Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

Development and investigation of efficient artificial bee colony algorithm for numerical function optimization

Guoqiang Li [a,b,*], Peifeng Niu [a,b], Xingjun Xiao [a]

[a] Institute of Electrical Engineering, Yanshan University, Qinhuangdao 066004, China
[b] Key Lab of Industrial Computer Control Engineering of Hebei Province, Yanshan University, Qinhuangdao 066004, China

A R T I C L E   I N F O

Article history:
Received 9 April 2011
Received in revised form 11 July 2011
Accepted 14 August 2011
Available online 22 August 2011

Keywords:
Artificial bee colony
Gbest-guided ABC
Biological-inspired optimization
Optimization

A B S T R A C T

Artificial bee colony algorithm (ABC), which is inspired by the foraging behavior of honey bee swarm, is a biological-inspired optimization. It shows more effective than genetic algorithm (GA), particle swarm optimization (PSO) and ant colony optimization (ACO). However, ABC is good at exploration but poor at exploitation, and its convergence speed is also an issue in some cases. For these insufficiencies, we propose an improved ABC algorithm called I-ABC. In I-ABC, the best-so-far solution, inertia weight and acceleration coefficients are introduced to modify the search process. Inertia weight and acceleration coefficients are defined as functions of the fitness. In addition, to further balance search processes, the modification forms of the employed bees and the onlooker ones are different in the second acceleration coefficient. Experiments show that, for most functions, the I-ABC has a faster convergence speed and better performances than each of ABC and the gbest-guided ABC (GABC). But I-ABC could not still substantially achieve the best solution for all optimization problems. In a few cases, it could not find better results than ABC or GABC. In order to inherit the bright sides of ABC, GABC and I-ABC, a high-efficiency hybrid ABC algorithm, which is called PS-ABC, is proposed. PS-ABC owns the abilities of prediction and selection. Results show that PS-ABC has a faster convergence speed like I-ABC and better search ability than other relevant methods for almost all functions.

1. Introduction

Optimization problems frequently encountered in multitudinous applications are at the heart of engineering design [1,2], economics [3], statistical physics [4], information theory and computer science, etc. [5–7]. However it is very widely believed that, for many actual optimization problems, searching optimal solutions is very extreme hardness and sometimes completely beyond any current or projected computational capacity. So there has been a growing interest in various algorithms [8,9] developed and investigated for two decades, especially biological-inspired optimization algorithms such as genetic algorithm (GA) [10,11], particle swarm optimization (PSO) [12–16], ant colony optimization (ACO) [17] and artificial bee colony (ABC) [18]. These algorithms have been adopted by researchers so far and are well suited to solve various complex computational problems such as optimization of objective functions [19–21], pattern recognition [22], filter modeling [23,24].

ABC recently proposed by D. Karaboga is a biological-inspired optimization algorithm which mimics the foraging behavior of honey bee swarm. This algorithm has been applied to search an optimum solution in many optimization problems. In some researches, the performance of ABC has already been compared with other search optimization techniques such as (GA), PSO, ACO, Differential Evolution (DE) algorithm. Various numerical benchmark functions, which consist of unimodal and multimodal distributions, are employed to evaluate the performance of ABC. The comparison results showed that ABC can find a better solution, and is more effective than other optimization techniques. The exploration and exploitation are extremely important mechanisms in ABC. However, there are still some insufficiencies, namely, ABC is good at exploration but poor at exploitation and its convergence speed is also an issue in some cases. The exploration process is related to the ability of independently seeking for the global optimum, while the exploitation process is related to the ability of applying the existing knowledge to look for better solutions. In order to further balance and accelerate the two processes, a few modified or improved algorithms [25] based on the foraging behavior of honey bee swarm are proposed in recent years, such as best-so-far ABC [26], gbest-guided ABC (GABC) [27], Bee Swarm Optimization (BSO) [28]. These modified ABCs have better performances than the original ABC.

However, there is no specific algorithm to substantially achieve the best solution for all optimization problems. Some algorithms only give a better solution for some particular problems than others. Hence, searching for a well improved or new optimization method

* Corresponding author at: Key Lab of Industrial Computer Control Engineering of Hebei Province, Yanshan University, Qinhuangdao 066004, China.
E-mail address: zhihuiyuang@163.com (G. Li).

1568-4946/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2011.08.040

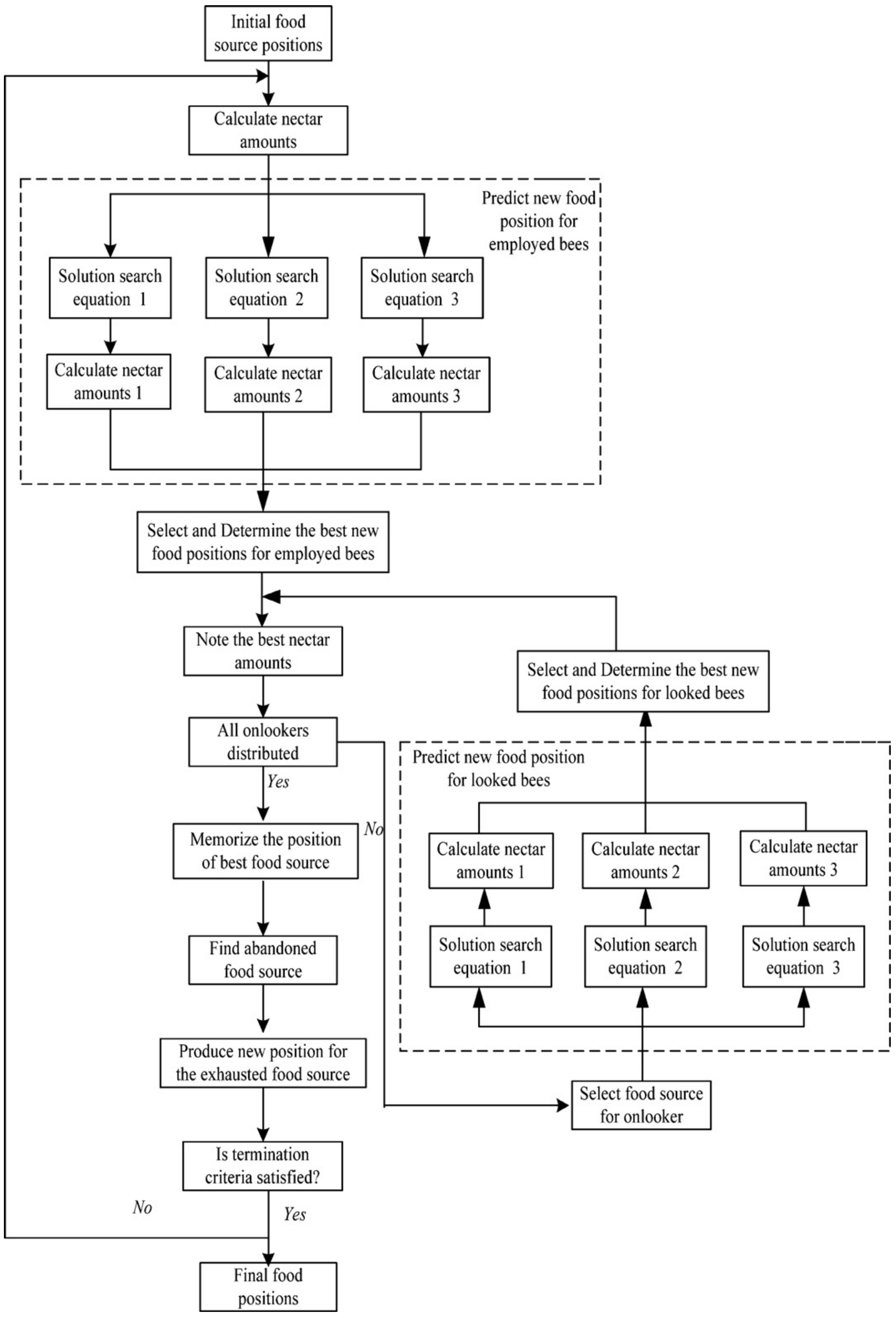
Initial food source positions
Calculate nectar amounts
Predict new food position for employed bees
Solution search equation 1
Solution search equation 2
Solution search equation 3
Calculate nectar amounts 1
Calculate nectar amounts 2
Calculate nectar amounts 3
Select and Determine the best new food positions for employed bees
Note the best nectar amounts
Select and Determine the best new food positions for looked bees
All onlookers distributed
Yes
No
Predict new food position for looked bees
Memorize the position of best food source
Find abandoned food source
Produce new position for the exhausted food source
Select food source for onlooker
Is termination criteria satisfied?
No
Yes
Final food positions

Fig. 1. Flowchart of the PS-ABC algorithm.

is very necessary. In this paper, we propose a high-efficiency ABC method to further improve the performance of ABC. The improved ABC, which is called I-ABC, has an extremely fast convergence speed. There are three major differences between ABC and I-ABC. Namely, the best-so-far solution, inertia weight and acceleration coefficients are introduced to modify the search process, and inertia weight and acceleration coefficients are defined as functions of the fitness. In addition, to further balance the exploitation and exploration processes, the modification forms of the employed bees and the onlooker ones are different in the second acceleration coefficient. Simulation results show that the I-ABC could not only find the global optimal values for many numerical benchmark functions, but also own an extremely fast convergence speed.

However, in only a few cases, the I-ABC traps in local optimal solutions and cannot find better solutions than ABC or GABC. In

Table 1
High-dimensional classical benchmark functions.

Test function	S
$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
$f_2(\vec{x}) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	$[-10, 10]^n$
$f_3(\vec{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]^n$
$f_4(\vec{x}) = \max_i \{\lvert x_i\rvert, \quad 1 \le i \le n\}$	$[-100, 100]^n$
$f_5(\vec{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^n$
$f_6(\vec{x}) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	$[-100, 100]^n$
$f_7(\vec{x}) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]^n$
$f_8(\vec{x}) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i\rvert})$	$[-500, 500]^n$
$f_9(\vec{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^n$
$f_{10}(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^n$
$f_{11}(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$
$f_{12}(\vec{x}) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	$[-50, 50]^n$
$y_i = 1 + \frac{x_i+1}{4};\ u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	
$f_{13}(\vec{x}) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]^n$

order to combine the bright sides of I-ABC, ABC and GABC, the paper proposes a high-efficiency hybrid ABC algorithm which has the abilities of prediction and selection. This hybrid optimization method is called as PS-ABC. In PS-ABC algorithm, every employed bee or looked bee could predict their next solutions by three different solution search equations, and select the best one. Then the bee could decide where they should go to explore or exploit. Simulation results show that the PS-ABC could not only have the advantages of I-ABC, but also own the bright sides of ABC and GABC. Namely, PS-ABC owns an extremely fast convergence speed like I-ABC and very good search performance.

Recently, the use of advanced soft computing techniques [29–43] is increasingly required in multitudinous applications for processing huge amounts of uncertain data. Soft computing, which is a consortium of powerful tools including Neural Networks, Fuzzy Systems, Evolutionary Computing, Swarm Intelligence and Probabilistic Reasoning, deals with imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low solution cost. Soft computing is being used successfully in many areas including pattern recognition [44,45], complex process control and optimization [46–48], medical diagnosis and engineering [49–51], stock market prediction [52]. Unlike most soft computing techniques, I-ABC and PS-ABC do not need gradient information of the objective function. They are proposed to improve the basic ABC algorithm in order to jump out of the local optimum as well as speed up the process of finding the optimal parameters. In addition, they are capable of producing low cost, fast, and reasonably accurate solutions to complex problems. Thus, they may be used, like other heuristic intelligence optimization algorithms, in multitudinous applications in order to enhance the performance of the entire process. The two proposed methods may be adopted to optimize other soft computing tools in order to make them effective and robust in coping with uncertainty, insufficient information, and noise. They could be taken as new optimization techniques which could not only play the role of compensation for soft computing, but also promote the development, investigation and application of other soft computing methods in the near future.

The paper is organized as follows: the original ABC algorithm is reviewed in Section 2. Section 3 describes the proposed I-ABC algorithm, which has an extremely fast convergence speed. Section 4 describes the PS-ABC algorithm, which has the abilities of prediction and selection. The simulation results are shown in Section 5. Finally, Section 6 makes a summary of this paper.

Table 2
Classical benchmark functions with fix dimension.

Test function	S
$f_{14}(\vec{x}) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{\sum_{i=1}^{2}(x_i - a_j)^6}\right)^{-1}$	$[-65.53, 65.53]^2$
$f_{15}(\vec{x}) = \sum_{i=1}^{11}\left(a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right)^2$	$[-5, 5]^4$
$f_{16}(\vec{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]^2$
$f_{17}(\vec{x}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos\ x_1 + 10$	$[-5, 10] \times [0, 15]$
$f_{18}(\vec{x}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	$[-5, 5]$
$f_{19}(\vec{x}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]^3$
$f_{20}(\vec{x}) = \sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]^6$
$f_{21}(\vec{x}) = \sum_{i=1}^{5} [(\vec{x} - a_i)(\vec{x} - a_i)^T + c_i]^{-1}$	$[0, 10]^4$
$f_{22}(\vec{x}) = \sum_{i=1}^{7} [(\vec{x} - a_i)(\vec{x} - a_i)^T + c_i]^{-1}$	$[0, 10]^4$
$f_{23}(\vec{x}) = \sum_{i=1}^{10} [(\vec{x} - a_i)(\vec{x} - a_i)^T + c_i]^{-1}$	$[0, 10]^4$

2. The original ABC algorithm

The artificial bee colony algorithm was proposed by D. Karaboga which is inspired by the foraging behavior of bees. This technique is very straightforward, robust and population-based on stochastic optimization algorithm. In the ABC algorithm, the colony of artificial bees is divided into three groups: employed bees, onlookers and scouts. Half of the colony consists of the employed bees, and another half consists of the onlookers. The position of a food source corresponds to a possible solution to the optimization problem, and the nectar amount of each food source represents their quality (fitness) of the associated solution. The number of the employed bees equals to the number of food sources. When a food source has been abandoned by bees, the abandoned employed bee would become a scout. In the first place, the ABC algorithm would generate a randomly distributed initial population $P(C=0)$ of SN food source positions, where SN is the size of food sources. Every solution $x_i (i = 1, 2, \ldots, SN)$ is a D-dimensional vector, where D denotes the number of optimization parameters. After the initialization, the population of solutions is subject to repeated cycles $C = 1, 2, \ldots, MCN$ of the search courses of employed bees, onlookers and scouts. An employed bee could produce a modification on the solution in its memory depending on local information and test its fitness value of the new source. If the fitness value of the new one is better than that of the previous one, the employed bee would memorize the new position and forget the previous one. Otherwise it keeps the position of the previous one in its memory. When all employed bees complete the search process, they will share the information about nectar amounts and positions of food sources with onlookers. An onlooker evaluates the nectar information which is owned by all employed bees, and then chooses a food source with a probability which is related to the nectar amount. As in the case of the employed bee, the onlooker can produce a modification on the position in its memory and check the nectar amount of the candidate source. If the nectar amount is more than that of the previous one, the bee would memorize the new position and forgets the previous one.

Table 3
Optima in functions of Table 2.

Function	X_{opt}	f_{opt}
f_{14}	$(-32, 32)$	1
f_{15}	$(0.1928, 0.1908, 0.1231, 0.1358)$	0.00030
f_{16}	$(0.089, -0.712), (-0.089, 0.712)$	−1.0316
f_{17}	$(-3.14, 12.27), (3.14, 2.275), (9.42, 2.42)$	0.398
f_{18}	$(0, -1)$	3
f_{19}	$(0.114, 0.556, 0.852)$	−3.86
f_{20}	$(0.201, 0.15, 0.477, 0.275, 0.311, 0.657)$	−3.32
f_{21}	5 local minima in $a_{ij}, j = 1, 2, \ldots, 5$	−10.1532
f_{22}	7 local minima in $a_{ij}, j = 1, 2, \ldots, 7$	−10.4028
f_{23}	10 local minima in $a_{ij}, j = 1, 2, \ldots, 10$	−10.5363

An onlooker chooses a food source completely depending on the probability value associated with the food source p_i, which is calculated by the following form:

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j} \tag{1}$$

where fit_i denotes the fitness value of the ith solution which is proportional to the nectar amount of the food source in the ith position. For the sake of producing a new food position from the previous one, the ABC could adopt the following modification form:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{2}$$

where $k \in \{1, 2, \ldots, SN\}$ and $j \in \{1, 2, \ldots, D\}$ are random generating indexes, but k must be different from i. ϕ_{ij} is a random number between $[-1, 1]$. It can control to produce a new food source around x_{ij} and represent the comparison of two food positions visually by a bee. As can be seen from Eq. (2), as the difference between the parameter x_{ij} and x_{kj} decreases, the perturbation on the position x_{ij} is also decreased. Thus, when the search approaches to the optimum solution in the search space, the step size is adaptively reduced. If a parameter value produced by the operation exceeds its predetermined limit, it is set to its limit value.

The food source which is abandoned by the bees would be replaced with a new food source found by scouts. In ABC algorithm, the foraging behavior is simulated by randomly producing a position and replacing the abandoned one with a new one. If a position cannot be improved further through a predetermined number of cycles, the food source should be abandoned. The predetermined number of cycles is an important control parameter in ABC algorithm, which is called “limit” for abandonment. Suppose that the abandoned source is x_i and $j \in \{1, 2, \ldots, D\}$, then the scout finds a new food source to be replaced with x_i. This operation can be defined as:

$$x_i^j = x_{\min}^j + rand(0, 1)(x_{\max}^j - x_{\min}^j) \tag{3}$$

After each new source position v_{ij} produced, it can be evaluated by the artificial bee, and its fitness is compared with that of its previous one. If the new food source equals or is better than the old one, it would be replaced with the previous one in its memory. Otherwise, the old one is retained in its memory. In other words, a greedy selection mechanism is employed as the selection operation between the old and the new one.

It is clear from the above explanation that there are three important control parameters in the ABC algorithm: the number of food sources (*SN*), the value of ‘limit’ and the maximum cycle number (*MCN*).

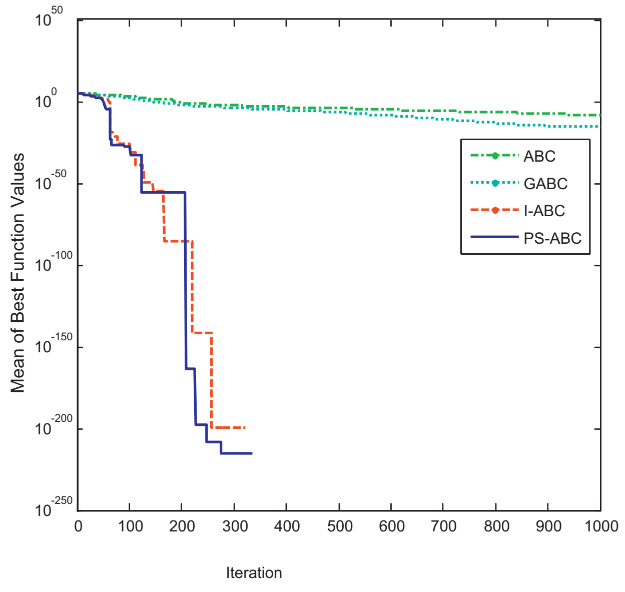

Fig. 2. Comparison of performance of 4 algorithms for minimization of f_1 with dim = 30.

Table 4
The mean and the standard deviations of the function values.

Function	Dim	ABC			IABC		
		C.I.	Mean	SD	C.I.	Mean	SD
f_1	20	998	6.1871×10^{-16}	2.1149×10^{-16}	208	0	0
	30	1000	3.6239×10^{-9}	5.8543×10^{-9}	322	0	0
	50	1000	1.1172×10^{-5}	1.2547×10^{-5}	726	0	0
f_2	20	1000	1.3569×10^{-10}	7.1566×10^{-11}	20	0	0
	30	1000	5.1168×10^{-6}	2.2351×10^{-6}	30	0	0
	50	1000	2.9232×10^{-3}	9.0507×10^{-4}	61	0	0
f_3	20	1000	3.1312×10^{3}	1.1869×10^{3}	1000	4.5409×10^{3}	2.6919×10^{3}
	30	1000	1.2412×10^{4}	3.0071×10^{3}	999	1.4344×10^{4}	2.7291×10^{3}
	50	1000	4.5746×10^{4}	6.4571×10^{3}	1000	4.6927×10^{4}	7.3584×10^{3}
f_4	20	1000	3.9602	1.3702	412	0	0
	30	1000	24.5694	5.6587	1000	1.207×10^{-197}	0
	50	1000	56.3380	4.8387	999	25.5055	5.6662
f_5	20	1000	1.1114	1.7952	1000	15.7165	1.4013
	30	1000	4.5509	4.8776	1000	26.4282	1.3956
	50	1000	48.0307	46.6576	1000	47.0287	0.8601
f_6	20	1000	5.5519×10^{-16}	1.6986×10^{-16}	999	6.3129×10^{-16}	2.1344×10^{-16}
	30	1000	2.4932×10^{-9}	3.6826×10^{-9}	999	3.8463×10^{-10}	2.3239×10^{-10}
	50	1000	1.3655×10^{-5}	1.7532×10^{-5}	1000	1.8434×10^{-5}	1.7466×10^{-5}
f_7	20	985	6.5059×10^{-2}	2.0264×10^{-2}	998	8.7159×10^{-3}	3.2459×10^{-3}
	30	1000	1.5639×10^{-1}	4.6539×10^{-2}	1000	1.9609×10^{-2}	9.3459×10^{-3}
	50	995	4.8847×10^{-1}	1.0767×10^{-1}	998	8.8306×10^{-2}	2.5540×10^{-2}
f_8	20	1000	−8327.49	60.3307	999	−8323.77	74.0475
	30	1000	−12,130.31	159.147	999	−12,251.03	166.741
	50	1000	−19,326.50	266.383	1000	−19,313.49	277.438
f_9	20	1000	1.4052×10^{-11}	4.0524×10^{-11}	52	0	0
	30	1000	0.45305	0.51495	84	0	0
	50	1000	8.4433	2.6951	172	0	0
f_{10}	20	1000	2.8343×10^{-9}	2.5816×10^{-9}	72	8.8817×10^{-16}	0
	30	1000	2.7591×10^{-5}	2.1321×10^{-5}	156	8.8817×10^{-16}	0
	50	1000	4.7018×10^{-2}	3.3957×10^{-2}	306	8.8817×10^{-16}	0
f_{11}	20	1000	3.7196×10^{-3}	6.6186×10^{-3}	472	0	0
	30	1000	3.8168×10^{-3}	8.4583×10^{-3}	684	0	0
	50	1000	1.1971×10^{-2}	1.9763×10^{-2}	696	0	0
f_{12}	20	996	4.0612×10^{-16}	9.4282×10^{-17}	997	4.1780×10^{-16}	1.0882×10^{-16}
	30	1000	1.1823×10^{-10}	2.5585×10^{-10}	1000	7.1096×10^{-12}	5.2513×10^{-12}
	50	1000	8.9552×10^{-6}	3.2107×10^{-5}	999	5.4223×10^{-7}	2.9821×10^{-7}
f_{13}	20	991	6.9303×10^{-8}	2.9269×10^{-7}	989	1.7549×10^{-16}	4.5409×10^{-16}
	30	1000	2.2763×10^{-7}	4.1239×10^{-7}	995	4.7831×10^{-8}	2.0359×10^{-7}
	50	1000	1.3563×10^{-5}	2.7822×10^{-5}	998	2.4156×10^{-5}	4.3568×10^{-5}

Mean, mean of objective values; C.I., convergence iteration.

Table 5
The mean and the standard deviations of the function values.

Function	Dim	GABC			PS-ABC		
		C.I.	Mean	SD	C.I.	Mean	SD
f_1	20	739	3.1943×10^{-16}	7.3909×10^{-17}	178	0	0
	30	986	6.2643×10^{-16}	1.0859×10^{-16}	336	0	0
	50	1000	1.2546×10^{-5}	6.0511×10^{-9}	628	0	0
f_2	20	1000	9.3611×10^{-16}	1.3278×10^{-16}	16	0	0
	30	1000	1.3019×10^{-10}	4.6859×10^{-11}	22	0	0
	50	1000	2.3671×10^{-5}	6.1989×10^{-6}	74	0	0
f_3	20	1000	2.6919×10^{3}	1.4619×10^{3}	1000	1.0369×10^{3}	6.1065×10^{2}
	30	1000	1.0939×10^{4}	2.5670×10^{3}	1000	6.1069×10^{3}	1.6947×10^{3}
	50	1000	4.1236×10^{4}	5.8269×10^{3}	1000	3.0118×10^{4}	4.1073×10^{3}
f_4	20	1000	0.3325	1.0786	440	0	0
	30	1000	12.6211	2.6556	1000	8.591×10^{-115}	4.705×10^{-114}
	50	1000	45.3075	4.3151	1000	19.6683	6.3094
f_5	20	1000	1.6769	2.9037	1000	0.5190	1.0764
	30	1000	7.47961	19.0926	1000	1.5922	4.4066
	50	1000	25.7164	31.75811	1000	34.4913	30.3412
f_6	20	721	3.3386×10^{-16}	1.0154×10^{-16}	851	2.6147×10^{-16}	3.8684×10^{-17}
	30	997	6.4499×10^{-16}	1.1126×10^{-16}	671	5.7169×10^{-16}	8.2549×10^{-17}
	50	1000	5.6529×10^{-9}	3.6854×10^{-9}	1000	1.1674×10^{-15}	1.4114×10^{-16}
f_7	20	996	3.3120×10^{-2}	7.9296×10^{-3}	943	6.5241×10^{-3}	2.2508×10^{-3}
	30	983	8.4786×10^{-2}	2.7907×10^{-2}	997	2.1514×10^{-2}	6.8816×10^{-3}
	50	1000	2.4609×10^{-1}	4.7278×10^{-2}	966	6.5309×10^{-2}	1.7705×10^{-2}
f_8	20	1000	−8355.92	72.2604	987	−8379.66	4.7288×10^{-12}
	30	1000	−12,407.29	106.409	995	−12,564.23	22.5354
	50	1000	−19,975.29	230.858	1000	−20,887.98	80.3524
f_9	20	836	0	0	50	0	0
	30	1000	3.3165×10^{-2}	1.8165×10^{-1}	80	0	0
	50	1000	2.1733	1.0728	140	0	0
f_{10}	20	1000	2.7533×10^{-14}	3.5832×10^{-15}	84	8.8817×10^{-16}	0
	30	1000	7.7828×10^{-10}	2.9817×10^{-10}	188	8.8817×10^{-16}	0
	50	1000	1.1137×10^{-4}	3.8873×10^{-5}	365	8.8817×10^{-16}	0
f_{11}	20	1000	6.0279×10^{-4}	2.2313×10^{-3}	502	0	0
	30	1000	6.9655×10^{-4}	2.2609×10^{-3}	834	0	0
	50	1000	1.0470×10^{-3}	2.7482×10^{-3}	836	0	0
f_{12}	20	702	3.2621×10^{-16}	6.6721×10^{-17}	875	2.5576×10^{-16}	4.9715×10^{-17}
	30	938	5.8570×10^{-16}	1.1349×10^{-16}	625	5.5312×10^{-16}	8.6858×10^{-17}
	50	1000	9.3017×10^{-11}	7.9664×10^{-11}	979	1.0252×10^{-15}	1.5815×10^{-16}
f_{13}	20	998	6.5528×10^{-8}	2.4413×10^{-7}	788	2.3456×10^{-18}	2.2088×10^{-18}
	30	996	2.1724×10^{-7}	5.6676×10^{-7}	567	6.0601×10^{-18}	5.6064×10^{-18}
	50	999	8.8776×10^{-7}	1.5324×10^{-6}	959	5.0541×10^{-17}	1.5350×10^{-16}

Mean, mean of objective values; C.I., convergence iteration.

Pseudo-code of the ABC algorithm is given below:

```
(1) Initialize the population of solutions $x_{ij}$, $i = 1, 2, \ldots, SN$, $j = 1, 2, \ldots, D$
(2) Evaluate the fitness the population
(3) $cycle = 1$
(4) repeat
(5) For each employed bee{
        Produce new solutions $v_{ij}$ by using (2) and evaluate them
        Adopt the greedy selection mechanism}
(6) Calculate the probability values $p_{ij}$ for the solutions $v_{ij}$ by (1)
(7) For each onlooker{
        Produce the new solutions $v_{ij}$ from the selected solution
        $x_{ij}$ depending on $p_{ij}$ and evaluate them
        Adopt the greedy selection mechanism}
(8) Determine the abandoned solution for the scout, if exists, and
    replace it with a new randomly produced solution $x_{ij}$ by (3)
(9) Memorize the best solution so far
(10) $cycle = cycle + 1$
(11) until $cycle = MCN$
```

3. The I-ABC algorithm

In ABC algorithm, the exploration process refers to the ability of seeking for the global optimum in the solution space of various unknown optimization problems, while the exploitation process refers to the ability of applying the knowledge of previous solutions to look for better solutions. However, the processes of the exploration and the exploitation contradict with each other, so the two abilities should be well balanced for achieving good optimization performance. According to the search form of ABC algorithm which is described as Eq. (2), a candidate solution would be generated by moving the previous one towards another solution selected randomly from the population. The ABC algorithm has already proved to be a very effective technique for solving global optimization. ABC is not only a high performance optimizer which is very easy to understand and implement, but also requires little computational bookkeeping. However, ABC could be slow to converge and sometimes trap in a local optimal solution. In order to further improve the performances of ABC, we propose to make three major changes by introducing the best-so-far solution, inertia weight and acceleration coefficients to modify the search process. In addition, the search form of ABC described as Eq. (2) is good at exploration but poor at exploitation. Therefore, to improve the exploitation, the modification forms of the employed bees and the onlooker ones are different in the second acceleration coefficient. The improved ABC algorithm is called as I-ABC.

The operation process can be modified as the following form:

$$v_{ij} = x_{ij}w_{ij} + 2(\phi_{ij} - 0.5)(x_{ij} - x_{kj})\Phi_1 + \varphi_{ij}(x_j - x_{kj})\Phi_2 \quad (4)$$

where v_{ij} is the new feasible solution that is an modified feasible solution depending on its previous solution x_{ij}. w_{ij} is the inertia weight which controls impacts of the previous solution x_{ij}. x_j is the jth parameter of the best-so-far solution, ϕ_{ij} and φ_{ij} are random

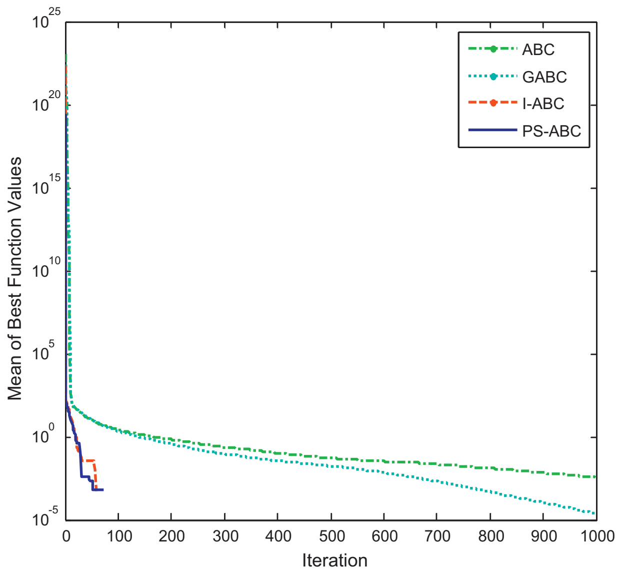

Fig. 3. Comparison of performance of 4 algorithms for minimization of f_2 with dim = 50.

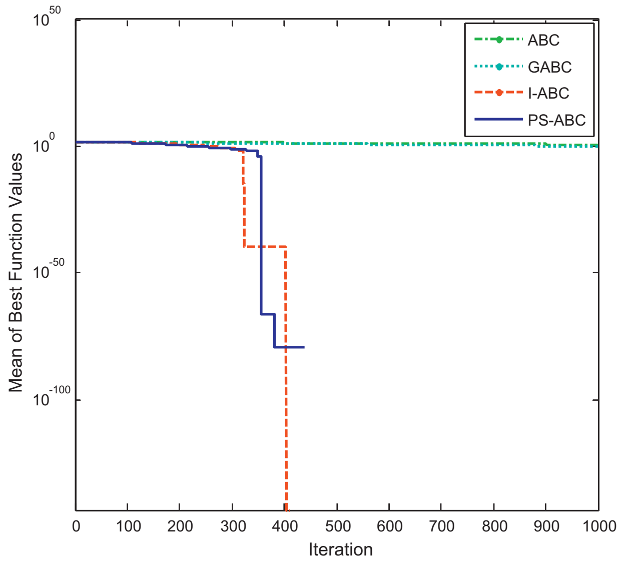

Fig. 5. Comparison of performance of 4 algorithms for minimization of f_4 with dim = 20.

numbers between [0, 1], Φ_1 and Φ_2 are positive parameters that could control the maximum step size. However, if the global fitness is very large, bees are far away from the optimum values. So a big correction is needed to search the global optimum solution and then w, Φ_1 and Φ_2 should be bigger values. Conversely, only a small modification is needed, then w, Φ_1 and Φ_2 must be smaller values. So in order to further improve the search efficiency of the bees, we propose to modify the parameters that are used to calculate new candidate food sources. In the paper, inertia weight and acceleration coefficients are defined as functions of the fitness in the search process of ABC. They are proposed as follows:

$$w_{ij} = \Phi_1 = \frac{1}{(1+\exp(-fitness(i)/ap))} \tag{5}$$

$$\Phi_2 = \begin{cases} 1 & \text{if a bee is employed one} \\ \dfrac{1}{(1+\exp(-fitness(i)/ap))} & \text{if a bee is looked one} \end{cases} \tag{6}$$

where ap is the $fitness(1)$ in the first iteration. In order to further balance the processes of the exploration and the exploitation, the modification forms of the employed bees and the onlooker ones are different in the acceleration coefficient Φ_2. The main advantages of I-ABC are to achieve a fast convergence speed and to find a good solution.

Pseudo-code of the IABC algorithm is given below:

```
(1) Initialize the population of solutions $x_{ij}$, $i = 1, 2, \ldots, SN$, $j = 1, 2, \ldots, D$
(2) Evaluate the fitness the population
(3) $cycle = 1$
(4) repeat
(5) If $cycle = 1$ then $ap = fitness(1)$
(6) For each employed bee{
        Produce new solutions $v_{ij}$ by Eqs. (4) and (5) where $\Phi_2 = 1$
        and then evaluate them
        Adopt the greedy selection mechanism}
```

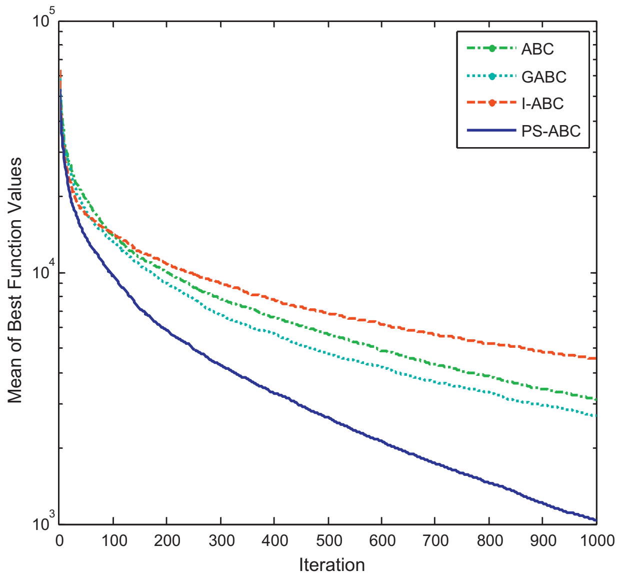

Fig. 4. Comparison of performance of 4 algorithms for minimization of f_3 with dim = 20.

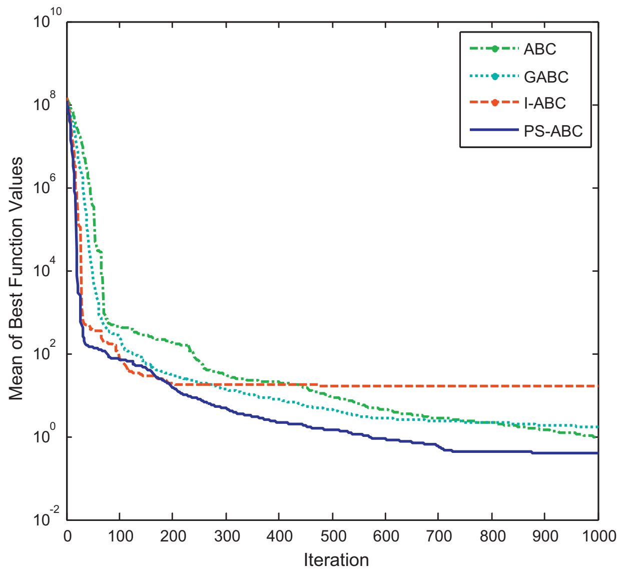

Fig. 6. Comparison of performance of 4 algorithms for minimization of f_5 with dim = 20.

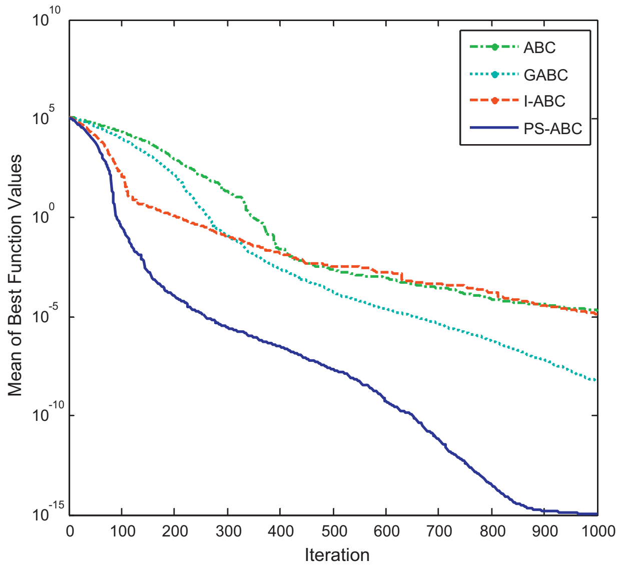

Fig. 7. Comparison of performance of 4 algorithms for minimization of f_6 with dim = 50.

```
(7) Calculate the probability values $p_{ij}$ for the solutions $x_{ij}$ by (1)
(8) For each onlooker{
        Produce the new solutions $v_{ij}$ by Eqs. (4)–(6) from the selected solution $x_{ij}$ depending on $p_{ij}$ and evaluate them
        Adopt the greedy selection mechanism}
(9) Determine the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution $x_{ij}$ by (3)
(10) Memorize the best solution so far
(11) $cycle = cycle + 1$
(12) until $cycle = MCN$
```

4. The PS-ABC algorithm

Simulation results, which are described in Section 5, show that the I-ABC could not only find the global optimal values for many numerical benchmark functions which consist of unimodal and multimodal distributions, but also own an extremely fast convergence speed. However, in only a few cases, the I-ABC does not

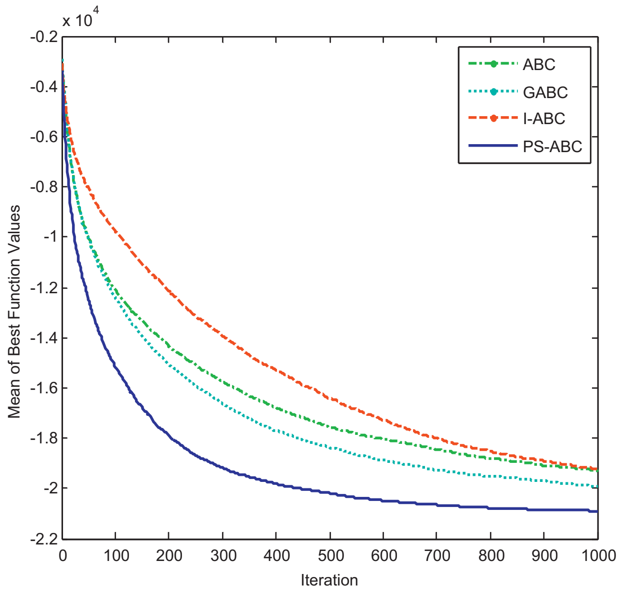

Fig. 9. Comparison of performance of 4 algorithms for minimization of f_8 with dim = 50.

overcome the problem of local optimal solutions and even find worse solutions than ABC or GABC.

However, for successful application to optimization problems, a population-based optimization algorithm that realizes rapid convergence and high diversity is required. Simultaneously, a good population-based optimization algorithm should have a stable performance regardless of initial population selection. In addition, for various optimization problems, ABC, GABC and I-ABC show different advantages in various fields such as search optimization ability, diversity, convergence speed and so on.

In order to achieve these goals and combine the advantages of I-ABC, ABC and GABC in various fields, the paper proposes a compounding high-efficiency ABC algorithm with the abilities of prediction and selection, which is called as PS-ABC. In initialization, PS-ABC like ABC starts by associating all employed bees with randomly generated food sources. After initialization, the population of the food sources is subject to repeated cycles of

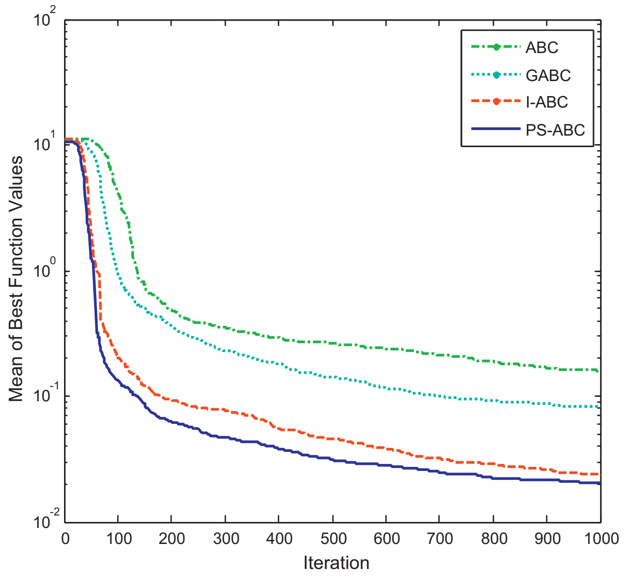

Fig. 8. Comparison of performance of 4 algorithms for minimization of f_7 with dim = 30.

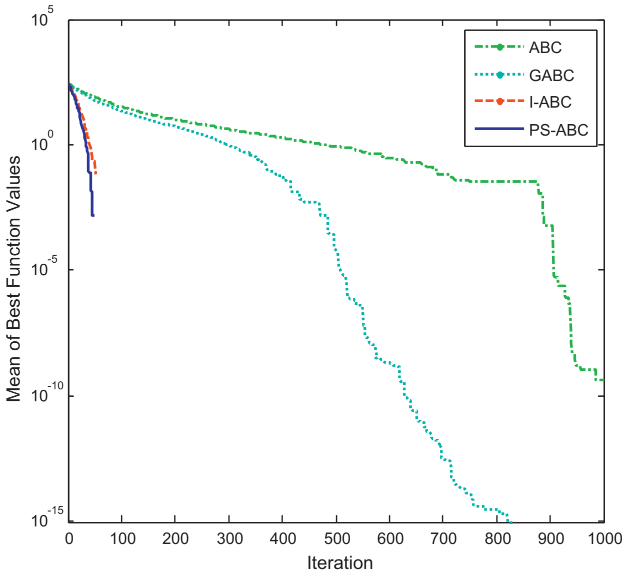

Fig. 10. Comparison of performance of 4 algorithms for minimization of f_9 with dim = 20.

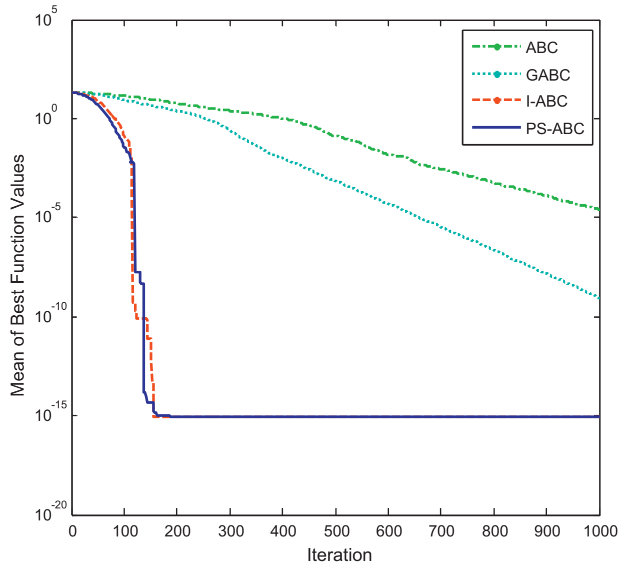

Fig. 11. Comparison of performance of 4 algorithms for minimization of f_{10} with dim = 30.

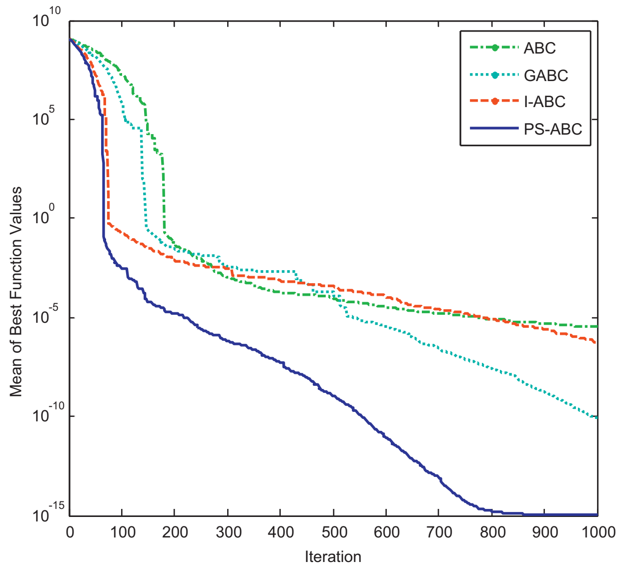

Fig. 13. Comparison of performance of 4 algorithms for minimization of f_{12} with dim = 50.

the search processes of the employed bees, the onlooker bees and the scout bees. The main difference between PS-ABC and anyone of ABC, I-ABC and GABC is how bees get the candidate solutions. In PS-ABC, an employed bee firstly works out three new solutions by three different solution search equations, and then chooses and determines the best one as the candidate solution. Here, due to calculating the candidate solution before the employed bees decide where they should go to explore, the process of calculating new food positions is called 'predict'. After bees 'predict' new candidate solutions by three different solution search equations, they select the best one from the three solutions as the candidate solution.

If the fitness value of the candidate solution is better than the best fitness value achieved so far, then the employed bee moves to this new food source and synchronously abandons the old one, otherwise it remains the previous food source in its mind. When all employed bees have finished this process, they share the fitness information with the onlookers, each of which selects a food source according to probability given in Eq. (1). As in the case of the employed bee, an onlooker 'predicts' three modifications on the position in her memory, and then selects the best one as the candidate source and checks the fitness value of the candidate source. Providing that the fitness value of the candidate source is better than that of the previous one, the bee would memorize the new position and forget the old one.

With this scheme, good food sources will get more onlookers than the bad ones. Each bee will search for better food source around neighborhood patch for a certain number of cycles, and if the fitness value will not improve, then that bee becomes scout bee.

The main difference between PS-ABC and anyone of ABC, I-ABC and GABC is how to determine the candidate solutions process. In PS-ABC algorithm, there are three different solution search equations. The first one is Eq. (2), which is the solution modification form of the original ABC Algorithm. The Second one is the Eq. (4), which is proposed in Section 3 of this paper. The third one is the GABC

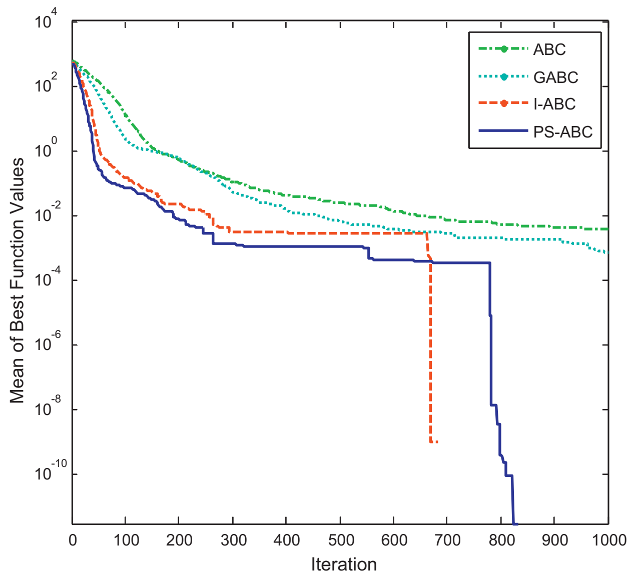

Fig. 12. Comparison of performance of 4 algorithms for minimization of f_{11} with dim = 30.

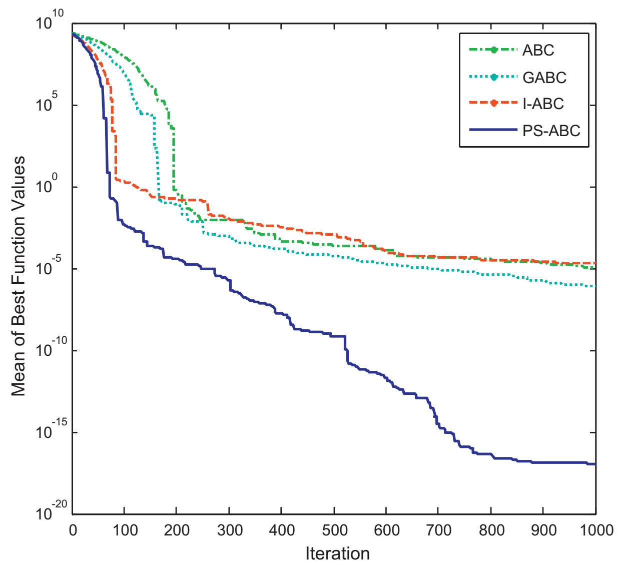

Fig. 14. Comparison of performance of 4 algorithms for minimization of f_{13} with dim = 50.

which is presented in [27]. The solution search equation of GABC is given by the following form:

$$v_{ij} = x_{ij} + 2(\phi_{ij} - 0.5)(x_{ij} - x_{kj}) + \varphi_{ij}(x_j - x_{kj}) \quad (7)$$

where v_{ij} is the new feasible solution that is an modified feasible solution depending on its previous solution x_{ij}. x_j is the jth parameter of the best-so-far solution, ϕ_{ij} is a random number between $[0, 1]$, φ_{ij} $[0, c]$, c is a nonnegative constant, which is set 1 in this paper.

The flowchart of the PS-ABC algorithm is shown in Fig. 1. The configuration of the three consisting methods, ABC, GABC and I-ABC, is investigated to improve global search ability and simultaneously enhance the fast convergence of PS-ABC. In PS-ABC, the three solution search equations are independently calculated, but influence each other by the chosen best solution. On the whole, the PS-ABC has inherited the bright sides of the other three algorithms.

5. Experimental study and discussion

To evaluate the performances of the I-ABC and the PS-ABC, we apply them to 23 classical benchmark functions [9]. These functions are presented in Tables 1 and 2. In Table 1, n is the dimension of classical functions, S denotes a subset of R^n. The global minimum values of 13 classical benchmark functions of Table 1 are zeros, except for the function $f_8(\vec{x})$ which has a minimum value of $-418.9829 \times n$, The optimum location for 13 functions presented in Table 1 is in $[0]^n$, except for $f_6(\vec{x})$ in $[-0.5]^n$ and $f_8(\vec{x})$ in $[420.96]^n$. Functions $f_1(\vec{x})$ to $f_7(\vec{x})$ are unimodal high-dimensional functions. Functions $f_8(\vec{x})$ to $f_{13}(\vec{x})$ are multimodal high-dimensional ones. In Table 2, functions $f_{14}(\vec{x})$ to $f_{23}(\vec{x})$ are multimodal test functions with fix dimension. The theoretical minimum values of these functions of Table 2 are shown in Table 3. A detailed symbol description of the functions of Table 2 is given in Appendix A.

The performances of the I-ABC and the PS-ABC are compared with those of the original ABC and GABC algorithms. In PS-ABC, the maximum number of cycles is set as 1000 for all functions; the number of colony size is taken as 40 and 'limit' equals to 200. These values of control parameters of PS-ABC equal to these of the I-ABC, ABC and GABC. In addition, for every benchmark function of Table 1, the dimension is set 20, 30 and 50 in turn.

Every experiment is repeated 30 times each starting from a random population with different random seeds, and the mean of best function values achieved has been recorded. The mean and the standard deviations of the function values of Table 1 are given in Tables 4 and 5. In addition, the mean and convergence iteration of the function of Table 2 are shown in Table 6.

5.1. Compare I-ABC with ABC and GABC

As seen from Tables 4–5 and Figs. 2–14, the I-ABC has found the global optimal values on 5 functions (f_1, f_2, f_4, f_9, and f_{11}), except for the function f_4 under dim = 50. On 6 functions (f_6, f_7, f_8, f_{10}, f_{12}, and f_{13}), the I-ABC could achieve the solutions quite close to the global optima. However, on two functions (f_3, f_5), the I-ABC could not search the best solutions under the specified maximum number of cycles.

Compared to ABC, I-ABC could achieve much better solutions than ABC on 10 functions (f_1, f_2, f_4, f_6, f_7, f_9, f_{10}, f_{11}, f_{12}, and f_{13}), while ABC could own better search performance than I-ABC on only 3 functions (f_3, f_5, and f_8), except for f_8 under dim = 30.

Compared to GABC, the I-ABC shows better performance than GABC on 8 functions (f_1, f_2, f_4, f_7, f_9, f_{10}, f_{11}, and f_{13}), except for f_{13} under dim = 50. While GABC outperforms I-ABC on 5 functions (f_3, f_5, f_6, f_8, and f_{12}). Both methods have their own relative merits.

As seen from Table 6 and Figs. 15–17, ABC has the same search ability as ABC and GABC for functions (f_{14}, f_{16}, f_{18}, f_{19}, f_{20}, f_{21}, f_{22},

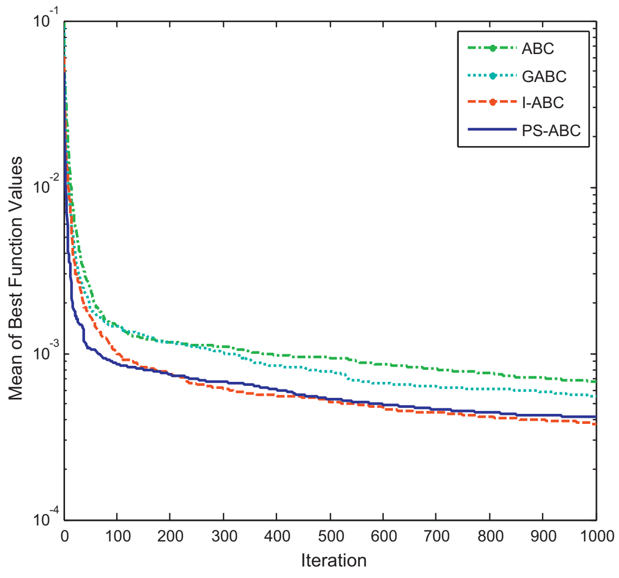

Fig. 15. Comparison of performance of 4 algorithms for minimization of f_{15}.

and f_{23}), but better than others for functions (f_{15}, f_{17}). In addition, I-ABC shows faster convergence speed on functions (f_{14}, f_{15}, f_{16}, f_{17}, f_{18}, f_{21}, f_{22}, and f_{23}). However, for functions f_{19} and f_{20}, I-ABC shows a little slower convergence speed than others.

As seen from Tables 4–6 and Figs. 2–17, although I-ABC could not find better solutions than ABC or GABC on only a few functions of Table 1 and has smaller descend gradient in the initial stage of the optimization process for functions of Table 2, the I-ABC could have a faster convergence speed and better search performances than ABC or GABC on most functions, especially for the functions (f_1, f_2, f_4, f_9, f_{10}, f_{11}, f_{15}, and f_{17}). So I-ABC has shown a very high efficiency on optimization problems.

5.2. Compare PS-ABC with ABC, GABC and I-ABC

As seen from Tables 4–6 and Figs. 2–17, it can be seen that the results of PS-ABC are the theoretical global optima on 5 functions (f_1, f_2, f_4, f_9, and f_{11}) except for f_4 under dim = 50. Meanwhile, these results of PS-ABC are extremely close to the theoretical optima on 12 functions (f_6, f_{10}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{18}, f_{19}, f_{20}, f_{21}, and f_{22}).

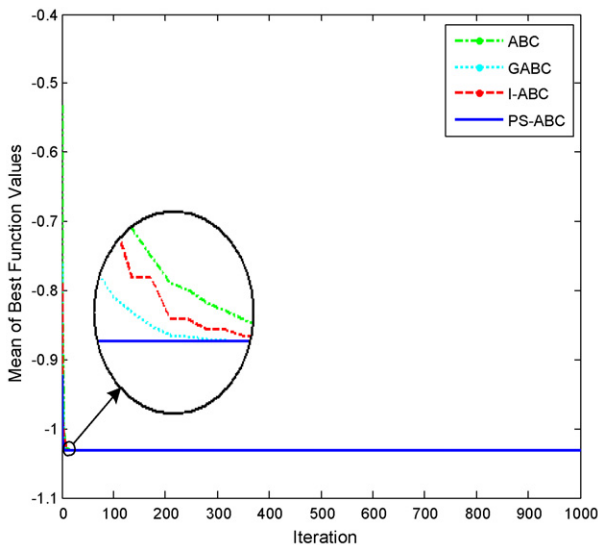

Fig. 16. Comparison of performance of 4 algorithms for minimization of f_{16}.

Table 6
The mean and convergence iteration of the function of Table 2.

Function	ABC		GABC		I-ABC		PS-ABC	
	C.I.	Mean	C.I.	Mean	C.I.	Mean	C.I.	Mean
f_{14}	214	0.9980	203	0.9980	295	0.9980	83	0.9980
f_{15}	1000	6.7445×10^{-4}	998	5.5425×10^{-4}	993	3.7589×10^{-4}	997	4.1386×10^{-4}
f_{16}	125	−1.0316	84	−1.0316	97	−1.0316	48	−1.0316
f_{17}	951	0.7012	999	0.6212	91	0.3978	996	0.6300
f_{18}	999	3.001	613	3.0000	396	3.0000	997	3.0000
f_{19}	278	−3.8628	76	−3.8628	347	−3.8628	79	−3.8628
f_{20}	699	−3.3220	341	−3.3220	553	−3.3220	224	−3.3220
f_{21}	895	−10.1532	539	−10.1532	483	−10.1532	213	−10.1532
f_{22}	944	−10.4029	528	−10.4029	617	−10.4029	480	−10.4029
f_{23}	998	−10.5364	999	−10.5364	613	−10.5364	1000	−10.5364

Mean, mean of objective values; C.I., convergence iteration.

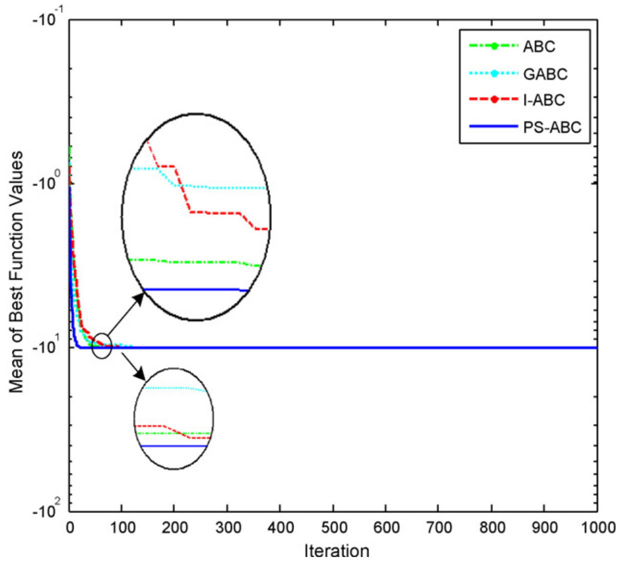

Fig. 17. Comparison of performance of 4 algorithms for minimization of f_{21}.

Compared with GABC and ABC, PS-ABC is superior to GABC and ABC in term of searching quality and derivation of the results for all functions except for f_5 under dim = 50 and f_{17}. Although PS-ABC could not find the best solution on the only functions f_5 under dim = 50 and f_{17}, the difference between the best solution and the one achieved by PS-ABC is not large. In addition, the PS-ABC, which have a faster convergence speed than ABC or GABC especially for the functions (f_1, f_2, f_4, f_9, f_{10}, and f_{11}), is similar to I-ABC in the convergence speed.

For all functions of Table 1, both I-ABC and PS-ABC could search the some optima on some functions (f_1, f_2, f_9, f_{10}, and f_{11}), In addition, PS-ABC have much better performances than I-ABC on the other functions (f_3, f_4, f_5, f_6, f_7, f_8, f_{12}, and f_{13}), except for f_4 under dim = 30. Although I-ABC is better than PS-ABC on the only function f_4 under dim = 30, the difference is extreme minuteness. So they could be thought as the same optimal values. So, it is concluded that PS-ABC is more effective than I-ABC for high-dimensional classical benchmark functions.

For functions of Table 2, I-ABC has find better solutions than PS-ABC on functions (f_{15}, f_{17}), and same search performance as PS-ABC on functions (f_{14}, f_{16}, f_{18}, f_{19}, f_{20}, f_{21}, f_{22}, and f_{23}). But PS-ABC could has faster convergence speed than I-ABC on all functions of Table 2, except for functions (f_{19}, f_{20}).

On the whole, I-ABC has very good search performance and extremely fast convergence speed on most functions. However, there are still some insufficiencies. Namely, sometimes, I-ABC trapped in local optimal solutions and cannot find better solutions than ABC or GABC in only a few cases of Table 1. It has smaller descent gradient in the initial stage of the optimization process for functions of Table 2.

Although the computational complexity of PS-ABC is a little bigger than the others, the PS-ABC inherits the bright sides of the other three algorithms. Namely, the PS-ABC owns a very fast convergence speed of I-ABC, much larger descend gradient and better search ability than any of other associated methods for almost all functions. In addition, the proposed hybrid algorithm shows stable search ability regardless of selection of the initial population. So it is concluded that the PS-ABC is much more effective.

6. Conclusion

The I-ABC is proposed to improve the performance of ABC and simultaneously quicken the convergence speed. Both PS-ABC and I-ABC are applied to 13 classical test function problems. Although the I-ABC could not find better solutions than ABC or GABC in only a few functions, the I-ABC could have a faster convergence speed and better performances than ABC or GABC for most functions. Therefore, the I-ABC could be thought as a very efficient optimization algorithm.

The sophisticated PS-ABC algorithm is proposed to apply to optimization problems in this paper. The configurations, which consist of three methods, ABC, GABC and I-ABC, are investigated to improve the global search ability and simultaneously enhance convergence speed of PS-ABC. The results indicate that the search ability of PS-ABC has been improved. In addition, the new hybrid algorithm shows the very fast convergence like I-ABC. In PS-ABC, the three search equations are independently calculated, but influence each other by the chosen best solution. On the whole, the PS-ABC could be thought as the combination of the bright sides of the other algorithms.

Acknowledgments

Project supported by the National Natural Science Foundation of China (Grant No. 60774028) and Natural Science Foundation of Hebei Province, China (Grant No. F2010001318).

Appendix A.

See Tables A1–A7.

Table A1
a_{ij} in f_{14}.

$$(a_{ij}) = \begin{pmatrix} -32, -16, 0, 16, 32, -32, \ldots, 0, 16, 32 \\ -32, -32, -32, -32, -16, \ldots, 32, 32, 32 \end{pmatrix}$$

Table A2
a_i and b_i in f_{15}.

i	1	2	3	4	5	6	7	8	9	10	11
a_i	0.1957	0.1947	0.1735	0.1600	0.0844	0.0627	0.0456	0.0342	0.0342	0.0235	0.0246
b_i^{-1}	0.25	0.5	1	2	4	6	8	10	12	14	16

Table A3
a_i and c_i in f_{19}.

i	$a_{ij}, j = 1, 2, 3$			c_i
1	3	10	30	1
2	0.1	10	35	1.2
3	3	10	30	3
4	0.1	10	30	1.2

Table A4
p_{ij} in f_{19}.

i	$p_{ij}, j = 1, 2, 3$		
1	0.3689	0.1170	0.2673
2	0.499	0.4387	0.7470
3	0.1091	0.8732	0.5547
4	0.03815	0.5743	0.8828

Table A5
a_{ij} and c_i in f_{20}.

i	$a_{ij}, j = 1, 2, 3, 4, 5, 6$						c_i
1	10	3	17	3.5	1.7	8	1
2	0.05	10	17	0.1	8	14	1.2
3	3	3.5	1.7	10	17	8	3
4	17	8	0.05	10	0.1	14	3.2

Table A6
p_{ij} in f_{20}.

i	$p_{ij}, j = 1, 2, 3, 4, 5, 6$					
1	0.131	0.169	0.556	0.012	0.828	0.588
2	0.232	0.413	0.830	0.373	0.100	0.999
3	0.234	0.141	0.352	0.288	0.304	0.665
4	0.404	0.882	0.873	0.574	0.109	0.038

Table A7
a_{ij} and c_i in f_{21}, f_{22}, f_{23}.

i	$a_{ij}, j = 1, 2, 3$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

References

[1] P. Pawar, R. Rao, J. Davim, Optimization of process parameters of milling process using particle swarm optimization and artificial bee colony algorithm, in: International Conference on Advances in Mechanical Engineering, 2008.

[2] R.S. Rao, S. Narasimham, M. Ramalingaraju, Optimization of distribution network configuration for loss reduction using artificial bee colony algorithm, International Journal of Electrical Power and Energy Systems Engineering (IJEPESE) 1 (2008) 116–122.

[3] L. dos Santos Coelho, V.C. Mariani, A novel chaotic particle swarm optimization approach using Henon map and implicit filtering local search for economic load dispatch, Chaos, Solitons and Fractals 39 (2009) 510–518.

[4] Q.-K. Pan, M.F. Tasgetiren, P.N. Suganthan, T.J. Chua, A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, Information Sciences 181 (2011) 2455–2468.

[5] D. Karaboga, C. Ozturk, A novel clustering approach: Artificial Bee Colony (ABC) algorithm, Applied Soft Computing 11 (2011) 652–657.

[6] L. Ozbakir, A. Baykasoglu, P. Tapkan, Bees algorithm for generalized assignment problem, Applied Mathematics and Computation 215 (2010) 3782–3795.

[7] A. Kaveh, S. Talatahari, Size optimization of space trusses using Big Bang–Big Crunch algorithm, Computers and Structures 87 (2009) 1129–1140.

[8] O.K. Erol, I. Eksin, A new optimization method: Big Bang–Big Crunch, Advances in Engineering Software 37 (2006) 106–111.

[9] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: a gravitational search algorithm, Information Sciences 179 (2009) 2232–2248.

[10] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

[11] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley, Boston, 1989.

[12] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995.

[13] B. Liu, L. Wang, Y.-H. Jin, F. Tang, D.-X. Huang, Improved particle swarm optimization combined with chaos, Chaos, Solitons and Fractals 25 (2005) 1261–1271.

[14] H. Modares, A. Alfi, M.-M. Fateh, Parameter identification of chaotic dynamic systems through an improved particle swarm optimization, Expert Systems with Applications 37 (2010) 3714–3720.

[15] G.I. Tsoulos, A. Stavrakoudis, Enhancing PSO methods for global optimization, Applied Mathematics and Computation 216 (2010) 2988–3001.

[16] T. Xiang, X. Liao, K.-W. Wong, An improved particle swarm optimization algorithm combined with piecewise linear chaotic map, Applied Mathematics and Computation 190 (2007) 1637–1645.

[17] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics 26 (1) (1996) 29–41.

[18] D. Karaboga, An idea based on honey bee swarm for numerical optimization. Erciyes University, Kayseri, Turkey, Technical Report-TR06, 2005.

[19] D. Karaboga, B. Basturk, Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems LNCS: Advances in Soft Computing-Foundations of Fuzzy Logic and Soft Computing, vol. 4529, Springer-Verlag, 2007, pp. 789–798.

[20] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Applied Mathematics and Computation 214 (2009) 108–132.

[21] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8 (2008) 687–697.

[22] D. Karaboga, B. Akay, C. Ozturk, Artificial Bee Colony (ABC) optimization algorithm for training feed-forward neural networks LNCS: Modeling Decisions for Artificial Intelligence, vol. 4617, Springer-Verlag, 2007, pp. 318–329.

[23] Y.-M. Huang, J.-C. Lin., A new bee colony optimization algorithm with idle-time-based filtering scheme for open shop-scheduling problems, Expert Systems with Applications 38 (2011) 5438–5447.

[24] N. Karaboga, A new design method based on artificial bee colony algorithm for digital IIR filters, Journal of the Franklin Institute 346 (2009) 328–348.

[25] B. Akay, D. Karaboga, A modified Artificial Bee Colony algorithm for real-parameter optimization, Information Sciences, in press.

[26] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, The best-so-far selection in Artificial Bee Colony algorithm, Applied Soft Computing 11 (2011) 2888–2901.

[27] G. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Applied Mathematics and Computation 217 (2010) 3166–3173.

[28] R. Akbari, A. Mohammadi, K. Ziarati, A novel bee swarm optimization algorithm for numerical function optimization, Communications in Nonlinear Science and Number Simulation 15 (2010) 3142–3155.

[29] K. Ziarati, R. Akbari, V. Zeighami, On the performance of bee algorithms for resource-constrained project scheduling problem, Applied Soft Computing 11 (2010) 3720–3733.

[30] M. Ma, J. Liang, M. Guo, Y. Fan, Y. Yin, SAR image segmentation based on Artificial Bee Colony algorithm, Applied Soft Computing, doi:10.1016/j.asoc.2011.05.039, in press.

[31] I.M.S. de Oliveira, R. Schirru, Swarm intelligence of artificial bees applied to In-Core Fuel Management Optimization, Applied Soft Computing 38 (2011) 1039–1045.

[32] T. Dereli, G.S. Das, A hybrid 'bee(s) algorithm' for solving container loading problems, Applied Soft Computing 11 (2011) 2854–2862.

[33] A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, Applied Soft Computing 9 (2009) 625–631.

[34] M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Applied Soft Computing 11 (2011) 2406–2418.

[35] T. Davidović, D. Ramljak, M. Šelmić, D. Teodorović, Bee colony optimization for the p-center problem, Applied Soft Computing 38 (2011) 1367–1376.

[36] K. Ziarati, R. Akbari, V. Zeighami, On the performance of bee algorithms for resource-constrained project scheduling problem, Applied Soft Computing 11 (2011) 3720–3733.
[37] M. Ko, A. Tiwari, J. Mehnen, A review of soft computing applications in supply chain management, Applied Soft Computing 10 (2010) 661–674.
[38] C. Kalloniatis, P. Belsis, S. Gritzalis, A soft computing approach for privacy requirements engineering: the PriS framework, Applied Soft Computing 11 (2011) 4341–4348.
[39] K. Manimala, K. Selevi, R. Ahila, Hybrid soft computing techniques for feature selection and parameter optimization in power quality data mining, in press.
[40] B. Peña, E. Teruel, L.I. Díez, Soft-computing models for soot-blowing optimization in coal-fired utility boilers, Applied Soft Computing 11 (2011) 1657–1668.
[41] R. Gil-Pita, L. Cuadra, E. Alexandre, D. Ayllón, L. Alvarez, M. Rosa-Zurera, Enhancing the energy efficiency of wireless-communicated binaural hearing aids for speech separation driven by soft-computing algorithms, Applied Soft Computing, doi:10.1016/j.asoc.2011.03.022, in press.
[42] S. Shrivastava, M.P. Singh, Performance evaluation of feed-forward neural network with soft computing techniques for hand written English alphabets, Applied Soft Computing 11 (2011) 1156–1182.
[43] C. Cruz, D. Pelta, Soft computing and cooperative strategies for optimization, Applied Soft Computing 9 (2009) 30–38.
[44] A. Kamiya, S.J. Ovaska, R. Roy, S. Kobayashi, Fusion of soft computing and hard computing for large-scale plants: a general model, Applied Soft Computing 5 (2005) 265–279.
[45] P. Nagabhushan, S.A. Angadi, B.S. Anami, A soft computing model for mapping incomplete/approximate postal addresses to mail delivery points, Applied Soft Computing 7 (2009) 806–816.
[46] B. Subudhi, A.S. Morris, Soft computing methods applied to the control of a flexible robot manipulator, Applied Soft Computing 9 (2009) 149–158.
[47] J.G. Yang, Y.B. Zhuang, An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem, Applied Soft Computing 10 (2010) 653–660.
[48] R. Venkata Rao, P.J. Pawar, Parameter optimization of a multi-pass milling process using non-traditional optimization algorithms, Applied Soft Computing 10 (2010) 445–456.
[49] A. Yardimci, Soft computing in medicine, Applied Soft Computing 9 (2009) 1029–1043.
[50] V. Oduguwa, R. Roy, D. Farrugia, Development of a soft computing-based framework for engineering design optimisation with quantitative and qualitative search spaces, Applied Soft Computing 7 (2007) 166–188.
[51] X.Z. Gao, S.J. Ovaska, Soft computing methods in motor fault diagnosis, Applied Soft Computing 1 (2001) 73–81.
[52] T.-J. Hsieh, H.-F. Hsiao, W.-C. Yeh, Forecasting stock markets using wavelet transforms and recurrent neural networks: an integrated system based on artificial bee colony algorithm, Applied Soft Computing 11 (2011) 2510–2525.