Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 526315, 9 pages
http://dx.doi.org/10.1155/2013/526315

Research Article

A Simple and Efficient Artificial Bee Colony Algorithm

Yunfeng Xu,[1,2] Ping Fan,[3] and Ling Yuan[1]

[1] *School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*
[2] *Department of Information Security Engineering, Chinese People's Public Security University, Beijing 100038, China*
[3] *School of Computer Science, Hubei University of Science and Technology, Xianning 437100, China*

Correspondence should be addressed to Ping Fan; bjfan208@126.com

Received 7 September 2012; Revised 30 November 2012; Accepted 30 December 2012

Academic Editor: Rui Mu

Artificial bee colony (ABC) is a new population-based stochastic algorithm which has shown good search abilities on many optimization problems. However, the original ABC shows slow convergence speed during the search process. In order to enhance the performance of ABC, this paper proposes a new artificial bee colony (NABC) algorithm, which modifies the search pattern of both employed and onlooker bees. A solution pool is constructed by storing some best solutions of the current swarm. New candidate solutions are generated by searching the neighborhood of solutions randomly chosen from the solution pool. Experiments are conducted on a set of twelve benchmark functions. Simulation results show that our approach is significantly better or at least comparable to the original ABC and seven other stochastic algorithms.

1. Introduction

Optimization problems arise in many application areas such as engineering, economy, and management. Effective and efficient optimization algorithms are always required to tackle increasingly complex real-world optimization problems. In the past several years, some swarm intelligence algorithms, inspired by the social behaviors of birds, fish, or insects, have been proposed to solve optimization problems, such as particle swarm optimization (PSO) [1], ant colony optimization (ACO) [2], artificial bee colony (ABC) [3], and firefly algorithm (FA) [4]. A recent study has shown that ABC performs significantly better or at least comparable to other swarm intelligence algorithms [5].

ABC is a new swarm intelligence algorithm proposed by Karaboga in 2005, which is inspired by the behavior of honey bees [3]. Since the development of ABC, it has been applied to solve different kinds of problems [6]. Similar to other stochastic algorithms, ABC also faces up some challenging problems. For example, ABC shows slow convergence speed during the search process. Due to the special search pattern of bees, a new candidate solution is generated by updating a random dimension vector of its parent solution. Therefore, the offspring (new candidate solution) is similar to its parent, and the convergence speed becomes slow. Moreover, ABC easily falls into local minima when handling complex multimodal problems. The search pattern of bees is good at exploration but poor at exploitation [7]. However, a good optimization algorithm should balance exploration and exploitation during the search process.

To improve the performance of ABC, this paper proposes a new search pattern for both employed and onlooker bees. In the new approach, some best solutions are utilized to accelerate the convergence speed. In addition, a solution pool is constructed by storing the best $100p\%$ solutions in the current swarm with $p \in (0, 1]$. The best solution used in the search pattern is randomly selected from the solution pool. This is helpful to balance the exploration and exploitation. Experiments are conducted on twelve benchmark functions. Simulation results show that our approach outperforms the original ABC and several other stochastic algorithms.

The rest of the paper is organized as follows. In Section 2, the original ABC algorithm is presented. Section 3 gives a brief overview of related work. Section 4 describes the proposed approach. In Section 5, experimental studies are presented. Finally, the work is concluded in Section 6.

2. Artificial Bee Colony

Artificial bee colony (ABC) algorithm is a recently proposed optimization technique which simulates the intelligent foraging behavior of honey bees. A set of honey bees is called swarm which can successfully accomplish tasks through social cooperation. In the ABC algorithm, there are three types of bees: employed bees, onlooker bees, and scout bees. The employed bees search food around the food source in their memory; meanwhile they share the information of these food sources to the onlooker bees. The onlooker bees tend to select good food sources from those found by the employed bees. The food source that has higher quality (fitness) will have a large chance to be selected by the onlooker bees than the one of lower quality. The scout bees are translated from a few employed bees, which abandon their food sources and search new ones [8].

In the ABC algorithm, the first half of the swarm consists of employed bees, and the second half constitutes the onlooker bees. The number of employed bees or the onlooker bees is equal to the number of solutions in the swarm [3].

The ABC generates a randomly distributed initial population of SN solutions (food sources), where SN denotes the swarm size. Let $X_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,D}\}$ represent the ith solution in the swarm, where D is the dimension size. Each employed bee X_i generates a new candidate solution V_i in the neighborhood of its present position as follows:

$$v_{i,j} = x_{i,j} + \phi_{i,j} \cdot \left(x_{i,j} - x_{k,j}\right), \tag{1}$$

where X_k is a randomly selected candidate solution ($i \neq k$), j is a random dimension index selected from the set $\{1, 2, \ldots, D\}$, and $\phi_{i,j}$ is a random number within $[-1, 1]$. Once the new candidate solution V_i is generated, a greedy selection is used. If the fitness value of V_i is better than that of its parent X_i, then update X_i with V_i; otherwise keep X_i unchangeable.

After all employed bees complete the search process, they share the information of their food sources with the onlooker bees through waggle dances. 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. This probabilistic selection is really a roulette wheel selection mechanism which is described as follows:

$$p_i = \frac{\text{fit}_i}{\sum_{j=1}^{\text{SN}} \text{fit}_j}, \tag{2}$$

where fit_i is the fitness value of the ith solution in the swarm. As seen, the better the solution i, the higher the probability of the ith food source selected.

If a position cannot be improved over a predefined number (called *limit*) of cycles, then the food source is abandoned. Assume that the abandoned source is X_i, then the scout bee discovers a new food source to be replaced with X_i as follows:

$$x_{i,j} = lb_j + \text{rand}\,(0, 1) \cdot \left(ub_j - lb_j\right), \tag{3}$$

where rand$(0, 1)$ is a random number within $[0, 1]$ based on a normal distribution and lb, ub are lower and upper boundaries of the jth dimension, respectively.

3. Related Work

Since the development of ABC, it has attracted much attention for its excellent characteristics. In the last decade, different versions of ABCs have been applied to various problems. In this section, we present a brief review of these ABC algorithms.

Karaboga and Akay [5] presented a comparative study of ABC. A large set of benchmark functions are tested in the experiments. Results show that the ABC is better than or similar to those of other population-based algorithms with the advantage of employing fewer control parameters. Inspired by differential evolution (DE) algorithm, Gao and Liu [7, 9] proposed two improved versions of ABC. In [7], a new search pattern called ABC/best/1 is utilized to accelerate the convergence speed. In [9], ABC/best/1 and another search pattern called ABC/rand/1 are employed. Moreover, a parameter q is intruded to control the frequency of these two patterns. Zhu and Kwong [8] utilized the search information of the global best solution (g_{best}) to guide the search of ABC. Reported results show that the new approach achieves better results than the original ABC algorithm. Akay and Karaboga [10] proposed a modified ABC algorithm, in which two new search patterns, frequency and magnitude of the perturbation, are employed to improve the convergence rate. Results show that the original ABC algorithm can efficiently solve basic and simple functions, while the modified ABC algorithm obtains promising results on hybrid and complex functions when compared to some state-of-the-art algorithms. Banharnsakun et al. [11] modified the search pattern of the onlooker bees, in which the best feasible solutions found so far are shared globally among the entire swarm. Therefore, the new candidate solutions are similar to the current best solution. Kang et al. [12] proposed a Rosenbrock ABC (RABC) algorithm which combines Rosenbrock's rotational direction method with the original ABC. There are two alternative phases of RABC: the exploration phase realized by ABC and the exploitation phased completed by the Rosenbrock method. Wu et al. [13] combined harmony search (HS) and the ABC algorithm to construct a hybrid algorithm. Comparison results show that the hybrid algorithm outperforms ABC, HS, and other heuristic algorithms. Li et al. [14] proposed an improved ABC algorithm called I-ABC, in which the best-so-far solution, inertia weight, and acceleration coefficients are introduced to modify the search process. Moreover, a hybrid ABC algorithm (PS-ABC) based on g_{best}-guided ABC (GABC) [8] and I-ABC is proposed. Results show that PS-ABC converges faster than I-ABC and ABC.

Karaboga and Ozturk [15] used ABC algorithm for data clustering. Experiments are conducted on thirteen typical test data sets from UCL Machine Learning Repository. The performance of ABC is compared with PSO and other nine classification techniques. Simulation results demonstrate that the ABC algorithm can efficiently solve data clustering. Zhang et al. [16] also used ABC algorithm for clustering. Three data sets are tested. The performance of ABC is compared with genetic algorithm, simulated annealing, tabu search, ACO, and K-NM-PSO. Results demonstrate the

effectiveness of ABC on clustering. Karaboga and Ozturk [17] applied ABC to solve fuzzy clustering. Three data sets including cancer, diabetes, and heart chosen from UCI database are tested. Results indicate that the performance of ABC is successful in fuzzy clustering.

The ABC algorithm is usually used to solve unconstrained optimization problems. In [18], Karaboga and Akay investigated the performance of ABC on constrained optimization problems. In order to handle constraints, Deb's rules consisting of three simple heuristic rules are employed. Mezura-Montes and Velez-Koeppel [19] proposed an elitist ABC algorithm for constrained real-parameter optimization, in which the operators used by different types of bees are modified. Additionally, a dynamic tolerance control mechanism for equality constraints is utilized to facilitate the approach to the feasible region of the search space. Yeh and Hsieh [20] proposed a penalty-guided ABC algorithm to solve reliability redundancy allocation problems. Sabat et al. [21] presented an application of ABC to extract the small signal equivalent circuit model parameters of GaAS metal-extended semiconductor field effect transistor (MESFT) device. The performance comparison shows that ABC is better than PSO.

It is known that the ABC algorithm is good at solving optimization problems over continuous search space. For discrete optimization problems, it is a big challenge for the ABC algorithm. Li et al. [22] used a hybrid Pareto-based ABC algorithm to solve flexible job shop-scheduling problems. In the new algorithm, each food sources is represented by two vectors, that is, the machine assignment and the operation scheduling. Moreover, an external Pareto archive set is utilized to record nondominated solutions. In [23], Kashan et al. designed a new ABC algorithm called DisABC to optimize binary structured problems. Szeto et al. [24] proposed an enhanced ABC algorithm to solve capacitated vehicle routing problem. The performance of the new approach is tested on two sets of standard benchmark instances. Simulation results show that the new algorithm outperforms the original ABC and several other existing algorithms. Pan et al. [25] presented a discrete ABC algorithm hybridized with a variant of iterated greedy algorithm to solve a permutation flow shop-scheduling problem with the total flow time criterion.

4. Proposed Approach

Differential evolution (DE) has shown excellent search abilities on many optimization problems. Like other population-based stochastic algorithms, DE also starts with an initial population with randomly generated candidate solutions. After initialization, DE repeats three operations: mutation, crossover, and selection. Among these operations, mutation operation is very important. The mutation scheme highly influences the performance of DE. There are several different mutation schemes, such as DE/rand/1, DE/rand/2, DE/best/1, and DE/best2 [26].

The property of a mutation scheme determines the search behavior of individuals in the population. For DE/rand/1, it results in good exploration but slow convergence speed. For DE/best/1, it obtains fast convergence speed but poor exploration. The DE/rand/1 and DE/best/1 are described as follows:

$$\begin{aligned} v_{i,j} &= x_{r1,j} + F \cdot \left(x_{r2,j} - x_{r3,j}\right), \\ v_{i,j} &= x_{\text{best},j} + F \cdot \left(x_{r1,j} - x_{r2,j}\right), \end{aligned} \tag{4}$$

where X_{r1}, X_{r2}, and X_{r3} are three randomly selected individuals from the current population, $i \neq r1 \neq r2 \neq r3$, X_{best} is the best individual found so far, and the parameter F is known as the scale factor which is usually set to 0.5.

As seen, the search pattern of employed and onlooker bees is similar to the mutation schemes of DE. It is known that the ABC algorithm is good at exploration, but it shows slow convergence speed. By combining the DE/best/1 and the ABC algorithm, it may accelerate the convergence speed of ABC. However, this hybridization is not a new idea. In [7], Gao and Liu embedded DE/rand/1 and DE/best/1 into the ABC algorithm. To balance the exploration and exploitation, a new parameter q is introduced. Results reported in [7] show that the parameter q is problem oriented, and an empirical value $q = 0.7$ is used.

In this paper, we propose a new ABC (called NABC) algorithm by employing a modified DE/best/1 strategy. NABC differs from other hybrid algorithms [7, 9], which combine ABC and DE. Although the global best individual used in DE/best/1 can accelerate the convergence speed by the attraction, it may result in attracting too fast. It means that new solutions move to the global best solution very quickly. To tackle this problem, a solution pool is constructed by storing the best $100p\%$ solutions in the current swarm with $p \in (0, 1]$. The idea is inspired by an adaptive DE algorithm (JADE) [27]. It shares in common with the concept of belief space of cultural algorithm (CA) [28]. Both of them utilize some successful solutions stored in solution pool or situational knowledge to guide other individuals. But the updating rule of the solution pool or situational knowledge is different. The new ABC/best/1 strategy is described as follows:

$$v_{i,j} = x^{p}_{\text{best},j} + \phi_{i,j} \cdot \left(x_{r1,j} - x_{r2,j}\right), \tag{5}$$

where $x^{p}_{\text{best},j}$ is randomly chosen from the solution pool, X_{r1}, X_{r2} are two randomly selected candidate solutions from the current swarm, $i \neq r1 \neq r2$, j is a random dimension index selected from the set $\{1, 2, \ldots, D\}$, and $\phi_{i,j}$ is a random number within $[-1, 1]$. Empirical studies show that a good choice of the parameter p should be set between 0.08 and 0.15. In this paper, p is set to 0.1 for all experiments.

According to the new search pattern described in (5), new candidate solutions are generated around some best solutions. This is helpful to accelerate the convergence speed. For the existing ABC/best/1 strategy proposed in [7], it only searches the neighborhood of the global best solution. In our approach, bees can search the neighborhood of different best solutions. This can help avoid fast attraction.

The main steps of our new approach NABC algorithm are listed as follows.

Step 1. Randomly initialize the swarm.

Step 2. Update the solution pool.

Table 1: Benchmark functions used in the experiments.

Name	Function	Range	Opt
Sphere	$f_1 = \sum_{i=1}^{D} x_i^2$	$[-100, 100]$	0
Schwefel 2.22	$f_2 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} x_i$	$[-10, 10]$	0
Schwefel 1.2	$f_3 = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]$	0
Schwefel 2.21	$f_4 = \max_i \left(\lvert x_i \rvert, 1 \le i \le D \right)$	$[-100, 100]$	0
Rosenbrock	$f_5 = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$	$[-30, 30]$	0
Step	$f_6 = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor \right)^2$	$[-100, 100]$	0
Quartic with noise	$f_7 = \sum_{i=1}^{D} i x_i^4 + \text{rand}[0, 1)$	$[-1.28, 1.28]$	0
Schwefel 2.26	$f_8 = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{\lvert x_i \rvert} \right)$	$[-500, 500]$	−12569.5
Rastrigin	$f_9 = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$	0
Ackley	$f_{10} = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right) - \exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + e$	$[-32, 32]$	0
Griewank	$f_{11} = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	$[-600, 600]$	0
Penalized	$f_{12} = \frac{\pi}{D} \{ 10\sin^2(3\pi y_i) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2 \} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	$[-50, 50]$	0

Step 3. For each employed bee, generate a new candidate solution V_i according to (5). Evaluate the fitness of V_i and use a greed selection to choose a better one between X_i and V_i as the new X_i.

Step 4. Each onlooker bee calculates p_i according to (2).

Step 5. Generate a new V_i according to (5) based on p_i and the current solution X_i (food source). Evaluate the fitness of V_i and use a greed selection to choose a better one between X_i and V_i as the new X_i.

Step 6. The scout bee determines the abandoned X_i, if exists and update it by (3).

Step 7. Update the best solution found so far, and cycle = cycle + 1.

Step 8. If the number of cycles reaches to the maximum value MCN, then stop the algorithm and output the results; otherwise go to Step 2.

Compared to the original ABC algorithm, our approach NABC does not add extra operations except for the construction of the solution pool. However, this operation does not add the computational complexity. Both NABC and the original ABC have the same computational complexity.

5. Experimental Study

5.1. Test Functions. In order to verify the performance of NABC, experiments are conducted on a set of twelve benchmark functions. These functions were early used in [29]. According to their properties, they are divided into two classes: unimodal functions ($f_1 - f_7$) and multimodal functions ($f_8 - f_{12}$). All functions are minimization problems, and their dimensional size is 30. Table 1 presents the descriptions of these functions, where Opt is the global optimum.

The experiments are performed on the same computer with Intel (R) Core (TM)2 Duo CPU T6400 (2.00 GHz) and 2 GB RAM. Our algorithm is implemented using C++ and complied with Microsoft Visual C++ 6.0 under the Windows XP (SP3).

5.2. Comparison of NABC with ABC. In order to investigate the effectiveness of our new search pattern, this section presents a comparison of NABC with the original ABC algorithm. In the experiments, both NABC and ABC use the same parameter settings. The population size SN, *limit*, and maximum number of cycles (MSN) are set to 100, 100, and 1000, respectively. The parameter p is set to 0.1 based on empirical studies. All results reported in this section are averaged over 30 independent runs.

Table 2 presents the computational results of ABC and NABC on the twelve functions, where "Mean" indicates the mean function value and "Std Dev" represents the standard deviation. The best results between ABC and NABC are shown in bold. From the results, it can be seen that NABC achieves better results than ABC on all test functions except for f_6. On this function, both the two algorithms can find the global optimum. For f_8 and f_9, NABC can successfully find the global optimum, while ABC converges to near-optimal solutions. It demonstrates that the new search pattern used in NABC is helpful to improve the accuracy of solutions.

In order to compare the convergence speed of NABC and ABC, Figure 1 lists the convergence processes of them on

Table 2: Results achieved by the ABC algorithm and NABC.

Functions	ABC		NABC	
	Mean	Std Dev	Mean	Std Dev
f_1	$3.75e-10$	$2.73e-10$	**4.75e − 16**	**3.86e − 16**
f_2	$2.29e-06$	$4.26e-06$	**1.79e − 15**	**2.53e − 15**
f_3	$1.23e+04$	$2.39e+03$	**9.90e + 03**	**1.67e + 03**
f_4	$3.92e+01$	$1.52e+01$	**1.45e + 01**	**4.32e + 00**
f_5	$2.83e+00$	$1.79e+00$	**4.50e − 02**	**2.38e − 02**
f_6	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**
f_7	$1.91e-01$	$2.33e-01$	**1.56e − 02**	**3.24e − 02**
f_8	−12332.4	$1.57e+02$	**−12569.5**	**1.23e − 10**
f_9	$2.90e-09$	$5.31e-09$	**0.00e + 00**	**0.00e + 00**
f_{10}	$3.93e-06$	$2.78e-06$	**3.97e − 14**	**5.12e − 15**
f_{11}	$4.52e-09$	$3.81e-09$	**1.13e − 16**	**3.39e − 16**
f_{12}	$1.04e-11$	$2.43e-11$	**3.19e − 16**	**3.26e − 16**

some representative functions. As seen, NABC shows faster convergence speed than ABC. It confirms that the new search pattern can accelerate the convergence speed.

5.3. Comparison of NABC with Other Algorithms. To further verify the performance of NABC, this section compares NABC with other population-based algorithms, including some recently proposed ABC algorithms.

5.3.1. Comparison of NABC with Evolution Strategies. This section focuses on the comparison of the NABC algorithm with Evolution Strategies (ES). The versions of the ES include classical evolution strategies (CES) [30], fast evolution strategies (FES) [30], covariance matrix adaptation evolution strategies (CMA-ES) [31], and evolutionary strategies learned with automatic termination (ESLAT) [32].

The parameter settings of CES, FES, CMA-ES, and ESLAT can be found in [32]. For NABC, the population size and the maximum number of fitness evaluations are set to 20 and 100000 (it means that the MSN is 2500), respectively. The parameter *limit* is set to 600 [5]. The parameter p is set to 0.1 based on empirical studies. All algorithms are conducted on 50 runs for each test function.

Table 3 presents the comparison results of CES, FES, CMA-ES, ESLAT, and NABC. Results of CES, FES, CMA-ES, and ESLAT were taken from Table 20 in [5]. Among these algorithms, the best results are shown in bold. The last column of Table 3 reports the statistical significance level of the difference of the means of NABC and the best algorithm among the four evolution strategies. Note that here "+" represents the t value of 49 degrees of freedom which is significant at a 0.05 level of significance by two-tailed test, "·" indicates the difference of means which is not statistically significant, and "NA" means not applicable, covering cases for which the two algorithms achieve the same accuracy results [33].

From the results, it can be seen that NABC outperforms CES and FES on eight functions, while CES and FES achieve better results on three. For f_6, CES, FES, and NABC find the global optimum, while ESLAT and CMA-ES fail to solve it. NABC performs better than ESLAT on ten functions, while ESLAT outperforms NABC for the rest of the two functions. CMA-ES achieves better results than NABC on three functions, while NABC performs better for the rest of the nine functions. The comparison results show that the evolutionary strategies perform better than NABC on unimodal functions, such as $f_1 - f_4$. NABC outperforms the evolutionary strategies on all multimodal functions ($f_8 - f_{12}$).

5.3.2. Comparison of NABC with Other Improved ABC Algorithms. In this section, we present a comparison of NABC with three recently proposed ABC algorithms. The involved algorithms are listed as follows.

(1) g_{best}-guided ABC algorithm (GABC) [8].

(2) Improved ABC algorithm (I-ABC) [14].

(3) Hybridization of GABC and I-ABC (PS-ABC) [14].

(4) Our approach NABC.

In the experiments, the population size SN is set to 40, and *limit* equals 200. The maximum number of cycles is set as 1000. Other parameter settings of GABC, I-ABC, and PS-ABC can be found in [14]. The parameter p used in NABC is set to 0.1 based on empirical studies. All algorithms are conducted 30 times for each test function, and the mean function values are reported.

Table 4 presents the comparison results of NABC with three other ABC algorithms. Results of GABC, I-ABC and PS-ABC were taken from Tables 4 and 5 in [14]. The best results among the four algorithms are shown in bold. From the results, NABC outperforms GABC on all test functions except for f_2. On this function, GABC is slightly better than NABC. I-ABC achieves better results than NABC on five functions, while NABC performs better on six functions. For the rest of f_9, I-ABC, PS-ABC, and NABC can find the global optimum. PS-ABC obtains better results than NABC on six functions, while NABC outperforms PS-ABC on five functions. Both I-ABC and PS-ABC achieve significantly better results on three unimodal functions, such as f_1, f_2, and

Table 3: Comparison of NABC with evolution strategies.

Functions	CES	FES	ESLAT	CMA-ES	NABC	Significance
	Mean	Mean	Mean	Mean	Mean	
f_1	**1.70e − 26**	2.50e − 04	2.00e − 17	9.70e − 23	2.88e − 16	.
f_2	**8.10e − 20**	6.00e − 02	3.80e − 05	4.20e − 11	1.37e − 15	.
f_3	3.38e + 02	1.40e − 03	6.10e − 06	**7.10e − 23**	6.86e + 03	.
f_4	2.41e + 00	5.50e − 03	7.80e − 01	**5.40e − 12**	4.30e − 01	.
f_5	2.77e + 01	3.33e + 01	1.93e + 00	4.00e − 01	**2.62e − 01**	+
f_6	**0.00e + 00**	**0.00e + 00**	2.00e − 02	1.44e + 00	**0.00e + 00**	NA
f_7	4.70e − 02	**1.20e − 02**	3.90e − 01	2.30e − 01	1.38e − 02	+
f_8	−8000	−12556.4	−2300	−7637.1	**−12569.5**	+
f_9	1.34e + 01	1.60e − 01	4.65e + 00	5.18e + 01	**0.00e + 00**	+
f_{10}	6.00e − 13	1.20e − 02	1.80e − 08	6.90e − 12	**3.25e − 14**	+
f_{11}	6.00e − 14	3.70e − 02	1.40e − 03	7.40e − 04	**1.11e − 16**	+
f_{12}	1.46e + 00	2.80e − 06	1.50e − 12	1.20e − 04	**2.52e − 16**	+

Table 4: Comparison of NABC with other ABC algorithms.

Functions	GABC	I-ABC	PS-ABC	NABC	Significance
	Mean	Mean	Mean	Mean	
f_1	6.26e − 16	**0.00e + 00**	**0.00e + 00**	5.43e − 16	.
f_2	9.36e − 16	**0.00e + 00**	**0.00e + 00**	6.24e − 15	.
f_3	1.09e + 04	1.43e + 04	**6.11e + 03**	8.44e + 03	.
f_4	1.26e + 01	1.27e − 197	**0.00e + 00**	5.79e + 00	.
f_5	7.48e + 00	2.64e + 01	1.59e + 00	**1.45e − 01**	+
f_6	2.49e − 09	3.84e − 10	5.72e − 16	**0.00e + 00**	+
f_7	1.56e − 01	1.96e − 02	2.15e − 02	**1.72e − 02**	+
f_8	−12407.3	−12251.03	−12564.2	**−12569.5**	+
f_9	3.31e − 02	**0.00e + 00**	**0.00e + 00**	**0.00e + 00**	NA
f_{10}	7.78e − 10	**8.88e − 16**	**8.88e − 16**	1.07e − 13	.
f_{11}	6.96e − 04	**0.00e + 00**	**0.00e + 00**	1.11e − 16	.
f_{12}	5.85e − 16	7.11e − 12	5.53e − 16	**4.67e − 16**	+

f_4. On these functions, they can find the global optimum, while GABC and NABC only find near-optimal solutions except for f_4. For f_4, both GABC and NABC fall into local minima. I-ABC and PS-ABC successfully find the global optimum on f_{11}, while GABC and NABC fail. For function f_{10}, I-ABC and PS-ABC are slightly better than NABC. For other two multimodal functions f_8 and f_{12}, NABC performs better than other three ABC algorithms. Compared to I-ABC and PS-ABC, our approach NABC is simpler and easier to implement.

6. Conclusions

Artificial bee colony is a new optimization technique which has shown to be competitive to other population-based stochastic algorithms. However, ABC and other stochastic algorithms suffer from the same problems. For example, the convergence speed of ABC is typically slower than PSO and DE. Moreover, the ABC algorithm easily gets stuck when handling complex multimodal problems. The main reason is that the search pattern of both employed and onlooker bees is good at exploration but poor at exploitation. In order to balance the exploration and exploitation of ABC, this paper proposes a new ABC variant (NABC). It is known that DE/best/1 mutation scheme is good at exploitation. Based on DE/best/1, a new search pattern called ABC/best/1 with solution pool is proposed. Our approach differs from other improved ABC algorithms by hybridization of DE/best/1 and ABC.

To verify the performance of our approach, a set of twelve benchmark functions are used in the experiments. Comparison of NABC with ABC demonstrates that our new search pattern can effectively accelerate the convergence speed and improve the accuracy of solutions. Another comparison demonstrates that NABC is significantly better or at least comparable to other stochastic algorithms. Compared to other improved ABC algorithms, our approach is simpler and easier to implement.

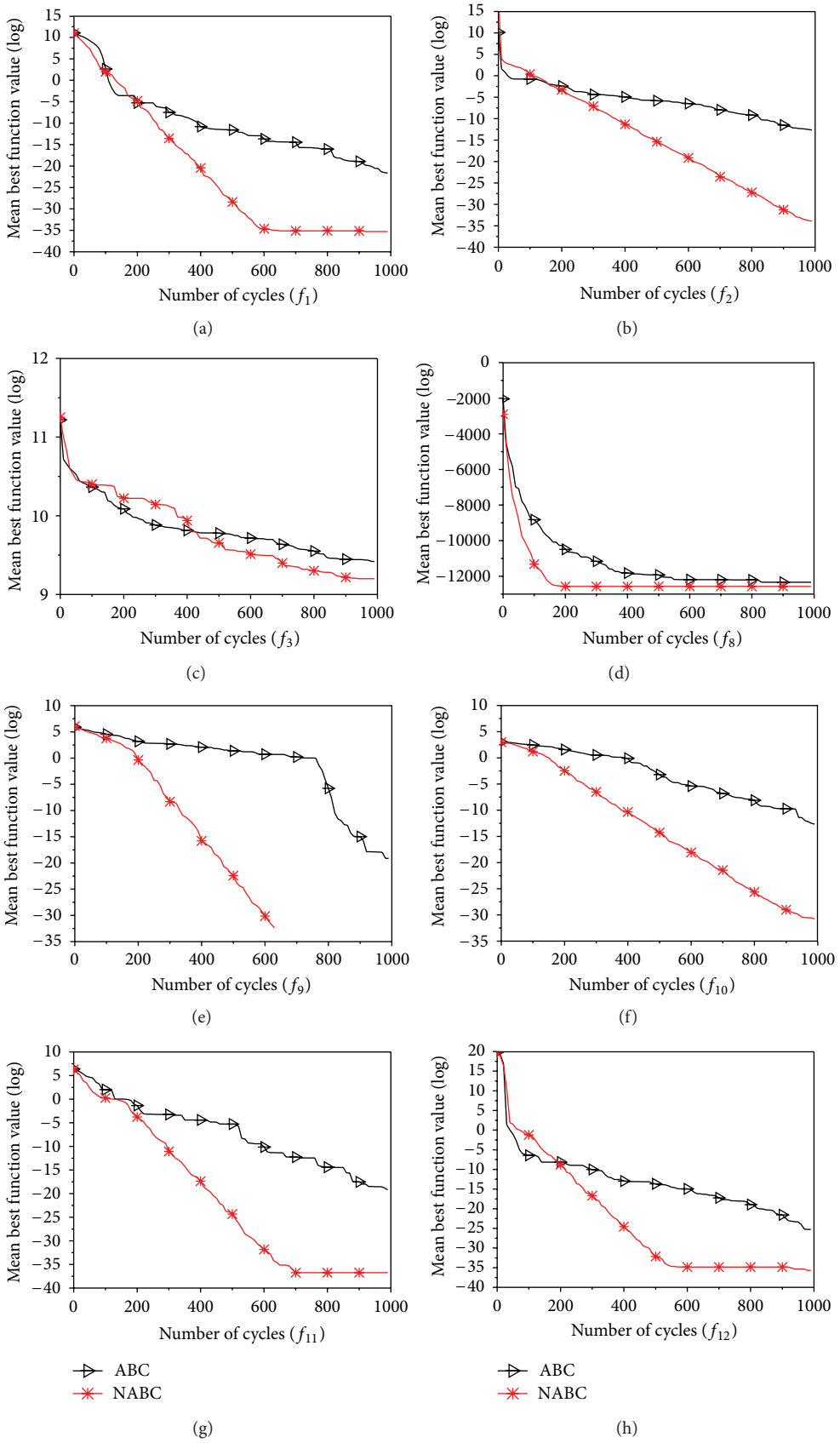

Figure 1: The convergence processes of ABC and NABC on some functions.

Acknowledgments

This work was supported by the National Natural Science Fund of Hubei Province under Grant 2012FFB00901, the Science and Technology Research Project of Xianning City under Grant XNKJ-1203, Doctoral Start Fund of Hubei University of Science and Technology under Grant BK1204, the Teaching Research Project of Hubei University of Science and Technology under Grant 2012X016B, Application Innovation Project of the Ministry of Public Security under Grant 2005yycxhbst117, The Key Research Project of Science and Technology of Hubei Province under Grant 2007AA301C33, Key Lab of Information Network Security of Ministry of Public Security under Grant C09602, and Application Innovative Project of Public Security of Hubei Province under Grant hbst2009sjkycx014.

References

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

[2] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 26, no. 1, pp. 29–41, 1996.

[3] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[4] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimization," *International Journal of Bio-Inspired Computing*, vol. 2, no. 2, pp. 78–84, 2010.

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

[6] D. Karaboga and B. Akay, "A survey: algorithms simulating bee swarm intelligence," *Artificial Intelligence Review*, vol. 31, no. 1–4, pp. 61–85, 2009.

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

[8] 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.

[9] W. Gao and S. Liu, "Improved artificial bee colony algorithm for global optimization," *Information Processing Letters*, vol. 111, no. 17, pp. 871–882, 2011.

[10] B. Akay and D. Karaboga, "A modified Artificial Bee Colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, pp. 120–142, 2012.

[11] A. Banharnsakun, T. Achalakul, and B. Sirinaovakul, "The best-so-far selection in Artificial Bee Colony algorithm," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 2888–2901, 2011.

[12] F. Kang, J. Li, and Z. Ma, "Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions," *Information Sciences*, vol. 181, no. 16, pp. 3508–3531, 2011.

[13] B. Wu, C. Qian, W. Ni, and S. Fan, "Hybrid harmony search and artificial bee colony algorithm for global optimization problems," *Computers & Mathematics with Applications*, vol. 64, no. 8, pp. 2621–2634, 2012.

[14] G. Li, P. Niu, and X. Xiao, "Development and investigation of efficient artificial bee colony algorithm for numerical function optimization," *Applied Soft Computing*, vol. 12, no. 1, pp. 320–332, 2012.

[15] D. Karaboga and C. Ozturk, "A novel clustering approach: Artificial Bee Colony (ABC) algorithm," *Applied Soft Computing Journal*, vol. 11, no. 1, pp. 652–657, 2011.

[16] C. Zhang, D. Ouyang, and J. Ning, "An artificial bee colony approach for clustering," *Expert Systems with Applications*, vol. 37, no. 7, pp. 4761–4767, 2010.

[17] D. Karaboga and C. Ozturk, "Fuzzy clustering with artificial bee colony algorithm," *Scientific Research and Essays*, vol. 5, no. 14, pp. 1899–1902, 2010.

[18] D. Karaboga and B. Akay, "A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems," *Applied Soft Computing Journal*, vol. 11, no. 3, pp. 3021–3031, 2011.

[19] E. Mezura-Montes and R. E. Velez-Koeppel, "Elitist artificial bee colony for constrained real-parameter optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1–8, 2010.

[20] W.-C. Yeh and T.-J. Hsieh, "Solving reliability redundancy allocation problems using an artificial bee colony algorithm," *Computers & Operations Research*, vol. 38, no. 11, pp. 1465–1473, 2011.

[21] S. L. Sabat, S. K. Udgata, and A. Abraham, "Artificial bee colony algorithm for small signal model parameter extraction of MESFET," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 5, pp. 689–694, 2010.

[22] J. Q. Li, Q. K. Pan, S. X. Xie, and S. Wang, "A hybrid artificial bee colony algorithm for flexible job shop scheduling problems," *International Journal of Computers, Communications and Control*, vol. 6, no. 2, pp. 286–296, 2011.

[23] M. H. Kashan, N. Nahavandi, and A. H. Kashan, "DisABC: a new artificial bee colony algorithm for binary optimization," *Applied Soft Computing*, vol. 12, no. 1, pp. 342–352, 2012.

[24] W. Y. Szeto, Y. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 215, no. 1, pp. 126–135, 2011.

[25] Q.-K. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Information Sciences*, vol. 181, no. 12, pp. 2455–2468, 2011.

[26] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

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

[28] R. G. Reynolds, "An introduction to cultural algorithms," in *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pp. 131–139, 1994.

[29] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[30] X. Yao and Y. Liu, "Fast evolution strategies," *Control and Cybernetics*, vol. 26, no. 3, pp. 467–496, 1997.

[31] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '96)*, pp. 312–317, May 1996.

[32] A. Hedar and M. Fukushima, "Evolution strategies learned with automatic termination criteria," in *Proceedings of the Conference on Soft Computing and Intelligent Systems and the International Symposium on Advanced Intelligent Systems*, pp. 1–9, Tokyo, Japan, 2006.

[33] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.