $a < b$, then Oldpop = Pop; meanwhile, the order of the individual in Oldpop will be arrayed randomly.

2.3. Population Evolution. At this stage, BSA will generate the new population, making full use of its previous experiment based on Pop and Oldpop; evolutionary equation is as follows:

$$M = \mathrm{Pop} + F * \left(\mathrm{Oldpop} - \mathrm{Pop}\right), \tag{2}$$

where $F_i = 3 * \mathrm{rand}$, $i \in [1, 2, 3, \ldots, N]$ and rand is a random number in the rang $[0, 1]$; F is a parameter, controlling range of the search direction matrix (Oldpop − Pop).

2.4. Population Crossover. The new population crossover strategy different from DE shown in BSA is that the mixed proportion parameter is employed to control the numbers of cross particle. Equation is given as follows:

$$T_{i,j} = \begin{cases} M_{i,j}, & \mathrm{map}_{i,j} = 1, \\ \mathrm{Pop}_{i,j}, & \mathrm{map}_{i,j} = 0, \end{cases} \tag{3}$$

where map is $N \times D$ matrix which only contains 0 and 1, and its initial value is 1. Its specific calculation formula is given as follows:

$$\begin{aligned} \mathrm{map}_{i,u(1:[mr*\mathrm{rand}*D])} &= 0, \quad a < b, \\ \mathrm{map}_{i,\mathrm{randi}(D)} &= 0, \quad a \geq b, \end{aligned} \tag{4}$$

where randi is an integer selected from 0 to D randomly; mr is a mixed proportion parameter, and $mr = 1$; rand is chosen randomly in the range $[0, 1]$, and so do a and b; u is an integer vector sorted randomly, and $u \in [1, 2, 3, \ldots, D]$. The map in the crossover stage controls the numbers of individual that will mutate by using $[mr * \mathrm{rand} * D]$ and randi(D). When $a < b$, map_i is a two-element vector which contains many random positions; otherwise, a vector just containing an element "0" will be shown in the map_i. Some individuals generated in the crossover stage can overflow the limited search space as a result of crossover strategy. For this, a boundary control to the individuals beyond the search space will be conducted using (1) again.

2.5. Greedy Selection. When BSA finishes above steps, the new population T_i that have better position than original population Pop_i are used to update the Pop_i based on greedy selection and become new candidate solution. Meanwhile, if the best individual of new population has a lower fitness than optimum obtained from previous iterations, the original one will be replaced to be the new optimum. Pop_i and optimum will remain when conditions mentioned above are not satisfied.

3. The Improved Backtracking Search Algorithm

In the above part, BSA is introduced briefly, and it can be known easily that BSA equipping a single control parameter, a trial population (Oldpop), and a simple structure has a good ability to explore search space in the early stage. However, it is a common phenomenon for evolutionary algorithms falling into local optimum easily when facing the multimodal and nonlinear problems, and so does BSA. In this regard, an improved BSA based on population control factor and optimal learning strategy enlightened by PSO and DE, respectively, is presented in this article. On the one hand, population control factor that regulates search direction of population is employed when BSA operates; on the other hand, disadvantaged group is guided to have optimal learning with the help of the optimal individual of the previous iteration. Specific improvements are as follows.

3.1. Improved BSA Based on Population Control Factor (CBSA). Shortage shown in the BSA is that the Pop in the variation equation (2) has stochastic character, which makes BSA lack memory about position of population when the trend of expanding search space and exploring the new search area follows. Namely, Pop has the trend of global optimization. Therefore, we can take global search firstly and local fine search later into account. Enlightened by literature [20], inertia weight is employed to the evolutionary equation (2), and the new evolutionary equation is as follows:

$$M = w * \text{Pop} + F * (\text{Oldpop} - \text{Pop}), \quad (5)$$

where w is a variable parameter, changing with the increase of iteration numbers linearly. So the target we just referred above can be reached on condition that the algorithm has a stronger ability of global search when w is larger, and the algorithm tends to local search when w is smaller. Thus, we can employ a global coefficient adaptively to improve the search accuracy and search efficiency when facing some complex optimization problems. But, every coin has two sides. This strategy cannot reflect the actual optimization search process when the algorithm we just proposed faces some nonlinear problems as a result of linearity shown in w, so this article presents the following search equation again:

$$M = m * w * \text{Pop} + F * (\text{Oldpop} - \text{Pop}), \quad (6)$$

where m is a $N \times D$ matrix and the elements in matrix m fit normal distribution following the condition that mean is $p = 1$ while variance is $q = 0.3$. Thus, a disturbance will be produced to $(w * \text{Pop})$ to help algorithm escape the local optimum when algorithm handles some nonlinear problems. In the last, the final evolution equation is summarized as

$$M = c * \text{Pop} + F * (\text{Oldpop} - \text{Pop}), \quad (7)$$

where $c = m * w$, named population control factor. The above evolutionary equation can not only improve the convergence speed with the help of w but also enhance the ability of escaping the local optimum with the help of m.

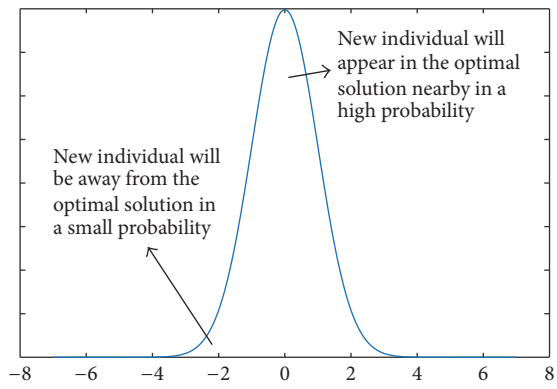

Figure 1: Standard normal distribution.

Based on simulation experiments, it is proved strongly that the improved algorithm has a great performance on the convergence speed.

3.2. Improved BSA Based on Optimal Learning Strategy (OBSA). Another shortage shown in BSA is that BSA does not acquire enough experience, taking full advantage of the best current individual, from DE. Besides, it is important for evolutionary algorithms to find a balance between global search and local search. It can be known that BSA, in focus on global optimization with (2), displays a poor ability of local search, because it ignores the importance of optimum individual. The optimum individual in the current population is very meaningful source that can be used to improve local search. In this regard, enlightened by literature [5] and literature [21], this article proposes the following search equation:

$$v_{i,j} = x_{\text{best},j} + m * \text{Pop}_{i,j}, \quad (8)$$

where $m = 3 * \text{randn}$, $x_{\text{best},j}$ is the optimum solution in the current population, and randn is standard normal distribution. Obvious observation can be acquired from Figure 1 that the new individual dominated by (8) can be random enough for local search owing a high probability that the new individual will appear around the best individual. Namely, the evolutionary equation described by (8) is good at local search but poor at global search.

3.3. Improved BSA Based on Population Control Factor and Optimal Learning Strategy (COBSA). With purpose of solving the shortage of slow convergence speed in the early stage and low convergence precision in the late stage shown in BSA, an improved BSA based on population control factor and optimal learning strategy in combination with Sections 3.1 and 3.2 is presented in this article. The improved BSA named COBSA produces the new population by employing (7), and optimal learning strategy based on (8) will be applied to the disadvantaged group that has a worse behavior than previous population. The specific steps of the COBSA are given as follows.

Step 1. Initialize the population Pop and Oldpop.

Step 2. Generate two numbers a and b randomly, if $a < b$, Oldpop = Pop, and Oldpop will be arrayed randomly later.

Step 3. Produce a new population on the basis of initial population by employing (7).

Step 4. Generate cross matrix map by using (4).

Step 5. Take cross strategy through (3).

Step 6. Initialize the individual that is out of boundary.

Step 7. Seek the value of the population, and disadvantaged group is guided to have optimal learning by employing (8).

Step 8. Select the optimal population according to the greedy selection mechanism.

Step 9. Judge whether the cycle meets the target; if not, return to Step 2.

Step 10. Output optimal solution.

4. Experiment and Result Analysis

In order to verify effectiveness of the improved algorithm proposed in this article, COBSA is applied to minimize a set of 18 benchmark functions commonly used in optimization algorithms. Table 1 presents in detail function form, search space, and function dimension used for the evolutionary algorithms in the tests. Some problems will arise when COBSA operates owing to the different characters shown in the 18 benchmark functions. $f1$~$f5$, $f11$~$f13$, and $f17$ are unimodal and continuous functions. $f6$~$f8$, $f10$, $f14$, $f16$, and $f18$ are multipeak and the number of local minimum values will increase obviously when dimension D gets larger. $f9$ is the Rosenbrock function which is unimodal when D = 2 and 3, but numerous local minimum values will arise when problem dimension is high [22]. $f15$ is the three-hump camel function in $D = 2$, having three local minimum values. This article adjusts the form of some functions for convenience to compare when different versions in different literatures shown in these functions are considered.

Enlightened by literatures [23–25], the following methods are designed to evaluate the performance of COBSA: (1) convergence performance in fixed iteration; (2) iteration performance in fixed precision; (3) comparison algorithms employed to compare with COBSA; (4) statistical test based on the Wilcoxon Signed-Rank Test used to analyse the property of COBSA and comparison algorithms.

The parameters of algorithm used in the evaluation methods (1), (2), (3), and (4) are listed as follows: the population size is 30, the maximum cycle of times to be evaluated for each benchmark functions is 1000, and the final result is the average value based on 20 independent running times.

4.1. Convergence Performance in Fixed Iteration. In this section, in order to analyse how much the population control factor and the optimal learning strategy contribute to improve the performance of BSA, this article compares the convergence speed and convergence precision of BSA, CBSA, OBSA, and COBSA based on the 18 objective functions when the maximum cycle is 1000. And the fitness of function takes the logarithm of 10 to display the difference shown in different strategy more obviously.

Important observation on convergence speed and convergence precision of different algorithms can be obtained from the result displayed in Figure 2. A better performance on convergence speed shown in CBSA is displayed when BSA works under the same condition, which implies the efficiency of evolution equation (7) powerfully. It also can be obtained from Figure 2 that OBSA has better convergence precision than that of BSA in the late stage, which shows the correctness of search equation (8) proposed in Section 3.2. Meanwhile, we can learn that the COBSA in combination with CBSA and OBSA, displaying faster speed and higher precision in Figure 2, has a more obvious influence on the performance of BSA when compared with BSA, CBSA, and OBSA.

4.2. Iteration Performance in Fixed Precision. In this section, the performance of each algorithm depended on the iteration numbers when evolution precision which is limited is employed to make a comparison. All the algorithms have been run 20 times independently based on 18 objective functions. For each function, the current cycle will stop either when the precision is less than the limitation given in Table 2 or when the maximum cycle is satisfied. The results, including BSA, CBSA, OBSA, and COBSA, are displayed in Table 2 in terms of min, max, mean, SR, and EI. Specific expression of each parameter we just mentioned is as follows: the success rate (SR) = numbers of achieving target precision ÷ total numbers of experiment; expected iterations (EI) = the numbers of particle × average iteration numbers ÷ success rate. And "∞" appearing in Table 2 indicates that the expected iterations are infinite.

An interesting result is that BSA does not meet the target precision for all functions except Sumpower $f3$, and the SR of BSA is just 0.15 on function Sumpower $f3$ when that of CBSA, OBSA, and COBSA is 1. Besides, for other functions, a higher SR and less EI are displayed when CBSA and OBSA are compared with BSA, while COBSA has 100% success rate for all functions with least expected iteration. The iteration performance in fixed precision in this section also proves the superiority of COBSA.

4.3. Experiment Based on Comparison Algorithms. In this section, some relatively new evolutionary algorithms evolving in recent decades are employed to compare with COBSA. ABC (limit = 100), MABC (limit = 100), BA ($A = 0.5$, $r = 0.5$, $f_{\max} = 2$, $f_{\min} = 0$, $\alpha = 0.9$, and $\gamma = 0.9$), DSA, and BSA are included. The result is judged in terms of mean, standard deviation (std) of the solution obtained from 20 independent tests.

As it can be obtained from Table 3, COBSA offers a higher convergence precision on almost all functions while the stander deviation of COBSA is also much smaller than that of the comparison algorithms. In particular, COBSA can

Table 1: Test function.

Function	Function expression	Dim	Search space
Sphere $f1$	$f(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$
Sumsquare $f2$	$f(x) = \sum_{i=1}^{n} i x_i^2$	60	$[-10, 10]$
Sumpower $f3$	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$	60	$[-1, 1]$
Schwefel2.22 $f4$	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	100	$[-10, 10]$
Exponential $f5$	$f(x) = \exp\left(0.5 * \sum_{i=1}^{n} x_i\right) - 1$	60	$[-1.28, 1.28]$
Alpine $f6$	$f(x) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1 x_i \rvert$	60	$[-10, 10]$
Rastrigin $f7$	$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos 2\pi x_i) + 10$	100	$[-100, 100]$
Griewank $f8$	$f(x) = \frac{1}{4000} \sum_{i=1}^{n} (x_i)^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{x_i}}\right) + 1$	60	$[-600, 600]$
Rosenbrock $f9$	$f(x) = \sum_{i=1}^{n-1} \left[100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right]$	30	$[-30, 30]$
Ackley $f10$	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	60	$[-32, 32]$
R-H-Ellipsoid $f11$	$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2$	80	$[-100, 100]$
Schwefel1.2 $f12$	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	10	$[-30, 30]$
Elliptic $f13$	$f(x) = \sum_{i=1}^{n} \left(10^6\right)^{(i-1)/(n-1)} x_i^2$	30	$[-100, 100]$
NCRastrigin $f14$	$f(x) = \sum_{i=1}^{n} \left(y_i^2 - 10\cos 2\pi y_i\right) + 10$, $y_i = \begin{cases} x_i, & \lvert x_i \rvert < \frac{1}{2}, \\ \frac{\text{round}(2x_i)}{2}, & \lvert x_i \rvert \geq \frac{1}{2} \end{cases}$	50	$[-5.12, 5.12]$
T-H camel $f15$	$f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1 x_2 + x_2^2$	2	$[-5, 5]$
Schaffer $f16$	$f(x) = \left(\sum_{i=1}^{n} x_i^2\right)^{0.25} \left(\sin^2\left(50\left(\sum_{i=1}^{n} x_i^2\right)^{0.1}\right) + 1\right)$	10	$[-100, 100]$
Powell $f17$	$f(x) = \sum_{i=1}^{n/4} \left[\left(x_{4i-3} + 10x_{4i-2}\right)^2 + 5\left(x_{4i-1} - x_{4i}\right)^2 + \left(x_{4i-2} - 2x_{4i-1}\right)^4 + 10\left(x_{4i-3} + x_{4i}\right)^4\right]$	70	$[5, -4]$
Weierstrass $f18$	$f(x) = \sum_{i=1}^{n} \left(\sum_{k=0}^{k\max} \left[a^k \cos\left(2\pi b^k \left(x_i + 0.5\right)\right)\right]\right) - n \sum_{k=0}^{k\max} \left[a^k \cos\left(2\pi b^k 0.5\right)\right]$, $a = 0.5\ b = 3\ k\max = 20$	60	$[-0.5, 0.5]$

find optimal solution when facing Rastrigin $f7$, Griewank $f8$, NCRastrigin $f14$, and Weierstrass $f18$. However, in the case of Rosenbrock $f9$, the convergence precision of COBSA is worse compared to that of ABC and MABC, since ABC is just one magnitude order higher when compared with COBSA on Rosenbrock $f9$; the superiority of ABC and MABC is not very obvious. Besides, COBSA displays a more stable search process than ABC and MABC owing to a lower stander deviation on Rosenbrock $f9$.

4.4. Experiment Based on Wilcoxon Signed-Rank Test. A comparative method based on statistical analysis by using Wilcoxon Signed-Rank Test is employed to evaluate the performance of COBSA and comparison algorithms.

Table 4 displays the statistical results using the average values based on 20 independent running times of COBSA and comparison algorithms to handle the objective functions in Table 1. The result presents that the COBSA displaying a better statistical property successfully outperforms all of the

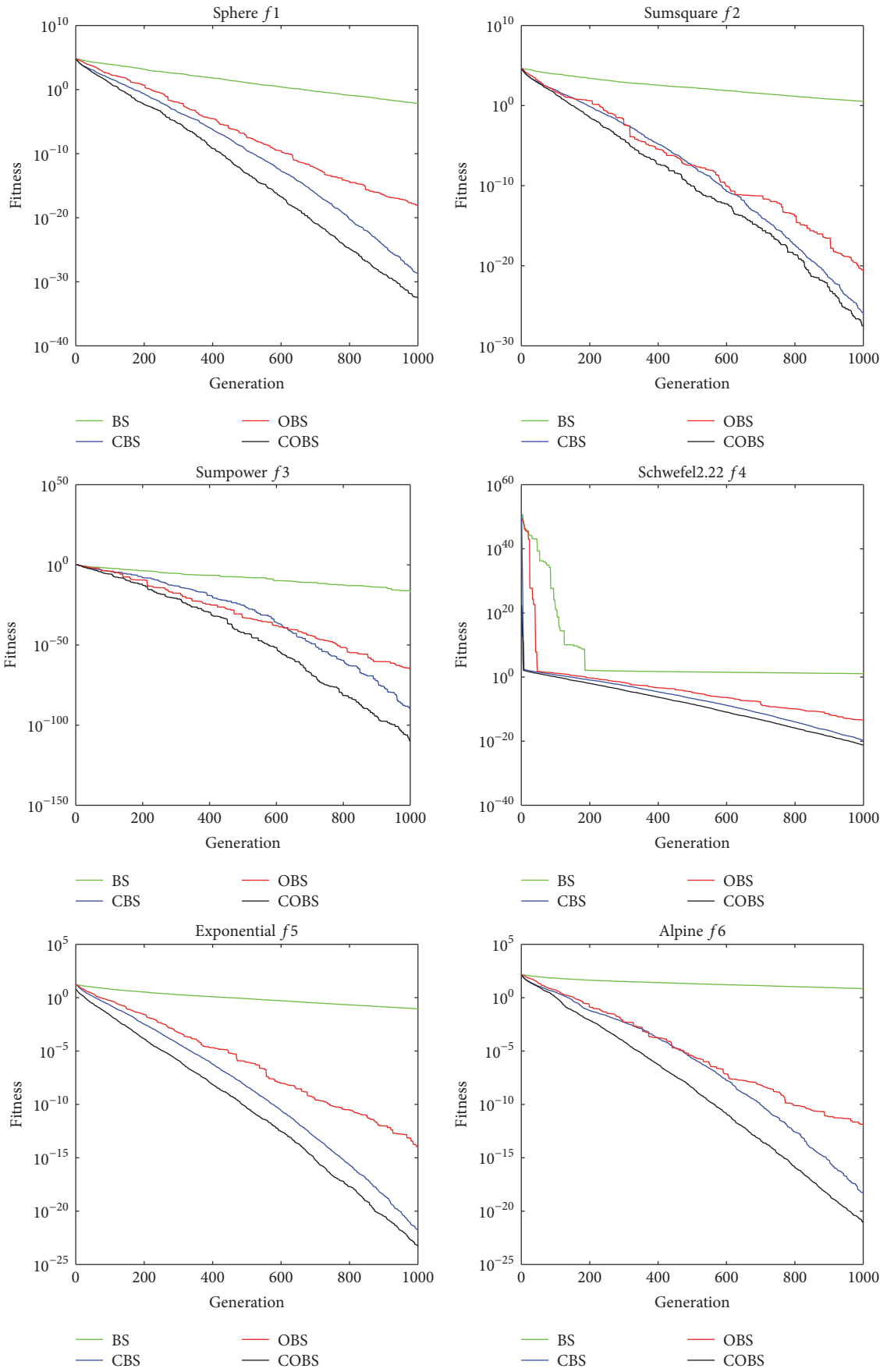

Figure 2: Continued.

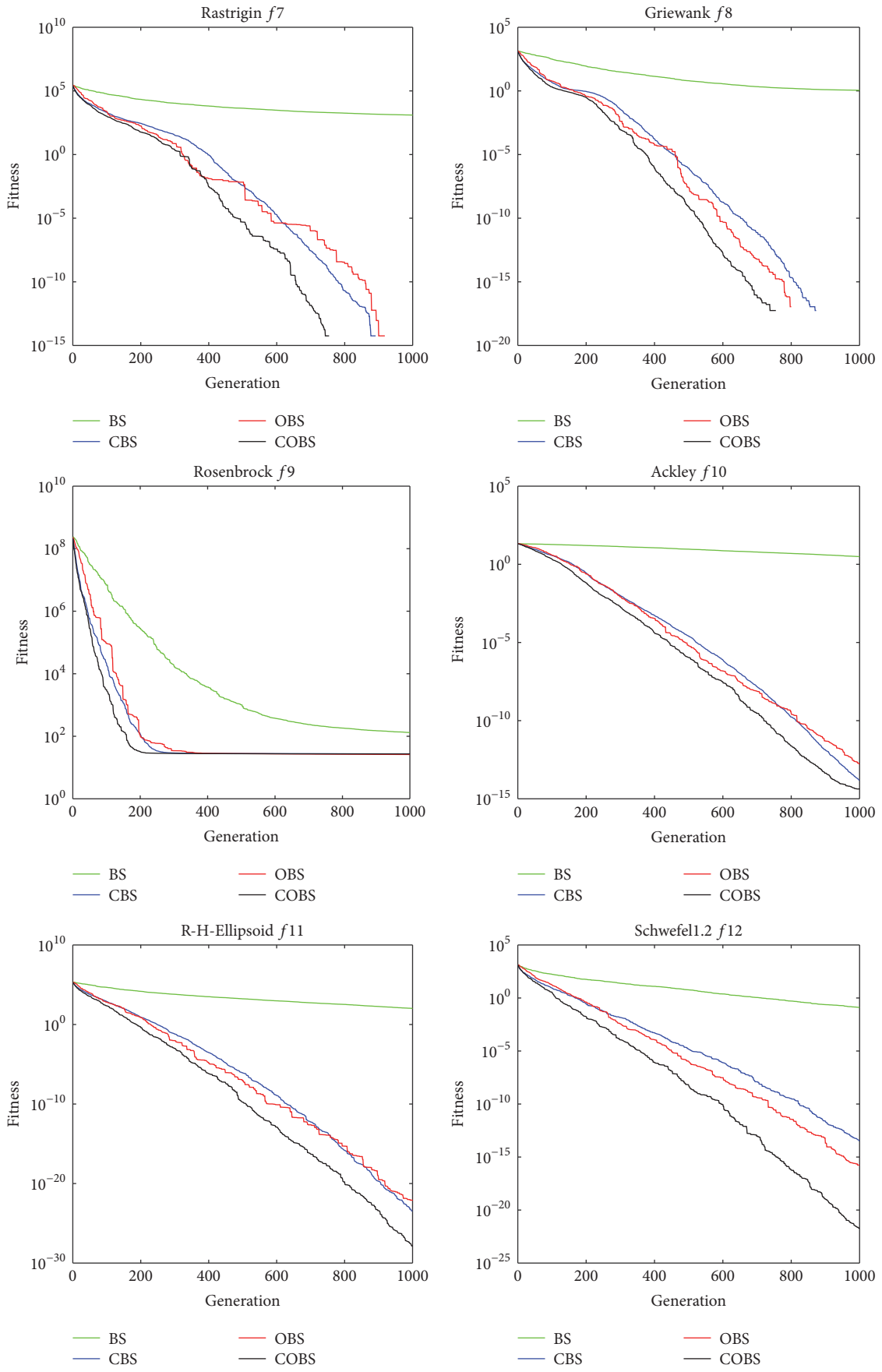

Figure 2: Continued.

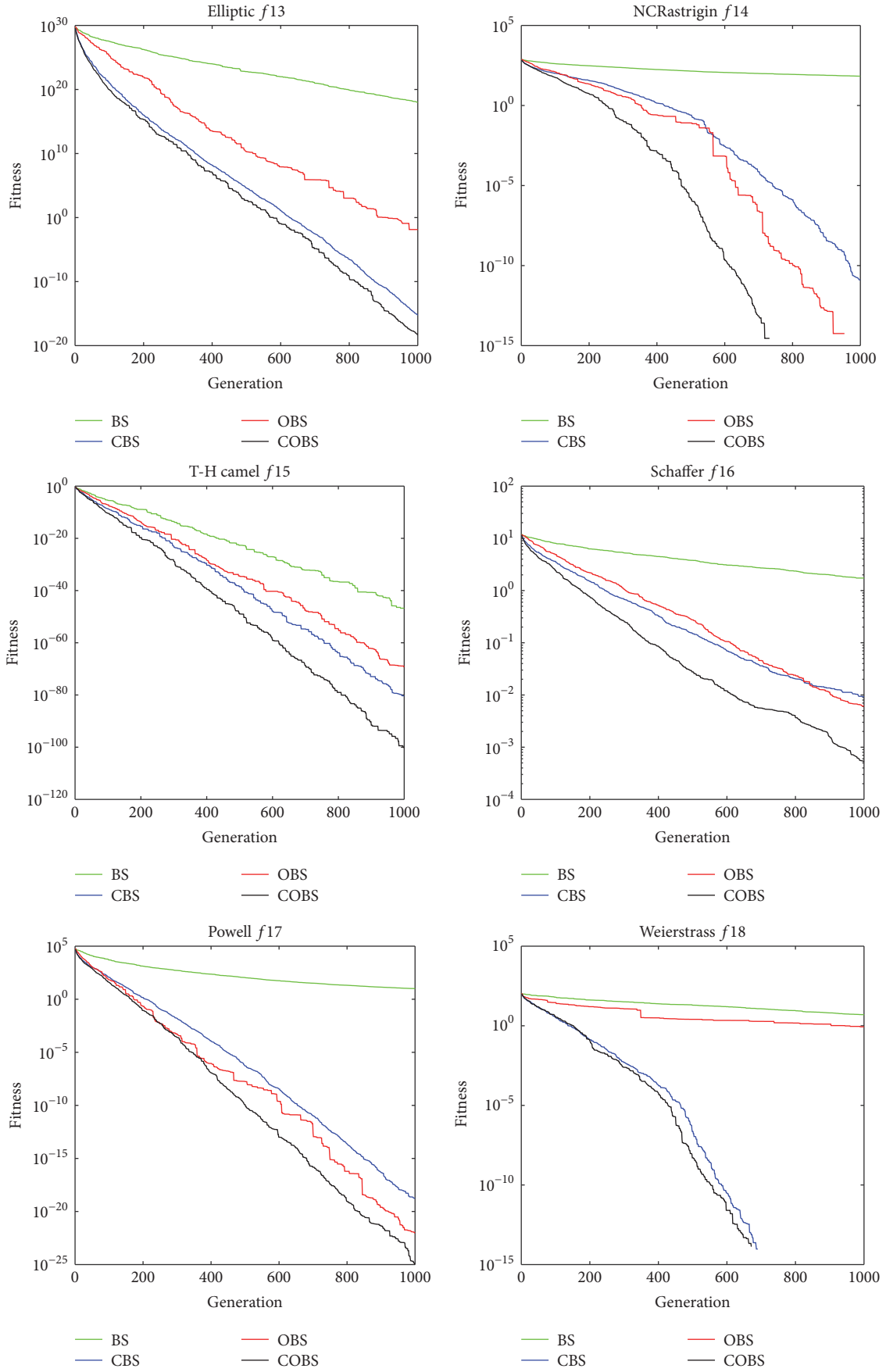

Figure 2: Convergence performance in fixed iteration.

Table 2: Iteration performance in fixed precision.

Function	Limitation	Algorithm	Iterations			Success rate	Expected iterations
			Min	Max	Mean		
$f1$	$e-24$	BSA	1000	1000	1000	0	∞
		CBSA	845	925	877	1	26310
		OBSA	870	1000	974	0.3	97400
		COBSA	635	734	692	1	20760
$f2$	$e-25$	BSA	1000	1000	1000	0	∞
		CBSA	938	1000	971	0.95	30663
		OBSA	720	1000	957	0.35	82028
		COBSA	664	802	742	1	22260
$f3$	$e-20$	BSA	927	1000	993	0.15	198600
		CBSA	334	442	381	1	11430
		OBSA	211	353	288	1	8640
		COBSA	159	259	223	1	6690
$f4$	$e-17$	BSA	1000	1000	1000	0	∞
		CBSA	844	981	877	1	26310
		OBSA	791	1000	961	0.35	82371
		COBSA	688	851	747	1	22410
$f5$	$e-16$	BSA	1000	1000	1000	0	∞
		CBSA	757	837	791	1	23730
		OBSA	725	1000	952	0.5	57120
		COBSA	613	755	675	1	20250
$f6$	$e-18$	BSA	1000	1000	1000	0	∞
		CBSA	913	1000	954	0.9	31800
		OBSA	951	1000	993	0.2	148950
		COBSA	711	830	784	1	23520
$f7$	$e-5$	BSA	1000	1000	1000	0	∞
		CBSA	561	646	599	1	17970
		OBSA	250	541	453	1	13590
		COBSA	354	488	422	1	12660
$f8$	$e-7$	BSA	1000	1000	1000	0	∞
		CBSA	477	558	529	1	15870
		OBSA	329	615	465	1	13680
		COBSA	247	469	411	1	12330
$f9$	30	BSA	1000	1000	1000	0	∞
		CBSA	215	298	267	1	8010
		OBSA	163	349	244	1	7320
		COBSA	141	241	189	1	5670
$f10$	$e-14$	BSA	1000	1000	1000	0	∞
		CBSA	958	1000	979	0.45	65266
		OBSA	830	1000	989	0.3	98900
		COBSA	687	860	784	1	23520
$f11$	$e-21$	BSA	1000	1000	1000	0	∞
		CBSA	873	947	914	1	27420
		OBSA	693	1000	927	0.9	30900
		COBSA	629	816	710	1	21300
$f12$	$e-14$	BSA	1000	1000	1000	0	∞
		CBSA	892	1000	986	0.35	84514
		OBSA	766	1000	883	0.9	29433
		COBSA	532	710	624	1	18720

Table 2: Continued.

Function	Limitation	Algorithm	Iterations			Success rate	Expected iterations
			Min	Max	Mean		
$f13$	$e-07$	BSA	1000	1000	1000	0	∞
		CBSA	630	700	667	1	20010
		OBSA	505	1000	821	0.85	28976
		COBSA	524	611	566	1	16980
$f14$	$e-05$	BSA	1000	1000	1000	0	∞
		CBSA	590	790	679	1	20370
		OBSA	286	960	608	1	18240
		COBSA	424	556	492	1	14760
$f15$	$e-80$	BSA	1000	1000	1000	0	∞
		CBSA	869	1000	973	0.6	48650
		OBSA	803	1000	991	0.45	66066
		COBSA	763	893	786	1	10560
$f16$	0.01	BSA	1000	1000	1000	0	∞
		CBSA	800	1000	948	0.7	40628
		OBSA	596	1000	831	0.8	31162
		COBSA	475	667	577	1	17310
$f17$	$e-20$	BSA	1000	1000	1000	0	∞
		CBSA	956	1000	990	0.5	59400
		OBSA	658	1000	854	0.75	34160
		COBSA	545	761	689	1	20670
$f18$	0.01	BSA	1000	1000	1000	0	∞
		CBSA	256	323	294	1	8820
		OBSA	172	1000	496	0.85	17505
		COBSA	218	327	260	1	7800

comparison algorithms, based on the Wilcoxon Signed-Rank Test with a statistical significance value $\alpha = 0.05$.

According to the above analyses of Sections 4.1, 4.2, 4.3, and 4.4, conclusion can be summarized safely as follows: COBSA can improve the performance of BSA effectively, and it is a feasible computation method.

5. The Choice of Variance q

In Section 3.1, the variance q of matrix m plays an important role in affecting the performance of CBSA. When q takes 0, the matrix m will lose its role. When q increases from 0 to 1, the fluctuation to population Pop will also increase correspondingly. However, a lower convergence speed will be displayed owing to the obvious fluctuation to population Pop when q is large. Therefore, four objective functions Sumsquare $f2$, Schwefel2.22 $f4$, Alpine $f6$, and Ackley $f10$ are employed to analyse the effect of variance q. The fitness of function takes the logarithm of 10 to show and observe the curves obviously.

The result is better when the fitness of function is smaller owing to the minimization process shown in these functions. We can obtain from Figure 3 that variance q affects the result obviously. CBSA plays a better performance on Sumsquare $f2$ and Alpine $f6$ when $q = 0.3$. When q takes 0.4, the fitness of Schwefel2.22 $f4$ is better. For Ackley $f10$, a lower precision is obtained when q is around 0.3. Therefore, the selective variance q is set at 0.3 in this article.

Besides, we can also observe from Figure 3 that the influence of matrix m cannot be ignored. When variance q is 0, the element in matrix m is 1, and as a result, the matrix m will lose its role. As it can be obtained from Figure 3, a better performance is displayed when q is around 0.3 rather than 0, which proves the significance of matrix m.

6. Conclusion

In order to solve the problem of slow convergence speed and low convergence precision shown in BSA, this article proposes an improved algorithm, named COBSA, through employing population control factor to the variation equation and helping disadvantaged group have optimal learning. The experiments based on 18 benchmark functions prove that COBSA is an effective evolutionary algorithm, and it is a feasible solution to improve the convergence speed and convergence precision of BSA.

However, COBSA displays a lower convergence precision in the function Rosenbrock $f9$ when compared with ABC and MABC. How to make algorithm display better performance on functions Rosenbrock $f9$ is the next step in our work. Besides, it is worth applying the algorithm proposed in

Table 3: Experiment based on comparison algorithm.

Function	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$	$f8$	$f9$	$f10$	$f11$	$f12$	$f13$	$f14$	$f15$	$f16$	$f17$	$f18$
ABC																		
Mean	9.25$e-$11	5.30$e-$05	4.45$e-$08	0.16	1.30$e-$03	0.23	78.54	0.03	4.59	0.35	7.20$e-$03	3.63	7.74$e+$14	11.87	5.89$e-$18	3.85	5.62	0.24
Std	3.11$e-$10	3.72$e-$04	2.32$e-$08	0.07	4.37$e-$04	0.12	24.66	0.05	3.41	0.32	1.97$e-$02	2.07	1.45$e+$15	2.24	4.79$e-$18	1.12	5.60	0.04
MABC																		
Mean	7.29$e-$16	9.69$e-$10	6.87$e-$15	2.76$e-$02	1.28$e-$06	7.95$e-$05	39.73	5.6$e-$03	23.06	2.30$e-$03	1.93$e-$05	17.07	6.25$e-$08	1.90	2.19$e-$12	2.51	25.06	0.13
Std	9.00$e-$17	2.05$e-$09	2.49$e-$14	5.70$e-$03	2.69$e-$07	7.33$e-$05	6.11	0.01	44.62	8.71$e-$03	9.94$e-$06	10.14	2.30$e-$07	1.48	9.78$e-$12	0.39	13.13	0.34
DSA																		
Mean	5.50$e-$03	1.46	5.01$e-$17	7.69	0.05	3.08	1.04$e+$03	1.01	171.17	3.75	51.35	10.44	5.03$e+$16	49.32	6.80$e-$36	2.14	57.01	3.81
Std	6.30$e-$03	0.87	1.19$e-$16	2.24	0.02	1.33	161.71	0.12	29.57	3.23	47.96	9.30	5.71$e+$16	4.47	3.05$e-$35	0.38	51.93	0.68
BSA																		
Mean	0.01	2.37	3.09$e-$17	11.58	0.09	7.57	1.27$e+$03	1.12	141.34	3.49	86.61	0.12	2.52$e+$25	64.67	4.64$e-$47	1.57	9.34	4.94
Std	0.92	1.28	6.14$e-$17	2.08	0.02	3.03	130.81	0.05	27.38	2.77	27.06	0.08	3.21$e+$25	5.34	2.05$e-$46	0.19	5.45	0.73
BA																		
Mean	9.17$e-$07	0.20	4.84$e-$07	4.76$e+$31	0.17	19.56	7.2$e+$03	1.01	86.84	19.25	7.94$e-$06	5.15$e-$08	5.02$e+$25	409.00	4.56$e-$12	11.81	3.48	3.65
Std	4.10$e-$08	0.35	2.16$e-$08	2.13$e+$32	0.16	6.39	195.00	4.50	108.27	0.14	3.55$e-$07	2.30$e-$09	2.25$e+$25	47.00	2.02$e-$12	2.68	1.07	16.34
COBSA																		
Mean	2.01$e-$32	3.23$e-$28	4.33$e-$111	3.95$e-$20	2.51$e-$22	3.99$e-$21	0	0	27.38	4.09$e-$15	1.47$e-$27	1.01$e-$20	1.38$e-$18	0	7.50$e-$104	4.32$e-$04	5.42$e-$27	0
Std	8.79$e-$32	1.43$e-$27	1.47$e-$110	2.01$e-$21	7.50$e-$22	1.19$e-$20	0	0	0.13	1.10$e-$15	5.86$e-$27	4.33$e-$20	6.16$e-$17	0	3.13$e-$103	1.20$e-$03	2.27$e-$26	0

Table 4: Statistical test based on Wilcoxon Signed-Rank Test ($\alpha = 0.05$).

Algorithm versus COBSA	p value	$R+$	$R-$	Winner
ABC versus COBSA	$2.50e-03$	16	155	COBSA
MABC versus COBSA	$2.10e-03$	15	156	COBSA
DSA versus COBSA	$1.96e-04$	0	171	COBSA
BSA versus COBSA	$1.96e-04$	0	171	COBSA
BA versus COBSA	$1.96e-04$	0	171	COBSA

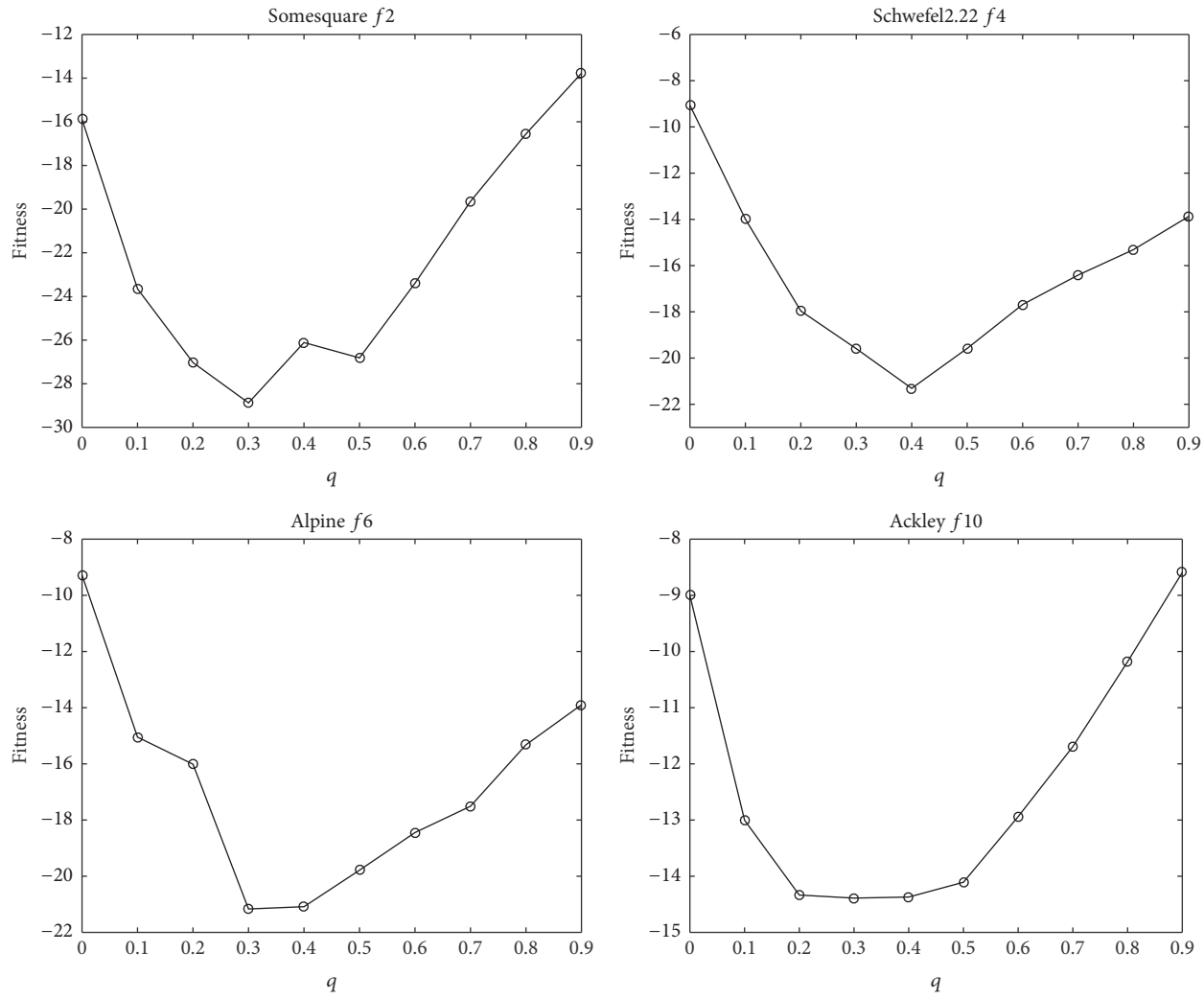

Figure 3: The influence of q.

this article to the blind source separation and hyperspectral image processing.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work is supported by National Nature Science Foundation of China (no. 61401307), China Postdoctoral Science Foundation (no. 2014M561184), Tianjin Research Program of Application Foundation and Advanced Technology of China (no. 15JCYBJC17100), and Tianjin Research Program of Science and Technology Commissioner of China (16JCTPJC48400).

References

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

[2] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, MIT Press, Combridge, MA, USA, 2004.

[3] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Combridge, MA, USA, 1992.

[4] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[5] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, Berlin, Germany, 2005.

[6] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[7] P. Civicioglu, "Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm," *Computers and Geosciences*, vol. 46, pp. 229–247, 2012.

[8] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[9] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, Coimbatore, India, December 2009.

[10] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO '10)*, J. R. Gonzalez, D. A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds., vol. 284 of *Studies in Computational Intelligence*, pp. 65–74, Springer, Berlin, Germany, 2010.

[11] L. Chen and L. Y. Zhang, "Blind signal separation algorithm based on temporal predictability and differential search algorithm," *Journal on Communications*, vol. 35, no. 6, pp. 117–125, 2014.

[12] Z. C. Jia, Y. Y. Xue, L. Chen, Y. J. Guo, and H. D. Xu, "Blind separation algorithm for hyperspectral image based on the denoising reduction and the bat optimization," *Acta Photonica Sinica*, vol. 45, no. 5, Article ID 0511001, 2016.

[13] J. H. Liang and M. Ma, "Artificial bee colony algorithm based research on image segmentation," *Computer Engineering and Application*, vol. 48, no. 8, pp. 194–197, 2012.

[14] P. Civicioglu, "Backtracking search optimization algorithm for numerical optimization problems," *Applied Mathematics and Computation*, vol. 219, no. 15, pp. 8121–8144, 2013.

[15] Z. L. Zhang and S. Y. Liu, "Decision-theoretic rough set attribute reduction based on backtracking search algorithm," *Computer Engineering and Applications*, vol. 52, no. 10, pp. 71–74, 2016.

[16] X. Chen, S. Y. Liu, and Y. Wang, "Emergency resources scheduling based on improved backtracking search optimization algorithm," *Computer Applications and Software*, vol. 32, no. 12, pp. 235–238, 2015.

[17] M. D. Li and H. Zhao, "Backtracking search optimization algorithm with comprehensive learning strategy," *System Engineering and Electronics*, vol. 37, no. 4, pp. 958–963, 2015.

[18] X. J. Wang, S. Y. Liu, and W. K. Tian, "Improved backtracking search optimization algorithm with new effective mutation scale factor and greedy crossover strategy," *Journal of Computer Applications*, vol. 34, no. 9, pp. 2543–2546, 2014.

[19] W. K. Tian, S. Y. Liu, and X. J. Wang, "Study and improvement of backtracking search optimization algorithm based on differential evolution," *Application Research of Computers*, vol. 32, no. 6, pp. 1653–1662, 2015.

[20] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation and IEEE World Congress on Computational Intelligence*, (Cat. No.98TH8360), pp. 69–73, Anchorage, Alaska, USA, May 1998.

[21] W. F. Gao and S. Y. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, no. 3, pp. 687–697, 2012.

[22] Y. W. Shang and Y. H. Qiu, "A note on the extended Rosenbrock function," *Evolutionary Computation*, vol. 14, no. 1, pp. 119–126, 2006.

[23] W. Hu and Z. S. Li, "A simpler and more effective particle swarm optimization algorithm," *Journal of Software*, vol. 18, no. 4, pp. 861–868, 2007.

[24] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[25] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.