Hybrid Artificial Bee Colony Algorithm with Differential Evolution

**Shimpi Singh Jadon · Ritu Tiwari · Harish Sharma ·
Jagdish Chand Bansal**

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.

Keywords Artificial Bee Colony ; Differential Evolution ; Optimization ; Hybridization ; Swarm intelligence

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.

Shimpi Singh Jadon
ABV-Indian Institute of Information Technology and Management, Gwalior E-mail: shimpisingh2k6@gmail.com

Ritu Tiwari
ABV-Indian Institute of Information Technology and Management, Gwalior E-mail: tiwariritu2@gmail.com

Harish Sharma
Rajasthan Technical University, Kota E-mail: harish.sharma0107@gmail.com

Jagdish Chand Bansal
South Asian University, New Delhi E-mail: jcbansal@gmail.com

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 Equation (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] 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 & 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 $x_i (i = 1, 2, ..., SN)$ of swarm is generated using a uniform distribution as follows:

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

here x_i is the ith potential solution in the swarm, x_{mind} and x_{maxd} are bounds of x_i in d^{th} 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) is modified as follows:

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

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:

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

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 Equation (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 Equation (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 $x_i(i = 1, 2, ..., D)$.

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:

$S = emptyset$
$d^* = U(1, D)$;
$S = S \cup d^*$;
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:

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

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 Equation (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 Equation (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 d^{th} 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 Equation (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 Equation (4). Then binomial crossover operator is applied to the trial solution and the current solution (the solution which is to be updated) using Equation (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 Equation (7) of GABC instead of Equation (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.

The search process of proposed HABCDE is also explained through Flow chart 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 to f_{20}) (listed in Table 1). Test problems $f_1 - f_{13}$ and $f_{17} - f_{20}$ are taken from [5] while test

---

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 Equation (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 Equation 4) to generate the trial solution u_i;
            Apply binomial crossover (as in Equation 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.

---

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} to 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 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 $1.0E - 01$.
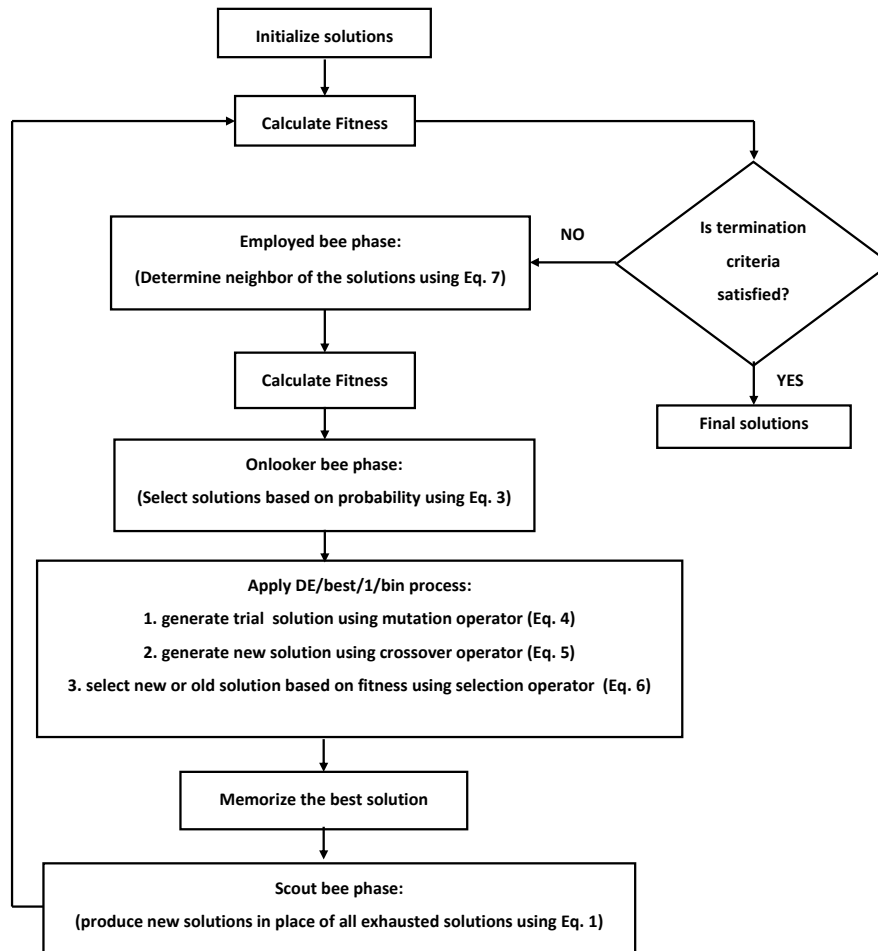
Fig. 1: Flowchart for HABCDE algorithm

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]	$f(\mathbf{0}) = 0$	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]	$f(\mathbf{1}) = 0$	30	UN	$1.0e-02$
Ackley	$f_3(x) = -20 + e + exp(-\frac{0.2}{d}\sqrt{\sum_{i=1}^{d} x_i{}^3})$ $-exp(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)x_i)$	[-1 1]	$f(\mathbf{0}) = 0$	30	MN	$1.0e-05$
Alpine	$f_4(x) = \sum_{i=1}^{d}|x_i\sin x_i + 0.1x_i|$	[-10 10]	$f(\mathbf{0}) = 0$	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]	$f(\mathbf{0}) = -d \times 0.1$	30	MS	$1.0e-05$
Exponential	$f_6(x) = -(exp(-0.5\sum_{i=1}^{d} x_i{}^2)) + 1$	[-1 1]	$f(\mathbf{0}) = -1$	30	MN	$1.0e-05$
Zakharov	$f_7(x) = \sum_{i=1}^{d} x_i{}^2 + (\sum_{i=1}^{d}\frac{ix_i}{2})^2 + (\sum_{i=1}^{d}\frac{ix_1}{2})^4$	[-5.12 5.12]	$f(\mathbf{0}) = 0$	30	MN	$1.0e-02$
Salomon Problem	$f_8(x) = 1 - cos(2\pi\sqrt{\sum_{i=1}^{d} x_i^2}) + 0.1(\sqrt{\sum_{i=1}^{d} x_i^2})$	[-100 100]	$f(\mathbf{0}) = 0$	30	MN	$1.0e-01$
Inverted cosine wave	$f_9(x) = -\sum_{i=1}^{d-1}\left(\exp\left(\frac{-(x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1})}{8}\right) \times \mathrm{I}\right)$ where, $\mathrm{I} = cos\left(4\sqrt{x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1}}\right)$	[-5 5]	$f(\mathbf{0}) = -d + 1$	10	MN	$1.0e-05$
Neumaier 3 Problem (NF3)	$f_{10}(x) = \sum_{i=1}^{d}(x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$	$[-d^2\ d^2]$	$f_{min} = -\frac{(d(d+4)(d-1))}{6}$	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]	$f(\mathbf{1}) = 0$	4	MN	$1.0e-05$
Braninss 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	$f_{13}(x) = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	[-5 5]	$f(0.192833, 0.190836, 0.123117, 0.135766) = 0.000307486$	4	MN	$1.0e-05$
Shifted Rosenbrock	$f_{14}(x) = \sum_{i=1}^{d-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}, z = x - o + 1, x = [x_1, x_2, ...x_d], o = [o_1, o_2, ...o_d]$	[-100 100]	$f(o) = f_{bias} = 390$	10	MN	$1.0e-01$
Shifted Sphere	$f_{15}(x) = \sum_{i=1}^{d} z_i^2 + f_{bias}, z = x - o, x = [x_1, x_2, ...x_d], o = [o_1, o_2, ...o_d]$	[-100 100]	$f(o) = f_{bias} = -450$	10	US	$1.0e-05$
Shifted Ackley	$f_{16}(x) = -20\exp(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{D} z_i^2}) - \exp(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi z_i)) + 20 + e + f_{bias}, z = (x - o), x = (x_1, x_2, ...x_d), o = (o_1, o_2, ...o_d)$	[-32 32]	$f(o) = f_{bias} = -140$	10	MN	$1.0e-05$
Goldstein-Price	$f_{17}(x) = (1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2)) \cdot (30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2))$	[-2 2]	$f(0, -1) = 3$	2	MN	$1.0e-14$
Six-hump camel back	$f_{18}(x) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1 x_2 + (-4 + 4x_2^2)x_2^2$	[-5 5]	$f(-0.0898, 0.7126) = -1.0316$	2	MN	$1.0e-05$
Easom's function	$f_{19}(x) = -\cos x_1\cos x_2 e^{((-(x_1-\pi)^2 - (x_2-\pi)^2))}$	[-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	$1.0e-6$

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 $S_3 = \{ABC\text{-}DE, DEBCO, HDABCA\}$ 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 200000 or acceptable error (Table 1) is found for the test problem,
- To set the DE parameters F and CR, sensitive analysis (see Figure 2) is carried out in the range $[0.1, 1]$. The horizontal axis in Figure 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.
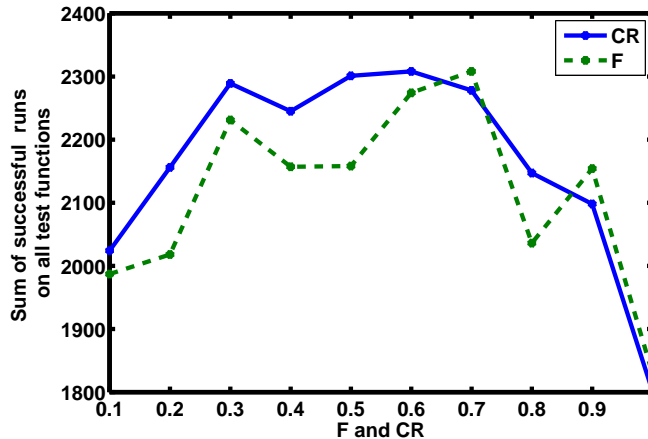
Fig. 2: Effect of parameters F and CR of DE on success rate

5.3 Analysis of Results

Tables 2, 3 and 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, 6 and 7 present the comparison for state-of-the-art algorithms of set S_2 and Tables 8, 9 and 10 present the comparison for hybrid algorithms of set S_3 for benchmark problems $f_1 - f_{20}$. Tables 2 to 10 numerically compare the SR, AFE and ME 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, 5 and 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, 6 and 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, 7 and 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 to 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 nonseparable 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.

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 Figures 3, 4 and 5 respectively. Figures 3, 4 and 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 Figures 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.

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 nonparametric Mann-Whitney test is used. The results output of this test for the AFEs of 100 runs is presented in Tables 11, 12 and 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, 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.

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	20409	30063	26735	22359	**9301**
f_2	186025	180325	177340	188136	**154936**
f_3	48727	42833	47957	43333	**17243**
f_4	98099	118190	59545	179705	**23572**
f_5	23016	30029	28709	22764	**9175**
f_6	16974	16707	22384	16648	**6902**
f_7	200000	200000	200000	200000	**100890**
f_8	186384	159415	122009	**28910**	38009
f_9	88543	87966	66979	63835	**37979**
f_{10}	198915	192527	186003	141729	**24389**
f_{11}	200000	152315	161103	143857	**6188**
f_{12}	2003	19603	21847	18969	**917**
f_{13}	182977	145154	96051	173568	**30506**
f_{14}	182883	168436	120690	163511	**43462**
f_{15}	9128	12045	9006	8675	**5676**
f_{16}	17744	31057	16654	14246	**7916**
f_{17}	125378	96395	97945	97907	**2506**
f_{18}	1014	97423	126874	102629	**660**
f_{19}	192327	24354	39735	172911	**5118**
f_{20}	76689	120126	111954	142171	**461**
f_{21}	187602	177523	173445	156234	**47961**
f_{22}	200000	200000	200000	200000	**9205**
f_{23}	200000	199693	188003	194993	**156593**
f_{24}	196985	52711	133956	35375	**3547**

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:

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

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**

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 $\alpha_3^i = \frac{Mo^i}{Ao^i}$

$i = 1, 2, ..., N_p$

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 Figures 6(a),(b),(c), 7(a),(b),(c) and 8(a),(b),(c) respectively. In PI Figures 6, 7, 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 Figures 6, 7 and 8 that PI of HABCDE is much higher than all the considered algorithms of sets S_1, S_2 and S_3 for

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	10873	22444	13977	10119	12694	**9301**
f_2	200000	200000	194000	200000	200000	**154936**
f_3	24486	42699	47663	23150	29744	**17243**
f_4	61987	60983	42537	**14293**	80728	23572
f_5	14606	30339	200000	200000	47354	**9174**
f_6	16004	17018	10169	200000	9696	**6902**
f_7	**48486**	68154	48528	106436	133128	100890
f_8	45564	58843	**12514**	186636	191535	38009
f_9	200000	176111	200000	200000	79381	**37979**
f_{10}	**11903**	17251	11905	180790	168642	24389
f_{11}	6286	22950	15985	21348	52228	**6188**
f_{12}	1012	1791	41554	200000	30221	**917**
f_{13}	**13434**	63953	49678	127142	44133	30506
f_{14}	**32584**	196183	93129	187111	156484	43462
f_{15}	9362	10325	8053	31015	5898	**5676**
f_{16}	17365	15537	12337	89123	9069	**7916**
f_{17}	4052	3829	8844	200000	3402	**2506**
f_{18}	1214	90867	112880	17752	114990	**660**
f_{19}	**2385**	4837	8881	192714	11790	5118
f_{20}	497	28804	13267	120001	189691	**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**

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 Figures 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 Figure 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

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	10119	20269	26376	**9301**
f_2	200000	199806	200000	**154936**
f_3	18672	40210	51250	**17243**
f_4	34569	178306	192466	**23572**
f_5	9584	20443	27137	**9175**
f_6	7671	15213	19743	**6902**
f_7	148546	200000	200000	**100890**
f_8	**11707**	31473	30285	38009
f_9	**27047**	71622	105196	37979
f_{10}	151418	197224	83330	**24389**
f_{11}	147268	193804	113683	**6188**
f_{12}	6511	30170	41794	**917**
f_{13}	198492	187817	195783	**30506**
f_{14}	180001	154473	181881	**43462**
f_{15}	6849	10402	8350	**5676**
f_{16}	10118	17426	13475	**7916**
f_{17}	17590	199353	109085	**2506**
f_{18}	2701	91578	90704	**660**
f_{19}	75818	200000	194872	**5118**
f_{20}	1387	31472	10765	**461**

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.

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

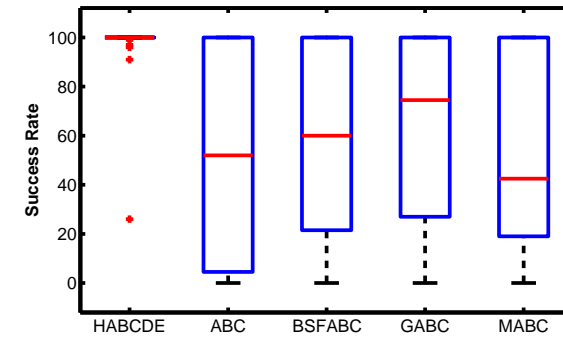Table 10: Mean Error (ME) for the algorithms of set S_3

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**

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.

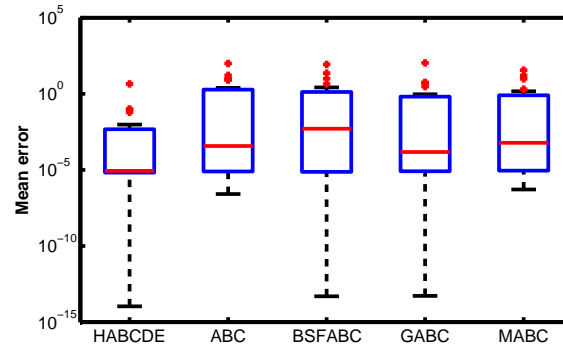| TP	Mann-Whitney U rank sum test with HABCDE					TP	Mann-Whitney U rank sum test with HABCDE				
	CMA-ES	DE	PSO	BBO	SMO		CMA-ES	DE	PSO	BBO	SMO
f_1	=	+	+	=	+	f_{11}	=	+	+	+	+
f_2	+	+	+	+	+	f_{12}	=	=	+	+	+
f_3	+	+	+	+	+	f_{13}	-	+	+	+	+
f_4	+	+	+	-	+	f_{14}	-	+	+	+	+
f_5	+	+	+	+	+	f_{15}	+	+	+	+	=
f_6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

(a) SR

(b) AFE

(c) ME

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

(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 be investigated in the proposed hybridization. The abil-

(a) SR

(b) AFE

(c) ME
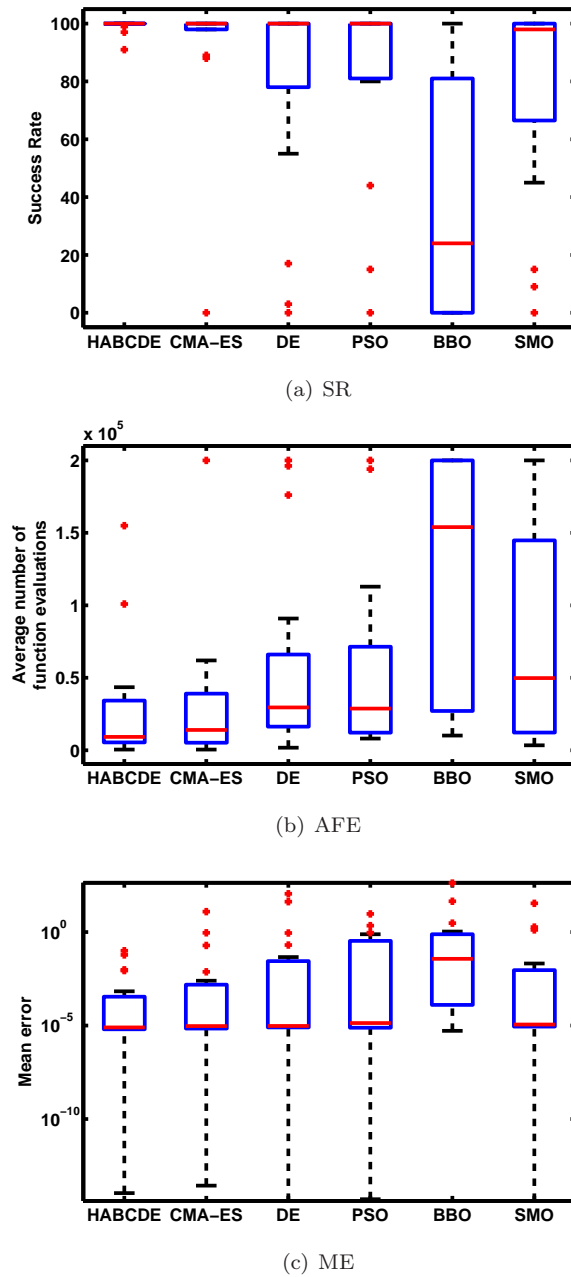
Fig. 4: Boxplots for algorithms of set S_2 on test problems $f_1 - f_{20}$

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

References

1. Afnizanfaizal Abdullah, Safaai Deris, and Mohd Saberi Mohamad. A new hybrid bee evoluation algorithm for parameter estimation in biological model. *ICIC express letters. Part B, Applications: an international journal of research and surveys*, 4(1):1–6, 2013.
2. Ajith Abraham, Ravi Kumar Jatoth, and A Rajasekhar. Hybrid differential artificial bee colony algorithm. *Journal of Computational and Theoretical Nanoscience*, 9(2):249–257, 2012.

(a) SR

(b) AFE

(c) ME
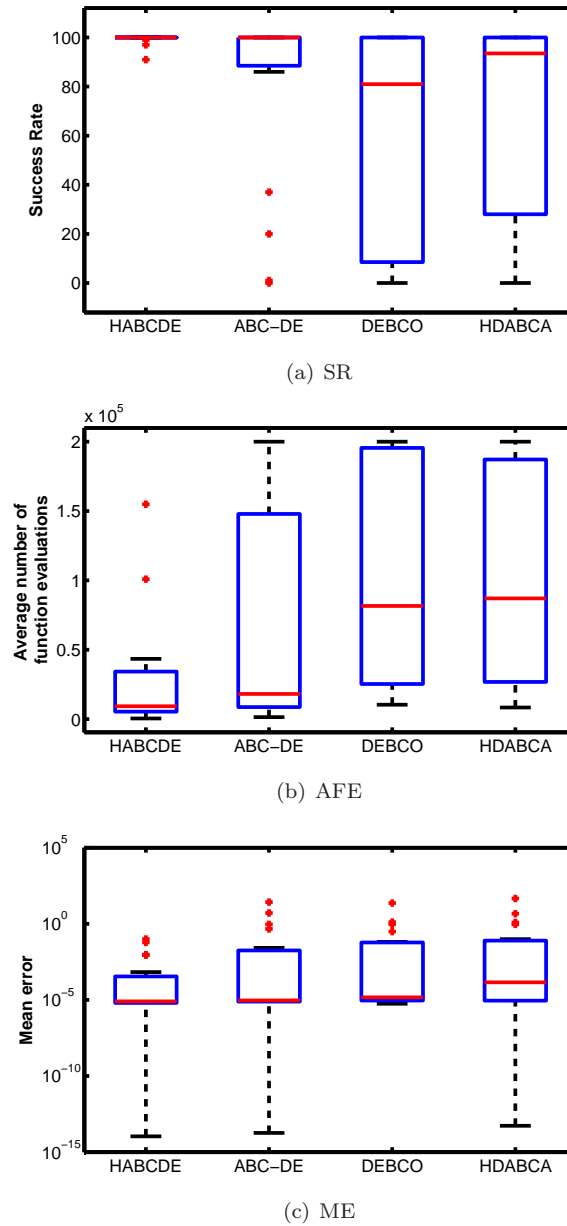
Fig. 5: Boxplots for algorithms of set S_3 on test problems $f_1 - f_{20}$

3. Ajith Abraham, Ravi Kumar Jatoth, and A Rajasekhar. Hybrid differential artificial bee colony algorithm. *Journal of Computational and Theoretical Nanoscience*, 9(2):249–257, 2012.
4. B. Akay and D. Karaboga. A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences*, doi:10.1016/j.ins.2010.07.015, 2010.
5. M.M. Ali, C. Khompatraporn, and Z.B. Zabinsky. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, 31(4):635–672, 2005.
6. A Alizadegan, MR Meybodi, and B Asady. A novel hybrid artificial bee colony algorithm and differential evolution for unconstrained optimization problems. *Advances in Computer Science and Engineering*, 8(1), 2012.
7. A. Banharnsakun, T. Achalakul, and B. Sirinaovakul. The best-so-far selection in artificial bee colony algorithm. *Applied Soft Computing*, 11(2):2888–2901, 2011.

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}	+	+	+

8. Jagdish Chand Bansal, Harish Sharma, Shimpi Singh Jadon, and Maurice Clerc. Spider monkey optimization algorithm for numerical optimization. *Memetic Computing*, pages 1–17, 2013.

9. J.C. Bansal and H. Sharma. Cognitive learning in differential evolution and its application to model order reduction problem for single-input single-output systems. *Memetic Computing*, pages 1–21, 2012.

10. S. Das and PN Suganthan. Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. *Jadavpur University, Kolkata, India, and Nangyang Technological University, Singapore, Tech. Rep*, 2010.

11. K. Diwold, A. Aderhold, A. Scheidler, and M. Middendorf. Performance evaluation of artificial bee colony optimization and new selection schemes. *Memetic Computing*, pages 1–14, 2011.

12. M. Dorigo and 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. Hai-Bin Duan, Chun-Fang Xu, and Zhi-Hui Xing. A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems. *International Journal of Neural Systems*, 20(01):39–50, 2010.

14. M. El-Abd. Performance assessment of foraging algorithms vs. evolutionary algorithms. *Information Sciences*, 182(1):243–263, 2011.

15. A.P. Engelbrecht. *Computational intelligence: an introduction*. Wiley, 2007.

16. W. Gao and S. Liu. A modified artificial bee colony algorithm. *Computers & Operations Research*, 2011.

17. Nikolaus Hansen. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006.

18. Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 312–317. IEEE, 1996.

19. R. Hooke and T.A. Jeeves. "direct search" solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961.

20. Shimpi Singh Jadon, Jagdish Chand Bansal, Ritu Tiwari, and Harish Sharma. Artificial bee colony algorithm with global and local neighborhoods. *International Journal of System Assurance Engineering and Management*, pages 1–13, 2014.

21. Shimpi Singh Jadon, Jagdish Chand Bansal, Ritu Tiwari, and Harish Sharma. Expedited artificial bee colony algorithm. In *Proceedings of the Third International Conference on Soft Computing for Problem Solving*, pages 787–800. Springer, 2014.

22. F. Kang, J. Li, Z. Ma, and H. Li. Artificial bee colony algorithm with local search for numerical optimization. *Journal of Software*, 6(3):490–497, 2011.

23. D. Karaboga. An idea based on honey bee swarm for numerical optimization. *Techn. Rep. TR06, Erciyes Univ. Press, Erciyes*, 2005.

24. D. Karaboga and B. Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108–132, 2009.

25. D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing*, 8(1):687–697, 2008.

26. Dervis Karaboga and Bahriye Akay. A modified artificial bee colony (abc) algorithm for constrained optimization problems. *Applied Soft Computing*, 11(3):3021–3031, 2011.
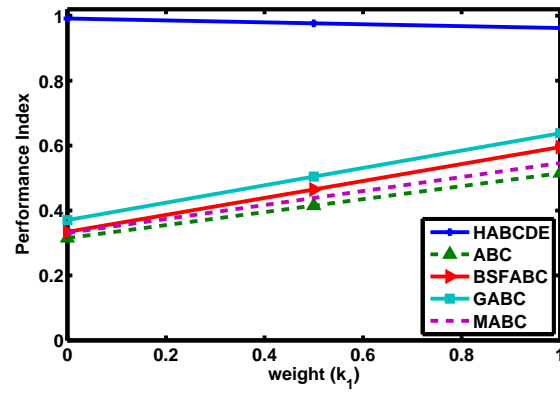
(a)
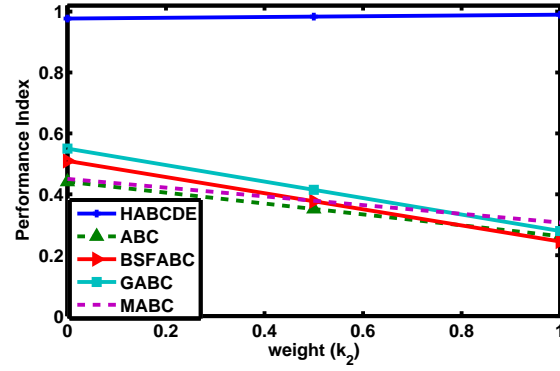
(b)
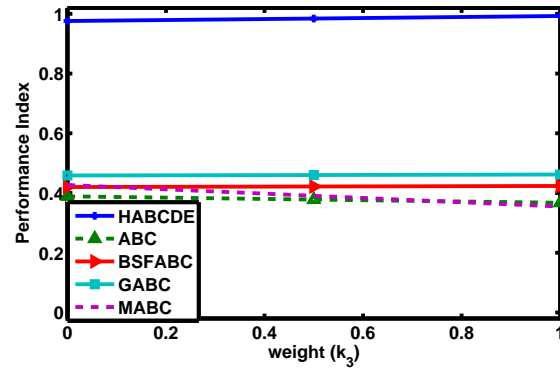
(c)

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).

27. J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.

28. Mustafa Servet Kıran and Mesut Gündüz. A novel artificial bee colony-based algorithm for solving the numerical optimization problems. *International Journal of Innovative Computing, Information & Control*, 8(9):6107–6121, 2012.

29. Yuancheng Li, Yiliang Wang, and Bin Li. A hybrid artificial bee colony assisted differential evolution algorithm for optimal reactive power flow. *International Journal of Electrical Power & Energy Systems*, 52:25–33, 2013.
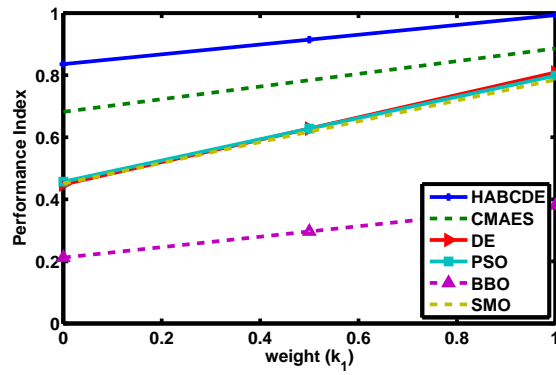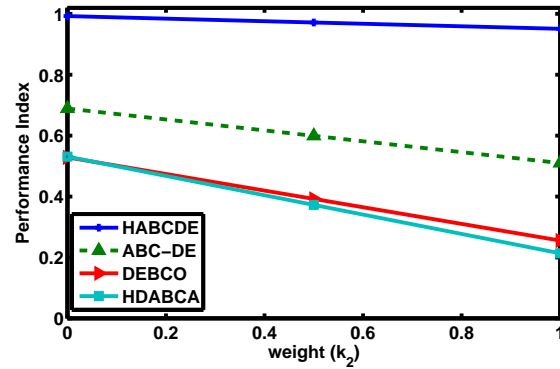
(a)
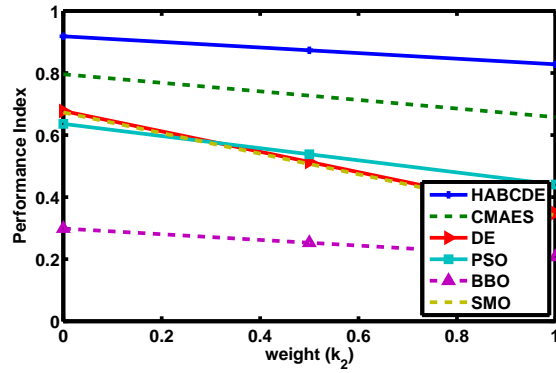
(b)
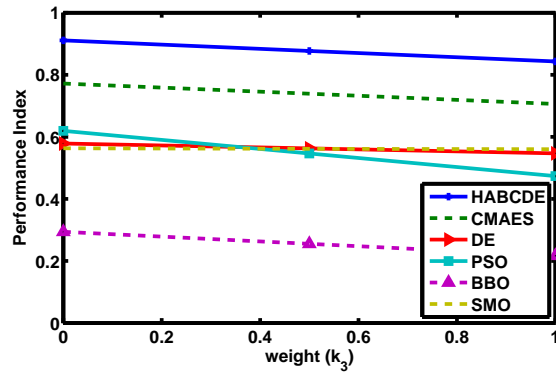
(c)

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).

30. Szymon Łukasik and Sławomir Żak. Firefly algorithm for continuous constrained optimization tasks. In *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, pages 97–106. Springer, 2009.
31. M. Mahdavi, M. Fesanghary, and E. Damangir. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188(2):1567–1579, 2007.
32. H.B. Mann and D.R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 18(1):50–60, 1947.
33. G.C. Onwubolu and BV Babu. *New optimization techniques in engineering*. Springer Verlag, 2004.
34. Celal Ozturk and Dervis Karaboga. Hybrid artificial bee colony algorithm for neural network training. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 84–88. IEEE, 2011.
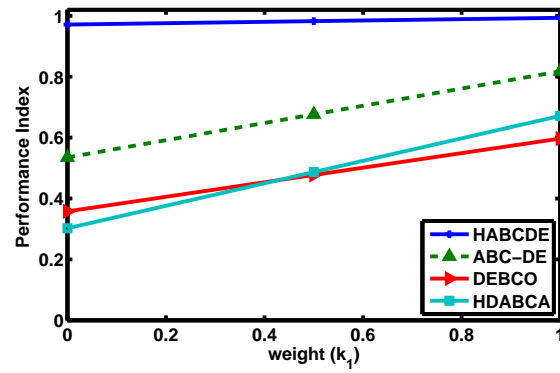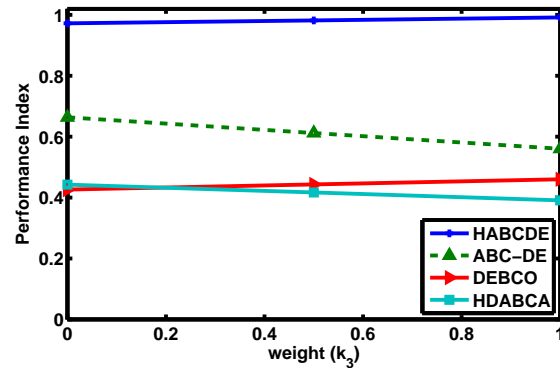
Fig. 8: Performance indices for algorithms of set S_3 on test problems; (a) for case (1), (b) for case (2) and (c) for case (3).

35. K.M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems Magazine, IEEE*, 22(3):52–67, 2002.
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*, pages 524–527. IEEE, 1996.
37. K.V. Price, R.M. Storn, and J.A. Lampinen. *Differential evolution: a practical approach to global optimization*. Springer Verlag, 2005.
38. KM Ragsdell and DT Phillips. Optimal design of a class of welded structures using geometric programming. *ASME Journal of Engineering for Industries*, 98(3):1021–1025, 1976.
39. E. Sandgren. Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design*, 112:223, 1990.
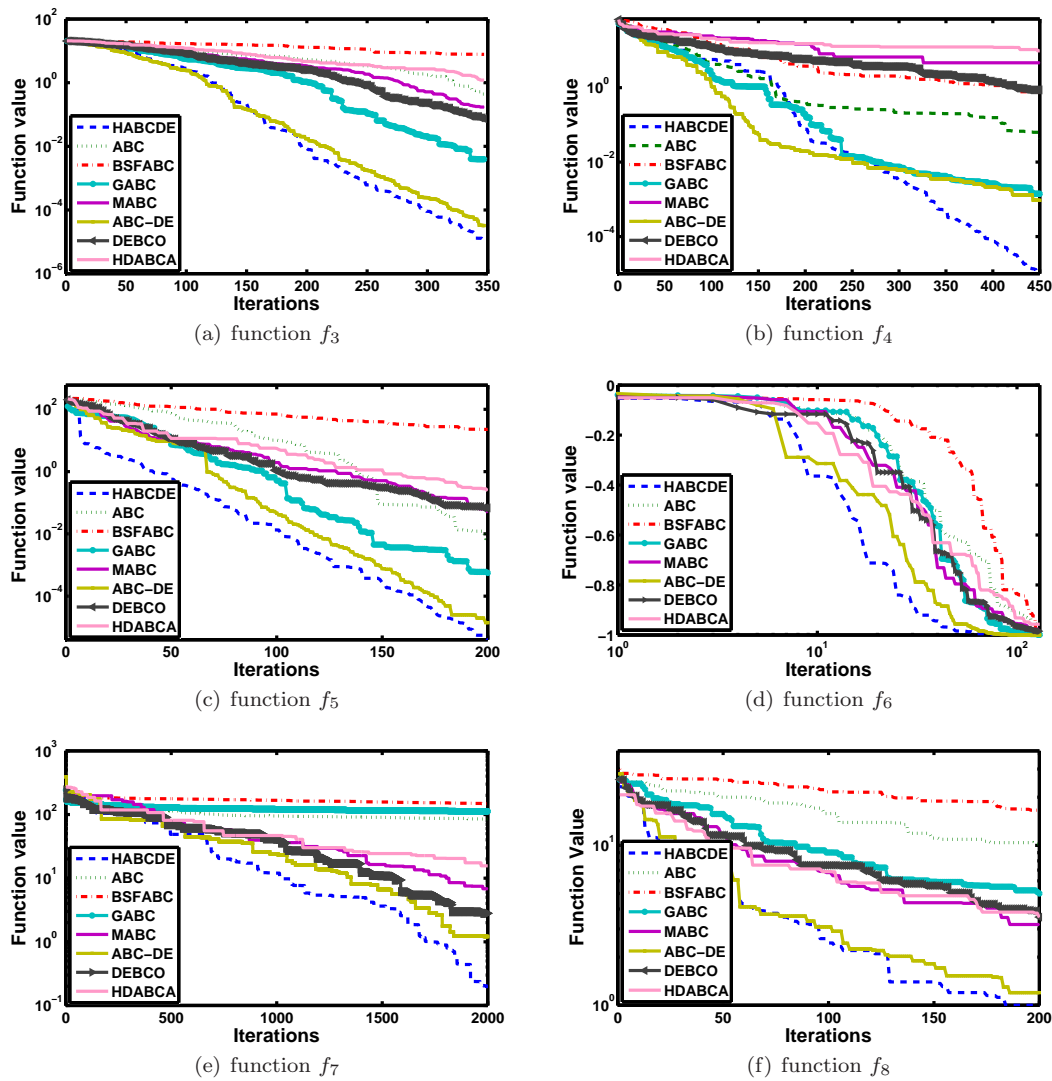
(a) function f_3 (b) function f_4

(c) function f_5 (d) function f_6

(e) function f_7 (f) function f_8

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

40. Dan Simon. Biogeography-based optimization. *Evolutionary Computation, IEEE Transactions on*, 12(6):702–713, 2008.
41. P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, YP Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. In *CEC 2005*, 2005.
42. Arit Thammano and Ajchara Phu-ang. A hybrid artificial bee colony algorithm with local search for flexible job-shop scheduling problem. *Procedia Computer Science*, 20:96–101, 2013.
43. J. Vesterstrom and 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, pages 1980–1987. IEEE, 2004.
44. X. Wang, XZ Gao, and SJ Ovaska. A simulated annealing-based immune optimization method. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning, Porvoo, Finland*, pages 41–47, 2008.
45. D.F. Williamson, R.A. Parker, and J.S. Kendrick. The box plot: a simple visual method to interpret data. *Annals of internal medicine*, 110(11):916, 1989.
46. David H Wolpert and William G Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

47. Ali R Yildiz. A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing. *Applied Soft Computing*, 13(5):2906–2912, 2013.

48. G. Zhu and S. Kwong. Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation*, 217(7):3166–3173, 2010.