Applied Mathematics and Computation 214 (2009) 108–132

Contents lists available at ScienceDirect

Applied Mathematics and Computation

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

A comparative study of Artificial Bee Colony algorithm

Dervis Karaboga *, Bahriye Akay

Erciyes University, The Department of Computer Engineering, Melikgazi, 38039 Kayseri, Turkey

ARTICLE INFO

Keywords:
Swarm intelligence
Evolution strategies
Genetic algorithms
Differential evolution
Particle swarm optimization
Artificial Bee Colony algorithm
Unconstrained optimization

ABSTRACT

Artificial Bee Colony (ABC) algorithm is one of the most recently introduced swarm-based algorithms. ABC simulates the intelligent foraging behaviour of a honeybee swarm. In this work, ABC is used for optimizing a large set of numerical test functions and the results produced by ABC algorithm are compared with the results obtained by genetic algorithm, particle swarm optimization algorithm, differential evolution algorithm and evolution strategies. Results show that the performance of the ABC is better than or similar to those of other population-based algorithms with the advantage of employing fewer control parameters.

1. Introduction

Population-based optimization algorithms find near-optimal solutions to the difficult optimization problems by motivation from nature. A common feature of all population-based algorithms is that the population consisting of possible solutions to the problem is modified by applying some operators on the solutions depending on the information of their fitness. Hence, the population is moved towards better solution areas of the search space. Two important classes of population-based optimization algorithms are evolutionary algorithms [1] and swarm intelligence-based algorithms [2]. Although Genetic Algorithm (GA) [3], Genetic Programming (GP) [4], Evolution Strategy (ES) [5,6] and Evolutionary Programming (EP) [7] are popular evolutionary algorithms, GA is the most widely used one in the literature. GA is based on genetic science and natural selection and it attempts to simulate the phenomenon of natural evolution at genotype level while ES and EP simulate the phenomenon of natural evolution at phenotype level. One of the evolutionary algorithms which has been introduced recently is Differential Evolution (DE) algorithm [8–10]. DE has been particularly proposed for numerical optimization problems. In the basic GA, a selection operation is applied to the solutions evaluated by the evaluation unit. At this operation the chance of a solution being selected as a parent depends on the fitness value of that solution. One of the main differences between the GA and the DE algorithm is that, at the selection operation of the DE algorithm, all solutions have an equal chance of being selected as parents, i.e. the chance does not depend on their fitness values. In DE, each new solution produced competes with its parent and the better one wins the competition. In recent years, swarm intelligence has also attracted the interest of many research scientists of related fields. Bonabeau has defined the swarm intelligence as "...any attempt to design algorithms or distributed problem-solving devices inspired by the collective behaviour of social insect colonies and other animal societies..." [11]. Bonabeau et al. focused their viewpoint on social insects alone such as termites, bees, wasps and different ant species. However, a swarm can be considered as any collection of interacting agents or individuals. An ant colony can be thought of as a swarm whose individual agents are ants; a flock of birds is a swarm of birds. An immune system [12] can be considered as a swarm of cells and molecules as well as a crowd is a swarm of people. A popular swarm-intelligence-based algorithm is the Particle Swarm Optimization (PSO) algorithm which was introduced by Eberhart and Kennedy in 1995 [13]. PSO is also a population-based stochastic optimization technique and is well adapted to the optimization of nonlinear functions in multidimensional space. It models

* Corresponding author.
E-mail address: karaboga@erciyes.edu.tr (D. Karaboga).

0096-3003/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2009.03.090

the social behaviour of bird flocking or fish schooling. PSO has received significant interest from researchers studying in different research areas and has been successfully applied to several real-world problems [14].

The classical example of a swarm is bees' swarming around their hive but it can be extended to other systems with a similar architecture. Some approaches have been proposed to model the specific intelligent behaviours of honeybee swarms and they have been applied for solving combinatorial type problems [15–23]. Tereshko considered a bee colony as a dynamical system gathering information from an environment and adjusting its behaviour in accordance to it [15]. Tereshko and Loengarov established a robot idea on foraging behaviour of bees. Usually, all these robots are physically and functionally identical, so that any robot can replace any other robot. The swarm possesses a significant tolerance; the failure of a single agent does not stop performance of the whole system. Like insects, the robots individually have limited capabilities and limited knowledge of the environment. On the other hand, the swarm develops collective intelligence. The experiments showed that insect-like robots are successful in real robotic tasks. Tereshko and Loengarov also developed a minimal model of forage selection that leads to the emergence of collective intelligence which consists of three essential components: food sources, employed foragers, and unemployed foragers, and defines two leading modes of the behaviour: recruitment to a nectar source and abandonment of a source [16,17]. Teodorović suggested to use bee swarm intelligence in the development of artificial systems aimed at solving complex problems in traffic and transportation [19,18]. Once more Teodorović proposed the Bee Colony Optimization (BCO) Metaheuristic which is capable of solving deterministic combinatorial problems, as well as combinatorial problems characterized by uncertainty [20]. Drias et al. introduced a new intelligent approach or meta-heuristic called Bees Swarm Optimization (BSO) and adapted it to the features of the maximum weighted satisfiability (max-sat) problem [21]. Similarly, Benatchba et al. introduced a metaheuristic based on the process of bees' reproduction to solve a 3-sat problem [22]. Wedde et al. presented a novel routing algorithm, called BeeHive, which was the inspiration gained from communicative and evaluative methods and procedures of honeybees [23]. In BeeHive algorithm, bee agents travel through network regions called foraging zones. On their way their information on the network state is delivered for updating the local routing tables.

The works mentioned in the previous paragraph introduced the algorithms of bees proposed for the combinatorial type problems. There are three continuous optimization algorithms based on intelligent behaviour of honeybee swarm [24–26] in the literature. Yang developed a Virtual Bee Algorithm (VBA) [24] to optimize only two-dimensional numeric functions. A swarm of virtual bees is generated and started to move randomly in two-dimensional search space. Bees interact when they find some target nectar and the solution of the problem is obtained from the intensity of these bee interactions. Pham et al. introduced the Bees Algorithm in 2005, which employs several control parameters [25]. For optimizing multi-variable and multi-modal numerical functions, Karaboga described an Artificial Bee Colony (ABC) algorithm [26] in 2005. Basturk and Karaboga compared the performance of ABC algorithm with those of GA [27], PSO and PS-EA [28]; and DE, PSO and EA [29] on a limited number of test problems. They have extended ABC algorithm for constrained optimization problems in [30] and applied ABC for training neural networks [31,32]. ABC algorithm was used for designing IIR filters in [33] and for the leaf-constrained minimum spanning tree problem in [34].

In this work, a comprehensive comparative study on the performances of well-known evolutionary and swarm-based algorithms for optimizing a very large set of numerical functions is presented. In Section 2, ES, GA, PSO and DE algorithms are briefly summarized. In Section 3, the foraging behaviour of real bees and then ABC algorithm simulating this behaviour are described. Finally, in Section 4 and Section 5, the simulation results obtained are presented and discussed, respectively.

2. Evolution strategies, genetic algorithm, differential evolution algorithm, particle swarm optimization algorithm

2.1. Evolution strategies

Evolution Strategy has been proposed for numerical optimization problems and it is one of the oldest evolutionary algorithms. Evolution Strategy produces n-dimensional real-valued vector by mutating its parent with identical standard deviations to each vector element. The produced individual is evaluated and a selection scheme is applied to determine which one will survive in the next generation and the other one will be discarded. This simple selection mechanism is expressed as $(1+1)$-selection. In a $(\mu+1)$-ES, which is a multi-membered evolution strategy, μ parent individuals recombine to form one offspring, which after being mutated eventually replaces the worst parent individuals, if it is better [35]. The canonical versions of the ES (CES) are denoted by $(\mu/\rho,\lambda)$-ES and $(\mu/\rho+\lambda)$-ES. Here μ indicates the number of parents, $\rho \leqslant \mu$ is the number of parents involved in the generation of an offspring, and λ is the number of offspring produced. The parents are deterministically chosen from the set of either the offspring (referred to as comma-selection and $\mu < \lambda$ must hold) or both the parents and offspring (referred to as plus-selection). Selection is based on the fitness of individuals. The main steps of ES are given below:

1: Initialize Population
2: **repeat**
3: Recombination
4: Mutation
5: Evaluation
6: Selection
7: **until** requirements are met

Mutation is carried out by adding a normally distributed random term to each vector. Evolution strategies using Cauchy mutation operators are called Fast Evolution Strategies (FES) [36]. Another mutation operator is Self-Organising Maps (SOM)-based mutation operator which is continuously trained on successful offspring only, thus reflecting the evolution path in away that allows for selection of further successful candidates with high probability [37]. Kohonen's Self-Organising Mapping Evolution Strategy (KSOM-ES) is based on this adaptation for the mutation operator [38]. Neural Gas networks Evolution Strategy (NG-ES) has been used to overcome bad scaling property of KSOM-ES [39]. In Covariance Matrix Adaptation Evolution Strategies (CMA-ES) devised for optimal exploitation of local information, the step size can be selected by self-adaptation or by covariance matrix adaptation [40]. Evolution Strategies Learned with Automatic Termination (ESLAT) criterion is another Evolution Strategy which uses an automatic termination criterion based on a gene matrix [41].

2.2. Genetic algorithm

A basic GA consists of five components. These are a random number generator, a *fitness* evaluation unit and genetic operators for *reproduction*, *crossover* and *mutation* operations. The basic algorithm is summarized below:

1: Initialize Population
2: **repeat**
3: Evaluation
4: Reproduction
5: Crossover
6: Mutation
7: **until** requirements are met

The initial population required at the start of the algorithm, is a set of number strings generated by the random generator. Each string is a representation of a solution to the optimization problem being addressed. Binary strings are commonly employed. Associated with each string is a fitness value computed by the evaluation unit. A fitness value is a measure of the goodness of the solution that it represents. The aim of the genetic operators is to transform this set of strings into sets with higher fitness values. The reproduction operator performs a natural selection function known as *seeded selection*. Individual strings are copied from one set (representing a generation of solutions) to the next according to their fitness values, the higher the fitness value, the greater the probability of a string being selected for the next generation. The crossover operator chooses pairs of strings at random and produces new pairs. The simplest crossover operation is to cut the original *parent* strings at a randomly selected point and to exchange their tails. The number of crossover operations is governed by a crossover rate. The mutation operator randomly mutates or reverses the values of bits in a string. The number of mutation operations is determined by a mutation rate. A phase of the algorithm consists of applying the evaluation, reproduction, crossover and mutation operations. A new generation of solutions is produced with each phase of the algorithm [42].

2.3. Differential evolution algorithm

The DE algorithm is a population-based algorithm like genetic algorithms using the similar operators; crossover, mutation and selection. The main difference in constructing better solutions is that genetic algorithms rely on crossover while DE relies on mutation operation. This main operation is based on the differences of randomly sampled pairs of solutions in the population. The algorithm uses mutation operation as a search mechanism and selection operation to direct the search toward the prospective regions in the search space. The DE algorithm also uses a non-uniform crossover that can take child vector parameters from one parent more often than it does from others. By using the components of the existing population members to construct trial vectors, the recombination (crossover) operator efficiently shuffles information about successful combinations, enabling the search for a better solution space. In DE, a population of solution vectors is randomly created at the start. This population is successfully improved by applying mutation, crossover and selection operators. In the DE algorithm, each new solution produced competes with a mutant vector and the better one wins the competition. DE has received significant interest from researchers studying in different research areas and has been applied to several real-world problems [8,43]. The main steps of the DE algorithm are presented below:

1: Initialize Population
2: Evaluation
3: **repeat**
4: Mutation
5: Recombination
6: Evaluation
7: Selection
8: **until** requirements are met

Mutation: Each of the N parameter vectors undergoes mutation. Mutation operation expands the search space. A mutant solution vector $\hat{x}_i$ is produced by (1):

$$\hat{x}_i = x_{r_1} + F(x_{r_3} - x_{r_2}), \tag{1}$$

where F is the scaling factor having values in the range of $[0,1]$ and solution vectors x_{r1}, x_{r2} and x_{r3} are randomly chosen and must satisfy (2):

$$x_{r_1}, x_{r_2}, x_{r_3} | r_1 \neq r_2 \neq r_3 \neq i, \tag{2}$$

where i is the index of current solution.

Crossover: The parent vector is mixed with the mutated vector to produce a trial vector by (3):

$$y_i^j = \begin{cases} \hat{x}_i^j & R_j \leqslant CR, \\ x_i^j & R_j > CR, \end{cases} \tag{3}$$

where CR is crossover constant and R_j is a randomly selected real number between $[0,1]$ and j denotes the jth component of the corresponding array.

All solutions in the population have the same chance of being selected as parents without dependence of their fitness value. The child produced after the mutation and crossover operations is evaluated. Then, the performance of the child vector and its parent is compared and the better one wins the competition. If the parent is still better, it is retained in the population.

2.4. Particle swarm optimization

In PSO, a population of particles starts to move in search space by following the current optimum particles and changing the positions in order to find out the optima. The position of a particle refers to a possible solution of the function to be optimized. Evaluating the function by the particle's position provides the fitness of that solution. In every iteration, each particle is updated by following the best solution of current particle achieved so far ($\vec{p}(t)$, particle best) and the best of the population ($\vec{g}(t)$, global best). When a particle takes part of the population as its topological neighbours, the best value is a local best. The particles tend to move to good areas in the search space by the information spreading to the swarm. The particle is moved to a new position calculated by the velocity updated at each time step t. This new position is calculated as the sum of the previous position and the new velocity by (4):

$$\vec{x}(t+1) = \vec{x}(t) + \vec{v}(t+1). \tag{4}$$

The velocity update is performed as indicated in Eq. (5):

$$\vec{v}(t+1) = \omega\vec{v}(t) + \phi_1 \mathrm{rand}(0,1)(\vec{p}(t) - \vec{x}(t)) + \phi_2 \mathrm{rand}(0,1)(\vec{g}(t) - \vec{x}(t)). \tag{5}$$

The parameter ω is called the inertia weight and controls the magnitude of the old velocity $\vec{v}(t)$ in the calculation of the new velocity, whereas ϕ_1 and ϕ_2 determine the significance of $\vec{p}(t)$ and $\vec{g}(t)$, respectively. Furthermore, v_i at any time step of the algorithm is constrained by the parameter v_{max}. The swarm in PSO is initialized by assigning each particle to a uniformly and randomly chosen position in the search space. Velocities are initialized randomly in the range $[v_{min}, v_{max}]$. Main steps of the procedure are:

1: Initialize Population
2: **repeat**
3: Calculate fitness values of particles
4: Modify the best particles in the swarm
5: Choose the best particle
6: Calculate the velocities of particles
7: Update the particle positions
8: **until** requirements are met

Particles' velocities on each dimension are clamped to a maximum velocity v_{max}. If the sum of accelerations would cause the velocity on that dimension to exceed v_{max}, which is a parameter specified by the user, then the velocity on that dimension is limited to v_{max}.

3. Artificial Bee Colony (ABC) algorithm

3.1. Behaviour of real bees

Tereshko developed a model of foraging behaviour of a honeybee colony based on reaction–diffusion equations [15–17]. This model that leads to the emergence of collective intelligence of honeybee swarms consists of three essential

components: food sources, employed foragers, and unemployed foragers, and defines two leading modes of the honeybee colony behaviour: recruitment to a food source and abandonment of a source. Tereshko explains the main components of his model as below:

1. Food Sources: In order to select a food source, a forager bee evaluates several properties related with the food source such as its closeness to the hive, richness of the energy, taste of its nectar, and the ease or difficulty of extracting this energy. For the simplicity, the quality of a food source can be represented by only one quantity although it depends on various parameters mentioned above.
2. Employed foragers: An employed forager is employed at a specific food source which she is currently exploiting. She carries information about this specific source and shares it with other bees waiting in the hive. The information includes the distance, the direction and the profitability of the food source.
3. Unemployed foragers: A forager bee that looks for a food source to exploit is called unemployed.It can be either a scout who searches the environment randomly or an onlooker who tries to find a food source by means of the information given by the employed bee. The mean number of scouts is about 5–10%.

The exchange of information among bees is the most important occurrence in the formation of collective knowledge. While examining the entire hive it is possible to distinguish some parts that commonly exist in all hives. The most important part of the hive with respect to exchanging information is the dancing area. Communication among bees related to the quality of food sources occurs in the dancing area. The related dance is called waggle dance. Since information about all the current rich sources is available to an onlooker on the dance floor, probably she could watch numerous dances and chooses to employ herself at the most profitable source. There is a greater probability of onlookers choosing more profitable sources since more information is circulating about the more profitable sources. Employed foragers share their information with a probability which is proportional to the profitability of the food source, and the sharing of this information through waggle dancing is longer in duration. Hence, the recruitment is proportional to profitability of a food source [17].

In order to better understand the basic behaviour characteristics of foragers, let us examine Fig. 1. Assume that there are two discovered food sources: A and B. At the very beginning, a potential forager will start as unemployed forager. That forager bee will have no knowledge about the food sources around the nest. There are two possible options for such a bee:

i. It can be a scout and starts searching around the nest spontaneously for food due to some internal motivation or possible external clue (S on Fig. 1).
ii. It can be a recruit after watching the waggle dances and starts searching for a food source (R on Fig. 1).

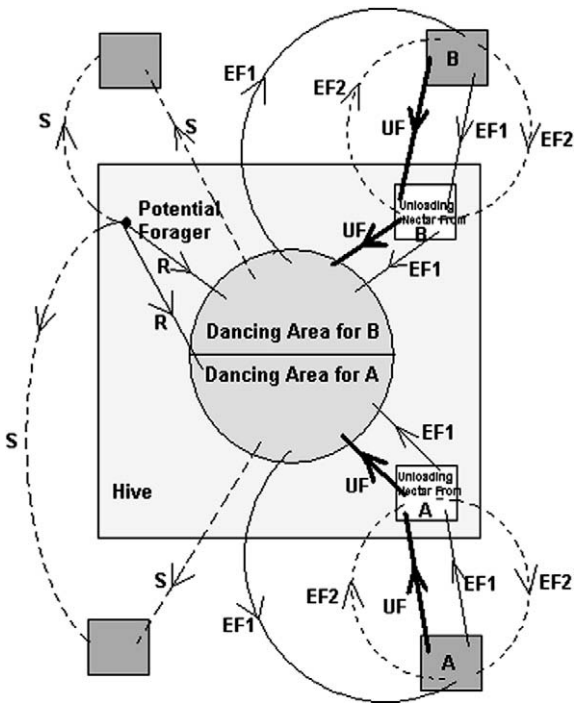

Fig. 1. Behaviour of honeybee foraging for nectar.

After finding the food source, the bee utilizes its own capability to memorize the location and then immediately starts exploiting it. Hence, the bee will become an employed forager. The foraging bee takes a load of nectar from the source and returns to the hive, unloading the nectar to a food store. After unloading the food, the bee has the following options:

i. It might become an uncommitted follower after abandoning the food source (UF).
ii. It might dance and then recruit nest mates before returning to the same food source (EF1).
iii. It might continue to forage at the food source without recruiting bees (EF2).

It is important to note that not all bees start foraging simultaneously. The experiments confirmed that new bees begin foraging at a rate proportional to the difference between the eventual total number of bees and the number of bees presently foraging.

3.2. Artificial Bee Colony (ABC) algorithm

In ABC algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The number of the employed bees or the onlooker bees is equal to the number of solutions in the population. At the first step, the ABC generates a randomly distributed initial population $P(C = 0)$ of SN solutions (food source positions), where SN denotes the size of employed bees or onlooker bees. Each solution $x_i (i = 1, 2, \ldots, SN)$ is a D-dimensional vector. Here, D is the number of optimization parameters. After initialization, the population of the positions (solutions) is subject to repeated cycles, $C = 1, 2, \ldots, MCN$, of the search processes of the employed bees, the onlooker bees and the scout bees. An employed bee produces a modification on the position (solution) in her memory depending on the local information (visual information) and tests the nectar amount (fitness value) of the new source (new solution). If the nectar amount of the new one is higher than that of the previous one, the bee memorizes the new position and forgets the old one. Otherwise she keeps the position of the previous one in her memory. After all employed bees complete the search process, they share the nectar information of the food sources and their position information with the onlooker bees. An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source with a probability related to its nectar amount. As in the case of the employed bee, she produces a modification on the position in her memory and checks the nectar amount of the candidate source. If the nectar is higher than that of the previous one, the bee memorizes the new position and forgets the old one.

The main steps of the algorithm are as below:

1: Initialize Population
2: **repeat**
3: Place the employed bees on their food sources
4: Place the onlooker bees on the food sources depending on their nectar amounts
5: Send the scouts to the search area for discovering new food sources
6: Memorize the best food source found so far
7: **until** requirements are met

In ABC algorithm, each cycle of the search consists of three steps: sending the employed bees onto their food sources and evaluating their nectar amounts; after sharing the nectar information of food sources, the selection of food source regions by the onlookers and evaluating the nectar amount of the food sources; determining the scout bees and then sending them randomly onto possible new food sources. At the initialization stage, a set of food sources is randomly selected by the bees and their nectar amounts are determined. At the first step of the cycle, these bees come into the hive and share the nectar information of the sources with the bees waiting on the dance area. A bee waiting on the dance area for making decision to choose a food source is called onlooker and the bee going to the food source visited by herself just before is named as employed bee. After sharing their information with onlookers, every employed bee goes to the food source area visited by herself at the previous cycle since that food source exists in her memory, and then chooses a new food source by means of visual information in the neighbourhood of the one in her memory and evaluates its nectar amount. At the second step, an onlooker prefers a food source area depending on the nectar information distributed by the employed bees on the dance area. As the nectar amount of a food source increases, the probability of that food source chosen also increases. After arriving at the selected area, she chooses a new food source in the neighbourhood of the one in the memory depending on visual information as in the case of employed bees. The determination of the new food source is carried out by the bees based on the comparison process of food source positions visually. At the third step of the cycle, when the nectar of a food source is abandoned by the bees, a new food source is randomly determined by a scout bee and replaced with the abandoned one. In our model, at each cycle at most one scout goes outside for searching a new food source, and the number of employed and onlooker bees is selected to be equal to each other. These three steps are repeated through a predetermined number of cycles called Maximum Cycle Number (MCN) or until a termination criterion is satisfied.

An artificial onlooker bee chooses a food source depending on the probability value associated with that food source, p_i, calculated by the following expression (6):

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}, \tag{6}$$

where fit_i is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i and SN is the number of food sources which is equal to the number of employed bees or onlooker bees.

In order to produce a candidate food position from the old one in memory, the ABC uses the following expression (7):

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \tag{7}$$

where $k \in \{1, 2, \ldots, SN\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indexes. Although k is determined randomly, it has to be different from i. $\phi_{i,j}$ is a random number between $[-1, 1]$. It controls the production of neighbour food sources around $x_{i,j}$ and represents the comparison of two food positions visually by a bee. As can be seen from (7), as the difference between the parameters of the $x_{i,j}$ and $x_{k,j}$ decreases, the perturbation on the position $x_{i,j}$ gets decreased, too. Thus, as the search approaches the optimum solution in the search space, the step length is adaptively reduced.

If a parameter value produced by this operation exceeds its predetermined limit, the parameter can be set to an acceptable value. In this work, the value of the parameter exceeding its limit is set to its limit value.

The food source of which the nectar is abandoned by the bees is replaced with a new food source by the scouts. In ABC, this is simulated by producing a position randomly and replacing it with the abandoned one. In ABC, if a position cannot be improved further through a predetermined number of cycles, then that food source is assumed to be abandoned. The value of predetermined number of cycles is an important control parameter of the ABC algorithm, which is called "*limit*" for abandonment. Assume that the abandoned source is x_i and $j \in \{1, 2, \ldots, D\}$, then the scout discovers a new food source to be replaced with x_i. This operation can be defined as in (8)

$$x_i^j = x_{min}^j + \mathrm{rand}[0, 1]\left(x_{max}^j - x_{min}^j\right). \tag{8}$$

After each candidate source position $v_{i,j}$ is produced and then evaluated by the artificial bee, its performance is compared with that of its old one. If the new food source has an equal or better nectar than the old source, it is replaced with the old one in the memory. Otherwise, the old one is retained in the memory. In other words, a greedy selection mechanism is employed as the selection operation between the old and the candidate one.

Totally, ABC algorithm employs four different selection processes: (1) a global probabilistic selection process, in which the probability value is calculated by (6) used by the onlooker bees for discovering promising regions, (2) a local probabilistic selection process carried out in a region by the employed bees and the onlookers depending on the visual information such as the color, shape and fragrance of the flowers (sources) (bees will not be able to identify the type of nectar source until they arrive at the right location and discriminate among sources *growing* there based on their scent) for determining a food source around the source in the memory in a way described by (7), (3) a local selection called greedy selection process carried out by onlooker and employed bees in that if the nectar amount of the candidate source is better than that of the present one, the bee forgets the present one and memorizes the candidate source produced by (7). Otherwise, the bee keeps the present one in the memory. (4) a random selection process carried out by scouts as defined in (8).

It is clear from the above explanation that there are three control parameters in the basic ABC: The number of food sources which is equal to the number of employed or onlooker bees (SN), the value of *limit* and the maximum cycle number (MCN).

In the case of honeybees, the recruitment rate represents a *measure* of how quickly the bee colony finds and exploits a newly discovered food source. Artificial recruiting could similarly represent the *measurement* of the speed with which the feasible solutions or the *good quality* solutions of the difficult optimization problems can be discovered. The survival and progress of the bee colony are dependent upon the rapid discovery and efficient utilization of the best food resources. Similarly; the successful solution of difficult engineering problems is connected to the relatively fast discovery of *good solutions* especially for the problems that need to be solved in real time. In a robust search process, exploration and exploitation processes must be carried out together. In the ABC algorithm, while onlookers and employed bees carry out the exploitation process in the search space, the scouts control the exploration process. Detailed pseudo-code of the ABC algorithm is given below:

1: Initialize the population of solutions $x_i, i = 1, \ldots, SN$
2: Evaluate the population
3: cycle = 1
4: **repeat**
5: Produce new solutions v_i for the employed bees by using (7) and evaluate them
6: Apply the greedy selection process for the employed bees
7: Calculate the probability values P_i for the solutions x_i by (6)
8: Produce the new solutions v_i for the onlookers from the solutions x_i selected depending on P_i and evaluate them
9: Apply the greedy selection process for the onlookers
10: Determine the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution x_i by (8)
11: Memorize the best solution achieved so far
12: cycle = cycle + 1
13: **until** cycle = MCN

4. Experiments 1: ABC vs. GA, DE and PSO

4.1. Settings

In all experiments in this section, the values of the common parameters used in each algorithm such as population size and total evaluation number were chosen to be the same. Population size was 50 and the maximum evaluation number was 500.000 for all functions. The other specific parameters of algorithms are given below:

GA Settings: In our experiments, we employed a binary coded standard GA having evaluation, fitness scaling, seeded selection, random selection, crossover, mutation and elite units. Single point crossover operation with the rate of 0.8 was employed. Mutation operation restores genetic diversity lost during the application of reproduction and crossover. Mutation rate in our experiments was 0.01. Stochastic uniform sampling technique was our selection method. Generation gap is the proportion of the population to be replaced. Chosen generation gap value in experiments was 0.9.

DE Settings: In DE, F is a real constant which affects the differential variation between two solutions and set to 0.5 in our experiments. Value of crossover rate, which controls the change of the diversity of the population, was chosen to be 0.9 as recommended in [43].

PSO Settings: Cognitive and social components (ϕ_1 and ϕ_2 in (5)) are constants that can be used to change the weighting between personal and population experience, respectively. In our experiments cognitive and social components were both set to 1.8. Inertia weight, which determines how the previous velocity of the particle influences the velocity in the next iteration, was 0.6 as recommended in [44].

Table 1
Benchmark functions used in experiments 1. D: Dimension, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable.

No	Range	D	C	Function	Formulation
1	$[-5.12, 5.12]$	5	US	Stepint	$f(x) = 25 + \sum_{i=1}^{5} \lfloor x_i \rfloor$
2	$[-100, 100]$	30	US	Step	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$
3	$[-100, 100]$	30	US	Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$
4	$[-10, 10]$	30	US	SumSquares	$f(x) = \sum_{i=1}^{n} i x_i^2$
5	$[-1.28, 1.28]$	30	US	Quartic	$f(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$
6	$[-4.5, 4.5]$	5	UN	Beale	$f(x) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$
7	$[-100, 100]$	2	UN	Easom	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$
8	$[-10, 10]$	2	UN	Matyas	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48 x_1 x_2$
9	$[-10, 10]$	4	UN	Colville	$f_{20}(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$
10	$[-D^2, D^2]$	6	UN	Trid6	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
11	$[-D^2, D^2]$	10	UN	Trid10	$f(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
12	$[-5, 10]$	10	UN	Zakharov	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$
13	$[-4, 5]$	24	UN	Powell	$f(x) = \sum_{i=1}^{n/k} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$
14	$[-10, 10]$	30	UN	Schwefel 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
15	$[-100, 100]$	30	UN	Schwefel 1.2	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$
16	$[-30, 30]$	30	UN	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
17	$[-10, 10]$	30	UN	Dixon-Price	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$
18	$[-65.536, 65.536]$	2	MS	Foxholes	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})6}\right]^{-1}$
19	$[-5, 10]$x$[0, 15]$	2	MS	Branin	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$
20	$[-100, 100]$	2	MS	Bohachevsky1	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
21	$[-10, 10]$	2	MS	Booth	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
22	$[-5.12, 5.12]$	30	MS	Rastrigin	$f(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$
23	$[-500, 500]$	30	MS	Schwefel	$f(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$
24	$[0, \pi]$	2	MS	Michalewicz2	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(i x_i^2/\pi))^{2m}$ $m = 10$
25	$[0, \pi]$	5	MS	Michalewicz5	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(i x_i^2/\pi))^{2m}$ $m = 10$
26	$[0, \pi]$	10	MS	Michalewicz10	$f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(i x_i^2/\pi))^{2m}$ $m = 10$

ABC Settings: Except common parameters (population number and maximum evaluation number), the basic ABC used in this study employs only one control parameter, which is called *limit*. A food source will not be exploited anymore and is

Table 2
Benchmark functions used in experiments 1. D: Dimension, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable.

No	Range	D	C	Function	Formulation
27	$[-100,100]$	2	MN	Schaffer	$f(x)=0.5+\frac{\sin^2\left(\sqrt{x_1^2+x_2^2}\right)-0.5}{(1+0.001(x_1^2+x_2^2))^2}$
28	$[-5,5]$	2	MN	Six Hump Camel Back	$f(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x1x2-4x_2^2+4x_2^4$
29	$[-100,100]$	2	MN	Bohachevsky2	$f(x)=x_1^2+2x_2^2-0.3cos(3\pi x_1)(4\pi x_2)+0.3$
30	$[-100,100]$	2	MN	Bohachevsky3	$f(x)=x_1^2+2x_2^2-0.3cos(3\pi x_1+4\pi x_2)+0.3$
31	$[-10,10]$	2	MN	Shubert	$f(x)=\left(\sum_{i=1}^{5} i\cos((i+1)x_1+i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2+i)\right)$
32	$[-2,2]$	2	MN	GoldStein-Price	$f(x)=\left[\begin{array}{c}1+(x_1+x_2+1)^2\\ (19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\end{array}\right]$ $\left[\begin{array}{c}30+(2x_1-3x_2)^2\\ (18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\end{array}\right]$
33	$[-5,5]$	4	MN	Kowalik	$f(x)=\sum_{i=1}^{11}\left(a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right)^2$
34	$[0,10]$	4	MN	Shekel5	$f(x)=-\sum_{i=1}^{5}[(x-a_i)(x-a_i)^T+c_i]^{-1}$
35	$[0,10]$	4	MN	Shekel7	$f(x)=-\sum_{i=1}^{7}[(x-a_i)(x-a_i)^T+c_i]^{-1}$
36	$[0,10]$	4	MN	Shekel10	$f(x)=-\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T+c_i]^{-1}$
37	$[-D,D]$	4	MN	Perm	$f(x)=\sum_{k=1}^{n}\left[\sum_{i=1}^{n}(i^k+\beta)((x_i/i)^k-1)\right]^2$
38	$[0,D]$	4	MN	PowerSum	$f(x)=\sum_{k=1}^{n}\left[\left(\sum_{i=1}^{n}x_i^k\right)-b_k\right]^2$
39	$[0,1]$	3	MN	Hartman3	$f(x)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right]$
40	$[0,1]$	6	MN	Hartman6	$f(x)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right]$
41	$[-600,600]$	30	MN	Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$
42	$[-32,32]$	30	MN	Ackley	$f(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$
43	$[-50,50]$	30	MN	Penalized	$f(x)=\frac{\pi}{n}\left\{\begin{array}{c}10\sin^2(\pi y_1)\\ +\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]\\ +(y_n-1)^2\end{array}\right\}$ $+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{1}{4}(x_i+1)$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\leqslant x_i\leqslant a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$
44	$[-50,50]$	30	MN	Penalized2	$f(x)=0.1\left\{\begin{array}{c}\sin^2(\pi x_1)+\\ \sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+\\ (x_n-1)^2[1+\sin^2(2\pi x_n)]\end{array}\right\}+$ $\sum_{i=1}^{n}u(x_i,5,100,4)$

Table 3
Benchmark functions used in experiments 1. D: Dimension, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable.

No	Range	D	C	Function	Formulation
45	$[0,10]$	2	MN	Langerman2	$f(x)=-\sum_{i=1}^{m}c_i\left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\right)$
46	$[0,10]$	5	MN	Langerman5	$f(x)=-\sum_{i=1}^{m}c_i\left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\cos(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2)\right)$
47	$[0,10]$	10	MN	Langerman10	$f(x)=-\sum_{i=1}^{m}c_i\left(\exp\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\right)$
48	$[-\pi,\pi]$	2	MN	FletcherPowell2	$f(x)=\sum_{i=1}^{n}(A_i-B_i)^2$ $A_i=\sum_{j=1}^{n}(a_{ij}\sin\alpha_j+b_{ij}\cos\alpha_j)$ $B_i=\sum_{j=1}^{n}(a_{ij}\sin x_j+b_{ij}\cos x_j)$
49	$[-\pi,\pi]$	5	MN	FletcherPowell5	$f(x)=\sum_{i=1}^{n}(A_i-B_i)^2$ $A_i=\sum_{j=1}^{n}(a_{ij}\sin\alpha_j+b_{ij}\cos\alpha_j)$ $B_i=\sum_{j=1}^{n}(a_{ij}\sin x_j+b_{ij}\cos x_j)$
50	$[-\pi,\pi]$	10	MN	FletcherPowell10	$f(x)=\sum_{i=1}^{n}(A_i-B_i)^2$ $A_i=\sum_{j=1}^{n}(a_{ij}\sin\alpha_j+b_{ij}\cos\alpha_j)$ $B_i=\sum_{j=1}^{n}(a_{ij}\sin x_j+b_{ij}\cos x_j)$

assumed to be abandoned when *limit* is exceeded for the source. This means that the solution of which "trial number" exceeds the *limit* value cannot be improved anymore. We defined a relation (9) for the *limit* value using the dimension of the problem and the colony size:

$$limit = (SN * D) \tag{9}$$

where D is the dimension of the problem and SN is the number of food sources or employed bees.

Table 4
a and c parameters of Langerman function.

i	$a_{ij}, j = 1, \ldots, 10$										c_i
1	9.681	0.667	4.783	9.095	3.517	9.325	6.544	0.211	5.122	2.020	0.806
2	9.400	2.041	3.788	7.931	2.882	2.672	3.568	1.284	7.033	7.374	0.517
3	8.025	9.152	5.114	7.621	4.564	4.711	2.996	6.126	0.734	4.982	1.5
4	2.196	0.415	5.649	6.979	9.510	9.166	6.304	6.054	9.377	1.426	0.908
5	8.074	8.777	3.467	1.863	6.708	6.349	4.534	0.276	7.633	1.567	0.965
6	7.650	5.658	0.720	2.764	3.278	5.283	7.474	6.274	1.409	8.208	0.669
7	1.256	3.605	8.623	6.905	0.584	8.133	6.071	6.888	4.187	5.448	0.524
8	8.314	2.261	4.224	1.781	4.124	0.932	8.129	8.658	1.208	5.762	0.902
9	0.226	8.858	1.420	0.945	1.622	4.698	6.228	9.096	0.972	7.637	0.531
10	7.305	2.228	1.242	5.928	9.133	1.826	4.060	5.204	8.713	8.247	0.876
11	0.652	7.027	0.508	4.876	8.807	4.632	5.808	6.937	3.291	7.016	0.462
12	2.699	3.516	5.874	4.119	4.461	7.496	8.817	0.690	6.593	9.789	0.491
13	8.327	3.897	2.017	9.570	9.825	1.150	1.395	3.885	6.354	0.109	0.463
14	2.132	7.006	7.136	2.641	1.882	5.943	7.273	7.691	2.880	0.564	0.714
15	4.707	5.579	4.080	0.581	9.698	8.542	8.077	8.515	9.231	4.670	0.352
16	8.304	7.559	8.567	0.322	7.128	8.392	1.472	8.524	2.277	7.826	0.869
17	8.632	4.409	4.832	5.768	7.050	6.715	1.711	4.323	4.405	4.591	0.813
18	4.887	9.112	0.170	8.967	9.693	9.867	7.508	7.770	8.382	6.740	0.811
19	2.440	6.686	4.299	1.007	7.008	1.427	9.398	8.480	9.950	1.675	0.828
20	6.306	8.583	6.084	1.138	4.350	3.134	7.853	6.061	7.457	2.258	0.964
21	0.652	2.343	1.370	0.821	1.310	1.063	0.689	8.819	8.833	9.070	0.789
22	5.558	1.272	5.756	9.857	2.279	2.764	1.284	1.677	1.244	1.234	0.360
23	3.352	7.549	9.817	9.437	8.687	4.167	2.570	6.540	0.228	0.027	0.369
24	8.798	0.880	2.370	0.168	1.701	3.680	1.231	2.390	2.499	0.064	0.992
25	1.460	8.057	1.336	7.217	7.914	3.615	9.981	9.198	5.292	1.224	0.332
26	0.432	8.645	8.774	0.249	8.081	7.461	4.416	0.652	4.002	4.644	0.817
27	0.679	2.800	5.523	3.049	2.968	7.225	6.730	4.199	9.614	9.229	0.632
28	4.263	1.074	7.286	5.599	8.291	5.200	9.214	8.272	4.398	4.506	0.883
29	9.496	4.830	3.150	8.270	5.079	1.231	5.731	9.494	1.883	9.732	0.608
30	4.138	2.562	2.532	9.661	5.611	5.500	6.886	2.341	9.699	6.500	0.326

Table 5
A parameter of Fletcher–Powell function.

i	$A_{ij}, j = 1, \ldots, 20$																			
1	−79	56	−62	−9	92	48	−22	−34	−39	−40	−95	−69	−20	−66	−98	−66	−67	37	−83	−45
2	91	−9	−18	−59	99	−45	88	−14	−29	26	71	−65	19	45	88	18	−11	−81	−10	42
3	−38	8	−12	−73	40	26	−64	29	−82	−32	−89	−3	88	98	53	58	45	−39	34	−23
4	−78	−18	−49	65	66	−40	88	−95	−57	10	−98	−11	−16	−55	33	84	21	−43	45	100
5	−1	−43	93	−18	−76	−68	−42	22	46	−14	69	27	−12	−26	57	−13	0	1	56	17
6	34	−96	26	−56	−36	−85	−62	13	93	78	−43	96	77	65	−34	−52	82	18	−59	−55
7	52	−46	−69	99	−47	−72	−11	55	−55	91	−30	7	−35	23	−20	55	61	−39	−58	13
8	81	47	35	55	67	−13	33	14	83	−42	8	−45	−44	12	100	−9	−33	−11	21	14
9	5	−43	−45	46	56	−94	−62	52	66	55	−86	−29	−52	−71	−91	−46	27	−27	6	67
10	−50	66	−47	−75	89	−16	82	6	−85	−62	−30	31	−7	−75	−26	−24	46	−95	−71	−57
11	24	98	−50	68	−97	−64	−24	81	−59	−7	85	−92	2	61	52	−59	−91	74	−99	−95
12	−30	−63	−32	−90	−35	44	−64	57	27	87	−70	−39	−18	−89	99	40	14	−58	−5	−42
13	56	3	88	38	−14	−15	84	−9	65	−20	−75	−37	74	66	−44	72	74	90	−83	−40
14	84	1	73	43	84	−99	−35	24	−78	−58	47	−83	94	−86	−65	63	−22	65	50	−40
15	−21	−8	−48	68	−91	17	−52	−99	−23	43	−8	−5	−98	−17	−62	−79	60	−18	54	74
16	35	93	−98	−88	−8	64	15	69	−65	−86	58	−44	−9	−94	68	−27	−79	−67	−35	−56
17	−91	73	51	68	96	49	10	−13	−6	−23	50	−89	19	−67	36	−97	0	3	1	39
18	53	66	23	10	−33	62	−73	22	−65	37	−83	−65	59	−51	−56	98	−57	−11	−48	88
19	83	48	67	27	91	−33	−90	−34	39	−36	−68	17	−7	14	11	−10	96	98	−32	56
20	52	−52	−5	19	−25	15	−1	−11	8	−70	−4	−7	−4	−6	48	88	13	−56	85	−65

4.2. Benchmark functions

In the field of evolutionary computation, it is common to compare different algorithms using a large test set, especially when the test involves function optimization. Attempting to design a perfect test set where all the functions are present in order to determine whether an algorithm is better than another for every function, is a fruitless task. That is the reason why, when an algorithm is evaluated, we must look for the kind of problems where its performance is good, in order to characterize the type of problems for which the algorithm is suitable [45]. We used 50 benchmark problems in order to test the performance of the GA, DE, PSO and the ABC algorithms. This set is large enough to include many different kinds of problems such as unimodal, multimodal, regular, irregular, separable, non-separable and multidimensional. Initial range, formulation, characteristics and the dimensions of these problems are listed in Tables 1–3.

If a function has more than one local optimum, this function is called multimodal. Multimodal functions are used to test the ability of algorithms getting rid of local minima. If the exploration process of an algorithm is poor that it cannot search whole space efficiently, this algorithm gets stuck at the local minima. Functions having flat surfaces are difficult for algorithms as well as multimodal functions since the flatness of the function does not give the algorithm any information to direct the search process towards the minima (Stepint, Matyas, PowerSum). Another group of test problems is separable/nonseparable functions. A p-variable separable function can be expressed as the sum of p functions of one variable. Non-separable functions

Table 6
B parameter of Fletcher–Powell function.

i	$B_{ij}, j = 1,\ldots,21$																				
1	−65	−11	76	78	30	93	−86	−99	−37	52	−20	−10	−97	−71	16	9	−99	−84	90	−18	−94
2	59	67	49	−45	52	−33	−34	29	−39	−80	22	7	3	−19	−15	7	−83	−4	84	−60	−4
3	21	−23	−80	86	86	−30	39	−73	−91	5	83	−2	−45	−54	−81	−8	14	83	73	45	32
4	−91	−75	20	−64	−15	17	−89	36	−49	−2	56	−6	76	56	2	−68	−59	−70	48	2	24
5	−79	99	−31	−8	−67	−72	−43	−55	76	−57	1	−58	3	−59	30	32	57	29	66	50	−80
6	−89	−35	−55	75	15	−6	−53	−56	−96	87	−90	−93	52	−86	−38	−55	−53	94	98	4	−79
7	−76	45	74	12	−12	−69	2	71	75	−60	−50	23	0	6	44	−82	37	91	84	−15	−63
8	−50	−88	93	68	10	−13	84	−21	65	14	4	92	11	67	−18	−51	4	21	−38	75	−59
9	−23	−95	99	62	−37	96	27	69	−64	−92	−12	87	93	−19	−99	−92	−34	−77	17	−72	29
10	−5	−57	−30	−6	−96	75	25	−6	96	77	−35	−10	82	82	97	−39	−65	−8	34	72	65
11	85	−9	−14	27	−45	70	55	26	−87	−98	−25	−12	60	−45	−24	−42	−88	−46	−95	53	28
12	80	−47	38	−6	43	−59	91	−41	90	−63	11	−54	33	−61	74	96	21	−77	−58	−75	−9
13	−66	−98	−4	96	−11	88	−99	5	5	58	−53	52	−98	−97	50	49	97	−62	79	−10	−80
14	80	−95	82	5	−68	−54	64	−2	5	10	85	−33	−54	−30	−65	58	40	−21	−84	−66	−11
15	94	85	−31	37	−25	60	55	−13	48	−23	−50	84	−71	54	47	18	−67	−30	5	−46	53
16	−29	54	−10	−68	−54	−24	−16	21	32	33	−27	48	37	−61	97	45	−90	87	−95	85	67
17	76	−11	−48	38	−7	86	−55	51	26	8	−96	99	69	−84	41	78	−53	4	29	38	16
18	−8	48	95	47	39	−11	−72	−95	−17	33	65	96	−52	−17	−22	−15	−91	−41	−16	23	14
19	92	87	63	−63	−80	96	−62	71	−58	17	−89	−35	−96	−79	7	46	−74	88	93	−44	52
20	−21	35	16	−17	54	−22	−93	27	88	0	−67	94	−24	−30	−90	−5	−48	45	−90	32	−81
21	−86	31	−80	−79	−5	11	−20	9	52	−38	67	64	−49	23	−86	39	−97	76	10	81	20

Table 7
α parameter of Fletcher–Powell function.

$\alpha_j, j = 1,\ldots,20$
−2.7910
2.5623
−1.0429
0.5097
−2.8096
1.1883
2.0771
−2.9926
0.0715
0.4142
−2.5010
1.7731
1.6473
0.4934
2.1038
−1.9930
0.3813
−2.2144
−2.5572
2.9449

cannot be written in this form. These functions have interrelation among their variables. Therefore, non-separable functions are more difficult than the separable functions. The dimensionality of the search space is an important issue with the problem [45]. In some functions, global minima is very small when compared to whole search space (Easom, Michalewicz ($m = 10$), Powell) or global minimum is very close to the local ones (Perm, Kowalik and Schaffer). As for multimodal functions, if the algorithm cannot explore the search space properly and cannot keep up the direction changes in the functions having narrow curving valley (Beale, Colville, Rosenbrock), it fails in these kinds of problems. Another problem that algorithms suffer is scaling problem with a difference of many magnitude orders between the domain and the function hypersurface [46] (Goldstein-Price, Trid). Fletcher-Powell and Langerman functions are non-symmetrical and their local optima are randomly distributed. In this way, the objective function has no implicit symmetry advantages that might simplify optimization for certain algorithms. Fletcher–Powell function achieves the random distribution of the optima choosing the values of the matrixes [45]. Quartic function is padded with noise. Expected fitness depends on the random noise generator (gaussian or uniform). The random noise makes sure that the algorithm never gets the same value on the same point. Algorithms that do not do well on this test function will do poorly on noisy data. Step has one minimum and it is a discontinuous function. Penalized functions are difficult due to the combinations of different periods of the sine function [47]. In Tables 1–3 characteristics of each function are given under the column titled *C*. In this column, *M* means that the function is multimodal, while *U* means that the function is unimodal. If the function is separable, abbreviation *S* is used to indicate this specification. Letter *N* refers that the function is non-separable. In our set, as seen from Tables 1–3, 33 functions are multimodal, 17 functions are unimodal, 14 functions are separable and 36 functions are non-separable. The increment in the dimension of function increases the difficulty. Dimensions of the problems we used can be found in Tables 1 and 3 under the column titled *D*.

Table 8
a parameter of FoxHoles function.

j	$a_{ij}, i = 1, 2$	
1	−32	−32
2	−16	−32
3	0	−32
4	16	−32
5	32	−32
6	−32	−16
7	−16	−16
8	0	−16
9	16	−16
10	32	−16
11	−32	0
12	−16	0
13	0	0
14	16	0
15	32	0
16	−32	16
17	−16	16
18	0	16
19	16	16
20	32	16
21	−32	32
22	−16	32
23	0	32
24	16	32
25	32	32

Table 9
a and *b* parameters of Kowalik function.

i	a_i	b_i^{-1}
1	0.1957	0.25
2	0.1947	0.5
3	0.1735	1
4	0.1600	2
5	0.0844	4
6	0.0627	6
7	0.0456	8
8	0.0342	10
9	0.0323	12
10	0.0235	14
11	0.0246	16

Table 4 presents a and c parameters of Langerman function, Tables 5–7 present a, b and α parameters of Fletcher–Powell function and Tables 8–12 present the parameters in Foxholes, Kowalik, Shekel family, Hartman3 and Hartman6 functions, respectively.

4.3. Results

In Experiments 1, we compared the GA, PSO, DE and ABC algorithms on a large set of functions described in the previous section and are listed in Tables 1–3. Each of the experiments in this section was repeated 30 times with different random seeds and the mean best values produced by the algorithms have been recorded. In order to make comparison clear, the values below 10^{-12} are assumed to be 0. The mean best values, standard deviations and standard errors of means are given in Tables 13–15.

In order to analyze the results whether there is significance between the results of each algorithm, we performed t-test on pairs of algorithms. We calculated p-values on each function but did not directly compare this p-value to α to determine the significance because probabilistic algorithms can produce significantly different results even if they start to run random values coming from the same distribution [48]. Hence, we adopted Modified Bonferroni Correction in order to address this issue. Modified Bonferroni Correction ranks the calculated p-values ascending and significance level α is gathered by dividing 0.05 level by the inverse rank. If $p > \alpha$, it is said to be significantly difference between algorithms on the function. These statistical tests are given in Tables 16–18.

From the results in Tables 13–15, on functions lying on flat surfaces such as Stepint and Matyas, all algorithms have equal performance, since all algorithms have operators providing diversity and variable step sizes. For producing mutant vector, the DE algorithm uses difference of randomly selected solution vectors as in (1), the PSO algorithm uses random coefficients as multiplier for calculating different vectors as in (5), for the GA, crossover operator provides this perturbation. In the ABC algorithm, an operator as in (7) is employed to perform the perturbation. Among these flat functions, on PowerSum function none of the algorithms have produced the optima but the result of the DE algorithm is better than that of others.

From Tables 16–18, for all algorithms, there is no significance on Goldstein-Price, Stepint, Beale, Easom, Matyas, Foxholes, Branin, Bohachevsky1, Booth, Six Hump Camel Back, Bohachevsky3, Shubert, and Fletcher-Powell2 functions. From Table 16,

Table 10
a and c parameters of Shekel functions.

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

Table 11
a, c and p parameters of 3-parameter Hartman function.

i	$a_{ij}, j = 1,2,3$			c_i	$p_{ij}, j = 1,2,3$		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.03815	0.5743	0.8828

Table 12
a, c and p parameters of 6-parameter Hartman function.

i	$a_{ij}, j = 1,\ldots,6$						c_i	$p_{ij}, j = 1,\ldots,6$					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

Table 13
Statistical results of 30 runs obtained by GA, PSO, DE and ABC algorithms. D: Dimension, Mean: Mean of the Best Values, StdDev: Standard Deviation of the Best Values, SEM: Standard Error of Means.

No	Range	D	Function	Min.		GA	PSO	DE	ABC
1	$[-5.12,5.12]$	5	Stepint	0	Mean	0	0	0	0
					StdDev	0	0	0	0
					SEM	0	0	0	0
2	$[-100,100]$	30	Step	0	Mean	1.17E+03	0	0	0
					StdDev	76.561450	0	0	0
					SEM	13.978144	0	0	0
3	$[-100,100]$	30	Sphere	0	Mean	1.11E+03	0	0	0
					StdDev	74.214474	0	0	0
					SEM	13.549647	0	0	0
4	$[-10,10]$	30	SumSquares	0	Mean	1.48E+02	0	0	0
					StdDev	12.4092893	0	0	0
					SEM	2.265616	0	0	0
5	$[-1.28,1.28]$	30	Quartic	0	Mean	0.1807	0.00115659	0.0013633	0.0300166
					StdDev	0.027116	0.000276	0.000417	0.004866
					SEM	0.004951	5.04E–05	7.61E–05	0.000888
6	$[-4.5,4.5]$	2	Beale	0	Mean	0	0	0	0
					StdDev	0	0	0	0
					SEM	0	0	0	0
7	$[-100,100]$	2	Easom	−1	Mean	−1	−1	−1	−1
					StdDev	0	0	0	0
					StdDev	0	0	0	0
8	$[-10,10]$	2	Matyas	0	Mean	0	0	0	0
					StdDev	0	0	0	0
					SEM	0	0	0	0
9	$[-10,10]$	4	Colville	0	Mean	0.014938	0	0.0409122	0.0929674
					StdDev	0.007364	0	0.081979	0.066277
					SEM	0.001344	0	0.014967	0.012100
10	$[-D^2,D^2]$	6	Trid6	−50	Mean	−49.9999	−50	−50	−50
					StdDev	2.25E–5	0	0	0
					StdDev	4.11E–06	0	0	0
11	$[-D^2,D^2]$	10	Trid10	−210	Mean	−209.476	−210	−210	−210
					StdDev	0.193417	0	0	0
					SEM	0.035313	0	0	0
12	$[-5,10]$	10	Zakharov	0	Mean	0.013355	0	0	0.0002476
					StdDev	0.004532	0	0	0.000183
					SEM	0.000827	0	0	3.34E–05
13	$[-4,5]$	24	Powell	0	Mean	9.703771	0.00011004	2.17E–07	0.0031344
					StdDev	1.547983	0.000160	1.36E–7	0.000503
					SEM	0.282622	2.92E–05	2.48E–08	9.18E–05
14	$[-10,10]$	30	Schwefel 2.22	0	Mean	11.0214	0	0	0
					StdDev	1.386856	0	0	0
					SEM	0.253204	0	0	0
15	$[-100,100]$	30	Schwefel 1.2	0	Mean	7.40E+03	0	0	0
					StdDev	1.14E+03	0	0	0
					SEM	208.1346	0	0	0
16	$[-30,30]$	30	Rosenbrock	0	Mean	1.96E+05	15.088617	18.203938	0.0887707
					StdDev	3.85E+04	24.170196	5.036187	0.077390
					SEM	7029.106155	4.412854	0.033333	0.014129
17	$[-10,10]$	30	Dixon-Price	0	Mean	1.22E+03	0.66666667	0.6666667	0
					StdDev	2.66E+02	E–8	E–9	0
					SEM	48.564733	1.8257e–009	1.8257e–0010	0
18	$[-65.536,65.536]$	2	Foxholes	0.998	Mean	0.998004	0.99800393	0.9980039	0.9980039
					StdDev	0	0	0	0
					SEM	0	0	0	0
19	$[-5,10]\times[0,15]$	2	Branin	0.398	Mean	0.397887	0.39788736	0.3978874	0.3978874
					StdDev	0	0	0	0
					SEM	0	0	0	0
20	$[-100,100]$	2	Bohachevsky1	0	Mean	0	0	0	0
					StdDev	0	0	0	0
					SEM	0	0	0	0

on 20 functions there is no significant difference between GA and ABC algorithms but on 28 functions ABC is better than GA while GA is better than ABC on only 2 functions. From Table 17, on 22 functions ABC and PSO algorithms show equal performance. On 4 of the remaining 28 functions, PSO performs better than ABC while ABC performs better on 24 functions. From Table 18, DE and ABC algorithms show equal performance on 37 functions. On 5 functions DE is better than ABC, while on 8 functions ABC performs better.

Table 14
Table 13 continued. Statistical results of 30 runs obtained by GA, PSO, DE and ABC algorithms. D: Dimension, Mean: Mean of the Best Values, StdDev: Standard Deviation of the Best Values, SEM: Standard Error of Means.

No	Range	D	Function	Min.		GA	PSO	DE	ABC
21	[−10,10]	2	Booth	0	Mean	0	0	0	0
					StdDev	0	0	0	0
					SEM	0	0	0	0
22	[−5.12,5.12]	30	Rastrigin	0	Mean	52.92259	43.9771369	11.716728	0
					StdDev	4.564860	11.728676	2.538172	0
					SEM	0.833426	2.141353	0.463405	0
23	[−500,500]	30	Schwefel	−12569.5	Mean	−11593.4	−6909.1359	−10266	−12569.487
					StdDev	93.254240	457.957783	521.849292	0
					SEM	17.025816	83.611269	95.276209	0
24	$[0, \pi]$	2	Michalewicz2	−1.8013	Mean	−1.8013	−1.5728692	−1.801303	−1.8013034
					StdDev	0	0.119860	0	0
					SEM	0	0.021883	0	0
25	$[0, \pi]$	5	Michalewicz5	−4.6877	Mean	−4.64483	−2.4908728	−4.683482	−4.6876582
					StdDev	0.097850	0.256952	0.012529	0
					SEM	0.017865	0.046913	0.002287	0
26	$[0, \pi]$	10	Michalewicz10	−9.6602	Mean	−9.49683	−4.0071803	−9.591151	−9.6601517
					StdDev	0.141116	0.502628	0.064205	0
					SEM	0.025764	0.091767	0.011722	0
27	[−100,100]	2	Schaffer	0	Mean	0.004239	0	0	0
					StdDev	0.004763	0	0	0
					SEM	0.000870	0	0	0
28	[−5,5]	2	6Hump CamelBack	−1.03163	Mean	−1.03163	−1.0316285	−1.031628	−1.0316285
					StdDev	0	0	0	0
					SEM	0	0	0	0
29	[−100,100]	2	Bohachevsky2	0	Mean	0.06829	0	0	0
					StdDev	0.078216	0	0	0
					SEM	0.014280	0	0	0
30	[−100,100]	2	Bohachevsky3	0	Mean	0	0	0	0
					StdDev	0	0	0	0
					SEM	0	0	0	0
31	[−10,10]	2	Shubert	−186.73	Mean	−186.731	−186.73091	−186.7309	−186.73091
					StdDev	0	0	0	0
					SEM	0	0	0	0
32	[−2,2]	2	GoldStein-Price	3	Mean	5.250611	3	3	3
					StdDev	5.870093	0	0	0
					SEM	1.071727	0	0	0
33	[−5,5]	4	Kowalik	0.00031	Mean	0.005615	0.00049062	0.0004266	0.0004266
					StdDev	0.008171	0.000366	0.000273	6.04E−5
					SEM	0.001492	6.68E−05	4.98E−05	1.10E−05
34	[0,10]	4	Shekel5	−10.15	Mean	−5.66052	−2.0870079	−10.1532	−10.1532
					StdDev	3.866737	1.178460	0	0
					SEM	0.705966	0.215156	0	0
35	[0,10]	4	Shekel7	−10.4	Mean	−5.34409	−1.9898713	−10.40294	−10.402941
					StdDev	3.517134	1.420602	0	0
					SEM	0.642138	0.259365	0	0
36	[0,10]	4	Shekel10	−10.53	Mean	−3.82984	−1.8796753	−10.53641	−10.53641
					StdDev	2.451956	0.432476	0	0
					SEM	0.447664	0.078959	0	0
37	[−D,D]	4	Perm	0	Mean	0.302671	0.03605158	0.0240069	0.0411052
					StdDev	0.193254	0.048927	0.046032	0.023056
					SEM	0.035283	0.008933	0.008404	0.004209
38	[0,D]	4	PowerSum	0	Mean	0.010405	11.3904479	0.0001425	0.0029468
					StdDev	0.009077	7.355800	0.000145	0.002289
					SEM	0.001657	1.342979	2.65E−05	0.000418
39	[0,1]	3	Hartman3	−3.86	Mean	−3.86278	−3.6333523	−3.862782	−3.8627821
					StdDev	0	0.116937	0	0
					SEM	0	0.021350	0	0
40	[0,1]	6	Hartman6	−3.32	Mean	−3.29822	−1.8591298	−3.226881	−3.3219952
					StdDev	0.050130	0.439958	0.047557	0
					SEM	0.009152	0.080325	0.008683	0

On Beale function having a curving shape in the vicinity of minimum, all algorithms show equal performance, but on Colville function,a kind of these functions, only the PSO algorithm produces the minimum of the function. On Rosenbrock function extended to 30 parameters, the ABC algorithm shows the highest performance compared to others. On the other hand, these functions are non-separable functions having interdependence among the variables as well as Griewank function.

Table 15
Table 13 continued. Statistical results of 30 runs obtained by GA, PSO, DE and ABC algorithms. D: Dimension, Mean: Mean of the Best Values, StdDev: Standard Deviation of the Best Values, SEM: Standard Error of Means.

No	Range	D	Function	Min.		GA	PSO	DE	ABC
41	[−600,600]	30	Griewank	0	Mean	10.63346	0.01739118	0.0014792	0
					StdDev	1.161455	0.020808	0.002958	0
					SEM	0.212052	0.003799	0.000540	0
42	[−32,32]	30	Ackley	0	Mean	14.67178	0.16462236	0	0
					StdDev	0.178141	0.493867	0	0
					SEM	0.032524	0.090167	0	0
43	[−50,50]	30	Penalized	0	Mean	13.3772	0.0207338	0	0
					StdDev	1.448726	0.041468	0	0
					SEM	0.2645	0.007571	0	0
44	[−50,50]	30	Penalized2	0	Mean	125.0613	0.00767535	0.0021975	0
					StdDev	12.001204	0.016288	0.004395	0
					SEM	2.191110	0.002974	0.000802	0
45	[0,10]	2	Langerman2	−1.08	Mean	−1.08094	−0.679268	−1.080938	−1.0809384
					StdDev	0	0.274621	0	0
					SEM	0	0.050139	0	0
46	[0,10]	5	Langerman5	−1.5	Mean	−0.96842	−0.5048579	−1.499999	−0.938150
					StdDev	0.287548	0.213626	0	0.000208
					SEM	0.052499	0.039003	0	3.80E−05
47	[0,10]	10	Langerman10	NA	Mean	−0.63644	−0.0025656	−1.0528	−0.4460925
					StdDev	0.374682	0.003523	0.302257	0.133958
					SEM	0.068407	0.000643	0.055184	0.024457
48	$[-\pi,\pi]$	2	FletcherPowell2	0	Mean	0	0	0	0
					StdDev	0	0	0	0
					SEM	0	0	0	0
49	$[-\pi,\pi]$	5	FletcherPowell5	0	Mean	0.004303	1457.88344	5.988783	0.1735495
					StdDev	0.009469	1269.362389	7.334731	0.068175
					SEM	0.001729	231.752805	1.339133	0.012447
50	$[-\pi,\pi]$	10	FletcherPowell10	0	Mean	29.57348	1364.45555	781.55028	8.2334401
					StdDev	16.021078	1325.379655	1048.813487	8.092742
					SEM	2.925035	1325.379655	241.980111	1.477526

On non-symmetrical Langerman functions, for the 2-parameter case ABC and DE show equal performance, while for 5 and 10-parameter cases, the DE algorithm performs the best.

Another conclusion that can be drawn is that the efficiency of ABC becomes much more clearer as the number of variables increases. As seen from Tables 1–3, in the experiments, there are 14 functions with 30 variables. ABC outperforms DE on 6, PSO on 8 and GA on 14 of these 14 functions. Four of the functions on which ABC and DE are unsuccessful are unimodal (Colville, Zakharov, Powell, Quartic for ABC, Colville, Quartic, Rosenbrock, Dixon-Price for DE). ABC is unsuccessful on 5 multimodal functions, while DE is unsuccessful on 9 multimodal functions. Consequently, ABC is more successful and the most robust on multimodal functions included in the set respect to DE, PSO and GA.

As mentioned in Section 4, in the experiments, a GA using binary representation was employed. It should be noted that a GA using floating point representation could be faster and more consistent from run to run and could provide higher precision for some specific problems [49].

As seen from Fig. 2, overall evaluation can be made by using the mean absolute error values produced by the algorithms. In the calculation of mean absolute errors, firstly the absolute differences between the values found by the algorithms and the optimum of the corresponding functions were computed. Secondly, the total absolute errors were found for each algorithm. Finally, the mean absolute error was calculated by dividing the total absolute error by the number of test functions. The minimum of the Langerman10 function is not available. Therefore, the mean absolute error was calculated for 49 functions. Since there is a big difference between the mean absolute error value of the ABC algorithm and the other algorithms, the graph in Fig. 2 demonstrates the logarithmic error values obtained for the algorithms. It is clear that the ABC has the smallest mean absolute error. It means that when the results of all functions are evaluated together, the ABC algorithm is the most successful algorithm.

5. ABC vs. evolution strategies

5.1. Experiments 2

5.1.1. Settings of Experiments 2

The experiments in this section constitute the comparison of the ABC algorithm versus Evolution Strategies. The versions of the Evolution Strategies include CES, FES, CMA-ES and ESLAT of which the results were taken from the work presented in [41]. For each experiment, ABC algorithm was run 50 times and it was terminated when it reached the maximum number

Table 16
Significance test for GA and ABC. t: *t*-value of student *t*-test, SED: Standard Error of Difference, p: *p*-value calculated for *t*-value, R: Rank of *p*-value, I.R.: Inverse Rank of *p*-value, Sign: Significance.

No	Function	t	SED	p	R.	I.R.	New α	Sign.
42	Ackley	451.107	0.033	0	1	50	0.001	ABC
2	Step	83.7021	13.978	0	2	49	0.00102	ABC
3	sphere	81.921	13.55	0	3	48	0.001042	ABC
4	SumSquares	65.3244	2.266	0	4	47	0.001064	ABC
22	Rastrigin	63.5001	0.833	0	5	46	0.001087	ABC
23	Schwefel	57.3298	17.026	0	6	45	0.001111	ABC
44	Penalized2	57.0767	2.191	0	7	44	0.001136	ABC
43	Penalized	50.5754	0.264	0	8	43	0.001163	ABC
41	Griewank	50.1456	0.212	0	9	42	0.00119	ABC
14	Schwefel2.22	43.5277	0.253	0	10	41	0.00122	ABC
15	Schwefel1.2	35.5539	208.135	0	11	40	0.00125	ABC
13	Powell	34.3237	0.283	0	12	39	0.001282	ABC
5	Quartic	29.9584	0.005	0	13	38	0.001316	ABC
16	Rosenbrock	27.884	7029.106	0	14	37	0.001351	ABC
17	Dixon-Price	25.1211	48.565	0	15	36	0.001389	ABC
10	Trid6	24.3432	0	0	16	35	0.001429	ABC
12	Zakharov	15.8047	0.001	0	17	34	0.001471	ABC
36	Shekel10	14.9813	0.448	0	18	33	0.001515	ABC
11	Trid10	14.8387	0.035	0	19	32	0.001563	ABC
49	FletcherPowell5	13.4681	0.013	0	20	31	0.001613	GA
35	Shekel7	7.8781	0.642	0	21	30	0.001667	ABC
37	Perm	7.683	0.036	0	22	29	0.001724	ABC
50	FletcherPowell10	6.512	3.277	0	23	28	0.001786	ABC
34	Shekel5	6.3639	0.706	0	24	27	0.001852	ABC
26	Michalewicz10	6.3391	0.026	0	25	26	0.001923	ABC
27	Schaffer	5.1869	0.008	0.000003	26	25	0.002	ABC
29	Bohachevsky2	4.7821	0.014	0.000001	27	24	0.002083	ABC
38	PowerSum	4.3638	0.002	0.000053	28	23	0.002174	ABC
9	Colville	4.3138	0.018	0.000063	29	22	0.002273	GA
33	Kowalik	3.4778	0.001	0.000965	30	21	0.002381	ABC
47	Langerman10	2.6201	0.073	0.011202	31	20	0.0025	–
40	Hartman6	2.5977	0.009	0.011876	32	19	0.002632	–
25	Michalewicz5	2.3973	0.018	0.019758	33	18	0.002778	–
32	GoldStein-Price	2.1	1.072	0.040089	34	17	0.002941	–
46	Langerman5	0.5766	0.052	0.5644	35	16	0.003125	–
1	Stepint	−Inf	0	1	36	15	0.003333	–
6	Beale	−Inf	0	1	37	14	0.003571	–
7	Easom	−Inf	0	1	38	13	0.003846	–
8	Matyas	−Inf	0	1	39	12	0.004167	–
18	Foxholes	−Inf	0	1	40	11	0.004545	–
19	Branin	−Inf	0	1	41	10	0.005	–
20	Bohachevsky1	−Inf	0	1	42	9	0.005556	–
21	Booth	−Inf	0	1	43	8	0.00625	–
24	Michalewicz2	−Inf	0	1	44	7	0.007143	–
28	6HumpCamelBack	−Inf	0	1	45	6	0.008333	–
30	Bohachevsky3	−Inf	0	1	46	5	0.01	–
31	Shubert	−Inf	0	1	47	4	0.0125	–
39	Hartman3	−Inf	0	1	48	3	0.016667	–
45	Langerman	−Inf	0	1	49	2	0.025	–
48	FletcherPowell2	−Inf	0	1	50	1	0.05	–

of evaluations or when it reached the global minima within a gap of 10^{-3} as used in [41]. Settings of the algorithms CES, FES, CMA-ES and ESLAT can be found in [41,36]. In ABC algorithm, the colony size was 20 and the maximum evaluation number was 100.000 for all functions. Value of the *limit* parameter was determined by (9) as in the experiments in Section 4.

5.1.2. Benchmark functions

In Experiments 2, the functions listed in Table 19 were employed. Sphere, Schwefel 1.2, Schwefel 2.21, Schwefel 2.22, Rosenbrock, Step and Quartic functions are unimodal and high dimensional problems. Schwefel, Rastrigin, Ackley, Griewank, Penalized and Penalized 2 are multimodal and high dimensional functions. Foxholes, Kowalik, Six Hump Camel Back, Branin, Goldstein-Price, Hartman family and Shekel Family functions are multimodal and low dimensional functions as mentioned before in Section 4.2.

5.1.3. Results of Experiments 2

In Experiments 2, the performance of ABC algorithm has been compared to those of the Evolution Strategies: CES, FES, CMA-ES and ESLAT. The results of these algorithms, in terms of both solution quality and solution costs, were taken from

Table 17
Significance test for PSO and ABC. t: *t*-value of student *t*-test, SED: Standard Error of Difference, p: *p*-value calculated for *t*-value, R: Rank of *p*-value, I.R.: Inverse Rank of *p*-value, Sign: Significance.

No	Function	t	SED	p	R.	I.R.	New α	Sign.
17	Dixon-Price	3.65E+08	0	0	1	50	0.001	ABC
23	Schwefel	67.6984	83.611	0	2	49	0.00102	ABC
26	Michalewicz10	61.6014	0.092	0	3	48	0.001042	ABC
25	Michalewicz5	46.827	0.047	0	4	47	0.001064	ABC
34	Shekel5	37.4899	0.215	0	5	46	0.001087	ABC
36	Shekel10	33.3766	0.259	0	6	45	0.001111	ABC
35	Shekel7	32.4372	0.259	0	7	44	0.001136	ABC
5	Quartic	32.433	0.001	0	8	43	0.001163	PSO
13	Powell	31.3832	0	0	9	42	0.00119	PSO
22	Rastrigin	20.5371	2.141	0	10	41	0.00122	ABC
40	Hartman6	18.2118	0.08	0	11	40	0.00125	ABC
47	Langerman10	18.1285	0.024	0	12	39	0.001282	ABC
46	Langerman5	11.1093	0.03	0	13	38	0.001316	ABC
39	Hartman3	10.7463	0.021	0	14	37	0.001351	ABC
24	Michalewicz2	10.4387	0.022	0	15	36	0.001389	ABC
38	PowerSum	8.4793	1.343	0	16	35	0.001429	ABC
45	Langerman2	8.0112	0.05	0	17	34	0.001471	ABC
9	Colville	7.683	0.012	0	18	33	0.001515	PSO
12	Zakharov	7.4107	0	0	19	32	0.001563	PSO
49	FletcherPowell5	6.2899	231.753	0	20	31	0.001613	ABC
50	FletcherPowell10	5.6046	241.985	0	21	30	0.001667	ABC
41	Griewank	4.5778	0.004	0.000025	22	29	0.001724	ABC
16	Rosenbrock	3.3991	4.413	0.001228	23	28	0.001786	ABC
43	Penalized	2.7386	0.008	0.008182	24	27	0.001852	–
44	Penalized2	2.581	0.003	0.012403	25	26	0.001923	–
42	Ackley	1.8257	0.09	0.073045	26	25	0.002	–
33	Kowalik	0.9453	0	0.34843	27	24	0.002083	–
37	Perm	0.5118	0.001	0.61073	28	23	0.002174	–
1	Stepint	−Inf	0	1	29	22	0.002273	–
2	Step	−Inf	0	1	30	21	0.002381	–
3	sphere	−Inf	0	1	31	20	0.0025	–
4	SumSquares	−Inf	0	1	32	19	0.002632	–
6	Beale	−Inf	0	1	33	18	0.002778	–
7	Easom	−Inf	0	1	34	17	0.002941	–
8	Matyas	−Inf	0	1	35	16	0.003125	–
10	Trid6	−Inf	0	1	36	15	0.003333	–
11	Trid10	−Inf	0	1	37	14	0.003571	–
14	Schwefel2.22	−Inf	0	1	38	13	0.003846	–
15	Schwefel1.2	−Inf	0	1	39	12	0.004167	–
18	Foxholes	−Inf	0	1	40	11	0.004545	–
19	Branin	−Inf	0	1	41	10	0.005	–
20	Bohachevsky1	−Inf	0	1	42	9	0.005556	–
21	Booth	−Inf	0	1	43	8	0.00625	–
27	Schaffer	−Inf	0	1	44	7	0.007143	–
28	6HumpCamelBack	−Inf	0	1	45	6	0.008333	–
29	Bohachevsky2	−Inf	0	1	46	5	0.01	–
30	Bohachevsky3	−Inf	0	1	47	4	0.0125	–
31	Shubert	−Inf	0	1	48	3	0.016667	–
32	GoldStein-Price	−Inf	0	1	49	2	0.025	–
48	FletcherPowell2	−Inf	0	1	50	1	0.05	–

[41], as mentioned above. Each experiment was repeated 50 times. Mean values and the standard deviations of the function values found after 50 runs are presented in Table 20. When the value found by the algorithm reached the global minima within gap of 10^{-3}, the algorithm was terminated. If the algorithm cannot find the global minima in this precision through the maximum evaluation number, the value found in the last cycle is recorded. Solution costs of these runs are given in Table 21. From the results in this table, it is clear that CMA-ES costs less on 12 functions and CES and ESLAT cost less on 2 functions while ABC costs less on 8 functions. Generally speaking, the cost of CMA-ES is lower than those of ESLAT, CES and ABC for the unimodal functions. This is because CMA-ES is a local method devised for optimal exploitation of local information [38]. When we focus on the success rates given in Table 22, it is seen that the performance of CMA-ES gets worse on multimodal functions. ESLAT has better performance on low dimensional functions than CMA-ES, but on high dimensional and multimodal functions, the performance of ESLAT deteriorates as that of CMA-ES. ABC algorithm achieved best success rate on 20 functions; and on sixteen of these 20 functions it achieved 100% success rate. ABC is successful on both unimodal and multimodal functions. Both CMA-ES and ESLAT have achieved best success rates on 9 functions. On two functions (Rosenbrock, Schewfel 2.21), CMA-ES has better performance than ABC. ABC outperforms CMA-ES on 13 functions (Step, Schewfel,

Table 18
Significance test for DE and ABC. t: *t*-value of student *t*-test, SED: Standard Error of Difference, p: *p*-value calculated for *t*-value, R: Rank of *p*-value, I.R.: Inverse Rank of *p*-value, Sign: Significance.

No	Function	t	SED	p	R.	I.R.	New α	Sign.
17	Dixon-Price	3651483719	0	0	1	50	0.001	ABC
46	Langerman5	14795.0659	0	0	2	49	0.00102	DE
13	Powell	34.1285	0	0	3	48	0.001042	DE
5	Quartic	32.1347	0.001	0	4	47	0.001064	DE
22	Rastrigin	25.284	0.463	0	5	46	0.001087	ABC
23	Schwefel	24.1769	95.276	0	6	45	0.001111	ABC
16	Rosenbrock	19.6993	0.92	0	7	44	0.001136	ABC
40	Hartman6	10.9545	0.009	0	8	43	0.001163	ABC
47	Langerman10	10.0513	0.06	0	9	42	0.00119	DE
38	PowerSum	6.6968	0	0	10	41	0.00122	DE
26	Michalewicz10	5.8863	0.012	0	11	40	0.00125	ABC
49	FletcherPowell5	4.3424	1.339	0.000057	12	39	0.001282	ABC
50	FletcherPowell10	4.0384	191.492	0.00016	13	38	0.001316	ABC
41	Griewank	2.739	0.001	0.008173	14	37	0.001351	–
44	Penalized2	2.7386	0.001	0.008182	15	36	0.001389	–
9	Colville	2.7046	0.019	0.008961	16	35	0.001429	–
25	Michalewicz5	1.8257	0.002	0.073045	17	34	0.001471	–
37	Perm	1.8191	0.009	0.074059	18	33	0.001515	–
33	Kowalik	0	0	1	19	32	0.001563	–
1	Stepint	−Inf	0	1	20	31	0.001613	–
2	Step	−Inf	0	1	21	30	0.001667	–
3	sphere	−Inf	0	1	22	29	0.001724	–
4	SumSquares	−Inf	0	1	23	28	0.001786	–
6	Beale	−Inf	0	1	24	27	0.001852	–
7	Easom	−Inf	0	1	25	26	0.001923	–
8	Matyas	−Inf	0	1	26	25	0.002	–
10	Trid6	−Inf	0	1	27	24	0.002083	–
11	Trid10	−Inf	0	1	28	23	0.002174	–
12	Zakharov	−Inf	0	1	29	22	0.002273	–
14	Schwefel2.22	−Inf	0	1	30	21	0.002381	–
15	Schwefel1.2	−Inf	0	1	31	20	0.0025	–
18	Foxholes	−Inf	0	1	32	19	0.002632	–
19	Branin	−Inf	0	1	33	18	0.002778	–
20	Bohachevsky1	−Inf	0	1	34	17	0.002941	–
21	Booth	−Inf	0	1	35	16	0.003125	–
24	Michalewicz2	−Inf	0	1	36	15	0.003333	–
27	Schaffer	−Inf	0	1	37	14	0.003571	–
28	SixHumpCamelBack	−Inf	0	1	38	13	0.003846	–
29	Bohachevsky2	−Inf	0	1	39	12	0.004167	–
30	Bohachevsky3	−Inf	0	1	40	11	0.004545	–
31	Shubert	−Inf	0	1	41	10	0.005	–
32	GoldStein-Price	−Inf	0	1	42	9	0.005556	–
34	Shekel5	−Inf	0	1	43	8	0.00625	–
35	Shekel7	−Inf	0	1	44	7	0.007143	–
36	Shekel10	−Inf	0	1	45	6	0.008333	–
39	Hartman3	−Inf	0	1	46	5	0.01	–
42	Ackley	−Inf	0	1	47	4	0.0125	–
43	Penalized	−Inf	0	1	48	3	0.016667	–
45	Langerman2	−Inf	0	1	49	2	0.025	–
48	FletcherPowell2	−Inf	0	1	50	1	0.05	–

Rastrigin, Griewank, Penalized, Penalized 2, Foxholes, Kowalik, Goldstein-Price, Hartman 6, Shekel 5, Shekel 7, and Shekel 10). On 7 functions, they have equal performance. On 9 functions, ESLAT has 100% success rate. On 1 function (Rosenbrock), ESLAT shows better performance than ABC while ABC is better than ESLAT on 11 functions (Step, Schwefel, Rastrigin, Griewank, Penalized 2, Foxholes, Kowalik, Hartman 6, Shekel 5, Shekel 7, and Shekel 10). On 9 functions, they perform equally. ABC algorithm has a higher success rate than CMA-ES and ESLAT since it does exploration and exploitation processes together efficiently.

5.2. Experiments 3

5.2.1. Settings

In the experiments of this section, the performance of ABC algorithm has been compared to those of SOM-ES, NG-ES and CMA-ES algorithms. The results of SOM-ES, NG-ES and CMA-ES algorithms were taken from the study described in [38]. The thresholds for satisfactory convergence were fixed to 40, 0.001 and 0.001 for functions 1, 2, and 3, respectively. Maximum

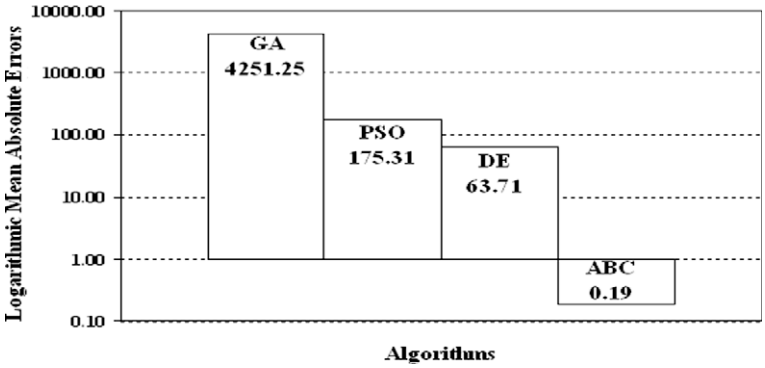

Fig. 2. Mean absolute errors of algorithms in Experiments 1.

Table 19
Benchmark functions used in experiments 2. D: Dimension, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable.

No	Range	D	C	Function	Formulation
1	$[-100,100]$	30	US	Sphere	$f(x)=\sum_{i=1}^{n}x_i^2$
2	$[-10,10]$	30	UN	Schwefel 2.22	$f(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$
3	$[-100,100]$	30	UN	Schwefel 1.2	$f(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$
4	$[-100,100]$	30	UN	Schwefel 2.21	$f(x)=$
5	$[-30,30]$	30	UN	Rosenbrock	$f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$
6	$[-100,100]$	30	US	Step	$f(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$
7	$[-1.28,1.28]$	30	US	Quartic	$f(x)=\sum_{i=1}^{n}ix_i^4+\text{random}[0,1)$
8	$[-500,500]$	30	MS	Schwefel	$f(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$
9	$[-5.12,5.12]$	30	MS	Rastrigin	$f(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$
10	$[-32,32]$	30	MN	Ackley	$f(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$
11	$[-600,600]$	30	MN	Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$
12	$[-50,50]$	30	MN	Penalized	$f(x)=\frac{\pi}{n}\left\{\begin{array}{l}10\sin^2(\pi y_1)\\+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]\\+(y_n-1)^2\end{array}\right\}$ $+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{1}{4}(x_i+1)$ $u(x_i,a,k,m)=\left\{\begin{array}{ll}k(x_i-a)^m, & x_i>a\\0, & -a\leqslant x_i\leqslant a\\k(-x_i-a)^m, & x_i<-a\end{array}\right.$
13	$[-50,50]$	30	MN	Penalized2	$f(x)=0.1\left\{\begin{array}{l}\sin^2(\pi x_1)+\\\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+\\(x_n-1)^2[1+\sin^2(2\pi x_n)]\end{array}\right\}+$ $\sum_{i=1}^{n}u(x_i,5,100,4)$
14	$[-65.536,65.536]$	2	MS	Foxholes	$f(x)=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})6}\right]^{-1}$
15	$[-5,5]$	4	MN	Kowalik	$f(x)=\sum_{i=1}^{11}\left(a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right)^2$
16	$[-5,5]$	2	MN	Six Hump Camel Back	$f(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x1x2-4x_2^2+4x_2^4$
17	$[-5,10]$x$[0,15]$	2	MS	Branin	$f(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2$ $+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$
18	$[-2,2]$	2	MN	GoldStein-Price	$f(x)=\left[\begin{array}{c}1+(x_1+x_2+1)^2\\(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\end{array}\right]$ $\left[\begin{array}{c}30+(2x_1-3x_2)^2\\(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\end{array}\right]$
19	$[0,1]$	3	MN	Hartman 3	$f(x)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right]$
20	$[0,1]$	6	MN	Hartman 6	$f(x)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right]$
21	$[0,10]$	4	MN	Shekel 5	$f(x)=-\sum_{i=1}^{5}[(x-a_i)(x-a_i)^T+c_i]^{-1}$
22	$[0,10]$	4	MN	Shekel 7	$f(x)=-\sum_{i=1}^{7}[(x-a_i)(x-a_i)^T+c_i]^{-1}$
23	$[0,10]$	4	MN	Shekel 10	$f(x)=-\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T+c_i]^{-1}$

Table 20
Results obtained by CES [41], FES [36], ESLAT [41], CMA-ES [41] and ABC algorithms. D: Dimension, SD: Standard Deviation.

No	Function	Range	D	CES		FES		ESLAT		CMA-ES		ABC	
				Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	Sphere	−100,100	30	1.7e−26	1.1e−25	2.5e−4	6.8e−5	2.0e−17	2.9e−17	9.7e−23	3.8e−23	7.57E−04	2.48E−04
2	Schwefel 2.22	−10,10	30	8.1e−20	3.6e−19	6.0e−2	9.6e−3	3.8e−5	1.6e−5	4.2e−11	7.1e−23	8.95E−04	1.27E−04
3	Schwefel 1.2	−100,100	30	337.62	117.14	1.4e−3	5.3e−4	6.1e−6	7.5e−6	7.1e−23	2.9e−23	7.01E−04	2.78E−04
4	Schwefel 2.21	−100,100	30	2.41	2.15	5.5e−3	6.5e−4	0.78	1.64	5.4e−12	1.5e−12	2.72E+00	1.18E+00
5	Rosenbrock	−30,30	30	27.65	0.51	33.28	43.13	1.93	3.35	0.4	1.2	9.36E−01	1.76E+00
6	Step	−100,100	30	0	0	0	0	2.0e−2	0.14	1.44	1.77	0.00E+00	0.00E+00
7	Quartic	−1.28,1.28	30	4.7e−2	0.12	1.2e−2	5.8e−3	0.39	0.22	0.23	8.7e−2	9.06E−02	1.89E−02
8	Schwefel	−500,500	30	−8.00E+93	4.9e+94	−12556.4	32.53	−2.3e+15	5.70E+15	−7637.14	895.6	−12563.67335	2.36E+01
9	Rastrigin	−5.12,5.12	30	13.38	43.15	0.16	0.33	4.65	5.67	51.78	13.56	4.66E−04	3.44E−04
10	Ackley	−32,32	30	6.0e−13	1.7e−12	1.2e−2	1.8e−3	1.8e−8	5.4e−9	6.9e−12	1.3e−12	7.81E−04	1.83E−04
11	Griewank	−600,600	30	6.0e−14	4.2e−13	3.7e−2	5.0e−2	1.4e−3	4.7e−3	7.4e−4	2.7e−3	8.37E−04	1.38E−03
12	Penalized	−50,50	30	1.46	3.17	2.8e−6	8.1e−7	1.5e−12	2.0e−12	1.2e−4	3.4e−2	6.98E−04	2.78E−04
13	Penalized 2	−50,50	30	2.40	0.13	4.7e−5	1.5e−5	6.4e−3	8.9e−3	1.7e−3	4.5e−3	7.98E−04	2.13E−04
14	Foxholes	−65.536,65.536	2	2.20	2.43	1.20	0.63	1.77	1.37	10.44	6.87	9.98E−01	3.21E−04
15	Kowalik	−5,5	4	1.3e−3	6.3e−4	9.7e−4	4.2e−4	8.1e−4	4.1e−4	1.5e−3	4.2e−3	1.18E−03	1.45E−04
16	SixHump	−5,5	2	−1.0310	1.2e−3	−1.0316	6.0e−7	−1.0316	9.7e−14	−1.0316	7.7e−16	−1.031	3.04E−04
17	Branin	(-5,10),(0,15)	2	0.401	3.6e−3	0.398	6.0e−8	0.398	1.0e−13	0.398	1.4e−15	0.3985	3.27E−04
18	Goldstein-Price	−2,2	2	3.007	1.2e−2	3.0	0	3	5.8e−14	14.34	25.05	3.000E+00	3.09E−04
19	Hartman 3	0,1	3	−3.8613	1.2e−3	−3.86	4.0e−3	−3.8628	2.9e−13	−3.8628	4.8e−16	−3.862E+00	2.77E−04
20	Hartman 6	0,1	6	−3.24	5.8e−2	−3.23	0.12	−3.31	3.3e−2	−3.28	5.8e−2	−3.322E+00	1.35E−04
21	Shekel 5	0,10	4	−5.72	2.62	−5.54	1.82	−8.49	2.76	−5.86	3.60	−10.151	1.17E−02
22	Shekel 7	0,10	4	−6.09	2.63	−6.76	3.01	−8.79	2.64	−6.58	3.74	−10.402	3.11E−04
23	Shekel 10	0,10	4	−6.42	2.67	−7.63	3.27	−9.65	2.06	−7.03	3.74	−10.535	2.02E−03

number of function evaluations was 5000 for each run. Totally, runs were repeated 10.000 times per test function for ABC. All settings were tuned as in [38] to make fair comparison.

5.2.2. Benchmark functions

In Experiments 3, functions given in Table 23 were employed. Functions in the set are low dimensional. Rastrigin and Griewank functions are the same in Experiments 1 and Experiments 2 while a Gaussian bump in $(-1,1)$ is added to Rosenbrock function used in other experiments. This modification causes a local minimum in $(1,1)$ and a global minimum in $(-1,-1)$. This modification makes the problem harder because local minimum basin is larger than global minimum basin [38].

5.2.3. Results

Table 24 presents the mean and the standard deviation of function evaluation numbers of the algorithms tested, and the mean and the standard deviations of success rates which is the percent of converged runs over 100 optimization runs. From the results, on Rosenbrock function, SOMES and NG-ES are more successful in terms of both solution and cost qualities. However, on the other multimodal functions (Griewank and Rastrigin), ABC outperforms SOM-ES, NG-ES and CMA-ES algorithms. On three functions, ABC shows better performance in terms of success rate than CMA-ES. As in Experiments 2, while ABC has slower convergence rate than CMA-ES which uses local information to converge quickly, ABC has better performance in terms of global optimization.

Table 21
Solution costs in terms of evaluation number of CES [41], FES [36], ESLAT [41], CMA-ES [41] and ABC algorithms.

No	Function	CES	FES	ESLAT	CMA-ES	ABC	
		Mean	Mean	Mean	Mean	Mean	StdFE
1	Sphere	69,724	150,000	69,724	10,721	9264	1481
2	Schwefel 2.22	60,859	200,000	60,859	12,145	12,991	673
3	Schwefel 1.2	72141	500,000	72,141	21,248	12,255	1390
4	Schwefel 2.21	69,821	500,000	69,821	20,813	100,000	0
5	Rosenbrock	66,609	1,500,000	66,609	55,821	100,000	0
6	Step	57,064	150,000	57,064	2184	4853	1044
7	Quartic	50,962	300,000	50,962	667,131	100,000	0
8	Schwefel	61,704	900,000	61,704	6621	64,632	23,897
9	Rastrigin	53,880	500,000	53,880	10,079	26,731	9311
10	Ackley	58,909	150,000	58,909	10,654	16,616	1201
11	Griewank	71,044	200,000	71,044	10,522	36,151	17,128
12	Penalized	63,030	150,000	63,030	13,981	7340	2020
13	Penalized 2	65,655	150,000	65,655	13,756	8454	1719
14	Foxholes	1305	10,000	1305	540	1046	637
15	Kowalik	2869	400,000	2869	13,434	6120	4564
16	SixHump	1306	10,000	1306	619	342	109
17	Branin	1257	10,000	1257	594	530	284
18	Goldstein-Price	1201	10,000	1201	2052	15,186	13,500
19	Hartman 3	1734	10,000	1734	996	4747	16,011
20	Hartman 6	3816	20,000	3816	2293	1583	457
21	Shekel 5	2338	10,000	2338	1246	6069	13,477
22	Shekel 7	2468	10,000	2468	1267	7173	9022
23	Shekel 10	2410	10,000	2410	1275	15,392	24,413

Table 22
Success rates (%) of ESLAT [41], CMA-ES [41] and ABC algorithms. Result in boldface indicates that it is the best success rate achieved in all of the algorithms.

No	Function	ESLAT	CMA-ES	ABC
1	Sphere	**100**	**100**	**100**
2	Schwefel 2.22	**100**	**100**	**100**
3	Schwefel 1.2	**100**	**100**	**100**
4	Schwefel 2.21	0	**100**	0
5	Rosenbrock	70	**90**	0
6	Step	98	36	**100**
7	Quartic	0	0	0
8	Schwefel	0	0	**86**
9	Rastrigin	40	0	**100**
10	Ackley	**100**	**100**	**100**
11	Griewank	90	92	**96**
12	Penalized	**100**	88	**100**
13	Penalized 2	60	86	**100**
14	Foxholes	60	0	**100**
15	Kowalik	94	88	**100**
16	SixHump	**100**	**100**	**100**
17	Branin	**100**	**100**	**100**
18	Goldstein-Price	**100**	78	**100**
19	Hartman 3	**100**	**100**	**100**
20	Hartman 6	94	48	**100**
21	Shekel 5	72	40	**98**
22	Shekel 7	72	48	**100**
23	Shekel 10	84	52	**96**
	Total:	9	9	20

Table 23
Benchmark functions used in experiments 3. D: Dimension, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable.

No	Range	D	C	Function	Formulation
1	$[-2,2]$	2	UN	Modified Rosenbrock	$f(x,y) = 74 + 100(y - x^2)^2 + (1-x)^2 - 400e^{-((x+1)^2+(y+1)^2/0.1)}$
2	$[-100,100]$	2	MN	Griewank	$f(x,y) = 1 + \frac{1}{200}(x^2+y^2) - \cos(x)\cos\left(\frac{y}{\sqrt{2}}\right)$
3	$[-5.12,5.12]$	2	MS	Rastrigin	$f(x) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$

Table 24
Statistical results of SOM-ES [38], NG-ES [38], CMA-ES [38] and ABC algorithms.

Method	Modified Rosenbrock		Griewank		Rastrigin	
	Mean Evals	Succ. (%)	Mean Evals	Succ. (%)	Mean Evals	Succ. (%)
5×5 SOM-ES	1600 ± 200	70 ± 8	130 ± 40	90 ± 7	180 ± 50	90 ± 7
7x7 SOM-ES	750 ± 90	90 ± 5	100 ± 40	90 ± 8	200 ± 50	90 ± 8
NG-ES (m=10)	1700 ± 200	90 ± 7	180 ± 50	90 ± 10	210 ± 50	90 ± 8
NG-ES (m=20)	780 ± 80	90 ± 9	150 ± 40	90 ± 8	180 ± 40	90 ± 7
CMA-ES	70 ± 40	30 ± 10	210 ± 50	70 ± 10	100 ± 40	80 ± 9
ABC	1371 ± 2678	52 ± 5	1121 ± 960	99 ± 1	1169 ± 446	100 ± 0

6. Discussion and conclusion

In this work, the performance of ABC algorithm was compared with those of GA, PSO, DE and ES optimization algorithms. Although there are several improved versions of GA, DE and PSO in the literature, their standard versions were used since the ABC algorithm presented in this work is its first version i.e. its standard version. Although the performance of ABC algorithm can be improved by integrating useful heuristics, in this work our purpose was to compare the performance of standard version of ABC with those of other well-known population-based algorithms. In Experiments 1, we used the same population number and the maximum evaluation number for all problems although it is a known fact that these control parameters affect the performance of algorithms significantly. However, in most comparative studies these parameter values are varied with respect to the dimension of the problems or to their other characteristics. The reason is that we assumed the users of an algorithm do not know much about the recommended values of these parameters for their problems to be optimized.

While GA and DE employ crossover operators to produce new or candidate solutions from the present ones, ABC algorithm does not. ABC algorithm produces the candidate solution from its parent by a simple operation based on taking the difference of randomly determined parts of the parent and a randomly chosen solution from the population. This process increases the convergence speed of search into a local minimum. In GA, DE and PSO the best solution found so far is always kept in the population and it can be used for producing new solutions in the case of DE and GA, new velocities in the case of PSO. However, in ABC, the best solution discovered so far is not always held in the population since it might be replaced with a randomly produced solution by a scout. Therefore, it might not contribute to the production of trial solutions. Both DE and ABC employ a greedy selection process between the candidate and the parent solutions. In ABC, on "employed bees" stage a trial solution is produced for each solution in the population as in the case of DE without depending on the quality of solutions. On "onlooker bees" stage, the solutions with the higher fitness value are used more often than those with less fitness values to produce trial solutions. It means that the promising regions of the search space are searched in shorter time and in detail. This selection process is similar to the natural selection or to the seeded selection employed in GA.

In GA or DE, mutation process creates a modification on a randomly selected part of a solution to provide required diversity in the population. In ABC, there are two types of mechanisms to control the diversity in the population: (a) As in DE or GA, a randomly chosen part of a parent is modified with a magnitude determined randomly to obtain a trail solution. This modification is relatively small and useful for local search and fine tuning. (b) Rather than changing a part of a solution, a whole solution in the population is removed and then a new one produced randomly is inserted into the population by a scout. This mechanism provides the ABC algorithm a global search ability and prevents the search from premature convergence problem. This feature weakens the dependency of the algorithms' performance on the population size, too. Hence, there is a good balance between the local search process carried out by artificial onlooker and employed bees and the global search process managed by artificial scouts. Therefore, the ABC algorithm produces better results on multimodal and multivariable problems than other algorithms considered in this paper.

Apart from the maximum evaluation number and population size, a standard GA has three more control parameters (crossover rate, mutation rate, generation gap), a standard DE has at least two control parameters (crossover rate, scaling factor) and a basic PSO has three control parameters (cognitive and social factors, inertia weight). Also, limit values for the velocities v_{max} have a significant effect on the performance of PSO. The ABC algorithm has only one control parameter (limit) apart from Colony Size and Maximum Cycle Number. In the present work, we described an expression for determining the value of "limit" depending on population (colony size) and dimension of problem. Therefore, now ABC has only two common control parameters: maximum cycle number (MCN) and colony size (SN). Consequently, ABC is as simple and flexible as DE and PSO; and also employs less control parameters.

ESs used in Experiments 2 and 3 employ recombination and mutation operators to produce offsprings (new individuals) while ABC uses only mutation operator. Although the basic version of ES is as simple as ABC, the improved versions used for comparison in this work are more complex than ABC. Moreover, all of them employ more control parameters than ABC.

This work compared the performance of ABC algorithm with those of GA, DE, PSO and ES algorithms on a large set of unconstrained test functions. From the results obtained in this work, it can be concluded that the performance of ABC algorithm is better than or similar to that of these algorithms although it uses less control parameters and it can be efficiently used for solving multimodal and multidimensional optimization problems.

Acknowledgement

This work is supported by Erciyes University, the Department of Research Projects under Contract FBA-06-22.

References

[1] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Springer, 2003.
[2] R.C. Eberhart, Y. Shi, J. Kennedy, Swarm Intelligence, Morgan Kaufmann, 2001.
[3] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[4] J.R. Koza, Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems, Technical Report STAN-CS-90-1314, Stanford University Computer Science Department, 1990.
[5] I. Rechenberg, in: Cybernetic Solution Path of an Experimental Problem, Library Translation, vol. 1122, Royal Aircraft Establishment, Farnborough, Hants, UK, 1965.
[6] H.P. Schwefel, Kybernetische evolution als strategie der experimentellen forschung in der stromungstechnik, Master's Thesis, Technical University of Berlin, Germany, 1965.
[7] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence Through Simulated Evolution, John Wiley & Son, New York, NY, 1966.
[8] R. Storn, K. Price, Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical report, International Computer Science Institute, Berkley, 1995.
[9] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.
[10] K. Price, R. Storn, A. Lampinen, Differential Evolution a Practical Approach to Global Optimization, Springer Natural Computing Series, 2005.
[11] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, NY, 1999.
[12] L.N. De Castro, F.J. Von Zuben, Artificial immune systems, Part I. Basic theory and applications, Technical Report Rt Dca 01/99, Feec/Unicamp, Brazil, 1999.
[13] J. Kennedy, R.C. Eberhart, in: Particle Swarm Optimization, 1995 IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.
[14] Y. Fukuyama, S. Takayama, Y. Nakanishi, H. Yoshida, A particle swarm optimization for reactive power and voltage control in electric power systems, in: Genetic and Evolutionary Computation Conference, 1999, pp. 1523–1528.
[15] V. Tereshko, Reaction–diffusion model of a honeybee colony's foraging behaviour, in: M. Schoenauer (Ed.), Parallel Problem Solving from Nature VI, Lecture Notes in Computer Science, vol. 1917, Springer–Verlag, Berlin, 2000, pp. 807–816.
[16] V. Tereshko, T. Lee, How information mapping patterns determine foraging behaviour of a honeybee colony, Open Systems and Information Dynamics 9 (2002) 181–193.
[17] V. Tereshko, A. Loengarov, Collective decision-making in honeybee foraging dynamics, Computing and Information Systems Journal 9 (3) (2005).
[18] D. Teodorović, Transport modeling by multi-agent systems: a swarm intelligence approach, Transportation Planning and Technology 26 (4) (2003).
[19] P. Lucic, D. Teodorović, Transportation modeling: an artificial life approach, in: ICTAI, 2002, pp. 216–223.
[20] D. Teodorović, M. Dell'Orco, Bee colony optimization – a cooperative learning approach to complex transportation problems, in: Poznan, 3–16 September 2005, 10th EWGT Meeting.
[21] H. Drias, S. Sadeg, S. Yahi, Cooperative bees swarm for solving the maximum weighted satisfiability problem, computational intelligence and bioinspired systems. in: 8th International Workshop on Artificial Neural Networks IWANN 2005, Vilanova, Barcelona, Spain, June 8–10 2005.
[22] K. Benatchba, L. Admane, M. Koudil, Using bees to solve a data-mining problem expressed as a max-sat one, artificial intelligence and knowledge engineering applications: a bioinspired approach. in: First International Work-Conference on the Interplay Between Natural and Artificial Computation IWINAC 2005, Palmas, Canary Islands, Spain, June 15–18 2005.
[23] H.F. Wedde, M. Farooq, Y. Zhang, Beehive: an efficient fault-tolerant routing algorithm inspired by honeybee behavior, ant colony, optimization and swarm intelligence, in: 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5–8 2004.
[24] X.S. Yang, Engineering optimizations via nature-inspired virtual bee algorithms. in: Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach, Lecture Notes in Computer Science, vol. 3562, Springer, Berlin/Heidelberg, 2005, pp. 317–323.
[25] D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi, The bees algorithm, Technical Report, Manufacturing Engineering Centre, Cardiff University, UK, 2005.
[26] D. Karaboga, An idea based on honeybee swarm for numerical optimization, Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
[27] B. Basturk, D. Karaboga, An artificial bee colony (abc) algorithm for numeric function optimization, in: IEEE Swarm Intelligence Symposium 2006, Indianapolis, Indiana, USA, May 2006.
[28] D. Karaboga, B. Basturk, A. powerful, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm, Journal of Global Optimization 39 (3) (2007) 459–471.
[29] D. Karaboga, B. Basturk, On the performance of artificial bee colony (abc) algorithm, Applied Soft Computing 8 (1) (2008) 687–697.
[30] D. Karaboga, B. Basturk, in: Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, LNCS, vol. 4529/2007, Springer-Verlag, 2007, pp. 789–798 (Chapter Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems).
[31] D. Karaboga, B. Basturk Akay, C. Ozturk, in: Modeling Decisions for Artificial Intelligence, LNCS, vol. 4617/2007, Springer-Verlag, 2007, pp. 318–329 (Chapter Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks).
[32] D. Karaboga, B. Basturk Akay, An artificial bee colony (abc) algorithm on training artificial neural networks, in: 15th IEEE Signal Processing and Communications Applications, SIU 2007, Eskisehir, Turkiye, June, pp. 1–4.
[33] N. Karaboga. A new design method based on artificial bee colony algorithm for digital iir filters, Journal of The Franklin Institute 346 (4) (2009) 328–348.
[34] Alok Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, Applied Soft Computing 9 (2) (2009) 625–631.
[35] T. Back, H.P. Schwefel, An overview of evolutionary algorithms for parameter optimization, Evolutionary Computation 1 (1) (1993) 1–23.
[36] X. Yao, Y. Liu, Fast evolution strategies, Control and Cybernetics 26 (3) (1997) 467–496.
[37] T. Kohonen, Self-Organizing Maps, Springer-Verlag, New York, 1995.
[38] M. Milano, P. Koumoutsakos, J. Schmidhuber, Self-organizing nets for optimization, IEEE Transactions on Neural Networks 15 (3) (2004) 758–765.
[39] T. Martinetz, S. Schulten, A neural-gas network learns topologies, in: K. Kohonen et al. (Eds.), Artificial Neural Networks, Elsevier, North-Holland, The Netherlands, 1991, pp. 397–402.
[40] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, in: IEEE Int. Conf. Evolution. Comput. (ICEC) Proc., 1996, pp. 312–317.
[41] A. Hedar, M. Fukushima, Evolution strategies learned with automatic termination criteria, in: Proceedings of SCIS-ISIS 2006, Tokyo, Japan, 2006.
[42] D.T. Pham, D. Karaboga, Optimum design of fuzzy logic controllers using genetic algorithms, Journal of Systems Engineering 1 (1991) 114–118.
[43] D. Corne, M. Dorigo, F. Glover, New Ideas in Optimization, McGraw-Hill, 1999.
[44] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution particle swarm optimization and evolutionary algorithms on numerical benchmark problems, in: IEEE Congress on Evolutionary Computation (CEC'2004), vol. 3, Piscataway, New Jersey, June 2004, pp. 1980–1987.
[45] D.O. Boyer, C.H. Martfnez, N.G. Pedrajas, Crossover operator for evolutionary algorithms based on population features, Journal of Artificial Intelligence Research 24 (2005) 1–48.

[46] A.D. Junior, R.S. Silva, K.C. Mundim, L.E. Dardenne, Performance and parameterization of the algorithm simplified generalized simulated annealing, Genetics and Molecular Biology 27 (4) (2004) 616–622.
[47] J.G. Digalakis, K.G. Margaritis, An experimental study of benchmarking functions for genetic algorithms, International Journal of Computer Mathematics 79 (4) (2002) 403–416.
[48] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: Swarm Intelligence Symposium, 2007, SIS 2007. IEEE, Honolulu, HI, 2007, pp. 120–127.
[49] Z. Michalewicz, C. Janikow, Genetic Algorithms + Data Structures = Evolution Programs, third ed., Springer-Verlag, Newyork, 1996.