Information Sciences 279 (2014) 587–603

Contents lists available at ScienceDirect

Information Sciences

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

Multi-strategy ensemble artificial bee colony algorithm

Hui Wang [a,*], Zhijian Wu [b], Shahryar Rahnamayan [c], Hui Sun [a], Yong Liu [d], Jeng-shyang Pan [a,e,f]

[a] School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, PR China
[b] State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, PR China
[c] Department of Electrical, Computer, and Software Engineering, University of Ontario Institute of Technology (UOIT), 2000 Simcoe Street North, Oshawa, ON L1H 7K4, Canada
[d] University of Aizu, Tsuruga, Ikki-machi, Aizu-Wakamatsu, Fukushima 965-8580, Japan
[e] Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, PR China
[f] Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung 807, Taiwan

ARTICLE INFO

Article history:
Received 21 January 2013
Received in revised form 16 March 2014
Accepted 8 April 2014
Available online 19 April 2014

Keywords:
Artificial bee colony (ABC)
Multi-strategy
Ensemble
Global optimization

ABSTRACT

Artificial bee colony (ABC) is a recently proposed optimization technique which has shown to be competitive to other population-based stochastic algorithms. However, ABC is good at exploration but poor at exploitation because of its solution search strategy. Thus, to obtain an efficient performance, utilizing different characteristics of solution search strategies can be appropriate during different stages of the search process to achieve a tradeoff between exploration and exploitation. In this paper, we propose a novel multi-strategy ensemble ABC (MEABC) algorithm. In MEABC, a pool of distinct solution search strategies coexists throughout the search process and competes to produce offspring. Experiments are conducted on a set of commonly used numerical benchmark functions, including the CEC 2013 shifted and rotated problems. Results show that MEABC performs significantly better than, or at least comparable to, some well-established evolutionary algorithms.

1. Introduction

Optimization problems exist in various engineering and science areas, such as structural design, scheduling, portfolio investment, and economic dispatch. As the complexity of problems increases, traditional optimization algorithms may not satisfy the problem requirements and more effective algorithms are needed. In the past decades, some swarm intelligence algorithms, inspired by the social behaviors of birds, fish or insects, have been proposed to solve NP-complete optimization problems [8], such as particle swarm optimization (PSO) [29], ant colony optimization (ACO) [10], artificial bee colony (ABC) [22], cat swarm optimization (CSO) [5], and firefly algorithm (FA) [64]. A recent study has shown that ABC performs significantly better, or at least comparable to other swarm intelligence algorithms [23]. Due to ABC's simple concept, easy implementation yet effectiveness, it has become popular in evolutionary optimization community.

ABC is a new optimization algorithm developed by Karaboga in 2005 [22], which simulates the foraging behavior of honey bees. Although ABC has shown a good performance over many optimization problems, it converges slowly, especially at the middle and last stages of the search process. The main reason is that ABC is good at exploration but poor at exploitation [71]. An ideal optimization algorithm should properly balance exploration and exploitation during the search process [14].

* Corresponding author. Tel.: +86 0791 88126661; fax: +86 0791 88126660.
E-mail addresses: huiwang@whu.edu.cn (H. Wang), zhijianwu@whu.edu.cn (Z. Wu), shahryar.rahnamayan@uoit.ca (S. Rahnamayan), sun_hui2006@163.com (H. Sun), yliu@u-aizu.ac.jp (Y. Liu), jspan@cc.kuas.edu.tw (J.-s. Pan).

http://dx.doi.org/10.1016/j.ins.2014.04.013
0020-0255/

Initially, the algorithm should concentrate on exploration; as iteration increases, it would be better to exploit to find more accurate solutions. However, it is difficult to determine when the algorithm should switch from an explorative behavior to an exploitative behavior.

In order to balance the exploration and exploitation of ABC during the search process, this paper proposes a multi-strategy ensemble ABC (MEABC) algorithm. In MEABC, different characteristics of solution search strategies are employed to construct a strategy pool. Initially, each food source (solution) is randomly assigned a solution search strategy from the strategy pool. When searching a food source, each bee generates offspring according to the assigned strategy of the food source. During the search process, the strategy for each food source is dynamically changed according to the quality of new generated candidate solutions. Thus, bees use different search strategies to find new food sources (solutions) and utilize variant search characteristics. The population may consist of explorative bees as well as exploitative bees simultaneously. Experimental studies are conducted on a set of benchmark functions. Simulation results demonstrate the efficiency and effectiveness of the proposed approach.

The remainder of the paper is organized as follows. Section 2 briefly introduces the background and related works. In Section 3, we describe the multi-strategy ensemble approach. Experimental results and discussions are presented in Section 4. Finally, the work is concluded and summarized in Section 5.

2. Background review and related work

2.1. Artificial bee colony algorithm

The ABC algorithm is a population-based stochastic algorithm that starts with an initial population of randomly generated bees. The bees are categorized into three groups: employed bees, onlooker bees, and scout bees. The employed bees search their food sources and share the information about these food sources to recruit the onlooker bees. The onlooker bees make a decision to choose a food source from those found by the employed bees, and then further search the food around the selected food source. The food source that has more nectar amount (fitness value) will has a higher probability to be selected by the onlooker bees than one of less nectar. The scout bees are translated from a few employed bees, which abandon their food sources and randomly search new ones [71].

For a search problem in a D-dimensional space, the position of a food source represents a potential solution. The nectar amount of a food source is the fitness value of the associate solution. Each food source is exploited by only one employed bee. The number of employed bees or the onlooker bees is equal to the number of solutions in the population.

Let $X_i = x_{i,1}, x_{i,2}, \ldots, x_{i,D}$ be the ith food source (solution) in the population, where D is the problem dimension size. Each employed bee generates a new food source V_i around the neighborhood of its parent position as follows:

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

where $i = 1, 2, \ldots, SN, SN$ is the population size, X_k is a randomly selected solution in the current population $(k \neq i), j \in \{1, 2, \ldots, D\}$ is a random index, and $\phi_{i,j}$ is a uniformly distributed random number in the range $[-1, 1]$. If the new food source V_i is better than its parent X_i, then V_i replaces X_i.

After all employed bees complete their searches according to Eq. (1), they share their information (nectar amounts and positions of food sources) with the onlooker bees. An onlooker bee chooses a food source depending on the probability p_i related to its nectar amount (fitness value).

$$p_i = \frac{f_i}{\sum_{i=1}^{SN} f_i}, \tag{2}$$

where f_i is the fitness value of the ith solution in the population. As seen, the probability p_i is proportional to the fitness value. The better a food source is, the higher chance to be selected.

If a food source cannot be improved further over a predefined number of cycles, the food source is considered to be abandoned. The value of the predefined number of cycles is another control parameter, called *limit*. Assume that the abandoned source is X_i, then the scout bee randomly searches a new food source to be replaced with X_i. This operation is defined as follows:

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

where $rand(0,1)$ is a uniformly distributed random number in the range $[0, 1]$, and $\left[x_j^{min}, x_j^{max}\right]$ is the boundary constraint for the jth variable.

2.2. ABC variants

Since ABC was introduced, it has become a popular optimizer and has widely been applied in practical problems. In the past several years, many variants of ABC have been proposed. A brief overview of these variants is presented as follows.

In order to investigate the performance of ABC, Karaboga and Akay [23] presented a comparative study on a large set of numerical test functions. Results show that the performance of ABC is better than or similar to other population-based stochastic algorithms, such as genetic algorithm (GA), PSO, differential evolution (DE), and evolution strategies (ES). As mentioned before, ABC is good at exploration but poor at exploitation. To tackle this problem, some improved strategies have been proposed. In [2], a chaotic ABC (CABC) algorithm is proposed by employing chaotic maps for parameter adaptation in order to improve the convergence characteristics and prevent the ABC to fall into local minima. Kang et al. [21] proposed a Rosenbrock ABC (RABC) algorithm by combining Rosenbrock's rotational direction method with the original ABC. In RABC, there are two alternative models, including the exploration model realized by ABC and the exploitation model completed by the rotational direction method. Numerical results show that RABC is promising in terms of convergence speed, success rate, and accuracy.

Inspired by PSO, Zhu and Kwong [71] proposed an improved ABC algorithm called *gbest*-guided ABC (GABC) algorithm, which modified the solution search model by incorporating the information of the global best (*gbest*) solution to improve the exploitation. Experimental results show that GABC outperforms the original ABC on most of test functions. Unlike GABC, Gao et al. [18] proposed another *gbest* ABC (ABC/best/1) inspired by DE. In the new algorithm, each bee searches only around the best solution of the previous iteration to enhance the exploitation. Moreover, chaotic systems and opposition-based learning (OBL) are employed for population initialization and scout bees. Computational results show that the new algorithm performs better than the original ABC and GABC. In [17], Gao and Liu proposed a modified ABC algorithm (MABC). In MABC, a new solution search model, called ABC/best/1, based on the DE/best/1 mutation scheme is proposed. Recently, Gao and Liu [16] introduced an enhanced MABC algorithm called IABC, in which two solution models, including ABC/best/1 and ABC/rand/1, are employed based on the DE mutation schemes. A parameter p is introduced to control the frequency of conducting ABC/best/1 and ABC/rand/1. Akay and Karaboga [1] modified the original ABC algorithm by employing two new search models, including frequency and magnitude of the perturbation. The original ABC modified only one dimension for producing a new solution, while the modified ABC introduced a control parameter that determined how many dimensions to be modified. Results show that the original ABC can effectively solve basic functions, while the modified ABC achieves promising solutions on hybrid complex functions. Rajasekhar et al. [46,47] proposed two versions of ABC with Sobol and Levy mutation. Experimental results show the superiority of the mutation operations, especially on high dimensional problems. In [30], Li et al. proposed an improved ABC algorithm called I-ABC, in which the best-so-far solution, inertia weight and acceleration coefficients are utilized to enhance the search process. Results show that I-ABC performs better than original ABC and GABC. El-Abd [12] embedded generalized opposition-based learning (GOBL) [61] into the original ABC. Experimental results on the CEC 2005 benchmark functions show the good performance of the new approach.

In [26], Karaboga and Akay firstly developed a new ABC method for symbolic regression. Simulation results show that the new algorithm is very feasible and robust on the considered test problems of symbolic regression. Sabat et al. [49] presented an application of ABC to extract the small signal equivalent circuit model parameter of metal extend semiconductor field effect transistor (MESFET). Simulation results demonstrate the effectiveness of this approach. In [59], an interactive ABC algorithm was applied to passive continuous authentication system.

The ABC algorithm was firstly proposed for unconstrained optimization problems. Recently, it was used to solve constrained optimization problems. In [25], Karaboga and Akay modified the ABC algorithm by employing Deb's rules for handling constraints. Sonmez [52] introduced an adaptive penalty function for ABC to solve truss structure optimization problems. Results show that the new approach is a powerful search and optimization technique for structural design. Mezura-Montes and Velez-Koeppel [39] proposed an elitist ABC to solve constrained real-parameter optimization problems. To facilitate the production of feasible solutions, a dynamic tolerance for equality constraints was employed. Moreover, two simple local search operators were applied to the best solution at certain times to obtain good solutions. Yeh and Hsieh [66] proposed a penalty guided ABC algorithm to solve the nonlinearly mixed-integer reliability design problems. Simulation results show that the best solutions achieved by the new ABC algorithm is better than the well-known best solutions found by other heuristic methods. Manoj and Elias [37] applied the ABC algorithm to design a multiplier-less nonuniform filter bank transmultiplexer. In [67], a new hybrid ABC (HABC) based on Taguchi method was proposed to solve the manufacturing optimization problems.

For discrete optimization problems, some improved ABC algorithms have been designed. Singh [51] proposed a new ABC algorithm for the leaf-constrained minimum spanning tree (LCMST) problem. Computational results demonstrate the superiority of the new ABC approach over all the other approaches. The new approach obtained better quality solutions in shorter time. In [53], Sundar and Singh applied ABC to solve the quadratic minimum spanning tree (Q-MST) problem. Pan et al. [42] proposed a discrete ABC (DABC) algorithm to solve the lot-streaming flow shop scheduling problem. Unlike the original ABC, DABC represented a food source as a discrete job permutation, and employed some discrete operators to generate new food sources. In [57], Tasgetiren et al. presented another discrete ABC algorithm hybridized with a variant of iterated greedy algorithms for minimizing the total flow-time in permutation flow shops. Karaboga and Gorkemli [27] proposed a combinatorial ABC to solve traveling salesman problem (TSP). Simulation results show that the new approach achieves reasonable solutions. Szeto et al. [56] proposed an enhanced ABC algorithm to solve a capacitated vehicle routing problem. Simulation results show that the enhanced ABC algorithm is able to produce much better solutions than the standard one. Sundar and Singh [54] combined ABC with a local search strategy to solve the non-unicost set covering problem (SCP). Computational results show that the hybrid algorithm is competitive in terms of solution quality with other meta-heuristic approaches

for the SCP problem. Kashan et al. [28] introduced a new version of ABC to solve binary optimization problems. The new approach uses a new differential expression which employs a measure of dissimilarity to respond the structure of binary problems. Sundar and Singh [55] applied ABC to solve the dominating tree problem (DTP). Simulation results show that ABC and ACO are comparable in terms of solution quality for the DTP problem.

Omkar et al. [41] proposed a vector evaluated ABC (VEABC) algorithm for multi-objective design optimization of composite structures. Results show that the VEABC based optimization model has performed quite satisfactorily in comparison with other nature inspired techniques, such as PSO, GA, and artificial immune system (AIS). Samanta and Chakraborty [50] applied ABC algorithm to solve multi-objective non-traditional machining (NTM) processes, in which a weighted method is used to transform multi-objective optimization into single objective optimization.

In this section, we only presented a brief overview of some recently proposed ABC variants; a comprehensive survey can be found in [24].

3. Multi-strategy ensemble ABC (MEABC) algorithm

The solution search strategy (see Eq. (1)) plays an important role in determining the performance of the original ABC algorithm. Generally, different optimization problems require different search strategies depending on the properties of problems. It has been pointed out in [25] that the original ABC performs effectively on basic functions, while ABC with frequency or magnitude perturbation (two improved solution search strategies) works well on hybrid complex problems. To solve a specific problem, different solution search strategies may be better during different stages of the evolution than a single search strategy as in the original ABC. Ensemble learning has proven to be very efficient and effective for adjusting different strategies or control parameters in an online manner [11,34–36,58,68,70].

In [11], Du and Li proposed a multi-strategy ensemble PSO (MEPSO) for dynamic optimization. In MEPSO, all particles are divided into two parts, denoted as part I and part II, respectively. Gaussian local search and differential mutation are introduced into these two parts, respectively. Experimental analyses show that the mechanisms used in part I can enhance the convergence ability, while mechanisms used in part II can improve the ability of catching up with the changing optimum in dynamic environments. Mallipeddi et al. [34] introduced ensemble strategies for evolutionary programming (EP) where each mutation operator has its associated population and every population benefits from every function call. This approach benefits from different mutation operators with different parameter values whenever they are effective during different stages of the search process. Yu et al. [68] proposed an ensemble of niching algorithm (ENA) which is realized using four different parallel populations. The offspring of each population is considered by all parallel populations. Results show that the competitiveness of the ENA method. The multi-objective evolutionary algorithm based on decomposition (MOEA/D) has shown a good performance, but the neighborhood size (NS) parameter is important to its performance. In [70], Zhao et al. presented an ensemble of different NSs with online self-adaptation. Results show that the ensemble method achieved superior performance than the original MOEA/D.

Motivated by these observations, this paper proposes an ensemble of multiple solution search strategies for ABC (MEABC), in which a pool of distinct solution search strategies coexists throughout the search process and competes to produce better offspring. To implement MEABC, we need to address two crucial questions: First, which solution search strategy should be chosen to construct the strategy pool? Second, how do we assign solution search strategies to food sources?

The solution search strategies in the pool should have diverse characteristics, so then they can exhibit distinct performance characteristics during different stages of the evolution. At present, there are several solution search strategies, such as the strategy of the original ABC, *gbest*-guided ABC [71], ABC/best/1 [17], ABC with frequency or magnitude perturbation [25], and chaotic ABC [2]. In this paper, three different solution search strategies are employed to construct the strategy pool: (1) the original ABC; (2) *gbest*-guided ABC; and (3) modified ABC/best/1.

According to the solution search strategy of the original ABC algorithm described in Eq. (1), the new candidate solution is generated by moving the old solution towards another solution selected randomly from the population. In fact, according to the probability theory, 50% of the time the randomly selected solution is a bad one. So, the new candidate solution is not promising to be a solution better than its parent. Inspired by PSO, Zhu and Kwong [71] proposed a *gbest*-guided ABC (GABC) algorithm, in which the information of the *gbest* is used to guide the search of candidate solutions. The solution search strategy in GABC is described as follows:

$$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}) + \psi_{i,j}(gbest_j - x_{i,j}), \tag{4}$$

where $gbest_j$ is the jth element of the global best solution, X_k is a randomly selected solution in the current population ($k \neq i$), $j \in \{1, 2, \ldots, D\}$ is a random index, $\psi_{i,j}$ is a uniform random number in the range $[0, C]$, and C is a non-negative constant. In [71], $C = 1.5$ is regarded as the best setting.

In [17], Gao and Liu also utilized the search information of the *gbest* to direct the movement of the current population and proposed a modified ABC (MABC) algorithm. Unlike GABC, MABC is inspired by the DE/best/1 mutation scheme. The new solution search strategy, called ABC/best/1, is given as follows:

$$v_{i,j} = x_{best,j} + \phi_{i,j}(x_{r1,j} - x_{r2,j}), \tag{5}$$

where $r1$ and $r2$ are two random integer numbers selected from $\{1,2,\ldots,SN\}, r1 \neq r2 \neq i$, and $j \in \{1,2,\ldots,D\}$ is a random index.

In our initial idea, the solution search strategies of the original ABC, GABC, and MABC are utilized to construct the strategy pool. It is expected that the original ABC shows the best exploration ability but the slowest convergence speed. MABC has the fastest convergence rate (good exploitation), and GABC provided a middle phase between the original ABC and MABC. By the ensemble of these strategies, MEABC algorithm can behave different search characteristics during different stages of the evolution. However, experimental results show that GABC converges faster than MABC. So, the initial design is not consistent with our expectations. To tackle this problem, we modified the solution search strategy of MABC (ABC/best/1) as follows:

$$v_{i,j} = x_{best,j} + \phi_{i,j}(x_{best,j} - x_{k,j}). \tag{6}$$

Based on the above analysis, the strategy pool consists of the solution search strategies of the original ABC, GABC, and modified ABC/best/1. Let SP be the strategy pool, and each strategy SP_j in the pool is defined by

$$SP_j = \begin{cases} \text{The original ABC} \\ \text{GABC} \\ \text{Modified ABC/best/1} \end{cases}, \tag{7}$$

where SP_j is the jth strategy in the strategy pool, and $j = 1,2,3$.

In order to address the second question, we employed a new encoding method as described in Fig. 1, where X_i is the position vector of the ith food source (solution), and S_i indicates the employed solution search strategy for the ith solution. The idea of the encoding method is inspired by [34,44]. Initially, each food source (solution) is randomly assigned a solution search strategy ($S_i = SP_j$, and $j \in \{1,2,3\}$ is a random integer index). During the search process, the value of S_i is changed according to the quality of the new candidate solution V_i. If the new solution V_i is better then its parent X_i, it means that the current strategy is suitable for the search. Keeping the current S_i may continue to obtain better solutions. If the new solution V_i is worse than its parent, it indicates that the current strategy does not work for improving the quality of solutions. So, changing the current strategy S_i may be more suitable for the follow-up search. Based on the above analysis, the method of updating S_i for each food source (solution) is illustrated in Algorithm 1.

Algorithm 1. Dynamic Adjustment for Solution Search Strategy

```
1 if the new candidate solution V_i is better than X_i then
2    X_i = V_i;
3    S_i = S_i;
4 end
5 else
6    Randomly select a strategy SP_h from the strategy pool SP, and SP_h ≠ S_i;
7    X_i = X_i;
8    S_i = SP_h;
9 end
```

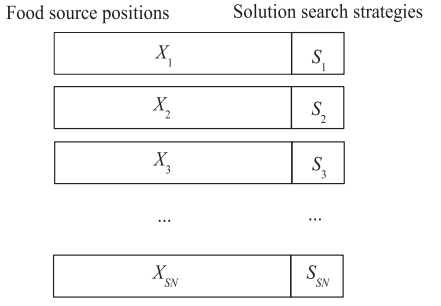


Fig. 1. The encoding method of food source in MEABC algorithm, where X_i is the position vector of the ith food source, and S_i indicates the employed solution search strategy for the ith food source.

Algorithm 2. The Proposed MEABC Algorithm

```
Randomly generate SN solutions (positions of food sources) as an initial population {X_i|i = 1,2,...,SN};
Randomly initialize S_i for each solution;
Calculate the function value f(X_i) of each solution in the population;
Initialize gbest;
while FEs ⩽ MAX_FEs do
  for i = 1 to SN do
    if S_i == 'ABC' then
      Randomly choose j from {1,2,...,D} and produce a random number φ_ij ∈ [−1,1];
      Randomly select a solution X_k from the current population, and k ≠ i;
      Generate a new candidate solution V_i according to Eq. (1);
    end
    if S_i == 'GABC' then
      Randomly choose j from {1,2,...,D} and produce a random number φ_ij ∈ [−1,1];
      Randomly select a solution X_k from the current population, and k ≠ i;
      Generate a new candidate solution V_i according to Eq. (4) (C = 1.5);
    end
    if S_i == 'Modified ABC/best/1' then
      Randomly choose j from {1,2,...,D} and produce a random number φ_ij ∈ [−1,1];
      Generate a new candidate solution V_i according to Eq. (6);
    end
    Calculate the function value of V_i;
    FEs++;
    Update S_i according to Algorithm 1 (Dynamic Adjustment for Solution Search Strategy);
  end
  Update gbest;
end
```

The main steps of MEABC are described in Algorithm 2, where FEs is the number of function evaluations, and MAX_FEs is the maximum number of function evaluations. By the suggestions of [3], the objective function value is directly used for solution comparison. Compared to the original ABC, MEABC eliminates the parameter *limit*, but it contains a new parameter C employed in GABC solution search strategy. Both MEABC and MABC also eliminates the parameter *limit*, but MABC introduced a new parameter p to control the frequency of conducting the original ABC algorithm, where $p \in [0,1]$. Results reported that the parameter p plays an important role in balancing the exploration and exploitation. $p = 0$ obtained faster convergence for Sphere and Ackley functions, while $p = 0.7$ achieved better results for the rest four functions. Besides SN and *limit*, GABC introduced another parameter C, which was set to 1.5 based on empirical studies.

To solve a specific problem f, assume that $O(f)$ is the computational time complexity of evaluating its function value. The maximum number of generations is set to G_{max}. For the original ABC, its computational time complexity is $O[G_{max} \cdot (SN \cdot f + SN \cdot f)] = O(G_{max} \cdot SN \cdot f)$. The GABC only modified the solution search strategy of the original ABC. So, the computational time complexity of GABC is the same with the original ABC. For MABC, it introduced a parameter $p \in [0,1]$ to decide the frequency of conducting the original ABC search, and eliminated the parameter *limit*. Therefore, the

Table 1
Benchmark functions used in the experiments, where $f(x^o)$ is the global optimum.

Functions	Name	Search range	$f(x^o)$
f_1	Sphere	$[-100,100]$	0
f_2	Schwefel 2.22	$[-10,10]$	0
f_3	Schwefel 1.2	$[-100,100]$	0
f_4	Schwefel 2.21	$[-100,100]$	0
f_5	Rosenbrock	$[-30,30]$	0
f_6	Step	$[-100,100]$	0
f_7	Quartic with noise	$[-1.28,1.28]$	0
f_8	Schwefel 2.26	$[-500,500]$	$-418.98 \cdot D$
f_9	Rastrigin	$[-5.12,5.12]$	0
f_{10}	Ackley	$[-32,32]$	0
f_{11}	Griewank	$[-600,600]$	0
f_{12}	Penalized	$[-50,50]$	0

computational time complexity of MABC is $O[G_{max} \cdot (SN \cdot f + p \cdot SN \cdot f)] = O(G_{max} \cdot SN \cdot f)$. For MEABC, its computational time complexity is $O[G_{max} \cdot (SN \cdot f)] = O(G_{max} \cdot SN \cdot f)$. Although the above four ABC algorithms have the same computational time complexity, our approach MEABC is simpler and easier to implement.

4. Experimental verifications

4.1. Test problems

There are twelve benchmark functions used in the following experiments. These problems were utilized in previous studies [16,62,63]. All these problems should be minimized. The brief descriptions of these benchmark problems are listed in Table 1. More details about the definition of benchmark problems can be found in [65].

4.2. Comparison of MEABC with other ABC variants

In this section, the proposed MEABC is compared with three other ABC algorithms including the original ABC, GABC [71], and MABC [17]. The current experiment includes three series: (1) comparison of MEABC with ABC; (2) comparison of MEABC with GABC and (3) comparison of MEABC with MABC.

4.2.1. Comparison of MEABC with ABC

By the suggestions of [23], the population size *SN* is set to 100 for the original ABC, and the number of food sources in the population is 50. For MEABC, all bees are regarded as the same type, and the number of food sources in the population is equal to the number of bees. So, the population size of MEABC is set to 50. The parameter *limit* used in ABC is set to 100, but MEABC does not contain this parameter. Based on a previous study [71], the constant value *C* is set to 1.5. Each algorithm stops when the number of function evaluations (FEs) reaches the maximum FEs (MAX_FEs), where MAX_FEs $= 5000 \cdot D$, and *D* is the dimension of the problem.

Table 2 summarizes the computational results achieved by ABC and MEABC for $D = 30$, where "Mean" indicates the mean best function value and "Std Dev" represents the corresponding standard deviation. The best results are shown in bold. As seen, MEABC outperforms ABC on all functions except for f_6; on this function, both of them can easily converge to the global optimum. By the ensemble of multi-strategy, MEABC achieves promising results on both unimodal and multimodal functions except f_3 (Schwefel 1.2). Though f_3 is a unimodal function, the rotated search space hinders the search of ABC. According to our literature review, the original ABC and its most modifications cannot find reasonable solutions for this function. Fig. 2 presents the convergence graphs of ABC and MEABC on some representative functions. It shows MEABC achieves faster convergence rate than ABC.

The above experiment compares the quality of the final solutions achieved by ABC and MEABC. In the following experiment, we present the comparison of the convergence speed and success rate between the two algorithms. A threshold value of the objective function is selected for each test function. For most functions, the threshold is set to 1E−20. For other functions that ABC algorithms are difficult to solve, low accuracy of the threshold is employed. The specific threshold values are listed in the third column of Table 3. The stopping criterion is that each algorithm is terminated when the best function value found so far is below the predefined threshold value or the number of FEs reaches to its maximum value (5.00E+05). For each test function, each algorithm is run 30 times. The mean number of FEs required to converge to the threshold and successful running rate (SR) are recorded.

Table 3 lists the results of mean number of FEs (Mean FEs) and successful running rate (SR), where "NA" represents that no runs of the corresponding algorithm converged below the predefined threshold before meeting the maximum number of

Table 2
Results achieved by ABC and MEABC when $D = 30$. The best results among the comparison are shown in bold.

Functions	ABC		MEABC	
	Mean	Std. dev.	Mean	Std. dev.
f_1	1.14E−15	3.58E−16	**4.85E−40**	**2.31E−40**
f_2	1.49E−10	2.34E−10	**1.25E−21**	**3.56E−21**
f_3	1.05E+04	3.37E+03	**9.81E+03**	**2.49E+03**
f_4	4.07E+01	1.72E+01	**4.89E+00**	**1.37E+00**
f_5	1.28E+00	1.05E+00	**2.86E−01**	**3.48E−01**
f_6	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_7	1.54E−01	2.93E−01	**2.29E−02**	**1.38E−02**
f_8	−12490.5	5.87E+01	**−12569.5**	**1.59E−12**
f_9	7.11E−15	2.28E−15	**0.00E+00**	**0.00E+00**
f_{10}	1.60E−09	4.32E−09	**2.90E−14**	**1.32E−14**
f_{11}	1.04E−13	3.56E−13	**0.00E+00**	**0.00E+00**
f_{12}	5.46E−16	3.46E−16	**3.02E−17**	**0.00E+00**

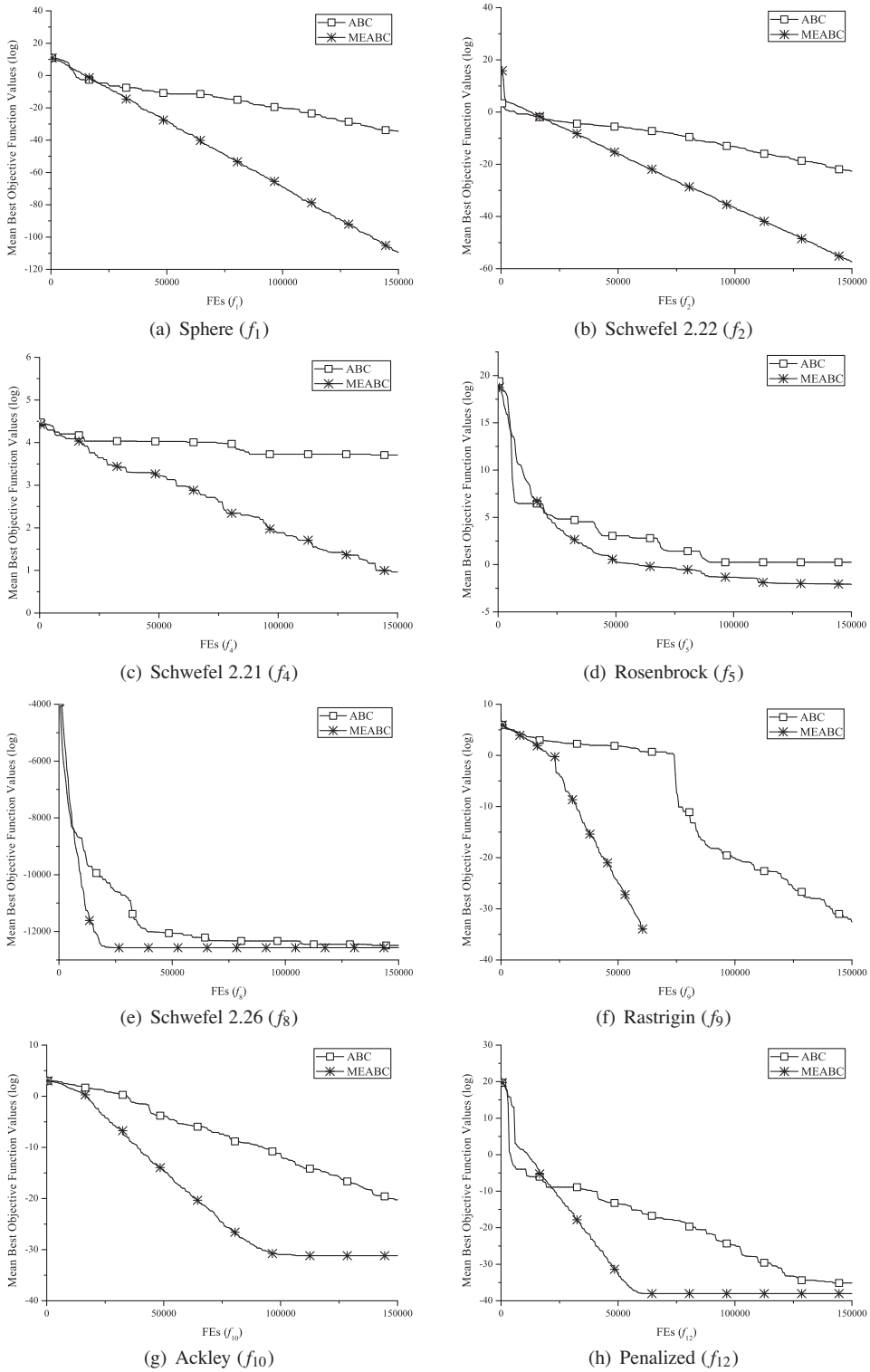


Fig. 2. The convergence characteristics of ABC and MEABC when $D = 30$.

FEs. The best results are shown in bold. From the results, MEABC shows faster convergence speed than ABC on all functions except for f_6; on this function, ABC is slightly faster than MEABC. From the results of total average FEs, MEABC costs less FEs to reach the threshold. The acceleration rate between MEABC and ABC is 1.83. The original ABC achieves a lower SR (27.5%)

Table 3
Results for mean number of FEs (Mean FEs) and successful running rate (SR) under predefined accuracy level (threshold). The best results among the comparison are shown in bold.

Functions	D	Threshold	ABC		MEABC	
			Mean FEs	SR (%)	Mean FEs	SR (%)
f_1	30	1.00E−20	NA	0	**8.85E+04**	**100**
f_2	30	1.00E−20	NA	0	**1.49E+05**	**100**
f_3	30	1.00E−01	NA	0	NA	0
f_4	30	1.00E−01	NA	0	**4.13E+05**	**100**
f_5	30	1.00E−01	3.35E+05	50	**2.18E+05**	**100**
f_6	30	1.00E−20	**1.71E+04**	**100**	1.77E+04	**100**
f_7	30	1.00E−01	NA	0	**4.68E+04**	**100**
f_8	30	−12,569	1.57E+05	**100**	**3.41E+04**	**100**
f_9	30	1.00E−20	2.69E+05	80	**5.39E+04**	**100**
f_{10}	30	1.00E−20	NA	0	NA	0
f_{11}	30	1.00E−20	NA	0	**9.23E+04**	**100**
f_{12}	30	1.00E−20	NA	0	NA	0
Total average			3.98E+05	27.5	**2.18E+05**	**75**

because it fails to solve 8 functions. MEABC successfully converges to the threshold on 9 functions, and achieves a higher SR (75%).

4.2.2. Comparison of MEABC with GABC

In this section, we compare the performance of MEABC with GABC. To have a fair comparison, we use the same parameter settings as described in [71]. For GABC and MEABC, the *SN* and the constant value *C* are set to 80 and 1.5, respectively. Each algorithm stops when the number of function evaluations (FEs) reaches the maximum FEs (MAX_FEs = 4.0E+05). For each function, experiment is repeated 30 times independently. And the reported results are the means and standard deviations of the statistical experimental data.

The literature [71] reported the computational results of GABC on 6 functions, and we also use the same benchmark for this comparison. Table 4 gives the comparison results between GABC and MEABC, where D is the dimensional size, "Mean" indicates the mean best function value, and "Std Dev" represents the corresponding standard deviation. The best results are shown in bold. From the results, GABC achieves better results on two low-dimensional problems. For Generalized Schaffer function, GABC outperforms MEABC for $D = 3$, while both of them can converge to the global optimum for $D = 2$. For Rosenbrock function, GABC performs better than MEABC for $D = 2$ and $D = 3$. For the rest 4 functions, MEABC obtains better results than GABC when $D = 30$ and 60.

4.2.3. Comparison of MEABC with MABC

This section presents a comparison between MABC and MEABC. For the sake of fair competition, the same parameter settings as described in [17]. For MABC and MEABC, the *SN* and MAX_FEs are set to 75 and 1.5E+05, respectively. The selective probability p used in MABC is set to 0.7. For MEABC, the constant value *C* is equal to 1.5. Throughout the experiment, each algorithm is run 30 times per function, and the mean and standard deviation results are reported.

Table 5 lists the computational results achieved by MABC and MEABC, where D is the dimensional size, "Mean" indicates the mean best function value, and "Std Dev" represents the corresponding standard deviation. The best results are shown in bold. It can be seen that MABC achieves better results than MEABC on only one function. Both MABC and MEABC can find the global optimum on 4 functions. For the rest 6 functions, MEABC outperforms MABC.

Table 4
Results achieved by GABC and MEABC when $D = 30$. The best results among the comparison are shown in bold.

Functions	D	GABC		MEABC	
		Mean	Std. dev.	Mean	Std. dev.
Generalized Schaffer	2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	3	**1.85E−18**	**1.01E−17**	1.69E−16	2.34E−16
Rosenbrock	2	**1.68E−04**	**1.45E−04**	5.67E−03	3.53E−03
	3	**2.66E−03**	**2.22E−03**	8.73E−03	5.92E−03
Sphere	30	4.18E−16	7.37E−17	**3.94E−72**	**4.18E−72**
	60	1.43E−15	1.38E−15	**2.35E−31**	**5.49E−31**
Griewank	30	2.96E−17	4.99E−17	**0.00E+00**	**0.00E+00**
	60	7.55E−16	4.13E−16	**0.00E+00**	**0.00E+00**
Rastrigin	30	1.33E−14	2.45E−14	**0.00E+00**	**0.00E+00**
	60	3.52E−13	1.24E−13	**0.00E+00**	**0.00E+00**
Ackley	30	3.22E−14	3.25E−15	**2.69E−14**	**3.76E−14**
	60	1.00E−13	6.09E−15	**8.16E−14**	**6.21E−14**

Table 5
Results achieved by MABC and MEABC when $D = 30$. The best results among the comparison are shown in bold.

Functions	MABC		MEABC	
	Mean	Std. dev.	Mean	Std. dev.
Sphere	9.43E−32	6.67E−32	**4.85E−40**	**2.31E−40**
Schwefel 2.22	2.40E−17	9.02E−18	**1.25E−21**	**3.56E−21**
Schwefel 2.21	1.02E+01	1.49E+00	**4.89E+00**	**1.37E+00**
Rosenbrock	6.11E−01	4.55E−01	**2.86E−01**	**3.48E−01**
Step	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Quartic with noise	3.71E−02	8.53E−03	**2.29E−02**	**1.38E−02**
Schwefel 2.26	**−12569.5**	**4.53E−13**	**−12569.5**	1.59E−10
Rastrigin	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Ackley	4.13E−14	2.17E−15	**2.90E−14**	**1.32E−14**
Griewank	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Penalized	**1.90E−32**	**3.70E−33**	3.02E−17	0.00E+00

4.3. Comparison of MEABC with some recently proposed PSO algorithms

PSO is one of the most popular swarm intelligence algorithms, which has shown fast convergence and good search abilities on many benchmark and real-world optimization problems [7,15,43]. In this section, MEABC is compared with some recently proposed PSO algorithms. The involved algorithms are listed as follows:

- Self-organizing hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC) [48].
- Fully informed particle swarm (FIPS) [38].
- Dynamic multi-swarm PSO (DMS-PSO) [32].
- Comprehensive learning PSO (CLPSO) [31].
- Adaptive PSO (APSO) [69].
- Our approach MEABC.

For MEABC, the same parameter settings are used as described in Section 4.2. For the rest five PSO algorithms, the population size is set to 20 by the suggestions of [69]. The other parameters of HPSO-TVAC, FIPS, DMS-PSO, CLPSO, and APSO are described in [69]. For all algorithms, the maximum number of fitness evaluations (MAX_FEs) is set to 2.00E+05. All the experiments are conducted 30 times, and the mean best function values are recorded.

Table 6 shows the mean best functions values achieved by MEABC and five other PSO algorithms. The best results among the comparison are shown in bold. Results of HPSO-TVAC, FIPS, DMS-PSO, CLPSO, and APSO were taken from Table 6 in [69]. From the results, MEABC outperforms FIPS and DMS-PSO on 6 functions, while FIPS and DMS-PSO performs better than MEABC on 3 functions. All algorithms can find the global optimum on the Step function. CLPSO, APSO and MEABC successfully solve Schwefel 2.26, while the other algorithms converge to near-optimal solutions. HPSO-TVAC achieves better results than MEABC on the penalized function, while MEABC find more accurate solutions on 8 functions. For the comparison of MEABC with APSO, both of them win 4 functions. It seems that they obtain similar performance on the test suite.

In order to compare the performance of multiple algorithms on the test suite, Friedman and Wilcoxon tests are conducted [9,19,20]. Table 7 presents the statistical results achieved by Friedman and Wilcoxon tests. The best ranking (with the lowest ranking value) and the p-values below 0.05 (the significant level) are shown in bold. As seen, the performance of the six algorithms ranks as follows: MEABC, APSO, CLPSO, FIPS, DMS-PSO, and HPSO-TVAC. The highest average ranking is obtained by the MEABC algorithm. It demonstrates that MEABC is the best one among the six algorithms. The p-values show that MEABC is only significantly better than HPSO-TVAC.

Table 6
Mean best function values for MEABC and five PSO algorithms, where "$w/t/l$" means that MEABC wins in w functions, ties in t functions, and loses in l functions, compared with its competitors. The best results among the comparison are shown in bold.

Functions	D	MAX_FEs	FIPS Mean	HPSO-TVAC Mean	DMS-PSO Mean	CLPSO Mean	APSO Mean	MEABC
Sphere	30	2.00E+05	3.21E−30	3.38E−41	3.85E−54	1.89E−19	**1.45E−150**	1.55E−56
Schwefel 2.22	30	2.00E+05	1.32E−14	6.90E−23	2.61E−29	1.01E−16	**5.15E−84**	1.08E−29
Rosenbrock	30	2.00E+05	2.25E+01	1.30E+01	3.23E+01	1.10E+01	2.84E+00	**1.34E−01**
Step	30	2.00E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Quartic with noise	30	2.00E+05	**2.55E−03**	5.54E−02	1.10E−02	3.92E−03	4.66E−03	2.18E−02
Schwefel 2.26	30	2.00E+05	−10113.8	−10868.6	−9593.3	**−12569.5**	**−12569.5**	**−12569.5**
Rastrigin	30	2.00E+05	3.00E+01	2.39E+00	2.81E+01	2.57E−11	5.80E−15	**0.00E+00**
Ackley	30	2.00E+05	**7.69E−15**	2.06E−10	8.52E−15	2.01E−12	1.11E−14	1.09E−14
Griewank	30	2.00E+05	9.04E−04	1.07E−02	1.31E−02	6.53E−13	1.67E−02	**0.00E+00**
Penalized	30	2.00E+05	1.22E−31	7.07E−30	**2.05E−32**	1.59E−21	3.76E−31	3.02E−17
w*/*t*/*l			6/1/3	8/1/1	6/1/3	6/2/2	4/2/4	–

Table 7
Results achieved by Friedman and Wilcoxon tests. The best ranking (with the lowest ranking value) and the *p*-values below 0.05 are shown in bold.

Friedman test		Wilcoxon test	
Algorithms	Rankings	MEABC vs.	*p*-values
MEABC	**2.65**	FIPS	1.73E−01
APSO	2.75	HPSO-TVAC	**2.10E−02**
CLPSO	3.65	DMS-PSO	2.14E−01
FIPS	3.75	CLPSO	2.08E−01
DMS-PSO	3.85	APSO	4.01E−01
HPSO-TVAC	4.35		

4.4. Comparison of MEABC with some recently proposed DE algorithms

Like PSO, DE is also a popular evolutionary optimization technique. Some experimental studies show that DE is competitive to PSO and simple genetic algorithm (SGA) [60]. For its simple concept, easy implementation yet effectiveness, it has attracted much attention. Various DE algorithms have been designed to satisfy the requirements of different problems. In this section, MEABC is compared with some recently proposed DE algorithms. The involved algorithms are listed as follows:

- Self-adapting DE (jDE) [4].
- Self-adaptive DE (SaDE) [44].
- Opposition-based DE (ODE) [45].
- Improved ABC (IABC) [16].
- Our approach MEABC.

For all algorithms, the same maximum number of fitness evaluations (MAX_FEs) are used. The specific settings of MAX_FEs are listed in the third column of Table 8 by the suggestions of [16]. For MEABC, the *SN* and *C* are set to 50, and 1.5, respectively. For IABC, the parameter settings are described in [16]. For other parameters of jDE, SaDE, and IABC, we follow the literature [16]. For ODE, the population size, scale factor, crossover rate, and jumping rate are set to 100, 0.5, 0.9, and 0.3, respectively [45]. All the experiments are conducted 30 times, and the mean best function values are recorded.

Table 8 shows the mean best functions values achieved by MEABC, jDE, SaDE, ODE, and IABC. Results of jDE, SaDE, and IABC were taken from Table 5 in [16]. From the results, MEABC achieves better results than SaDE on 10 functions. For the rest 2 functions, SaDE performs better than MEABC. jDE outperforms MEABC on 3 functions, while MEABC wins on 8 functions. MEABC achieves better results than ODE on 9 functions, while ODE performs better on the rest 3 functions. For the comparison of IABC and MEABC, IABC outperforms MEABC on 6 functions, while MEABC wins on 3 functions. For the rest 3 functions, both MEABC and IABC can converge to the global optimum. The Step function is a simple unimodal function, and many algorithms can easily converge to the global optimum. However, all algorithms except for MEABC fail to solve it under the predefined MAX_FEs (1.00E+04). It demonstrates that MEABC converges faster than other algorithms on this function. For Schwefel 2.26, jDE, IABC, and MEABC can find the global optimum, while SaDE and ODE fail. For Rastrigin, both MEABC and IABC can find the global optimum, but the other three DE algorithms are trapped into local minima.

Table 9 presents the statistical results achieved by Friedman and Wilcoxon tests. The best ranking (with the lowest ranking value) and the *p*-values below 0.05 (the significant level) are shown in bold. As seen, the performance of the six

Table 8
Mean best function values for MEABC, SaDE, jDE, ODE, and IABC, where "*w*/*t*/*l*" means that MEABC wins in *w* functions, ties in *t* functions, and loses in *l* functions, compared with its competitors. The best results among the comparison are shown in bold.

Functions	*D*	MAX_FEs	SaDE Mean	jDE Mean	ODE Mean	IABC Mean	MEABC Mean
Sphere	30	1.50E+05	4.50E−20	2.50E−28	5.53E−28	**5.34E−178**	9.51E−82
Schwefel 2.22	30	2.00E+05	1.90E−14	1.50E−23	5.71E−12	**8.82E−127**	6.04E−57
Schwefel 1.2	30	5.00E+05	9.00E−37	5.20E−14	1.49E−13	**1.78E−65**	1.86E+03
Schwefel 2.21	30	5.00E+05	7.40E−11	1.40E−15	**3.92E−66**	4.98E−38	6.08E−06
Rosenbrock	30	2.00E+06	1.80E+01	8.00E−02	7.98E+00	**4.75E−03**	2.38E+00
Step	30	1.00E+04	9.30E+02	1.00E+03	1.52E+02	3.33E−02	**0.00E+00**
Quartic with noise	30	3.00E+05	4.80E−03	3.30E−03	**7.35E−04**	2.42E−03	2.29E−03
Schwefel 2.26	30	1.00E+05	−12564.8	**−12569.5**	−5695.9	**−12569.5**	**−12569.5**
Rastrigin	30	1.00E+05	1.20E−03	1.50E−04	9.41E+01	**0.00E+00**	**0.00E+00**
Ackley	30	5.00E+04	2.70E−03	3.50E−04	1.06E−03	3.87E−14	**3.37E−14**
Griewank	30	5.00E+04	7.80E−04	1.90E−05	4.45E−06	**0.00E+00**	**0.00E+00**
Penalized	30	5.00E+04	1.90E−05	1.60E−07	1.05E−07	**1.57E−32**	3.02E−17
w*/*t*/*l			10/0/2	8/1/3	9/0/3	3/3/6	–

Table 9
Results achieved by Friedman and Wilcoxon tests. The best ranking (with the lowest ranking value) and the p-values below 0.05 are shown in bold.

Friedman test		Wilcoxon test	
Algorithms	Rankings	MEABC vs.	p-values
IABC	**1.58**	SaDE	5.97E−02
MEABC	2.33	jDE	4.24E−01
jDE	3.25	ODE	2.09E−01
ODE	3.50	IABC	5.15E−01
SaDE	4.33		

algorithms ranks as follows: IABC, MEABC, jDE, ODE, and SaDE. The highest average ranking is obtained by the IABC algorithm. The p-values show that MEABC is not significantly better than other algorithms.

The IABC employs a new parameter M to control how many dimensions to be updated, where $1 \leqslant M \leqslant D$. Results confirm that the performance of IABC does highly depend on the M which has been set experimentally for each specific problem. For different problems, IABC defined different M to obtain good results (please see Table 1 in [16]). For example, IABC achieved better results for Sphere, Schwefel 2.21, Rastrigin, and Griewank when M is set to 25, 10, 1, and 3, respectively [16]. From the comparison between MEABC and IABC, IABC wins 5 on all 7 unimodal functions. For multimodal functions, IABC achieves better results than MEABC one only one function, while MEABC also performs better than IABC on one function. For the rest 3 multimodal functions, both MEABC and IABC achieve the same results. The computational time complexity of IABC is $O(G_{max} \cdot SN \cdot M \cdot f)$, which is higher than the original ABC and MEABC $(O(G_{max} \cdot SN \cdot f))$. The IABC has 3 parameters (SN, p, and M), while our approach MEABC has two (SN and C). IABC outperforms our approach MEABC, but on the other hand, MEABC is simpler and its performance is less sensitive to its parameters. It is worth to mention that both MEABC and IABC obtain similar performance on complex multimodal problems.

4.5. Test on the CEC 2013 shifted and rotated benchmark problems

To further verify the performance of MEABC, a set of recently proposed 28 CEC 2013 shifted and rotated benchmark problems are used. A summary of these functions are listed in Table 10. More detailed definitions of them can be found in [33]. In this paper, we only consider the test suite with $D = 30$.

Table 10
Summary of the CEC 2013 benchmark problems.

Problems	Name	Global optimum
F_1	Sphere Function	−1400
F_2	Rotated High Conditioned Elliptic Function	−1300
F_3	Rotated Bent Cigar Function	−1200
F_4	Rotated Discus Function	−1100
F_5	Different Powers Function	−1000
F_6	Rotated Rosenbrock's Function	−900
F_7	Rotated Schaffers F7 Function	−800
F_8	Rotated Ackley's Function	−700
F_9	Rotated Weierstrass Function	−600
F_{10}	Rotated Griewank's Function	−500
F_{11}	Rastrigin's Function	−400
F_{12}	Rotated Rastrigin's Function	−300
F_{13}	Non-Continuous Rotated Rastrigin's Function	−200
F_{14}	Schwefel's Function	−100
F_{15}	Rotated Schwefel's Function	100
F_{16}	Rotated Katsuura Function	200
F_{17}	Lunacek Bi_Rastrigin Function	300
F_{18}	Rotated Lunacek Bi_Rastrigin Function	400
F_{19}	Expanded Griewank's plus Rosenbrock's Function	500
F_{20}	Expanded Scaffer's F6 Function	600
F_{21}	Composition Function 1 ($n = 5$, Rotated)	700
F_{22}	Composition Function 2 ($n = 3$, Unrotated)	800
F_{23}	Composition Function 3 ($n = 3$, Rotated)	900
F_{24}	Composition Function 4 ($n = 3$, Rotated)	1000
F_{25}	Composition Function 5 ($n = 3$, Rotated)	1100
F_{26}	Composition Function 6 ($n = 5$, Rotated)	1200
F_{27}	Composition Function 7 ($n = 5$, Rotated)	1300
F_{28}	Composition Function 8 ($n = 5$, Rotated)	1400

In this section, we compare MEABC with three other recently published algorithms on CEC 2013. The involved algorithms are listed as follows.

- Self-adaptive heterogeneous PSO (fk-PSO) [40].
- Population variance-based adaptive DE (ADE) [6].
- DE with automatic parameter configuration (DE-APC) [13].
- The proposed MEABC.

For fk-PSO, ADE, and DE-APC, we use the same parameter settings for these three methods as in their original papers. For MEABC, the SN and C are set to 60, and 1.5, respectively. By the suggestions of [33], the number of MAX_FEs is set to 3.00E+05 for all algorithms ($D = 30$). For each test function, each algorithm is run 51 times. Throughout the experiments, the mean function error value ($f(x) - f(x^o)$) are reported, where x is the best solution found by the algorithm in a run, and x^o is the global optimum of the test function. Error values less than 1E−08 are taken as zero [33].

Table 11 presents the mean error function values achieved by the four algorithms on the test suite. To have a fair comparison, results of DE-APC, fk-PSO, and ADE were taken from their original papers. From the results, MEABC performs better than DE-APC on 13 functions, while DE-APC outperforms MEABC on 11 functions. For the rest 4 functions, both MEABC and DE-APC achieve the same results. Compared to fk-PSO and ADE, MEABC obtains better results on 13 functions, and loses in 12 functions. For the rest 3 functions, they achieve the same results.

Table 12 presents the statistical results achieved by Friedman and Wilcoxon tests. The best ranking (with the lowest ranking value) is shown in bold. From the results, the performance of the four algorithms ranks as follows: MEABC,

Table 11
Results for CEC 2013 benchmark functions when $D = 30$, where "$w/t/l$" means that MEABC wins in w functions, ties in t functions, and loses in l functions, compared with its competitors. The best results among the comparison are shown in bold.

Functions	DE-APC Mean error	fk-PSO Mean error	ADE Mean error	MEABC Mean error
F_1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_2	**1.75E+05**	1.59E+06	2.18E+06	1.23E+06
F_3	3.21E+06	2.40E+08	**1.65E+03**	1.40E+08
F_4	**2.21E−01**	4.78E+02	1.70E+04	8.35E+04
F_5	**0.00E+00**	**0.00E+00**	1.40E−07	**0.00E+00**
F_6	9.35E+00	2.99E+01	**8.29E+00**	1.01E+01
F_7	2.18E+01	6.39E+01	**1.29E+00**	9.23E+01
F_8	**2.09E+01**	**2.09E+01**	**2.09E+01**	**2.09E+01**
F_9	3.07E+01	1.85E+01	**6.30E+00**	2.88E+01
F_{10}	6.42E−02	2.29E−01	**2.16E−02**	5.57E+00
F_{11}	3.08E+00	2.36E+01	5.84E+01	**0.00E+00**
F_{12}	**3.17E+01**	5.64E+01	1.15E+02	2.07E+02
F_{13}	**7.55E+01**	1.23E+02	1.31E+02	2.29E+02
F_{14}	3.84E+03	7.04E+02	3.20E+03	**1.37E+01**
F_{15}	4.14E+03	3.42E+03	5.61E+03	**3.41E+02**
F_{16}	2.46E+00	**8.48E−01**	2.39E+00	1.44E+00
F_{17}	5.92E+01	5.26E+01	1.02E+02	**3.04E+01**
F_{18}	**6.04E+01**	6.81E+01	1.82E+02	1.80E+02
F_{19}	2.30E+00	3.12E+00	5.40E+00	**3.94E−01**
F_{20}	1.26E+01	1.20E+01	**1.13E+01**	1.56E+01
F_{21}	2.67E+02	3.11E+02	3.19E+02	**2.10E+02**
F_{22}	4.56E+03	8.59E+02	2.50E+03	**1.78E+01**
F_{23}	4.18E+03	**3.57E+03**	5.81E+03	5.16E+03
F_{24}	2.92E+02	2.48E+02	**2.02E+02**	2.81E+02
F_{25}	2.99E+02	2.49E+02	**2.30E+02**	2.74E+02
F_{26}	3.29E+02	2.95E+02	2.18E+02	**2.01E+02**
F_{27}	1.19E+03	7.76E+02	**3.26E+02**	4.02E+02
F_{28}	**3.00E+02**	4.01E+02	**3.00E+02**	**3.00E+02**
w/t/l	13/4/11	13/3/12	13/3/12	–

Table 12
Results achieved by Friedman and Wilcoxon tests. The best ranking is shown in bold.

Friedman test		Wilcoxon test	
Algorithms	Rankings	MEABC vs.	p-values
MEABC	**2.43**	DE-APC	**8.64E−01**
fk-PSO	2.46	fk-PSO	**4.59E−01**
DE-APC	2.54	ADE	**8.82E−01**
ADE	2.57		

fk-PSO, DE-APC, and ADE. The highest average ranking is obtained by the MEABC algorithm. It demonstrates that MEABC is the best algorithm. The p-values show that MEABC is not significantly better than other algorithms. Although MEABC is slightly better than fk-PSO, DE-APC, and ADE, MEABC achieves significant improvements on 5 functions $f_{11}, f_{14}, f_{15}, f_{19}$, and f_{22}.

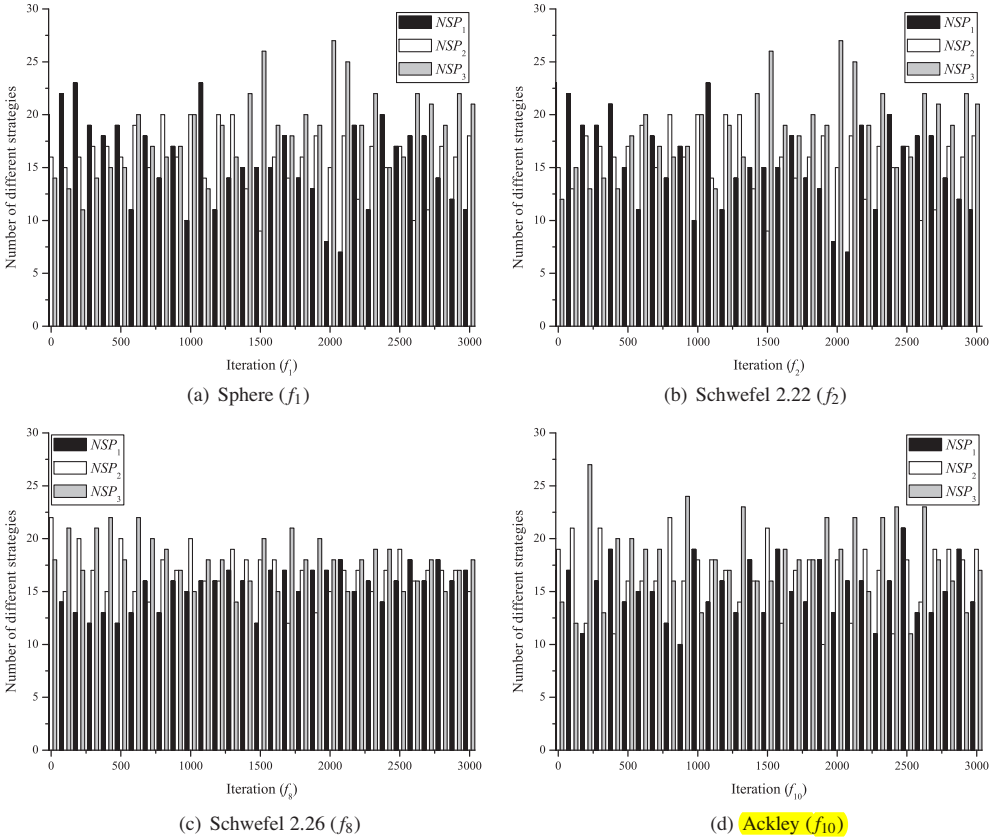
4.6. Study on the ensemble of multiple solution search strategies

The MEABC employs a new encoding method, in which each food source (solution) consists a position vector (X_i) and an independent solution search strategy (S_i). During the search process, the solution search strategy for each food source is dynamically adjusted in terms of the quality of new candidate solutions (see Algorithm 1). In this section, we investigate the effects of the ensemble of multi-strategy.

Initially, the solution search strategy S_i for each solution is randomly assigned a strategy SP_j, where $j \in \{1, 2, 3\}$ is a random integer. Let $NSP_j(t)$ be the number of jth strategy assigned to the food sources in the population at the tth iterations (*cycles*). After population initialization, $NSP_1(0)$, $NSP_2(0)$, and $NSP_3(0)$ food sources are assigned the original ABC, GABC, and modified ABC/best/1 solution search strategies, respectively.

In order to study the ensemble of multiple solution search strategies, two series of experiments are conducted. The first one focuses on investigating the changes of $NSP_j(t)$. This helps to recognize the roles of different strategies at different evolutionary stages. The second one calculates the improvement of probability for solution search strategies achieved by different ABC algorithms. It aims to compare the efficiency of the ensemble of multi-strategy with a single solution search strategy. In the experiments, the parameter settings of ABC, GABC, MABC, and MEABC are listed as follows. All these algorithms use the same MAX_FEs (1.5E+05). For other parameters, we use the same values as described in Section 4.2.

Fig. 3 presents the changes of strategies in the population during the search process. As seen, NSP_1 is larger than NSP_2 and NSP_3 for f_1 and f_2 at the beginning stage. It means that the original ABC solution search strategy plays an important role in the current population, and more candidate solutions are generated by this strategy. As iterations increase, more operations are completed by the modified ABC/best/1 strategy than the original ABC and GABC. For f_8, the search prefers to conduct the GABC and modified ABC/best/1 at the beginning stage. At the middle and last stages, MEABC has converged to the global optimum, and the strategy for each food source is changed every iteration. Therefore, three strategies have the same chance to be conducted. For f_{10}, the modified ABC/best/1 strategy generates new candidate solutions with the highest frequency,



(a) Sphere (f_1) (b) Schwefel 2.22 (f_2)

(c) Schwefel 2.26 (f_8) (d) Ackley (f_{10})

Fig. 3. The changes of strategies in the population during the search process.

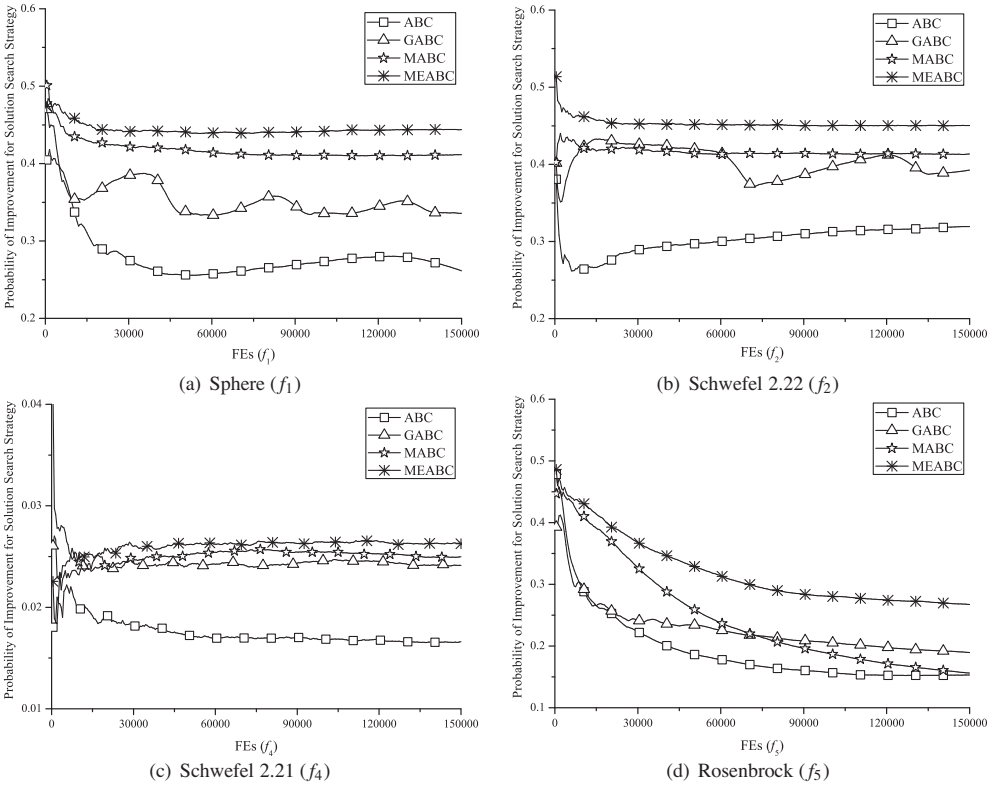


Fig. 4. The probability of improvement for solution search strategies achieved by the four ABC algorithms.

while less operations are conducted by the original ABC than the other two strategies. The above results show that three strategies in MEABC are dynamically changed during the search process. If a solution search strategy can generate better candidate solutions than their corresponding parent solutions, more food sources will be assigned this strategy. By the ensemble of multi-strategy, the strategy selection in MEABC is performed based on the problem characteristics and search status.

Although the first experiment confirms that the strategies assigned to food sources are dynamically changed during the search process, it does not mean that the ensemble of multi-strategy is better than a single solution search strategy. In the second experiment, we investigate the probability of improvement for the ensemble of multi-strategy (MEABC) and one or two strategies (ABC, GABC, and MABC). The probability of improvement (*pro_imp*) is defined by

$$pro\_imp = \frac{suc\_num\_sol}{num\_sol}, \tag{8}$$

where *num_sol* is the total number of candidate solutions generated by the solution search strategy(s) for a ABC algorithm, and *suc_num_sol* is the number of candidate solutions which are better than their corresponding parent solutions (called successful solutions).

Fig. 4 shows the probability of improvement for solution search strategies achieved by ABC, GABC, MABC, and MEABC on four representative functions. For all algorithms, the initial *pro_imp* is high. As iterations increase, the *pro_imp* is changed according to the quality of new candidate solutions. MEABC can continue to find better solutions under the predefined maximum number of FEs. So, the *pro_imp* of them keep high levels during the whole evolution. Compared to other three ABC algorithms, MEABC achieves a higher *pro_imp*. It demonstrates that the ensemble of multi-strategy provides more chances of finding better candidate solutions than using one or two solution search strategies. By the hybridization of two solution search strategies (the original ABC and ABC/best/1), MABC obtains higher *pro_imp* than ABC and GABC which employ only one strategy (except for f_5 at the last stage of the evolution). It seems that ABC with two or more solution search strategies are better than that with a single strategy.

5. Conclusions

This paper presents a new ABC algorithm called multi-strategy ensemble ABC (MEABC) to achieve a tradeoff between exploration and exploitation. In MEABC, a pool of distinct solution search strategies coexists throughout the search process and competes to produce offspring. Each food source (solution) consists a position vector and an independent solution search

strategy. Initially, each food source is randomly assigned a solution search strategy from the strategy pool. When searching a food source, each bee generates offspring according to the assigned strategy of the food source. During the search process, the strategy for each food source is dynamically changed in terms of the quality of new candidate solutions. In order to verify the performance of MEABC, a set of numerical benchmark functions are tested in the experiments.

Compared to ABC, GABC, and MABC, MEABC achieves faster convergence speed or more accurate solutions. Compared to other popular PSO and DE variants, MEABC achieves the highest ranking and outperforms them on the majority of test functions. Although IABC performs better than MEABC, MEABC is simpler and has less control parameters to be tuned. Moreover, both MEABC and IABC obtain similar performance on complex multimodal problems. Computational results on the CEC 2013 benchmark show that MEABC is competitive to other recently proposed algorithms.

Experimental results confirm that the solution search strategies assigned to food sources are dynamically changed during the search process. It helps to switch the search behaviors of bees and balance exploration and exploitation. By the ensemble of multi-strategy, MEABC obtains a higher probability to improve the quality of offspring than ABC with one or two solution search strategies. It seems that ABC with two or more solution search strategies are better than that with a single strategy. More solution search strategies will be tried in the future work.

The multi-strategy ensemble method can effectively improve the performance of ABC on continuous optimization problems. It may also work well on combinatorial optimization problems. This will be another research direction for future work.

Acknowledgment

The authors would like to thank the editor and anonymous reviewers for their detailed and constructive comments that help us to increase the quality of this work. This work is supported by the Humanity and Social Science Foundation of Ministry of Education of China (No. 13YJCZH174), the National Natural Science Foundation of China (Nos. 61305150, 61261039, and 61175127), the Science and Technology Plan Projects of Jiangxi Provincial Education Department (Nos. GJJ13744 and GJJ13762), and the Foundation of State Key Laboratory of Software Engineering (No. SKLSE2012-09-19).

References

[1] B. Akay, D. Karaboga, A modified Artificial Bee Colony algorithm for real-parameter optimization, Inform. Sci. (192) (2012) 120–142.
[2] B. Alatas, Chaotic bee colony algorithms for global numerical optimization, Exp. Syst. Appl. 37 (2010) 5682–5687.
[3] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, The best-so-far selection in artificial bee colony algorithm, Appl. Soft Comput. 11 (2) (2011) 2888–2901.
[4] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Žumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evolut. Comput. 10 (6) (2006) 646–657.
[5] S.C. Chu, P.W. Tsai, Computational intelligence based on behaviors of cats, Int. J. Innov. Comput., Inform. Control 3 (1) (2007) 163–173.
[6] L.S. Coelho, H.V.H. Ayala, R.Z. Freire, Population's variance-based adaptive differential evolution for real parameter optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, 2013, pp. 1672–1677.
[7] Z. Cui, X. Cai, J.C. Zeng, Y.F. Yin, PID-controlled particle swarm optimization, J. Multip.-Val. Logic Soft Comput. 16 (6) (2010) 585–609.
[8] Z. Cui, X. Gao, Theory and applications of swarm intelligence, Neur. Comput. Appl. 21 (2) (2012) 205–206.
[9] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolut. Comput. 1 (1) (2011) 3–18.
[10] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst., Man Cybernet. – Part B: Cybernet. 26 (1996) 29–41.
[11] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Inform. Sci. 178 (15) (2008) 3096–3109.
[12] M. El-Abd, Generalized opposition-based artificial bee colony algorithm, in: Proceedings of IEEE Congress on Evolutionary Computation, 2012, pp. 1–4.
[13] S.M.M. Elsayed, R.A. Sarker, T. Ray, Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, 2013, pp. 1932–1937.
[14] A.P. Engelbrecht, Heterogeneous particle swarm optimization, in: International Conference on Swarm Intelligence, 2010, pp. 191–202.
[15] H.M. Feng, J.H. Horng, S.M. Jou, Bacterial foraging particle swarm optimization algorithm based fuzzy-VQ compression systems, J. Inform. Hid. Multim. Sig. Process. 3 (3) (2012) 227–239.
[16] W. Gao, S. Liu, Improved artificial bee colony algorithm for global optimization, Inform. Process. Lett. 111 (2011) 871–882.
[17] W. Gao, S. Liu, A modified artificial bee colony algorithm, Comp. Operat. Res. 39 (2012) 687–697.
[18] W. Gao, S. Liu, L. Huang, A global best artificial bee colony algorithm for global optimization, J. Comput. Appl. Math. 236 (2012) 2741–2753.
[19] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inform. Sci. 180 (20) (2010) 2044–2064.
[20] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization, J. Heurist. 15 (2009) 617–644.
[21] F. Kang, J. Li, Z. Ma, Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions, Inform. Sci. 181 (6) (2011) 3508–3531.
[22] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer engineering Department, 2005.
[23] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. 214 (2009) 108–132.
[24] D. Karaboga, B. Akay, A survey: algorithms simulating bee swarm intelligence, Artif. Intell. Rev. 31 (2009) 61–85.
[25] D. Karaboga, B. Akay, A modified artificial bee colony algorithm for constrained optimization problems, Appl. Soft Comput. 11 (2011) 3021–3031.
[26] D. Karaboga, B. Akay, Artificial bee colony programming for symbolic regression, Inform. Sci. 209 (2012) 1–15.
[27] D. Karaboga, B. Gorkemli, A combinatorial artificial bee colony algorithm for traveling salesman problem, in: International Symposium on Innovations in Intelligent Systems and Applications, 2011, pp. 50–53.
[28] M.H. Kashan, N. Nahavandi, A.H. Kashan, DisABC: a new artificial bee colony algorithm for binary optimization, Appl. Soft Comput. 12 (1) (2012) 342–352.
[29] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[30] G. Li, P. Niu, X. Xiao, Development and investigation of efficient artificial bee colony algorithm for numerical function optimization, Appl. Soft Comput. 12 (1) (2012) 320–332.

[31] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evolut. Comput. 10 (2006) 281–295.
[32] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer with local search, in: Proceedings of IEEE Congress on Evolutionary Computation, 2005, pp. 522–528.
[33] J.J. Liang, B.Y. Qu, P.N. Suganthan, A.G. Hernández-Díaz, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, Tech. Rep. 201212, 2013.
[34] R. Mallipeddi, S. Mallipeddi, P.N. Suganthan, Ensemble strategies with adaptive evolutionary programming, Inform. Sci. 180 (9) (2010) 1571–1581.
[35] R. Mallipeddi, P.N. Suganthan, Ensemble of constraint handling techniques, IEEE Trans. Evolut. Comput. 14 (4) (2010) 561–579.
[36] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2) (2011) 1679–1696.
[37] V.J. Manoj, E. Elias, Artificial bee colony algorithm for the design of multiplier-less nonuniform filter bank transmultiplexer, Inform. Sci. 192 (2012) 193–203.
[38] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evolut. Comput. 8 (3) (2004) 204–210.
[39] E. Mezura-Montes, R.E. Velez-Koeppel, Elitist artificial bee colony for constrained real-parameter optimization, in: IEEE Congress on Evolutionary Computation, 2010, pp. 1–8.
[40] F.V. Nepomuceno, A.P. Engelbrecht, A self-adaptive heterogeneous pso for real-parameter optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, 2013, pp. 361–368.
[41] S.N. Omkar, J. Senthilnath, R. Khandelwal, G.N. Naik, S. Gopalakrishnan, Artificial bee colony (ABC) for multi-objective design optimization of composite structures, Appl. Soft Comput. 11 (2011) 489–499.
[42] 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, Inform. Sci. 181 (12) (2011) 2455–2468.
[43] P. Puranik, P. Bajaj, A. Abraham, P. Palsodkar, A. Deshmukh, Human perception-based color image segmentation using comprehensive learning particle swarm optimization, J. Inform. Hid. Multim. Sig. Process. 2 (3) (2011) 227–235.
[44] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaption for global numerical optimization, IEEE Trans. Evolut. Comput. 13 (2) (2009) 398–417.
[45] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution, IEEE Trans. Evolut. Comput. 12 (1) (2008) 64–79.
[46] A. Rajasekhar, A. Abraham, M. Pant, Design of fractional order PID controller using sobol mutated artificial bee colony alogrithm, in: International Conference on Hybrid Intelligent Systems, 2011, pp. 151–156.
[47] A. Rajasekhar, A. Abraham, M. Pant, Levy mutated artificial bee colony algorithm for global optimization, in: IEEE International Conference on Systems, Man, and Cybernetics, 2011, pp. 655–662.
[48] A. Ratnaweera, S. Halgamuge, H. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evolut. Comput. 8 (2004) 240–255.
[49] S.L. Sabat, S.K. Udgata, A. Abraham, Artificial bee colony algorithm for small signal model parameter extraction of MESFET, Eng. Appl. Artif. Intell. 23 (5) (2010) 689–694.
[50] S. Samanta, S. Chakraborty, Parametric optimization of some non-traditional machining processes using artificial bee colony algorithm, Eng. Appl. Artif. Intell. 24 (2011) 946–957.
[51] A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, Appl. Soft Comput. 9 (2) (2009) 625–631.
[52] M. Sonmez, Artificial bee colony algorithm for optimization of truss structures, Appl. Soft Comput. 11 (2) (2011) 2406–2418.
[53] S. Sundar, A. Singh, A swarm intelligence approach to the quadratic minimum spanning tree problem, Inform. Sci. 180 (17) (2010) 3182–3191.
[54] S. Sundar, A. Singh, A hybrid heuristic for the set covering problem, Operat. Res. 12 (3) (2012) 345–365.
[55] S. Sundar, A. Singh, New heuristic approaches for the dominating tree problem, Appl. Soft Comput. 13 (12) (2013) 4695–4703.
[56] W.Y. Szeto, Y. Wu, S.C. Ho, An artificial bee colony algorithm for the capacitated vehicle routing problem, Euro. J. Operat. Res. 215 (1) (2011) 126–135.
[57] M.F. Tasgetiren, Q.K. Pan, P.N. Suganthan, A.H. Chen, A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops, Inform. Sci. 181 (16) (2011) 3459–3475.
[58] M.F. Tasgetiren, P.N. Suganthan, Q.K. Pan, An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem, Appl. Math. Comput. 215 (9) (2010) 3356–3368.
[59] P.W. Tsai, M.K. Khan, J.S. Pan, B.Y. Liao, Interactive artificial bee colony supported passive continuous authentication system, IEEE Syst. J. (2012), http://dx.doi.org/10.1109/JSYST.2012.2208153.
[60] J. Vesterstrom and R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, in: Proceedings of IEEE Congress on Evolutionary Computation, 2004, pp. 1980–1987.
[61] H. Wang, Z.J. Wu, S. Rahnamayan, Enhanced opposition-based differential evolution for high-dimensional optimization problems, Soft Comput. 15 (11) (2011) 2127–2140.
[62] H. Wang, S. Rahnamayan, H. Sun, M.G.H. Omran, Gaussian bare-bones differential evolution, IEEE Trans. Cybernet. 43 (2) (2013) 634–647.
[63] H. Wang, H. Sun, C. Li, S. Rahnamayan, J.S. Pan, Diversity enhanced particle swarm optimization with neighborhood search, Inform. Sci. 223 (2013) 119–135.
[64] X.S. Yang, Firefly algorithm, stochastic test functions and design optimization, Int. J. Bio-Insp. Comput. 2 (2) (2010) 78–84.
[65] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evolut. Comput. 3 (2) (1999) 82–102.
[66] W.C. Yeh, T.J. Hsieh, Solving reliability redundancy allocation problems using an artificial bee colony algorithm, Comp. Operat. Res. 38 (2011) 1465–1473.
[67] A.R. Yildiz, Optimization of cutting parameters in multi-pass turning using artificial bee colony-based approach, Inform. Sci. 220 (2013) 339–407.
[68] E.L. Yu, P.N. Suganthan, Ensemble of niching algorithms, Inform. Sci. 180 (15) (2010) 2815–2833.
[69] Z. Zhan, J. Zhang, Y. Li, H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst., Man Cybernet. – Part B: Cybernet. 39 (6) (2009) 1362–1381.
[70] S.Z. Zhao, P.N. Suganthan, Q. Zhang, Decomposition based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes, IEEE Trans. Evolut. Comput. 16 (3) (2012) 442–446.
[71] G. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Appl. Math. Comput. 217 (2010) 3166–3173.