A novel hybrid Artificial Bee Colony algorithm and Differential Evolution for unconstrained optimization problems

Amir Alizadegan. Islamic Azad University, Arak, Iran. amiralizadegan@yahoo.com
Mohammad Reza Meybodi. Islamic Azad University, Arak, Iran. mmeybodi@aut.ac.ir
Babak Asadi. Islamic Azad University, Arak, Iran. babakmz2002@yahoo.com

Abstract

ABC (Artificial Bee Colony) algorithm is one of the most popular approaches that is used in optimization problems. ABC overcomes other well-known heuristic methods, such as GA, PSO and DE (Differential Evolution). In this paper, we propose a hybrid ABC-DE algorithm that combines properties of ABC and DE approaches. The results show that our proposed algorithm is better than both ABC and DE and other methods.

Key words: Artificial bee colony, Differential evolution, Numerical function optimization

1. Introduction

Karaboga has described Artificial Bee Colony (ABC) algorithm based on the foraging behavior of honey bees for numerical optimization problems [1], and Karaboga and Basturk have compared the performance of the ABC algorithm with those of other well-known modern heuristic algorithms such as Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) on unconstrained problems [2,3]. Because ABC is a robust, easy and flexible algorithm, it has used in different problems. For example, it is used in Neural Networks [4], spanning tree [5], digital filters [6], in clustering [7], or in constrained optimization problem [8]. In [9], a hybrid algorithm of PSO and DE combination is described. In this work we introduce three algorithms of ABC and DE combination. In Section 2, the ABC algorithm and in section 3, DE is described. In section 4, the ABC-DE algorithm will be suggested and in section 5 experiments and results are presented and discussed.

2. Artificial Bee Colony (ABC) algorithm

Artificial Bee Colony (ABC) algorithm was proposed by Karaboga for optimizing numerical problems in 2005 [1]. The algorithm simulates the intelligent foraging behavior of honey bee swarms. It is a very simple, robust and population based stochastic optimization algorithm. Karaboga and Basturk have compared the performance of the ABC algorithm with those of other well-known modern heuristic algorithms such as Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) on unconstrained problems [2].

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

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 $P(G = 0)$ of SN solutions (food source positions), where SN denotes the size of population. Each solution x_i $(i = 1, 2, \dots, SN)$ is a D-dimensional vector. Here, D is the number of optimization parameters. After initialization, the population of the positions (solutions) is subjected to repeated cycles, $C = 1,2, \dots, MCN$, of the search processes of the employed bees, the onlooker bees and 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). Provided that 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 on the dance area. 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. Providing that its nectar is higher than that of the previous one, the bee memorizes the new position and forgets the old one.

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 (1):

$$\mathrm{p_i} = \frac{\mathrm{fit_i}}{\sum_{\mathrm{n=1}}^{\mathrm{SN}} \mathrm{fit_n}} \qquad (1)$$

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 (BN).

In order to produce a candidate food position from the old one in memory, the ABC uses the following expression (2):

$$v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj}) \qquad (2)$$

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

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, providing that a position can not 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 $j \epsilon \{1,2, \dots, D\}$, then the scout discovers a new food source to be replaced with x_i. This operation can be defined as in (3):

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

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. There are three control parameters in the ABC: The number of food sources which is equal to the number of employed or onlooker bees (SN), the value of $limit$, the maximum cycle number (MCN).

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.

3. Differential Evolution

The differential evolution (exploration) [DE] algorithm introduced by Storn and Price (1995) [10] is a novel parallel direct search method, which utilizes NP parameter vectors as a population for each generation G. DE can be categorized into a class of floating-point encoded, evolutionary optimization algorithms. Currently, there are several variants of DE. The particular variant used throughout this investigation is the DE/rand/1/bin scheme. This scheme will be discussed here and more detailed descriptions are provided (Storn and Price). Since the DE algorithm was originally designed to work with continuous variables, the optimization of continuous problems is discussed first. Handling discrete variables is explained later.

Generally, the function to be optimized, $\beth$, is of the form $\beth(X): R^{\mathrm{D}} \rightarrow \mathrm{R}$. The optimization target is to minimize the value of this objective function $\beth(X)$,

$$\min(\beth(X)),$$

by optimizing the values of its parameters $X = \{x_1, x_2, \dots, x_D\}$, $X \epsilon R^D$, where X denotes a vector composed of D objective function parameters. Usually, the parameters of the objective function are also subject to lower and upper boundary constraints, $x^{(L)}$ and $x^{(U)}$, respectively,

$$x_j^{(L)} \le x_j \le x_j^{(U)} \qquad \forall j \epsilon [1, D]$$

3.1. Initialization

As with all evolutionary optimization algorithms, DE works with a population of solutions, not with a single solution for the optimization problem. Population P of generation G contains NP solution vectors called individuals of the population and each vector represents potential solution for the optimization problem

$$P^{(G)} = X_i^{(G)} = X_{j,i}^{(G)}, \qquad i = 1, \dots, NP; \; j = 1, \dots, D; \; G = 1, \dots, G_{max} \tag{4}$$

In order to establish a starting point for optimum seeking, the population must be initialized. Often there is no more knowledge available about the location of a global optimum than the boundaries of the problem variables. In this case, a natural way to initialize the population $P^{(0)}$ (initial population) is to seed it with random values within the given boundary constraints:

$$P^{(0)} = x_{j,i}^{(0)} = x_j^{(L)} + rand_j[0,1] \times \left(x_j^{(U)} - x_j^{(L)}\right) \qquad \forall i \epsilon [1, NP]; \qquad \forall j \epsilon [1, D] \tag{5}$$

where $rand_j[0,1]$ represents a uniformly distributed random value that ranges from zero to one.

3.2. Mutation

The self-referential population recombination scheme of DE is different from the other evolutionary algorithms. From the first generation onward, the population of the subsequent generation $P^{(G+1)}$ is obtained on the basis of the current population $P^{(G)}$. First a temporary or trial population of candidate vectors for the subsequent generation, $P'^{(G+1)} = V^{(G+1)} = v_{j,i}^{(G+1)}$, is generated as follows:

$$v_{j,i}^{(G+1)} = \begin{cases} x_{j,r3}^{(G)} + F \times \left(x_{j,r1}^{(G)} - x_{j,r2}^{(G)}\right), & \text{if } rand_j[0,1] < CR \text{ or } j = k \\ x_{j,i}^{(G)}, & \text{otherwise} \end{cases} \tag{6}$$

where $i \epsilon [1, NP]$; $j \epsilon [1, D]$, $r1, r2, r3 \epsilon [1, NP]$, randomly selected, except: $r1 \ne r2 \ne r3 \ne i$, $k = (int(rand_i[0,1] \times D) + 1)$, and $CR \epsilon [0,1]$, $F \epsilon [0,1]$.

Three randomly chosen indices, for $r1$, $r2$, and $r3$ refer to three randomly chosen vectors of population. They are mutually different from each other and also different from the running index i. New random values for $r1$, $r2$, and $r3$ are assigned for each value of index i (for each vector). A new value for the random number rand[0, 1] is assigned for each value of index j (for each vector parameter).

3.3. Crossover

The index k refers to a randomly chosen vector parameter and it is used to ensure that at least one vector parameter of each individual trial vector $V^{(G+1)}$ differs from its counterpart in the previous generation $X^{(G)}$. A new random integer value is assigned to k for each value of the index i (prior to construction of each trial vector). F and CR are DE control parameters. Both values remain constant during the search process. Both values as well as the third control parameter, NP (population size), remain constant during the search process. F is a real-valued factor in range [0.0, 1.0] that controls the amplification of differential variations. CR is a real-valued crossover factor in the range [0.0, 1.0] that controls the probability that a trial vector will be selected form the randomly chosen, mutated vector, $V_{j,i}^{(G+1)}$ instead of from the current vector, $X_{j,i}^{(G)}$. Generally, both F and CR affect the convergence rate and robustness of the search process. Their optimal values are dependent both on objective function characteristics and on the population size, NP. Usually, suitable values for F, CR and NP can be found by experimentation after a few tests using different values. Practical advice on how to select control parameters NP, F and CR can be found in Storn and Price [10].

3.4. Selection

The selection scheme of DE also differs from the other evolutionary algorithms. On the basis of the current population $P^{(G)}$ and the temporary population $P^{'(G+1)}$, the population of the next generation $P^{(G+1)}$ is created as follows:

$$X_{\mathrm{i}}^{(\mathrm{G}+1)} = \begin{cases} V_{\mathrm{i}}^{(\mathrm{G}+1)}, & \text{if } \beth\left(V_{\mathrm{i}}^{(\mathrm{G}+1)}\right) \leq \beth\left(X_{\mathrm{i}}^{(\mathrm{G})}\right), \\ X_{\mathrm{i}}^{(\mathrm{G})}, & \text{otherwise.} \end{cases} \tag{7}$$

Thus, each individual of the temporary or trial population is compared with its counterpart in the current population. The one with the lower value of cost-function $\beth(X)$ to be minimized will propagate the population of the next generation. As a result, all the individuals of the next generation are as good or better than their counterparts in the current generation. The interesting point concerning the DE selection scheme is that a trial vector is only compared to one individual vector, not to all the individual vectors in the current population.

4. Proposed ABC-DE Algorithm

In ABC-DE algorithm, we have three types of bees: employed bees, onlooker bees and scouts. The half of the colony contains employed bees and the remaining half contains onlooker bees. For each food source (solution), there exists one employed bee. In other words, the number of employed bees is equal to number of food sources. The role of employed bees is producing new solutions from current solutions. They do this by (6).

The onlooker bees, based on probability, choose solutions with better fitness and produce new solutions by (6). The probability of choosing a solution is calculated by (1). The new produced solutions is replaced the current solution, if the fitness of new solution is equal or better than the current one. This condition is done by (7). After some iterations, determined by control parameter $limit$, If a better solution can not be produced by employed or onlooker bees, the employed bee converts to scout and will initialize the solution randomly.

Like in DE, control parameters F and CR are important elements of ABC-DE algorithm, too. These control parameters still have major role in producing new solutions.

As can be observed, the structure of ABC-DE algorithm is as same as the ABC; But the way of producing new solution, is similar to DE. In fact, the way of producing new solutions in DE are embedded in ABC algorithm.

The detailed pseudo code of this algorithm is shown below:

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

5. Experiments and results

5.1. Benchmark functions

In this section, the result of proposed algorithm is evaluated. We have tested the proposed algorithm for optimizing some well-known numerical functions. Details of these functions are shown in Table 1.

Table 1: benchmark functions

Function	Formulation	Type	Range	Optimum value
Sphere	$f(x) = \sum_{i=1}^{D} x_i^2$	US	[-100,100]	0
Rosenbrock	$f(x) = \sum_{i=1}^{D} 100(x_{i+1} - x_i^2)^2 + (1 - x_i^2)^2$	UN	[-15,15]	0
Rastrigin	$f(x) = \sum_{i=1}^{D} x_i^2 - 10\cos(2\pi x_i) + 10$	MS	[-15,15]	0
Schwefel	$f(x) = 418.9829 * D + \sum_{i=1}^{D} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	MS	[-500,500]	0
Griewank	$f(x) = \frac{1}{4000}\left(\sum_{i=1}^{D}(x_i^2)\right) - \left(\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$	MN	[-600,600]	0
Schaffer	$f(x) = 0.5 + \frac{sin^2\left(\sqrt{x_1^2+x_2^2}\right)-0.5}{\left(1+0.001(x_1^2+x_2^2)\right)^2}$	MN	[-100,100]	0

The type of functions that are shown in column *Type*, consists two important characteristics; a function may be unimodal (U), or multimodal (M). In the first, there is just one optima; but in the second there is a global optima with a lot of local optimums. Surely, optimizing in multimodal environments is more difficult with respect to unimodal environments. The second characteristic is that the function is separable (S) or non-separable (N).

The initializing range for functions differs and relates to their optimums` distribution. Also the minimum values of all of the functions are 0.

5.2. Settings of the algorithm and results

In ABC-DE, we have three sorts of bees. 50% of bees` populations are employed bees and the remaining 50% are onlooker bees. The colony size is set to 100. Two important control parameters of ABC-DE algorithm are F and CR. In experiments, the value of F is chosen randomly from the range [-1,-0.4] ∪ [0.4,1]. The experiments were run for different values of CR; i.e. $CR \in \{0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 0.999, 1\}$.

For the first 5 benchmark functions showed in Table 1, the experiments were done for dimensions 2, 5, 10, 20 and 30 and the number of cycles depends on the number of dimensions; 200, 300, 500, 750 and 1000 respectively. For the function *Schaffer*, there exists just 2 dimensions and we used 200 cycle in experiments. The value of control parameter $limit$ is 0.3 of cycle. Each of the experiments was repeated 30 times with different random seeds.

The mean and the standard deviations (SD) of the function values obtained by the ABC-DE and ABC are given in Table 2. Because ABC has better performance in respect to other population-based and evolutionary algorithms, such as GA, DE, PSO and PS-EA, [1-3], we just have compared the results of ABC and ABC-DE.

In Table 2, the values in column *CR* indicate a value of control parameter CR, so that via this value the best result has been achieved for ABC-DE algorithm. Also, the values in column *Range*, indicate a range of values that for these values of control parameter CR, the results of ABC-DE is better than ABC. For example, in function Sphere and for dimensionality 20, the best value of ABC-DE algorithm is obtained for $CR = 0.3$; but for all of values in the range [0.001, 0.5] for control parameter CR, the results of ABC-DE is better than ABC algorithm.

As can be seen from Table 2, it is obvious that almost for all of benchmark functions and for any dimension, the results of ABC-DE algorithm is equal or better than the results of ABC algorithm. The higher dimensionality of the function *Rosenbrock* is an exception; i.e. the results of ABC algorithm for $D = 20$ and 30, is better than ABC-DE algorithm.

The most important step of new proposed algorithm is turning the control parameter CR, so that for any objective function, either unimodal or multimodal, either separable or non-separable, ABC-DE algorithm can produce the best results. From column *Range*, it seems that for small values of CR, around $CR = 0.1$, the ABC-DE algorithm has better performance in various environment conditions.

Table 2: Mean and SD of ABC and ABC-DE algorithms

Functions	D	ABC		ABC-DE			
		Mean	SD	Mean	SD	CR	Range
Sphere	2	2.0085E-18	1.9551E-18	3.5596E-19	2.5650E-19	0.8	[0.001,1]
	5	3.2082E-17	1.2373E-17	7.6249E-18	3.3637E-18	0.9	[0.001,1]
	10	9.2742E-17	2.8934E-17	2.7316E-17	7.1854E-18	0.99	[0.001,1]
	20	6.3911E-15	6.2180E-15	1.4665E-16	6.2004E-17	0.3	[0.001,0.5]
	30	2.1526E-12	1.9876E-12	3.2331E-16	5.3861E-17	0.2	[0.001,0.4]
Rosenbrock	2	0.0030	0.0035	2.0381E-19	2.3726E-19	0.9	[0.1,1]
	5	0.0358	0.0376	7.2118E-18	3.6604E-18	0.99	[0.4,0.999]
	10	0.0815	0.0978	2.2229E-7	3.4902E-7	0.9	[0.8,0.9]
	20	0.2065	0.2319	2.1273	1.0013	0.001	-
	30	0.3908	0.3825	5.1592	2.7294	0.001	-
Rastrigin	2	0.0	0.0	0.0	0.0	[0.001,1]	[0.001,1]
	5	0.0	0.0	0.0	0.0	[0.001,0.6]	[0.001,0.6]
	10	4.8553E-15	5.9728E-15	0.0	0.0	[0.001,0.3]	[0.001,0.3]
	20	3.7232E-5	2.0389E-4	0.0	0.0	[0.001,0.01]	[0.001,0.1]
	30	0.0930	0.2848	0.0	0.0	[0.001,0.01]	[0.001,0.1]
Griewank	2	1.7081E-9	7.8492E-9	0.0	0.0	[0.1,1]	[0.1,1]
	5	0.0023	0.0035	1.6689E-13	8.0476E-13	0.2	[0.001,0.3]
	10	0.0036	0.0058	7.0866E-11	3.5249E-10	0.1	[0.001,0.2]
	20	1.8107E-9	4.8144E-9	1.1102E-17	3.3876E-17	0.1	[0.1,0.4]
	30	4.2446E-6	2.1987E-5	9.2518E-17	7.1911E-17	0.1	[0.001,0.5]
Schwefel	2	2.5455E-5	0.0	2.5455E-5	0.0	[0.001,1]	[0.001,1]
	5	6.3637E-5	3.0873E-13	6.3637E-5	0.0	[0.001,0.8]	[0.001,0.8]
	10	1.2727E-4	1.5965E-10	1.2727E-4	0.0	[0.001,0.6]	[0.001,0.6]
	20	19.7508	44.8891	2.5455E-4	3.3210E-13	[0.001,0.2]	[0.001,0.2]
	30	184.7264	122.4497	3.8182E-4	6.7555E-13	[0.001,0.1]	[0.001,0.1]
Schaffer	2	9.6955E-6	4.5729E-5	0.0	0.0	[0.3,0.6]	[0.2,0.6]

6. Conclusion

In this work, we proposed the ABC-DE algorithm that combining the ABC and DE approaches and we evaluated the results of proposed algorithm on well-known test beds. Approximately, for all of test beds, the new algorithm work better than original ABC algorithm. As a suggestion for future studies, the effect of control parameter *F* on new proposed algorithm can be evaluated.

References

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

[2] D. Karaboga, B. Basturk, "On the performance of Artificial Bee Colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, pp. 687–697, January 2008.

[3] D. Karaboga, B. Akay, "A comparative study of Artificial Bee Colony algorithm," *J Appl Mathe Comput*, vol. 214, pp. 108-132, August *2009*.

[4] D. Karaboga, B. Basturk, C. Ozturk, "Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks," *Springer*, pp. 318–329, 2007.

[5] A. Singh, "An Artificial Bee Colony algorithm for the leaf-constrained minimum spanning tree problem,*"* *Applied Soft Computing*, vol. 9, pp. 625–631, March 2009.

[6] N. Karaboga, "A new design method based on Artificial Bee Colony algorithm for digital IIR filters," *Journal of The Franklin Institute*, vol. 346, pp. 328–348, May 2009.

[7] C. Zhang, D. Ouyang, J. Ning, "An Artificial Bee Colony approach for clustering," *Expert Systems with Applications*, vol. 37, pp 4761-4767, July 2010.

[8] D. Karaboga, B. Basturk, "Artificial Bee Colony (ABC) Optimization algorithm for Solving Constrained Optimization Problems", 12th Foundations of fuzzy logic and soft computing, Cancun, Mexico, pp.789-798, 2007.

[9] C. Zhang, J. Ning, S. Lu, D. Ouyang, T. Ding, "A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization," *Operations Research Letters* vol. 37, pp. 117_122, 2009.

[10] R. Storn, K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization* vol. 11, pp.341–359, 1997.