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; wangchunfengl0@126.com

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

Academic Editor: Guangming Xie

Copyright © 2014 Wang Chun-Feng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 convergencespeed, 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) algorithm [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) algorithm 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 the solution 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 opti-
mization, 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 algo-

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 those 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 onlook-
ers. 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 starts 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-dimensional, the position
of the ith food source (solution) can be expressed as a
d-dimensional vector \( x_i = (x_{i1}, x_{i2}, \ldots, x_{id}) \), \( i = 1, 2, \ldots, Sn \),
where \( Sn \) 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:

\[
x_{ij} = x_{ij} + \text{rand}(0,1) \ast (\bar{x}_j - x_{ij}),
\]

where \( i \in \{1, 2, \ldots, Sn\}, j \in \{1, 2, \ldots, d\}, \bar{x}_j \) and \( x_{ij} \) are the
lower bound and upper bound of the \( j \)th dimension, respec-
tively.

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:

\[
P_i = \frac{\text{fit}(x_i)}{\sum_{j=1}^{Sn} \text{fit}(x_j)},
\]

where \( \text{fit}(x_i) \) is the nectar amount of the \( i \)th food source and
it is associated with the objective function value \( f(x_i) \) of
the \( i \)th 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)

\[
x'_{ij} = x_{ij} + \psi \ast (x_{ij} - x_{kj}),
\]

where \( i, k \in \{1, 2, \ldots, Sn\}, k \neq i, x'_{ij} \) is a new feasible solution
that produced from its previous solution \( x_{ij} \) and the randomly
selected neighboring solution \( x_{kj} \); \( \psi \) is a random number
between \([-1, 1]\), which controls the production of a neighbor-

food source position around \( x_{ij} \); \( f \) 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 \( x_i = (x_{i1}, x_{i2}, \ldots, x_{id}), i = 1, 2, \ldots, Sn \) which
are randomly distributed 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:

\[
v'_i(k+1) = v_i(k) + c_1 \ast r_1 \ast \left[ p_i^\text{best} - x_i(k) \right]
\]

\[+ c_2 \ast r_2 \ast \left[ g_i^\text{best} - x_i(k) \right],
\]

where \( v_i(k+1) \) is the velocity of the \( i \)th particle at step \( k \); \( v_i(k) \) is the velocity
of the \( i \)th particle at step \( k \); \( p_i^\text{best} \) is the personal
best position of the \( i \)th particle at step \( k \); \( g_i^\text{best} \) 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:

\[
x_i^1(k + 1) = x_i^1(k) + v_i^1(k + 1).
\]

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

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

The global best position \( g_{\text{best}} \) is determined by using (7) \((S_n)\) is the number of the particles:

\[
g_{\text{best}} = \min \{ f(p_i^1_{\text{best}}), \quad i = 1, 2, \ldots, S_n\}, \tag{7}
\]

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 \( \omega \)” [20]. The inertia weight controls the momentum of the particle by weighing the contribution of the previous velocity. By adding the inertia weight \( \omega \), (4) is changed

\[
v_i^j(k + 1) = \omega \cdot v_i^j(k) + c_1 \cdot r_1 \cdot \left[ p_i^j_{\text{best}}(k) - x_i^j(k) \right] + c_2 \cdot r_2 \cdot \left[ g_{\text{best}}^j(k) - x_i^j(k) \right]. \tag{8}
\]

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 \( \mathcal{X} = \{x_1, x_2, \ldots, x_d\} \in G_d \), 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 \( p_n(k) = \{ (x_i^{(n)}(k), x_d^{(n)}(k)) \mid 0 \leq x_i^{(n)}(k) \leq 1, 1 \leq k \leq n, 1 \leq i \leq d \} \). Let \( \tilde{r} = (r_1, r_2, \ldots, r_d) \) be a point in \( G_d \), for \( N_r(\tilde{r}) = N_r(r_1, r_2, \ldots, r_d) \) 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 \( \phi(n) = \sup_{r \in \mathbb{G}} |N_r(n)/n - |r||, |r| = r_1 r_2 \cdots r_d \); then \( \phi(n) \) is called discrepancy of point set \( p_n(k) \).

**Definition 2.** Let \( \mathcal{X} \in G_d \); if \( p_n(k) = (r_1^{(n)}*k, \ldots, r_d^{(n)}*k) \), \( k = 1, \ldots, n \) has discrepancy \( \phi(n) \), \( \phi(n) = c(r, e)n^{-1+\epsilon} \) where \( c(r, e) \) is constant only relate with \( r, e \) \((e > 0)\), then \( p_n(k) \) is called good point set and \( \tilde{r} \) is called good point.

**Remark 3.** If let \( r_1 = 2 \cos(2\pi i/r), 1 \leq i \leq d \), where \( p \) is the minimum prime number content with \((p - 3)/2 \geq d \), then \( \tilde{r} \) is a good point. If let \( r_1 = e^i, 1 \leq i \leq d, \tilde{r} \) is a good point also.

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

\[
\left| \int_{x \in \mathcal{X}} f(x) dx - \sum_{i=1}^{n} \frac{f(p_n(i))}{n} \right| \leq \nu(f) \phi(n), \tag{9}
\]

where \( B_d \) is a \( d \)-dimensional Banach space of functions \( f \) with norm \( \| f \| \), \( \nu(f) = \| f - u \| \) 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 \( \phi(n) \).

**Theorem 6.** If \( f(x) \) content with \( |\partial f/\partial x| \leq L, 1 \leq i \leq d, |\partial^2 f/\partial x_i \partial x_j| \leq L, 1 \leq i,j \leq d, \ldots, \| \partial^2 f/\partial x_1 \partial x_d \| \leq L \), 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 offset 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 \( \phi(n) = c(r, e)n^{-1+\epsilon} \) 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 make 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:

\[
p_n(k) = \{ (r_1 \cdot k, r_2 \cdot k, \ldots, r_d \cdot k), k = 1, \ldots, n \}, \tag{10}
\]

where \( r_i = 2 \cos(2\pi i/r), 1 \leq i \leq d \), \( r \) is the minimum prime number which content with \( \rho \geq 2d + 3 \), and \( \{ r_1 \cdot k \} \) is the decimal fraction of \( r_1 \cdot k \) (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 \( T : [\mathcal{X}, \mathcal{X}]^d \), we define the following transformation:

\[
x_i = x_i + \{ r_i \cdot k \} \cdot (x_i - x_i). \tag{11}
\]

In the following, for two-dimensional space \([-1, 1]\), we generate 100 points by using good point set method and
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 \( g_{\text{best}} \) is the best solution of the current iteration. To enrich the searching behavior in \( g_{\text{best}} \) 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:

\[
ch_{i+1} = 4 \cdot ch_i \cdot (1 - ch_i), \quad 1 \leq i \leq K,
\]

where \( K \) is the length of chaotic sequence. Then map \( ch_i \) to a chaotic vector in the interval \([x_\ell, \bar{x}]\):

\[
CH_i = x_\ell + ch_i \cdot (\bar{x} - x_\ell), \quad i = 1, \ldots, K,
\]

where \( x_\ell \) and \( \bar{x} \) are the lower bound and upper bound of variable \( x \), respectively. Finally, adopt the following equation to generate the new candidate solution \( \tilde{x} \):

\[
\tilde{x} = (1 - \lambda) \cdot g_{\text{best}} + \lambda \cdot CH_i, \quad i = 1, \ldots, K,
\]

where \( \lambda \) is the shrinking factor, which is defined as follows:

\[
\lambda = \frac{\text{maxcycle} - \text{iter} + 1}{\text{maxcycle}},
\]

where maxcycle is the maximum number of iterations and \( \text{iter} \) is the number of current iteration.

By (15), it can be seen that \( \lambda \) will become small with the evolution generations increasing. Furthermore, combing (13), it is easy to see that \( \lambda \) 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 \( S_n \), give the maximum number of iteration maxcycle, \( w_{\text{max}}, w_{\text{min}}, v_{\text{max}}, \) and \( v_{\text{min}} \).
2. Use (11) to creat an initial population \( \{x_i \mid i = 1, \ldots, S_n\} \). Calculate the function value of the population \( \{f_i \mid i = 1, \ldots, S_n\} \), find the best solution \( g_{\text{best}} \) and the personal bests of the population \( p_{\text{best}} \).
3. While the stopping criterion is not meet do
   4. For \( i = 1 \) to \( S_n \) do % the employed bee phase
      5. Update the velocity of the food source \( i \) and its the position by using (8) and (5), respectively.
      6. Determine personal bests of the particles by using (6), and update trail.
   7. End if
   8. Determine the \( g_{\text{best}} \) of the population.
   9. End for
10. For \( i = 1 \) to \( S_n \) do % the onlooker phase
11. If rand < \( \text{Prob}(i) \)
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 trail \( i = \) max(trail) > limit, then % the scout phase
16. End if
17. Determine the \( g_{\text{best}} \) of the population.
18. Chaotic search \( K \) times in \( g_{\text{best}} \), and redetermine the \( g_{\text{best}} \) of the population.
19. \( \text{iter} = \text{iter} + 1 \).
20. End while

where \( \tilde{x} \) and \( \bar{x} \) are the lower bound and upper bound of variable \( x \), respectively. Finally, adopt the following equation to generate the new candidate solution \( \tilde{x} \):

\[
\tilde{x} = (1 - \lambda) \cdot g_{\text{best}} + \lambda \cdot CH_i, \quad i = 1, \ldots, K,
\]
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:

$$\omega = \frac{\omega_{\text{max}} - \omega_{\text{min}}}{\text{maxcycle}} \ast \text{iter},$$  \hspace{1cm} (16)

where $\omega_{\text{max}}$ and $\omega_{\text{min}}$ 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. $\omega_{\text{max}}$ and $\omega_{\text{min}}$ are set to 0.9 and 0.4, respectively. $v_{\text{min}} = -1$, $v_{\text{max}} = 1$, 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 I, which include Matyas, Booth, 6 Hump Camelback, and GoldsteinCPrice functions. The characteristics, dimensions, initial range, and formulations of these functions are given in Table I. 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.

**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, except to $f_5$ and $f_6$. 
Table 1: Benchmark functions used in Experiment 1.

<table>
<thead>
<tr>
<th>Functions</th>
<th>CR</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1 = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$</td>
<td>UN</td>
<td>$[-10, 10]$</td>
</tr>
<tr>
<td>$f_2 = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$</td>
<td>MS</td>
<td>$[-10, 10]$</td>
</tr>
<tr>
<td>$f_3 = 4x_1^2 - 2.1x_1^3 + x_1x_2 - 4x_2^2 + 4x_2^4$</td>
<td>MN</td>
<td>$[-5, 5]$</td>
</tr>
<tr>
<td>$f_4 = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_2 - 14x_2^2 + 6x_1x_2 + 3x_2^2)\right]$</td>
<td>MN</td>
<td>$[-2, 2]$</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Min</th>
<th>Max iteration</th>
<th>Algorithm</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>0</td>
<td>1000</td>
<td>ABC</td>
<td>$6.03e-07$</td>
<td>$3.64e-07$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COABC</td>
<td>$4.45e-07$</td>
<td>$4.63e-07$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$f_2$</td>
<td>0</td>
<td>1000</td>
<td>ABC</td>
<td>$1.68e-17$</td>
<td>$1.38e-17$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COABC</td>
<td>$6.19e-23$</td>
<td>$2.06e-22$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$f_3$</td>
<td>$-1.03$</td>
<td>1000</td>
<td>ABC</td>
<td>$-1.03$</td>
<td>$7.20e-17$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COABC</td>
<td>$-1.03$</td>
<td>$1.76e-16$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>$-1.03$</td>
<td>0</td>
</tr>
<tr>
<td>$f_4$</td>
<td>3</td>
<td>1000</td>
<td>ABC</td>
<td>3</td>
<td>$1.47e-3$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COABC</td>
<td>3</td>
<td>$3.21e-06$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>2.99999999999992</td>
<td>$6.28e-16$</td>
</tr>
</tbody>
</table>

Table 3: Benchmark functions used in Experiment 2.

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

* 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].

<table>
<thead>
<tr>
<th>Function</th>
<th>Min</th>
<th>Max iteration</th>
<th>Algorithm</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>0</td>
<td>1000</td>
<td>ABC</td>
<td>6.03e−07</td>
<td>3.64e−07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COABC (with UTEB = 3)</td>
<td>1.42e−152</td>
<td>4.44e−152</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$f_2$</td>
<td>0</td>
<td>2000</td>
<td>ABC</td>
<td>0.24</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COABC (with UTEB = 3)</td>
<td>0.08</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>5.494e−18</td>
<td>0</td>
</tr>
<tr>
<td>$f_3$</td>
<td>−1.03</td>
<td>1000</td>
<td>ABC</td>
<td>−1.03</td>
<td>7.20e−17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COABC (with UTEB = 3)</td>
<td>−1.03</td>
<td>2.10e−16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>−1.03162845348988</td>
<td>0</td>
</tr>
<tr>
<td>$f_4$</td>
<td>0</td>
<td>1000</td>
<td>ABC</td>
<td>3</td>
<td>1.47e−3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COABC (with UTEB = 3)</td>
<td>3.40e−13</td>
<td>6.35e−14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>8.881e−016</td>
<td>0</td>
</tr>
<tr>
<td>$f_5$</td>
<td>0</td>
<td>2000</td>
<td>ABC</td>
<td>4.46e−09</td>
<td>6.68e−09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COABC (with UTEB = 3)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>1.551e−012</td>
<td>3.468e−012</td>
</tr>
<tr>
<td>$f_6$</td>
<td>0</td>
<td>1000</td>
<td>ABC</td>
<td>2.05e+02</td>
<td>1.63e+02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COABC (with UTEB = 3)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>−3.637e−012</td>
<td>0</td>
</tr>
<tr>
<td>$f_7$</td>
<td>0</td>
<td>2000</td>
<td>ABC</td>
<td>1.71e−01</td>
<td>6.94e−02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COABC (with UTEB = 3)</td>
<td>9.35e−03</td>
<td>4.64e−03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>8.042e−07</td>
<td>3.680e−07</td>
</tr>
<tr>
<td>$f_8$</td>
<td>0</td>
<td>1000</td>
<td>ABC</td>
<td>7.69e−22</td>
<td>2.18e−21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PABC</td>
<td>4.46e−61</td>
<td>1.09e−60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>8.344e−067</td>
<td>1.865e−066</td>
</tr>
<tr>
<td>$f_9$</td>
<td>0</td>
<td>1000</td>
<td>GABC</td>
<td>4.73e−13</td>
<td>1.56e−13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PABC</td>
<td>1.65e−18</td>
<td>1.63e−18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>2.494e−018</td>
<td>3.734e−018</td>
</tr>
<tr>
<td>$f_{10}$</td>
<td>0</td>
<td>1000</td>
<td>GABC</td>
<td>4.73e−13</td>
<td>1.56e−13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PABC</td>
<td>1.65e−18</td>
<td>1.63e−18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>2.50e−006</td>
<td>3.01e−006</td>
</tr>
<tr>
<td>$f_{11}$</td>
<td>−78.33236</td>
<td>1000</td>
<td>GABC</td>
<td>−78.33232</td>
<td>1.32e−05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PABC</td>
<td>−78.3323</td>
<td>2.62e−14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>−78.33233</td>
<td>1.004e−014</td>
</tr>
<tr>
<td>$f_{12}$</td>
<td>0</td>
<td>1000</td>
<td>GABC</td>
<td>9.35e−03</td>
<td>1.87e−00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PABC</td>
<td>9.58e−03</td>
<td>6.27e−03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABC-PS</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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 (14210221005); 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).

References

Submit your manuscripts at http://www.hindawi.com