mathematics

MDPI

Article

A New Hybrid BA_ABC Algorithm for Global Optimization Problems

Gülnur Yildizdan [1,*] and Ömer Kaan Baykan [2]

[1] Kulu Vocational School, Selcuk University, Kulu, 42770 Konya, Turkey
[2] Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Konya Technical University, 42250 Konya, Turkey; okbaykan@ktun.edu.tr
* Correspondence: gavsar@selcuk.edu.tr

Received: 1 September 2020; Accepted: 21 September 2020; Published: 12 October 2020

Abstract: Bat Algorithm (BA) and Artificial Bee Colony Algorithm (ABC) are frequently used in solving global optimization problems. Many new algorithms in the literature are obtained by modifying these algorithms for both constrained and unconstrained optimization problems or using them in a hybrid manner with different algorithms. Although successful algorithms have been proposed, BA's performance declines in complex and large-scale problems are still an ongoing problem. The inadequate global search capability of the BA resulting from its algorithm structure is the major cause of this problem. In this study, firstly, inertia weight was added to the speed formula to improve the search capability of the BA. Then, a new algorithm that operates in a hybrid manner with the ABC algorithm, whose diversity and global search capability is stronger than the BA, was proposed. The performance of the proposed algorithm (BA_ABC) was examined in four different test groups, including classic benchmark functions, CEC2005 small-scale test functions, CEC2010 large-scale test functions, and classical engineering design problems. The BA_ABC results were compared with different algorithms in the literature and current versions of the BA for each test group. The results were interpreted with the help of statistical tests. Furthermore, the contribution of BA and ABC algorithms, which constitute the hybrid algorithm, to the solutions is examined. The proposed algorithm has been found to produce successful and acceptable results.

Keywords: artificial bee colony algorithm; bat algorithm; continuous optimization; heuristic algorithms; large-scale optimization

1. Introduction

Meta-heuristic algorithms are often used in the solution of optimization problems. These algorithms use natural phenomena to achieve a specific purpose. Meta-heuristic algorithms have convergence features and can guarantee to find a solution close to the exact solution. Furthermore, these algorithms are frequently preferred for the solution of optimization problems due to their simple structures, easiness to understand, and realize [1]. Exploration and exploitation concepts are two important components for meta-heuristic algorithms. Exploration refers to the ability to explore various unknown regions of the solution space to find the global best value, while exploitation refers to the ability to use knowledge from previous best results to find better results. To achieve good performance in an optimization problem, the balance of these two components must be well adjusted [2,3].

The bat algorithm [4] was proposed by Xin-She Yang in 2010. The loudness (A) and pulse emission rate (r) parameters used in the bat algorithm significantly affect the exploration and exploitation abilities of the algorithm. As iterations progress, loudness decreases and pulse emission rate increases. This situation causes the exploitation ability to become prominent in the first iterations in the algorithm and the exploration ability to become prominent in the further iterations and decreases the probability

of the inclusion of new solutions found in the following steps of the algorithm to the population. There are many studies conducted to balance BA's exploration and exploitation ability and to overcome the mentioned structural problems. Cai et al. [5] added the optimal forage and random disturbance strategy concepts to the algorithm to determine the search direction of bats and improve their global search capability. Meng et al. [6] added the habitat selection and Doppler effect concepts to the bat algorithm. Thus, the algorithm has begun to imitate bat behaviors further. Cai et al. [7] developed the algorithm using the triangle-flipping strategy in the speed update formula that affects the global search capability of the bat algorithm. Zhu et al. [8] proposed a quantum-behaved bat algorithm. In this new algorithm, the position of each bat is determined by the optimal solution initially available, and by the mean best position, which can increase the convergence rate of the algorithm in the following iterations. Ghanem and Jantan [9] have included a special mutation operator that increases the diversity of the BA in the algorithm. Thus, they overcame the problem of getting stuck in the local minimum, which is frequently experienced in the bat algorithm. Shan and Cheng [10] proposed an advanced bat algorithm based on the covariance adaptive evolution process to diversify the direction and population distribution of the search in BA. Nawi et al. [11] proposed a new algorithm that determines the step size in BA using the Gaussian distributed random step logic to form sub-optimal solutions encountered in BA and to overcome the problems experienced in solving large-scale problems. Chakri et al. [12] added the directional echolocation strategy to the algorithm to overcome the early convergence caused by the low exploration ability of BA. Al-Betar and Awadallah [13] added the island model strategy to the algorithm to increase BA's capability to control diversity. Gan et al. [14] developed the algorithm with iterative local search and stochastic inertia weight strategies to improve BA's global search capability and reduce the risk of getting stuck at the local minimum. Topal and Altun [15] suggested the dynamic virtual bat's algorithm. In the algorithm, there are explorer bats that explore the search area and exploiter bats that are very likely to determine the desired target and search locally, and the roles of the bats vary depending on their location. Wang et al. [16] proposed a novel bat algorithm that includes multiple strategies for speed and position determination formulas to overcome BA's weakness in solving complex problems.

There are hybrid algorithms proposed in the literature to increase the performance of BA. Liu et al. [17] made three modifications in the algorithm to improve the performance of BA and developed a hybrid algorithm with the extremal optimization algorithm. Wang and Guo [18] proposed a hybrid algorithm that was created by adding the pitch adjustment operation of the harmony search algorithm as a mutation operator to the bat update process to speed up the convergence of the BA. Fister et al. [19] added the Differential Evolution Algorithm strategies to the BA to develop the best available solution that directs the found solutions to better regions of the search space. Imane and Nadjet [20] proposed a hybrid algorithm where BA's new solution selection phase is transformed into a tabu search to detect overlapping communities in social networks. Cincy and Jeba [21] proposed a new algorithm that hybridizes the BA and ABC algorithm, improves convergence speed, and the optimal accuracy. Chaudhary and Banati [22] proposed the Swarm Bat Algorithm with Improved Search(SBAIS) algorithm, which uses BA and the shuffled complex evolution (SCE) algorithm together to improve the exploration ability of BA. Rauf et al. [23] proposed an algorithm that combines Adaptive inertia weight and Sugeno-Function Fuzzy Search concepts to overcome the BA's premature convergence problem. Yıldızdan and Baykan [24] proposed a new algorithm using the hybrid differential evolution algorithm with an advanced BA algorithm to overcome the structural problems of BA and increase its exploration ability. Pan et al. [25] proposed a hybrid algorithm by developing a communication strategy between BA and ABC. In this hybrid algorithm, first, the population is divided into two subpopulations, then run in parallel with BA in one of the subpopulations and ABC in the other. At the end of each iteration, the worst individuals in the BA are replaced with the best individuals in the ABC; the worst individuals in the ABC are replaced by the best individuals in the BA. In the BA_ABC algorithm we proposed, dividing the population into two subpopulations and parallel run of BA and ABC algorithms on the subpopulations are similar to this article. Unlike this article, in the BA_ABC

algorithm, when a certain number of iterations is completed, the performances of BA and ABC are evaluated. A certain number of the worst individuals of the failed algorithm are replaced by the best individuals of the more successful algorithm. Also, when an algorithm reaches a certain ratio in the exchange of information, the remaining iterations are continued with this algorithm over the entire population; that is, the parallel run is terminated.

Many studies have been done on the BA and its performance has been significantly improved. However, performance decrease due to increased problem complexity and size is still an ongoing problem. In this study, a hybrid algorithm using the ABC algorithm, which works in parallel with the BA algorithm and can exchange information with the BA algorithm, was proposed to overcome the problems predicted in the BA algorithm. The ABC algorithm was preferred due to its features, such as the fact that its global search capability was stronger than the BA, and it had a mechanism to get rid of the local minimum. The performance of the hybrid algorithm proposed in this study was tested on both small and large problems. Its performance on real-world problems was also examined through tests on selected engineering design problems. The results obtained for each function set from the proposed hybrid algorithm were compared with the BA and ABC algorithms, recently proposed algorithms in the literature or the BA versions. Since not all algorithms selected from the literature contained results for all function sets that we worked with, the algorithms compared for each set of functions varied.

The rest of this study is organized as follows. In the second section, which includes materials and methods, information regarding the BA algorithm, the ABC algorithm, and the structure of the proposed hybrid BA_ABC algorithm is explained, respectively. In the third section, experimental studies and their results are presented. In the fourth section, the contributions of BA and ABC algorithms to the hybrid system are examined. In the fifth section, the complexity of BA, ABC, and BA_ABC algorithms are compared. In the sixth section, the results are discussed. Finally, in the seventh section, the conclusion and future work are explained.

2. Materials and Methods

2.1. Notations and Nomenclatures

In this section, all notations and nomenclatures used for the bat algorithm, artificial bee colony algorithm, and the proposed hybrid algorithm are given. Tables 1 and 2 show the notations and nomenclatures, respectively.

Table 1. Notations of Bat Algorithm (BA), Artificial Bee Colony (ABC), and BA_ABC.

BA Notations			
N	The number of individuals	v_i^{t-1}	The velocity of the ith bat in the previous iteration
t	The iteration number	x_i^t	The current location of the ith bat
v_i	The speed of the ith bat	x_i^{t+1}	The new location of the ith bat
x_i	The location of the ith bat	x_{old}	A solution among the best solutions
A	Loudness	x^*	Global best value
r	Pulse emission rate	ε	A random value in the range [−1,1]
A_0	The max value of loudness	α	Constant
A_{min}	The min value of loudness	γ	Constant
f_i	The frequency of the ith bat	A_i^t	The current loudness of the ith bat
f_{min}	The min value of frequency	A_i^{t+1}	The new loudness of the ith bat
f_{max}	The max value of frequency	$\overline{A^t}$	The average of loudness values of all bats
β	A random value in the range [0,1]	r_i^0	The initial value of the pulse emission rate of ith bat
v_i^t	The current velocity of the ith bat	r_i^t	The current pulse emission rate of ith bat
ABC Notations			
FN	The number of food source	$v_{i,j}$	The jth parameter value of the new food source
D	The number of parameters(dimension)	k	A random value in the range [1, FN]
ub_j	The upper limit value of the jth parameter	$x_{k,i}$	A randomly selected food source
lb_j	The lower limit value of the jth parameter	ϕ	A random value in the range [−1,1]
x_i	ith food source	f_i	The value of the objective function
v_i	A new food source located around the ith food source	p_i	The probability value of the ith food source
$x_{i,j}$	The jth parameter value of the ith food source	$fitness_i$	The fitness value of the ith food source

Table 1. *Cont.*

BA_ABC Notations			
w	The inertia weight coefficient	max_iteration	The number of maximum iteration
x_{new}	A new location located around the bat	$f(x_{new})$	The fitness value of the new location
$f(x^*)$	The fitness value of the global best		

Table 2. Nomenclatures of BA, ABC, and BA_ABC.

BA Nomenclatures	
echolocation	It is the biological sonar used by some mammals such as bats, dolphins, and whales.
sonar	The system detects the location and condition of an object with sound waves.
prey	The optimal solution the bats want to find
loudness	The intensity of the bat's sound when the bat is approaching prey
pulse emission rate	The spread rate of sound produced by bats during echolocation
global best value	The individual with the best fitness value in the population
step	Amount of progress in solution space
location	A possible solution for the problem
population	Set of possible solutions for the problem
ABC Nomenclatures	
food source	A possible solution for the problem
the amount of nectar	The quality of the solution
scout bee	Bees looking for a random food source in the environment
employed bee	A bee responsible for a food source and carrying information about that food source to the hive
onlooker bee	A bee waiting in the hive and selecting a food source depending on the nectar quality and searching around
fitness value	The value of the objective function
greedy selection	Selecting the one that is possible and closest to the result
BA_ABC Nomenclatures	
inertia weight	Coefficient determining the contribution of the previous speed
benchmark functions	The function which is used to test the performance of any optimization problem

2.2. Bat Algorithm

The Bat Algorithm (BA) is an algorithm proposed by Xin-She Yang [4] in 2010, based on the detection of direction and distance (echolocation behavior) of an object/prey, using sound reverberation. Bats can detect their prey/obstacles through echolocation.

According to Yang, the bat algorithm takes place according to the following rules [26].

- Each bat uses echolocation to measure how far the prey is.
- Bats fly with speed v_i to location x_i at a fixed frequency range [f_{min},f_{max}], emitting signals at various wavelengths (λ) and loudnesses (A) to detect their prey.
- When bats calculate the distance to their prey, they can adjust the pulse emission rate (r) along with the wavelength of the signal they send.
- Despite the variation in loudness, it is assumed that value A decreases from A_0 with a large value to a fixed minimum value (A_{min}).

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \tag{1}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^*) f_i \tag{2}$$

$$x_i^{t+1} = x_i^t + v_i^t \tag{3}$$

In the algorithm, the frequency, speed, and position values for the bat are calculated according to Equations (1)–(3). Frequency determines the step size in the algorithm. β refers to a random value in the range [0,1], and x^* refers to the global best value.

$$x_i^{t+1} = x_{old} + \varepsilon \overline{A^t} \tag{4}$$

In the local search section of the algorithm, a new solution is created for each bat by local random step around this solution by selecting a solution among the best solutions available (x_{old}). This is

done according to Equation (4). In this equation, provided that $\varepsilon \in [-1,1]$, ε is a random number that expresses the magnitude and direction of the step. $\overline{A^t}$ refers to the average of the current loudness values of all bats.

$$A_i^{t+1} = \alpha A_i^t \tag{5}$$

$$r_i^t = r_i^0[1 - \exp(-\gamma t)] \tag{6}$$

As bats approach their prey, loudness decreases, pulse emission rate increases (Equations (5) and (6)). α and γ values are also constant ($\alpha = \gamma = 0.9$).

2.3. Artificial Bee Colony Algorithm

The Artificial Bee Colony algorithm (ABC) is a swarm intelligence based optimization algorithm proposed by Karaboga [27] in 2005. This algorithm was created with inspiration from the foraging behavior of honeybees. Bees in this algorithm are named as scouts, employed bees, and onlooker bees according to their duties. In the algorithm, the number of employed bees is generally equal to the total number of food sources. The employed bee of the source becomes a scout bee when the nectar in the source runs out. In the ABC algorithm, the locations of food sources represent possible solutions to the optimization problem, and the amount of nectar refers to the quality of the solution [28]. The steps of the ABC algorithm can be listed as follows.

2.3.1. Beginning

At this step, scout bees begin to search for food sources randomly. The initial food sources are produced according to Equation (7) between the lower (lb_j) and upper limit (ub_j) values of each parameter.

$$x_{i,j} = lb_j + rand(0,1) \times (ub_j - lb_j) \tag{7}$$

In Equation (7), $i = 1,2,\ldots,FN$, $j = 1,2,\ldots,D$, where D is the number of parameters, and FN is the number of food sources. A counter that is used to indicate whether the amount of nectar of a food source has been exhausted is defined. This control parameter is called "limit". The defined number of limits is an important control parameter in the algorithm [29].

2.3.2. The Employed Bee Phase

The employed bee searches for a new food source around the current food source. It evaluates the food source found. If the amount of nectar in the new food source is better, it saves the new source and deletes the old food source. The new food source is calculated according to Equation (8).

$$v_{i,j} = x_{i,j} + \varphi \times \left(x_{i,j} - x_{k,j}\right) \tag{8}$$

In the equation, provided that $k \in \{1,2,\ldots,FN\}$, where k is a randomly selected food source, and ϕ is a random number in the range $[-1,1]$. After the quality (objective function value) of the source found is calculated, the fitness value of the source is assigned.

$$fitness_i = \begin{cases} 1/1 + f_i & f_i \geq 0 \\ 1 + |f_i| & f_i < 0 \end{cases} \tag{9}$$

In Equation (9), f_i represents the value of the objective function, that is, the quality of the source solution. When choosing between the existing food source and the new food source, the greedy selection method is applied depending on the fitness value. If food source v_i is better quality than existing source x_i, v_i is used as the new source and the limit counter is reset. Otherwise, the process continues with source x_i, and the limit counter is increased by one.

2.3.3. The Onlooker Bee Phase

Onlooker bees gather the sources information they obtained from employed bees. Onlooker bees choose a source in line with the probabilities proportional to the fitness values. For each source, a probability value is calculated according to Equation (10).

$$p_i = fitness_i / \sum_{k=1}^{FS} fitness_k \tag{10}$$

In Equation (10), p_i is the probability of the source i, and the $fitness_i$ is expressed the fitness value of the source i. If the probability value calculated for each source is less than a random value produced between [0,1], onlooker bees produce a new source according to Equation (8). The existing source and the newly produced source are evaluated according to the greedy selection method.

2.3.4. The Scout Bee Phase

If a food source has been exhausted or if a solution has not been developed during a trial number (limit) determined for this source, the employed bee of the source becomes scout bee, and a new food source is generated randomly according to Equation (7) instead of source x_i.

2.4. The Proposed Hybrid Algorithm (BA_ABC)

In the bat algorithm, r and A parameters are determinative in establishing the balance of exploitation and exploration. As the iteration progresses, the pulse emission rate increases, and loudness decreases due to the nature of the algorithm. The expression that performs the local search in the algorithm (Equation (4)) is executed depending on the condition that the parameter r is lower than a randomly generated number in the range [0,1]. Therefore, the increase in r value causes to perform a local search in the initial steps of the iteration and a global search in the next iterations. The inclusion of new and better solutions in the algorithm is done depending on the condition that parameter A is greater than a randomly generated number in the range [0,1]. As iterations progress, the decrease in A significantly lowers the possibility of including new solutions in the population towards the end of the iteration. In this study, a new hybrid algorithm has been proposed to overcome these structural problems of the BA and to improve global search capability.

In the literature, the inertia weight coefficient previously used in BA [24,30–32] was used in the velocity formula to improve the search capability of the bat algorithm in this study.

$$v_i^t = w \times v_i^{t-1} + \left(x_i^t - x^*\right) \times f_i \tag{11}$$

Accordingly, as seen in Equation (11), the inertia weight coefficient (w) was added to the velocity formula and used by reducing it chaotically [33] in the range of [0.9, 0.4].

As explained in Section 2.3, the ABC algorithm generates new solutions just by making changes in a randomly selected dimension during the new solution-searching process. This approach provides a more detailed search around the current solution and prevents rapid convergence towards the current solution. With this structure, the ABC algorithm is a stronger algorithm than the BA algorithm in terms of variety and global search capability. In this study, a hybrid algorithm (BA_ABC), in which BA and ABC are operated in parallel, is proposed to benefit from these capabilities of the ABC. In the hybrid algorithm, the population is divided into two. New solutions are produced by applying the BA algorithm to individuals in the first half of the population and applying the ABC algorithm to individuals in the second half. The best solution value (x^*) is updated with new solutions as better solutions are found by the algorithms. When each certain number of iterations (sc) is completed, the performances of the algorithms are examined. In other words, when they find a better solution than the solution defined for BA and ABC and the best solution available, the increased counters (ba_ni or bee_ni) are checked. In the past process, whichever algorithm has produced more new solutions

(i.e., whichever counter reaches a larger number), some of the individuals with the best fitness values in the population of that algorithm (as many as the number of ac's) supersede the same number of individuals in the other algorithm with the worst fitness values. Thus, an exchange of information between the algorithms is provided. After each information exchange, the counter of the successful algorithm (ba_sn or bee_sn), which has produced more new solutions, is increased. When any of these counters reach the maximum number of changes (mnc), the algorithm of the counter is considered to be more successful for that problem, and the remaining iterations are continued by executing this algorithm on the entire population.

Thanks to the proposed hybrid algorithm, the BA algorithm gains abilities such as developing global search capability, increasing diversity, and conducting detailed research around the optimal solution with the algorithm that is more successful towards the end of the iterations. The pseudo-code of the proposed algorithm is given in Algorithm 1. Some parameters used in the pseudo-code of the BA_ABC can be explained as follows.

x^*: It is the individual with the best fitness value in the population.

Bat_new_individual (ba_ni) and bee_new_individual (bee_ni): These are the counters that keep records of how many new solutions are created by BA and ABC, respectively.

Success control (sc): It is a predetermined number of iterations. When iterations equal to each sc value are completed, the performance of the algorithms is examined. In other words, the number of new solutions produced by BA and ABC is checked. Accordingly, the direction of the information exchange is determined. In the study, sc value was chosen equal to the limit value, which is an important parameter in the ABC algorithm.

Bat_success_number (ba_sn) and bee_success_number (bee_sn): This is the counter that keeps track of how many times BA and ABC have been successful in the exchange of information.

Amount of change (ac): It is the number that determines how many individuals with good fitness values in the population of the algorithm with high success will supersede the individuals with bad fitness values in the population of the other algorithm when the iterations as many as the sc value are completed (in this study, this number was determined as 10% of the number of individuals in the population).

Maximum number of change (mnc): It is the number that determines how many times the information exchange will be made between populations. This number is calculated according to Equation (12) in the study.

$$\text{mnc} = \left(\max^{f()}\text{\_iteration} / \text{sc}\right) \times 0.6 \quad (12)$$

The 0.6 multiplier in this equation was determined as a result of the tests performed. In the test performed, for the multiplier values of [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1], five general cycle averages were taken on the CEC2010 test functions and compared. According to these results, it was determined as 0.6. Accordingly, if an algorithm has been successful in 60% of the total number of changes, the remaining iterations are continued with this algorithm on the entire population.

Algorithm 1: The pseudo-code of BA_ABC

Input parameters: Population size (N), general number of cycles (gc), maximum iteration (max_i), dimension (D)

Output: Result of x^*

1. Determine population size (N), general number of cycles (gc), maximum iteration (max_i), dimension (D)
2. Set target function $f(x)$.
3. Construct the initial population of D-dimensional N individuals. $x=(x_1,x_2, \ldots ,x_N)^D$
4. Define the parameters of BA and ABC algorithms and bring them to the initial state.
5. Find the best value of the population x^*
6. Set the sc value.
7. Determine the mnc value according to Equation (12).
8. G = 1

9. While (G ≤ gc)
10. ba_ni = 0, bee_ni = 0, ba_sn = 0, bee_sn = 0
11. For t = 1: max_i
12. If (ba_sn < mnc and bee_sn < mnc)
13. For i = 1:N/2
14. Generate a new solution x_{new} according to BA.
15. If ($f(x_{new}) < f(x^*)$)
16. Accept the new solution.
17. Update x^*
18. ba_ni = ba_ni + 1;
19. End if
20. End For
21. For i = (N/2 + 1):N
22. Generate a new solution x_{new} according to ABC.
23. If ($f(x_{new}) < f(x^*)$)
24. Accept the new solution.
25. Update x^*
26. bee_ni = bee_ni + 1;
27. End if
28. End For
29. If (mod (t, sc) = = 0)
30. If (ba_ni ≥ bee_ni)
31. Write the individuals as many as ac with the best fitness values in BA in the place of the same number of individuals with the worst fitness values in the ABC population.
32. ba_sn = ba_sn + 1;
33. Else
34. Write the individuals as many as ac with the best fitness values in ABC in the place of the same number of individuals with the worst fitness values in the BA population.
35. bee_sn = bee_sn + 1;
36. End if
37. bee_ni = 0; ba_ni = 0;
38. End if
39. Else If (ba_sn ≥ mnc)
40. Find the remaining number of iterations for BA (t_BA)
41. For t1 = 1:t_BA
42. For i = 1: N
43. Generate a new solution x_{new} according to BA.
44. If ($f(x_{new}) < f(x^*)$)
45. Accept the new solution.
46. Update x^*
47. End if
48. End For
49. End For
50. Break;
51. Else
52. Find the remaining number of iterations for ABC (t_ABC)
53. For t2 = 1:t_ABC
54. For i = 1:N
55. Generate a new solution x_{new} according to ABC.
56. If ($f(x_{new}) < f(x^*)$)
57. Accept the new solution.
58. Update x^*
59. End if
60. End For
61. End For
62. Break;

63. End if
64. End For
65. G = G + 1
66. End While
67. Display the results

The computational complexity of BA_ABC is calculated as follows. Since the algorithm is a hybrid algorithm, it is necessary to consider the complexity of BA and ABC algorithms separately. Since there are no extra loops in the BA and ABC algorithms used in the BA_ABC algorithm, the algorithms have same computational complexity as the standard algorithms. For a P problem, let the computational complexity of its fitness evaluation function be O(P). Accordingly, the worst-case complexity of the standard BA is calculated as O(P × N), where N is the population size [7]. Similarly, the computational complexity of the standard ABC is also O (P × N). Additionally, two scenarios should be considered when calculating the complexity of the proposed hybrid algorithm. The first is the parallel run of BA and ABC in different halves of the population until the total number of iterations (t) is completed. In this case, the computational complexity of BA_ABC is t × (O(P × N/2) + O(2× P × N//2)). The second is that the algorithms work in parallel at the beginning, then one of the algorithms becomes more successful, and the remaining iterations continue with that algorithm. In this case, the complexity of BA_ABC is t1 × O(P × N/2) + t2 × O(2 × P × N/2), where t1 and t2 are the iteration number of BA and ABC, respectively, and t = t1 +t2.

3. Experimental Studies

In this section, the performance of the proposed algorithm on different test sets was examined and the results obtained were interpreted. The best mean values in the tables were highlighted in bold. In the first part, the performance of BA_ABC on 10 selected benchmark test functions was examined for different dimensions. In the second part, the performance of BA_ABC on CEC2005 small-scale test functions [34] was evaluated. The obtained results were compared to the BA versions. In the third section, the performance of BA_ABC on CEC2010 large-scale test functions [35] was examined. The obtained results were compared with the results of different algorithms proposed in recent years in the literature. Also, in the fourth section, the performance of BA_ABC on classical engineering design problems was tested and compared with different algorithms selected from the literature. Results were statistically interpreted with Wilcoxon signed-rank and Friedman tests [36,37]. These are non-parametric tests. Wilcoxon signed-rank test evaluates the differences between the paired results and determines whether there is a significant difference between the results. The Friedman test determines the differences between two or more algorithms and sorts the algorithms according to mean rank values [38].

The common parameter values used in all test sets for BA, ABC, and BA_ABC are as follows. The number of individuals (N) varies depending on dimension but is generally chosen between 10 and 100. f_{min} = 0, f_{max} = 1, r = 0.5, A = 0.9. The number of general cycles, dimensions, and the number of function evaluations (FEs) were chosen according to the rules of the function set (if any) or according to the most frequently used values in the literature. In addition, these parameters were the same for all of the compared algorithms for each function set.

In this study, ABC's limit value was chosen as 100 for small-scale problems and 2000 for large- scale problems. Since the ABC algorithm produces a new solution by making changes in one dimension, using a limit value similar to the small size in large-scale problems prevents a detailed search around the current solution. It often leads to the search for a new solution from a random point of the research space without much research. Therefore, it will be useful to increase the limit value depending on the dimension.

3.1. Performance of BA_ABC on Classic Benchmark Test Functions

The increase in the dimension of the problem is an important factor affecting the performance of metaheuristic algorithms. The performance of an algorithm that produces successful solutions for small scale problems usually decreases with the increase in dimension. In this section, the effect of dimension increase on the performance of the proposed algorithm is examined. The performance of the algorithm proposed was tested for different dimensions (10, 30, 50, 100, and 1000) on 10 classic benchmark functions that are frequently used in the literature. The maximum number of evaluations (FEs) was determined as 10,000 × D, and the number of general cycles as 25. The properties of the selected benchmark functions are given in Table 3.

The results obtained from BA_ABC were compared with standard BA and standard ABC algorithms for all dimensions. Comparison results are given in Table 4. Table 4 shows the mean value, standard deviation value, and statistical test results obtained for each dimension. In the Test (T) column, whether there was a significant difference between BA_ABC, and BA and ABC algorithms were examined by applying the Wilcoxon signed-rank test at $\alpha < 0.05$ significance level. The results of this test were shown with the symbols +, =, and −, which means that one algorithm is significantly better, equal, and worse than another algorithm, respectively. The numbers in parentheses indicate the total number of functions where BA_ABC produced better, worse, and equal solutions compared to other algorithms, respectively. The table also includes mean rank values that express the Friedman test-based ordering of the results obtained from BA, ABC, and BA_ABC algorithms for each dimension. When the Wilcoxon sign rank test results were analyzed for all dimensions, a significant difference was found between BA_ABC and the other algorithms in 89 of the total comparisons. The other algorithms were more successful than BA_ABC in 13 of the remaining comparisons, and no significant difference was found between the compared algorithms in the 18. According to the Friedman test mean rank values given for each dimension, the BA_ABC algorithm ranked first in all dimensions, the ABC algorithm ranked second, and the BA algorithm ranked third. In general, the number of errors obtained from the algorithms was observed to have increased due to the increase in dimensions. However, optimum results were obtained from the BA_ABC algorithm in all dimensions.

Similar studies in the literature for the classical functions used in this section are generally conducted for small dimensions. Therefore, the BA_ABC algorithm was compared with the algorithms selected from the literature for D = 30. The mean and standard deviation values of the results are given in Table 5. The results obtained for 30 dimensions were compared with the method used in the study of Fister Jr et al. (HSABA) [19], and with the results from BA (Bat Algorithm), FA (Firefly Algorithm), DE (Differential Algorithm), and ABC (Artificial Bee Colony) in the same study and those from the recently proposed Modified Bat Algorithm hybridizing by Differential Evolution algorithm(MBADE) [24]. When Table 5 was examined, it was found that the proposed method produced the best average values in six functions. Wilcoxon test revealed that there was no significant difference between this method and the hybrid versions of BA like HSABA and MBADE; however, there was a significant difference between this method and other well-known algorithms. According to the results of the Friedman test, BA_ABC ranked first with a mean rank value of 1.8; the MBADE algorithm ranked second with a mean rank value of 2.3.

Table 3. Classic benchmark function.

Function	Name	Definition	Domain/Characterisric	f^*
F1	Griewangk's function	$f(x) = 1/4000\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(x_i/(i)^{1/2}\right) + 1$	[−600, 600]/M	0
F2	Rastrigin's function	$f(x) = 10 * n + \sum_{i=1}^{n} \left(x_i^2 - 10 * \cos(2\lambda x_i)\right)$	[−15, 15]/M	0
F3	Rosenbrock's function	$f(x) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$	[−15, 15]/M	0
F4	Ackley's function	$f(x) = \sum_{i=1}^{n-1} (20 + \exp(-20)\exp(-0.2\sqrt{0.5(x_{i+1}^2 + x_i^2)}) - \exp(0.5(\cos(2\lambda x_{i+1}) + \cos(2\lambda x_i))))$	[−32.768, 32.768]/M	0
F5	Schwefel's function	$f(x) = 418.9829 * n - \sum_{i=1}^{n}\left[-x_i \sin(\lvert x_i\rvert^{1/2})\right]$	[−500, 500]/M	0
F6	Sphere function	$f(x) = \sum_{i=1}^{n} x_i^2$	[−600, 600]/U	0
F7	Easom's function	$f(x) = -(-1^n)(\prod_{i=1}^{n} \cos^2(x_i))\exp\left[-\sum_{i=1}^{n} (x_i - \lambda)^2\right]$	[−2λ, 2λ]/U	−1
F8	Michalewicz's function	$f(x) = -\sum_{i=1}^{n} \sin(x_i)\left[\sin\left(ix_i^2/\lambda\right)\right]^{20}$	[0, λ]/M	*
F9	Xin-She Yang's function	$f(x) = \left(\sum_{i=1}^{n} \lvert x_i\rvert\right)\exp\left[-\sum_{i=1}^{n} \sin(x_i^2)\right]$	[−2λ, 2λ]/M	0
F10	Zakharov's function	$f(x) = \sum_{i=1}^{n} x_i^2 + \left((1/2)\sum_{i=1}^{n} ix_i\right)^2 + \left((1/2)\sum_{i=1}^{n} ix_i\right)^4$	[−5, 10]/U	0

f^*: optimal solution *: depending on the size ($f^* = -1.8013$ for 2D). U: unimodal function. M: multimodal function.

Table 4. Results of BA, ABC, and BA_ABC algorithms in classical benchmark test function.

		D = 10			D = 30			D = 50		
		Mean	**Std**	**T (+, −, =)**	**Mean**	**Std**	**T (+, −, =)**	**Mean**	**Std**	**T (+, −, =)**
F1	BA	2.92×10^{1}	1.25×10^{1}	+(25,0,0)	1.92×10^{-1}	4.29×10^{-1}	+(25,0,0)	5.14×10^{3}	6.20×10^{3}	+(25,0,0)
	ABC	$\mathbf{4.70 \times 10^{-3}}$	5.19×10^{-3}	−(22,3,0)	2.83×10^{-14}	1.33×10^{-13}	+(24,0,1)	5.34×10^{-14}	2.58×10^{-13}	+(25,0,0)
	BA_ABC	8.80×10^{-3}	8.16×10^{-3}		$\mathbf{1.07 \times 10^{-16}}$	1.49×10^{-16}		$\mathbf{2.26 \times 10^{-16}}$	2.25×10^{-16}	
F2	BA	5.46×10^{1}	1.89×10^{1}	+(25,0,0)	2.00×10^{2}	4.67×10^{1}	+(25,0,0)	3.75×10^{2}	7.69×10^{1}	+(25,0,0)
	ABC	**0**	0	=(0,0,25)	5.68×10^{-14}	4.02×10^{-14}	=(11,3,11)	1.57×10^{-12}	2.60×10^{-12}	+(21,0,4)
	BA_ABC	**0**	0		$\mathbf{3.87 \times 10^{-14}}$	3.92×10^{-14}		$\mathbf{8.87 \times 10^{-14}}$	5.46×10^{-14}	
F3	BA	1.12×10^{1}	3.34×10^{1}	+(24,1,0)	3.25×10^{1}	2.75×10^{1}	+(25,0,0)	4.23×10^{1}	3.30×10^{0}	+(25,0,0)
	ABC	1.58×10^{-1}	1.73×10^{-1}	=(13,12,0)	5.92×10^{-2}	6.45×10^{-2}	+(23,2,0)	3.96×10^{-2}	6.14×10^{-2}	+(23,2,0)
	BA_ABC	$\mathbf{3.37 \times 10^{-2}}$	5.13×10^{-1}		$\mathbf{5.38 \times 10^{-3}}$	1.92×10^{-2}		$\mathbf{5.29 \times 10^{-3}}$	1.24×10^{-2}	
F4	BA	1.42×10^{2}	1.05×10^{0}	+(25,0,0)	4.74×10^{2}	7.58×10^{0}	+(25,0,0)	8.19×10^{2}	8.44×10^{0}	+(25,0,0)
	ABC	$\mathbf{1.40 \times 10^{2}}$	4.65×10^{-4}	+(25,0,0)	$\mathbf{4.50 \times 10^{2}}$	2.37×10^{-2}	+(23,2,0)	7.61×10^{2}	5.32×10^{-2}	+(24,1,0)
	BA_ABC	$\mathbf{1.40 \times 10^{2}}$	1.12×10^{-4}		$\mathbf{4.50 \times 10^{2}}$	1.84×10^{-2}		$\mathbf{7.60 \times 10^{2}}$	6.65×10^{-2}	
F5	BA	1.72×10^{3}	4.68×10^{2}	+(25,0,0)	5.36×10^{3}	8.24×10^{2}	+(25,0,0)	9.38×10^{3}	9.33×10^{2}	+(25,0,0)
	ABC	$\mathbf{1.27 \times 10^{-4}}$	3.96×10^{-13}	=(0,0,25)	$\mathbf{3.82 \times 10^{-4}}$	2.18×10^{-9}	=(1,0,24)	1.32×10^{-1}	4.58×10^{-1}	+(25,0,0)
	BA_ABC	$\mathbf{1.27 \times 10^{-4}}$	2.52×10^{-13}		$\mathbf{3.82 \times 10^{-4}}$	1.23×10^{-12}		$\mathbf{6.36 \times 10^{-4}}$	1.10×10^{-11}	
F6	BA	7.83×10^{3}	7.49×10^{3}	+(25,0,0)	6.77×10^{2}	2.58×10^{3}	+(25,0,0)	3.73×10^{-4}	7.18×10^{-5}	+(25,0,0)
	ABC	8.3×10^{-17}	2.49×10^{-17}	+(25,0,0)	5.73×10^{-16}	8.52×10^{-17}	+(25,0,0)	1.19×10^{-15}	1.72×10^{-16}	+(25,0,0)
	BA_ABC	$\mathbf{2.86 \times 10^{-17}}$	1.73×10^{-17}		$\mathbf{2.07 \times 10^{-16}}$	7.49×10^{-17}		$\mathbf{4.09 \times 10^{-16}}$	1.26×10^{-16}	
F7	BA	-9.92×10^{-1}	4.77×10^{-3}	+(25,0,0)	$\mathbf{-5.72 \times 10^{-1}}$	4.92×10^{-1}	−(11,14,0)	-7.27×10^{-3}	1.27×10^{-2}	−(7,18,0)
	ABC	-7.16×10^{-3}	1.29×10^{-2}	+(25,0,0)	-6.07×10^{-108}	3.04×10^{-107}	+(25,0,0)	-2.54×10^{-255}	0	+(25,0,0)
	BA_ABC	**−1**	1.05×10^{-4}		-2.14×10^{-1}	4.01×10^{-1}		$\mathbf{-4.15 \times 10^{-2}}$	2.00×10^{-1}	
F8	BA	-5.89×10^{0}	4.04×10^{-1}	+(25,0,0)	-1.04×10^{1}	8.39×10^{-1}	+(25,0,0)	-1.42×10^{1}	9.63×10^{-1}	+(25,0,0)
	ABC	$\mathbf{-9.66 \times 10^{0}}$	1.56×10^{-7}	+(5,0,20)	$\mathbf{-2.96 \times 10^{1}}$	8.66×10^{-3}	=(19,6,0)	-4.95×10^{1}	2.05×10^{-2}	+(23,2,0)
	BA_ABC	$\mathbf{-9.66 \times 10^{0}}$	2.51×10^{-13}		$\mathbf{-2.96 \times 10^{1}}$	1.92×10^{-2}		$\mathbf{-4.96 \times 10^{1}}$	1.48×10^{-2}	
F9	BA	2.33×10^{-3}	4.94×10^{-4}	+(25,0,0)	2.06×10^{-11}	8.60×10^{-12}	+(25,0,0)	$\mathbf{7.34 \times 10^{-20}}$	4.18×10^{-20}	=(9,16,0)
	ABC	$\mathbf{5.66 \times 10^{-4}}$	1.65×10^{-16}	=(0,0,25)	$\mathbf{3.51 \times 10^{-12}}$	6.71×10^{-16}	+(24,1,0)	1.96×10^{-17}	2.20×10^{-18}	+(25,0,0)
	BA_ABC	$\mathbf{5.66 \times 10^{-4}}$	1.07×10^{-16}		$\mathbf{3.51 \times 10^{-12}}$	2.40×10^{-16}		1.15×10^{-19}	8.52×10^{-20}	
F10	BA	2.56×10^{-3}	1.93×10^{-3}	+(25,0,0)	2.94×10^{-2}	6.67×10^{-3}	+(25,0,0)	1.07×10^{-1}	1.68×10^{-2}	+(25,0,0)
	ABC	3.90×10^{1}	3.65×10^{1}	+(25,0,0)	2.49×10^{3}	3.29×10^{2}	+(25,0,0)	5.09×10^{3}	2.87×10^{2}	+(25,0,0)
	BA_ABC	$\mathbf{1.41 \times 10^{-4}}$	$\mathbf{8.86 \times 10^{-5}}$		$\mathbf{9.12 \times 10^{-4}}$	3.29×10^{-4}		$\mathbf{2.74 \times 10^{-3}}$	6.03×10^{-4}	
Mean Rank		BA	ABC	BA_ABC	BA	ABC	BA_ABC	BA	ABC	BA_ABC
		2.80	1.85	**1.35**	2,70	2.00	**1.30**	2.60	2.30	**1.10**

Table 4. *Cont.*

		D = 100			D = 1000		
		Mean	**Std**	**T (+, − , =)**	**Mean**	**Std**	**T (+, − , =)**
F1	BA	1.84×10^{-1}	7.05×10^{-1}	+(25,0,0)	1.91×10^{-3}	1.93×10^{-3}	+(25,0,0)
	ABC	6.83×10^{-15}	6.19×10^{-15}	+(25,0,0)	1.38×10^{-14}	7.46×10^{-16}	+(23,2,0)
	BA_ABC	$\mathbf{7.42 \times 10^{-16}}$	6.59×10^{-16}		$\mathbf{4.26 \times 10^{-15}}$	8.36×10^{-15}	
F2	BA	9.83×10^{2}	1.23×10^{2}	+(25,0,0)	1.18×10^{4}	3.13×10^{3}	+(25,0,0)
	ABC	4.66×10^{-5}	1.77×10^{-4}	+(25,0,0)	2.84×10^{1}	1.47×10^{0}	+(25,0,0)
	BA_ABC	$\mathbf{3.91 \times 10^{-13}}$	2.77×10^{-13}		$\mathbf{5.73 \times 10^{0}}$	1.53×10^{0}	
F3	BA	9.42×10^{1}	1.44×10^{1}	+(20,5,0)	1.05×10^{3}	3.58×10^{1}	+(25,0,0)
	ABC	$\mathbf{2.88 \times 10^{-2}}$	2.65×10^{-2}	−(1,24,0)	$\mathbf{1.21 \times 10^{2}}$	4.53×10^{1}	=(14,11,0)
	BA_ABC	4.30×10^{1}	6.46×10^{1}		1.50×10^{2}	1.08×10^{2}	
F4	BA	1.60×10^{3}	6.94×10^{1}	+(25,0,0)	$\mathbf{1.55 \times 10^{4}}$	7.95×10^{-2}	−(0,25,0)
	ABC	$\mathbf{1.54 \times 10^{3}}$	1.87×10^{-1}	+(25,0,0)	$\mathbf{1.55 \times 10^{4}}$	1.26×10^{-1}	+(25,0,0)
	BA_ABC	$\mathbf{1.54 \times 10^{3}}$	2.40×10^{-1}		$\mathbf{1.55 \times 10^{4}}$	1.75×10^{-1}	
F5	BA	1.90×10^{4}	1.43×10^{3}	+(25,0,0)	2.00×10^{5}	4.34×10^{3}	+(25,0,0)
	ABC	$\mathbf{1.87 \times 10^{2}}$	9.32×10^{1}	−(6,19,0)	1.30×10^{4}	5.45×10^{2}	+(25,0,0)
	BA_ABC	4.12×10^{2}	2.30×10^{2}		$\mathbf{7.37} \times 10^{3}$	1.64×10^{3}	
F6	BA	5.26×10^{1}	1.81×10^{2}	+(25,0,0)	1.31×10^{4}	2.40×10^{4}	+(25,0,0)
	ABC	3.60×10^{-15}	5.56×10^{-16}	+(25,0,0)	$\mathbf{3.80 \times 10^{-14}}$	3.10×10^{-15}	=(19,6,0)
	BA_ABC	$\mathbf{1.23 \times 10^{-15}}$	4.49×10^{-16}		6.76×10^{-14}	9.05×10^{-14}	
F7	BA	**0**	0	=(0,0,25)	**0**	0	=(0,0,25)
	ABC	**0**	0	=(0,0,25)	**0**	0	=(0,0,25)
	BA_ABC	**0**	0		**0**	0	
F8	BA	-2.25×10^{1}	1.22×10^{0}	+(25,0,0)	-1.49×10^{2}	4.27×10^{0}	+(25,0,0)
	ABC	-9.91×10^{1}	5.52×10^{-2}	+(24,1,0)	-9.68×10^{2}	5.81×10^{-1}	+(25,0,0)
	BA_ABC	$\mathbf{-9.93 \times 10^{1}}$	6.64×10^{-2}		$\mathbf{-9.80 \times 10^{2}}$	4.60×10^{-1}	
F9	BA	3.89×10^{-41}	3.40×10^{-41}	=(11,14,0)	**0**	0	=(0,0,25)
	ABC	1.56×10^{-17}	2.04×10^{-18}	+(25,0,0)	3.02×10^{-73}	7.60×10^{-73}	+(25,0,0)
	BA_ABC	$\mathbf{3.50 \times 10^{-41}}$	2.93×10^{-41}		**0**	0	
F10	BA	1.14×10^{-1}	3.62×10^{-2}	+(25,0,0)	1.30×10^{4}	8.67×10^{2}	+(25,0,0)
	ABC	1.19×10^{4}	4.77×10^{2}	+(25,0,0)	1.59×10^{5}	1.12×10^{3}	+(25,0,0)
	BA_ABC	$\mathbf{1.14 \times 10^{-2}}$	1.72×10^{-3}		$\mathbf{1.43 \times 10^{3}}$	1.29×10^{2}	
Mean Rank		BA	ABC	BA_ABC	BA	ABC	BA_ABC
		2.70	1.95	**1.35**	2.55	2.00	**1.45**

Table 5. Comparison with the other evolutionary algorithm (D = 30).

Function		BA_ABC	BA	FA	DE	ABC	HSABA	MBADE
F1	Mean	$\mathbf{1.07 \times 10^{-16}}$	1.16×10^{0}	6.65×10^{-1}	1.05×10^{0}	1.09×10^{0}	7.71×10^{-2}	6.39×10^{-3}
	Std	1.49×10^{-16}	1.15×10^{0}	6.40×10^{-1}	2.22×10^{-2}	1.23×10^{-1}	2.85×10^{-2}	1.12×10^{-2}
F2	Mean	$\mathbf{3.87 \times 10^{-14}}$	9.28×10^{2}	2.44×10^{2}	2.28×10^{2}	7.33×10^{1}	4.63×10^{1}	8.23×10^{0}
	Std	3.92×10^{-14}	8.90×10^{2}	2.35×10^{2}	1.33×10^{1}	2.24×10^{1}	2.99×10^{1}	4.10×10^{0}
F3	Mean	$\mathbf{5.38 \times 10^{-3}}$	2.84×10^{6}	1.12×10^{2}	4.57×10^{2}	5.18×10^{2}	1.02×10^{2}	1.14×10^{1}
	Std	1.92×10^{-2}	2.95×10^{6}	1.01×10^{2}	2.27×10^{2}	4.72×10^{2}	1.41×10^{1}	1.25×10^{1}
F4	Mean	4.50×10^{2}	2.00×10^{1}	2.11×10^{1}	$\mathbf{1.77 \times 10^{0}}$	7.17×10^{0}	9.44×10^{0}	4.56×10^{2}
	Std	1.84×10^{-2}	2.00×10^{1}	2.11×10^{1}	3.17×10^{-1}	1.03×10^{0}	6.62×10^{0}	1.16×10^{0}
F5	Mean	$\mathbf{3.82 \times 10^{-4}}$	9.45×10^{3}	6.78×10^{3}	7.57×10^{3}	2.64×10^{3}	2.70×10^{2}	1.17×10^{2}
	Std	1.23×10^{-12}	9.52×10^{3}	6.75×10^{3}	4.40×10^{2}	3.30×10^{2}	3.06×10^{1}	1.55×10^{2}
F6	Mean	2.07×10^{-16}	5.87×10^{-2}	5.19×10^{0}	1.77×10^{2}	1.63×10^{2}	2.63×10^{-2}	$\mathbf{2.65 \times 10^{-20}}$
	Std	7.49×10^{-17}	6.53×10^{-5}	5.14×10^{0}	7.12×10^{1}	1.96×10^{2}	1.29×10^{-13}	2.91×10^{-20}
F7	Mean	-2.14×10^{-1}	0	-3.81×10^{-30}	-2.76×10^{-175}	-1.76×10^{-136}	0	$\mathbf{-1}$
	Std	4.01×10^{-1}	0	-3.73×10^{-30}	0	8.79×10^{-136}	0	3.20×10^{-17}
F8	Mean	$\mathbf{-2.96 \times 10^{1}}$	-8.62×10^{0}	-5.15×10^{0}	-1.07×10^{1}	-2.30×10^{1}	-1.30×10^{1}	-2.56×10^{1}
	Std	1.92×10^{-2}	-8.39×10^{0}	-5.35×10^{0}	6.70×10^{-1}	6.98×10^{-1}	-1.36×10^{1}	4.91×10^{-1}
F9	Mean	$\mathbf{3.51 \times 10^{-12}}$	1.57×10^{-11}	1.70×10^{-4}	2.46×10^{-11}	1.10×10^{-11}	6.06×10^{-12}	6.81×10^{-12}
	Std	2.40×10^{-16}	1.03×10^{-11}	4.72×10^{-5}	1.20×10^{-12}	1.91×10^{-12}	3.85×10^{-12}	4.42×10^{-13}
F10	Mean	9.12×10^{-4}	2.76×10^{2}	1.32×10^{4}	3.78×10^{1}	2.53×10^{2}	2.72×10^{1}	$\mathbf{1.46 \times 10^{-9}}$
	Std	3.29×10^{-4}	2.82×10^{2}	1.32×10^{4}	8.74×10^{0}	3.15×10^{1}	1.37×10^{0}	3.13×10^{-9}
Wilcoxon Test								
+			9	9	9	9	9	7
-			1	1	1	1	1	3
=			0	0	0	0	0	0
P			0.037(+)	0.047(+)	0.047(+)	0.047(+)	0.074(≈)	0.059(≈)
Friedman Test								
Mean Rank		**1.80**	5.90	5.60	4.80	4.30	3.30	2.30

3.2. Performance of BA_ABC on CEC2005 Test Functions

In this section, the performance of BA_ABC on CEC2005 test functions [34] is examined. CEC2005 functions consist of 25 functions with different characteristics. These functions were modified and made more complex by performing operations such as translation, rotation, or hybridization. CEC2005 functions and features were given in Table 6. Under CEC2005 rules, the number of general cycles was determined as 25. The maximum number of evaluations was chosen as 10,000 × D. After each cycle, the function error $f(x) - f(x^*)$ was recorded and sorted in descending order. The 1st (best), 7th, 13th (median), 19th, and 25th (worst) values of the function error were taken, and mean and standard deviation values were calculated using all function error values.

Results of BA, ABC, and BA_ABC on CEC2005 test functions for D = 10 are given in Table 7. According to Table 7, BA, ABC produced the best mean values in 22 of 25 functions and BA in the remaining three. Again, BA_ABC produced better mean values in all functions than ABC. When BA and ABC were compared, it was seen that BA produced better mean values than ABC in 22 functions. When compared by the test line in the table, BA_ABC performed better in 22 functions compared to BA, and a significant difference was found between them. BA_ABC had an equal performance to that of BA in one of the remaining functions, and worse performance than that of BA in two functions. According to test results between ABC_BA and ABC, BA_ABC performed better in all functions, and there was a significant difference between them. According to the results of the Friedman test given at the bottom of Table 7, the BA_ABC algorithm ranked first with a mean rank value of 1.12, the BA algorithm ranked second with a mean rank value of 1.96, and the ABC algorithm ranked third with a mean rank value of 2.92.

Table 6. CEC2005 benchmark functions.

Function	Name	Bound	f_{min}
Unimodal Functions			
F1	Shifted Sphere Function	[−100, 100]	−450
F2	Shifted Schwefel's Problem 1.2	[−100, 100]	−450
F3	Shifted Rotated High Conditioned Elliptic Function	[−100, 100]	−450
F4	Shifted Schwefel's Problem 1.2 with Noise in Fitness	[−100, 100]	−450
F5	Schwefel's Problem 2.6 with Global Optimum on Bounds	[−100, 100]	−310
Multimodal functions			
F6	Shifted Rosenbrock's Function	[−100, 100]	390
F7	Shifted Rotated Griewank's Function without Bounds	[0, 600]	−180
F8	Shifted Rotated Ackley's Function with Global Optimum on Bounds	[−32, 32]	−140
F9	Shifted Rastrigin's Function	[−5, 5]	−330
F10	Shifted Rotated Rastrigin's Function	[−5, 5]	−330
F11	Shifted Rotated Weierstrass Function	[−0.5, 0.5]	90
F12	Schwefel's Problem 2.13	[$-\lambda$,λ]	−460
Expanded functions			
F13	Expanded Extended Griewank's plus Rosenbrock's Function (F8F2)	[−3, 1]	−130
F14	Expanded Rotated Extended Scaffe's F6	[−100, 100]	−300
Hybrid composition functions			
F15	Hybrid Composition Function 1	[−5, 5]	120
F16	Rotated Hybrid Composition Function 1	[−5, 5]	120
F17	Rotated Hybrid Composition Function 1 with Noise in Fitness	[−5, 5]	120
F18	Rotated Hybrid Composition Function 2	[−5, 5]	10
F19	Rotated Hybrid Composition Function 2 with a Narrow Basin for the Global Optimum	[−5, 5]	10
F20	Rotated Hybrid Composition Function 2 with the Global Optimum on the Bounds	[−5, 5]	10
F21	Rotated Hybrid Composition Function 3	[−5, 5]	360
F22	Rotated Hybrid Composition Function 3 with High Condition Number Matrix	[−5, 5]	360
F23	Non-Continuous Rotated Hybrid Composition Function 3	[−5, 5]	360
F24	Rotated Hybrid Composition Function 4	[−5, 5]	260
F25	Rotated Hybrid Composition Function 4 without Bounds	[2, 5]	260

BA_ABC was also compared with other BA versions in the literature. Accordingly, BA_ABC was compared with the recently proposed SBAIS [22], MBADE [24], a Novel Bat Algorithm with habitat selection and doppler effect in echoes (NBA) [6], Island Bat Algorithm (iBA) [13], Bat Algorithm based on Iterative Local Search and Stochastic Inertia Weight (ILSSIWBA) [14], and Global-best Bat-inspired Algorithm(GBA), Tournament Bat-inspired Algorithm (TBA), Proportional Bat-inspired Algorithm (PBA), Linear rank Bat-inspired Algorithm (LBA), Exponential rank Bat-inspired Algorithm (EBA), and Random Bat-inspired Algorithm (RBA) versions [26] proposed in the same study. The mean values of these algorithms are given in Table 8. Accordingly, BA_ABC produced the best mean values in five functions. According to the Wilcoxon test results, it was determined that BA_ABC had an equal performance to the MBADE, ILSSIWBA, TBA, and EBA algorithms, and there was no significant difference between them. BA_ABC performed worse than the SBAIS algorithm and better than the remaining algorithms, and there was a significant difference between them. According to Friedman test results, the SBAIS algorithm ranked first with a mean rank value of 2.78, and the MBADE algorithm ranked second with a mean rank value of 4.16. In addition, the BA_ABC ranked fourth with a mean rank value of 5.22.

Table 7. CEC2005 test results of BA, ABC, and BA_ABC (D = 10).

Function		1st (Best)	7th	13th (Median)	19th	25th (Worst)	Mean	Std	Test (+, −, =)
F1	BA	1.5×10^{-3}	2.89×10^{-3}	4.01×10^{-3}	5.47×10^{-3}	6.74×10^{-3}	4.17×10^{-3}	1.53×10^{-3}	+(25, 0, 0)
	ABC	2.5×10^{3}	7.49×10^{3}	9.43×10^{3}	1.07×10^{4}	1.16×10^{4}	8.92×10^{3}	2.37×10^{3}	+(25, 0, 0)
	BA_ABC	0	0	0	0	0	**0**	0	
F2	BA	2.44×10^{-4}	9.29×10^{-4}	1.16×10^{-3}	1.71×10^{-3}	2.71×10^{-3}	1.34×10^{-3}	6.83×10^{-4}	+(24, 1, 0)
	ABC	5.26×10^{3}	8.72×10^{3}	1.06×10^{4}	1.33×10^{4}	1.84×10^{4}	1.09×10^{4}	3.32×10^{3}	+(25, 0, 0)
	BA_ABC	6.02×10^{-7}	9.92×10^{-6}	2.80×10^{-5}	7.01×10^{-5}	3.35×10^{-4}	$\mathbf{7.77 \times 10^{-5}}$	1.10×10^{-4}	
F3	BA	9.63×10^{2}	9.16×10^{3}	2.48×10^{4}	7.85×10^{4}	1.80×10^{5}	4.28×10^{4}	4.45×10^{4}	+(23, 2, 0)
	ABC	9.37×10^{6}	3.62×10^{7}	5.72×10^{7}	8.00×10^{7}	1.35×10^{8}	6.17×10^{7}	3.39×10^{7}	+(25, 0, 0)
	BA_ABC	1.73×10^{2}	8.71×10^{2}	2.11×10^{3}	4.14×10^{3}	9.90×10^{3}	$\mathbf{2.95 \times 10^{3}}$	2.62×10^{3}	
F4	BA	2.81×10^{3}	7.86×10^{3}	1.05×10^{4}	1.59×10^{4}	2.25×10^{4}	1.14×10^{4}	5.16×10^{3}	+(25, 0, 0)
	ABC	6.09×10^{3}	1.00×10^{4}	1.14×10^{4}	1.17×10^{4}	1.66×10^{4}	1.10×10^{4}	2.49×10^{3}	+(25, 0, 0)
	BA_ABC	4.89×10^{-4}	7.95×10^{-4}	2.03×10^{-3}	4.30×10^{-3}	4.15×10^{-2}	$\mathbf{4.72 \times 10^{-3}}$	8.66×10^{-3}	
F5	BA	6.62×10^{1}	5.41×10^{2}	9.78×10^{2}	1.71×10^{3}	5.35×10^{3}	1.20×10^{3}	1.10×10^{3}	+(25, 0, 0)
	ABC	7.66×10^{3}	1.16×10^{4}	1.32×10^{4}	1.39×10^{4}	1.52×10^{4}	1.28×10^{4}	1.76×10^{3}	+(25, 0, 0)
	BA_ABC	8.59×10^{-8}	1.81×10^{-5}	3.11×10^{-4}	1.16×10^{-2}	1.99×10^{0}	$\mathbf{1.47 \times 10^{-1}}$	4.29×10^{-1}	
F6	BA	5.49×10^{-1}	4.25×10^{0}	5.73×10^{0}	9.65×10^{0}	3.01×10^{2}	4.77×10^{1}	8.98×10^{1}	+(22, 3, 0)
	ABC	1.28×10^{8}	7.02×10^{8}	9.11×10^{8}	1.17×10^{9}	3.56×10^{9}	1.15×10^{9}	8.46×10^{8}	+(25, 0, 0)
	BA_ABC	7.31×10^{-8}	2.76×10^{-2}	7.72×10^{-1}	2.92×10^{0}	1.68×10^{1}	$\mathbf{2.29 \times 10^{0}}$	3.98×10^{0}	
F7	BA	8.92×10^{2}	1.11×10^{3}	1.26×10^{3}	1.38×10^{3}	1.78×10^{3}	$\mathbf{1.26 \times 10^{3}}$	2.21×10^{2}	≈(13, 12, 0)
	ABC	1.93×10^{3}	2.30×10^{3}	2.47×10^{3}	2.60×10^{3}	2.88×10^{3}	2.45×10^{3}	2.06×10^{2}	+(25, 0, 0)
	BA_ABC	1.27×10^{3}	1.27×10^{3}	1.27×10^{3}	1.27×10^{3}	1.27×10^{3}	1.27×10^{3}	4.57×10^{-13}	
F8	BA	2.01×10^{1}	2.03×10^{1}	2.03×10^{1}	2.04×10^{1}	2.04×10^{1}	2.03×10^{1}	9.19×10^{-2}	+(21, 4, 0)
	ABC	2.05×10^{1}	2.07×10^{1}	2.08×10^{1}	2.08×10^{1}	2.09×10^{1}	2.07×10^{1}	9.22×10^{-2}	+(25, 0, 0)
	BA_ABC	2.00×10^{1}	2.01×10^{1}	2.02×10^{1}	2.03×10^{1}	2.04×10^{1}	$\mathbf{2.02 \times 10^{1}}$	1.18×10^{-1}	
F9	BA	1.25×10^{1}	2.45×10^{1}	3.34×10^{1}	4.16×10^{1}	5.53×10^{1}	3.32×10^{1}	1.08×10^{1}	+(25, 0, 0)
	ABC	5.21×10^{1}	8.04×10^{1}	8.52×10^{1}	9.62×10^{1}	1.05×10^{2}	8.57×10^{1}	1.32×10^{1}	+(25, 0, 0)
	BA_ABC	0	0	0	0	0	**0**	0	

Table 7. *Cont.*

Function		1st (Best)	7th	13th (Median)	19th	25th (Worst)	Mean	Std	Test (+, −, =)
F10	BA	2.04×10^{1}	2.87×10^{1}	3.41×10^{1}	4.35×10^{1}	5.57×10^{1}	3.58×10^{1}	9.62×10^{0}	+(21, 4, 0)
	ABC	9.76×10^{1}	1.08×10^{2}	1.18×10^{2}	1.25×10^{2}	1.44×10^{2}	1.18×10^{2}	1.30×10^{1}	+(25, 0, 0)
	BA_ABC	7.96×10^{0}	1.39×10^{1}	1.69×10^{1}	2.38×10^{1}	4.58×10^{1}	$\mathbf{1.99 \times 10^{1}}$	9.27×10^{0}	
F11	BA	7.99×10^{0}	8.62×10^{0}	9.01×10^{0}	9.47×10^{0}	1.00×10^{1}	9.10×10^{0}	5.46×10^{-1}	+(25, 0, 0)
	ABC	9.99×10^{0}	1.08×10^{1}	1.15×10^{1}	1.18×10^{1}	1.26×10^{1}	1.14×10^{1}	7.32×10^{-1}	+(25, 0, 0)
	BA_ABC	2.33×10^{0}	3.50×10^{0}	4.74×10^{0}	5.37×10^{0}	6.06×10^{0}	$\mathbf{4.50 \times 10^{0}}$	1.10×10^{0}	
F12	BA	3.87×10^{2}	5.50×10^{2}	8.12×10^{2}	1.09×10^{3}	2.49×10^{3}	9.60×10^{2}	5.47×10^{2}	+(23, 2, 0)
	ABC	4.10×10^{4}	6.65×10^{4}	7.80×10^{4}	9.00×10^{4}	1.13×10^{5}	7.80×10^{4}	2.08×10^{4}	+(25, 0, 0)
	BA_ABC	1.52×10^{1}	8.42×10^{1}	1.66×10^{2}	3.16×10^{2}	6.07×10^{2}	$\mathbf{2.17 \times 10^{2}}$	1.64×10^{2}	
F13	BA	1.64×10^{0}	2.82×10^{1}	3.09×10^{1}	3.54×10^{1}	3.89×10^{1}	3.12×10^{1}	5.65×10^{-1}	+(25, 0, 0)
	ABC	8.03×10^{0}	1.37×10^{1}	1.68×10^{1}	2.11×10^{1}	3.47×10^{1}	1.74×10^{1}	5.90×10^{0}	+(25, 0, 0)
	BA_ABC	2.84×10^{-14}	1.42×10^{-1}	1.93×10^{-1}	2.39×10^{-1}	4.15×10^{-1}	$\mathbf{2.03 \times 10^{-1}}$	1.08×10^{-1}	
F14	BA	3.54×10^{0}	4.00×10^{0}	4.25×10^{0}	4.40×10^{0}	4.54×10^{0}	$\mathbf{4.20 \times 10^{0}}$	2.66×10^{-1}	−(0, 25, 0)
	ABC	4.81×10^{0}	4.83×10^{0}	4.84×10^{0}	4.85×10^{0}	4.86×10^{0}	4.84×10^{0}	1.31×10^{-2}	+(24, 1, 0)
	BA_ABC	4.77×10^{0}	4.80×10^{0}	4.81×10^{0}	4.81×10^{0}	4.84×10^{0}	4.80×10^{0}	1.71×10^{-2}	
F15	BA	1.77×10^{2}	3.01×10^{2}	4.59×10^{2}	5.53×10^{2}	6.74×10^{2}	4.43×10^{2}	1.39×10^{2}	+(25, 0, 0)
	ABC	4.93×10^{2}	7.37×10^{2}	7.65×10^{2}	7.99×10^{2}	8.43×10^{2}	7.58×10^{2}	6.91×10^{1}	+(25, 0, 0)
	BA_ABC	0	0	0	0	0	**0**	0	
F16	BA	1.43×10^{2}	1.55×10^{2}	1.70×10^{2}	1.98×10^{2}	6.60×10^{2}	2.17×10^{2}	1.28×10^{2}	+(24, 1, 0)
	ABC	3.70×10^{2}	3.90×10^{2}	4.20×10^{2}	4.55×10^{2}	5.46×10^{2}	4.31×10^{2}	5.02×10^{2}	+(25, 0, 0)
	BA_ABC	1.00×10^{2}	1.17×10^{2}	1.27×10^{2}	1.35×10^{2}	1.75×10^{2}	$\mathbf{1.29 \times 10^{2}}$	1.76×10^{1}	
F17	BA	1.44×10^{2}	1.73×10^{2}	2.02×10^{2}	2.16×10^{2}	5.11×10^{2}	2.20×10^{2}	9.09×10^{1}	+ (24, 1, 0)
	ABC	2.87×10^{2}	3.86×10^{2}	4.11×10^{2}	4.45×10^{2}	5.40×10^{2}	4.22×10^{2}	5.45×10^{1}	+ (25, 0, 0)
	BA_ABC	1.22×10^{2}	1.43×10^{2}	1.58×10^{2}	1.65×10^{2}	1.73×10^{2}	$\mathbf{1.53 \times 10^{2}}$	1.54×10^{1}	
F18	BA	3.41×10^{2}	8.24×10^{2}	9.31×10^{2}	1.01×10^{3}	1.05×10^{3}	9.11×10^{2}	1.48×10^{2}	+ (24, 1, 0)
	ABC	1.12×10^{3}	1.16×10^{3}	1.19×10^{3}	1.24×10^{3}	1.34×10^{3}	1.20×10^{3}	5.21×10^{1}	+ (25, 0, 0)
	BA_ABC	3.00×10^{2}	3.57×10^{2}	4.44×10^{2}	5.00×10^{2}	9.82×10^{2}	$\mathbf{5.00 \times 10^{2}}$	1.97×10^{2}	
F19	BA	3.24×10^{2}	8.22×10^{2}	9.59×10^{2}	9.86×10^{2}	1.03×10^{3}	8.88×10^{2}	1.51×10^{2}	+ (22, 3, 0)
	ABC	1.08×10^{3}	1.17×10^{3}	1.20×10^{3}	1.23×10^{3}	1.26×10^{3}	1.19×10^{3}	4.50×10^{1}	+ (25, 0, 0)
	BA_ABC	3.56×10^{2}	4.53×10^{2}	5.00×10^{2}	8.00×10^{2}	9.32×10^{2}	$\mathbf{6.15 \times 10^{2}}$	1.96×10^{2}	

Table 7. *Cont.*

Function		1st (Best)	7th	13th (Median)	19th	25th (Worst)	Mean	Std	Test (+, −, =)
F20	BA	3.03×10^2	8.01×10^2	9.67×10^2	9.97×10^2	1.09×10^3	8.36×10^2	2.59×10^2	+ (20, 5, 0)
	ABC	1.02×10^3	1.19×10^3	1.23×10^3	1.26×10^3	1.28×10^3	1.21×10^3	5.94×10^1	+ (25, 0, 0)
	BA_ABC	3.00×10^2	3.56×10^2	5.00×10^2	8.00×10^2	9.19×10^2	$\mathbf{5.28 \times 10^2}$	1.99×10^2	
F21	BA	3.25×10^2	9.22×10^2	1.08×10^3	1.16×10^3	1.25×10^3	9.50×10^2	3.11×10^2	+ (23, 2, 0)
	ABC	1.30×10^3	1.37×10^3	1.39×10^3	1.42×10^3	1.45×10^3	1.39×10^3	4.29×10^1	+ (25, 0, 0)
	BA_ABC	2.00×10^2	4.10×10^2	4.11×10^2	5.00×10^2	9.00×10^2	$\mathbf{4.45 \times 10^2}$	1.57×10^2	
F22	BA	7.67×10^2	7.77×10^2	8.74×10^2	9.08×10^2	9.37×10^2	8.55×10^2	6.24×10^1	+ (25, 0, 0)
	ABC	9.85×10^2	1.07×10^3	1.12×10^3	1.14×10^3	1.30×10^3	1.11×10^3	7.10×10^1	+ (25, 0, 0)
	BA_ABC	3.00×10^2	7.58×10^2	7.65×10^2	7.71×10^2	8.00×10^2	$\mathbf{7.29 \times 10^2}$	1.29×10^2	
F23	BA	5.59×10^2	7.21×10^2	1.04×10^3	1.22×10^3	1.25×10^3	9.67×10^2	2.77×10^2	+ (25, 0, 0)
	ABC	1.28×10^3	1.36×10^3	1.39×10^3	1.41×10^3	1.47×10^3	1.39×10^3	4.28×10^1	+ (25, 0, 0)
	BA_ABC	4.25×10^2	5.48×10^2	5.48×10^2	5.48×10^2	5.59×10^2	$\mathbf{5.24 \times 10^2}$	5.04×10^1	
F24	BA	2.05×10^2	3.96×10^2	4.09×10^2	7.22×10^2	1.24×10^3	5.79×10^2	3.10×10^2	+ (25, 0, 0)
	ABC	1.09×10^3	1.28×10^3	1.32×10^3	1.36×10^3	1.38×10^3	1.30×10^3	7.27×10^1	+ (25, 0, 0)
	BA_ABC	2.00×10^2	2.00×10^2	2.00×10^2	2.00×10^2	2.00×10^2	$\mathbf{2.00 \times 10^2}$	1.53×10^{-12}	
F25	BA	2.06×10^2	3.95×10^2	4.15×10^2	5.63×10^2	1.15×10^3	$\mathbf{5.34 \times 10^2}$	2.34×10^2	− (3, 22, 0)
	ABC	1.34×10^3	1.41×10^3	1.42×10^3	1.44×10^3	1.48×10^3	1.42×10^3	2.80×10^1	+ (25, 0, 0)
	BA_ABC	6.01×10^2	6.23×10^2	8.20×10^2	8.24×10^2	8.31×10^2	7.41×10^2	1.03×10^2	
	BA	ABC	BA_ABC						
Mean Rank	1.96	2.92	**1.12**						

Table 8. Comparison results of BA_ABC and other algorithms (D = 10).

Function	Algorithms											
	BA_ABC	**SBAIS**	**MBADE**	**NBA**	**iBA**	**ILSSIWBA**	**GBA**	**TBA**	**PBA**	**LBA**	**EBA**	**RBA**
F1	**0**	2.43×10^{-5}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	8.09×10^{3}
F2	7.77×10^{-5}	1.90×10^{-7}	**0**	4.38×10^{-7}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	7.88×10^{3}
F3	2.95×10^{3}	1.06×10^{2}	1.33×10^{3}	3.46×10^{3}	$\mathbf{2.48 \times 10^{1}}$	1.67×10^{3}	2.41×10^{2}	1.43×10^{2}	2.74×10^{3}	5.11×10^{2}	2.21×10^{2}	1.66×10^{7}
F4	4.72×10^{-3}	2.57×10^{2}	**0**	2.54×10^{3}	2.99×10^{3}	1.92×10^{-4}	3.81×10^{2}	1.54×10^{2}	9.79×10^{3}	5.88×10^{3}	1.77×10^{3}	1.16×10^{4}
F5	1.47×10^{-1}	5.25×10^{0}	**0**	8.81×10^{-2}	1.47×10^{-1}	**0**	2.92×10^{1}	6.26×10^{0}	1.22×10^{2}	2.56×10^{1}	5.74×10^{0}	3.60×10^{3}
F6	2.29×10^{0}	9.94×10^{-1}	**0**	7.97×10^{-1}	1.69×10^{-3}	5.64×10^{0}	4.96×10^{-2}	2.91×10^{-2}	4.36×10^{2}	2.20×10^{-2}	1.44×10^{0}	4.48×10^{8}
F7	1.27×10^{3}	$\mathbf{6.73 \times 10^{0}}$	4.85×10^{2}	9.32×10^{0}	4.59×10^{1}	1.26×10^{3}	7.04×10^{1}	5.36×10^{1}	8.74×10^{1}	8.98×10^{1}	5.42×10^{1}	4.26×10^{2}
F8	2.02×10^{1}	$\mathbf{2.15 \times 10^{0}}$	2.03×10^{1}	2.01×10^{1}	2.00×10^{1}	2.01×10^{1}	2.01×10^{1}	2.01×10^{1}	2.02×10^{1}	2.02×10^{1}	2.02×10^{1}	2.02×10^{1}
F9	**0**	2.17×10^{0}	**0**	3.82×10^{1}	2.48×10^{1}	2.09×10^{1}	3.94×10^{1}	3.59×10^{1}	7.35×10^{1}	4.79×10^{1}	5.96×10^{1}	7.87×10^{1}
F10	1.99×10^{1}	$\mathbf{8.17 \times 10^{-1}}$	1.05×10^{1}	6.64×10^{1}	4.08×10^{1}	1.97×10^{1}	5.53×10^{1}	4.58×10^{1}	9.17×10^{1}	5.22×10^{1}	7.85×10^{1}	9.88×10^{1}
F11	4.50×10^{0}	1.21×10^{0}	2.61×10^{0}	7.07×10^{0}	3.12×10^{0}	5.54×10^{0}	$\mathbf{1.17 \times 10^{0}}$	1.30×10^{0}	7.94×10^{0}	1.44×10^{0}	1.46×10^{0}	8.66×10^{0}
F12	2.17×10^{2}	1.75×10^{2}	3.15×10^{2}	3.69×10^{3}	$\mathbf{2.20 \times 10^{-2}}$	7.29×10^{1}	7.54×10^{1}	4.25×10^{1}	2.82×10^{1}	4.18×10^{1}	7.40×10^{1}	1.02×10^{5}
F13	$\mathbf{2.03 \times 10^{-1}}$	1.52×10^{0}	3.42×10^{-1}	2.61×10^{0}	2.56×10^{0}	1.03×10^{0}	1.68×10^{0}	1.34×10^{0}	1.43×10^{0}	1.03×10^{0}	1.99×10^{0}	4.65×10^{0}
F14	4.80×10^{0}	$\mathbf{2.18 \times 10^{0}}$	4.62×10^{0}	3.34×10^{0}	3.92×10^{0}	3.88×10^{0}	3.97×10^{0}	3.94×10^{0}	4.03×10^{0}	3.99×10^{0}	4.24×10^{0}	4.13×10^{0}
F15	**0**	2.14×10^{2}	2.36×10^{2}	4.99×10^{2}	2.83×10^{2}	1.25×10^{2}	6.52×10^{2}	6.23×10^{2}	6.62×10^{2}	6.66×10^{2}	7.31×10^{2}	7.02×10^{2}
F16	1.29×10^{2}	$\mathbf{7.37 \times 10^{1}}$	1.14×10^{2}	2.43×10^{2}	1.93×10^{2}	1.39×10^{2}	4.58×10^{2}	4.99×10^{2}	4.80×10^{2}	4.61×10^{2}	4.38×10^{2}	4.67×10^{2}
F17	1.53×10^{2}	$\mathbf{1.07 \times 10^{2}}$	1.14×10^{2}	2.89×10^{2}	1.81×10^{2}	1.39×10^{2}	2.03×10^{2}	2.31×10^{2}	4.73×10^{2}	3.39×10^{2}	3.76×10^{2}	5.61×10^{2}
F18	5.00×10^{2}	6.49×10^{2}	$\mathbf{3.88 \times 10^{2}}$	9.81×10^{2}	9.53×10^{2}	7.48×10^{2}	1.15×10^{3}	1.04×10^{3}	1.08×10^{3}	1.11×10^{3}	1.07×10^{3}	1.16×10^{3}
F19	6.15×10^{2}	4.10×10^{2}	$\mathbf{3.97 \times 10^{2}}$	1.02×10^{3}	8.97×10^{2}	7.89×10^{2}	9.77×10^{2}	8.94×10^{2}	1.00×10^{3}	1.03×10^{3}	1.09×10^{3}	1.09×10^{3}
F20	5.28×10^{2}	$\mathbf{3.03 \times 10^{2}}$	5.14×10^{2}	1.07×10^{3}	8.05×10^{2}	6.97×10^{2}	8.80×10^{2}	9.95×10^{2}	1.01×10^{3}	1.03×10^{3}	1.08×10^{3}	1.14×10^{3}
F21	4.45×10^{2}	4.28×10^{2}	5.00×10^{2}	1.09×10^{3}	1.23×10^{3}	$\mathbf{3.20 \times 10^{2}}$	1.29×10^{3}	1.23×10^{3}	1.29×10^{3}	1.27×10^{3}	1.29×10^{3}	1.41×10^{3}
F22	7.29×10^{2}	$\mathbf{4.22 \times 10^{2}}$	7.29×10^{2}	9.25×10^{2}	8.54×10^{2}	7.84×10^{2}	8.26×10^{2}	8.55×10^{2}	9.10×10^{2}	8.24×10^{2}	9.82×10^{2}	9.66×10^{2}
F23	5.24×10^{2}	$\mathbf{4.48 \times 10^{2}}$	5.60×10^{2}	1.20×10^{3}	1.23×10^{3}	6.93×10^{2}	1.42×10^{3}	1.40×10^{3}	1.39×10^{3}	1.40×10^{3}	1.42×10^{3}	1.47×10^{3}
F24	$\mathbf{2.00 \times 10^{2}}$	$\mathbf{2.00 \times 10^{2}}$	$\mathbf{2.00 \times 10^{2}}$	1.04×10^{3}	9.49×10^{2}	$\mathbf{2.00 \times 10^{2}}$	1.34×10^{3}	7.43×10^{2}	9.76×10^{2}	9.95×10^{2}	7.60×10^{2}	1.43×10^{3}
F25	7.41×10^{2}	$\mathbf{3.11 \times 10^{2}}$	4.65×10^{2}	1.07×10^{3}	9.46×10^{2}	1.64×10^{3}	8.14×10^{2}	5.72×10^{2}	9.82×10^{2}	9.36×10^{2}	5.43×10^{2}	1.12×10^{3}
Wilcoxon Test												
+		7	6	18	15	12	16	15	18	16	15	22
-		17	15	6	8	11	8	9	5	7	8	2
=		1	4	1	2	2	1	1	2	2	2	1
p value		0.026(-)	0.085(≈)	0.001(+)	0.042(+)	0.248(≈)	0.037(+)	0.092(≈)	0.004(+)	0.026(+)	0.052(≈)	1.15×10^{-4}(+)
Friedman Test												
Mean Rank	5.22	**2.78**	4.16	7.44	5.24	4.78	6.92	5.74	8.66	7.58	8.06	11.42

3.3. Performance of BA_ABC on CEC2010 Large-Scale Test Functions

In this section, the performance of BA_ABC on large-scale test functions was evaluated. For this purpose, CEC2010 benchmark functions [35] were used. The properties of the functions were given in Table 9. According to CEC2010 rules, the maximum number of evaluations (FEs) was taken as 3.00×10^6 and the number of general cycles as 25. Comparative results of BA_ABC with BA and ABC were given in Table 10 for 3.00×10^6 FEs.

Table 9. CEC2010 benchmark functions.

Function	Name	Modality	Range
Separable functions			
F1	Shifted Elliptic Function	Unimodal	[−100, 100]
F2	Shifted Rastrigin's Function	Multimodal	[−5, 5]
F3	Shifted Ackley's Function	Multimodal	[−32, 32]
Single-group m-nonseparable functions			
F4	Single-group Shifted and m-rotated Elliptic Function	Unimodal	[−100, 100]
F5	Single-group Shifted and m-rotated Rastrigin's Function	Multimodal	[−5, 5]
F6	Single-group Shifted and m-rotated Ackley's Function	Multimodal	[−32, 32]
F7	Single-group Shifted m-dimensional Schwefel's Function 1.2	Unimodal	[−100, 100]
F8	Single-group Shifted m-dimensional Rosenbrock's Function	Multimodal	[−100, 100]
D/2m groups m-nonseparable functions			
F9	D/2m -group Shifted and m-rotated Elliptic Function	Unimodal	[−100, 100]
F10	D/2m -group Shifted and m-rotated Rastrigin's Function	Multimodal	[−5, 5]
F11	D/2m -group Shifted and m-rotated Ackley's Function	Multimodal	[−32, 32]
F12	D/2m -group Shifted m-dimensional Schwefel's Problem 1.2	Unimodal	[−100, 100]
F13	D/2m -group Shifted m-dimensional Rosenbrock's Function	Multimodal	[−100, 100]
D/m groups m-nonseparable functions			
F14	D/m -group Shifted and m-rotated Elliptic Function	Unimodal	[−100, 100]
F15	D/m -group Shifted and m-rotated Rastrigin's Function	Multimodal	[−5, 5]
F16	D/m -group Shifted and m-rotated Ackley's Function	Multimodal	[−32, 32]
F17	D/m -group Shifted m-dimensional Schwefel's Problem 1.2	Unimodal	[−100, 100]
F18	D/m -group Shifted m-dimensional Rosenbrock's Function	Multimodal	[−100, 100]
Nonseparable functions			
F19	Shifted Schwefel's Problem 1.2	Unimodal	[−100, 100]
F20	Shifted Rosenbrock's Function	Multimodal	[−100, 100]

According to Table 10, BA_ABC produced the best mean value in 12 functions, BA in one function, and ABC in the remaining seven functions. When Table 10 test column was examined in general, it was seen that the BA_ABC algorithm was better than the BA algorithm in a total of 19 functions, and there was a significant difference between them. No significant difference was found between the BA and BA_ABC algorithms in function number five. When the test results of BA_ABC and ABC were examined, it was found that BA_ABC had significantly worse results in six functions, equal result in one, and better results in 13. According to the results of the Friedman test given at the bottom of Table 10, the BA_ABC algorithm ranked first with a mean rank value of 1.40, the ABC algorithm ranked second with a mean rank value of 1.85, and the BA algorithm ranked third with a mean rank value of 2.75.

Table 10. CEC2010 test results of BA, ABC, and BA_ABC (D = 1000).

	F1			F2			F3		
	BA	ABC	BA_ABC	BA	ABC	BA_ABC	BA	ABC	BA_ABC
Best	7.89×10^{9}	2.34×10^{-12}	1.64×10^{-11}	1.35×10^{4}	3.12×10^{1}	3.89×10^{1}	1.98×10^{1}	5.14×10^{-7}	1.42×10^{-6}
Median	2.74×10^{10}	8.80×10^{-12}	1.72×10^{-10}	1.44×10^{4}	3.97×10^{1}	4.85×10^{1}	2.04×10^{1}	9.57×10^{-7}	3.34×10^{-6}
Worst	3.23×10^{10}	3.29×10^{-11}	6.05×10^{-8}	2.46×10^{4}	4.72×10^{1}	5.30×10^{1}	2.14×10^{1}	1.89×10^{-6}	6.43×10^{-6}
Mean	2.63×10^{10}	$\mathbf{1.10 \times 10^{-11}}$	4.81×10^{-9}	1.73×10^{4}	$\mathbf{3.95 \times 10^{1}}$	4.74×10^{1}	2.06×10^{1}	$\mathbf{1.00 \times 10^{-6}}$	3.54×10^{-6}
Std	4.93×10^{9}	7.82×10^{-12}	1.46×10^{-8}	4.47×10^{3}	3.88×10^{0}	3.58×10^{0}	7.83×10^{-1}	3.69×10^{-7}	1.21×10^{-6}
Test(+, −, =)	+(25, 0, 0)	−(0, 25, 0)		+(25, 0, 0)	−(4, 21, 0)		+(25, 0, 0)	−(0, 25, 0)	
	F4			F5			F6		
	BA	ABC	BA_ABC	BA	ABC	BA_ABC	BA	ABC	BA_ABC
Best	2.91×10^{11}	3.58×10^{13}	7.80×10^{9}	2.83×10^{8}	4.25×10^{8}	2.78×10^{8}	1.94×10^{7}	1.97×10^{7}	1.88×10^{7}
Median	5.14×10^{11}	4.57×10^{13}	1.82×10^{10}	3.76×10^{8}	5.90×10^{8}	3.71×10^{8}	1.97×10^{7}	2.00×10^{7}	1.94×10^{7}
Worst	8.87×10^{11}	5.86×10^{13}	4.07×10^{10}	4.90×10^{8}	7.23×10^{8}	5.43×10^{8}	2.08×10^{7}	2.00×10^{7}	1.98×10^{7}
Mean	5.18×10^{11}	4.63×10^{13}	$\mathbf{2.15 \times 10^{10}}$	$\mathbf{3.74 \times 10^{8}}$	5.81×10^{8}	3.81×10^{8}	2.01×10^{7}	1.99×10^{7}	$\mathbf{1.93 \times 10^{7}}$
Std	1.07×10^{11}	6.46×10^{12}	9.19×10^{9}	6.04×10^{7}	6.74×10^{7}	7.12×10^{7}	5.56×10^{5}	8.69×10^{4}	2.53×10^{5}
Test(+, −, =)	+(25, 0, 0)	+(25, 0, 0)		≈(14,11,0)	+(25, 0, 0)		+(24, 1, 0)	+(25, 0, 0)	
	F7			F8			F9		
	BA	ABC	BA_ABC	BA	ABC	BA_ABC	BA	ABC	BA_ABC
Best	5.73×10^{6}	2.97×10^{10}	6.09×10^{4}	5.79×10^{6}	3.75×10^{5}	5.93×10^{4}	8.82×10^{9}	5.83×10^{8}	2.63×10^{6}
Median	6.05×10^{6}	4.02×10^{10}	1.64×10^{5}	6.30×10^{6}	3.33×10^{6}	3.06×10^{5}	2.99×10^{10}	6.58×10^{8}	3.52×10^{6}
Worst	5.41×10^{8}	4.86×10^{10}	2.97×10^{5}	2.21×10^{8}	2.03×10^{7}	7.24×10^{7}	4.13×10^{10}	7.06×10^{8}	7.11×10^{6}
Mean	4.11×10^{7}	3.97×10^{10}	$\mathbf{1.74 \times 10^{5}}$	3.73×10^{7}	$\mathbf{6.34 \times 10^{6}}$	1.06×10^{7}	2.96×10^{10}	6.57×10^{8}	$\mathbf{3.78 \times 10^{6}}$
Std	1.25×10^{8}	5.64×10^{9}	5.32×10^{4}	7.21×10^{7}	5.49×10^{6}	1.93×10^{7}	6.21×10^{9}	3.33×10^{7}	1.05×10^{6}
Test(+, −, =)	+(25, 0, 0)	+(25, 0, 0)		+(19, 6, 0)	≈(16, 9, 0)		+(25, 0, 0)	+(25, 0, 0)	
	F10			F11			F12		
	BA	ABC	BA_ABC	BA	ABC	BA_ABC	BA	ABC	BA_ABC
Best	1.35×10^{4}	6.79×10^{3}	4.21×10^{3}	2.15×10^{2}	2.01×10^{2}	1.95×10^{2}	2.04×10^{6}	6.12×10^{5}	2.19×10^{-1}
Median	1.47×10^{4}	7.27×10^{3}	4.78×10^{3}	2.34×10^{2}	2.01×10^{2}	1.96×10^{2}	3.53×10^{6}	6.62×10^{5}	2.65×10^{-1}
Worst	2.45×10^{4}	7.54×10^{3}	7.48×10^{3}	2.35×10^{2}	2.02×10^{2}	2.00×10^{2}	4.01×10^{6}	6.91×10^{5}	3.16×10^{-1}
Mean	1.76×10^{4}	7.23×10^{3}	$\mathbf{4.95 \times 10^{3}}$	2.27×10^{2}	2.01×10^{2}	$\mathbf{1.96 \times 10^{2}}$	3.46×10^{6}	6.65×10^{5}	$\mathbf{2.74 \times 10^{-1}}$
Std	4.61×10^{3}	1.92×10^{2}	7.45×10^{2}	8.74×10^{0}	1.81×10^{-1}	1.80×10^{0}	3.68×10^{5}	1.66×10^{4}	2.37×10^{-2}
Test(+, −, =)	+(25, 0, 0)	+(25, 0, 0)		+(25, 0, 0)	+(25, 0, 0)		+(25, 0, 0)	+(25, 0, 0)	
	F13			F14			F15		
	BA	ABC	BA_ABC	BA	ABC	BA_ABC	BA	ABC	BA_ABC
Best	4.8×10^{9}	3.90×10^{2}	4.27×10^{2}	9.23×10^{9}	1.32×10^{9}	7.48×10^{6}	9.42×10^{3}	1.39×10^{4}	8.74×10^{3}
Median	1.02×10^{11}	5.11×10^{2}	8.36×10^{2}	3.07×10^{10}	1.44×10^{9}	9.64×10^{6}	1.49×10^{4}	1.46×10^{4}	9.38×10^{3}
Worst	1.33×10^{11}	1.16×10^{3}	2.97×10^{3}	3.94×10^{10}	1.54×10^{9}	1.18×10^{7}	2.49×10^{4}	1.49×10^{4}	1.06×10^{4}
Mean	1.00×10^{11}	$\mathbf{5.48 \times 10^{2}}$	1.11×10^{3}	3.10×10^{10}	1.44×10^{9}	$\mathbf{9.59 \times 10^{6}}$	1.90×10^{4}	1.45×10^{4}	$\mathbf{9.45 \times 10^{3}}$
Std	2.53×10^{10}	1.75×10^{2}	6.47×10^{2}	5.93×10^{9}	6.88×10^{7}	1.07×10^{6}	5.35×10^{3}	2.70×10^{2}	4.76×10^{2}
Test(+, −, =)	+(25, 0, 0)	−(2, 23, 0)		+(25, 0, 0)	+(25, 0, 0)		+(24, 1, 0)	+(25, 0, 0)	
	F16			F17			F18		
	BA	ABC	BA_ABC	BA	ABC	BA_ABC	BA	ABC	BA_ABC
Best	3.90×10^{2}	4.03×10^{2}	3.90×10^{2}	2.69×10^{6}	1.23×10^{6}	1.15×10^{0}	8.15×10^{10}	1.36×10^{3}	8.36×10^{2}
Median	3.96×10^{2}	4.03×10^{2}	3.91×10^{2}	4.00×10^{6}	1.30×10^{6}	1.34×10^{0}	8.79×10^{11}	5.28×10^{3}	1.05×10^{4}
Worst	4.29×10^{2}	4.04×10^{2}	3.99×10^{2}	5.44×10^{6}	1.39×10^{6}	1.63×10^{0}	1.01×10^{12}	1.54×10^{4}	3.41×10^{4}
Mean	4.09×10^{2}	4.03×10^{2}	$\mathbf{3.92 \times 10^{2}}$	4.08×10^{6}	1.33×10^{6}	$\mathbf{1.33 \times 10^{0}}$	8.58×10^{11}	$\mathbf{6.31} \times 10^{3}$	1.38×10^{4}
Std	1.61×10^{1}	2.64×10^{-1}	2.20×10^{0}	5.57×10^{5}	4.04×10^{4}	1.27×10^{-1}	1.77×10^{11}	4.12×10^{3}	1.07×10^{4}
Test(+, −, =)	+(23, 2, 0)	+(25, 0, 0)		(25, 0, 0)	+(25, 0, 0)		+(25, 0, 0)	−(7, 18, 0)	
	F19			F20			Mean Rank		
	BA	ABC	BA_ABC	BA	ABC	BA_ABC	BA	ABC	BA_ABC
Best	4.06×10^{6}	6.87×10^{6}	3.51×10^{4}	1.50×10^{11}	9.44×10^{0}	2.46×10^{2}			
Median	5.70×10^{6}	8.03×10^{6}	4.71×10^{4}	1.04×10^{12}	2.47×10^{1}	6.60×10^{2}			
Worst	7.51×10^{6}	8.38×10^{6}	6.60×10^{4}	1.22×10^{12}	7.11×10^{1}	1.18×10^{3}	2.75	1.85	**1.40**
Mean	5.86×10^{6}	7.94×10^{6}	$\mathbf{4.72 \times 10^{4}}$	1.01×10^{12}	$\mathbf{2.80 \times 10^{1}}$	6.77×10^{2}			
Std	7.57×10^{5}	3.73×10^{5}	8.31×10^{3}	1.98×10^{11}	1.45×10^{1}	2.26×10^{2}			
Test(+, −, =)	+(25, 0, 0)	+(25, 0, 0)		+(25, 0, 0)	−(0, 25, 0)				

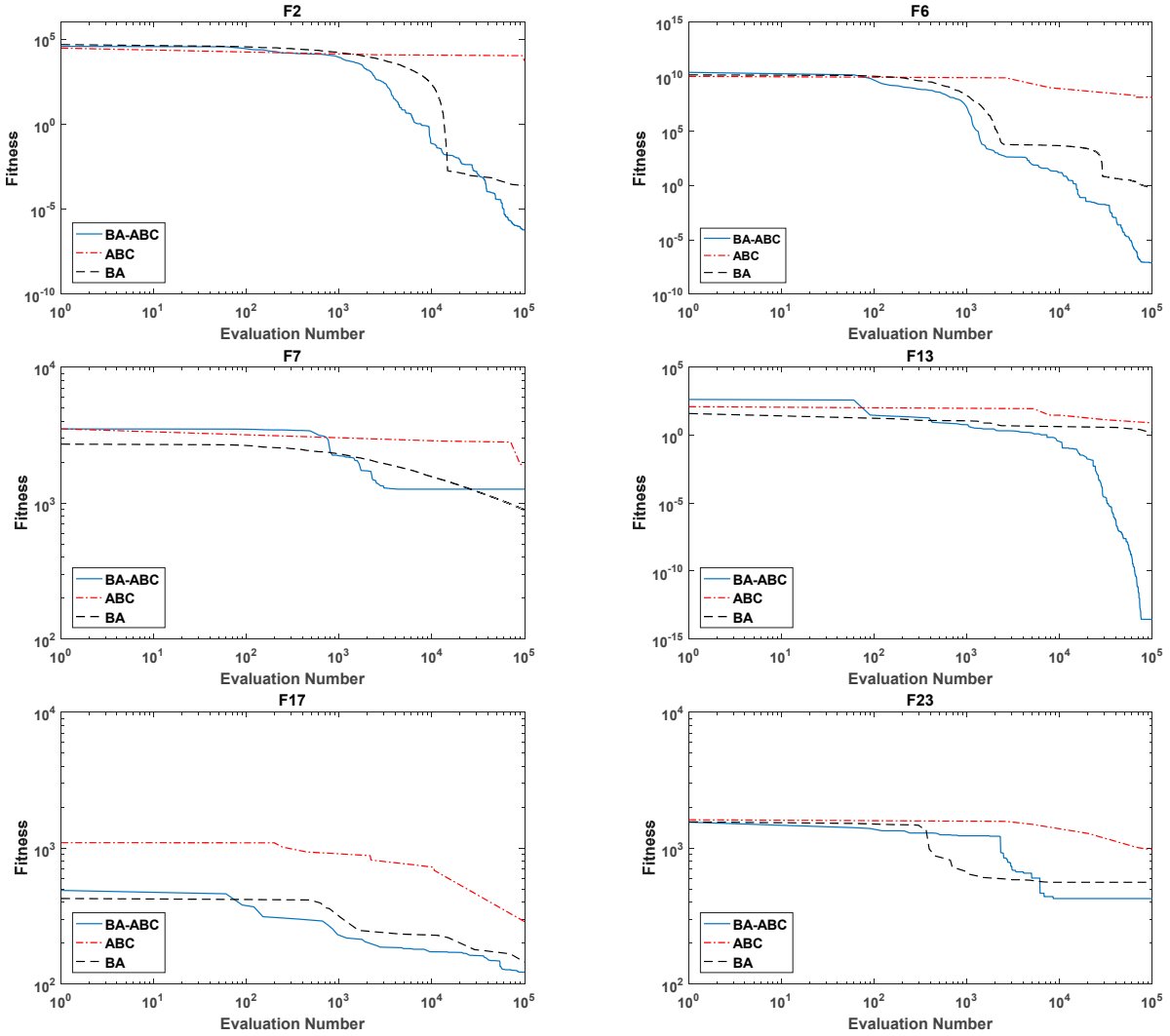
For CEC2010 test functions; the BA_ABC algorithm was compared with recently proposed algorithms such as Adaptive Hybrid Differential Evolution with circular sliding window (AHDE) [39], Quantum-behaved Particle Swarm Optimization with Random Selection (RSQPSO) [40], Improved Sine Cosine Algorithm (ISCA) [41], Micro Differential Evolution with local Directional Search (μDSDE) [42], and Adaptive Enhanced Unidimensional Search (aEUS) [43]. The mean values of the algorithms are given in Table 11. According to the results, BA_ABC produced the best mean values in five functions, the AHDE algorithm in one, the RSQPSO algorithm in four, the ISCA algorithm in five, and the aEUS algorithm in five. The μDSDE algorithm, on the other hand, did not produce the best mean value in any function. When the Wilcoxon test results in Table 11 were examined, it was concluded that

there was no significant difference between the BA_ABC and aEUS algorithms; however, BA_ABC was better than other algorithms and that there was a significant difference between them. When Friedman test results were examined, it was found that BA_ABC ranked first among the algorithms with a mean rank value of 2.65 and the aEUS algorithm ranked second with a mean rank value of 2.83.

Table 11. Comparison results of BA_ABC and other algorithms (D = 1000).

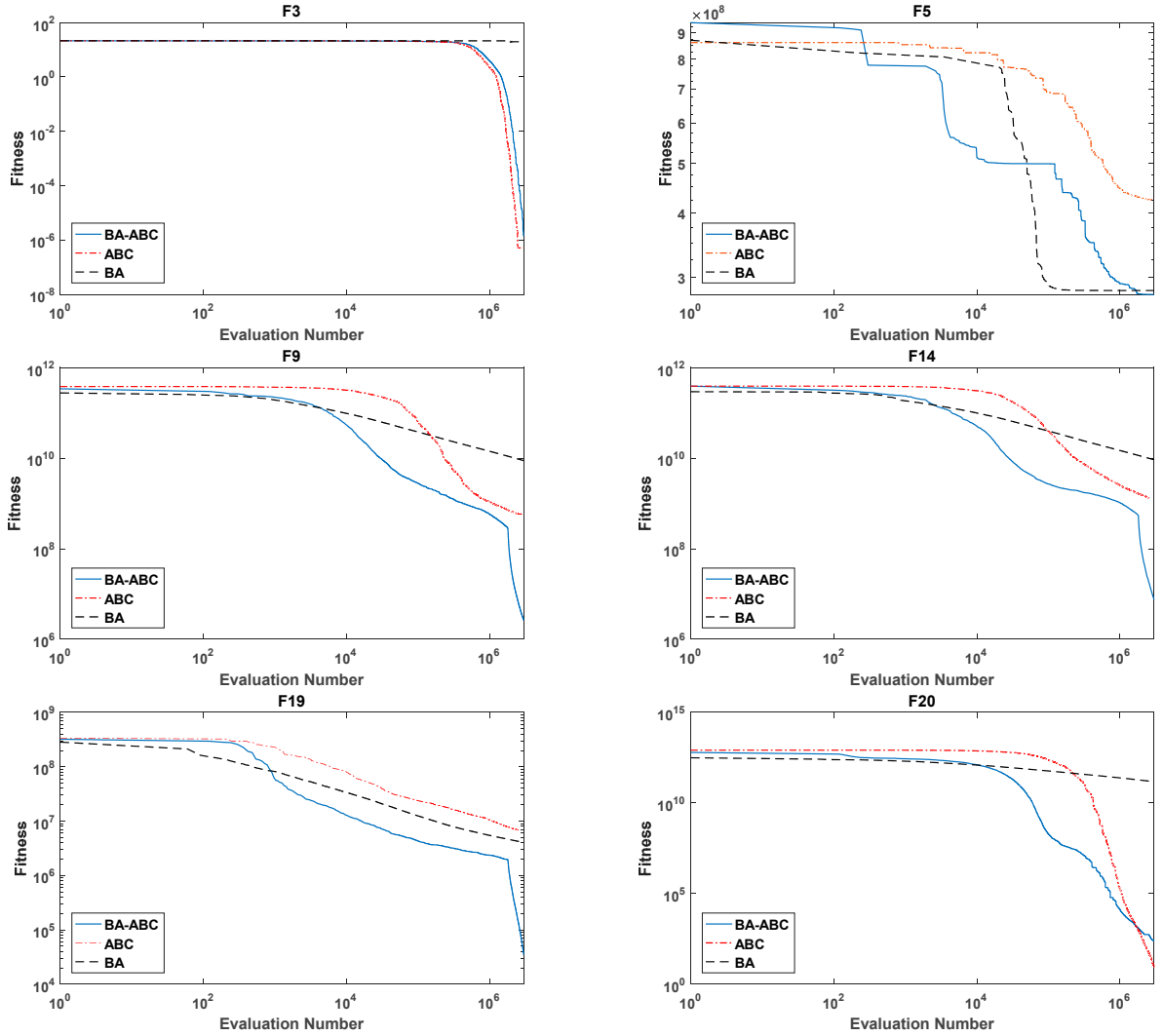
Function	Algorithms					
	BA_ABC	**AHDE**	**RSQPSO**	**ISCA**	**μDSDE**	**aEUS**
F1	4.81×10^{-9}	1.81×10^{-20}	$\mathbf{8.66 \times 10^{-32}}$	1.55×10^{11}	2.05×10^{4}	8.32×10^{-24}
F2	4.74×10^{1}	3.57×10^{-6}	1.18×10^{1}	1.64×10^{0}	1.47×10^{2}	**0**
F3	3.54×10^{-6}	3.73×10^{-9}	$\mathbf{1.27 \times 10^{-14}}$	3.10×10^{-1}	2.45×10^{0}	1.90×10^{-12}
F4	$\mathbf{2.15 \times 10^{10}}$	2.33×10^{12}	7.09×10^{11}	1.37×10^{12}	3.16×10^{11}	2.84×10^{11}
F5	3.81×10^{8}	3.88×10^{8}	5.39×10^{8}	3.27×10^{8}	1.11×10^{8}	$\mathbf{7.18 \times 10^{7}}$
F6	1.93×10^{7}	1.97×10^{7}	1.94×10^{7}	$\mathbf{1.29 \times 10^{5}}$	1.50×10^{7}	1.99×10^{7}
F7	1.74×10^{5}	2.01×10^{6}	$\mathbf{7.50 \times 10^{0}}$	3.71×10^{10}	5.53×10^{5}	2.73×10^{3}
F8	1.06×10^{7}	$\mathbf{9.09 \times 10^{5}}$	8.78×10^{7}	1.59×10^{13}	4.08×10^{7}	1.29×10^{8}
F9	$\mathbf{3.78 \times 10^{6}}$	7.06×10^{7}	1.75×10^{7}	1.66×10^{11}	7.16×10^{8}	7.57×10^{6}
F10	4.95×10^{3}	5.84×10^{3}	5.33×10^{3}	$\mathbf{2.96 \times 10^{3}}$	4.63×10^{3}	7.15×10^{3}
F11	1.96×10^{2}	2.01×10^{2}	1.98×10^{2}	$\mathbf{1.89 \times 10^{2}}$	1.98×10^{2}	1.99×10^{2}
F12	$\mathbf{2.74 \times 10^{-1}}$	3.99×10^{4}	1.19×10^{2}	1.33×10^{6}	2.59×10^{5}	3.28×10^{-1}
F13	1.11×10^{3}	1.64×10^{3}	$\mathbf{1.06 \times 10^{3}}$	6.30×10^{10}	1.10×10^{3}	1.09×10^{3}
F14	$\mathbf{9.59 \times 10^{6}}$	1.71×10^{8}	5.58×10^{7}	2.29×10^{8}	1.85×10^{9}	1.71×10^{7}
F15	9.45×10^{3}	1.09×10^{4}	1.34×10^{4}	$\mathbf{1.64 \times 10^{3}}$	8.79×10^{3}	1.42×10^{4}
F16	3.92×10^{2}	3.98×10^{2}	9.37×10^{2}	$\mathbf{3.18 \times 10^{2}}$	3.90×10^{2}	3.98×10^{2}
F17	$\mathbf{1.33 \times 10^{0}}$	2.00×10^{5}	2.31×10^{3}	3.27×10^{6}	1.03×10^{6}	3.98×10^{0}
F18	1.38×10^{4}	4.77×10^{3}	$\mathbf{2.28 \times 10^{3}}$	1.41×10^{4}	3.01×10^{4}	2.97×10^{3}
F19	4.72×10^{4}	5.84×10^{5}	3.08×10^{6}	5.90×10^{6}	4.56×10^{6}	$\mathbf{9.44 \times 10^{3}}$
F20	6.77×10^{2}	1.21×10^{3}	1.13×10^{3}	1.59×10^{11}	5.29×10^{3}	$\mathbf{4.51 \times 10^{2}}$
Wilcoxon Test						
+		15	14	13	14	11
−		5	6	7	6	9
=		0	0	0	0	0
p		0.009(+)	0.012(+)	0.025(+)	0.038(+)	0.455(≈)
Friedman Test						
Mean Rank	**2.65**	4.03	3.23	4.25	4.03	2.83

Figure 1 shows the convergence graphs of BA, ABC, and BA_ABC algorithms for every six functions randomly selected from CEC2005 and CEC2010. When the graphs of the CEC2005 functions were examined, it was seen that the BA algorithm converged faster in the graph of function F7, and the BA_ABC algorithm converged faster in the other functions and produced a better solution. When the graphs of CEC2010 functions were examined, it was seen that the ABC algorithm produced the best solution in F3 and F20 functions. The convergence rate and characteristics of ABC and BA_ABC algorithms were similar in function F3. It can be said that BA produced the best solution in function F5, and converged faster than others, but was not able to produce new solutions towards the end of the iteration. In the remaining functions, BA_ABC converged faster and produced better solutions.



(A)

Figure 1. *Cont.*



(**B**)

Figure 1. Convergence Graphics: (**A**) CEC2005 test functions; (**B**) CEC2010 test functions.

3.4. Performance of BA_ABC in Classical Engineering Design Problems

Engineering Design can be defined as the process of meeting the requirements that are needed to create a product. Today, meta-heuristic algorithms emerge as an alternative to traditional optimization methods used in this process. The method we proposed in this section was applied to three engineering optimization problems, including pressure vessel design problem, tension/compression spring design problem, and gear train design problem, which are frequently used in literature, and its performance was examined.

3.4.1. Pressure Vessel Design Problem

The pressure vessel design problem is a classic engineering design problem that aims to minimize pressure vessel welding, manufacturing, and material costs [44,45]. It is a problem with four decision variables and four constraints (thickness of shell T_s, the thickness of head T_h, inner radius R, length of the cylindrical section of the vessel L). The schematic representation of the problem is given in Figure 2.

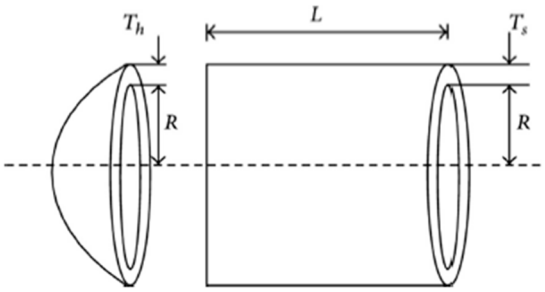


Figure 2. Structure of the pressure vessel design problem [46].

The mathematical model of the problem can be summarized as in Equation (13).

$$\begin{gathered} X = (Ts, Th, R, L) = (x_1, x_2, x_3, x_4) \\ \text{Minimize } f(X) = 0.6224\, x_1 x_3 x_4 + 1.7781\, x_2 x_3^2 + 3.1661\, x_2^1 x_4 + 19.84\, x_1^2 x_3 \\ \text{Subject to } g_1(X) = -x_1 + 0.0193\, x_3 \le 0 \\ g_2(X) = -x_2 + 0.0095\, x_3 \le 0 \\ g_3(X) = -\lambda x_3^2 x_4 - 4/3\, \lambda x_3^3 + 1{,}296{,}000 \le 0 \\ g_4(X) = x_4 - 240 \le 0 \\ 1 \times 0.0625 \le x_1, x_2 \le 99 \times 0.0625,\ 10 \le x_3, x_4 \le 200 \end{gathered} \tag{13}$$

The result obtained from the proposed method is given in Table 12, with the general cycle number taken as 30 and FES value taken as 30,000. The table shows the values of decision variables and constraint values for the best fitness value. The statistical information obtained after 30 general cycles are also shown in the table. Table 13 shows the comparison results between the ten algorithms selected from the literature and the BA_ABC algorithm. The BA_ABC algorithm has been shown to produce an acceptable result similar to the literature.

Table 12. Optimal solutions of pressure vessel design problem by BA_ABC.

	x_1	x_2	x_3	x_4	$f(x)$
Decision variables	0.7781878	0.3846586	40.32058	199.9867	5885.3715
	g_1	g_2	g_3	g_4	
Constraint values	-6.0599×10^{-7}	-2.6680×10^{-7}	−0.4149	−40.0133	
	Best	Median	Worst	Mean	Std.
	5885.3715	6008.1344	6248.7054	6017.5957	91.5592

Table 13. Comparison of BA_ABC with other algorithms for pressure vessel design optimization problem.

Algorithms	Worst	Mean	Best	Std
EBA [31]	6370.77	6173.67	6059.71	142.33
IAPSO [47]	6090.53	6068.75	6059.71	14.01
CGWO [48]	6188.11	**5783.58**	**5034.18**	254.50
WCA [49]	6590.21	6198.61	5885.33	213.04
MBA [50]	6392.50	6200.64	5889.32	160.34
PSO_GA [51]	5885.48	5885.38	5885.33	0.05
ABC [52]	5895.12	5887.55	5885.40	2.74
BA [53]	6318.95	6179.13	6059.71	137.22
CSDE [54]	1.52×10^{22}	6261.41	6059.71	$\mathbf{1.44 \times 10^{-8}}$
TLNNA [55]	6114.95	5935.42	5885.33	66.28
BA_ABC	6248.70	6017.59	5885.37	91.55

3.4.2. Tension/Compression Spring Design Problem

This design problem was studied by Belegundu and Arora [56] and it is an optimization problem whose main purpose is to reduce the weight of spring with three decision variables such as wire diameter (d), mean coil diameter (D) and the number of active coils (N). The schematic representation of the problem is given in Figure 3.

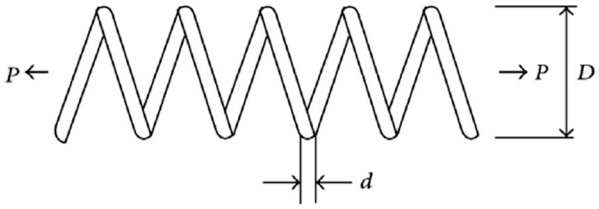


Figure 3. The structure of the tension/compression spring design problem [46].

The mathematical model of the problem can be summarized as in Equation (14).

$$\begin{gathered} X = (d\ D\ N) = x_1, x_2, x_3 \\ \text{Minimize } f(X) = (x_3 + 2)x_2x_1^2 \\ \text{Subject to } g_1(X) = 1 - \left(x_2^3x_3/717.854x_1^4\right) \leq 0 \\ g_2(X) = \left(4x_2^2 - x_1x_2/12.566\left(x_2x_1^3 - x_1^4\right)\right) + \left(1/5108x_1^2\right) \leq 0 \\ g_3(X) = 1 - \left(140.45x_1/x_2^2x_3\right) \leq 0 \\ g_4(X) = ((x_1 + x_2)/1.5) - 1 \leq 0 \\ 0.005 \leq x_1 \leq 2,\ 0.25 \leq x_2 \leq 1.3,\ 2 \leq x_3 \leq 15 \end{gathered} \tag{14}$$

The result obtained from the proposed method is given in Table 14, with the general cycle number taken as 30 and FES value taken as 1000. In Table 15, the comparison results of BA_ABC with the literature are given. The BA_ABC algorithm has been shown to produce an acceptable result similar to the literature.

Table 14. Optimal solutions of tension/compression spring design problem by BA_ABC.

	x_1	x_2	x_3	$f(x)$	
Decision variables	0.054007	0.417747	8.39565	0.012667	
	g_1	g_2	g_3	g_4	
Constraint values	−0.002253	−0.115968	−4.177092	−0.685497	
	Best	Median	Worst	Mean	Std.
	0.012667	0.012688	0.01273352	0.01268755	1.46×10^{-5}

Table 15. Comparison of BA_ABC with other algorithms for the tension/compression spring design optimization problem (NA = Not Available).

Algorithms	Worst	Mean	Best	Std
EBA [31]	NA	NA	0.01267	NA
IAPSO [47]	0.01782	0.01367	0.01266	1.57×10^{-3}
CGWO [48]	0.01217	**0.01217**	**0.01195**	1.04×10^{-5}
WCA [49]	0.01295	0.01274	0.01266	8.06×10^{-5}
MBA [50]	0.01290	0.01271	0.01266	6.30×10^{-5}
ABC [52]	0.01271	0.01266	0.01266	$\mathbf{9.42 \times 10^{-6}}$
BA [53]	0.01689	0.01350	0.01267	1.42×10^{-3}
CSDE [54]	0.01266	0.01269	0.01266	3.00×10^{-5}
TLNNA [55]	0.01283	0.01268	0.01266	3.24×10^{-5}
BA_ABC	0.01273	0.01268	0.01266	1.46×10^{-5}

3.4.3. Gear Train Design Problem

The gear train design problem is an optimization problem proposed by Sandgren [44], whose aim is to minimize the cost of the gear ratio of the gear train. This problem only has a boundary constraint. There is no constraint on equality or inequality. It has four decision variables, including T_a, T_b, T_d, and T_f. The schematic representation of the problem is given in Figure 4.

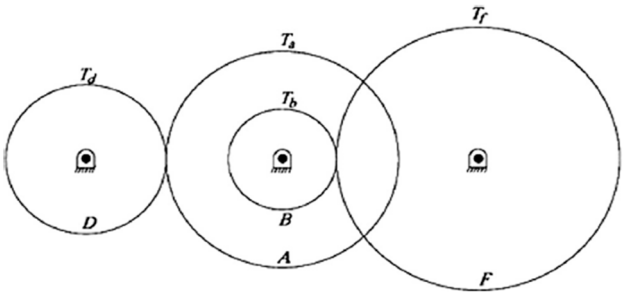


Figure 4. The structure of the gear train design problem [46].

The mathematical model of the problem can be summarized as in Equation (15).

$$\begin{gathered} X = (T_a T_b T_d T_f) = (x_1, x_2, x_3, x_4) \\ \text{Minimize } f(X) = ((1/6931) - (x_2 x_3 / x_1 x_4))^2 \\ 12 \leq x_1, x_2, x_3, x_4 \leq 60 \end{gathered} \tag{15}$$

The result obtained from the proposed method is given in Table 16, with the general cycle number taken as 30 and FES value taken as 1000. In Table 17, the comparison results of BA_ABC with the literature are given. When the results were examined, it was seen that the best solution was obtained with the BA_ABC algorithm.

Table 16. Optimal solutions of gear train design problem by BA_ABC.

	x_1	x_2	x_3	x_4	$f(x)$
Decision variables	12.00	19.7523	51.7153	31.7670	2.0732×10^{-14}
	Best	Median	Worst	Mean	Std.
	2.07×10^{-14}	5.14×10^{-11}	7.64×10^{-9}	7.42×10^{-10}	1.92×10^{-9}

Table 17. Comparison of BA_ABC with other algorithms for the gear train design optimization problem.

Algorithms	Worst	Mean	Best	Std
IAPSO [47]	1.82×10^{-8}	5.49×10^{-9}	2.70×10^{-12}	6.36×10^{-9}
CGWO [48]	2.71×10^{-10}	$\mathbf{7.09 \times 10^{-11}}$	2.83×10^{-13}	$\mathbf{1.02 \times 10^{-10}}$
MBA [50]	2.06×10^{-8}	2.47×10^{-9}	2.70×10^{-12}	3.94×10^{-9}
BA	5.11×10^{-2}	7.89×10^{-3}	1.31×10^{-8}	1.38×10^{-2}
ABC	1.02×10^{-9}	1.54×10^{-10}	1.47×10^{-13}	2.50×10^{-10}
BA_ABC	7.64×10^{-9}	7.42×10^{-10}	$\mathbf{2.07 \times 10^{-14}}$	1.92×10^{-9}

4. Investigation of Contribution of BA and ABC Algorithms to the Solution of BA_ABC Algorithm

The contribution of the algorithms that make up the hybrid system to the solution produced by the hybrid system may vary depending on the problem or function studied. Sometimes an algorithm may be more successful at producing new solutions, or algorithms may perform similarly. Determining the algorithms' contribution amounts can be a guide for future hybrid studies on these functions. For this purpose, in this section, the contribution of the BA and ABC algorithms carried out in parallel to the BA_ABC algorithm to the solution for each function is examined. The results of the contribution of the algorithms to the solution in CEC2005 functions in Table 18 and CEC2010 in Table 19 are stated. In these tables, the BA column indicates the number of times the BA produced more solutions and the number of times the BA was more successful than the ABC algorithm during the control process of the algorithms' success status. Similarly, the ABC column shows the number of times ABC produced more solutions and was more successful. These columns also provide information about the direction of information exchange. For example, BA and ABC columns for the F1 function in Table 19 show that information exchange took place 15 times from the BA population to the ABC population and nine times from the ABC population to the BA population. BA+ and BA- columns respectively indicate the number of new successful solutions produced by BA and the number of new solutions that failed to develop the current best solution. Again, ABC+ and ABC- columns show the number of successful and unsuccessful solutions produced by ABC, respectively. C+ and C- columns indicate the number of successful and unsuccessful solutions produced in the process that continued with the successful algorithm, after the parallel operation of the algorithms had ended, respectively. In general, for the relevant algorithm, it can be said that the "+" symbol indicates the number of successful solutions that developed the current best solution, while the "-" symbol indicates the number of failed solutions that could not develop the current best solution.

In the function groups studied, *mnc* was determined as 15. Accordingly, if any of the BA or ABC columns took the value of 15, it meant that the algorithm was more successful in the information exchange process, and the remaining iterations would continue with this algorithm. Therefore, it can be concluded that the remaining iterations in F1 in Table 19 were continued with the BA algorithm, while those in F2 function were continued with the ABC algorithm.

When the tables are examined, it can be seen that the contribution of the algorithms to the BA_ABC solution changes depending on the function. According to Table 18, it can be said that the contribution of the ABC algorithm to the result is higher in the F25 function, and the contribution of the algorithms to the solution is similar in the F12 function. In the remaining 23 functions, it was determined that the information exchange was mostly from BA to ABC, and the iterations remaining after the information exchange process were continued with the BA algorithm. In other words, the contribution of the BA algorithm to BA_ABC was higher in 92% of the functions, while the ABC algorithm's contribution to BA_ABC was higher in 4% of the functions. In the remaining 4%, their contribution to BA_ABC was similar. Therefore, it can be said that the BA algorithm is more successful in this set of functions, and it provided a better contribution to the solution.

Table 18. Contribution of BA and ABC to BA_ABC solutions in CEC2005.

Function	BA	ABC	BA+	BA-	ABC+	ABC-	C+	C-
F1	15	2	2813	19,797	242	44,978	7298	24,872
F2	15	0	613	19,337	87	39,813	368	39,782
F3	15	0	530	19,420	6	39,894	657	39,493
F4	15	1	271	21,009	14	42,546	74	36,086
F5	15	1	375	20,905	90	42,470	127	36,033
F6	15	5	440	26,160	284	52,916	236	19,964
F7	15	5	872	25,728	183	53,017	621	19,579
F8	15	6	216	27,182	126	56,266	96	16,114
F9	15	5	3715	22,885	321	52,879	4278	15,922
F10	15	7	599	28,661	395	58,125	849	11,371
F11	15	6	293	27,637	108	55,752	46	16,164
F12	15	10	235	33,098	185	66,482	0	0
F13	15	9	339	31,581	362	63,478	74	41,66
F14	15	1	358	20,922	131	42,429	379	35,781
F15	15	9	3418	28,502	463	63,377	590	3650
F16	15	6	163	27,767	159	55,701	74	16,136
F17	15	9	53	31,867	11	63,829	2	4238
F18	15	4	670	24,600	162	50,378	633	23,557
F19	15	6	321	27,609	235	55,625	165	16,045
F20	15	1	1225	20,055	134	42,426	1952	34,208
F21	15	7	346	28,914	327	58,193	75	12,145
F22	15	5	173	26,428	203	52,996	357	19,843
F23	15	4	114	25,156	40	50,500	96	24,094
F24	15	8	751	29,839	295	60,885	1053	7177
F25	6	15	48	27,882	23	55,837	0	16,210

Table 19. Contribution of BA and ABC to BA_ABC solutions in CEC2010.

Function	BA	ABC	BA+	BA-	ABC+	ABC-	C+	C-
F1	15	9	5155	954,845	4710	1,915,290	707	119,293
F2	1	15	497	639,503	18,193	1,261,807	7093	1,072,907
F3	12	13	2329	997,671	8919	1,991,081	0	0
F4	15	2	3595	676,405	2125	1,357,875	26,946	933,054
F5	14	11	3111	996,889	6243	1,993,757	0	0
F6	15	2	3739	676,261	1762	1,358,238	2755	957,245
F7	15	2	4414	684,474	1669	1,349,443	4699	955,301
F8	15	1	3737	636,262	1152	1,278,849	6788	1,073,212
F9	15	0	6718	593,282	705	1,199,295	152,842	1,047,158
F10	1	15	767	639,233	11,503	1,268,497	5873	1,074,127
F11	9	15	2091	957,908	11,075	1,908,926	1842	118,158
F12	15	0	8181	591,819	356	1,199,644	44,117	1,155,883
F13	13	12	5294	994,706	9139	1,990,861	0	0
F14	15	0	7295	592,705	501	1,199,499	175,931	1,024,069
F15	15	5	3532	796,468	4044	1,595,956	1946	598,054
F16	15	6	2989	837,011	4005	1,675,995	1925	478,075
F17	15	0	7078	592,923	385	1,199,614	108,336	1,091,664
F18	15	9	7395	952,605	6125	1,913,875	1472	118,528
F19	15	0	6052	593,949	34	1,199,965	171,282	1,028,718
F20	15	3	7752	712,248	2429	1,437,571	4593	835,407

When Table 19 is examined, it is seen that the information exchange took place from BA to ABC in a total of 14 functions (F1, F4, F6, F7, F8, F9, F12, F14, F15, F16, F17, F18, F19, F20) and the remaining iterations after the information exchange process was continued with the BA algorithm. It can be said that BA contributed more to the solution in these functions. In three of the remaining functions (F2, F10, F11), ABC contributed more to the solution, and the information exchange took place from ABC to

BA. The remaining iterations after this process were continued with ABC. In F3, F5, and F13 functions, the contribution of the algorithms to the solution of the BA_ABC algorithm was similar. In other words, the contribution of the BA algorithm to BA_ABC was higher in 70% of the functions, while the ABC algorithm's contribution to BA_ABC was higher in 15% of the functions. In the remaining 15%, their contribution to BA_ABC was similar.

According to the results given in Tables 18 and 19, it was determined that the contribution of BA algorithm to the solutions of the hybrid BA_ABC algorithm was higher.

5. Algorithm Complexity

In this section, the complexity of BA, ABC, and BA_ABC algorithms were calculated according to the rules defined for the CEC2020 [57] test set, and the obtained results were compared. Algorithm complexity is calculated as follows.

T0 denotes the computing time for the problem given in Equation (16).

$$\begin{gathered} x = 0.55; \\ \text{for } i = 1: 1000000 \\ x = x + x;\ x = x/2;\ x = x^* x;\ x = \text{sqrt}(x);\ x = \log(x); \\ x = \exp(x);\ x = x/(x + 2); \\ \text{end} \end{gathered} \tag{16}$$

T1 is the computing time just for Function (1) in the test set for a particular dimension(D) and 200,000 function evaluation number. The total computing time for the proposed algorithm with 200,000 evaluations of the same D dimensional function is T2. T2 is executed five times, and the average of the T2 values found is calculated (T2 = mean(T2)). Finally, the algorithm complexity is denoted by (T2 − T1)/T0 and evaluated according to the linear growth. Additionally, the algorithm complexities are calculated on the 5, 10, and 15 dimensions to determine the effect of dimension increase.

The algorithms were coded using Matlab R2016a, and algorithm complexity calculation was done by running the algorithms on a PC with Intel CPU (1.50 GHz) and 4 GB RAM. The complexity of the BA, ABC, and BA_ABC were shown in Table 20.

Table 20. The complexity of the BA, ABC, and BA_ABC algorithms.

CEC2020 (Function 1)				
		BA		
	T0	*T1*	*T2*	*(T2 − T1)/T0*
D = 5	0.2901	1.5922	13.0710	39.5684
D = 10		1.6110	13.1026	39.6125
D = 15		1.9029	13.6737	40.5749
		ABC		
	T0	*T1*	*T2*	*(T2 − T1)/T0*
D = 5	0.2901	1.5922	11.1987	33.1144
D = 10		1.6110	11.9764	35.7304
D = 15		1.9029	12.4667	36.4143
		BA_ABC		
	T0	*T1*	*T2*	*(T2 − T1)/T0*
D = 5	0.2901	1.5922	13.0957	39.6535
D = 10		1.6110	13.3799	40.5684
D = 15		1.9029	14.2354	42.5112

According to Table 20, it was seen that the complexities of all algorithms rise depending on the increased dimension. The complexity of the proposed BA_ABC algorithm was higher than the BA and

ABC algorithms, in all dimensions. Furthermore, the lowest complexity values for all dimensions were obtained in the ABC algorithm.

6. Results and Discussion

In this study, to examine the performance of the proposed BA_ABC algorithm in different dimensions and on different test sets, the algorithm was tested on classical benchmark functions, CEC2005 small-scale test problems, CEC2010 large-scale test problems, and classical engineering design problems. Table 4 shows the results of BA, ABC, and BA_ABC algorithms in classical benchmark test functions, and Table 5 shows the comparative results of BA_ABC with other algorithms (BA, ABC, FA, DE, HSABA, and MBADE). When Table 4 was examined, it was determined that the rise in dimension increased the amount of error; however, BA_ABC performed better than BA and ABC algorithms in most functions of different dimensions. Statistical tests also confirmed that BA_ABC was more successful. According to the statistical test results in Table 5, it was found that BA_ABC showed similar performance with hybrid algorithms (HSABA and MBADE). There was a significant difference between the remaining algorithms, and BA_ABC performed better than these algorithms. According to Friedman test results, the BA_ABC algorithm ranked first with the smallest mean rank value. Table 7 shows the results of BA, ABC, and BA_ABC algorithms in CEC2005 small-scale test functions. When Table 7 was examined, it was seen that BA_ABC performed better in BA and ABC algorithms in most of the functions and ranked first according to Friedman results. The comparison results of the BA_ABC algorithm with the recently proposed bat algorithms (SBAIS, MBADE, NBA, iBA, ILSSIWBA, GBA, TBA, PBA, LBA, EBA, and RBA) are given in Table 8. In the table, the BA_ABC algorithm was found to be worse than one of the compared bat versions, similar to four, and better performing than the remaining six. BA_ABC ranked fifth among the algorithms. Table 10 shows the results of BA, ABC, and BA_ABC algorithms in CEC2010 large-scale test functions. Comparison results of BA_ABC with AHDE, RSQPSO, ISCA, μDSDE, and aEUS algorithms are given in Table 11. When Table 10 was examined, it was found that BA_ABC performed better in BA and ABC algorithms in most functions and ranked first according to Friedman results. In Table 11, it was found that the BA_ABC algorithm performed similarly with one of the compared algorithms and performed better than the remaining four. BA_ABC ranked first in the ranking between algorithms. Finally, the algorithm's performance over classical engineering design problems was examined. The results of the three engineering design problems are given in Tables 12–17. When the results were examined, it was seen that the algorithm produced acceptable and successful results for these problems. Overall, the BA_ABC algorithm produced very successful results for all test sets and dimensions, and the success of the algorithm was verified by statistical test results. Also, the contributions of BA and ABC algorithms to the hybrid algorithm were examined in the fourth section. The results obtained for CEC2005 and CEC2010 are given in Tables 18 and 19. It was determined that the BA algorithm contributed more to the hybrid system in most of the functions. Finally, in the fifth section, the complexity of BA, ABC, and BA_ABC algorithms were examined. The complexity of the BA_ABC algorithm was found to be higher than the standard algorithms.

However, the increased dimension-related performance loss is still an ongoing problem for BA_ABC. According to the results of Table 4, it can be said that BA_ABC is relatively less affected compared to BA and ABC algorithms. The structural difficulty of functions (shift, rotation, etc.) is another reason for the loss in BA_ABC performance. Despite this, the algorithm is seen to find more successful results than BA and ABC algorithms in most of the functions. Consequently, BA_ABC is a successful hybrid algorithm, and the reason for its success can be said to be the reduction of convergence speed to the current best solution using inertia weight and the increase of diversity and global search capability thanks to the hybrid system created with the ABC algorithm.

Techniques other than metaheuristic algorithms also can be used to improve the performance of BA_ABC. For example, using machine learning techniques with metaheuristic algorithms might be a good option. Fine-tuning of parameters in metaheuristic algorithms affects algorithm performance

substantially. The most suitable parameter value may vary depending on the structural features and dimensions of the problem. Usually, researchers choose the parameters used for similar problems or try to find the most suitable parameter by performing tests with these values. Parameter selection of metaheuristic algorithms can be performed using machine learning strategies (such as fuzzy logic, Bayesian networks, neural networks, support vector machines, etc.). Population management and diversity are another factor affecting performance in metaheuristic algorithms. Machine learning techniques such as the Apriori algorithm, clustering techniques, etc., can be used for extracting information from previously visited solutions, discovering more promising search areas, and increasing population variety [58]. Furthermore, metaheuristic algorithms can be associated with the concept of big data, which has been frequently studied recently. Using metaheuristic algorithms in this field allows us to produce fast responses in real-time areas where data are regularly updated, to work with different data types at the same time, to cope with uncertainties, and to evaluate the information obtained through the use of the objective function [59]. As a result, it may be appropriate to use machine learning options for increasing the population diversity of the proposed BA_ABC and setting its parameters. Furthermore, examining the performance of the proposed BA_ABC algorithm on big data, it can contribute to the literature and present successful results.

7. Conclusions and Future Work

In this study, a new hybrid algorithm was proposed to improve BA's global search capability and increase its performance on different test sets. In this algorithm, which is called BA_ABC, the population was divided into two, and the BA algorithm was performed in one part and the ABC algorithm in the other. When each certain number of iterations was completed, the performance of the algorithms was evaluated, and the information exchange was ensured by replacing some of the individuals with the best fitness values in the successful algorithm with the individuals with the worst fitness values in the population of the unsuccessful algorithm. When the maximum number of exchanges was reached, the remaining iterations were continued with the successful algorithm. Thanks to the proposed BA_ABC algorithm, performance decreases due to structural problems of the BA algorithm were reduced, and its global search capability was improved.

The BA_ABC algorithm was firstly tested on 10 classic benchmark functions. This test was done on dimensions of 10, 30, 50, 100, and 1000. Despite the increased dimension, the proposed algorithm was found to be more successful than the BA and ABC algorithms. Again, the BA_ABC algorithm performed better than the algorithms selected from the literature. Secondly, CEC2005 small-scale test functions were used to determine how the BA_ABC algorithm was performing compared to the latest BA versions. The algorithm performed better than the BA and ABC algorithms. It produced acceptable results compared to the BA versions. Thirdly, the performance of BA_ABC in large-scale problems was tested on CEC2010 large-scale test functions. It was determined that the proposed algorithm performed better than BA, ABC, and the latest algorithms in the literature. Finally, the BA_ABC algorithm was tested on three frequently used problems of classical engineering design problems. The BA_ABC algorithm produced acceptable results, which were similar to those in the literature in these problems. Also, the contribution of BA and ABC algorithms, which constituted the hybrid algorithm, to the solutions was examined on CEC2005 and CEC2010 functions. It was observed that the BA algorithm contributes more to the solutions of BA_ABC in most of the functions. Finally, in the calculation about the algorithm complexity, it was found that the complexity of the BA_ABC algorithm is more than the standard algorithms. In general, when all the results were examined, it was determined that the proposed algorithm produced successful and acceptable results in different test groups. As a future study, the hybrid system components used in the BA_ABC algorithm can be replaced with different algorithms and tested on CEC functions in recent years. Furthermore, machine learning techniques can be added to increase the performance of the algorithm, or its performance can be examined on big data problems as a different field of study.

Author Contributions: Formal analysis, G.Y. and Ö.K.B.; methodology, G.Y. and Ö.K.B.; software, G.Y.; writing—original draft, G.Y.; writing—review and editing, Ö.K.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Karaboga, D. *Yapay Zekâ Optimizasyonu Algoritmaları*; Atlas Yayın Dağıtım: İstanbul, Turkey, 2004.
2. Gao, W.-F.; Liu, S.-Y. A modified artificial bee colony algorithm. *Comput. Oper. Res.* **2012**, *39*, 687–697. [CrossRef]
3. Zhu, G.; Kwong, S. Gbest-guided artificial bee colony algorithm for numerical function optimization. *Appl. Math. Comput.* **2010**, *217*, 3166–3173. [CrossRef]
4. Yang, X.; Allan, R.J. Web-based Virtual Research Environments. *Nat. Artif. Reason.* **2009**, *284*, 65–80. [CrossRef]
5. Cai, X.; Gao, X.Z.; Xue, Y. Improved bat algorithm with optimal forage strategy and random disturbance strategy. *Int. J. Bio-Inspired Comput.* **2016**, *8*, 205. [CrossRef]
6. Meng, X.-B.; Gao, X.; Liu, Y.; Zhang, H. A novel bat algorithm with habitat selection and Doppler effect in echoes for optimization. *Expert Syst. Appl.* **2015**, *42*, 6350–6364. [CrossRef]
7. Cai, X.; Wang, H.; Cui, Z.; Cai, J.; Xue, Y.; Wang, L. Bat algorithm with triangle-flipping strategy for numerical optimization. *Int. J. Mach. Learn. Cybern.* **2017**, *9*, 199–215. [CrossRef]
8. Zhu, B.; Zhu, W.; Liu, Z.; Duan, Q.; Cao, L. A Novel Quantum-Behaved Bat Algorithm with Mean Best Position Directed for Numerical Optimization. *Comput. Intell. Neurosci.* **2016**, *2016*, 1–17. [CrossRef]
9. Ghanem, W.A.H.M.; Jantan, A. An enhanced Bat algorithm with mutation operator for numerical optimization problems. *Neural Comput. Appl.* **2017**, *31*, 617–651. [CrossRef]
10. Shan, X.; Cheng, H. Modified bat algorithm based on covariance adaptive evolution for global optimization problems. *Soft Comput.* **2017**, *22*, 5215–5230. [CrossRef]
11. Nawi, N.M.; Rehman, M.Z.; Khan, A.; Chiroma, H.; Herawan, T. A Modified Bat Algorithm Based on Gaussian Distribution for Solving Optimization Problem. *J. Comput. Theor. Nanosci.* **2016**, *13*, 706–714. [CrossRef]
12. Chakri, A.; Khelif, R.; Benouaret, M.; Yang, X.-S. New directional bat algorithm for continuous optimization problems. *Expert Syst. Appl.* **2017**, *69*, 159–175. [CrossRef]
13. Al-Betar, M.A.; Awadallah, M.A. Island bat algorithm for optimization. *Expert Syst. Appl.* **2018**, *107*, 126–145. [CrossRef]
14. Gan, C.; Cao, W.; Wu, M.; Chen, X. A new bat algorithm based on iterative local search and stochastic inertia weight. *Expert Syst. Appl.* **2018**, *104*, 202–212. [CrossRef]
15. Topal, A.O.; Altun, O. A novel meta-heuristic algorithm: Dynamic Virtual Bats Algorithm. *Inf. Sci.* **2016**, *354*, 222–235. [CrossRef]
16. Wang, Y.; Wang, P.; Zhang, J.; Cui, Z.; Cai, X.; Zhang, W.; Chen, J. A Novel Bat Algorithm with Multiple Strategies Coupling for Numerical Optimization. *Mathematics* **2019**, *7*, 135. [CrossRef]
17. Liu, Q.; Wu, L.; Xiao, W.; Wang, F.; Zhang, L. A novel hybrid bat algorithm for solving continuous optimization problems. *Appl. Soft Comput.* **2018**, *73*, 67–82. [CrossRef]
18. Wang, G.-G.; Guo, L. A Novel Hybrid Bat Algorithm with Harmony Search for Global Numerical Optimization. *J. Appl. Math.* **2013**, *2013*, 1–21. [CrossRef]
19. Fister, I.; Fong, S.; Brest, J.; Fister, I. A Novel Hybrid Self-Adaptive Bat Algorithm. *Sci. World J.* **2014**, *2014*, 709738. [CrossRef] [PubMed]
20. Imane, M.; Nadjet, K. Hybrid Bat algorithm for overlapping community detection. *IFAC-PapersOnLine* **2016**, *49*, 1454–1459. [CrossRef]
21. Cincy, W.; Jeba, J. Performance Analysis of Novel Hybrid A-BAT Algorithm in Crowdsourcing Environment. *Int. J. Appl. Eng. Res.* **2017**, *12*, 14964–14969.
22. Chaudhary, R.; Banati, H. Swarm bat algorithm with improved search (SBAIS). *Soft Comput.* **2018**, *23*, 11461–11491. [CrossRef]

23. Rauf, H.T.; Malik, S.; Shoaib, U.; Irfan, M.N.; Lali, M.I. Adaptive inertia weight Bat algorithm with Sugeno-Function fuzzy search. *Appl. Soft Comput.* **2020**, *90*, 106159. [CrossRef]
24. Yildizdan, G.; Baykan, Ö.K. A novel modified bat algorithm hybridizing by differential evolution algorithm. *Expert Syst. Appl.* **2020**, *141*, 112949. [CrossRef]
25. Nguyen, T.-T.; Pan, J.-S.; Dao, T.-K.; Kuo, M.-Y.; Horng, M.-F. Hybrid Bat Algorithm with Artificial Bee Colony. In *Advances in Intelligent Systems and Computing*; Springer: Berlin, Germany, 2014; Volume II, pp. 45–55.
26. Al-Betar, M.A.; Awadallah, M.A.; Faris, H.; Yang, X.-S.; Khader, A.T.; AlOmari, O.A. Bat-inspired algorithms with natural selection mechanisms for global optimization. *Neurocomputing* **2018**, *273*, 448–465. [CrossRef]
27. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization; Technical Report-TR06*; Computer Engineering Department, Engineering Faculty, Erciyes University: Kayseri, Turkey, 2005.
28. Karaboga, D.; Basturk, B. On the performance of artificial bee colony (ABC) algorithm. *Appl. Soft Comput.* **2008**, *8*, 687–697. [CrossRef]
29. Karaboga, D.; Akay, B. A comparative study of Artificial Bee Colony algorithm. *Appl. Math. Comput.* **2009**, *214*, 108–132. [CrossRef]
30. Yılmaz, S.; Küçüksille, E.U. Improved Bat Algorithm (IBA) on Continuous Optimization Problems. *Lect. Notes Softw. Eng.* **2013**, *1*, 279–283. [CrossRef]
31. Yılmaz, S.; Küçüksille, E.U. A new modification approach on bat algorithm for solving optimization problems. *Appl. Soft Comput.* **2015**, *28*, 259–275. [CrossRef]
32. Arora, U.; Lodhi, E.A.; Saxena, T. PID Parameter Tuning Using Modified BAT Algorithm. *J. Autom. Control. Eng.* **2016**, *4*, 347–352. [CrossRef]
33. Feng, Y.; Teng, G.-F.; Wang, A.-X.; Yao, Y.-M. Chaotic Inertia Weight in Particle Swarm Optimization. In Proceedings of the Second International Conference on Innovative Computing, Information and Control (ICICIC 2007), Kumamoto, Japan, 5–7 September 2007; p. 475.
34. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.-P.; Auger, A.; Tiwari, S. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *KanGAL Rep.* **2005**, *2005005*, 2005.
35. Nanyang Technological University. Available online: https://www3.ntu.edu.sg/home/epnsugan/index_files/CEC10-LSO/CEC10.htm (accessed on 28 September 2020).
36. Sheskin, D.J. *Handbook of Parametric and Nonparametric Statistical Procedures*; CRC Press: Boca Raton, FL, USA, 2020.
37. García, S.; Molina, D.; Lozano, M.; Herrera, F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 Special Session on Real Parameter Optimization. *J. Heuristics* **2008**, *15*, 617–644. [CrossRef]
38. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
39. Ge, H.; Sun, L.; Yang, X. Adaptive hybrid differential evolution with circular sliding window for large scale optimization. In Proceedings of the 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Changsha, China, 13–15 August 2016; pp. 87–94.
40. Fang, W.; Zhang, L.; Zhou, J.; Wu, X.; Sun, J. A novel quantum-behaved particle swarm optimization with random selection for large scale optimization. *2017 IEEE Congr. Evol. Comput. (CEC)* **2017**, 2746–2751. [CrossRef]
41. Long, W.; Wu, T.; Liang, X.; Xu, S. Solving high-dimensional global optimization problems using an improved sine cosine algorithm. *Expert Syst. Appl.* **2019**, *123*, 108–126. [CrossRef]
42. Yıldız, Y.E.; Topal, A.O. Large scale continuous global optimization based on micro differential evolution with local directional search. *Inf. Sci.* **2019**, *477*, 533–544. [CrossRef]
43. Gardeux, V.; Omran, M.G.H.; Chelouah, R.; Siarry, P.; Glover, F. Adaptive pattern search for large-scale optimization. *Appl. Intell.* **2017**, *35*, 1095–1330. [CrossRef]
44. Sandgren, E. Nonlinear Integer and Discrete Programming in Mechanical Design Optimization. *J. Mech. Des.* **1990**, *112*, 223–229. [CrossRef]
45. Kannan, B.K.; Kramer, S.N. An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design. *J. Mech. Des.* **1994**, *116*, 405–411. [CrossRef]

46. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput. Des.* **2011**, *43*, 303–315. [CrossRef]
47. Ben Guedria, N. Improved accelerated PSO algorithm for mechanical engineering optimization problems. *Appl. Soft Comput.* **2016**, *40*, 455–467. [CrossRef]
48. Kohli, M.; Arora, S. Chaotic grey wolf optimization algorithm for constrained optimization problems. *J. Comput. Des. Eng.* **2017**, *5*, 458–472. [CrossRef]
49. Eskandar, H.; Sadollah, A.; Bahreininejad, A.; Hamdi, M. Water cycle algorithm—A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Comput. Struct.* **2012**, *110*, 151–166. [CrossRef]
50. Sadollah, A.; Bahreininejad, A.; Eskandar, H.; Hamdi, M. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Appl. Soft Comput.* **2013**, *13*, 2592–2612. [CrossRef]
51. Garg, H. A hybrid PSO-GA algorithm for constrained optimization problems. *Appl. Math. Comput.* **2016**, *274*, 292–305. [CrossRef]
52. Garg, H. Solving structural engineering design optimization problems using an artificial bee colony algorithm. *J. Ind. Manag. Optim.* **2014**, *10*, 777–794. [CrossRef]
53. Gandomi, A.H.; Yang, X.-S.; Alavi, A.H.; Talatahari, S. Bat algorithm for constrained optimization tasks. *Neural Comput. Appl.* **2012**, *22*, 1239–1255. [CrossRef]
54. Zhang, Z.; Ding, S.; Jia, W. A hybrid optimization algorithm based on cuckoo search and differential evolution for solving constrained engineering problems. *Eng. Appl. Artif. Intell.* **2019**, *85*, 254–268. [CrossRef]
55. Zhang, Y.; Jin, Z.; Chen, Y. Hybrid teaching–learning-based optimization and neural network algorithm for engineering design optimization problems. *Knowl.-Based Syst.* **2020**, *187*, 104836. [CrossRef]
56. Belegundu, A.D.; Arora, J.S. A study of mathematical programmingmethods for structural optimization. Part II: Numerical results. *Int. J. Numer. Methods Eng.* **1985**, *21*, 1601–1623. [CrossRef]
57. Nanyang Technological University. Available online: https://www.ntu.edu.sg/home/epnsugan/index_files/CEC2020/CEC2020-2.htm (accessed on 28 September 2020).
58. Calvet, L.; De Armas, J.; Masip, D.; Juan, A.A. Learnheuristics: Hybridizing metaheuristics with machine learning for optimization with dynamic inputs. *Open Math.* **2017**, *15*, 261–280. [CrossRef]
59. Dhaenens, C.; Jourdan, L. *Metaheuristics for Big Data*; Wiley Online Library: Hoboken, NJ, USA, 2016.