Applied Soft Computing 58 (2017) 11–24

Contents lists available at ScienceDirect

Applied Soft Computing

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

Hybrid Artificial Bee Colony algorithm with Differential Evolution

Shimpi Singh Jadon[a,*], Ritu Tiwari[a], Harish Sharma[b], Jagdish Chand Bansal[c]

[a] ABV-Indian Institute of Information Technology and Management, Gwalior, India
[b] Rajasthan Technical University, Kota, India
[c] South Asian University, New Delhi, India

ARTICLE INFO

Article history:
Received 7 November 2014
Received in revised form 3 April 2017
Accepted 13 April 2017
Available online 20 April 2017

Keywords:
Artificial Bee Colony
Differential Evolution
Optimization
Hybridization
Swarm intelligence

ABSTRACT

Artificial Bee Colony (ABC) and Differential Evolution (DE) are two very popular and efficient meta-heuristic algorithms. However, both algorithms have been applied to various science and engineering optimization problems, extensively, the algorithms suffer from premature convergence, unbalanced exploration-exploitation, and sometimes slow convergence speed. Hybridization of ABC and DE may provide a platform for developing a meta-heuristic algorithm with better convergence speed and a better balance between exploration and exploitation capabilities. This paper proposes a hybridization of ABC and DE algorithms to develop a more efficient meta-heuristic algorithm than ABC and DE. In the proposed hybrid algorithm, Hybrid Artificial Bee Colony with Differential Evolution (HABCDE), the onlooker bee phase of ABC is inspired from DE. Employed bee phase is modified by employing the concept of the best individual while scout bee phase has also been modified for higher exploration. The proposed HABCDE has been tested over 20 test problems and 4 real-world optimization problems. The performance of HABCDE is compared with the basic version of ABC and DE. The results are also compared with state-of-the-art algorithms, namely Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Particle Swarm Optimization (PSO), Biogeography Based Optimization (BBO) and Spider Monkey Optimization (SMO) to establish the superiority of the proposed algorithm. For further validation of the proposed hybridization, the experimental results are also compared with other hybrid versions of ABC and DE, namely ABC-DE, DE-BCO and HDABCA and with modified ABC algorithms, namely Best-So-Far ABC (BSFABC), Gbest guided ABC (GABC) and modified ABC (MABC). Results indicate that HABCDE would be a competitive algorithm in the field of meta-heuristics.

1. Introduction

Swarm intelligence has emerged as an effective tool for numerical optimization, which runs over the collaborative trial and error method. The popular techniques based on swarm intelligence are Particle Swarm Optimization (PSO) [27], Biogeographic based optimization (BBO) [40], Bacterial foraging optimization (BFO) [35], Firefly algorithm [30], Ant colony optimization (ACO) [12] and Spider monkey optimization [8]. The work proposed in the articles [12,27,37,43] provides the evidence of its efficiency to find the solution of optimization problems of typical characteristics like nonlinearity, nonconvexity, and discrete search space. Artificial Bee Colony (ABC) [23] is well known optimization algorithm in this category.

ABC algorithm is a simulation of a particular behavior of honey bees known as foraging behavior (a search for food). It is easy to implement population-based optimization algorithm with very few number of parameters. Here the population includes possible solutions as food sources for honey bees. The food source's fitness is proportional to the nectar amount that it contains. In ABC, each bee moves to other food sources through position update Eq. (2) in Section 2 which is a linear combination of position of its current food source and position of a randomly selected food source with a random coefficient ϕ as step size. Due to the involvement of these random quantities, ABC is found to be good at exploration and with the lack of exploitation, i.e., incapable of applying available information to find better solution [48]. Researchers in [25,28] also analyzed this fact and found that it will ultimately affect the ABC algorithm's convergence rate. Li et al. [29] also found that ABC suffers from convergence speed when we are dealing with some complex problems. These drawbacks may be dealt by modifying existing position update equation and/or by hybridizing another fast optimization algorithm with ABC. The articles [7,16,26,48]

* Corresponding author.
E-mail addresses: shimpi@iiitm.ac.in (S.S. Jadon), ritutiwari@iiitm.ac.in (R. Tiwari), hsharma@rtu.ac.in (H. Sharma), jcbansal@sau.ac.in (J.C. Bansal).

http://dx.doi.org/10.1016/j.asoc.2017.04.018
1568-4946/

worked to improve exploitation in ABC by modifying its position update equation while previous research in [13,29,34,47] have also shown that hybrid ABC with different algorithms can perform better by integrating the respective advantages of the independent algorithms. Jadon et al. [21] modified position update equation of basic ABC algorithm. In this modification, fitness of randomly selected solution directs the sign of step size to be added in the current position to generate its neighborhood solution in employed and onlooker bee phases. Li et al. [29] proposed a hybrid version of ABC and DE algorithms and applied it to optimal reactive power flow. Amir et al. [6] hybridized DE in ABC to create new solutions for both employed and onlooker bees for unconstrained optimization problems. Alizadegan et al. [6] proposed a hybrid ABC and DE (ABC–DE) in which DE is incorporated in employed and onlooker phases. Ali et al. [47] also introduced a novel hybrid optimization method (HRABC) consisting of ABC and Taguchi method for structural design optimization. Duan et al. [13] hybridized ABC into Quantum Evolutionary Algorithm (QEA) where ABC is adopted to enhance the local exploitation capacity and randomness of the QEA populations. Abraham et al. [2] also incorporated DE process after each ABC iteration. In [34], Levenberq–Marquardt (LM) strategy is also hybridized with ABC and tested to train neural networks. In DEBCO [1], employed bee of ABC finds the neighborhood solutions through DE. Thammano et al. [42] hybridized five distinct search techniques at various levels of the ABC to solve job shop scheduling problem. Here, harmony search algorithm is used for initialization of the population. The iterated local search scheme, the scatter search method, and the filter and fan techniques are applied to search neighborhood solutions. The simulated annealing algorithm is also hybridized and applied to get a solution out of local optimum. Kang et al. [22] hybridized basic ABC with Hooke-Jeeves based local search method [19] known as HJABC. In HJABC, a selection pressure and solution ranking are used to calculate fitness function.

This paper proposes a hybrid version of ABC and DE, which also incorporates the modification in the position update equation of ABC. The proposed hybrid algorithm is named as HABCDE. In HABCDE, the employed, onlooker and scout bee phases of ABC are modified. In employed bee phase, a bee moves to other food source not only based on a randomly selected food source but also based on the current best food source. Gbest-guided ABC [48] has already applied the best solution information to update the position of any bee. In onlooker bee phase, it updates the bee's position through evolutionary operations of Differential Evolution (DE/best/1/bin) algorithmic process. The number of scouts is increased in scout phase to give a chance to re-initialize themselves to all those bees who are not being updated to a predefined number of times.

The organization for the rest in this article is as follows: Standard ABC is explained in Section 2. Section 3 reports classical DE algorithm. Section 4 details the proposed hybrid ABC (HABCDE). In Section 5, the performance of the proposed scheme is examined and measured with recent variants of ABC. Comparison is also done with some state-of-the-art algorithms and with hybrid versions of ABC. Finally, paper is concluded in Section 6.

2. Artificial Bee Colony (ABC) algorithm

The ABC algorithm was developed by Karaboga which is a simulation of food foraging behavior of real honey bees. In ABC, the food source for bees are called as solutions. ABC is composed of three types of bees namely, the employed, the onlooker and the scout. ABC colony consists equal number of employed and onlooker bees. Employed bees explore the food source in the surroundings of their hive and store the related information in their memories. Onlooker bees collect the information from employed bees in the hive to select food sources for further extraction of nectar. If the nectar quantity in food source is low or exhausted, then scout bee randomly finds a new food source in search space. The step by step description of ABC algorithmic procedure is as follows:

2.1. Initialization of the swarm

The initial solutions $x_i (i = 1, 2, \ldots, SN)$ of swarm is generated using a uniform distribution as follows:

$$x_{id} = x_{mind} + rand[0, 1](x_{maxd} - x_{mind}) \quad (1)$$

here x_i is the ith potential solution in the swarm, x_{mind} and x_{maxd} are bounds of x_i in dth dimension and $rand[0, 1]$ is a uniformly distributed random number in the interval [0, 1].

2.2. Employed bee phase

In this phase, each bee moves to other solution (food source) and this process is executed as follows:

$$v_{id} = x_{id} + \phi_{id}(x_{id} - x_{kd}) \quad (2)$$

here, i is current solution, k is a randomly selected solution among the SN solutions of the swarm such that $k \neq i$ and d is any randomly chosen dimension. ϕ_{id} is a random number in the interval $[-1, 1]$. Now, the greedy selection is applied between the new solution and old solution to memorize better one in terms of fitness.

2.3. Onlooker bees phase

In this phase, we try to exploit better solutions in their surroundings. So, solutions are selected based on a probability $prob_i$ for further generation of new solutions in their neighborhoods. Here $prob_i$ is calculated based on fitness:

$$prob_i(G) = \frac{0.9 \times fitness_i}{maxfit} + 0.1, \quad (3)$$

here $fitness_i$ represents the fitness of the ith solution. In case of maximization optimization problem, the $fitness_i$ is equal to objective function value and it is equal to negative of objective function value in case of minimization problem. $maxfit$ is the fitness of best solution so far. After selecting solutions, the new neighborhood solutions are generated using the same Eq. (2). Again greedy selection is applied between the current and the old positions to memorize one of these by the onlooker bees.

2.4. Scout bees phase

If for a prefix duration or iterations, any solution is unable to update itself then it is considered as abandoned solution and the corresponding bee becomes the scout. In ABC, the crucial control parameter (the number of iterations) after which a particular solution is considered exhausted is known as *limit*. After each iteration, only one solution which has not been updated for a maximum number of iterations out of all exhausted solutions is selected. The bee associated with selected solution search new food source in the search space randomly using the Eq. (1). In basic ABC, at most one scout bee can reinitialize itself.

2.5. Artificial Bee Colony algorithm

The pseudo-code of the basic ABC is taken from [24] and is shown in Algorithm 1.

Algorithm 1. ABC algorithm:

Initialize the control parameters;
while !Termination criteria **do**
  Employed bee phase: generate neighborhood solutions for each solution in the swarm;
  Onlooker bees phase: generate neighborhood solutions for the solutions selected based on their probabilities;
  Scout bee phase: reinitialize the exhausted solutions in the search space randomly;
  Memorize the best solution;
end while
Output the best solution.

3. Differential Evolution

DE is a population-based optimization algorithm, where members of the population are potential solutions which collaboratively search solutions. DE has both the evolutionary and swarm intelligence based features as it includes evolutionary operators like mutation, crossover, selection and swarm intelligence concept like distance and direction of the individual solutions to guide the search process further. DE has different formats to apply to solve the optimization problem, e.g., it has various selection methods for target vector, the number of difference vectors used in update equation and the types of crossover operator to be used [36]. This paper uses the *DE/best/1/bin* format of DE, which explains that selection of target vector will be the best solution, exactly '1' number of differential vectors will be used, and 'bin' indicates that DE will use the *binomial* crossover. Each individual member of DE population is represented by a D-dimensional vector $x_i(i = 1, 2, \ldots, D)$.

The whole DE process consists of three phases: generation of the trial vector, generation of the offspring and selection between the parents and the offsprings to form next generation. Mutation, crossover, and selection are the three operators which make phases mentioned above to be executed. In DE, Initially by using uniform distribution, a population of fixed size is generated in the search region. Then, to generate next population, these three mentioned phases take place. The critical and essential part of whole DE process is the generation of offspring vector which involves two, the mutation and the crossover operators. Finally greedy selection is made between parents and offsprings to select the best vectors for the next generation. Following subsections briefly describe the working of DE operators.

3.1. Mutation

For each member of the population, DE mutation operator develops a trial vector. For generating the trial vector, best solution in the current population is selected as a target vector and is mutated with a weighted differential. The mutation process to generate a trial vector $u_i(g)$ for the parent vector $x_i(g)$ is defined as follows:

- Choose the best vector $x_{best}(g)$ in the population,
- Choose two members x_{i_1} and x_{i_2} randomly in the population such that $i \neq i_1 \neq i_2$.
- Generate trial vector by mutating the target vector as follows:

$$u_i(g) = x_{best}(g) + F \times (x_{i_1}(g) - x_{i_2}(g)) \tag{4}$$

where, g is the iteration count, $F \in [0, 1]$ is the scale factor which indicates that how much amount of differential variation [15] will influence on target vector.

3.2. Crossover

The trial vector $u_i(g)$ and parent $x_i(g)$ are used in crossover to generate Offspring $x'_i(g)$ as follows:

$$x'_{id}(g) = \begin{cases} u_{id}(g), & \text{if } d \in S \\ x_{id}(g), & \text{otherwise.} \end{cases} \tag{5}$$

where S is the set of crossover points, $x_{id}(g)$ is the dth dimension of the vector $x_i(g)$.

Literature suggests various techniques to define the set S, exponential and binomial are the most frequent crossover strategies amongst them [15]. Here the binomial crossover is used in the proposed hybridization HABCDE. In binomial crossover, the set S randomly picks the crossover points from the set of possible crossover points. Following Algorithm 2 explains the procedure to generate crossover points [15].

Algorithm 2. Binomial crossover:

$S = empty\ set$
$d^* = U(1, D)$;
$S = S \cup d^*$;
for every $d \in 1 .. D$ **do**
  if $U(0, 1) < CR$ and $d \neq d^*$ **then**
    $S = S \cup d$;
  end if
end for

Here, D is the dimension of the problem, d^* is a randomly selected dimension to be included in D to ensure at least one crossover point, CR represents the probability which decides the inclusion of considered crossover point. The higher CR reflects in the selection of more crossover points. $U(1, D)$ is a random integer between 1 and D and S is a set of crossover points.

3.3. Selection

In DE, selection operator is applied at two points: first, where individual members of the population are selected for the mutation operation to generate the trial vector. This selection may be either random or any individual having best fitness in the current population. Second, where greedy selection (explained below) is made between the parent and the offspring, i.e., the one who has higher fitness will move into the next generation. If $f(*)$ is the fitness function then:

$$x_i(g+1) = \begin{cases} x'_i(g), & \text{if } f(x'_i(g)) > f(x_i(g)). \\ x_i(g), & \text{else.} \end{cases} \tag{6}$$

Algorithm 3 explains the pseudo-code for DE algorithm [15]. In Algorithm 3, crossover probability CR and scale factor F are the control parameters of the DE algorithm. P represents the population vector.

Algorithm 3. Differential Evolution algorithm:

Initialize population $P(0)$, control parameters F and CR;
while !Termination criteria **do**
  for every individual $x_i(g) \in P(g)$ **do**
    Evaluate $f(x_i(g))$;
    Employ mutation operator to generate trial vector $u_i(g)$;
    Employ crossover operator to generate offspring $x'_i(g)$;
    if $f(x'_i(g))$ is better than $f(x_i(g))$ **then**
      Include $x'_i(g)$ to $P(g+1)$;
    else
      Include $x_i(g)$ to $P(g+1)$;
    end if
  end for
end while
output the best solution;

4. Hybrid Artificial Bee Colony with Differential Evolution algorithm

As mentioned in Section 1, ABC algorithm may be improved by modifying its position update equation and/or by hybridizing it with other promising optimization algorithms. In this article, both the concepts are applied to improve basic ABC's efficiency. The proposed algorithm is a hybridization of ABC with DE algorithm and named as Hybrid Artificial Bee Colony with Differential Evolution (HABCDE) Algorithm. In HABCDE, three modifications, one in each phase of ABC, are proposed as follows:

1. **Employed bee phase**: From position update Eq. (2), it is clear that in basic ABC, a solution is updated using the information from a random solution x_{kj} of the current swarm. The selection of this random solution has a significant influence on the quality of the updated solution. The randomness of this solution is also required to maintain the diversified search throughout the search space. This fact motivates the authors to modify the Eq. (2) of ABC so that it can manage the property of diversity while not entirely dependent on the choice of the random solution x_{kj}. Therefore, best solution information is incorporated like Gbest-guided ABC (GABC) [48] in HABCDE. Therefore, the amount of change in the solution (say, step size) is supposed to balance the exploration (because of random solution) and exploitation (because of the presence of best solution) capabilities of the ABC algorithm. The position update equation used in proposed algorithm is:

$$v_{id} = x_{id} + \phi_{id}(x_{id} - x_{kd}) + \psi_{id}(y_d - x_{id}) \tag{7}$$

   where y_d is dth dimension of the best solution of the current swarm and ψ is a random number between (0, C), C is a positive constant defined by the user.
2. **Onlooker bee phase**: Since the convergence speed of DE is relatively better than ABC, the position update process of DE algorithm is implemented in onlooker bee phase of ABC. This has been done in expectation of achieving the faster convergence by ABC. In this phase, instead of using Eq. (2), onlooker bee exploits current food position through evolutionary operations of Differential Evolution (as in DE/best/1/bin, Algorithm 3). In the proposed hybridized solution search process of ABC and DE, the new solution (offspring) is generated through mutation and crossover operations. In this process first a trial solution is generated by mutating the best solution of the current swarm with the help of two randomly selected solutions from the swarm as in Eq. (4). Then binomial crossover operator is applied to the trial solution and the current solution (the solution which is to be updated) using Eq. (5) to create the offspring (new solution). Now, the greedy selection is applied to the current and offspring solution to select the better one.
3. **Scout bee phase**: To increase the exploration capability of the proposed algorithm, the number of scout bees is increased. It is reminded that in ABC if the trial counter (the not updating count) is maximum and crosses the 'limit' for a solution, the associated bee is called scout bee. Usually, only one scout can search new solution in the ABC. While, all the scout bees who are crossing the limit are allowed in the proposed strategy to reinitialize corresponding solutions in the given search space. Consequently, the bees corresponding to exhausted solutions are forced to explore the new solutions in search space.

The HABCDE is composed of three phases. Here, Employed bee phase uses position update Eq. (7) of GABC instead of Eq. (2). In onlooker bee phase, solutions are exploited through DE Algorithm 3 (DE/best/1/binomial crossover). The number of scout bees is increased in scout bee phases and rest of this phase remains same like in basic ABC. The Algorithm 4 explains the pseudo-code for the proposed HABCDE algorithm. DE algorithm is fast in convergence and ABC is better in exploration. Thus the hybrid variant of ABC and DE is expected to converge towards optimal solution quickly while maintaining the diversified search.

Algorithm 4. Hybrid ABC with Differential Evolution (HABCDE):

```
Initialize the swarm and control parameters;
while !Termination criteria do
  Employed bee phase: generate neighborhood solutions for each solution
    in the swarm using position update Eq. (7).
  Onlooker bees phase: generate neighborhood solutions for the solutions
    selected based on their probabilities as follows:
  for (each onlooker's solution x_i) do
    if (U(0, 1) < prob_i) then
          Apply mutation (as in Eq. (4)) to generate the trial solution u_i;
        Apply binomial crossover (as in Eq. (5)) to generate an offspring x'_i;
          if (f(x'_i) is better than f(x_i)) then
          x_i = x'_i;
        end if
      end if
  end for
  Scout bee phase: randomly reinitialize all the exhausted solutions in the
    search space.
end while
output the best solution.
```

The search process of proposed HABCDE is also explained through Fig. 1.

5. Experimental results and discussion

According to famous No Free Lunch Theorem (NFL) [46], designing a single algorithm which is better than all other algorithms, is not possible provided it is tested over a sufficient number of problems. However, the authors are not claiming that the proposed algorithm HABCDE is superior than other algorithms for all kind of problems; the experimental results have been carried out below over a set of benchmark optimization problems and a set of real-world optimization problems to show its wider applicability.

5.1. Benchmark optimization problems under consideration

The performance of the proposed HABCDE algorithm is tested over 20 mathematical optimization problems (f_1–f_{20}) (listed in Table 1). Test problems f_1–f_{13} and f_{17}–f_{20} are taken from [5] while test problems f_{14}–f_{16} are taken from [41]. This considered set of test problems is chosen such that the algorithm can be well tested for various challenges like separability, non-separability, unimodality and multimodality of the problems. Further, the proposed scheme is also applied to solve 4 well-known real-world optimization problems (f_{21}–f_{24}), described as follows:

Compression spring (f_{21}): This problem is 3 dimensional constrained optimization problem where the objective is to minimize the weight of a compression spring, subject to some physical, operational and practical constraints. The mathematical form of the problem is taken from [33,39]. The best known fitness solution is f^* = 2.6254214578 for solution vector (7, 1.386599591, 0.292). Here, acceptable error to consider a algorithm's run successful is fixed to be 1.0E−10.

Pressure vessel design (f_{22}): The pressure vessel design problem is a 4 dimensional constrained optimization problem where the objective is to minimize the total cost of the material, forming, and welding of a cylindrical vessel. The mathematical form of this problem is taken from [44]. The problem has best known solution f(1.125, 0.625, 55.8592, 57.7315) = 7197.729 [44]. The acceptable error is fixed to be 1.0E−05 for a successful run.

Frequency-modulated (FM) sound wave (f_{23}): Frequency-modulated sound wave synthesis is an important part of many modern music systems. Here, the objective is to optimize the

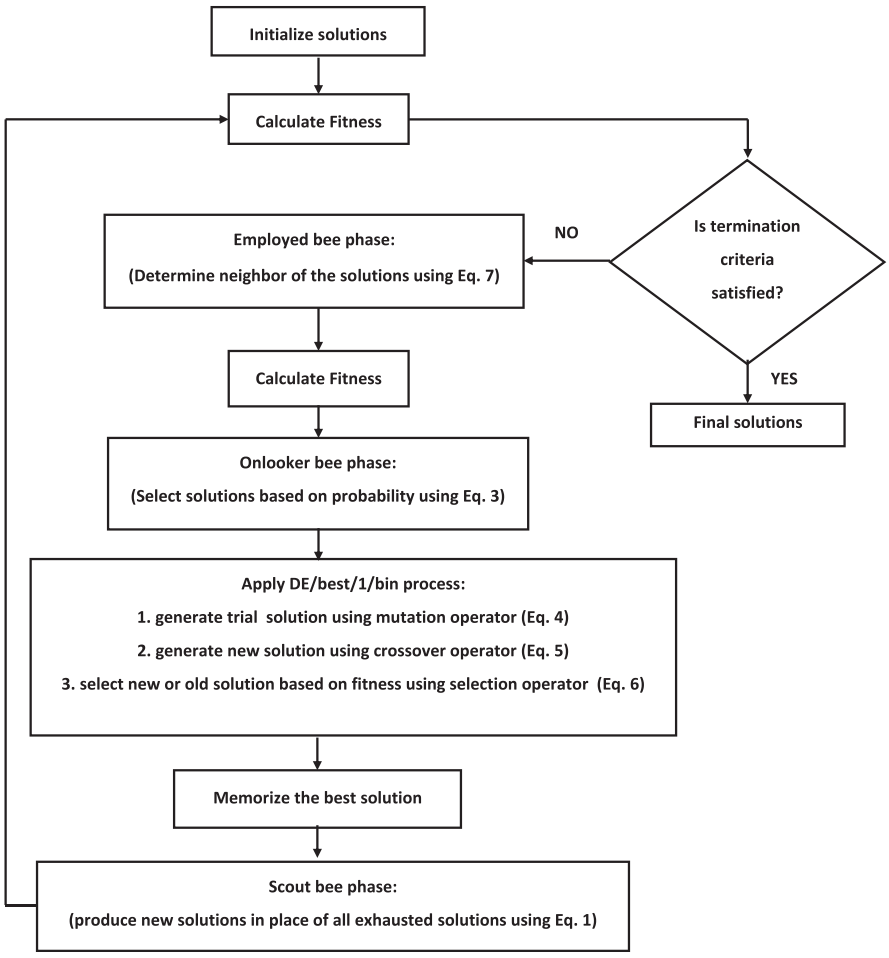


Fig. 1. Flowchart for HABCDE algorithm.

parameters of the sound wave. The mathematical form of the problem is taken from [10]. Acceptable error is fixed as 1.0E−05 for this problem.

Welded beam design optimization problem (f_{24}): The objective of Welded beam design problem is to minimize the fabricating cost of designing a welded beam subject to some physical, operational and practical constraints. The mathematical form of the problem is taken from [38,31]. The best known solution for the problem is (0.205730, 3.470489, 9.036624, 0.205729) and corresponding function value is 1.724852. Acceptable error for this problem is set as 1.0E−01.

5.2. Experimental setting

The proposed HABCDE algorithm is implemented in C++. All the experiments (HABCDE and other considered algorithms) are implemented on an Intel Core i5-2410M PC with 2.30 (GHz) processor speed, 4 (GB) RAM and window 7 operating system. The performance of the HABCDE are compared based on four factors, namely: success rate (SR), average number of function evaluations (AFE) and mean error (ME). Proposed algorithm HABCDE is compared with the basic ABC and some promising ABC variants: Gbest-guided ABC (GABC) [48], Modified ABC (MABC) [4], Best-So-Far ABC (BSFABC) [7]. HABCDE is also compared with some state-of-the-art algorithms like CMA-ES [18,17], DE, PSO, BBO and SMO and other hybrid versions of ABC and DE namely, ABC–DE [6], DEBCO [1] and HDABCA [3]. We have divided these considered algorithms in three sets namely, modified ABC algorithms S_1 = {ABC,BSFABC,GABC, MABC}, The state-of-the-art algorithms S_2 = {CMA-ES, DE, PSO, BBO, SMO} and hybrid ABC–DE algorithms S_3 = {ABC–DE, DEBCO, HDABCA} for the comparison purpose.

Parameter settings:

- The maximum number of runs =100,
- Population size SN = 25 [11,14],
- limit= $d \times SN$ [26,4], where d is the dimension of the problem,
- C = 1.5 in GABC update equation [48],
- The algorithm is terminated if either function evaluations reaches 200,000 or acceptable error (Table 1) is found for the test problem,
- To set the DE parameters F and CR, sensitive analysis (see Fig. 2) is carried out in the range [0.1, 1]. The horizontal axis in Fig. 2 shows a values of F, CR and the vertical axis shows the sum of successful runs for all the test problems under consideration. We can observe from this figure that a combination of F = 0.7 and CR = 0.6 clearly provides the highest success rate.
- The settings for the parameters of the algorithms in the sets S_1, S_2 and S_3 are selected as they are in respective original articles.

5.3. Analysis of results

Tables 2–4 present numerical comparison of all the considered algorithms of set S_1 for test problems f_1–f_{20} and real-world optimization problems f_{21}–f_{24} while Tables 5–7 present the comparison for state-of-the-art algorithms of set S_2 and Tables 8–10 present the comparison for hybrid algorithms of set S_3 for benchmark problems f_1–f_{20}. Tables 2–10 numerically compare the SR, AFE and ME

Table 1
Test functions used in experiments. d: dimension, C: characteristic, U: unimodal, M: multimodal, S: separable, N: non-separable, Ae: acceptable error.

Test problem	Objective function	Search range	Optimum value	d	C	Ae
Sphere	$f_1(x)=\sum_{i=1}^{d} x_i^2$	$[-5.12\ 5.12]$	$f(\vec{0})=0$	30	US	1.0E−05
Rosenbrock	$f_2(x)=\sum_{i=1}^{d}(100(x_{i+1}-x_i{}^2)^2+(x_i-1)^2)$	$[-30\ 30]$	$f(\vec{1})=0$	30	UN	1.0E−02
Ackley	$f_3(x)=-20+e+\exp\left(-\frac{0.2}{d}\sqrt{\sum_{i=1}^{d} x_i{}^3}\right)$ $-\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)x_i\right)$	$[-1$ $1]$	$f(\vec{0})=0$	30	MN	1.0E−05
Alpine	$f_4(x)=\sum_{i=1}^{d}\lvert x_i\sin x_i+0.1x_i\rvert$	$[-10\ 10]$	$f(\vec{0})=0$	30	MS	1.0E−05
Cosine Mixture	$f_5(x)=\sum_{i=1}^{d} x_i{}^2-0.1(\sum_{i=1}^{d}\cos 5\pi x_i)+0.1d$	$[-1\ 1]$	$f(\vec{0})=-d\times 0.1$	30	MS	1.0E−05
Exponential	$f_6(x)=-(\exp(-0.5\sum_{i=1}^{d} x_i{}^2))+1$	$[-1\ 1]$	$f(\vec{0})=-1$	30	MN	1.0E−05
Zakharov	$f_7(x)=\sum_{i=1}^{d} x_i{}^2+\left(\sum_{i=1}^{d}\frac{ix_i}{2}\right)^2+\left(\sum_{i=1}^{d}\frac{ix_1}{2}\right)^4$	$[-5.12\ 5.12]$	$f(\vec{0})=0$	30	MN	1.0E−02
Salomon Problem	$f_8(x)=1-\cos\left(2\pi\sqrt{\sum_{i=1}^{d} x_i^2}\right)+0.1\left(\sqrt{\sum_{i=1}^{d} x_i^2}\right)$	$[-100\ 100]$	$f(\vec{0})=0$	30	MN	1.0E−01
Inverted cosine wave	$f_9(x)=-\sum_{i=1}^{d-1}\left(\exp\left(\frac{-(x_i^2+x_{i+1}^2+0.5x_ix_{i+1})}{8}\right)\times \mathrm{I}\right)$ where, $\mathrm{I}=\cos\left(4\sqrt{x_i^2+x_{i+1}^2+0.5x_ix_{i+1}}\right)$	$[-5\ 5]$	$f(\vec{0})=-d+1$	10	MN	1.0E−05
Neumaier 3 Problem (NF3)	$f_{10}(x)=\sum_{i=1}^{d}(x_i-1)^2-\sum_{i=2}^{n} x_ix_{i-1}$	$[-d^2\ d^2]$	$f_{\min}=-\frac{(d(d+4)(d-1))}{6}$	10	UN	1.0E−01
Colville	$f_{11}(x)=100[x_2-x_1^2]^2+(1-x_1)^2+90(x_4-x_3^2)^2+(1-x_3)^2+10.1[(x_2-1)^2+(x_4-1)^2]+19.8(x_2-1)(x_4-1)$	$[-10\ 10]$	$f(\vec{1})=0$	4	MN	1.0E−05
Branins's function	$f_{12}(x)=a(x_2-bx_1^2+cx_1-d)^2+e(1-f)\cos x_1+e$	$-5\le x_1\le 10,\ 0\le x_2\le 15$	$f(-\pi, 12.275)=0.3979$	2	MN	1.0E−05
Kowalik	$f_{13}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	$[-5\ 5]$	$f(0.192833, 0.190836, 0.123117, 0.135766)=0.000307486$	4	MN	1.0E−05
Shifted Rosenbrock	$f_{14}(x)=\sum_{i=1}^{d-1}(100(z_i^2-z_{i+1})^2+(z_i-1)^2)+f_{bias}$, $z=x-o+1$, $x=[x_1, x_2, \ldots, x_d]$, $o=[o_1, o_2, \ldots, o_d]$	$[-100\ 100]$	$f(o)=f_{bias}=390$	10	MN	1.0E−01
Shifted Sphere	$f_{15}(x)=\sum_{i=1}^{d} z_i^2+f_{bias}$, $z=x-o$,$x=[x_1, x_2, \ldots, x_d]$, $o=[o_1, o_2, \ldots, o_d]$	$[-100\ 100]$	$f(o)=f_{bias}=-450$	10	US	1.0E−05
Shifted Ackley	$f_{16}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{D} z_i^2}\right)-\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi z_i)\right)+20+e+f_{bias}$, $z=(x-o)$, $x=(x_1, x_2, \ldots, x_d)$, $o=(o_1, o_2, \ldots, o_d)$	$[-32\ 32]$	$f(o)=f_{bias}=-140$	10	MN	1.0E−05
Goldstein-Price	$f_{17}(x)=(1+(x_1+x_2+1)^2\cdot(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2))\cdot(30+(2x_1-3x_2)^2\cdot(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2))$	$[-2\ 2]$	$f(0,-1)=3$	2	MN	1.0E−14
Six-hump camel back	$f_{18}(x)=(4-2.1x_1^2+x_1^4/3)x_1^2+x_1x_2+(-4+4x_2^2)x_2^2$	$[-5\ 5]$	$f(-0.0898, 0.7126)=-1.0316$	2	MN	1.0E−05
Easom's function	$f_{19}(x)=-\cos x_1\cos x_2e^{((-(x_1-\pi)^2-(x_2-\pi)^2))}$	$[-10\ 10]$	$f(\pi,\pi)=-1$	2	UN	1.0E−13
Hosaki Problem	$f_{20}=(1-8x_1+7x_1^2-7/3x_1^3+1/4x_1^4)x_2^2\exp(-x_2)$	$x_1\in[0,5]$, $x_2\in[0,6]$	-2.3458	2	MN	1.0E−6

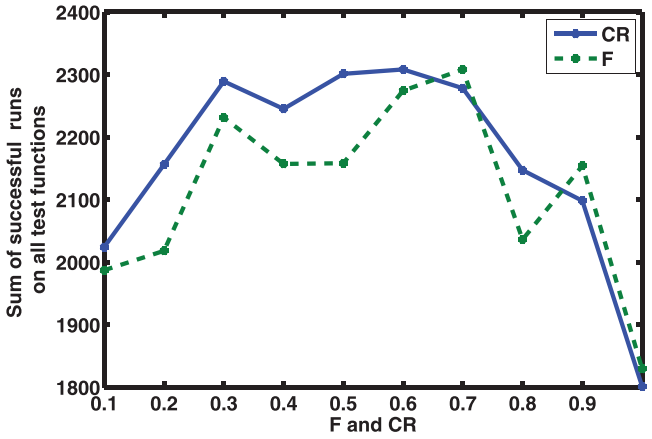


Fig. 2. Effect of parameters *F* and *CR* of DE on success rate.

Table 2
Success rate (SR) for the algorithms of set S_1.

TP	ABC	BSFABC	GABC	MABC	HABCDE
f_1	**100**	**100**	**100**	**100**	**100**
f_2	20	21	24	10	**100**
f_3	**100**	**100**	**100**	**100**	**100**
f_4	**100**	99	99	12	**100**
f_5	**100**	**100**	**100**	**100**	**100**
f_6	**100**	**100**	**100**	**100**	**100**
f_7	0	0	0	0	**100**
f_8	54	66	97	98	**100**
f_9	86	90	99	**100**	97
f_{10}	3	8	21	36	**100**
f_{11}	0	33	30	34	**100**
f_{12}	**100**	91	90	92	**100**
f_{13}	18	51	86	26	**91**
f_{14}	19	22	57	29	**99**
f_{15}	**100**	**100**	**100**	**100**	**100**
f_{16}	**100**	**100**	**100**	**100**	**100**
f_{17}	50	54	51	54	**100**
f_{18}	**100**	47	37	49	**100**
f_{19}	6	**100**	**100**	14	**100**
f_{20}	68	35	37	24	**100**
f_{21}	10	12	18	31	**96**
f_{22}	0	0	0	0	**100**
f_{23}	0	1	22	2	**26**
f_{24}	2	99	63	**100**	**100**

Table 3
Average number of function evolutions (AFE) called by the algorithms of set S_1.

TP	ABC	BSFABC	GABC	MABC	HABCDE
f_1	20,409	30,063	26,735	22,359	**9301**
f_2	186,025	180,325	177,340	188,136	**154,936**
f_3	48,727	42,833	47,957	43,333	**17,243**
f_4	98,099	118,190	59,545	179,705	**23,572**
f_5	23,016	30,029	28,709	22,764	**9175**
f_6	16,974	16,707	22,384	16,648	**6902**
f_7	200,000	200,000	200,000	200,000	**100,890**
f_8	186,384	159,415	122,009	**28,910**	38,009
f_9	88,543	87,966	66,979	63,835	**37,979**
f_{10}	198,915	192,527	186,003	141,729	**24,389**
f_{11}	200,000	152,315	161,103	143,857	**6188**
f_{12}	2003	19,603	21,847	18,969	**917**
f_{13}	182,977	145,154	96,051	173,568	**30,506**
f_{14}	182,883	168,436	120,690	163,511	**43,462**
f_{15}	9128	12,045	9006	8675	**5676**
f_{16}	17,744	31,057	16,654	14,246	**7916**
f_{17}	125,378	96,395	97,945	97,907	**2506**
f_{18}	1014	97,423	126,874	102,629	**660**
f_{19}	192,327	24,354	39,735	172,911	**5118**
f_{20}	76,689	120,126	111,954	142,171	**461**
f_{21}	187,602	177,523	173,445	156,234	**47,961**
f_{22}	200,000	200,000	200,000	200,000	**9205**
f_{23}	200,000	199,693	188,003	194,993	**156,593**
f_{24}	196,985	52,711	133,956	35,375	**3547**

Table 4
Mean error (ME) for the algorithms of set S_1.

TP	ABC	BSFABC	GABC	MABC	HABCDE
f_1	8.17E−06	7.49E−06	8.26E−06	8.95E−06	**7.34E−06**
f_2	1.34E+00	1.68E+00	3.00E+00	3.60E+01	**9.58E−03**
f_3	8.63E−06	8.71E−06	8.74E−06	9.51E−06	**8.35E−06**
f_4	8.10E−06	7.68E−06	1.02E−05	1.01E−03	**7.35E−06**
f_5	7.47E−06	7.05E−06	8.09E−06	9.23E−06	**6.30E−06**
f_6	7.12E−06	7.36E−06	8.15E−06	9.11E−06	**7.02E−06**
f_7	9.75E+01	8.50E+01	1.07E+02	1.47E+00	**8.73E−03**
f_8	9.82E−01	9.79E−01	9.34E−01	9.31E−01	**9.99E−02**
f_9	2.57E−02	6.05E−02	**7.38E−06**	8.45E−06	7.61E−06
f_{10}	8.88E+00	4.39E+00	5.55E+00	2.03E+00	**8.66E−06**
f_{11}	1.64E−01	2.41E−02	2.25E−02	1.14E−02	**6.79E−04**
f_{12}	5.12E−06	5.70E−06	5.68E−06	6.05E−06	**4.90E−06**
f_{13}	1.75E−04	1.39E−04	9.92E−05	1.92E−04	**1.54E−05**
f_{14}	8.03E+00	2.63E+00	3.71E−01	6.71E−01	**6.04E−02**
f_{15}	6.60E−06	7.23E−06	7.04E−06	7.97E−06	**6.38E−06**
f_{16}	**7.76E−06**	7.87E−06	8.26E−06	8.93E−06	8.60E−06
f_{17}	2.60E−07	4.67E−07	6.06E−07	5.19E−07	**1.09E−14**
f_{18}	**1.28E−05**	1.75E−04	2.01E−04	1.84E−04	1.34E−05
f_{19}	4.89E−05	4.97E−14	5.27E−14	8.04E−06	**4.44E−14**
f_{20}	5.65E−04	9.94E−03	1.08E−03	2.00E−03	**4.91E−06**
f_{21}	1.36E−02	2.92E−02	1.51E−02	3.87E−03	**5.55E−04**
f_{22}	1.65E+01	2.30E+01	5.40E+00	1.61E+01	**1.79E−05**
f_{23}	9.98E+00	9.76E+00	5.14E+00	9.65E+00	**4.47E+00**
f_{24}	2.47E+00	9.52E−02	1.04E−01	9.50E−02	**9.35E−02**

Table 5
Success rate (SR) for the algorithms of set S_2.

TP	CMA-ES	DE	PSO	BBO	SMO	HABCDE
f_1	**100**	**100**	**100**	**100**	**100**	**100**
f_2	0	0	15	0	0	**100**
f_3	**100**	**100**	99	**100**	99	**100**
f_4	98	**100**	**100**	**100**	97	**100**
f_5	**100**	96	0	0	87	**100**
f_6	**100**	**100**	**100**	0	**100**	**100**
f_7	**100**	**100**	**100**	48	**100**	**100**
f_8	**100**	99	**100**	12	15	**100**
f_9	0	17	0	0	**99**	97
f_{10}	**100**	**100**	**100**	30	91	**100**
f_{11}	**100**	91	**100**	96	**100**	**100**
f_{12}	**100**	**100**	80	0	86	**100**
f_{13}	88	70	82	21	**96**	91
f_{14}	**99**	3	83	19	47	**99**
f_{15}	**100**	**100**	**100**	74	**100**	**100**
f_{16}	**100**	**100**	**100**	39	**100**	**100**
f_{17}	89	**100**	**100**	0	**100**	**100**
f_{18}	**100**	55	44	88	45	**100**
f_{19}	**100**	**100**	**100**	7	**100**	**100**
f_{20}	98	86	94	27	9	**100**

Table 6
Average number of function evolutions (AFE) called by the algorithms of set S_2.

TP	CMA-ES	DE	PSO	BBO	SMO	HABCDE
f_1	10,873	22,444	13,977	10,119	12,694	**9301**
f_2	200,000	200,000	194,000	200,000	200,000	**154,936**
f_3	24,486	42,699	47,663	23,150	29,744	**17,243**
f_4	61,987	60,983	42,537	**14,293**	80,728	23,572
f_5	14,606	30,339	200,000	200,000	47,354	**9174**
f_6	16,004	17,018	10,169	200,000	9696	**6902**
f_7	**48,486**	68,154	48,528	106,436	133,128	100,890
f_8	45,564	58,843	**12,514**	186,636	191,535	38,009
f_9	200,000	176,111	200,000	200,000	79,381	**37,979**
f_{10}	**11,903**	17,251	11,905	180,790	168,642	24,389
f_{11}	6286	22,950	15,985	21,348	52,228	**6188**
f_{12}	1012	1791	41,554	200,000	30,221	**917**
f_{13}	**13,434**	63,953	49,678	127,142	44,133	30,506
f_{14}	**32,584**	196,183	93,129	187,111	156,484	43,462
f_{15}	9362	10,325	8053	31,015	5898	**5676**
f_{16}	17,365	15,537	12,337	89,123	9069	**7916**
f_{17}	4052	3829	8844	200,000	3402	**2506**
f_{18}	1214	90,867	112,880	17,752	114,990	**660**
f_{19}	**2385**	4837	8881	192,714	11,790	5118
f_{20}	497	28,804	13,267	120,001	189,691	**461**

Table 7
Mean error (ME) for the algorithms of set S_2.

TP	CMA-ES	DE	PSO	BBO	SMO	HABCDE
f_1	8.59E−06	9.06E−06	9.11E−06	**5.18E−06**	8.89E−06	7.34E−06
f_2	1.24E+01	4.24E+01	9.30E+00	4.45E+01	3.46E+01	**9.58E−03**
f_3	9.29E−06	9.46E−06	2.50E−02	9.31E−06	9.32E−03	**8.35E−06**
f_4	9.47E−06	9.43E−06	7.38E−06	8.91E−06	1.08E−05	**7.35E−06**
f_5	7.25E−06	5.92E−03	6.46E−01	2.84E−04	2.07E−02	**6.30E−06**
f_6	8.61E−06	8.99E−06	9.12E−06	1.00E+00	8.97E−06	**7.02E−06**
f_7	**7.48E−03**	9.47E−03	9.48E−03	5.19E−01	9.12E−03	8.73E−03
f_8	1.91E−01	2.01E−01	9.28E−01	1.06E+00	1.85E+00	**9.99E−02**
f_9	9.12E−01	8.93E−01	2.21E+00	7.98E−02	8.95E−06	**7.61E−06**
f_{10}	**7.08E−06**	8.25E−06	9.20E−06	6.89E−04	1.19E−05	8.66E−06
f_{11}	**5.68E−04**	4.61E−02	9.64E−03	3.98E−03	7.71E−04	**6.79E−04**
f_{12}	3.26E−05	5.11E−06	6.62E−06	3.98E−01	5.84E−06	**4.90E−06**
f_{13}	2.35E−05	2.81E−04	1.01E−04	9.72E−04	**1.51E−05**	1.54E−05
f_{14}	**2.48E−03**	1.11E+02	7.59E−01	4.14E+02	1.30E+00	6.04E−02
f_{15}	6.69E−06	7.67E−06	7.96E−06	9.70E−06	7.65E−06	**6.38E−06**
f_{16}	**5.38E−06**	9.03E−06	8.85E−06	1.30E−01	8.66E−06	8.60E−06
f_{17}	1.44E−12	**4.20E−15**	5.13E−15	3.00E+00	**4.20E−15**	1.09E−14
f_{18}	1.37E−05	1.65E−05	1.79E−05	2.04E−05	1.76E−05	**1.34E−05**
f_{19}	**2.72E−14**	4.28E−14	4.96E−14	7.00E−02	4.71E−14	4.44E−14
f_{20}	5.42E−06	5.60E−06	4.96E−06	2.35E−04	1.06E−05	**4.91E−06**

Table 8
Success rate (SR) for the algorithms of set S_3.

TP	ABC–DE	DEBCO	HDABCA	HABCDE
f_1	**100**	**100**	**100**	**100**
f_2	0	1	0	**100**
f_3	**100**	**100**	**100**	**100**
f_4	**100**	77	35	**100**
f_5	**100**	**100**	**100**	**100**
f_6	**100**	**100**	**100**	**100**
f_7	**100**	0	0	**100**
f_8	**100**	**100**	**100**	**100**
f_9	**100**	**100**	98	97
f_{10}	86	16	**100**	**100**
f_{11}	37	7	92	**100**
f_{12}	**100**	94	80	**100**
f_{13}	1	10	4	**91**
f_{14}	20	48	21	**99**
f_{15}	**100**	**100**	**100**	**100**
f_{16}	**100**	**100**	**100**	**100**
f_{17}	**100**	1	57	**100**
f_{18}	**100**	55	55	**100**
f_{19}	91	0	6	**100**
f_{20}	**100**	85	95	**100**

Table 9
Average number of function evolutions (AFE) called by the algorithms of set S_3.

TP	ABC–DE	DEBCO	HDABCA	HABCDE
f_1	10,119	20,269	26,376	**9301**
f_2	200,000	19,9806	200,000	**154,936**
f_3	18,672	40,210	51,250	**17,243**
f_4	34,569	178,306	192,466	**23,572**
f_5	9584	20,443	27,137	**9175**
f_6	7671	15,213	19,743	**6902**
f_7	148,546	200,000	200,000	**100,890**
f_8	**11,707**	31,473	30,285	38,009
f_9	**27,047**	71,622	105,196	37,979
f_{10}	151,418	197,224	83,330	**24,389**
f_{11}	147,268	193,804	113,683	**6188**
f_{12}	6511	30,170	41,794	**917**
f_{13}	198,492	187,817	195,783	**30,506**
f_{14}	180,001	154,473	181,881	**43,462**
f_{15}	6849	10,402	8350	**5676**
f_{16}	10,118	17,426	13,475	**7916**
f_{17}	17,590	199,353	109,085	**2506**
f_{18}	2701	91,578	90,704	**660**
f_{19}	75,818	200,000	194,872	**5118**
f_{20}	1387	31,472	10,765	**461**

Table 10
Mean error (ME) for the algorithms of set S_3.

TP	ABC–DE	DEBCO	HDABCA	HABCDE
f_1	9.00E−06	9.23E−06	9.05E−06	**7.34E−06**
f_2	2.67E+01	2.32E+01	4.65E+01	**9.58E−03**
f_3	9.48E−06	9.60E−06	9.46E−06	**8.35E−06**
f_4	9.43E−06	1.38E−05	5.93E−04	**7.35E−06**
f_5	8.90E−06	9.22E−06	8.94E−06	**6.30E−06**
f_6	9.08E−06	9.28E−06	8.97E−06	**7.02E−06**
f_7	9.18E−03	6.55E−02	4.59E+00	**8.73E−03**
f_8	9.20E−01	9.30E−01	9.26E−01	**9.99E−02**
f_9	7.85E−06	7.84E−06	6.00E−02	**7.61E−06**
f_{10}	4.62E−01	1.30E+00	9.80E−02	**8.66E−06**
f_{11}	2.63E−02	5.43E−02	8.86E−03	**6.79E−04**
f_{12}	5.67E−06	5.62E−06	6.73E−06	**4.90E−06**
f_{13}	3.58E−04	2.09E−04	2.73E−04	**1.54E−05**
f_{14}	5.19E+00	3.07E−01	1.23E+00	**6.04E−02**
f_{15}	7.80E−06	7.73E−06	7.85E−06	**6.38E−06**
f_{16}	8.91E−06	9.02E−06	9.08E−06	**8.60E−06**
f_{17}	1.84E−14	7.36E−05	5.34E−14	**1.09E−14**
f_{18}	**1.12E−05**	1.58E−05	1.49E−05	1.34E−05
f_{19}	4.39E−12	4.82E−02	3.75E−04	**4.44E−14**
f_{20}	5.54E−06	5.83E−06	5.51E−06	**4.91E−06**

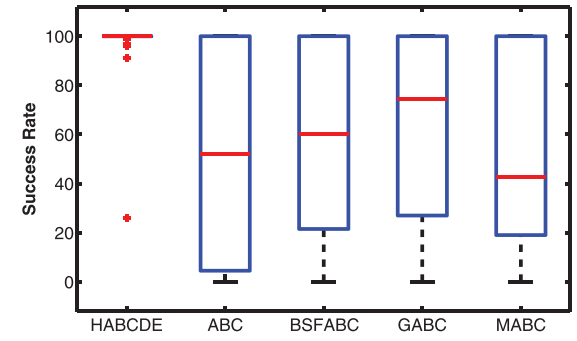
achieved by the algorithms. Here *SR* is the count of successful runs i.e., how many times the algorithm succeed to find function optima with acceptable error in 100 trials. *AFE* is the average count of function calls to reach at the termination criteria in 100 trials and *ME* is the mean error in solution in 100 trials. It can be observed from Tables 2–8 that reliability for proposed HABCDE algorithm is higher as it has achieved higher or equal success rate than on all functions except f_9 and f_{13}. Here, it is to be noticed that the algorithm CMA-ES gave tough competition to proposed HABCDE algorithm as it achieved equivalent highest success rate on 14 out of 20 considered benchmark test functions. It can be verified from Tables 3–9 that HABCDE took lesser function calls on 96%, 65% and 90% test problems than that of algorithms of set S_1, S_2 and S_3 respectively, so it is considered more efficient. The algorithm CMA-ES placed on second position with respect to AFE and took less AFEs than the proposed and other considered algorithms on 25% test functions. Tables 4–10 shows the clear superiority of HABCDE over other algorithms of sets S_1, S_2 and S_3 in terms of accuracy as mean error obtained by HABCDE's is lesser than that of algorithms of set S_1 on 21 out of 24 functions, lesser than algorithms of set S_2 on 12 out of 20 test problems and lesser than algorithms of set S_3 on 19 out of 20 test problems. Here, also the CMA-ES algorithm is promising as it got less mean error than the proposed and other algorithms in set S_2 on 6 out of 20 test functions. Further, we compared HABCDE and other algorithms in the order: SR–AFE–ME i.e., first compare the SR and if the two algorithms are achieving the same SR then compare the AFE. Still, if there is a tie then ME is used for comparison. As a result available in Tables 2–10, HABCDE performs better than all algorithms of set S_1 on all test problems except f_9 and it costs less than the algorithms of set S_3 on all test functions except f_8 and f_9. The algorithms of set S_2, particularly CMA-ES which costs less on 20% test functions, gave good competition to proposed HABCDE. It can also be observed from results that no algorithm of set S_1 and S_3 has beaten HABCDE on any real-world optimization problem. As these functions are of different complexities, it can be stated that HABCDE is able to balance the exploration and exploitation capabilities efficiently. MABC algorithm performed well on single test function f_9. Each of the hybrid algorithms (ABC–DE, DEBCO and HDABCA) is cost effective than HABCDE only on two functions f_8 and f_9. Among the state-of-the-art algorithms, BBO and DE performed better than HABCDE on single function f_4 and f_7 respectively, PSO and SMO on two functions (f_8, f_9) and (f_9, f_{13}) respectively. The CMA-ES algorithm performed better than HABCDE on four test functions (f_7, f_{10}, f_{14}) and (f_{19}). This should be noticed that HABCDE performed

worst than all modified, state-of-the-art algorithms and hybrid ABC algorithms except ABC, DE, BBO and BSFABC on multimodel non-separable test function f_9. On the other hand, popular modified ABC strategy BSFABC has not defeated HABCDE in any sense. So, if we conclude the results of all considered test functions, the HABCDE algorithm is the cost effective algorithm on most of the considered test functions but not for all considered functions. This behavior is not an exception but actually verifies the statement of NFL. Here, the bold values in Tables 2–10 reflect the best value among the values achieved by all algorithms for that particular function.

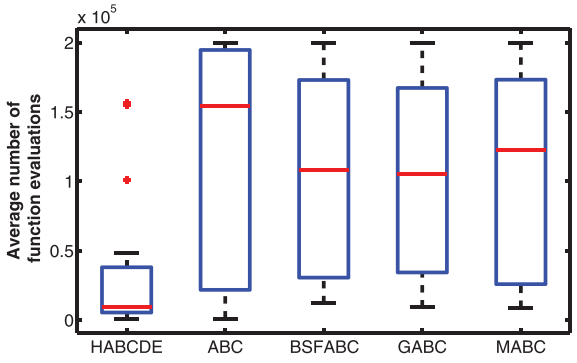
The proposed algorithm HABCDE is analyzed more intensively in next subsection through some statistical test like Mann–Whitney U rank sum test, boxplot etc.,

5.4. Statistical analysis

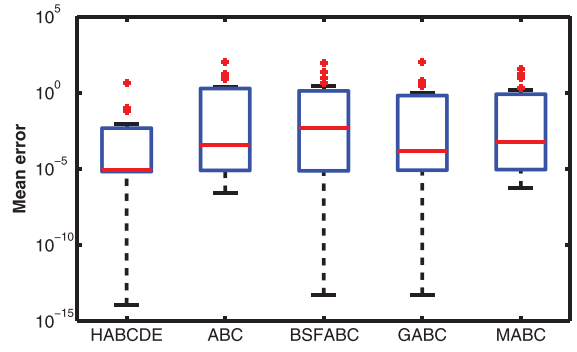
Since the empirical distribution of results can efficiently be represented by boxplot [45], the boxplots for SR, AFE and ME for HABCDE and all other considered algorithms of sets S_1, S_2 and S_3, are represented in Figs. 3–5 respectively. Figs. 3–5 show that HABCDE is better in terms of all SR, AFE and ME evaluations as in AFE and ME cases, the interquartile range and median in each figure are very low and in SR case the median is very high for HABCDE. These figures reflect that the algorithm CMA-ES is also promising in terms SR, AFE and ME. It can be observed from Figs. 3(a), 4(a) and 5(a) for SR the boxes for proposed HABCDE are not being seen in almost all cases because of the negligible variation.



(a) SR

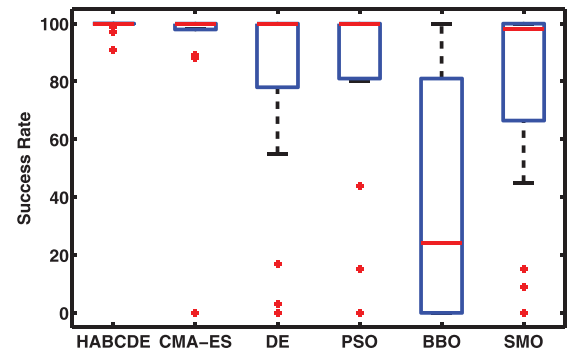


(b) AFE

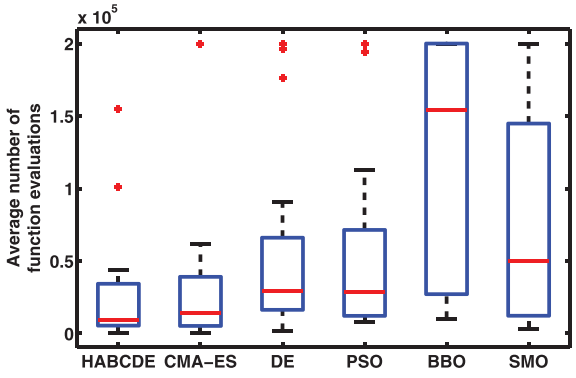


(c) ME

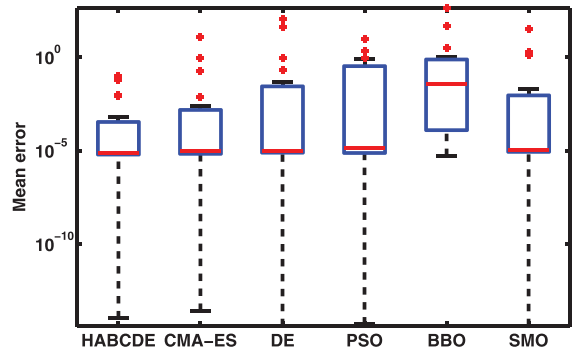
Fig. 3. Boxplots for algorithms of set S_1 on test and real-world problems f_1–f_{24}.



(a) SR



(b) AFE



(c) ME

Fig. 4. Boxplots for algorithms of set S_2 on test problems f_1–f_{20}.

Now, the Mann–Whitney U rank sum test [32] is applied on AFE to check the significant difference between outputs of considered algorithms. We applied this test at 5% level of significance. Since AFE called by the algorithms are not normally distributed, the non-parametric Mann–Whitney test is used. The results output of this test for the AFEs of 100 runs is presented in Tables 11–13 for sets S_1, S_2 and S_3, respectively. If the test observes the significant difference then the signs '+' and '−' are used as output for the cases where HABCDE takes less or more AFEs than the other algorithms,

Table 11
Performance comparison of S_1 algorithms based on the Mann–Whitney U rank sum test at a $\alpha = 0.05$ significance level and average number of function evaluations, TP: test problem.

TP	Mann–Whitney U rank sum test with HABCDE				TP	Mann–Whitney U rank sum test with HABCDE			
	ABC	BSFABC	GABC	MABC		ABC	BSFABC	GABC	MABC
f_1	+	+	+	+	f_{13}	+	+	+	+
f_2	+	+	+	+	f_{14}	+	+	+	+
f_3	+	+	+	+	f_{15}	+	+	+	+
f_4	+	+	+	+	f_{16}	+	+	+	+
f_5	+	+	+	+	f_{17}	+	+	+	+
f_6	+	+	+	+	f_{18}	=	+	+	+
f_7	+	+	+	+	f_{19}	+	+	+	+
f_8	+	+	+	−	f_{20}	+	+	+	+
f_9	+	+	+	+	f_{21}	+	+	+	+
f_{10}	+	+	+	+	f_{22}	+	+	+	+
f_{11}	+	+	+	+	f_{23}	+	+	+	+
f_{12}	+	+	+	+	f_{24}	+	+	+	+

Table 12
Comparison of S_2 algorithms based on AFEs and the Mann–Whitney test, TP: test problem.

TP	Mann–Whitney U rank sum test with HABCDE					TP	Mann–Whitney U rank sum test with HABCDE				
	CMA-ES	DE	PSO	BBO	SMO		CMA-ES	DE	PSO	BBO	SMO
f_1	=	+	+	=	+	f_{11}	=	+	+	+	+
f_2	+	+	+	+	+	f_{12}	=	=	+	+	+
f_3	+	+	+	+	+	f_{13}	−	+	+	+	+
f_4	+	+	+	−	+	f_{14}	−	+	+	+	+
f_5	+	+	+	+	+	f_{15}	+	+	+	+	=
f_6	+	+	+	+	+	f_{16}	+	=	+	+	=
f_7	−	−	−	+	+	f_{17}	+	+	+	+	=
f_8	+	+	−	+	+	f_{18}	=	+	+	+	+
f_9	+	+	+	+	+	f_{19}	−	=	+	+	+
f_{10}	−	−	−	+	+	f_{20}	=	+	+	+	+

Table 13
Performance comparison of S_3 algorithms based on AFEs and the Mann–Whitney test, TP: test problem.

TP	Mann–Whitney U rank sum test with HABCDE			TP	Mann–Whitney U rank sum test with HABCDE		
	ABC–DE	DEBCO	HDABCA		ABC–DE	DEBCO	HDABCA
f_1	=	+	+	f_{11}	+	+	+
f_2	+	+	+	f_{12}	+	+	+
f_3	+	+	+	f_{13}	+	+	+
f_4	+	+	+	f_{14}	+	+	+
f_5	=	+	+	f_{15}	+	+	+
f_6	+	+	+	f_{16}	+	+	+
f_7	+	+	+	f_{17}	+	+	+
f_8	−	−	−	f_{18}	+	+	+
f_9	−	+	+	f_{19}	+	+	+
f_{10}	+	+	+	f_{20}	+	+	+

respectively. If considered test does not observe any significant difference then output is '=' sign. Since Table 11 for set S_1 contains **94 '+' signs out of 96 comparisons**, Table 12 for set S_2 contains **77 '+' signs out of 100 comparisons** and Table 13 for set S_3 contains **54 '+' signs out of 60 comparisons**, it can be stated that HABCDE performance is significantly better than other considered algorithms over the set of test problems of Table 1 and considered 4 real-world optimization problems.

Further, performance indices (PIs) [9] are calculated to compare the considered algorithms by giving weighted importance to SR, AFE and ME. The values of *PI* for the HABCDE and other considered algorithms, are calculated as:

$$PI = \frac{1}{N_p}\sum_{i=1}^{N_p}(k_1\alpha_1^i + k_2\alpha_2^i + k_3\alpha_3^i)$$

where $\alpha_1^i = \frac{Sr^i}{Tr^i}$; $\alpha_2^i = \begin{cases} \frac{Mf^i}{Af^i}, & \text{if } Sr^i > 0. \\ 0, & \text{if } Sr^i = 0. \end{cases}$; and $\alpha_3^i = \frac{Mo^i}{Ao^i}$

$i = 1, 2, \ldots, N_p$

Here, the symbols have their usual meanings and more details about these can be found in article [20]. Actually, there are three cases. In one case, weight of the single variable out of SR, AFE and ME, varies from 0 to 1 and remaining weight is equally distributed to other two variables as suggested in [9]. The *PI* graphs corresponding to these three cases for the algorithms of sets S_1, S_2 and S_3 are shown in Figs. 6(a)–(c), 7(a)–(c) and 8(a)–(c) respectively. In PI Figs. 6–8, horizontal axis represents the weights k_1, k_2 and k_3 and *PI* is represented by vertical axis. The weight to SR, AFE and ME varies in the range (0, 1) in case (1), case (2) and case (3) respectively. It can be observed from Figs. 6–8 that *PI* of HABCDE is much

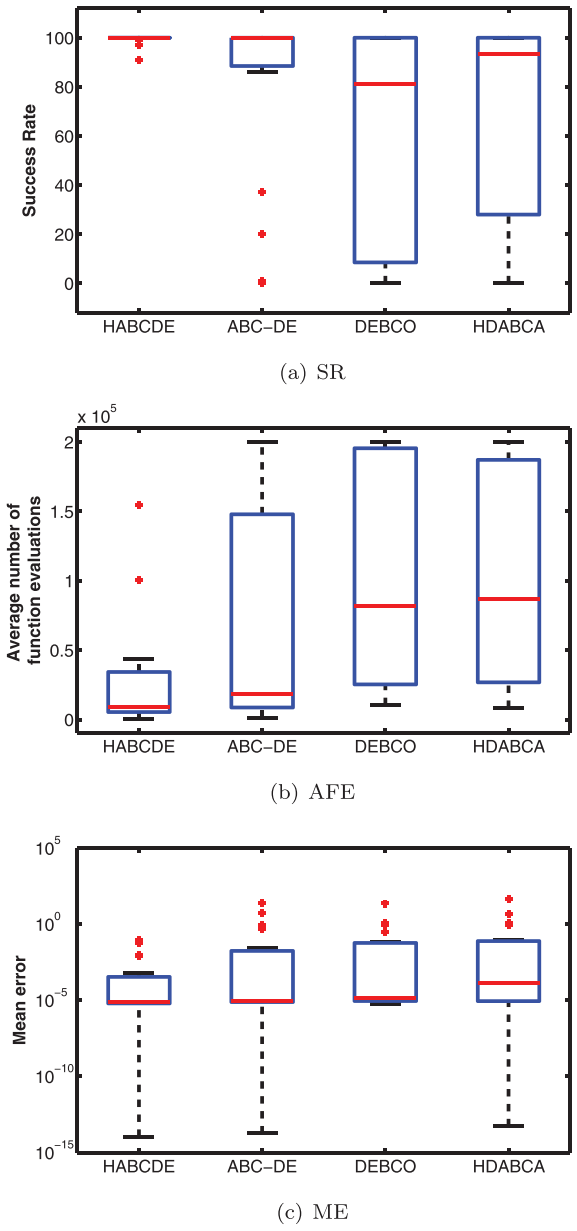


Fig. 5. Boxplots for algorithms of set S_3 on test problems f_1–f_{20}.

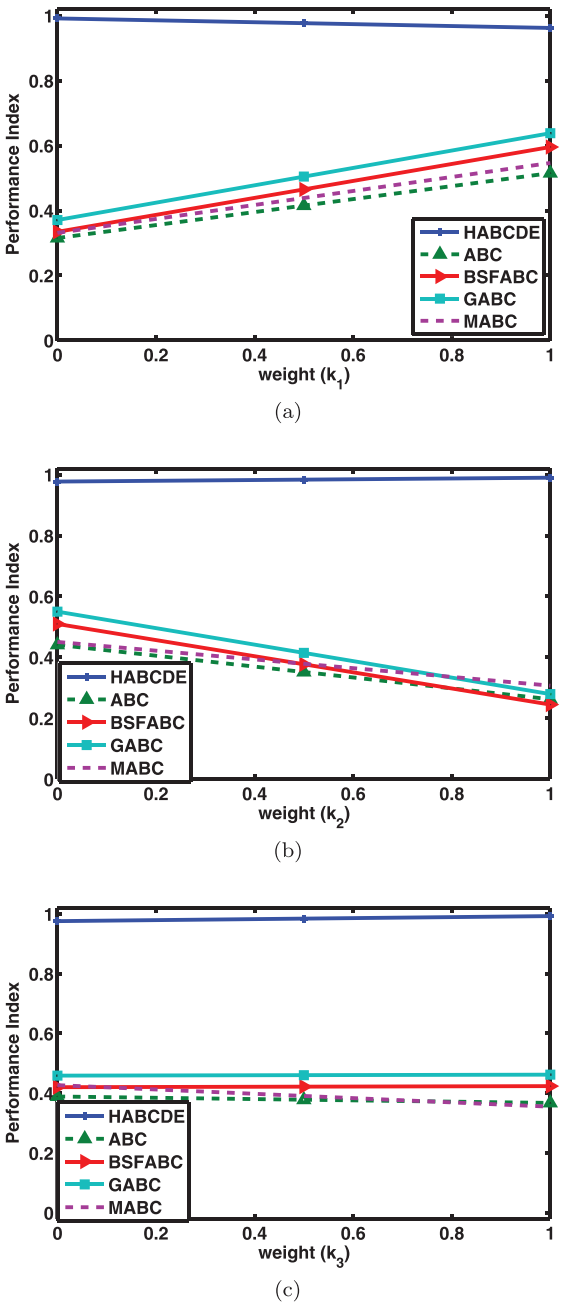
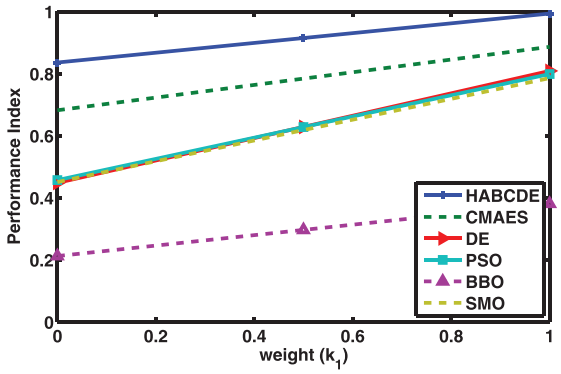


Fig. 6. Performance indices for algorithms of set S_1 on test and real-world problems; (a) for case (1), (b) for case (2) and (c) for case (3).

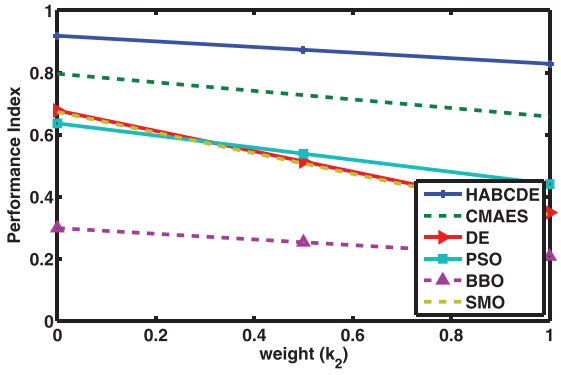
higher than all the considered algorithms of sets S_1, S_2 and S_3 for all three cases. Here, it can be seen that PI for the algorithm CMA-ES is slightly less than the proposed HABCDE, but better than the other considered state-of-the-art optimization algorithms.

The convergence speed comparison of proposed HABCDE and other considered algorithm can be judged through convergence Figs. 9(a)–(f). In these figures, one can seen that proposed HABCDE algorithm is moving faster to minima through the iterations in a single run for selective functions f_3, f_4, f_5, f_6, f_7 and f_8, respectively. It can be observed easily from Fig. 9(a) that as compare to other algorithms, HABCDE converges fast because of modified onlooker bee phase and employed bee phase. After about 60 iterations, the rate of convergence of HABCDE is highest. The same behavior can be observed in other convergence graphs. DE algorithm is fast in convergence and ABC is better in exploration. Thus the hybrid variant of ABC and DE is expected to converge towards optimal solution quickly while maintaining the diversified search.

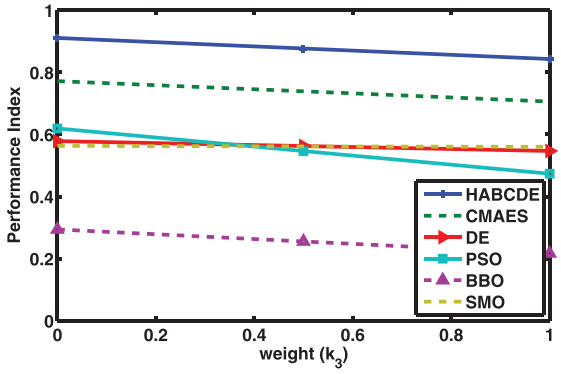
The motivation of this paper was to develop an algorithm which can come out of local optima in the case of premature convergence while maintaining the convergence speed. With these experiments and associated statistical analyzes, it can be established that however the proposed algorithm HABCDE is not superior to all other considered algorithms over all problems under consideration, it fulfills our requirements and is one of the most performing algorithms. The outcome of the experiments coincides with the statement of NFL which states that "regardless of the performance measure, the performance of all optimization algorithms averaged uniformly over any finite set F of functions is equal if and only if F is closed under permutation". Overall, we can say that HABCDE is relatively a better choice for multimodal and nonseparable problems.



(a)

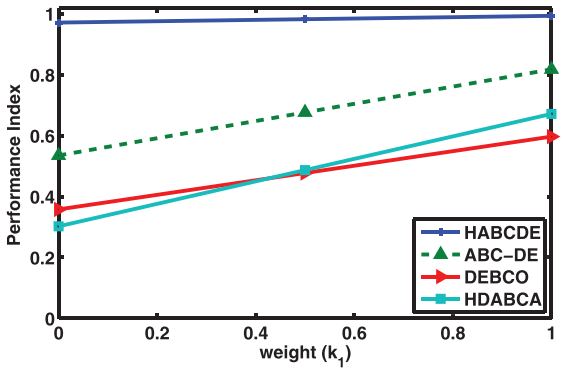


(b)

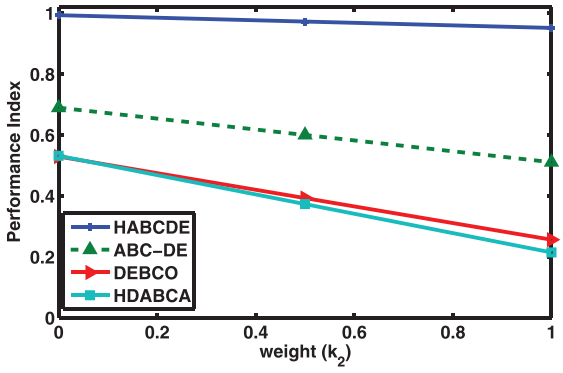


(c)

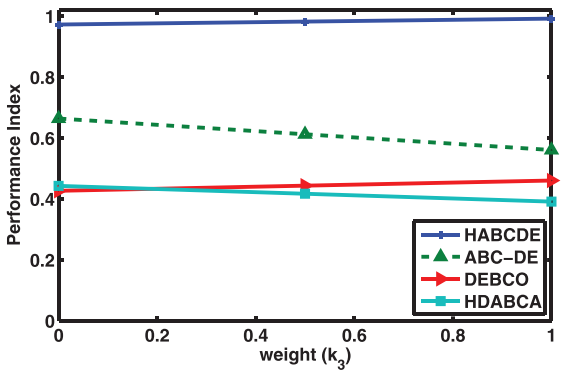
Fig. 7. Performance indices for algorithms of set S_2 on test problems; (a) for case (1), (b) for case (2) and (c) for case (3).



(a)



(b)



(c)

Fig. 8. Performance indices for algorithms of set S_3 on test problems; (a) for case (1), (b) for case (2) and (c) for case (3).

6. Conclusion

Artificial Bee Colony algorithm is simple swarm intelligence based algorithm with few parameters but with drawbacks, like slow convergence and poor balance between exploration and exploitation. Differential Evolution, on the other hand, exhibits relatively faster convergence. Therefore, in this paper, ABC and DE have hybridized and a new hybrid algorithm, HABCDE is proposed. In this HABCDE, all three phases of ABC have been modified. Employed bee phase is modified by incorporating the Gbest concept into it for better exploitation; Onlooker bee phase is improved by including evolutionary operators (mutation, crossover, and selection) of basic DE process for faster convergence. Further, the number of scout bees in Scout bee phase is increased to achieve better exploration. The proposed algorithm has been compared with other hybrid algorithms of ABC and DE namely, ABC–DE, DEBCO, and HDABCA. HABCDE has also been compared with recent variants of ABC namely, BSFABC, GABC and MABC and state-of-the-art algorithms namely, CMA-ES, DE, PSO, BBO, SMO, ABC on benchmark and real-world optimization problems. Through the extensive statistical analyzes, it can be stated that the proposed algorithm HABCDE is an excellent choice from reliability, efficiency and accuracy point of views.

One can extend the proposed work in several directions. It would be interesting to hybridize the basic ABC algorithm with another state-of-the-art algorithms, like CMA-ES, SMO, PSO and other improved versions of DE. Combinatorial and multi-objective optimization problems may also be handled by modifying proposed HABCDE. A new phase in ABC may be introduced using operators of DE. Other variations of crossover, mutation and selection may also

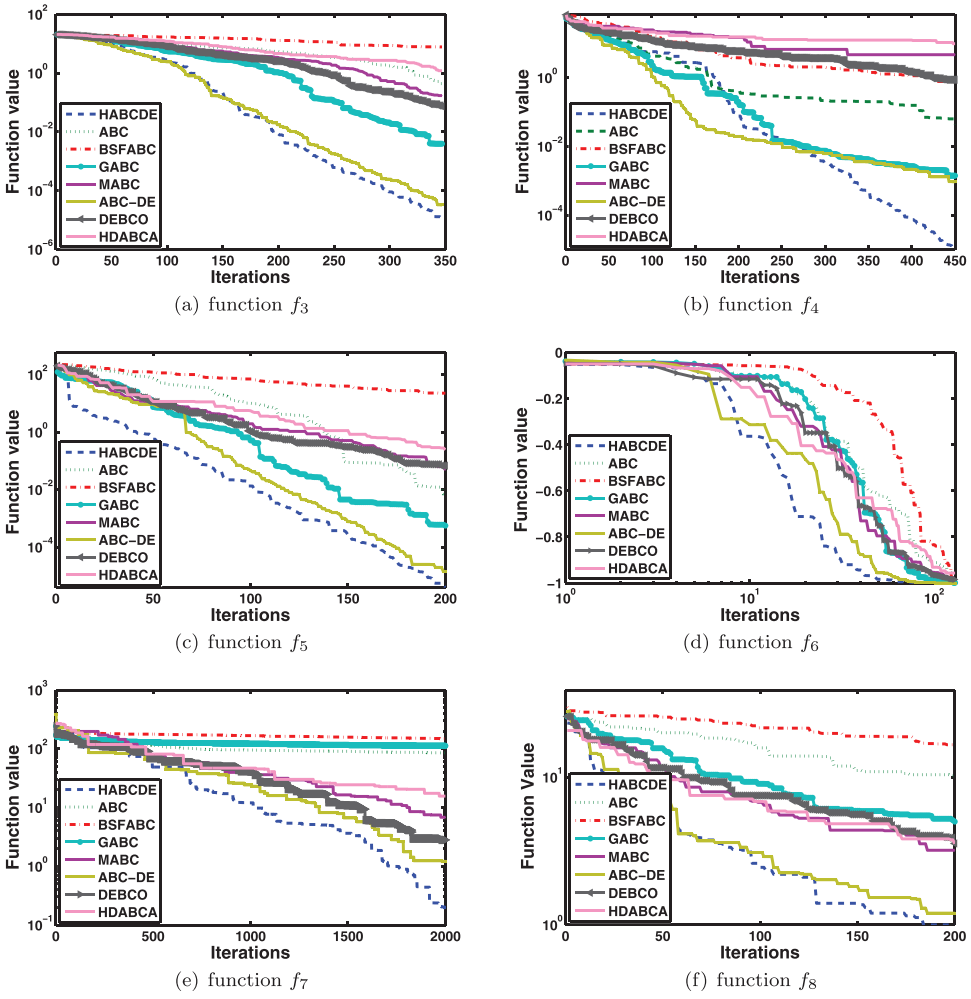


Fig. 9. Convergence graph for all algorithms of sets S_1 and S_3 on test problems f_3–f_8.

be investigated in the proposed hybridization. The ability of the proposed algorithm can also be tested on higher dimensional problems like problems in IEEE competition for 300 and 1000 dimensions.

References

[1] A. Abdullah, S. Deris, M.S. Mohamad, A new hybrid bee evolution algorithm for parameter estimation in biological model. ICIC express letters. Part B, Appl. Int. J. Res. Surv. 4 (1) (2013) 1–6.

[2] A. Abraham, R.K. Jatoth, A. Rajasekhar, Hybrid Differential Artificial Bee Colony algorithm, J. Comput. Theor. Nanosci. 9 (2) (2012) 249–257.

[3] A. Abraham, R.K. Jatoth, A. Rajasekhar, Hybrid Differential Artificial Bee Colony algorithm, J. Comput. Theor. Nanosci. 9 (2) (2012) 249–257.

[4] B. Akay, D. Karaboga, A modified Artificial Bee Colony algorithm for real-parameter optimization, Inf. Sci. (2010), http://dx.doi.org/10.1016/j.ins.2010.07.015.

[5] M.M. Ali, C. Khompatraporn, Z.B. Zabinsky, A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems, J. Global Optim. 31 (4) (2005) 635–672.

[6] A. Alizadegan, M.R. Meybodi, B. Asady, A novel hybrid Artificial Bee Colony algorithm and Differential Evolution for unconstrained optimization problems, Adv. Comput. Sci. Eng. 8 (1) (2012).

[7] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, The best-so-far selection in Artificial Bee Colony algorithm, Appl. Soft Comput. 11 (2) (2011) 2888–2901.

[8] J.C. Bansal, H. Sharma, S.S. Jadon, M. Clerc, Spider Monkey Optimization algorithm for numerical optimization, Memet. Comput. (2013) 1–17.

[9] J.C. Bansal, H. Sharma, Cognitive learning in Differential Evolution and its application to model order reduction problem for single-input single-output systems, Memet. Comput. (2012) 1–21.

[10] S. Das, P.N. Suganthan, Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems. Tech. Rep., Jadavpur University; Nangyang Technological University, Kolkata, India; Singapore, 2010.

[11] K. Diwold, A. Aderhold, A. Scheidler, M. Middendorf, Performance evaluation of Artificial Bee Colony optimization and new selection schemes, Memet. Comput. (2011) 1–14.

[12] M. Dorigo, G. Di Caro, Ant colony optimization: a new meta-heuristic, in: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 2, IEEE, 1999.

[13] H.-B. Duan, C.-F. Xu, Z.-H. Xing, A hybrid Artificial Bee Colony optimization and quantum evolutionary algorithm for continuous optimization problems, Int. J. Neural Syst. 20 (01) (2010) 39–50.

[14] M. El-Abd, Performance assessment of foraging algorithms vs. evolutionary algorithms, Inf. Sci. 182 (1) (2011) 243–263.

[15] A.P. Engelbrecht, Computational Intelligence: An Introduction, Wiley, 2007.

[16] W. Gao, S. Liu, A Modified Artificial Bee Colony Algorithm, Computers & Operations Research, 2011.

[17] N. Hansen, The CMA evolution strategy: a comparing review, in: Towards a New Evolutionary Computation, Springer, 2006, pp. 75–102.

[18] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, in: Evolutionary Computation, 1996., Proceedings of IEEE International Conference on, IEEE, 1996, pp. 312–317.

[19] R. Hooke, T.A. Jeeves, "Direct search" solution of numerical and statistical problems, J. ACM (JACM) 8 (2) (1961) 212–229.

[20] S.S. Jadon, J.C. Bansal, R. Tiwari, H. Sharma, Artificial Bee Colony algorithm with global and local neighborhoods, Int. J. Syst. Assur. Eng. Manag. (2014) 1–13.

[21] S.S. Jadon, J.C. Bansal, R. Tiwari, H. Sharma, Expedited Artificial Bee Colony algorithm, in: Proceedings of the Third International Conference on Soft Computing for Problem Solving, Springer, 2014, pp. 787–800.

[22] F. Kang, J. Li, Z. Ma, H. Li, Artificial Bee Colony algorithm with local search for numerical optimization, J. Softw. 6 (3) (2011) 490–497.

[23] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization. Techn. Rep. TR06, Erciyes Univ. Press, Erciyes, 2005.

[24] D. Karaboga, B. Akay, A comparative study of Artificial Bee Colony algorithm, Appl. Math. Comput. 214 (1) (2009) 108–132.

[25] D. Karaboga, B. Basturk, On the performance of Artificial Bee Colony (ABC) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697.

[26] D. Karaboga, B. Akay, A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems, Appl. Soft Comput. 11 (3) (2011) 3021–3031.
[27] J. Kennedy, R. Eberhart, Particle Swarm Optimization, in: Neural Networks, 1995. Proceedings., IEEE International Conference on, volume 4, IEEE, 1995, pp. 1942–1948.
[28] M.S. Kıran, M. Gündüz, A novel Artificial Bee Colony-based algorithm for solving the numerical optimization problems, Int. J. Innov. Comput. Inf. Control 8 (9) (2012) 6107–6121.
[29] Y. Li, Y. Wang, B. Li, A hybrid Artificial Bee Colony assisted Differential Evolution algorithm for optimal reactive power flow, Int. J. Electr. Power Energy Syst. 52 (2013) 25–33.
[30] S. Łukasik, S. Żak, Firefly algorithm for continuous constrained optimization tasks, in: Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems, Springer, 2009, pp. 97–106.
[31] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2) (2007) 1567–1579.
[32] H.B. Mann, D.R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, Ann. Math. Stat. 18 (1) (1947) 50–60.
[33] G.C. Onwubolu, B.V. Babu, New Optimization Techniques in Engineering, Springer Verlag, 2004.
[34] C. Ozturk, D. Karaboga, Hybrid Artificial Bee Colony algorithm for neural network training, in: Evolutionary Computation (CEC), 2011 IEEE Congress on, IEEE, 2011, pp. 84–88.
[35] K.M. Passino, Biomimicry of bacterial foraging for distributed optimization and control, Control Syst. Mag. IEEE 22 (3) (2002) 52–67.
[36] K.V. Price, Differential Evolution: a fast and simple numerical optimizer, in: Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American, IEEE, 1996, pp. 524–527.
[37] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer Verlag, 2005.
[38] K.M. Ragsdell, D.T. Phillips, Optimal design of a class of welded structures using geometric programming, ASME J. Eng. Ind. 98 (3) (1976) 1021–1025.
[39] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des. 112 (1990) 223.
[40] S. Dan, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (6) (2008) 702–713.
[41] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, CEC 2005 (2005).
[42] A. Thammano, A. Phu-ang, A hybrid Artificial Bee Colony algorithm with local search for flexible job-shop scheduling problem, Proc. Comput. Sci. 20 (2013) 96–101.
[43] J. Vesterstrom, R. Thomsen, A comparative study of Differential Evolution, Particle Swarm Optimization, and evolutionary algorithms on numerical benchmark problems, in: Evolutionary Computation, 2004. CEC2004. Congress on, volume 2, IEEE, 2004, pp. 1980–1987.
[44] X. Wang, X.Z. Gao, S.J. Ovaska, A simulated annealing-based immune optimization method, in: Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning, Porvoo, Finland, 2008, pp. 41–47.
[45] D.F. Williamson, R.A. Parker, J.S. Kendrick, The box plot: a simple visual method to interpret data, Ann. Intern. Med. 110 (11) (1989) 916.
[46] H. David, Wolpert, G. William, Macready, No free lunch theorems for optimization, Evol. Comput., IEEE Trans. 1 (1) (1997) 67–82.
[47] A.R. Yildiz, A new hybrid Artificial Bee Colony Algorithm for robust optimal design and manufacturing, Appl. Soft Comput. 13 (5) (2013) 2906–2912.
[48] G. Zhu, S. Kwong, Gbest-guided Artificial Bee Colony algorithm for numerical function optimization, Appl. Math. Comput. 217 (7) (2010) 3166–3173.