Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 402616, 20 pages
http://dx.doi.org/10.1155/2014/402616

Research Article

Discrete and Continuous Optimization Based on Hierarchical Artificial Bee Colony Optimizer

Lianbo Ma,[1,2] Kunyuan Hu,[1] Yunlong Zhu,[1] Ben Niu,[3,4,5] Hanning Chen,[1] and Maowei He[1,2]

[1] *Department of Information Service & Intelligent Control, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China*
[2] *University of Chinese Academy of Sciences, Beijing 100039, China*
[3] *College of Management, Shenzhen University, Shenzhen 518060, China*
[4] *Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong*
[5] *Hefei Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei 230031, China*

Correspondence should be addressed to Ben Niu; drniuben@gmail.com

Received 11 October 2013; Revised 23 January 2014; Accepted 4 February 2014; Published 17 March 2014

Academic Editor: Roberto Natalini

This paper presents a novel optimization algorithm, namely, hierarchical artificial bee colony optimization (HABC), to tackle complex high-dimensional problems. In the proposed multilevel model, the higher-level species can be aggregated by the subpopulations from lower level. In the bottom level, each subpopulation employing the canonical ABC method searches the part-dimensional optimum in parallel, which can be constructed into a complete solution for the upper level. At the same time, the comprehensive learning method with crossover and mutation operator is applied to enhance the global search ability between species. Experiments are conducted on a set of 20 continuous and discrete benchmark problems. The experimental results demonstrate remarkable performance of the HABC algorithm when compared with other six evolutionary algorithms.

1. Introduction

Swarm intelligence (SI) is a new paradigm of artificial intelligence system whereby small entities are working together to produce an intelligent behaviour from simple rules [1–4]. Artificial bee colony algorithm (ABC) is one of the most popular members of the family of swarm intelligence, which simulates the social foraging behavior of a honeybee swarm [5, 6]. Due to its simple arithmetic and good robustness, the ABC algorithm has been widely used in solving many numerical optimizations [7–9] and engineering optimization problems [10–14].

However, when solving complex multimodal problems, ABC algorithm suffers from the following drawbacks [8]. (1) It is easily trapped into local minimums in the early generations, which leads to low population diversity in successive generations. (2) With the dimension increasing, the information exchange of each individual is limited in a random dimension, resulting in a slow convergence rate. (3) Due to the random selection of the neighbor bee and dimensions, food sources with higher fitness are not utilized, which influences the ability of global search.

Moreover, compared to the huge in-depth studies of other evolutionary and swarm intelligence algorithms, such as evolutionary algorithm (EA) [15–18] and particle swarm optimization (PSO) [19–23], how to improve the diversity of swarm or overcome the local convergence of ABC is still a challenging to the researchers in optimization domain.

In order to improve the performance of ABC on complex and high-dimensional problems, in this paper, a novel optimization algorithm called hierarchical ABC optimization (HABC) is proposed to extend the topology of original ABC algorithm from flat (one level) to hierarchical (multiple levels), which adopts multipattern cooperative evolutionary strategies.

It is noted that the hierarchical coevolutionary scheme has been incorporated in these intelligent algorithms. For instance, a hierarchical version of PSO was proposed in [24] on IEEE Trans on Evol Comp, in which a tree type hierarchy was incorporated in PSO. Chen et al. [25] proposed

a hierarchical swarm model where each subswarm evolved based on original PSO. Peng and Lu [26] presented a similar PSO framework, in which several swarms evolve in parallel with mutation operator. This paper further extends the above studies based on hierarchical frameworks with more complex strategies, such as variables decomposing approach, random grouping of variables, and cross and mutation operation. In particular, instead of simply employing original SI algorithm in parallel like other hierarchical algorithms, the HABC employing the variables decomposing strategy with random grouping technique divides the complex vectors into smaller components assigned to different subswarms, which achieves a significant improvement of solving high-dimensional problem.

In terms of neighborhood concept, Kennedy and Mendes [27] analyzed effects of different topologies-based PSO algorithms. In 2004, a fully informed PSO using topologies and index-based neighborhood was proposed in [28]. Recently, Qu et al. [29] developed a neighborhood-based mutation strategy and integrated it with various niching DE algorithms, in which various topology types were considered. A distance-based locally informed particle swarm model was proposed in [30], which eliminates the need to specify any niching parameter and enhance the fine search ability of PSO. Obviously, many topology patterns can be used in different levels of our mode. In this work, we employ the most common topology, ring type, as the main structure of the proposed model.

The proposed HABC model is inherently different from others in the following aspects.

(1) The cooperative coevolving approach based on divide-and-conquer strategy enhances the local search ability (exploitation); that is, by applying this method, the complex high-dimensional vectors can be decomposed into smaller components that are assigned to the lower hierarchy.

(2) The traditional evolution operators, namely, the crossover and mutation, are applied to interaction of different species instead of the traditional individual exchange between populations. In this case, the neighbor bees with higher fitness can be chosen to crossover and mutation, which effectively enhances the global search ability and convergence speed to the global best solution (exploration).

Extensive studies based on a suit of 20 benchmark functions (including both continuous and discrete cases) are conducted to evaluate the performance of HABC. For comparison purposes, we also implemented the particle swarm optimization algorithm (PSO), cooperative particle swarm optimization algorithm (CPSO), artificial bee colony algorithm (ABC), cooperative artificial bee colony algorithm (CABC), covariance matrix adaptation evolution strategy (CMA-ES), and HABC variants on these functions. The experimental results are encouraging: the proposed HABC algorithm achieved remarkable search performance in terms of solution accuracy and convergence rate on almost all benchmark functions.

The rest of the paper is organized as follows. Section 2 describes the canonical ABC algorithm. In Section 3, the proposed hierarchical artificial bee colony (HABC) model is given. Section 4 tests the algorithm on the benchmarks and illustrates the results. Finally, Section 5 outlines the conclusions.

2. Canonical ABC Algorithm

The ABC algorithm is a relatively new SI algorithm by simulating the foraging behaviors of honey bee swarm, initially proposed by Karaboga and further developed by Basturk and Li et al. [8, 9, 31, 32]. In ABC, the colony of artificial bees are classified as three types: employed bees, onlookers, and scouts. Employed bees exploit the specific food sources; meanwhile, they pass the food information to onlooker bees. Onlooker bees choose good food sources based on the received information and then further exploit the food near their selected food source. The employed bee will become a scout when its food source has been abandoned. The fundamental mathematic representations are listed as follows.

In initialization phase, a group of food sources are generated randomly in the search space using the following equation:

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

where $i = 1, 2, \ldots, \text{SN}$ and $j = 1, 2, \ldots, D$. SN is the number of food sources. D is the number of variables, that is, problem dimension. $x_j^{\min}$ and $x_j^{\max}$ are the lower upper and upper bounds of the jth variable, respectively.

In the employed bees' phase, the neighbor food source (candidate solution) can be generated from the old food source of each employed bee in its memory using the following expression:

$$v_{i,j} = x_{i,j} + \varphi\left(x_{i,j} - x_{k,j}\right), \tag{2}$$

where k is a randomly selected food source and must be different from i; j is a randomly chosen indexes; ϕ is a random number in range $[-1, 1]$.

In the onlooker bees' phase, a onlooker bee selects a food source depending on the probability value associated with that food source, and P_i can be calculated as follows:

$$P_i = \frac{\text{fitness}_i}{\sum_{j=1}^{\text{SN}} \text{fitness}_j}, \tag{3}$$

where fitness_i is the fitness value of jth solution.

In scout bees' phase, if a food source cannot be improved further through a predetermined cycle (called "limit" in ABC), the food source is supposed to be abandoned. The employed bee subsequently become a scout. A new food source will be produced randomly in the search space using (1).

The employed, onlooker, and scout bees' phases will recycle until the termination condition is met.

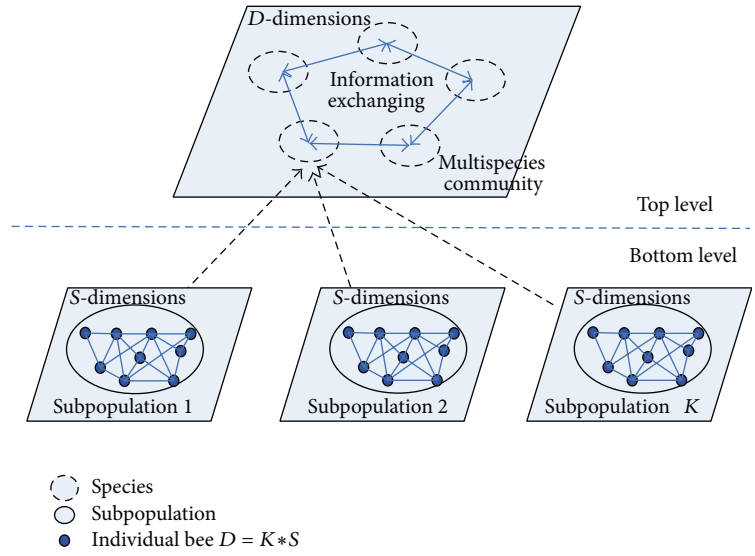

Figure 1: Hierarchical optimization model.

3. Hierarchical Artificial Bee Colony Algorithm

The HABC integrates a two-level hierarchical coevolution scheme inspired by the concept and main ideas of multipopulation coevolution strategy and crossover and mutation operation. The flowchart of the HABC is shown in Figure 1. It includes four key strategy approaches: variables decomposing approach, random grouping of variables, background vector calculating approach, and crossover and mutation operation, which is presented as follows.

3.1. Hierarchical Multipopulation Optimization Model. As described in Section 2, we can see that the new food source is produced by a perturbation coming from a random single dimension in a randomly chosen bee. This causes an individual to may have discovered some good dimensions, while the other individuals that follow this bee are likely to choose worse vectors in D-dimensions and abandon the good ones. On the other hand, when solving complex problems, single population-based artificial bee algorithms suffer from the following drawback of premature convergence at the early generations.

Hence, the HABC contains two levels, namely, the bottom level and top level, to balance exploring and exploiting ability. In Figure 1, in the bottom level, with the variables decomposing strategy, each subpopulation employs the canonical ABC method to search the part-dimensional optimum in parallel; that is, in each iteration, K subpopulations in the bottom level generate K best solutions, which are constructed into a complete solution species that are updated to the top level. In the top level, the multispecies community adopts the information exchange mechanism based on crossover operator, by which each species can learn from its neighborhoods in a specific topology. The vectors decomposing strategy and information exchange crossover operator can be described in detail as follows.

3.2. Variables Decomposing Approach. The purpose of this approach is to obtain finer local search in single dimensions inspired by the divide-and-conquer approach. Notice that two aspects must be analyzed: (1) how to decompose the whole solution vector, and (2) how to calculate the fitness of each individual of each subpopulation. The detailed procedure is presented as follows.

Step 1. The simplest grouping method is permitting a D-dimensional vector to be split into K subcomponents, each corresponding to a subpopulation of s-dimensions, with M individuals (where $D = K * s$). The jth subpopulation is denoted as P_j, $j \in [1 \cdots K]$.

Step 2. Construct complete evolving solution *Gbest*, which is the concatenation of the best subcomponents' solutions P_j by following:

$$Gbest = \left(P_1 \cdot g, P_2 \cdot g, P_j \cdot g \cdots P_K \cdot g\right), \tag{4}$$

where $P_j \cdot g$ represents the personal best solution of the jth subpopulation.

Step 3. For each component P_j, $j \in [1 \cdots K]$, do the following.

(a) At employed bees' phase, for each individual X_i, $i \in [1 \cdots M]$, replace the ith component of the *Gbest* by using the ith component of individual X_i. Caculate the new solution fitness: $f(newGbest(P_1 \cdot g, P_2 \cdot g, X_i, \ldots P_k \cdot g))$. If $f(newgbest) < f(Gbest)$, then *Gbest* is replaced by *newGbest*.

(b) Update X_i positions by using (8).

(c) At onlooker bees' phase, repeat (a)-(b).

Step 4. Memorize the best solution achieved so far, and compare the best solution with *Gbest* and memorize the best one.

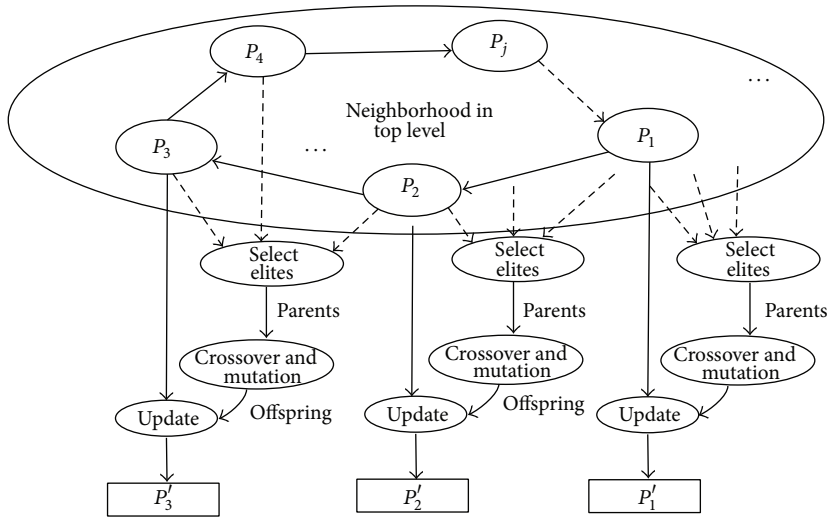

Figure 2: The information exchange mechanism based on crossover operator.

3.2.1. Random Grouping of Variables. To increase the probability of two interacting variables allocated to the same subcomponent, without assuming any prior knowledge of the problem, according to the random grouping of variables proposed by [21], we adopt the same random grouping scheme by dynamically changing group size. For example, for a problem of 100 dimensions, we can define that

$$G = \{2, 5, 10, 20, 100\},$$
$$K \subset G. \tag{5}$$

Here, if we randomly decompose the D-dimensional object vector into K subcomponents at each iteration (i.e., we construct each of the K subcomponents by randomly selecting S-dimensions from the D-dimensional object vector), the probability of placing two interacting variables into the same subcomponent becomes higher, over an increasing number of iterations [21].

3.3. The Information Exchange Mechanism Based on Crossover Operator between Multispecies. In the top level, we adopt crossover operator with a specific topology to enhance the information exchange between species, in which each species P_j can learn from its symbiotic partner in the neighborhood. The key operations of this crossover procedure are described in Figure 2.

Step 5 (Select elites to the best-performing list (BPL)). First, a set of competent individuals from current species P_j's neighborhood (i.e., ring topology) is selected to construct the best-performing list (BPL) with higher fitness that has larger probability to be selected. The size of BPL is equal with the number of current species P_j. These individuals of BPL are regarded as elites. The selection operation tries to mimic the maturing phenomenon in nature, where the generated offspring will become more suitable to the environment by using these elites as parents.

Table 1: Eight versions of HABC.

Algorithms	Selection methods	Crossover mode
HABC_SB	Select the best individuals	Single point
HABC_SW	Select the worst individuals	Single point
HABC_SM	Select medium individuals	Single point
HABC_SR	Select random individuals	Single point
HABC_AB	Select the best individuals	Arithmetic point
HABC_AW	Select the worst individuals	Arithmetic point
HABC_AM	Select medium individuals	Arithmetic point
HABC_AR	Select random individuals	Arithmetic point

Step 6 (Crossover and mutation between species). To produce well-performing individuals, parents are selected from the BPL's elites only for the crossover operation. To select parents effectively, the tournament selection scheme is used. Firstly, two enhanced elites are selected randomly, and their fitness values are compared to select the elites. Then, the one with better fitness value is viewed as parent. Then, another parent is selected in the same way. Two offsprings are created by performing crossover on the selected parents. This paper adopts two representative crossover methods: single-point crossover and arithmetic crossover. For single-point crossover, the offspring is produced as the traditional GA crossover method [33]. For arithmetic crossover method, the offspring is produced by (6) as follows:

$$s_{\text{new}} = \text{rand}(0, 1) \times \text{parent1} + \text{rand}(0, 1) \times \text{parent2}, \tag{6}$$

where S_{new} is newly produced offspring, parent1 and parent2 are randomly selected from BPL.

Step 7 (Update with greedy selection strategy). Not all current species are replaced by the elites from BPL; we set a selecting rate CR to determine the replaced individuals. Assuming that species size of P_j is M, then the replaced individuals number is $M * \text{CR}$. For the selected individual,

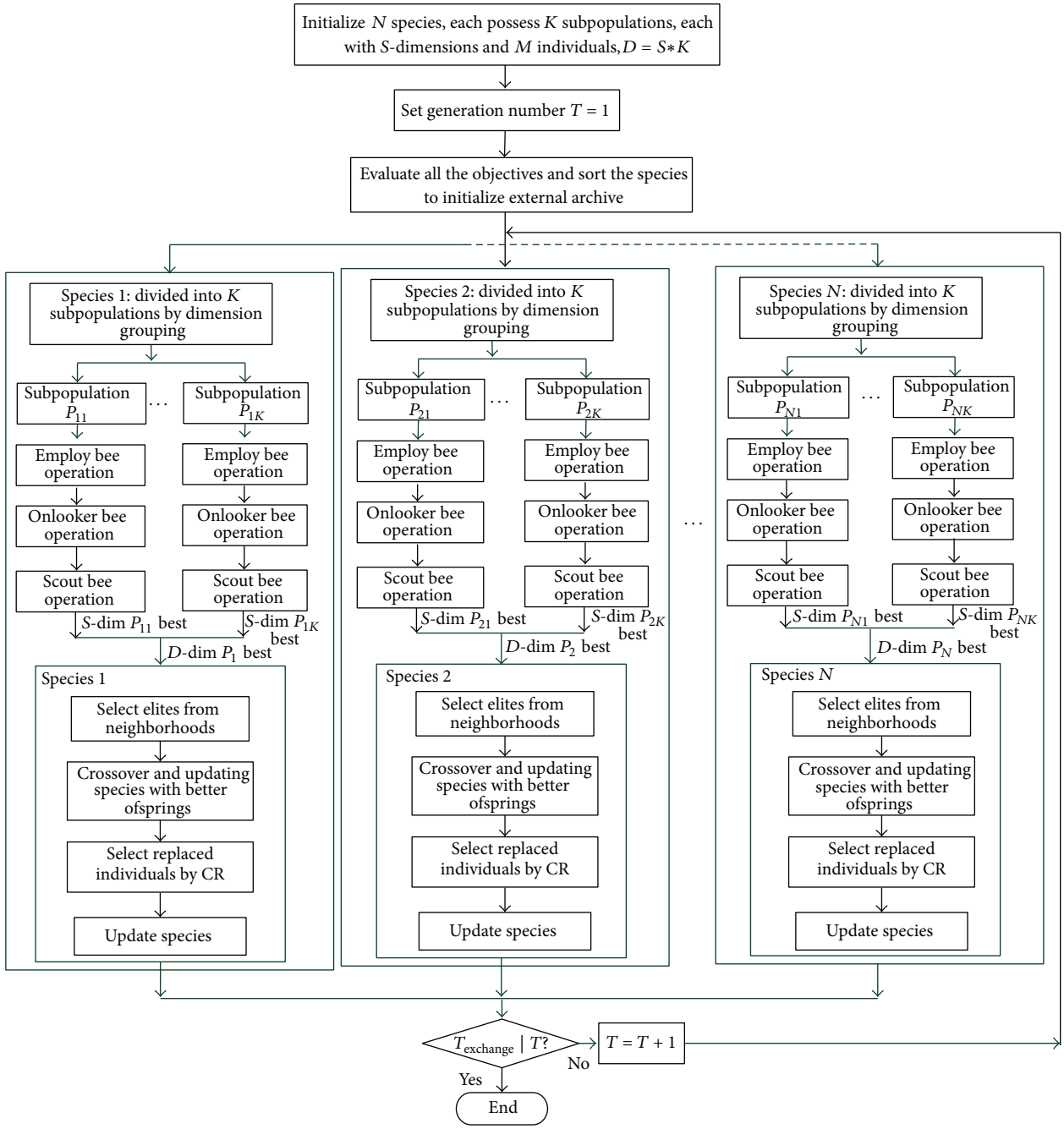

Figure 3: Flowchart of the HABC algorithm.

S_j, the newly produced offspring S_{new} is then compared with S_j, applying a greedy selection mechanism, in which the best one is remained. We can choose four selecting approaches: selecting the best individuals (i.e., $M * \text{CR}$ individuals), a medium level of individuals, the worst individuals, and random individuals.

Hence, there are eight HABC variants according to different crossover and selection approaches, as listed in Table 1. In the next experiments, we will study the effect of different crossovers, selection modes, and CRs. With applying the effective social learning strategy, the information exchange between species is enhanced and the food sources with higher fitness are fully utilized.

In summary, in order to facilitate the below presentation and test formulation, we define a unified parameters for HABC model in Table 2. According to the process description as mentioned above, the flowchart of HABC algorithm is summarized in Figure 3, while the pseudocode for HABC algorithm is presented in Algorithm 1.

HABC algorithm

Set $t := 0$;
INITIALIZE.
Randomly divide the whole population into N species (P_i) each possesses K sub-populations (P_{ij}), each possesses M bees;
Randomize P_{ij}'s D-dimensions food source positions $P_{ij} \cdot X_k$; $i \in [1 : N]$, $j \in [1 : K]$, $k \in [1 : M]$.
Each sub-population P_{ij} with S dimensions (where S is randomly chosen from a set G, and $D = K * S$).
WHILE (the termination conditions are not met)
for each species i, $i = [1 : N]$
Initialize D-dimensions complete vector $Gbest = (P_{i1} \cdot g, P_{i2} \cdot g, \ldots, P_{il-1} \cdot g, z, P_{il+1} \cdot g, \ldots, P_{iK} \cdot g)$,
which consists of the S-dimensions
best solution $P_{il}\widehat{y}$.
Randomly all D dimension indices;
WHILE (the termination conditions are not met)
for each sub-population P_{ij}, $j = [1 : K]$ do
repeat
Employed Bees' Phase:
For each employed bee $P_{ij} \cdot x_{k:}$
Produce a new solution by using (2)
Evaluate the new solution
Apply Greedy selection choosing the better solution
end
Calculate the probability values Pi for the solution by using (2)
Onlooker Bees' Phase:
for each employed bee $P_{ij} \cdot x_k$
Probabilistically choose a solution according to P_i
Produce a new solution by (2)
Evaluate the new solution
Apply Greedy selection choosing the better solution
end
Re-initialize solutions not improved for *Limit* cycles
Memorize the best solution $P_{ij} \cdot x$
for each individual of $P_{ij} \cdot x_k$, $k = [1 : M]$ do
Place best solution in the complete solution *newGbest by:*
$newGbest = (P_{i1} \cdot g, P_{i2} \cdot g, \ldots, P_{ij} \cdot x_k, \ldots, P_{iK} \cdot g)$
Update complete solution if it improves:
If $(newGbest) < f\ (Gbest))$
Then $P_{ij} \cdot g = P_{ij} \cdot x_k$
end
end
end WHILE
Select elites form neighborhood of P_i
BPL = the top M best individuals of the ring topology $\{P_{i-1}, P_i, P_{i+1}\}$
Crossover and Mutation P_i' by (4)
Update P_i with applying Greedy selection mechanism from P_i'
end
find the global best solution *gbest* from the whole population P
memorize the best solution of each P_{ij}
Set $t := t + 1$;
end WHILE

Algorithm 1: Pseudocode for the HABC algorithm.

4. Experimental Study

In the experimental studies, to fully evaluate the performance of the HABC algorithm fairly, we employ a set of 20 benchmark functions as listed in the appendix, which can be classified as basic continuous benchmarks (f_1–f_8), CEC2005 benchmarks (f_9–f_{15}), and discrete benchmarks ($f_{16} \sim f_{20}$) [34]. The number of function evaluations (FEs) is adopted as the time measure criterion substitute the number of iterations.

4.1. Experimental Settings. Eight variants of HABC based on different crossover methods and CR values were executed with six state-of-the-art EA and SI algorithms for comparisons: artificial bee colony algorithm (ABC) [4], cooperative

Table 2: Parameters of the HABC.

HABC = $(N, M, P_{ij}, C, \mathrm{CR}, T, O, S)$	
N	The number of species in top level
K	Subpopulation dividing D-dimensions into S-dimensions
M	Subpopulation size
S	Corresponding to a population of S-dimensions, where $S = D/K$
D	Dimensions of optimization problem
i	Top level population's (species) ID counter from 1 to N
j	Button level population's ID counter from 1 to K
P_{ij}	The ith population (of the jth species)
CR	Selection rate for replacing the offspring to the selected individuals
T	The hierarchical interaction topology of HABC
O	The objective optimization goals

artificial bee colony algorithm (CABC) [35], canonical PSO with constriction factor (PSO) [36], cooperative PSO (CPSO) [20], standard genetic algorithm (GA) [33], and covariance matrix adaptation evolution strategy (CMA-ES) [37].

CABC adopts the similar cooperative approach into original ABC algorithm. CPSO is a cooperative PSO model, cooperatively coevolving multiple PSO subpopulations. GA is the classical stochastic search technique mimicking the process of natural selection; the principle of CMA-ES is to apply the information of successful search steps to adjust the covariance matrix of the mutation distribution within an iterative procedure.

The population size and total generation number of all involved algorithms in this experiment are set as 50 and 100000. For the continuous testing functions used, the dimensions are all set as 100D. For the five discrete testing functions, the dimensions are set as Goldberg-30D, Bipolar-60D, Mulenbein-120D, Clerc's problem 1-120D, and Clerc's problem 2-120D.

According to the original literatures about the control parameters for the other algorithms involved, in continuous optimization experiment, the initialization conditions of CMA-ES are the same as in [37], and the number of offspring candidate solutions generated per time step is $\lambda = 4\mu$; for ABC and CABC, the limit parameter is set to be SN×D, where D is the objective function dimension and SN is the employed bees number. The split factor for CABC and CPSO is equal to the dimensions [20, 35]. For PSO and CPSO, the learning rates c_1 and c_2 were both set as 2.05 and the constriction factor $\chi = 0.729$.

In discrete optimization experiment, HABC is compared with the binary PSO version, ABC and GA. GA employs single point crossover operation (i.e., crossover rate is 0.8 and mutation rate is 0.01). For discrete HABC and discrete ABC, the update process used in the employ and onlooker stages is changed based on (1). The learning factor of producing new solution is calculated using (7). Then, the position update equation is defined by (8) for discrete problems. Discrete HABC variant adopts single-point crossover method. Consider

$$q_{ij} = \left|\phi_{ij}\left(x_{kj} - x_{ij}\right)\right|, \tag{7}$$

$$v_{ij} = \begin{cases} \overline{x_{ij}} & \text{if } \mathrm{rand}_j \le q_{ij} \\ x_{ij} & \text{otherwise.} \end{cases} \tag{8}$$

4.2. Parameter Sensitivity Analysis of HABC

4.2.1. Effects of Species Number N. The species number of the top level in HABC needs to be tuned. Six continuous benchmark functions, Sphere 100D, Rosenbrock 100D, Rastrigin 100D, Schwefel 100D, Ackley 100D, and Griewank 100D and three discrete benchmark functions—Goldberg 30D, Bipolar 30D, and Clerc problem 1 120D—are employed to investigate the impact of this parameter. Set CR equal to 1 and all the functions run 30 sample times. As shown in Figure 4, it is visible that our proposed ABC got faster convergence rate and better optimal solutions on the involved test functions with increased N. However, the performance improvement is not very remarkable using this parameter.

4.2.2. Choices of Crossover Mode and CR Value. The basic benchmark functions (f_1–f_8) are adopted to evaluate the performance of HABC variants with different crossover modes andCRs. Four discrete benchmark functions (f_{16}–f_{20}) are used to test the discrete versions of HABC_AB, HABC_AW, HABC_AM, and HABC_AR, which adopt single-point crossover mode.

Firstly, when CR is fixed at 0.2, all the functions are implemented for 30 times. From Table 3, we can observe that HABC_AW outperformed other variants on seven functions, except f_2, while HABC_AM and HABC_AR get nearly the same results as HABC_AW on four functions. From the discrete functions results, HABC_SB markedly outperformed other variants, except f_{18}. According to the rank listed in Table 3, the performances of the continuous algorithms involved are ordered as HABC_AW > HABC_AB > HABC_AR > HABC_AM > HABC_SR > HABC_SM > HABC_SB > HABC_SW and for the discrete versions are ordered as HABC_SB > HABC_SM > HABC_SW > HABC_SR.

Then, the HABC_AW for continuous functions and HABC_SB for discrete functions are implemented to determine CR value. Form Tables 4 and 5, we can find that HABC variant with CR equal to 1 performs best on four functions among all five functions while CR equal to 0.05 gets best result on one function. According to the results with different CRs, we chose CR equal to 1 as an optimal value for the next experiments.

4.2.3. Effects of Dynamically Changing Group Size K. Obviously, the choice of value for split factor K (i.e., subpopulation number) had a significant impact on the performance of the proposed algorithm. In order to vary K during a run, we defined $S = \{2, 5, 10, 50, 100\}$ for 100D function optimization and set K randomly to choose one element of S. Then, the variant of HABC with dynamically changing K is compared

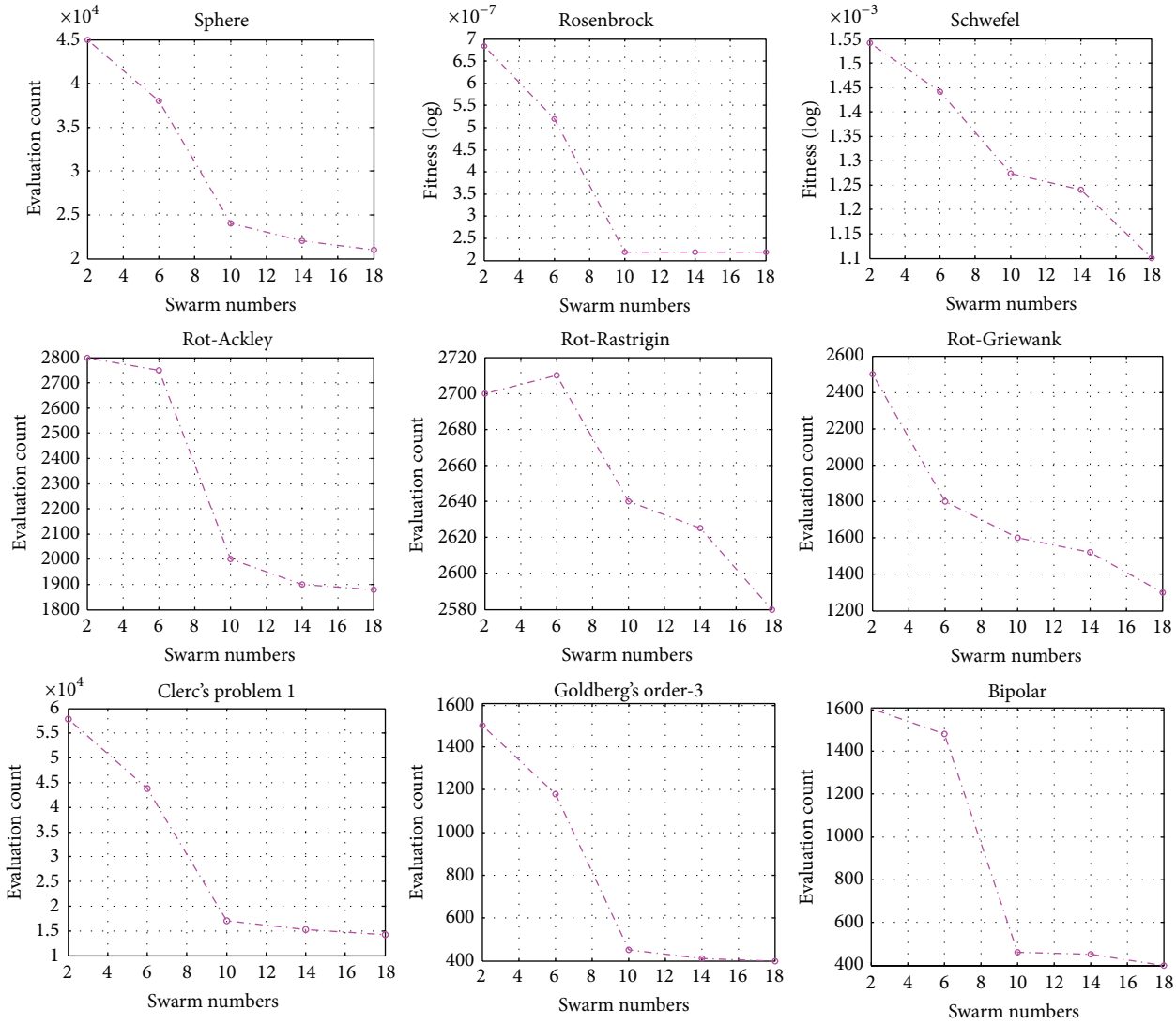

Figure 4: HABC's results on nine-test functions with different species numbers N.

with that with fixed split number on four benchmark functions for 30 sample runs. From the results listed in Table 6 and Figure 5, we can observe that the performance is sensitive to the predefined K value. HABC, with the dynamically changing K, consistently obtained superior performance to other variants except f_2. Moreover, it is not very easy to obtain the prior knowledge about the optimal s value of the most real-world cases, so the random grouping scheme can be a suitable solution.

4.3. Comparing HABC with Other State-of-the-Art Algorithms on Benchmark Problems. To validate the effectiveness of the proposed algorithm, the basic benchmark functions (f_1–f_8) and CEC2005 functions (f_9–f_{15}) are employed [38]. HABC (adopting HABC_AW for continuous problems, CR = 1) is tested on a set of benchmark functions (f_1~f_{15}) in comparison with CABC, CPSO, CMA-ES, ABC, PSO, and GA algorithms. Table 7 demonstrates the corresponding results of mean and standard deviation for each algorithm on f_1 ~ f_{15} with 100 dimensions. Figure 6 shows the convergence characteristics in terms of the best mean run of each algorithm for $f_1 \sim f_{15}$ with 100 dimensions.

4.3.1. Results on Basic Benchmark Continuous Functions. On the unimodal basic benchmark functions (f_1–f_4), from Table 7 and Figures 6(a)–6(d), HABC converged faster than all other algorithms. HABC was able to consistently find the minimum to functions f_1, f_2, and f_3 within 100000 FEs. Statistically, HABC has significantly superior performance on continuous unimodal functions $f_1 \sim f_3$. On f_4, HABC, CABC, and CMA-ES have almost the same average value and CMA-ES is a little better. According to the rank in Table 7, the performance order of the algorithms involved is CMA-ES > HABC > CABC > ABC > CPSO > PSO > GA.

On the multimodal functions ($f_5 \sim f_8$), from Table 7 and Figures 6(e)–6(i), it is visible that, on most functions, HABC algorithm markedly outperforms other algorithms. For example, HABC quickly finds the global minimum on

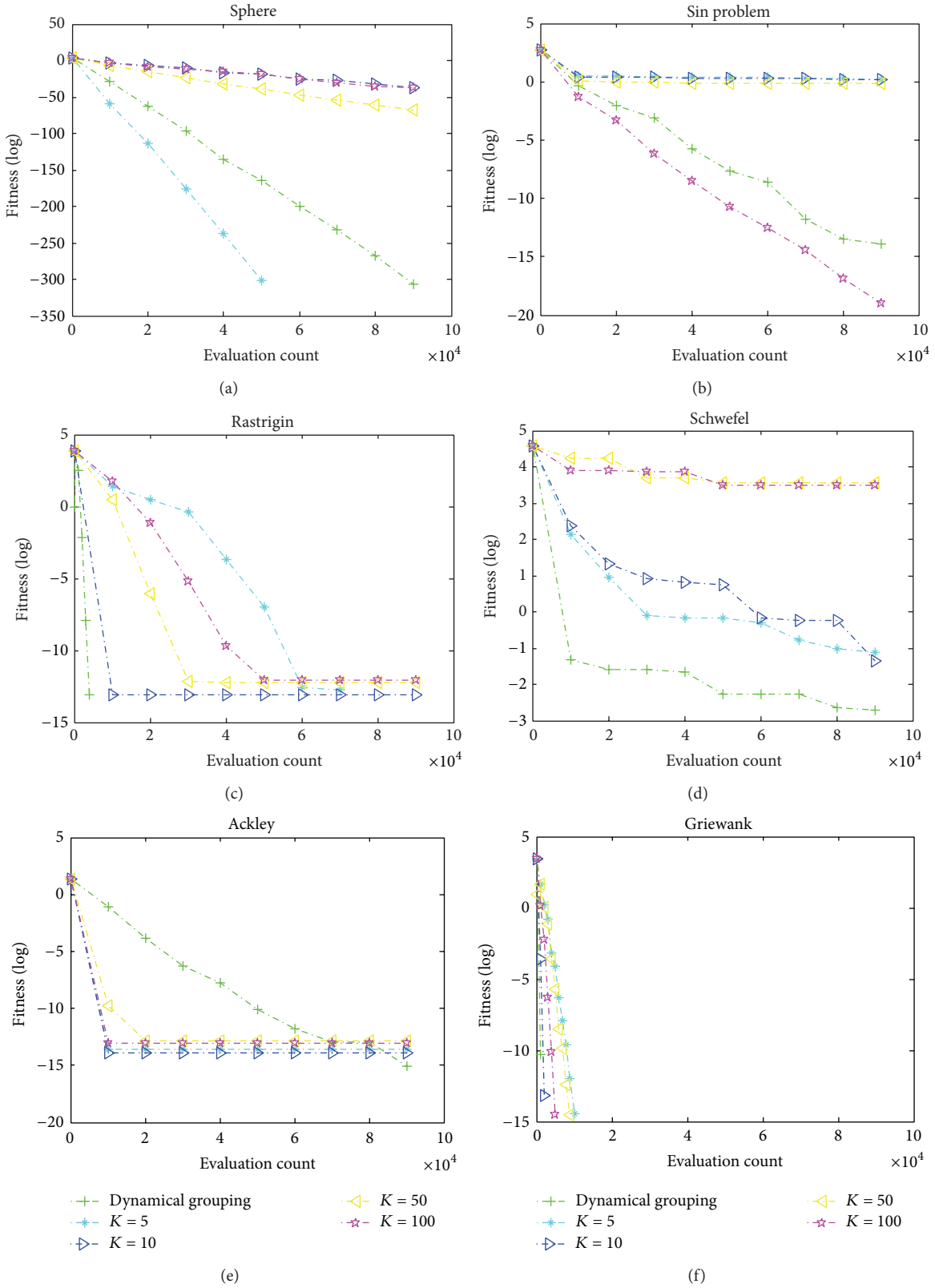

FIGURE 5: The median convergence results of 100D continuous functions. (a) Sphere function. (b) Sin problem function. (c) Rastrigin function. (d) Schwefel function. (e) Ackley function. (f) Griewank function.

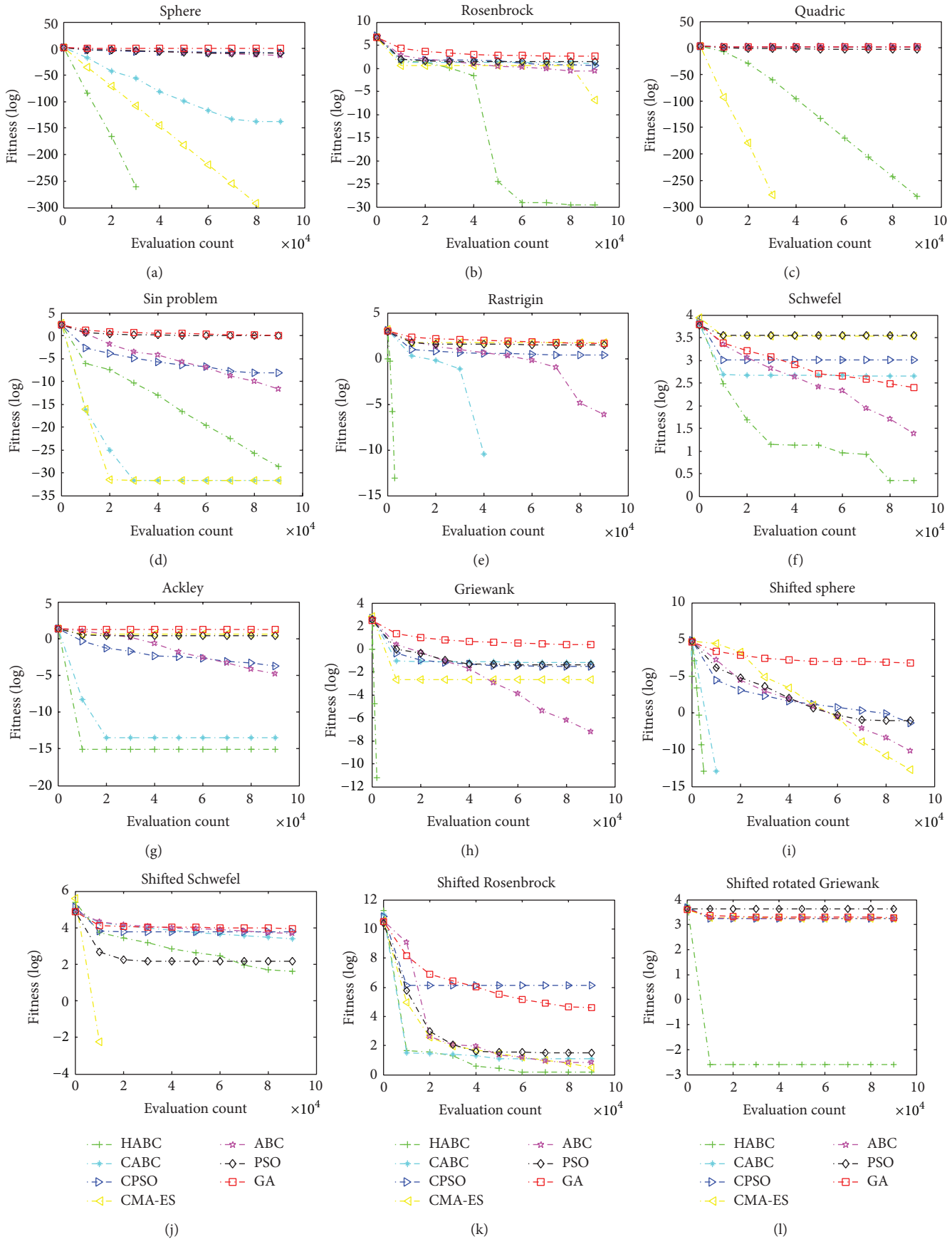

Figure 6: Continued.

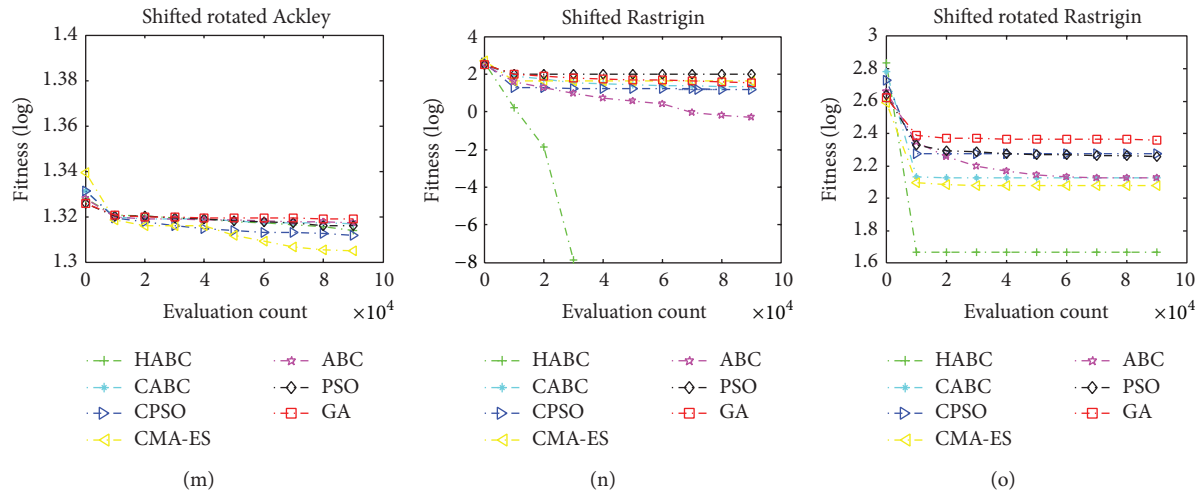

Figure 6: The median convergence results of 100D continuous functions. (a) Sphere function. (b) Rosenbrock function. (c) Quadratic function. (d) Sin problem function. (e) Rastrigin function. (f) Schwefel function. (g) Ackley function. (h) Griewank function. (i) Shifted sphere function. (j) Shifted Schwefel function. (k) Shifted Rosenbrock function. (l) Shifted rotated Griewank function. (m) Shifted rotated Ackley function. (n) Shifted Rastrigin function. (o) Shifted rotated Rastrigin function.

function f_5 and f_8, and CABC also can consistently find the minimum of f_5 within relatively more FEs while other algorithms perform poorer. On f_6 and f_7, the HABC and CABC obtain similar performance. This can be explained that the multipopulation cooperative coevolution strategy integrated by HABC and CABC enhance the local search ability, contributing to their better performances in the multimodal problems. According to the rank values in Table 7, the performance order of the algorithms tested is HABC > CABC > ABC > CPSO> CMA-ES > PSO > GA.

4.3.2. Results on CEC2005 Continuous Functions. Seven shifted and rotated functions $f_9 \sim f_{15}$ are viewed as the most difficult functions to tackle. From Table 7, HABC outperformed CMA-ES on five out of the seven functions, except f_{10} and f_{13}. CMA-ES also outperformed CABC on six functions, except f_{14}. HABC can find the global optimum for f_9 and f_{14} within 10000 Fes, because HABC balances the exploration and exploitation by decomposing high-dimensional problems and using crossover and mutation operations to maintain its population diversity, which is a key contributing factor. On the other hand, CMA-ES converged extremely fast. However, CMA-ES either converged very fast or tended to be trapped into local minima very quickly, especially on multimodal functions. According to the rank values, the performance order of the algorithms involved is HABC > CMA-ES > CABC > ABC > CPSO > PSO > GA.

4.3.3. Results on Discrete Functions. In this experiment, the HABC_SB version (CR = 0.1) is employed on a set of discrete benchmark functions $f_{16} \sim f_{20}$. The results obtained by HABC_SB, ABC, GA, and PSO on each function are listed in Table 8, consisting of mean and standard deviations found over 30 runs. Figure 7 shows the search progress of the average values found by the four algorithms over 30 runs on $f_{16} \sim f_{20}$. From the results in Table 8, we can observe that HABC gets the best means, and converges faster than other algorithms on all discrete cases. According to the rank values presented in Table 9; the search performance order of the algorithms involved is HABC > PSO > ABC > GA.

4.3.4. Algorithm Complexity Analysis. Algorithm complexity analysis is presented briefly as follows. If we assume that the computation cost of one individual in the HABC is *Cost_a*, the cost of the crossover operator is *Cost_c* and the total computation cost of HABC for one generation is $N * K * M * Cost\_a + N * Cost\_c$. However, because the heuristic algorithms used in this paper cannot ensure comprehensive convergence, it is very difficult to give a brief analysis in terms of time for all algorithms. Through directly evaluating the algorithmic time response on different objective functions, the average computing time in 30 sample runs of all algorithms is given in Figure 8. From the results in Figure 8, it is observed that the HABC takes the most computing time in all compared algorithms and its time increasing rate is the highest one. This can be explained that the multipopulation cooperative coevolution strategy integrated by HABC enhanced the local search ability at cost of increasing the computation amount. In summary, it is concluded that, compared with other algorithms, the HABC requires more computing time to achieve better results.

5. Conclusion

In this paper, we propose a novel hierarchical artificial bee colony algorithm (HABC), extending single-level to multi-level aiming to improve the performance of solving complex and high-dimensional problem. The concept and main idea is extending single artificial bee colony (ABC) algorithm to hierarchical and cooperative mode by combining the multipopulation cooperative coevolution approach based on vector decomposing strategy and the comprehensive learning

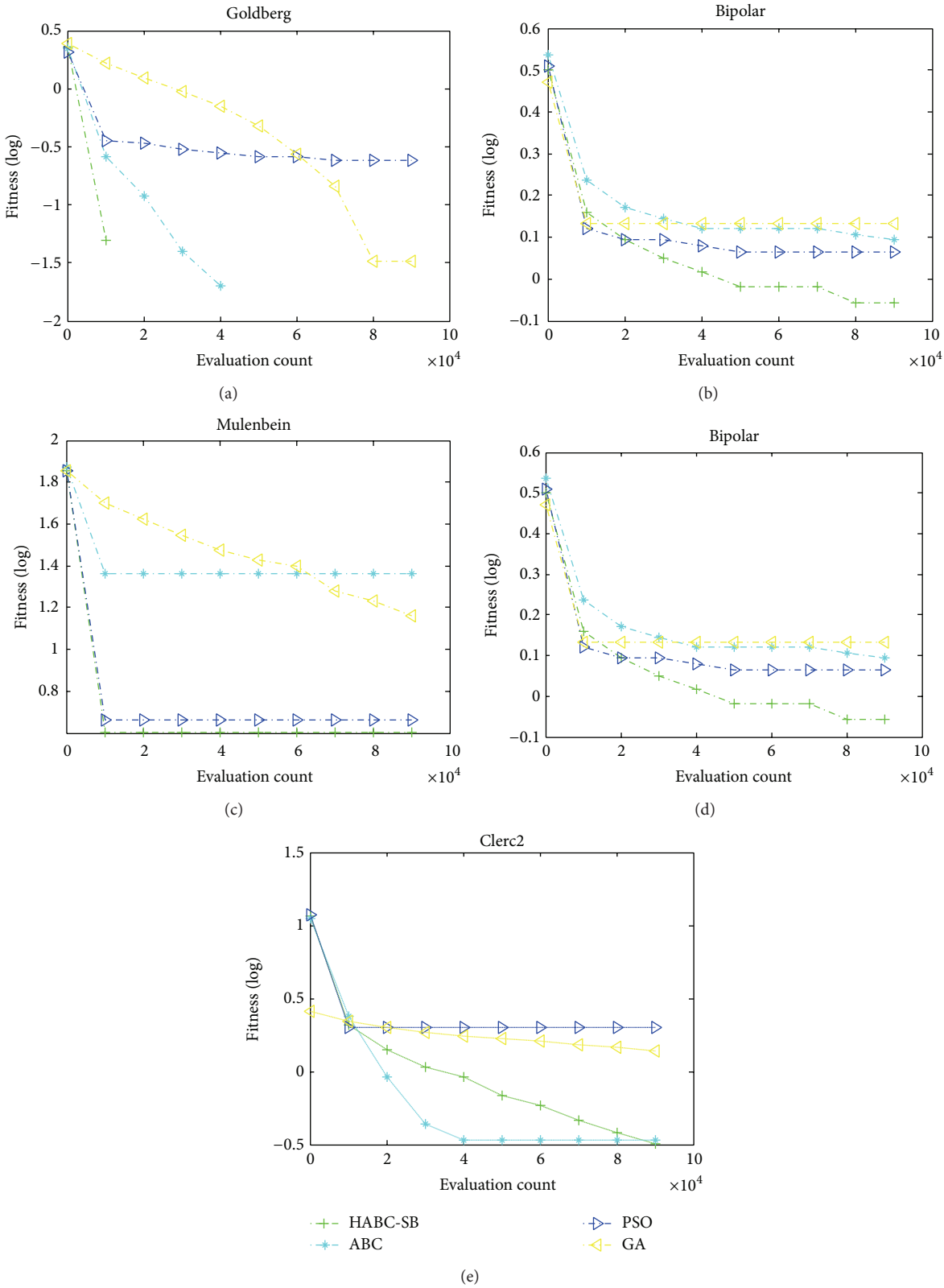

FIGURE 7: The median results of discrete functions. (a) Goldberg's order-3. (b) Bipolar order-6. (c) Mulenbein's order-5. (d) Clerc's order-3 problem 1. (e) Clerc's order-3 problem 2.

Table 3: Results of all algorithms on benchmark continuous functions f_1–f_8 and discrete functions f_{16}–f_{19}. In bold are the best results.

Function		HABC_SB	HABC_SW	HABC_SM	HABC_SR	HABC_AB	HABC_AW	HABC_AM	HABC_AR
f_1	Mean	3.84e − 20	9.87e − 4	2.14e − 14	1.25e − 7	4.66e − 144	**4.52e − 300**	5.84e − 301	5.67e − 259
	Std	5.44e − 20	1.390e − 3	3.03e − 14	1.77e − 7	6.59e − 144	**0**	**0**	**0**
	Rank	**5**	**8**	**6**	**7**	**4**	**1**	**2**	**3**
f_2	Mean	2.12e − 002	2.15e + 001	1.0e + 001	7.54	3.96e − 005	1.26e − 004	2.75	**1.10e − 005**
	Std	5.03e − 002	1.54e + 001	1.39e + 001	1.26e + 001	7.43e − 005	3.38e − 004	8.69	**1.92e − 005**
	Rank	**4**	**8**	**7**	**6**	**2**	**3**	**5**	**1**
f_3	Mean	2.20e − 005	2.36e − 018	2.99e − 007	1.21e − 009	3.57e − 148	**1.07e − 282**	4.02e − 282	6.57e − 278
	Std	2.65e − 005	4.74e − 018	5.99e − 007	2.41e − 009	7.14e − 148	**0**	0	0
	Rank	**8**	**5**	**7**	**6**	**4**	**1**	**2**	**3**
f_4	Mean	1.77e − 004	4.11e − 007	8.44e − 007	**0**	**0**	**0**	**0**	**0**
	Std	2.27e − 004	5.66e − 007	1.12e − 006	**0**	**0**	**0**	**0**	**0**
	Rank	**8**	**6**	**7**	**1**	**1**	**1**	**1**	**1**
f_5	Mean	1.74e − 004	9.39	3.59e − 006	9.12e − 001	**0**	**0**	**0**	**0**
	Std	3.87e − 004	11.47	6.14e − 006	8.61e − 001	**0**	**0**	**0**	**0**
	Rank	**6**	**8**	**5**	**7**	**1**	**1**	**1**	**1**
f_6	Mean	5.01e − 004	3.84e − 002	3.87e − 004	3.84e − 004	4.14e − 004	**3.81e − 004**	3.99e − 004	3.82e − 004
	Std	2.86e − 004	5.15e − 002	7.92e − 006	3.42e − 006	3.99e − 005	6.43e − 012	2.54e − 012	0
	Rank	**7**	**8**	**5**	**4**	**6**	**1**	**2**	**3**
f_7	Mean	1.77e − 003	1.21	6.83e − 005	2.78e − 001	**8.88e − 016**	**8.88e − 016**	**8.88e − 016**	**8.88e − 16**
	Std	5.12e − 003	1.93	2.13e − 004	8.24e − 001	**0**	**0**	**0**	**0**
	Rank	**6**	**8**	**5**	**7**	**1**	**1**	**1**	**1**
f_8	Mean	1.76e − 004	4.10e − 007	8.44e − 007	**0**	**0**	**0**	**0**	**0**
	Std	2.27e − 004	5.80e − 007	1.19e − 006	**0**	**0**	**0**	**0**	**0**
	Rank	**8**	**6**	**7**	**1**	**1**	**1**	**1**	**1**
Total Rank		**52**	**57**	**49**	**39**	**20**	**10**	**15**	**14**
f_{16}	Mean	**0.12**	0.3	0.34	0.26				
	Std	**0.10**	0.19	0.09	0.16				
	Rank	**1**	**3**	**4**	**2**				
f_{17}	Mean	**0.07**	0.25	0.21	0.26				
	Std	**0.08**	0.18	0.73	0.13				
	Rank	**1**	**3**	**2**	**4**				
f_{18}	Mean	1.76e − 004	4.11e − 007	8.44e − 007	**0**				
	Std	2.22e − 004	5.81	1.19	**0**				
	Rank	**4**	**2**	**3**	1				
f_{19}	Mean	**1.94**	**3.61**	2.72	3.86				
	Std	**0.59**	**0.95**	0.90	1.10				
	Rank	**1**	**3**	**2**	**4**				
Total Rank		**7**	**11**	**11**	**11**				

method. At the same time, the discrete version of HABC is presented in this paper. We demonstrate that the hierarchical cooperative framework proposed is useful to improve ABC's performance to tackle high-dimensional optimization problems (up to 100 dimensions).

The effects of using random grouping scheme has shown that the performance is sensitive to the grouping number, and without any prior knowledge, the random grouping can be a suitable solution. Subsequently, a group of appropriate parameters consisting of crossover modes, selection methods, and CRshas been determined for both continuous and discrete optimizations, respectively.

Then, HABC was compared with other SI and EA algorithms on a set of 20 benchmark functions (including

Table 4: Results of HABC_AW on benchmark functions f_1–f_8 with different CR. In bold are the best results.

Function		0.05	0.1	0.2.	0.4	0.6	1
f_1	Mean	1.69e − 109	3.74e − 215	4.25e − 290	0 (64800)	0 (56000)	**0 (36000)**
	Std	3.79879e − 109	0	0	0	0	**0**
f_2	Mean	2.17e − 004	1.16e − 006	1.08e − 005	5.50	4.26e − 006	**4.35e − 007**
	Std	4.07e − 004	2.1e − 006	8.15e − 006	12.30	6.38e − 006	**5.69e − 007**
f_3	Mean	2.36e − 101	2.99e − 194	1.71e − 267	0 (69700)	0 (49800)	**0 (38000)**
	Std	4.74e − 101	0	0	0	0	**0**
f_4	Mean	**1.93e − 028**	4.46e − 028	4.97e − 028	5.14e − 026	7.62e − 25	7.87e − 022
	Std	**2.03e − 028**	8.01e − 028	3.44e − 028	5.77e − 026	5.55e − 025	1.34e − 021
f_5	Mean	0 (20000)	0 (13500)	0 (7900)	0 (6860)	0 (4400)	**0 (3100)**
	Std	0	0	0	0	0	**0**
f_6	Mean	**3.84e − 004**	3.81e − 004	3.81e − 004	3.96e − 004	6.11e − 004	5.79e − 004
	Std	**1.25e − 012**	1.62e − 012	8.42e − 013	3.22e − 005	3.43e − 004	2.84e − 004
f_7	Mean	8.88e − 016 (22000)	8.88e − 016 (21500)	8.88e − 016 (15400)	8.88e − 016 (14220)	8.88e − 016 (13800)	**8.88e − 016 (12500)**
	Std	0	0	0	0	0	**0**
f_8	Mean	0 (19000)	0 (15800)	0 (13000)	0 (9800)	0 (6100)	**0 (5400)**
	Std	**0**	0	0	0	0	**0**

Table 5: Results of HABC_SB on benchmark functions f_{16}–f_{20} with different CR. In bold are the best results.

Function		0.05	0.1	0.2	0.4	0.6	1
f_{16}	Mean	5.11e − 001	**3.00e − 002**	9.00e − 002	1.00e − 001	2.6e − 001	1.6e − 001
	Std	1.61e − 001	**6.03e − 002**	1.03e − 001	8.32e − 002	1.68e − 001	2.11e − 001
f_{17}	Mean	2.00e − 002	**0**	2.00e − 002	1.50e − 001	2.61e − 001	5.15e − 001
	Std	6.37e − 002	**0**	6.35e − 002	1.23e − 001	2.38e − 001	1.69e − 001
f_{18}	Mean	**4.86**	6.81	8.25	1.69e + 001	2.22e + 001	4.21e + 001
	Std	**2.34**	2.24	2.45	3.32	4.79	5.00
f_{19}	Mean	6.70	**1.69**	1.75	3.61	5.28	8.73
	Std	6.1e − 001	**5.11e − 001**	3.11e − 001	5.77e − 001	7.44e − 001	1.41
f_{20}	Mean	**5.34e − 001**	1.51	1.96	2.36	3.31	6.53
	Std	**2.25e − 002**	3.32e − 001	5.82e − 001	3.42e − 001	6.03e − 001	9.54e − 004

Table 6: Performance of HABC_AW on 100D f_1 and f_4–f_8 with the different grouping number K. In bold are the best results.

Function		$K \subset S$	$K = 5$	$K = 10$	$K = 50$	$K = 100$
f_1	Mean	**0 (91000)**	**0 (45000)**	2.00e − 080	3.70e − 043	6.56e − 042
	Std	**0**	**0**	1.03e − 001	6.31e − 043	1.27e − 041
f_4	Mean	1.10e − 014	1.47	1.56	6.50e − 001	**1.61e − 022**
	Std	1.37e − 014	3.12e − 001	2.47e − 001	1.1 3	**1.38e − 022**
f_5	Mean	**0**	**0**	7.25e − 014	5.39e − 013	8.22e − 013
	Std	**0**	**0**	1.45e − 013	2.32e − 013	1.33e − 013
f_6	Mean	**1.70e − 003**	7.69e − 002	4.45e − 002	3.23e + 002	1.99e + 002
	Std	**1.10e − 003**	1.11e − 001	2.71e − 002	5.37e + 002	3.94e + 002
f_7	Mean	**8.34e − 016**	2.51e − 014	1.33e − 014	1.13e − 013	7.71e − 014
	Std	**0**	5.25e − 015	1.25e − 014	2.42e − 014	2.03e − 015
f_8	Mean	**0**	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**	**0**

Table 7: Performance of all algorithms on benchmark continuous functions f_1–f_{15}.

Function		HABC	CABC	CPSO	CMA-ES	ABC	PSO	GA
f_1	Mean	**0**	9.87*e* − 138	2.14*e* − 014	**0**	1.93*e* − 015	8.52*e* − 010	3.84*e* − 001
	Std	**0**	1.390*e* − 3	1.93*e* − 137	**0**	8.59*e* − 016	1.83*e* − 009	1.50*e* − 001
	Rank	**1**	3	5	**1**	4	6	7
f_2	Mean	**0**	5.41	3.81	1.32*e* − 007	1.92*e* − 001	2.98*e* + 001	3.26*e* + 002
	Std	**0**	6.25	2.56	5.03*e* − 002	2.03*e* − 001	2.48*e* + 001	7.61*e* + 002
	Rank	**1**	5	4	2	3	6	7
f_3	Mean	**0**	3.80*e* + 001	2.66*e* + 001	**0**	2.75*e* + 001	2.67*e* − 003	8.55*e* + 001
	Std	**0**	8.47*e* + 001	3.65*e* + 001	**0**	4.79	4.45*e* − 003	3.76*e* + 001
	Rank	**1**	6	4	**1**	5	3	7
f_4	Mean	7.72*e* − 032	2.55*e* − 032	6.65*e* − 009	**2.34e − 032**	8.07*e* − 014	8.27*e* − 001	8.52*e* − 001
	Std	5.75*e* − 032	3.55*e* − 032	1.38*e* − 008	**8.75e − 034**	6.23*e* − 014	6.22*e* − 001	3.54*e* − 001
	Rank	3	2	5	**1**	4	7	6
	Total Rank1	**6**	**16**	**18**	**5**	**16**	**22**	**27**
f_5	Mean	**0**	**0**	2.58	3.61*e* + 001	7.65*e* − 009	3.42*e* + 001	5.29*e* + 001
	Std	**0**	**0**	1.55	8.31	5.76*e* − 009	154*e* + 001	4.52
	Rank	**1**	1	4	6	3	5	7
f_6	Mean	**2.19**	4.50*e* + 002	1.02*e* + 003	2.53*e* + 003	2.31*e* + 001	3.43*e* + 003	1.99*e* + 002
	Std	**4.99**	1.00*e* + 003	1.06*e* + 002	3.98*e* + 002	5.2*e* + 001	4.39*e* + 002	7.81*e* + 001
	Rank	**1**	4	5	6	2	7	3
f_7	Mean	**8.01e − 016**	3.44*e* − 014	1.15*e* − 004	1.89*e* + 001	2.24*e* − 006	2.42	5.29*e* + 001
	Std	**0**	6.45*e* − 015	1.22*e* − 004	9.54*e* − 001	1.09*e* − 006	5.43	4.52
	Rank	**1**	2	5	7	4	6	8
f_8	Mean	**0**	1.20*e* − 008	3.43*e* − 002	1.97*e* − 003	6.23*e* − 002	4.37*e* − 002	2.21
	Std	**0**	1.17*e* − 008	2.22*e* − 002	4.40*e* − 003	5.05*e* − 002	4.23*e* − 002	1.46*e* − 001
	Rank	**1**	2	4	3	6	5	7
	Total Rank2	**4**	**9**	**18**	**22**	**15**	**23**	**25**
f_9	Mean	**0**	1.13*e* − 013	3.37*e* + 002	5.68*e* − 014	2.27*e* − 012	2.06*e* + 001	7.12*e* + 001
	Std	**0**	4.92*e* − 014	5.84*e* + 002	4.92*e* − 014	1.58*e* − 012	4.61*e* + 001	1.66*e* + 001
	Rank	**1**	3	7	2	4	5	6
f_{10}	Mean	3.91*e* + 001	2.10*e* + 003	6.14*e* + 003	**0**	4.50*e* + 003	1.44*e* + 002	8.93*e* + 003
	Std	4.96*e* + 001	1.52*e* + 002	1.17*e* + 004	**5.68e − 014**	1.32*e* + 003	1.10*e* + 002	5.18*e* + 003
	Rank	2	4	6	**1**	5	3	7
f_{11}	Mean	1.39	1.21*e* + 001	1.33*e* + 006	1.59	6.74	3.20*e* + 001	2.66*e* + 004
	Std	**8.90e − 001**	7.09	1.21*e* + 006	2.18	4.47	3.31*e* + 001	2.11*e* + 004
	Rank	**1**	4	7	2	3	5	6
f_{12}	Mean	**2.46e − 003**	1.72*e* + 003	1.73*e* + 003	1.72*e* + 003	1.72*e* + 003	4.13*e* + 003	1.92*e* + 003
	Std	**5.51e − 003**	2.39*e* − 011	3.30	3.02*e* − 001	4.41*e* − 008	4.23*e* + 002	4.62*e* + 001
	Rank	**1**	3	5	2	3	7	6
f_{13}	Mean	**2.06e + 001**	2.07*e* + 001	2.04*e* + 001	**2.01e + 001**	2.07*e* + 001	2.05*e* + 001	2.08*e* + 001
	Std	**5.22e − 002**	1.01*e* − 001	6.10*e* − 002	**3.82e − 001**	4.04*e* − 002	1.40*e* − 001	1.69*e* − 002
	Rank	3	5	2	**1**	5	4	7
f_{14}	Mean	**0**	1.01*e* + 003	1.58*e* + 001	4.16*e* + 001	4.26*e* − 002	9.79*e* + 001	3.26*e* + 001
	Std	**4.04e − 014**	3.82*e* + 001	1.15*e* + 001	9.98*e* + 001	9.51*e* − 002	7.66	8.18
	Rank	**1**	3	4	6	2	7	5

Table 7: Continued.

Function		HABC	CABC	CPSO	CMA-ES	ABC	PSO	GA
	Mean	**4.63e + 001**	1.34*e* + 002	1.87*e* + 002	1.21*e* + 002	1.35*e* + 002	1.79*e* + 002	4.22*e* + 002
f_{15}	Std	**1.22e + 001**	4.33*e* + 001	5.77*e* + 001	2.47*e* + 001	1.97*e* + 001	5.06*e* + 001	3.62*e* + 001
	Rank	**1**	3	6	2	4	5	7
Total Rank3		**9**	**25**	**30**	**16**	**26**	**36**	**44**

Table 8: Performance of all algorithms on benchmark discrete functions f_{16}–f_{20}.

Function		HABC	ABC	PSO	GA
	Mean	**0**	0	2.20*e* − 001	3.25*e* − 002
f_{16}	Std	**0**	0	7.27*e* − 002	7.27*e* − 002
	Rank	**1**	1	4	3
	Mean	**0.88**	1.12	1.16	1.36
f_{17}	Std	**1.09e − 001**	1.78*e* − 001	2.60*e* − 001	3.57*e* − 001
	Rank	**1**	3	6	7
	Mean	**4.00**	2.31*e* + 001	4.60	1.11*e* + 001
f_{18}	Std	**1.76**	7.53	1.19	4.95
	Rank	**1**	4	2	3
	Mean	0	1.33	2.51*e* − 001	1.65
f_{19}	Std	0	1.12*e* − 002	7.07*e* − 002	3.56*e* − 001
	Rank	1	3	2	4
	Mean	**2.21e − 001**	3.4*e* − 001	2.00	1.33
f_{20}	Std	**2.11e − 002**	1.67*e* − 001	1.09*e* − 001	2.05*e* − 001
	Rank	**1**	2	4	3
Total rank		**5**	**13**	**17**	**20**

Table 9: Parameters of the test functions.

f	Dimensions	Initial range	x^*	$f(x^*)$
f_1	100	$[-100, 100]^D$	$[0, 0, \ldots, 0]$	0
f_2	100	$[-30, 30]^D$	$[1, 1, \ldots, 1]$	0
f_3	100	$[-65.536, 65.536]^D$	$[0, 0, \ldots, 0]$	0
f_4	100	$[-10, 10]^D$	$[0, 0, \ldots, 0]$	0
f_5	100	$[-5.12, 5.12]^D$	$[0, 0, \ldots, 0]$	0
f_6	100	$[-500, 500]^D$	$[420.9867, \ldots, 420.9867]$	0
f_7	100	$[-32.768, 32.768]^D$	$[0, 0, \ldots, 0]$	0
f_8	100	$[-600, 600]^D$	$[0, 0, \ldots, 0]$	0
f_9	100	$[-100, 100]^D$	$[0, 0, \ldots, 0]$	−450
f_{10}	100	$[-100, 100]^D$	$[0, 0, \ldots, 0]$	−450
f_{11}	100	$[-100, 100]^D$	$[0, 0, \ldots, 0]$	390
f_{12}	100	No bounds	$[0, 0, \ldots, 0]$	−180
f_{13}	100	$[-32, 32]^D$	$[0, 0, \ldots, 0]$	−140
f_{14}	100	$[-5, 5]^D$	$[0, 0, \ldots, 0]$	−330
f_{15}	100	$[-5, 5]^D$	$[0, 0, \ldots, 0]$	−330
f_{16}	30	$[0, 1]$	$[1, 1, \ldots, 1]$	0
f_{17}	60	$[0, 1]$	$[0, 0, \ldots, 0]$ or $[1, 1, \ldots, 1]$ or $[\ldots, 6 * 0, \ldots, 6 * 1, \ldots]$	0
f_{18}	120	$[0, 1]$	$[0, 0, \ldots, 0]$	0
f_{19}	120	$[0, 1]$	$[0, 1, 1, 0, 1, 1, \ldots]$	0
f_{20}	120	$[0, 1]$	$[0, 0, 1, 0, 0, 1, \ldots]$	0

*Represents the solution corresponding to the optimal value of the objective function, namely the optimal solution.

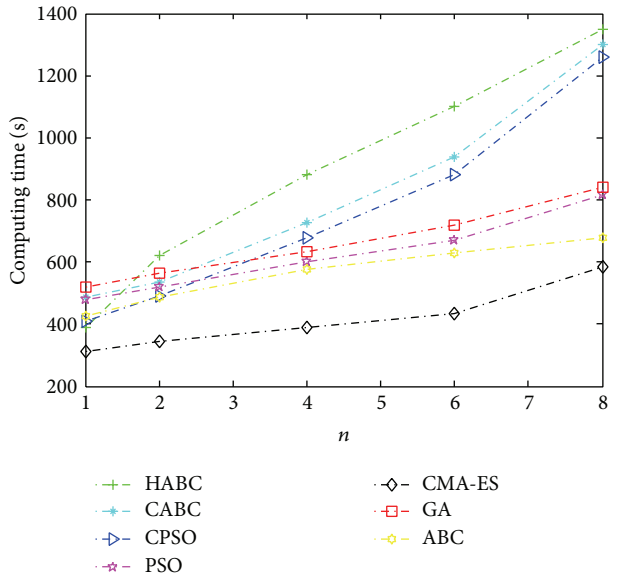

FIGURE 8: Computing time of all algorithms on different RNP problems.

continuous and discrete cases). Our results show that, for all the test functions, HABC is more competitive than other classical powerful algorithms. HABC is the only one to consistently find the minimum of f_1 (Sphere), f_2 (Rosenbrock), f_3 (Quadratic), f_5 (Rastrigin), and f_8 (Griewank), though CMA-ES got significantly better performance on the unimodal functions (f_1–f_4). On CEC2005 functions, HABC exhibits more significant advantages in terms of accuracy, convergence rate, and robustness. In the future work, we will extend the optimization problem to higher-dimensional (up to 1000 dimensions) objective function solved by our proposed HABC.

Appendix

A. List of Test Functions

The involved benchmark functions consist of the basic benchmark continuous functions $f_1 \sim f_4$ (unimodal), $f_5 \sim f_8$ (multimodal), the shifted and rotated functions $f_9 \sim f_{15}$ (CEC2005), and the discrete functions $f_{16} \sim f_{20}$. Functions $f_1 \sim f_8$ were adopted widely in evolutionary computation domain to show solution quality and convergence rate. $f_9 \sim f_{15}$ are shifted and rotated functions selected from CEC2005 functions; their global optimum values are different to each other. In particular, function f_{12} has no bounds, and its initialization range is [0, 600]. The discrete functions $f_{16} \sim f_{20}$ were used in Clerc's literature [39, 40] and can be found at http://clerc.maurice.free.fr/pso/.

A.1. Unimodal Benchmark Function

(1) Sphere function

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \tag{A.1}$$

(2) Rosenbrock function

$$f_2(x) = \sum_{i=1}^{n} 100 \times \left(x_{i+1} - x_i^2\right)^2 + \left(1 - x_i\right)^2 \tag{A.2}$$

(3) Quadratic function

$$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \tag{A.3}$$

(4) Sin function

$$f_4(x) = \frac{\pi}{n} \left\{ 10 \sin^2 \pi x_1 + \sum_{i=1}^{n-1} (x_i - 1)^2 \left(1 + 10 \sin^2 \pi x_{i+1}\right) + (x_n - 1)^2 \right\}. \tag{A.4}$$

A.2. Multimodal Benchmark Function

(5) Rastrigin function

$$f_5(x) = \sum_{i=1}^{n} \left(x_i^2 - 10 \cos\left(2\pi x_i\right)\right) + 10 \tag{A.5}$$

(6) Schwefel function

$$f_6(x) = D * 418.9829 + \sum_{i=1}^{D} - x_i \sin\left(\sqrt{|x_i|}\right) \tag{A.6}$$

(7) Ackley's function

$$f_7(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos 2\pi x_i\right) + 20 + e \tag{A.7}$$

(8) Griewank function

$$f_8(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1. \tag{A.8}$$

A.3. CEC2005 Function

(9) Shifted sphere function

$$f_9(x) = \sum_{i=1}^{D} z_i^2 + f_{\text{bias1}}, \quad z = x - o \tag{A.9}$$

(10) Shifted Schwefel's problem 1.2

$$f_{10}(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} z_j \right)^2 + f_{\text{bias}}, \quad z = x - o \tag{A.10}$$

(11) Shifted Rosenbrock's function

$$f_{11}(x) = \sum_{i=1}^{D-1} \left(100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i^2 - 1\right)^2\right) + f_{\text{bias}} \quad \text{(A.11)}$$

(12) Shifted rotated Griewank's function without bounds

$$f_{12}(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_{\text{bias}} \quad \text{(A.12)}$$

(13) Shifted rotated Ackley's function with global optimum on bounds

$$f_{13}(x) = -\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\sum_{i=1}^{D} \cos\left(2\pi z_i\right)\right) + 20 + e + f_{\text{bias}}, \quad z = (x - o) * M \quad \text{(A.13)}$$

(14) Shifted Rastrigin's function

$$f_{14}(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos\left(2\pi z_i\right) + 10\right) + f_{\text{bias}}, \quad z = x - o \quad \text{(A.14)}$$

(15) Shifted rotated Rastrigin's function

$$f_{15}(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos\left(2\pi z_i\right) + 10\right) + f_{\text{bias}}, \quad z = (x - o) * M. \quad \text{(A.15)}$$

A.4. Discrete Benchmark Functions

(16) Goldberg's order-3

The fitness f of a bit string is the sum of the results of separately applying the following function to consecutive groups of three components each:

$$f_{16}(x) = \begin{cases} 0.9 & \text{if } |y| = 0 \\ 0.6 & \text{if } |y| = 1 \\ 0.3 & \text{if } |y| = 2 \\ 1.0 & \text{if } |y| = 3. \end{cases} \quad \text{(A.16)}$$

If the string size (the dimension of the problem) is D, the maximum value is $D/3$ for the string $111\cdots111$. In practice, we will then use as fitness the value $D/3 - f$ so that the problem is now to find the minimum 0.

(17) Bipolar order-6

The fitness f is the sum of the results of applying the following function to consecutive groups of six components each:

$$f_{17}(x) = \begin{cases} 1.0 & \text{if } |y| = 0 \text{ or } 6 \\ 0.0 & \text{if } |y| = 1 \text{ or } 5 \\ 0.4 & \text{if } |y| = 2 \text{ or } 4 \\ 0.8 & \text{if } |y| = 3. \end{cases} \quad \text{(A.17)}$$

The maximum value is $D/6$. In practice, we will use as fitness the value $D/6 - f$ so that the problem is now to find the minimum 0.

(18) Mulenbein's order-5

The fitness f is the sum of the results of applying the following function to consecutive groups of five components each:

$$f_{18}(x) = \begin{cases} 4.0 & \text{if } y = 00000 \\ 3.0 & \text{if } y = 00001 \\ 2.0 & \text{if } y = 00011 \\ 1.0 & \text{if } y = 00111 \\ 3.5 & \text{if } y = 11111 \\ 0.0 & \text{ortherwise.} \end{cases} \quad \text{(A.18)}$$

The maximum value is $3.5D/5$. In practice, the value $3.5D/5 - f$ is used as the fitness so that the problem is now to find the minimum 0.

(19) Clerc's order-3 problem 1

The fitness f is the sum of the results of applying the following function to consecutive groups of three components each:

$$f_{19}(x) = \begin{cases} 0.9 & \text{if } y = 000 \\ 0.6 & \text{if } y = 001 \\ 0.3 & \text{if } y = 010 \\ 1.0 & \text{if } y = 011 \\ 0.2 & \text{if } y = 100 \\ 0.4 & \text{if } y = 101 \\ 0.6 & \text{if } y = 110 \\ 0.8 & \text{if } y = 111. \end{cases} \quad \text{(A.19)}$$

The maximum value is $D/3$. In practice, we will then use as fitness the value $D/3 - f$ so that the problem is now to find the minimum 0.

(20) Clerc's order-3 problem 2

The fitness f is the sum of the results of applying the following function to consecutive groups of three components each:

$$f_{20}(x) = \begin{cases} 0.2 & \text{if } y = 000 \\ 1.0 & \text{if } y = 001 \\ 0.3 & \text{if } y = 010 \\ 0.8 & \text{if } y = 011 \\ 0.6 & \text{if } y = 100 \\ 0.4 & \text{if } y = 101 \\ 0.6 & \text{if } y = 110 \\ 0.9 & \text{if } y = 111. \end{cases} \quad \text{(A.20)}$$

The maximum value is $D/3$. In practice, we will use as fitness the value $D/3 - f$ so that the problem is now to find the minimum 0.

B. Parameters of the Test Functions

The dimensions, initialization ranges, global optima x^*, and the corresponding fitness value $f(x^*)$ of each function are listed in Table 9.

Conflict of Interests

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

Acknowledgments

This work is partially supported by the Natural Science Foundation of China (Grants nos. 61105067, 61174164, 71271140, and 71001072), the National Key Technology R&D Program of China under Grants nos. 2012BAF10B11 and 2012BAF10B06, The Hong Kong Scholars Program 2012 (Grant no. G-YZ24), China Postdoctoral Science Foundation (Grant no. 20100480705), Special Financial Grant from the China Postdoctoral Science Foundation (Grants nos. 2012T50584 and 2012T50639), and the Natural Science Foundation of Guangdong Province (Grants nos. S2012010008668 and 9451806001002294).

References

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

[2] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[3] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52–67, 2002.

[4] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri, Turkey, 2005.

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

[6] D. Karaboga and B. Basturk, "Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems," in *Foundations of Fuzzy Logic and Soft Computing*, P. Melin, O. Castillo, L. T. Aguilar, J. Kacprzyk, and W. Pedrycz, Eds., vol. 4529 of *Lecture Notes in Computer Science*, pp. 789–798, 2007.

[7] B. Basturk and D. Karaboga, "An Artificial Bee Colony (ABC) algorithm for numerical function optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium*, Indianapolis, Ind, USA, 2006.

[8] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[9] D. Karaboga and B. Basturk, "Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems," in *Lecture Notes in Computer Science*, vol. 4529, pp. 789–798, 2007.

[10] D. Karaboga, B. Akay, and C. Ozturk, "Artificial Bee Colony (ABC) optimization algorithm for training feed-forward neural networks," in *Modeling Decisions for Artificial Intelligence*, vol. 4617, pp. 318–329, 2007.

[11] A. Baykasoglu, L. Ozbakir, and P. Tapkan, "Artificial Bee Colony algorithm and its application to generalized assignment problem," in *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, pp. 113–144, I-Tech Education and Publishing, Vienna, Austria, 2007.

[12] M. F. Tasgetiren, Q. Pan, P. N. Suganthan, and A. H. Chen, "A discrete Artificial Bee Colony algorithm for the total flowtime minimization in permutation flow shops," *Information Sciences*, vol. 181, no. 16, pp. 3459–3475, 2011.

[13] M. F. Tasgetiren, Q. Pan, P. N. Suganthan, and A. H. Chen, "A discrete Artificial Bee Colony algorithm for the total flowtime minimization in permutation flow shops," *Information Sciences*, vol. 181, no. 16, pp. 3459–3475, 2011.

[14] L. S. Coelho and P. Alotto, "Gaussian Artificial Bee Colony algorithm approach applied to Loney's solenoid benchmark problem," *IEEE Transactions on Magnetics*, vol. 47, no. 5, pp. 1326–1329, 2011.

[15] M. A. Potter and K. A. de Jong, "A cooperative coevolutionary approach to function optimization," in *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, Y. Davidor, H.-P. Schwefel, and R. Manner, Eds., vol. 866 of *Lecture Notes in Computer Science*, pp. 249–257, 1994.

[16] M. A. Potter and K. A. De Jong, "Cooperative coevolution: an architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.

[17] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *Proceedings of the 2007 Congress on Evolutionary Computation (CEC '07)*, pp. 3523–3530, September 2007.

[18] Y. Shi, H. Teng, and Z. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Proceedings of the 1st International Conference on Natural Computation (ICNC '05)*, pp. 1080–1088, August 2005.

[19] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[20] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[21] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210–224, 2012.

[22] H. Chen, Y. Zhu, K. Hu, and X. He, "Hierarchical swarm model: a new approach to optimization," *Discrete Dynamics in Nature and Society*, vol. 2010, Article ID 379649, 30 pages, 2010.

[23] B. Niu, Y. Zhu, X. He, and H. Wu, "MCPSO: a multi-swarm cooperative particle swarm optimizer," *Applied Mathematics and Computation*, vol. 185, no. 2, pp. 1050–1062, 2007.

[24] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer and its adaptive variant," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 35, no. 6, pp. 1272–1282, 2005.

[25] H. Chen, Y. Zhu, K. Hu, and X. He, "Hierarchical swarm model: a new approach to optimization," *Discrete Dynamics in Nature and Society*, vol. 2010, Article ID 379649, 30 pages, 2010.

[26] Y. Peng and B. Lu, "A hierarchical particle swarm optimizer with latin sampling based memetic algorithm for numerical optimization," *Applied Soft Computing*, vol. 13, no. 5, pp. 2823–2836, 2013.

[27] J. Kennedy and R. Mendes, "Topological structure and particle swarm performance," in *Proceedings of the 4th Congress on Evolutionary Computation (CEC '02)*, pp. 1671–1676, 2002.

[28] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[29] B. Y. Qu, P. N. Suganthan, and J. J. Liang, "Differential evolution with neighborhood mutation for multimodal optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 5, pp. 601–614, 2012.

[30] B. Y. Qu, P. N. Suganthan, and S. Das, "A distance-based locally informed particle swarm model for multimodal optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 387–402, 2013.

[31] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 1, pp. 150–169, 2010.

[32] 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.

[33] S. Sumathi, T. Hamsapriya, and P. Surekha, *Evolutionary Intelligence: An Introduction to Theory and Applications with Matlab*, Springer, New York, NY, USA, 2008.

[34] D. H. Wolpert and W. G. Macready, "No free lunch theorems for search," Tech. Rep. SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, USA, 1995.

[35] W. Zou, Y. Zhu, H. Chen, and X. Sui, "A clustering approach using cooperative Artificial Bee Colony algorithm," *Discrete Dynamics in Nature and Society*, vol. 2010, Article ID 459796, 30 pages, 2010.

[36] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[37] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[38] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the CEC, 2005 special session on real-parameter optimization," MayKanGAL Report 2005005, Nanyang Technological University, Singapore; IIT Kanpur, Uttar Pradesh, India, 2005.

[39] M. Clerc, "Binary Particle Swarm Optimisers: Toolbox, Derivations, and Mathematical Insights," 2005, http://clerc.maurice.free.fr/pso/.

[40] M. Clerc, "Discrete particle swarm optimization," in *New Optimization Techniques in Engineering*, Springer, New York, NY, USA, 2004.