Journal of
Computational and Theoretical Nanoscience
Vol. 9, 1–9, 2012

Hybrid Differential Artificial Bee Colony Algorithm

Ajith Abraham[1, *], Ravi Kumar Jatoth[2], and A. Rajasekhar[3]

[1] *Machine Intelligent Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, USA*
[2] *Department of Electronics and Communication Engineering, National Institute of Technology Warangal, India*
[3] *Department of Electrical and Electronics Engineering, National Institute of Technology Warangal, India*

Artificial Bee Colony Algorithm (ABCA) is a new population-based meta-heuristic approach inspired by the foraging behaviour of bees. This article describes an application of a novel Hybrid Differential Artificial Bee Colony Algorithm (HDABCA), which combines Differential Evolution strategy with Artificial Bee Colony algorithm. We illustrate the proposed method using several test functions and also compared with classical differential evolution algorithm and artificial bee colony algorithm. Simulation results illustrate that the proposed method is very efficient.

Keywords: Artificial Bee Colony, Differential Evolution, Optimization Models.

1. INTRODUCTION

During the last few decades, many general-purpose optimization algorithms have been proposed for obtaining optimal solutions to intricate and complex optimization problems sprouting in the arena of Engineering. Some of these optimization techniques are evolution Strategies,[23] Evolutionary Programming (EP),[2] Genetic Algorithms (GA),[3] Particle Swarm Optimization (PSO),[4] Differential Evolution (DE),[12] Bacterial Foraging Optimization Algorithm (BFOA)[26] and Artificial Bee colony algorithm[5–7] etc. These algorithms are also Nature Inspired or Bio Inspired Algorithms because they are based on the intelligent simple rules of nature. These algorithms have been successfully applied to a wide range of engineering design problems.[1, 8–11, 13–20, 22–25]

Differential Evolution (DE) is simple yet powerful evolutionary algorithm (EA) for global optimization introduced by Price and Stom.[12] It has been successfully been applied to many fields of science and engineering, such as mechanical design, signal processing, chemical engineering, machine intelligence and pattern recognition. According to recent studies DE outperforms many other optimization methods in terms of convergence speed and robustness over numerical benchmark functions and real world problems.[16] An artificial bee colony algorithm (ABC) algorithm is one among the swarm intelligent based algorithms which employs the intelligence of the foraging behaviour of swarm of bees in solving numerical optimization problems. ABC algorithm and its variants have been applied successfully to unconstrained numerical optimization problems.[5–7] The experiments show that the extended version of ABC algorithm has better performance than DE and PSO.

Most of the population-based optimization algorithms, suffer from long computational times because of their evolutionary/stochastic nature. This crucial drawback sometimes limits their applications to offline problems with little or no real-time constraints. So in order to obtain the most of advantages of the nature inspired heuristic methods and to eliminate their disadvantages like premature convergence and computational time, hybridization is performed. In this article we proposed a new Hybrid Differential Artificial Bee Colony (HDABC), which combines ABCA with Differential Evolution algorithm. The proposed algorithm is tested on several benchmark functions and also compared with the state-of the-art optimization techniques like Differential Evolution and Artificial Bee Colony Optimization algorithms.

The rest of paper is organized as follows. Section 2 deals with ABC algorithm followed by Section 3, which presents the essential concepts of DE and the hybrid method involving DE and ABC. Section 4 illustrates the performance of the proposed approach on various benchmark functions. Finally, a few conclusions are provided towards the end.

2. ARTIFICIAL BEE COLONY ALGORITHM

Honey bees live in extremely populous colony and maintain a unique elaborate social organization. These bees are creatures of unexpected complexity-models of domesticity, able to generate food, explore the food sources (flower

*Author to whom correspondence should be addressed.

RESEARCH ARTICLE

patches), communicate through intricate dances. Their colonies represent the ultimate socialist state, with complete selflessness and redistribution of "income." A Bee Colony can be considered as swarm whose individual social agents are bees. The exchange of information among bees leads to the formation of tuned collective knowledge. Virtually the bee colony consists of a single "queen bee," a few hundred drones (males), and tens of thousands of workers (non-reproductive females). The queen is an egg lying machine and a chemical factory, her job is to lay eggs and start new colonies. She manipulates the behavior of workers through various pheromones (chemical messengers) by realizing them time to time. The sole function of drones is to mate with queen bee and chased away during times of food scarcity. The workers take the responsibility of comb construction, maintenance, brood care, cleaning the colony and foraging for nectar and pollen from the environment to feed drones and queen. The success of honey bees to maintain such large colonies lies in their ability to efficiently harvest large but ephemeral sources of pollen and nectar form flowers in their neighborhood. Female (worker) bees goes probably at their early stages of age, begin foraging for food making trip after trip.

The minimal model of forage selection that leads to the emergence of collective intelligence of honey bee swarms consists of three essential components: food sources, employed foragers, unemployed foragers, and two leading models of the behavior, recruitment to a nectar source and abandonment of a source. The value of food source depends on several factors such as its proximity to the nest, its richness or concentration of energy and ease of extracting this energy. In other words, the "profitability" of food source can be represented in single quantity. Employed foragers are associated with a particular food source, which are being exploited. They carry with them information about this particular source such as its distance (and also direction) form nest and share this information with certain probability. Unemployed foragers are continually looking for a food source to exploit. There are two types of unemployed foragers: scouts searching the environment and surrounding the nest for new food sources and onlookers waiting in the nest and searching a food source through the information shared by employed foragers.

The exchange of information among bees is most important occurrence in the formation of collective knowledge and this depends on communication. Communication among bees related to the quality of food occurs in dancing area of hive. This dance is called round dance or waggle dance. The direction of the waggle run contains about the direction of food. Employed foragers share their information with a probability proportional to the food source and sharing of this information through waggle dancing is longer in duration. Hence the recruitment is directly proportional to the profitability of the food source.

ABCA is a swarm intelligent optimization algorithm inspired by honey bee foraging. It is a new optimizer proposed by Karaboga for multivariable and multi-modal continuous function optimization.[5–7] The ABC algorithm classifies the foraging artificial bees into three groups namely employed bees, onlooker bees and scouts. The first half colony consists of the employed bees and second half consists of onlooker bees. For every food source, there is only one employed bee and the employed bee of abandoned food source becomes scout. In ABC algorithm, each solution to the problem is considered as *food source and* represented by a D-dimensional real-valued vector, whereas the fitness of the solution corresponds to the *nectar amount* of associated food resource. ABC algorithm is also similar to other swarm intelligent based approaches i.e., it is also an iterative process.

It starts with population of randomly distributed solutions or food sources. The following steps are repeated until a termination criterion is met.

(1) Calculate the nectar amounts by sending the employed bees on to the food sources.

(2) After sharing the information from employed bees select the food sources by the onlookers and determine the nectar amount of food sources.

(3) Determine the scout bees and send them to find out new food sources.

The main components of artificial bee colony algorithm are explained as follows:

2.1. Parameters Initialization

The basic parameters of ABC algorithm are number of food sources (FS) which is equal to number of employed bees (n_e) or onlooker bees (n_o), the number of trails after which food source is assumed to be abandoned (*limit*) and at last termination criterion. In the basic ABC algorithm, the number of employed bees is set equal to number of food sources in the population i.e., for every food source there is one employed bee.

2.2. Population Initialization

The initial population of solutions is set to FS number of randomly generated D-dimensional real-valued vectors (i.e., food sources). Let the i_{th} food source in the population is represented by $X_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}$, and then each food source is generated by:

$$x_{ij} = lb_j + r^*(ub_j - lb_j) \quad \text{for } j = 1, 2, 3 \ldots, D \text{ and} i = 1, 2, 3, \ldots, FS$$

Where r is a random umber in range of [0 1]; lb_j and u_{bj} are upper and lower bounds for the dimension j, respectively. These food sources are assigned randomly to n_e number of employed bees and their fitness values are evaluated.

2.3. Initialization of Bee Phase (Employed Bee)

In this stage, a new food source x_{new} is generated by each employed bee x_i in the neighbourhood of its present position by using:

$$x_{\text{new}} = x_{ij} + r^*(x_{ij} - x_{kj}) \quad (1)$$

Where $k \in \{1, 2, 3, \ldots, FS\}$ such that $k \neq i$ and $j \in \{1, 2, 3, \ldots, D\}$ are randomly chosen indexes. r is an uniformly distributed random number in the range $[-1\ 1]$.

Once the new solution is obtained, a greedy selection mechanism is employed between the old and new candidate solutions i.e., obtained x_{new} is evaluated and compared with x_i. I the fitness of x_{new} is equal to or better than that of x_i, x_{new} will replace and becomes new member of population; otherwise x_i is retained.

2.4. Onlooker Bee Phase

At this stage, onlooker bees evaluates the nectar information taken from all of the employed bees and then selects the food source x_i depending on its probability value p_i calculated by:

$$p_i = \frac{\text{fit}_i}{\sum_{k=1}^{FS} \text{fit}_k}$$

Where fit_i is the nectar amount (i.e., fitness value) of the i_{th} food source x_i. From the above equation it is clear that the higher fit_i is, greater the probability of selecting i_{th} food source. Once a food source i_{th} is selected by the onlooker bee, she performs modification on x_i using (1) same in the case of employed bees. If the modified food source has better or equal nectar amount than that of x_i, the modified food source will replace x_i and become new member of the population.

Let $f(X_i)$ be the function to be minimized (i.e., here ITAE is the objective function in this problem) then the fitness is computed by:

$$\text{fit}_i = \frac{1}{1 + f(X_i)}$$

2.5. Scout Bee Phase

A food source x_i is assumed to abandoned, if it cannot be further improved through predetermined number of trails (*limit*), and corresponding employed bee becomes scout. The scout bee produces its food source randomly by making use of this equation

$$x_{ij} = lb_j + r^*(ub_j - lb_j) \quad \text{for } j = 1, 2, 3 \ldots, D$$

Where r is a random number in the uniform range of [0 1].

2.6. Pseudo Code of ABC Algorithm

(1) Initialize the population of solutions $x_{i,j}$
(2) Population is evaluated.
(3) FOR cycle = 1; REPEAT
(4) New solutions (food sources positions) $y_{i,j}$ in the neighbourhood of $x_{i,j}$ are produced for the employed bees (n_e) using $y_{i,j} = x_{i,j} + rand(i,j) * (x_{i,j} - x_{k,j})$($k$ is the solution in the ith neighbourhood, $rand(i,j)$ being a random number in $-1 \leq rand \leq 1$) and evaluate them
(5) Store the best values between $x_{i,j}$ and $v_{i,j}$ after greedy selection process
(6) Probability values P_i for different solutions of x_i are calculated by means of their fitness values using equation

$$p_i = \frac{\text{fit}_i}{\sum_{k=1}^{FS} \text{fit}_k}$$

Here fit represents the fitness values of solutions and these are calculated using

$$\text{fit}_i = \frac{1}{1 + f(X_i)}$$

Then after P_i values are normalized into [0, 1]
(7) Based on the probabilities P_i, new solutions v_i for the onlookers are produced from the x_i
(8) REPEAT **Step-5**
(9) Next, the abandoned solution (position or source) is determined if exits, and it is replaced with a newly produced random solution x_i for the scout as explained in scout bee phase i.e., using

$$x_{ij} = lb_j + r^*(ub_j - lb_j)$$

(10) Memorize the best food source solution obtained so far
(11) cycle = cycle + 1
(12) UNTIL cycle = Maximum;
(13) STOP

3. DIFFERENTIAL EVOLUTION

Differential Evolution an evolutionary algorithm, proposed by Price and Storn,[12] is simple yet powerful heuristic method for solving non-linear, non-differentiable and multi model optimization problems. Recently, DE algorithm has gained a quite attention in the area of research related to machine intelligence and cybernetics communities. Till now Differential Evolution had found its application in various fields of science and technology of which some of them are pattern recognition and machine intelligence,[13] Chemical engineering,[14] design of fractional order controllers, clustering etc.[29–31] It had outperformed Genetic algorithms (GA) and particle swarm optimization (PSO) techniques over several benchmarks.[15] Many of the most

RESEARCH ARTICLE

interesting advancing and recent developments in DE algorithm design and applications can be found in.[16]

This technique combines the classical evolutionary operators such as mutation, crossover and selection with simple arithmetic operator, which is based on differential vector between two randomly chosen parameter vectors to evolve the population. The basic idea behind DE is a scheme for generating trail parameter vectors. Mutation and crossover are used to generate new vectors (trial vectors), and selection then after determines which of the vectors have to be survive for the next generation.

3.1. The Classical Differential Algorithm-An Outline

Differential evolution belongs to the class of evolutionary algorithms and it has no exception in method of initialization process. As that of remaining evolutionary algorithm, DE starts with *NP* *D*-dimensional parameter vectors representing the candidate or individual solutions. The subsequent generations in DE is denoted by $G = 0, 1, \ldots, G_{\max}$. Since the parameter vectors are likely to be change over several generations, the following notation is used for representing the i_{th} vector of the population at the current generation as

$$\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \ldots, x_{D,i,G}]$$

For each parameter of the problem, there may be certain range within which the value of the parameter should lie for more accurate search results at less computational cost. The initial population (at $G = 0$) should cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed maximum and minimum bounds: $\vec{X}_{\max} = \{x_{1,\max}, x_{2,\max}, \ldots, x_{D,\max}\}$ and $\vec{X}_{\min} = \{x_{1,\min}, x_{2,\min}, \ldots, x_{D,\min}\}$. Hence the j-th component of the i-th vector as.

$$x_{j,i,0} = x_{j,\min} + rand_{i,j}(0,1) \cdot (x_{j,\max} - x_{j,\min})$$

where $rand_{i,j}(0,1)$ is a uniformly distributed random number lying between 0 and 1. DE's strategy can be described as follows.

3.1.1. Mutation

Mutation is a process in which DE generates a *donor* vector $\vec{V}_{i,G}$ corresponding to each population individual (member) or *target* vector $\vec{X}_{i,G}$ in the current generation, after initialization of population. It is the method of creating this donor vector, which differentiates between various DE schemes. The different strategies are distinguished by the following notation: DE/x/y/z, where DE stands for differential evolution, x represents a string denoting the vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z denotes the recombination strategy which is used to create the trail vector.

For each target vector $\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \ldots, x_{D,i,G}]$. a mutant vector $V_{i,G}$ is generated according to:

$$\text{“DE/rand/1:”}\ \vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}) \qquad (2)$$

The indices r_1^i, r_2^i, r_3^i are mutually exclusive integers randomly chosen from the range [1, *NP*], and all different from the base index i. These indices are randomly generated once for each donor vector. The scaling factor Falso called mutation factor between [0, 2] controls the amplification of the differential variation $(\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G})$

3.1.2. Crossover

To increase the potential diversity of the population, a crossover operation comes into play after generating the donor vector through mutation. The donor vector exchanges its components with the target vector $\vec{X}_{i,G}$ under this operation to form the *trail* vector $\vec{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, u_{3,i,G}, \ldots, u_{D,i,G}]$. This discrete recombination is adopted by DE using the following scheme:

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & \text{if } (rand_{i,j}(0,1) \le Cr \text{ or } j = j_{rand} \\ x_{j,i,G}, & \text{otherwise} \end{cases} \qquad (3)$$
$$j = 1, 2, \ldots, D$$

where $rand_{i,j}(0,1) \in [0,1]$ is a uniformly distributed random number, which is called a new for each jth component of the ith parameter vector. $j_{rand} \in [1, 2, \ldots, D]$ is a randomly chosen index, which ensures that $\vec{U}_{i,G}$ gets at least one component form $\vec{V}_{i,G}$. Here *CR* is a crossover constant which controls the recombination and lies between [0, 1].

3.1.3. Selection

To keep the population size constant over subsequent generations, the next step of the algorithm calls for *selection* to the next generation i.e., at $G = G + 1$. The selection operation is described as:

$$\vec{X}_{i,G+1} = \vec{U}_{i,G}, \text{ if } f(\vec{U}_{i,G}) \le f(\vec{X}_{i,G})$$
$$= \vec{X}_{i,G}, \text{ if } f(\vec{U}_{i,G}) \le f(\vec{X}_{i,G})$$

where $f(\vec{X})$ is the function to be minimized. So if the new trail vector yields an equal or lower value of the objective function, it replaces the corresponding target vector in the next generation; otherwise the target is retained in the population. Hence the population either gets better (with respect to the minimization of the objective function) or remains the same in fitness status, but never deteriorates.

The Pseudo code of DE algorithm is described as follows:

Step 1. Set the generation number $G = 0$ and randomly initialize a population of *NP* individuals $P_G = \{\vec{X}_{1,G}, \ldots,$

$\vec{X}_{NP,G}\}$ with $\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \ldots, x_{D,i,G}]$ and each individual is uniformly distributed in the range $[\vec{X}_{\min}, \vec{X}_{\max}]$, where $\vec{X}_{\min} = \{x_{1,\min}, x_{2,\min}, \ldots, x_{D,\min}\}$ and $\vec{X}_{\max} = \{x_{1,\max}, x_{2,\max}, \ldots, x_{D,\max}\}$ with $i = [1, 2, \ldots, NP]$.

Step 2. WHILE the stopping criterion is not satisfied

DO

FOR $i = 1$ to NP /* do for each individual sequentially*/

Step 2.1 *Mutation Step*

Generate a donor vector $\vec{V}_{i,G} = \{v_{1,i,G}, \ldots, v_{D,i,G}\}$ corresponding to the ith target vector $\vec{X}_{i,G}$ using the mutation step in (2).

Step 2.2 *Crossover Step*

Generate a trail vector $\vec{U}_{i,G} = \{u_{1,i,G}, \ldots, u_{D,i,G}\}$ for the ith target vector $\vec{X}_{i,G}$ through the crossover step given in (3).

Step 2.3 *Selection Step*

Evaluate the trail vector $\vec{U}_{i,G}$.

IF $f(\vec{U}_{i,G}) \leq f(\vec{X}_{i,G})$

THEN $\vec{X}_{i,G+1} = \vec{U}_{i,G}$, $f(\vec{X}_{i,G+1}) = f(\vec{U}_{i,G})$

END IF

ELSE $\vec{X}_{i,G+1} = \vec{X}_{i,G}$, $f(\vec{X}_{i,G+1}) = f(\vec{X}_{i,G})$

END FOR

Step 2.3 Generation Count is incremented using $G = G + 1$

END WHILE

3.2. Hybrid Differential Artificial Bee Colony Algorithm

In some real world problems, DE may occasionally stop proceeding towards the global optimum even though the population has not converged to a local optimum or any other point.[17] In few situations occasionally, even when new individuals may enter the population, but the algorithm does not progress by finding better solutions. This situation is usually referred to as *stagnation*. DE also suffers from the problem of premature convergence, where the population converges to some local optima of a multimodal objective function, losing its diversity. Like other evolutionary computing algorithms, the performance of DE deteriorates with the growth of the dimensionality of the search space as well. The advantage of nature inspired heuristic global search algorithms is less likely to be entrapped in local optima, but the convergence rate will slow down and the computational complexity is high at later stage. Hybridizing the Differential Evolution with heuristic algorithms is expected to provide better convergence and desired values.

Artificial Bee Colony Algorithm (ABCA) is a new Meta heuristic global search algorithm. The goal of integrating DE with ABCA is to combine their advantages, and to decrease their disadvantages.[32] In addition, the search pattern of ABC combining with DE can enrich the search strategies.

In general hybridization schemes are broadly categorized into two types: the staged pipelining type hybrid and the other is additional-operator type hybrid. In the first type of hybridization, the optimization process is applied to each and every individual in the population, followed by further improvement using Differential Evolution search. In the second type of hybridization, the Differential Evolution is applied as a standard genetic operator for a given corresponding probability. In this hybridization scheme we employed pipelining type hybrid method because of its advantages. According to this method every generation, after the stochastic optimization process (i.e., Artificial Bee Colony) is applied to all individuals in the population, select n best Differential vectors from the current population based on the fitness values to generate initial the population required for local search via Differential Evolution. This search continues till it reach maximum number of generations or it satisfies pre defined criterion.

The steps of HDABCA are summarized as follows

(1) Initialization

(2) Move the employed bees onto their food sources and evaluate their nectar amounts.

(3) Place the onlookers depending upon the nectar amounts obtained by employed bees.

(4) Send the scouts for exploiting new food sources.

(5) Memorize the best food sources obtained so far.

(6) Select best (best fitness) n differential vectors from the population based on the fitness calculated to generate initial population for Differential evolution

(7) DO

FOR i = 1 to number of particles

```
DO
{
        Mutation
        Crossover
        Selection
}
```

END FOR

(8) If a termination criterion is not satisfied, go to step 2; otherwise stop the procedure and display the best solution is obtained so far

4. EXPERIMENTATION AND RESULTS

In order to evaluate the performance of Hybrid Differential Artificial Bee Colony Algorithm, we have used a test bed of 10 traditional numerical benchmarks as illustrated in Table I. Empirical results of the proposed Hybrid method have been compared with results obtained with that of basic Differential Evolution and Artificial Bee Colony algorithm. Based on the complexity of the function, the number of iterations is changed i.e., for example

RESEARCH ARTICLE

Table I. Description of the benchmark functions used.

Function	Mathematical representation	Dimension (D)	Range of search (S)	Theoretical optimum $f_{\min}$
Sphere function (f_1)	$f_1(\vec{x}) = \sum_{i=1}^{D} x_i^2$	30	$(-100, 100)^D$	$f_1(\vec{0}) = 0$
Rosenbrock (f_2)	$f_2(\vec{x}) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	30	$(-30, 30)^D$	$f_2(\vec{1}) = 0$
Rastrigin (f_3)	$f_3(\vec{x}) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	$(-5.12, 5.12)^D$	$f_3(\vec{0}) = 0$
Grienwank (f_4)	$f_4(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$(-600, 600)^D$	$f_4(\vec{0}) = 0$
Ackley (f_5)	$f_1(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{\mathrm{D}}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{\mathrm{D}}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	30	$(-30, 30)^D$	$f_5(\vec{0}) = 0$
Step (f_6)	$f_6(\vec{x}) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	30	$(-100, 100)^D$	$f_6(\vec{p}) = 0$ $-\frac{1}{2} \le p_i < \frac{1}{2}$
Schwefel's problem (f_7)	$f_7(\vec{x}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	$(-500, 500)^D$	$f_7(\vec{0}) = 0$
Schaffer's F6 function (f_8)	$f_8(\vec{x}) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$	2	$(-100, 100)^D$	$f_8(\vec{0}) = 0$
Six-Hump Camel-Back function (f_9)	$f_9(\vec{x}) = (4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^6)$	2	$(-5, 5)^D$	$f_9(0.08983, -0.7126) = f_9(-0.08983, 0.7126) = -1.0316285$
Goldstein-Price function (f_{10})	$f_{10}(\vec{x}) = \{1 + (x_0 + x_1 + 1)^2 (19 - 14x_0 + 3x_0^2 - 14x_1 - 6x_0x_1 + 3x_1^2)\}\{30 + (2x_0 - 3x_1)^2 \times (18 - 32x_0 + 12x_0^2 + 48x_1 - 36x_0x_1 + 27x_1^2)\}$	2	$(-2, 2)^D$	$f_{10}(0, -1) = 3$

if the maximum iterations are equal to 1000 it indicates that the maximum iterations are equivalent to 100,000 fitness evaluations. The values of the control parameters of HDABC algorithm used in the simulation studies and the values assigned for the control parameters of ABC and DE are given in Table II. D denotes the dimensionality of the test problem, S denotes the ranges of the variable, and $f_{\min}$ is the function value of the global optimum.

4.1. Benchmarks

Functions $f_1 - f_7$ are high-dimensional problems. Functions f_1, f_2, f_7 are uni-modal. Function f_6 is the step function, which has only one minimum and that too it is discontinuous. Functions f_3, f_4, f_5 are multimodal functions, where the local minima number increases exponentially with respect to problem dimension. Functions $f_8 - f_{10}$ are low-dimensional functions which have only a few local minima.

Table II. Algorithmic parameters.

Parameters	DE	ABC
Population size:- particles/bees	20	20
F	0.5	Not required
CR	0.8	Not required
Colony size	—	20
n_e	—	50% of colony
n_o	—	50% of colony
n_e	—	1
FS	Not required	10
Limit	Not required	$n_e{}^*D$

popsize, population size; CR, crossover factor in DE; f, scaling factor; n_e, employed bee number; n_o, onlooker number; n_e, scout number; FS, no of food sources (position); D, dimension of the problem.

4.2. Algorithms Used for the Comparative Study and Their Parametric Setup

(1) *Artificial Bee Colony Algorithm:-* Various control parameters of artificial bee colony algorithm used in the simulation studies are given in Table II. The percentage of onlooker bees was 50% of the colony size; the employed bees were 50% of the colony and the number scout bees were selected to be one for each and every cycle. In the ABC algorithm the number of onlooker bees us taken equal to that of employed bees to decrease the control parameters and to be handy.
(2) *Differential Evolution:-* The control parameters, which govern the search pattern of Differential Evolution, are presented in Table II. Here *popsize* represents the population or particles, F denotes the scaling factor followed by CR that represents the crossover factor that controls the recombination of vectors.
(3) *Hybrid Differential Artificial Bee Colony Algorithm:-* The proposed hybrid method also employs the same parametric set up that had been employed by the Artificial Bee Colony Algorithm and Differential Evolution. The only change is that some of the numerical values are been effected than that of basic parameters. In this hybrid

scheme, once the ABC completes stochastic optimization. Best 10 food source are picked out based on the fitness and these serves as the initial population for Differential Evolution algorithm. Then the population is finely tuned by DE based on the operators: Mutation, crossover and selection. As the optimal value cannot be obtained in one iteration we set it as 20 generations i.e., after every ABC iteration DE was allowed to run based on the user defined generations. Hybridizing these two optimization techniques are expected to bring best values with in less generations and also to have greater convergence speed.

4.3. Simulation Strategy

The comparative study used in this paper on different benchmarks, focuses on the following performance indices: 1) the quality of final solution; 2) the convergence speed [measured in terms of number of generations]. Thirty independent runs of each of the algorithms were carried out and the mean (average) and the standard deviation of the best-of-run values were recorded.

The proposed algorithm along with other algorithms i.e., DE and ABC, total three algorithms were executed for a given function of a given dimension for about 30 independent runs, and the average best-of-run value and the standard deviation were obtained. Different maximum numbers of generations were used according to the complexity involved in the problem.

4.4. Empirical Results

Table III compares the algorithms on the quality of the best solutions obtained. The mean and the standard deviation (within parentheses) of the best-of-run solution for 50 independent runs of each of the 10 algorithms are presented in Table III. A special note that in this table, if all the runs of particular algorithm converge to or below the pre-specified objective function value (0.001 for f_1 to f_7; 0, -1.0316, and 3.00 for f_8, f_9 and f_{10}, respectively) within the maximum number of generations, then we report this threshold value as the average or mean of 30 runs. Table III illustrates, for all test functions and all algorithms, the number of runs i.e., out of 50 that managed to find the optimum solution (within the given tolerance) and also the average number of generations taken to reach the optima value along with the standard deviation (in parentheses). The convergence characteristics of eight most difficult benchmarks have been provided in Figure 1. Each Figure depicts how the objective function value of the best individual in a population changes with increasing number of generations (or iterations).

4.5. Discussion and Analysis on the Results

From Table III, it is observed that the performance of hybrid algorithm remained consistently superior to that of the original DE and ABC over all benchmark problems. The sphere function (f_1) is perhaps the easiest among all test functions (benchmarks). From Table III, we clearly understand that for the 30-dimensional sphere 30 runs of ABC, hybrid ABC-DE converged to or below the pre-specified objective function value of 0.001. Similar is the case for the step function (f_6) in 30 dimensions. However hybrid algorithm was found to yield better average accuracy than the remaining two methods over the functions f_3, f_4 and f_5. For functions f_9 and f_{10}, since the optimum is not located at the origin, as expected, hybrid ABC-DE outperformed than the ABC and DE. In the case of function f_2 even though the pre specified criterion is not obtained through any of the algorithm, hybrid method has got good convergence and best values than the other methods. In almost all test functions concerned ABC out performed DE (nine out of ten) except in the case of function f_7, where DE got better solution and it is unable to achieve the pre specified value which is obtained by making use of proposed method.

Table III and Figures 1(a–f) are indicative of the fact that the convergence behaviour of the hybrid method has been considerably improved in comparison to that of their remaining methods. From Table III, it is clear and evident that in almost all the cases considered, hybrid method produce most accurate results but they do so consuming the least amount of computational time (measured in terms of the generations needed to converge).

Table III. Average and the standard deviation (in parenthesis) of the best-of-run solution for 50 independent runs tested on 10 benchmark functions.

		Mean best value (Standard deviation)		
Function	#Gen	DE	ABC	Hybrid ABC-DE
f_1	2000	208.696 (411.022)	6.66217E-016 (1.1712E-016)	3.17421E-017 (1.25367E-032)
f_2	2000	1.24701 (2.272191)	2.21131 (3.89826)	0.109065 (5.64601E-017)
f_3	3000	50.326 (14.353)	7.82E-007 (2.54E-006)	0 (0)
f_4	2000	26.517 (21.9722)	0.0079474 (0.0129966)	0 (0)
f_5	2000	7.3131 (2.08131)	1.45708E-12 (7.25094E-13)	4.44089E-15 (0)
f_6	2000	135.133 (298.731)	0 (0)	0 (0)
f_7	2000	146.997 (198.103)	1.11565E-008 (2.49478E-008)	5.52032E-017 (2.50733E-032)
f_8	200	0.0001042 (0.0005708)	0.0024785 (0.0038469)	0 (0)
f_9	200	-1.03154 (0.0004728)	-1.03163 (3.6153E-007)	-1.03163 (6.77522E-016)
f_{10}	200	3.9 (4.9295)	3.40524 (0.743419)	3 (1.80672E-15)

RESEARCH ARTICLE

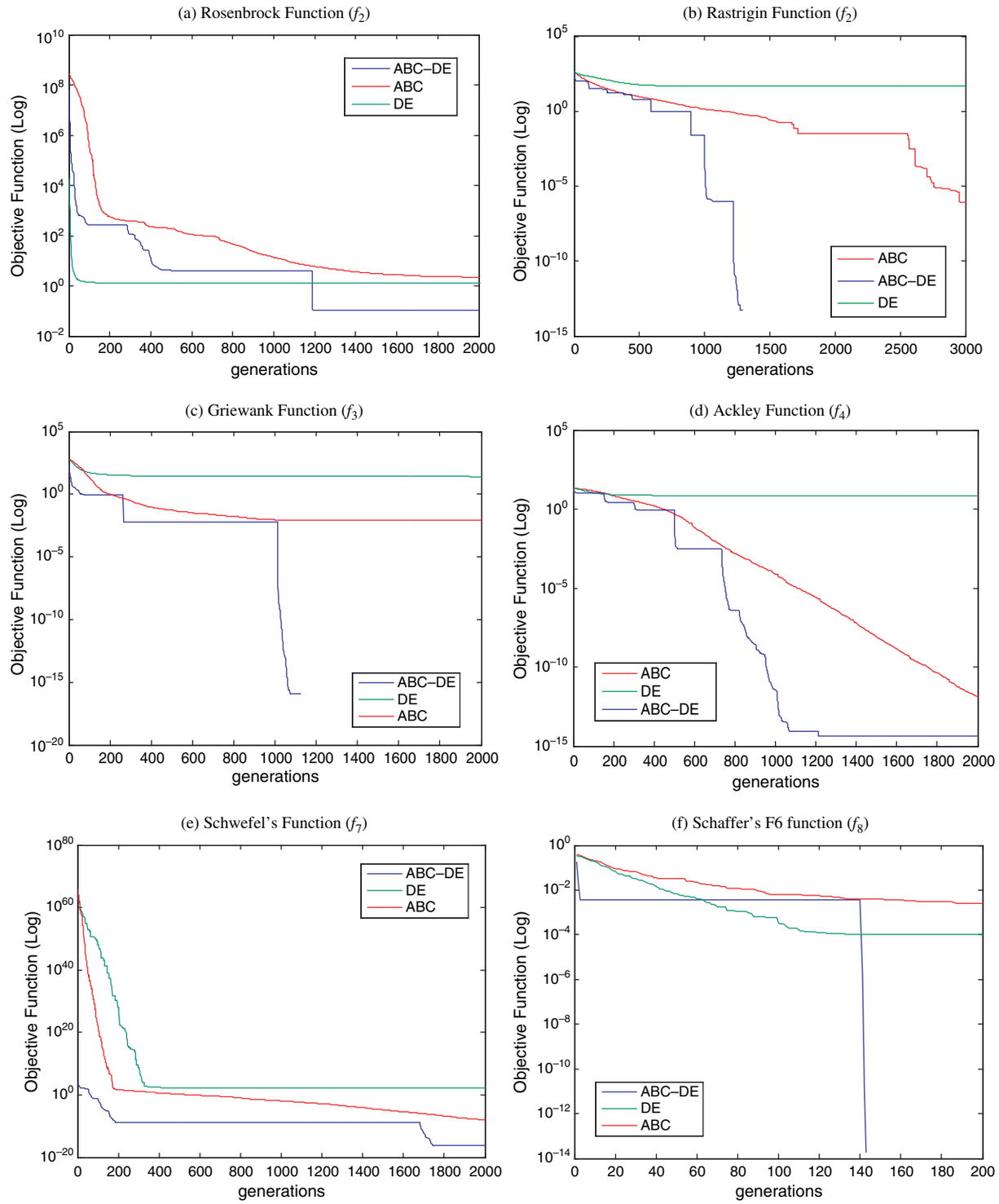

Fig. 1. Progress toward the optimum solution: (a)–(e) for dimension = 30 and (f) for dimension = 2.

5. CONCLUSIONS

In the article, the performance of hybrid differential artificial bee colony algorithm (ABC-DE) was investigated for function optimization problems. The proposed method is illustrated using various test functions and as evident from the graphical and empirical results the suggested hybrid method ABC-DE performed extremely well. All the design criteria have been satisfied with in less computational time for the test functions.

Our further research would include the performance and analysis fractional order controller in a sensor less motor drives. The proposed approach is likely to be useful for other real world applications (for e.g., dynamic load dispatch, harmonic estimation, radar tracking) as swarm intelligent techniques work very well in such domains.

References

1. A. Abraham, A.-E. Hassanien, P. Siarry, and A. Engelbrecht, Foundations of Computational Intelligence Global Optimization, Studies in Computational Intelligence, Springer Verlag, Germany **(2009)**, Vol. 3, p. 300.
2. L. J. Fogel, A. J. Owens, and M. J. Walsh, Artificial intelligence through a simulation of evolution, edited by M. Maxfield, A. Callahan, and L. J. Fogel, Biophysics and Cybernetic systems, *Proceeding of the 2nd Cybernetic Sciences Symposium* **(1965)**, pp. 131–155.
3. D. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison Wesley Publishing Company, Reading, Massachutes **(1986)**.
4. J. Kennedy and R. C. Eberhart, Particle Swarm Optimization, *Proceeding of IEEE International Conference on Neural Networks*, Perth, Australia, IEEE Service Center, Piscataway, NJ **(1995)**, pp. 1942–1948.
5. D. Karaboga, An Idea based on Bee Swarm for Numerical Optimization, Technical Report, TR-06, Erciyes University Engineering Faculty, Computer Engineering Department **(2005)**.
6. D. Karaboga and B. Basturk, A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) algorithm, Journal of Global Optimization, Springer Netherlands **(2007a)**, Vol. 39, pp. 459–471.
7. D. Karaboga and B. Basturk, Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems, LNCS: Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, Springer-Verlag, IFSA **(2007b)**, pp. 789–798.
8. G. D. H. Kim, *Applied Soft Computing* 7, 601 **(2006)**.
9. M. Pant, R. Thangaraj, and A. Abraham, Optimization of a Kraft Pulping System: Using Particle Swarm Optimization and Differential Evolution, *Proceeding of 2nd Asia International Conference on Modeling and Simulation*, Malaysia, IEEE Computer Society Press, USA **(2008a)**, pp. 637–641.
10. M. Pant, R. Thangaraj, and A. Abraham, *Journal of Electrical Engineering* 2, 36 **(2008b)**.
11. M. Pant, R. Thangaraj, and V. P. Singh, *Int. Journal of Recent Trends in Engineering* 1, 21 **(2009)**.
12. K. Price and R. Storn, Differential Evolution – a Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces, Technical Report, International Computer Science Institute, Berkley **(1995)**.
13. M. Omran, P. Engelbrecht, and A. Salman, Differential Evolution Methods for Unsupervised Image Classification, *Proceeding of the 7th Congress Evol. Comput.*, Piscataway, NJ: IEEE Press **(2005)**, Vol. 2, pp. 966–973.
14. F. S. Wang and H. J. Jang, Parameter Estimation of a Bio-Reaction Model by Hybrid Differential Evolution, *Proceeding IEEE Congress Evol. Comput.*, Piscataway, NJ: IEEE Press **(2000)**, Vol. 2, pp. 410–417.
15. J. Vesterstrom and R. Thomson, A Comparative Study of Differential Evolution, Particle Swarm Optimization, and evolutionary algorithms on numerical benchmark problems, *Proceeding of the 6th Congress Evol. Comput.*, Piscataway, NJ: IEEE Press **(2004)**, Vol. 2, pp. 1980–1987.
16. U. K. Chakraborty, Advances in Differential Evolution, Heidelberg, Germany: Springer-Verlag **(2008)**.
17. J. Lampinen and I. Zelinka, On stagnation of the differential evolution algorithm. *Proceeding MENDEL, 6th International Mendel Conference Soft Computing*, Brno, Czech Republic **(2000)**, pp. 76–83.
18. F. Kang, J. J. Li, and Q. Xu, *Computers and Structures* 87, 861 **(2009)**.
19. H. J. C. Barbosa, C. C. Lavor, and F. M. P. Raupp, *Ann. Oper. Res* 138, 189 **(2005)**.
20. B. Pryymak, J. M. Moreno-Eguilaz, and J. Peracaula, *Mathematics and Computers in Simulation* 71, 290 **(2006)**.
21. I. Rechenberg, Evolution Strategy: Optimization of Technical Systems by Means of Biological Evolution, Fromman-Holzboog **(1973)**.
22. S. Sabat, S. Udgata, and A. Abraham, *Elsevier Science* 23, 689 **(2010)**.
23. R. Thangaraj, M. Pant, and A. Abraham, Evolutionary Algorithms based Speed Optimization of Servo Motor in Optical Disc systems, *Proceeding of 8th International Conference on Computer Information Systems and Industrial Management Applications*, India, IEEE Computer Society Press **(2009)**, pp. 855–860.
24. R. Thangaraj, M. Pant, and K. Kusum Deep, *Elsevier Science* 23, 820 **(2010)**.
25. Z. L. Gaing, *IEEE Trans. on Energy Conversion* 19, 384 **(2004)**.
26. D. H. Kim, A. Abraham, and J. H. Cho, *Elsevier Science* 177, 3918 **(2007)**.
27. G. G. Roy, P. Chakraborty, and S. Das, *International Journal of Bio-Inspired Computation* 2, 303 **(2010)**.
28. L.-P. Wong, C. Y. Puan, M. Y. H. Low, Y. W. Wong, and C. S. Chong, *International Journal of Bio-Inspired Computation* 2, 85 **(2010)**.
29. S. Das, A. Biswas, A. Abraham, and S. Dasgupta, *Elsevier Science* 22, 343 **(2009)**.
30. S. Das, A. Abraham, and A. Konar, Automatic Clustering Using an Improved Differential Evolution Algorithm, IEEE Transactions on Systems Man and Cybernetics-Part A, IEEE Press, USA **(2008)**, Vol. 38, pp. 218–237.
31. S. Das, A. Abraham, U. Chakraborty, and A. Konar, Differential Evolution Using a Neighborhood based Mutation Operator, IEEE Transactions on Evolutionary Computation, IEEE Press, USA **(2009)**, Vol. 13, pp. 526–553.
32. F. Kang, J. Lia, and Q. Xua, *Computers and Structures* 87, 861 **(2009)**.

Received: 5 November 2009. Accepted: 30 November 2010.

RESEARCH ARTICLE