Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 832949, 8 pages
http://dx.doi.org/10.1155/2014/832949

Research Article

Hybrid Artificial Bee Colony Algorithm and Particle Swarm Search for Global Optimization

Wang Chun-Feng, Liu Kui, and Shen Pei-Ping

College of Mathematics and Information, Henan Normal University, Xinxiang 453007, China

Correspondence should be addressed to Wang Chun-Feng; wangchunfeng10@126.com

Received 17 July 2014; Accepted 1 October 2014; Published 28 October 2014

Academic Editor: Guangming Xie

Artificial bee colony (ABC) algorithm is one of the most recent swarm intelligence based algorithms, which has been shown to be competitive to other population-based algorithms. However, there is still an insufficiency in ABC regarding its solution search equation, which is good at exploration but poor at exploitation. To overcome this problem, we propose a novel artificial bee colony algorithm based on particle swarm search mechanism. In this algorithm, for improving the convergence speed, the initial population is generated by using good point set theory rather than random selection firstly. Secondly, in order to enhance the exploitation ability, the employed bee, onlookers, and scouts utilize the mechanism of PSO to search new candidate solutions. Finally, for further improving the searching ability, the chaotic search operator is adopted in the best solution of the current iteration. Our algorithm is tested on some well-known benchmark functions and compared with other algorithms. Results show that our algorithm has good performance.

1. Introduction

Optimization problems play a very important role in many scientific and engineering fields. In the last two decades, several swarm intelligence algorithms, such as ant colony optimization (ACO) [1, 2], particle swarm optimization (PSO) [3, 4], and artificial bee colony (ABC) algoritm [5, 6], have been developed for solving difficult optimization problem. Researchers have shown that algorithms based on swarm intelligent have great potential [7–9] and have attracted much attention.

The ABC algorithm was first proposed by Karaboga in 2005, inspired by the intelligent foraging behavior of honey bee [5]. Since the invention of the ABC algorithm, it has been used to solve both numerical and nonnumerical optimization problems. The performance of ABC algorithm has been compared with some other intelligent algorithms, such as GA [10], differential evolution algorithm (DE) [11]. The results show that ABC algorithm is better than or at least comparable to the other methods. Recently, for improving the performance of ABC algorithm, many variant ABC algorithms have been developed. Alatas proposed a ABC algorithm by using chaotic map as efficient alternatives to generate pseudorandom sequence [12]. To improve the exploitation ability, Zhu and Kwong presented a global-best-solution-guided ABC (GABC) algortihm by incorporating the information of global best solution into the solution search equation [13]. By combing Powell's method, Gao et al. proposed an improved ABC algorithm-Powell ABC (PABC) algorithm [14]. In order to improve the exploitation ability, a converge-onlookers ABC (COABC) was developed by applying the best solution of the previous iteration in search equation at the onlooker stage [15]. More extensive review of ABC can refer to [16].

In addition, considering PSO has good exploitation ability, a few of hybrid ABC algorithms have been presented based on PSO algorithm. For example, a novel hybrid approach referred to as IABAP based on the PSO and ABC is presented in [17]. In this algorithm, the flow of information from bee colony to particle swarm is exchanged based on scout bees. Another hybrid approach is the ABC-SPSO algorithm based on the ABC and PSO [18]. In ABC-SPSO algorithm, the update rule (solution updating equation) of the ABC algorithm is executed among personal best solutions of the particles after the main loop of the PSO is finished. Unlike the IABAP and ABC-SPSO, a hybrid method named HPA is proposed. The global best solution of the HPA is created using recombination procedure between global best solutions of the PSO and the ABC.

In this paper, we present a hybrid artificial bee colony algorithm based on particle swarm search for global optimization, which is named "ABC-PS." For furthermore improving the performance, some strategies have been applied. The experimental results show that the algorithm could do well in improving the performance of ABC algorithm in most areas.

The rest of the paper is organized as follows. The original ABC algorithm is introduced in Section 2. The PSO is explained in material and methods in Section 3. The proposed ABC-PS approach is described in Section 4. The performance of the ABC-PS is compared with that of original ABC algorithm and the state-of-art algorithm in Section 5. Finally, conclusions are given in Section 6.

2. The Original ABC Algorithm

ABC algorithm contains three groups of bees: employed bees, onlookers, and scouts. The numbers of employed bees and onlookers are set equally. Employed bees are responsible for searching available food sources and gathering required information. They also pass their food information to onlookers. Onlookers select good food source from those found by employed bees to further search the foods. When the quality of the food source is not improved through a predetermined number of cycles, the food source is abandoned by its employed bee. At the same time, the employed bee becomes a scout and start to search for a new food source. In ABC algorithm, each food source represents a feasible solution of the optimization problem and the nectar amount of a food source is evaluated by the fitness value (quality) of the associated solution. The number of employed bees is set to that of food sources.

Assume that the search space is d-dimension, the position of the ith food source (solution) can be expressed as a d-dimension vector $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,d})$, $i = 1, 2, \dots, Sn$, Sn is the number of food sources. The detail of the orginal ABC algorithm is given as follows.

At the initialization stage, a set of food source positions is randomly selected by the bees as in (1) and their nectar amounts are determined:

$$x_{ij} = \underline{x}_j + \operatorname{rand}(0,1) * \left(\overline{x}_j - \underline{x}_j\right), \tag{1}$$

where $i \in \{1, 2, \dots, Sn\}$, $j \in \{1, 2, \dots, d\}$, $\underline{x}_j$ and $\overline{x}_j$ are the lower bound and upper bound of the jth dimension, respectively.

An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source with a probability related to its nectar amount. The food source with higher quality would have a larger opportunity to be selected by onlookers. The probability could be obtained from the following equation:

$$P_i = \frac{\operatorname{fit}(x_i)}{\sum_{i=1}^{Sn} \operatorname{fit}(x_i)}, \tag{2}$$

where $\operatorname{fit}(x_i)$ is the nectar amount of the ith food source and it is associated with the objective function value $f(x_i)$ of the ith food source. Once a food source x_i is selected, she utilizes (3) to produce a modification on the position (solution) in her memory and checks the nectar amount of the candidate source (solution)

$$x'_{ij} = x_{ij} + \psi * \left(x_{ij} - x_{kj}\right), \tag{3}$$

where $i, k \in \{1, 2, \dots, Sn\}$, $k \neq i$, x'_{ij} is a new feasible solution that produced from its previous solution x_{ij} and the randomly selected neighboring solution x_{kj}; ψ is a random number between $[-1, 1]$, which controls the production of a neighbor food source position around x_{ij}; j and k are randomly chosen indexes. In each iteration, only one dimension of each position is changed. Providing that its nectar is higher than that of the previous one, the bee memorizes the new position and forgets the old one.

In ABC algorithm, there is a control parameter called limit in the original ABC algorithm. If a food source is not improved anymore when limit is exceeded, it is assumed to be abandoned by its employed bee and the employed bee associated with that food source becomes a scout to search for a new food source randomly, which would help avoiding local optima.

3. Particle Swarm Optimization (PSO)

As a swarm-based stochastic optimization method, the PSO algorithm was developed by Kennedy and Eberhart [19], which is based on social behavior of bird flocking or fish schooling. The original PSO maintains a population of particles $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,d})$, $i = 1, 2, \dots, Sn$ which distribute uniformly around search space at first. Each particle represents a potential solution to an optimization problem. After randomly produced solutions are assigned to the particles, velocities of the particles are updated by using self-best solution of the particle obtained previous in iterations and global best solution obtained by the particles so far at each iteration. This is formulated as follows:

$$\begin{aligned} v_i^j(k+1) = {} & v_i^j(k) + c_1 * r_1 * \left[p_{i,j}^{\text{best}}(k) - x_i^j(k)\right] \\ & + c_2 * r_2 * \left[g_j^{\text{best}}(k) - x_i^j(k)\right], \end{aligned} \tag{4}$$

where $v_i(k+1)$ $(-v\max \le v_i(k+1) \le v\max)$ which represents the rate of the position change for the particle; $x_i^j(k)$ is the ith particle in jth dimension at step k; $v_i^j(k)$ is the velocity of the ith particle in jth dimension at step k; $p_{i,j}^{\text{best}}$ is the personal best position of the ith particle in jth dimension at time step k; g^{best} is the global best position obtained by the population at step k; c_1 and c_2 are the positive acceleration constants used to scale the contribution of cognitive and social components, respectively; r_1 and r_2 which are stochastic elements of the algorithm are random numbers in the range $[0, 1]$. For each particle i, Kennedy and Eberhart [19] proposed that the position x_i can be updated in the following manner:

$$x_i^j(k+1) = x_i^j(k) + v_i^j(k+1). \tag{5}$$

Considering the minimization problem, the personal best solution of the particle at the next step $k+1$ is calculated as

$$p_i^{\text{best}}(k+1) = \begin{cases} p_i^{\text{best}}(k), & \text{if } f(x_i(k+1)) \geq f(p_i^{\text{best}}(k)), \\ x_i(k+1), & \text{if } f(x_i(k+1)) < f(p_i^{\text{best}}(k)). \end{cases} \tag{6}$$

The global best position g^{best} is determined by using (7) (Sn is the number of the particles):

$$g^{\text{best}} = \min\{f(p_i^{\text{best}})\}, \quad i = 1, 2, \ldots, Sn, \tag{7}$$

where f is the objective function.

In (4), to control the exploration and exploitation abilities of the swarm, Shi and Eberhart proposed a new parameter called as "inertia weight ω" [20]. The inertia weight controls the momentum of the particle by weighing the contribution of the previous velocity. By adding the inertia weight ω, (4) is changed

$$\begin{aligned} v_i^j(k+1) = {} & \omega * v_i^j(k) + c_1 * r_1 * [p_{i,j}^{\text{best}}(k) - x_i^j(k)] \\ & + c_2 * r_2 * [g_j^{\text{best}}(k) - x_i^j(k)]. \end{aligned} \tag{8}$$

Based on the description of PSO, we can see that the particles have a tendency to fly towards the better and better search area over the course of search process. So, the PSO algorithm can enforce a steady improvement in solution quality.

4. Hybrid Approach (ABC-PS)

From the above discussion of ABC and PSO, it is clear that the global best solution of the population does not be directly used in ABC algorithm; at the same time, it can be concluded that when the particles in the PSO get stuck in the local minima, it may not get rid of the local minima. For overcoming these disadvantages of two algorithms, we propose a hybrid global optimization approach by combing ABC algorithm and PSO searching mechanism. In this algorithm, the initial population is generated by using good point set theory.

4.1. Background on Good Point Set. Let G_d be d-dimensional unit cube in Euclidean space. If $\overline{x} = (x_1, x_2, \ldots, x_d) \in G_d$, then $0 \leq x_i \leq 1$ $(i = 1, 2, \ldots, d)$. Let $p_n(k)$ be a set of n points in G_d; then $p_n(k) = \{(x_1^{(n)}(k), \ldots, x_d^{(n)}(k)) \mid 0 \leq x_i^{(n)}(k) \leq 1, 1 \leq k \leq n, 1 \leq i \leq d\}$. Let $\overline{r} = (r_1, r_2, \ldots, r_d)$ be a point in G_d, for $N_n(\overline{r}) = N_n(r_1, r_2, \ldots, r_d)$ denoted the number of points in $p_n(k)$ which content with $0 \leq x_i^{(n)}(k) \leq r_i$, $i = 1, \ldots, d$.

Definition 1. Let $\varphi(n) = \sup_{r \in G_d} |(N_n(r)/n) - |r||$, $|r| = r_1 r_2 \cdots r_d$; then $\varphi(n)$ is called discrepancy of point set $p_n(k)$.

Definition 2. Let $\overline{x} \in G_d$; if $p_n(k) = (r_1^{(n)} * k, \ldots, r_D^{(n)} * k)$, $(k = 1, \ldots, n)$ has discrepancy $\varphi(n)$, $\varphi(n) = c(r, \epsilon) n^{(-1+\epsilon)}$, where $c(r, \epsilon)$ is constant only relate with r, ϵ $(\epsilon > 0)$, then $p_n(k)$ is called good point set and $\overline{r}$ is called good point.

Remark 3. If let $r_i = 2\cos(2\pi i / p)$, $1 \leq i \leq d$, where p is the minimum prime number content with $(p-3)/2 \geq d$, then $\overline{r}$ is a good point. If let $r_i = e^i$, $1 \leq i \leq d$, $\overline{r}$ is a good point also.

Theorem 4. *If $p_n(k)$ $(1 \leq k \leq n)$ has discrepancy $\varphi(n)$, $f \in B_d$, then*

$$\left| \int_{x \in G_d} f(x)\, dx - \sum \frac{f(p_n(i))}{n} \right| \leq v(f)\, \varphi(n), \tag{9}$$

where B_d is a d-dimensional Banach space of functions f with norm $\|\bullet\|$, $v(f) = \|f - u\|$ measures the variability of the function f.

Theorem 5. *For arbitrary $f \in B_d$, if (9) holds, then one has point $p_n(k)$ $(1 \leq k \leq n)$, in which discrepancy is not more than $\varphi(n)$.*

Theorem 6. *If $f(x)$ content with $|\partial f / \partial x_i| \leq L$, $1 \leq i \leq d$, $|\partial^2 f / \partial x_i \partial x_j| \leq L$, $1 \leq i \leq j \leq d, \ldots, |\partial^d f / \partial x_1 \cdots \partial x_d| \leq L$, where L is an absolute constant, when we want to estimate the integral of a function f over the d-dimensional unit hypercube G_d, namely $u = \int_{x \in G_d} f(x) dx$, by the average value off over any point set $p_n(k)$ $(1 \leq k \leq n)$, $Q_n = \sum (f(p_n(i))/n)$, then the integration error $E_n = u - Q_n$ is not smaller than $o(n^{-1})$.*

By Theorems 4–6, it can be seen that if we estimate the integral based on good point set, the degree of discrepancy $\varphi(n) = c(r, \epsilon) n^{-1+\epsilon}$ is only related with n. This is a good idea for high dimensional approximation computation. In other words, the idea of good point set is to take the point set more evenly than random point.

For the d-dimensional local search space H, the so-called good point set which contains n points can be found as follows:

$$p_n(k) = \{(\{r_1 * k\}, \{r_2 * k\}, \ldots, \{r_d * k\}), k = 1, \ldots, n\}, \tag{10}$$

where $r_i = 2\cos(2\pi i / \rho)$, $1 \leq i \leq d$, ρ is the minimum prime number which content with $\rho \geq 2d + 3$, and $\{r_i * k\}$ is the decimal fraction of $r_i * k$ (or with $r_i = e^i$, $1 \leq i \leq d$). Since good point set principle is based on unit hypercube or hypersphere, in order to map n good points from space $H : [0, 1]^d$ to the search space $T : [\underline{x}, \overline{x}]^d$, we define the following transformation:

$$x_i = \underline{x}_i + \{r_i * k\} * (\overline{x}_i - \underline{x}_i). \tag{11}$$

In the following, for two-dimensional space $[-1, 1]$, we generate 100 points by using goog point set method and

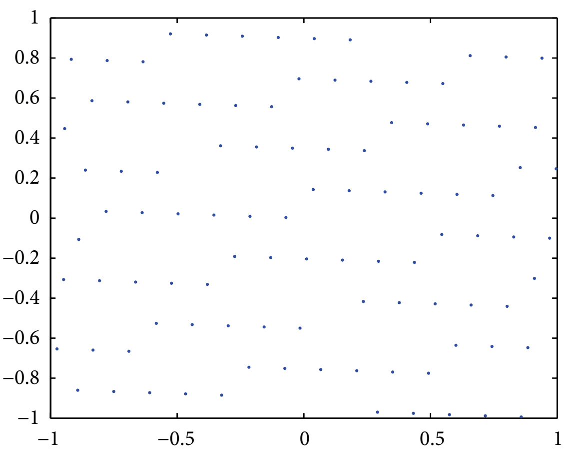

Figure 1: 100 points by using goog point set method.

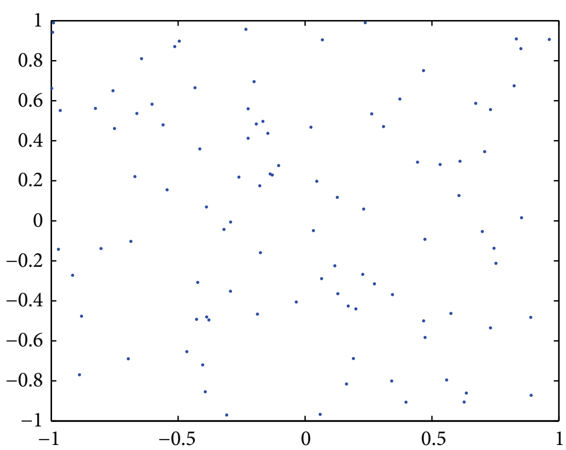

Figure 2: 100 points by using random method.

random method, respectively, and give the distribution effect of them (see Figures 1 and 2). It can be seen that good point set is uniform, and as long as the sampling number is certain, the income distribution effect is the same at every time; so good point set method has good stability.

4.2. Chaotic Search Operation. In our algorithm, assume that g^{best} is the best solution of the current iteration. To enrich the searching behavior in g^{best} and to avoid being trapped into local optimum, chaotic dynamics is incorporated into our algorithm and the detail is given as follows. Firstly, the well-known logistic equation is employed to generate a chaotic sequence, which is defined as follows:

$$\text{ch}_{i+1} = 4 * \text{ch}_i * (1 - \text{ch}_i), \quad 1 \le i \le K, \tag{12}$$

where K is the length of chaotic sequence. Then map ch_i to a chaotic vector in the interval $[\underline{x}, \overline{x}]$:

$$\text{CH}_i = \underline{x} + \text{ch}_i * (\overline{x} - \underline{x}), \quad i = 1, \ldots, K, \tag{13}$$

where $\underline{x}$ and $\overline{x}$ are the lower bound and upper bound of variable x, respectively. Finally, adopt the following equation to generate the new candidate solution $\widehat{x}$:

$$\widehat{x} = (1 - \lambda) * g^{\text{best}} + \lambda * \text{CH}_i, \quad i = 1, \ldots, K, \tag{14}$$

where λ is the shrinking factor, which is defined as follows:

$$\lambda = \frac{\text{maxcycle} - \text{iter} + 1}{\text{maxcycle}}, \tag{15}$$

where maxcycle is the maximum number of iterations and iter is the number of current iteration.

By (15), it can be seen that λ will become small with the evolution generations increasing. Furthermore, combing (13), it is easy to see that λ is smaller, less chaotic search is needed. Thus, from the above discussion, we know that the local search range becomes smaller with the process of evolution.

4.3. The Statement of ABC-PS Algorithm. Based on the above, the ABC-PC algorithm is given in this subsection.

Algorithm 7.

(1) Set the population size Sn, give the maximum number of iteration maxcycle, $w_{\max}$, $w_{\min}$, $v_{\max}$, and $v_{\min}$.

(2) Use (11) to creat an initial population $\{x_i \mid i = 1, \ldots, Sn\}$. Calculatethe function value of the population $\{f_i \mid i = 1, \ldots, Sn\}$, find the best solution g^{best} and the personal bests of the population p_i^{best}.

(3) While the stopping criterion is not meet do

(4) For $i = 1$ to Sn do % **the employed bee phase**

(5) Update the velocities of the particles and the positions of the particlesby using (8) and (5), respectively.

(6) Determine personal bests of the particles by using (6), and update trail.

(7) End if

(8) Determine the g^{best} of the population.

(9) End for

(10) For $i = 1$ to Sn do % **the onlooker phase**

(11) If rand < Prob(i)

(12) Update the velocity of the food source i and its the position by using (8) and (5).

(13) Determine personal bests of the particles by using (6), and update trail.

(14) End if

(15) If $\text{trail}_i = \max(\text{trail}) > \text{limit}$, then % **the scout phase** replace x_i with a new solution produced by (2).

(16) End if

(17) Determine the g^{best} of the population.

(18) Chaotic search K times in g^{best}, and redetermine the g^{best} of the population.

(19) iter = iter + 1.

(20) End while

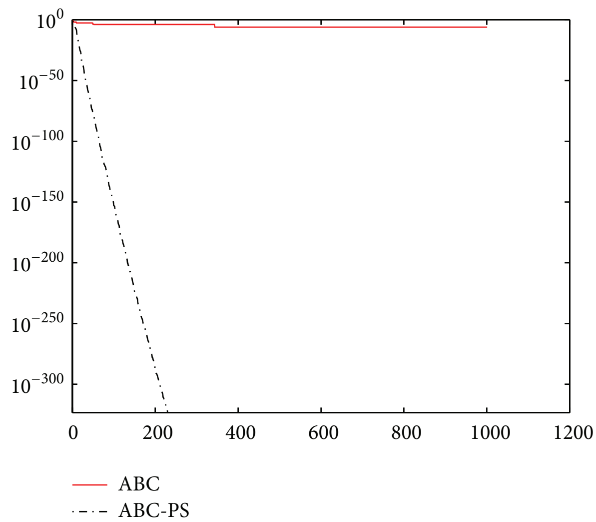

Figure 3: The relation of the best value and each iteration (the function Matyas).

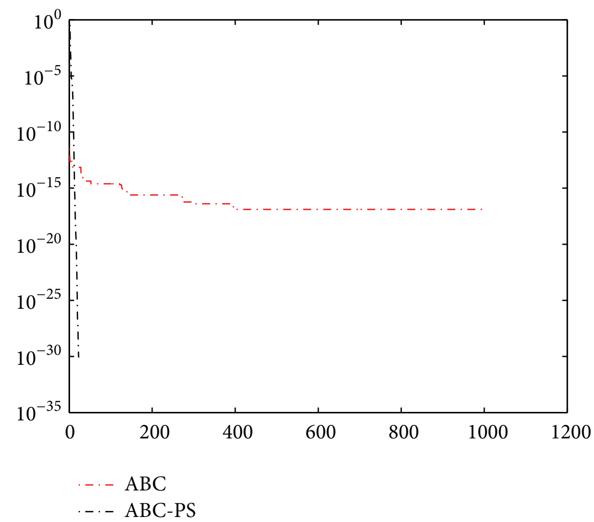

Figure 4: The relation of the best value and each iteration (the function Booth).

5. Comparison of the ABC-PS with the Other Hybrid Methods Based on the ABC

In this section, ABC-PS is applied to minimize a set of benchmark functions. In all simulations, the inertia weight in (4) is defined as follows:

$$\omega = \frac{\omega_{\max} - \omega_{\min}}{\text{maxcycle}} * \text{iter}, \tag{16}$$

where $\omega_{\max}$ and $\omega_{\min}$ are the maximum inertia weight and minimum weight, respectively, iter denotes the times of current iteration. From the above formula, it can be seen that, with the iteration increasing, the velocity v_i of the particle x_i becomes important more and more. $\omega_{\max}$ and $\omega_{\min}$ are set to 0.9 and 0.4, respectively. $v_{\min} = -1$, $v_{\max} = 1$, and $c_1 = c_2 = 1.3$.

Experiment 1. In order to evaluate the performance of ABC-PS algorithm, we have used a test bed of four traditional numerical benchmarks as illustrated in Table 1, which include Matyas, Booth, 6 Hump Camelback, and GoldsteinCPrice functions. The characteristics, dimensions, initial range, and formulations of these functions are given in Table 1. Empirical results of the proposed hybrid method have been compared with results obtained with that of basic ABC algorithm and a latest algorithm COABC [15].

The values of the common parameters used in three algorithms such as population size and total evaluation number are chosen in the same. Population size is 50 for all functions, the limit is 10. For each function, all the methods were run 30 times independently. In order to make comparison clear, the global minimums, maximum number of iterations, mean best values, standard deviations are given in Table 2. For ABC and ABC-PS, Figures 3, 4, 5, and 6 illustrate the change of the best value of each iteration. The experiment shows that the ABC-PS method is much better than the initial ABC algorithm and COABC.

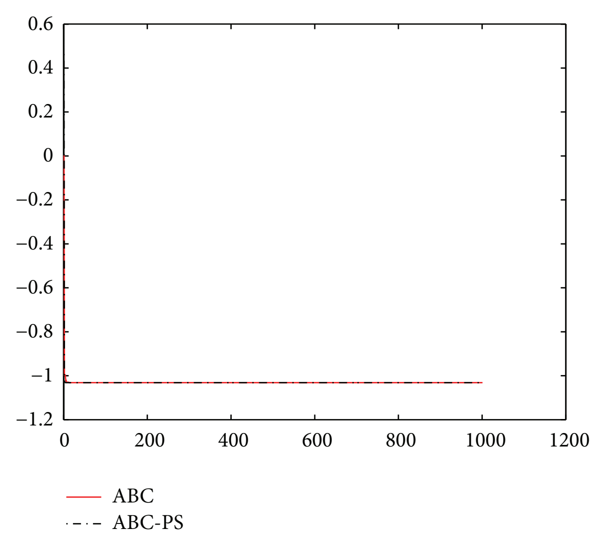

Figure 5: The relation of the best value and each iteration (the function Camelback).

Experiment 2. To further verify the performance of ABC-PS, 12 numerical benchmark functions are selected from the literatures [13–15]. This set consists of many different kinds of problems such as unimodal, multimodal, regular, irregular, separable, nonseparable, and multidimensional. The characteristics, dimensions, initial range, and formulations of these functions are listed in Table 3.

In order to fairly compare the performance of ABC-PS, COAB, GABC [13], and PABC [14], the experiments are conducted the same way as described [13–15]. The minimums, max iterations, mean best values, and standard deviations found after 30 runs are given in Table 4. The bold font in Table 4 is the optimum value among different methods. From Table 4, it can be see that the method ABC-PS is superior to other algorithms in most cases, expect to f_5 and f_6.

Table 1: Benchmark functions used in Experiment 1.

Functions	C	Range
$f_1 = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	UN	$[-10, 10]$
$f_2 = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	MS	$[-10, 10]$
$f_3 = 4x_1^2 - 2.1x_1^4 + x_1^6/3 + x_1x_2 - 4x_2^2 + 4x_2^4$	MN	$[-5, 5]$
$f_4 = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right]$ $* \left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	MN	$[-2, 2]$

C: characteristic; U: unimodal; M: multimodal; N: nonseparable; S: separable.

Table 2: Results obtained by ABC, COABC, and ABC-PS algorithms.

Function	Min	Max iteration	Algorithm	Mean	SD
			ABC	$6.03e-07$	$3.64e-07$
f_1	0	1000	COABC	$4.45e-07$	$4.63e-07$
			ABC-PS	**0**	**0**
			ABC	$1.68e-17$	$1.38e-17$
f_2	0	1000	COABC	$6.19e-23$	$2.06e-22$
			ABC-PS	**0**	**0**
			ABC	-1.03	$7.20e-17$
f_3	-1.03	1000	COABC	-1.03	$1.76e-16$
			ABC-PS	**−1.03162845348988**	**0**
			ABC	3	$1.47e-3$
f_4	3	1000	COABC	3	$3.21e-06$
			ABC-PS	**2.99999999999992**	**6.28e − 16**

Table 3: Benchmark functions used in Experiment 2.

Functions	C	D	Range
$f_1 = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	UN	2	$[-10, 10]$
$f_2 = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	UN	30	$[-30, 30]$
$f_3 = 4x_1^2 - 2.1 * x_1^4 + \frac{x_1^6}{3} + x_1x_2 - 4x_2^2 + 4x_2^4$	MN	2	$[-5, 5]$
$f_4 = -20\exp\left(-0.2 * \sqrt{\sum_{i=1}^{n}\frac{x_i^2}{n}}\right) - \exp\left(\sum_{i=1}^{n}\cos\left(2\Pi\frac{x_i}{n}\right)\right) + 20 + e$	MN	60	$[-32, 32]$
$f_5 = \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	MN	60	$[-600, 600]$
$f_6 = 418.982887 * n - \sum_{i=1}^{n}\left(x_i\sin(\sqrt{\lvert x_i\rvert})\right)$	MN	30	$[-500, 500]$
$f_7 = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2$ $+10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$	UN	4	$[-10, 10]$
$f_8 = \sum_{i=1}^{n}\lvert x_i\rvert^{i+1}$	US	30	$[-1, 1]$
$f_9 = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$	US	60	$[-10, 10]$
$f_{10} = \max\{\lvert x_i\rvert \mid 1 \le i \le n\}$	US	60	$[-100, 100]$
$f_{11} = \frac{1}{n}\sum_{i=1}^{n}\left(x_i^4 - 16x_i^2 + 5x_i\right)$	MS	60	$[-5, 5]$
$f_{12} = \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	MS	60	$[-5.12, 5.12]$

C: characteristic; U: unimodal; M: multimodal; N: nonseparable; S: separable.

Table 4: ABC-PS performance comparison of ABC and the state-of-art algorithms in [13–15].

Function	Min	Max iteration	Algorithm	Mean	SD
f_1	0	1000	ABC	$6.03e-07$	$3.64e-07$
			COABC (with UTEB = 3)	$1.42e-152$	$4.44e-152$
			ABC-PS	**0**	**0**
f_2	0	2000	ABC	0.24	0.46
			COABC (with UTEB = 3)	0.08	0.10
			ABC-PS	**5.494e − 18**	**0**
f_3	−1.03	1000	ABC	−1.03	$7.20e-17$
			COABC (with UTEB = 3)	−1.03	$2.10e-16$
			ABC-PS	**−1.03162845348988**	**0**
f_4	0	1000	ABC	3	$1.47e-3$
			COABC (with UTEB = 3)	$3.40e-13$	$6.35e-14$
			ABC-PS	**8.881e − 016**	**0**
f_5	0	2000	ABC	$4.46e-09$	$6.68e-09$
			COABC (with UTEB = 3)	**0**	**0**
			ABC-PS	$1.551e-012$	$3.468e-012$
f_6	0	1000	ABC	$2.05e+02$	$1.63e+02$
			COABC (with UTEB = 3)	**0**	**0**
			ABC-PS	$-3.637e-012$	0
f_7	0	2000	ABC	$1.71e-01$	$6.94e-02$
			COABC (with UTEB = 3)	$9.35e-03$	$4.64e-03$
			ABC-PS	**8.042e − 07**	**3.680e − 07**
f_8	0	1000	ABC	$7.69e-22$	$2.18e-21$
			PABC	$4.46e-61$	$1.09e-60$
			ABC-PS	**8.344e − 067**	**1.865e − 066**
f_9	0	1000	GABC	$4.73e-13$	$1.56e-13$
			PABC	$1.65e-18$	$1.63e-18$
			ABC-PS	**2.494e − 018**	**3.734e − 018**
f_{10}	0	1000	GABC	$6.38e-01$	$1.75e-01$
			PABC	$8.82e-01$	$1.26e-01$
			ABC-PS	**2.50e − 006**	**3.01e − 006**
f_{11}	−78.33236	1000	GABC	−78.3322	$1.32e-05$
			PABC	−78.3323	$2.62e-14$
			ABC-PS	**−78.33233**	**1.004e − 014**
f_{12}	0	1000	GABC	$9.35e-01$	$1.87e-00$
			PABC	$9.58e-03$	$6.27e-03$
			ABC-PS	**0**	**0**

6. Conclusion

In this paper, a hybrid ABC algorithm based on particle swarm searching mechanism (ABC-PS) was presented. For overcoming the disadvantage of ABC algorithm, we adopted good point set theory to generate the initial food source; then, the mechanism of PSO was utilized to search new candidate solutions for improving the exploitation ability of bee swarm; finally, the chaotic search operator was adopted in the best solution of the current iteration to increase the searching ability. The experimental results show that the ABC-PS exhibits a magnificent performance and outperforms other algorithms such as ABC, GABC, COABC, and PABC in most case.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors thank anonymous reviews for their suggestions and contributions and corresponding editor for his/her valuable efforts. The research was supported by NSFC (U1404105, 11171094); the Key Scientific and Technological Project of Henan Province (142102210058); the Doctoral Scientific Research Foundation of Henan Normal University (qd12103); the Youth Science Foundation of Henan Normal University

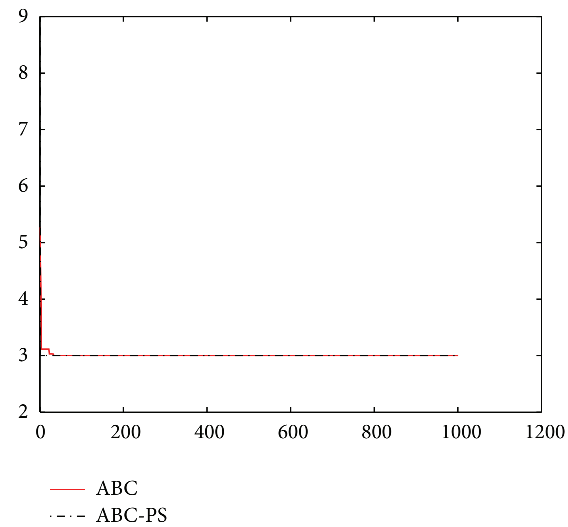

Figure 6: The relation of the best value and each iteration (the function Goldstein).

(2013qk02); Henan Normal University National Research Project to Cultivate the Funded Projects (01016400105); the Henan Normal University Youth Backbone Teacher Training.

References

[1] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, MIT Press, Cambridge, Mass, USA, 2004.

[2] T. Liao, T. Stützle, M. M. de Oca, and M. Dorigo, "A unified ant colony optimization algorithm for continuous optimization," *European Journal of Operational Research*, vol. 234, no. 3, pp. 597–609, 2014.

[3] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, November-December 1995.

[4] D. Chen and C. Zhao, "Particle swarm optimization with adaptive population size and its application," *Applied Soft Computing Journal*, vol. 9, no. 1, pp. 39–48, 2009.

[5] D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Erciyes University Press, Erciyes, Turkey, 2005.

[6] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 687–697, 2008.

[7] M. H. Aghdam, N. Ghasem-Aghaee, and M. E. Basiri, "Text feature selection using ant colony optimization," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6843–6853, 2009.

[8] B. Yagmahan and M. M. Yenisey, "Ant colony optimization for multi-objective flow shop scheduling problem," *Computers and Industrial Engineering*, vol. 54, no. 3, pp. 411–420, 2008.

[9] R. E. Perez and K. Behdinan, "Particle swarm approach for structural design optimization," *Computers and Structures*, vol. 85, no. 19-20, pp. 1579–1588, 2007.

[10] Y.-T. Kao and E. Zahara, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions," *Applied Soft Computing Journal*, vol. 8, no. 2, pp. 849–857, 2008.

[11] C. Zhang, J. Ning, S. Lu, D. Ouyang, and T. Ding, "A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization," *Operations Research Letters*, vol. 37, no. 2, pp. 117–122, 2009.

[12] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682–5687, 2010.

[13] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Applied Mathematics and Computation*, vol. 217, no. 7, pp. 3166–3173, 2010.

[14] W.-F. Gao, S.-Y. Liu, and L.-L. Huang, "A novel artificial bee colony algorithm with Powell's method," *Applied Soft Computing Journal*, vol. 13, no. 9, pp. 3763–3775, 2013.

[15] J. Luo, Q. Wang, and X. Xiao, "A modified artificial bee colony algorithm based on converge-onlookers approach for global optimization," *Applied Mathematics and Computation*, vol. 219, no. 20, pp. 10253–10262, 2013.

[16] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, no. 1, pp. 21–57, 2014.

[17] X. Shi, Y. Li, H. Li, R. Guan, L. Wang, and Y. Liang, "An integrated algorithm based on artificial bee colony and particle swarm optimization," in *Proceedings of the 6th International Conference on Natural Computation (ICNC '10)*, pp. 2586–2590, August 2010.

[18] M. El-Abd, "A hybrid ABC-SPSO algorithm for continuous function optimization," in *Proceedings of the IEEE Symposium on Swarm Intelligence (SIS '11)*, pp. 1–6, Paris, France, April 2011.

[19] J. Kennedy and R. C. Eberhart, "A new optimizer using particle swarm theory," in *Proceedings of 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, Nagoya, Japan, 1995.

[20] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, Anchorage, Alaska, USA, May 1998.