ABFIA: A hybrid algorithm based on artificial bee colony and Fibonacci indicator algorithm

Alireza Etminaniesfahani[1,*], Hanyu Gu[1], Amir Salehipour[1]

[1] *School of Mathematical and Physical Sciences, University of Technology Sydney 15 Broadway Ultimo NSW 2007*

Abstract

The artificial bee colony (ABC) is a metaheuristic optimization algorithm known for its simplicity, flexibility, and efficiency. The algorithm, however, suffers from slow convergence due to a lack of a powerful local search capability. The Fibonacci indicator algorithm (FIA), on the other hand, is a recently proposed derivative-free metaheuristic that incorporates a powerful local search mechanism based on the line search method. This paper proposes hybridizing the artificial bee colony with the Fibonacci indicator algorithm to achieve strong exploration and highly efficient exploitation capabilities. We show that the hybrid algorithm is better than ABC and FIA and delivers superior outcomes for various optimization functions widely used in the literature, including 20 scalable basic and ten complex CEC2019 test functions.

Keywords: Artificial bee colony algorithm, Fibonacci indicator algorithm, Hybrid algorithms, Metaheuristics

1. Introduction

Many engineering problems are formulated as nonlinear mathematical programs, which are generally challenging to optimize. Indeed, many nonlinear optimization problems are NP-hard [1]. A nonlinear optimization prob-

*Corresponding author

Email address: Alireza.Etminaniesfahani@student.uts.edu.au (Alireza Etminaniesfahani)

lem can be formulated as [2, 3]:

$$z = \min_x \{f(x) : l \leq x \leq u\}, \tag{1}$$

where $x = (x_1, x_2, \ldots, x_D) \in \mathbb{R}^D$ is a D-dimensional vector of decision variables, $f(x)$ is a nonlinear objective function, $l = (l_1, l_2, \ldots, l_D)$ and $u = (u_1, u_2, \ldots, u_D)$ are the lower and upper limits for decision variables, respectively.

Due to computational challenges of solving nonlinear optimization problems, various non-exact methods, such as metaheuristic algorithms, have been developed to obtain "good" solutions instead of finding the global optimal solutions. Many metaheuristic algorithms are relatively easy to implement, and they do not require the problem's gradient information. Furthermore, by employing stochastic operators [2, 4], the metaheuristic algorithms will have the capability of escaping from the local optima, leading therefore to obtaining solutions that are reasonably close to the global optima. Those advantages have led the metaheuristics to be a common tool for solving not only engineering optimization problems, but also problems in areas such as finance and science [5, 6, 7, 8].

Among the metaheuristic optimization methods widely used for global optimization, the artificial bee colony (ABC) algorithm and its variants have been the most successful in solving various nonlinear optimization problems [9]. The ABC algorithm has a powerful exploration ability, is easy to implement, and benefits from a small number of parameters. A powerful exploration capability is critical for the non-exact algorithms because the lack of a robust global search scheme in an algorithm increases the possibility of getting trapped in low quality local optima [10, 11, 12, 13]. Compared to other metaheuristics, e.g., the particle swarm optimization (PSO), the ABC algorithm, however, has a slower convergence rate [14]. One strategy to improve the slow convergence rate of the ABC algorithm without losing its strong global search capability is through hybridizing the ABC with local search algorithms such as the line search method [15].

The line search algorithm [16] is well-known for its high convergence speed. In the line search method, first, a descent direction along which the objective function is optimized (e.g., decreased for a minimization problem) is identified. Then, the step size is calculated, either exactly or approximately, to determine how far the candidate solution can move in the descent direction [17]. Combining the line search with other algorithms has been shown to produce promising results [18, 19, 20]. Nonetheless, the line search method has

certain limitations. For example, using an exact method in the line search to identify the step size restricts the method's usability on continuously differentiable functions, i.e., on smooth functions. Further, computing the step size exactly is computationally expensive, and as such, the line search usually needs to be terminated early to reduce the cost of evaluating the objective function.

The newly published Fibonacci indicator algorithm (FIA) [21] proposes a new line search scheme based on the Fibonacci percentages. The FIA does not suffer from the shortcomings of the original line search method, and because the FIA applies different methods to avoid getting trapped in low quality local optima, it may obtain superior results. The results in [21] show an excellent performance for the FIA, along with a fast convergence speed for FIA in optimizing a broad range of objective functions, which are not restricted to smooth functions. Nonetheless, as discussed by [21], the FIA has certain shortcomings. For example, the FIA may not be suitable for solving high-dimensional multimodal functions, which demand a robust exploration scheme. Also, the FIA may not deliver quality solutions for functions with multiple local optima, and where a balanced exploration and exploitation strategy is needed. The ABC's robust global search capability and the FIA's fast convergence speed motivate us to hybridize the ABC and FIA to improve the searchability of the FIA and the convergence speed of the ABC algorithm.

The contributions of our study can be summarized as follows: (*i*) we propose a strategy to hybridize the ABC and FIA methods; (*ii*) we show that the hybridized method benefits from a fast convergence speed and a robust search capability; and (*iii*) we show that the hybridized method leads to quality solutions and can outperform the stand-alone ABC and FIA, as well as the state-of-the-art variants of the ABC and a number of available metaheuristics.

The remaining of the paper is organized as follows. In Section 2, we present the related works to our research and explain the ABC algorithm. In Section 3, we detail the operation of the FIA method. In Section 4, we propose the hybridized algorithm, which we name ABFIA, and discuss components of the hybrid ABFIA algorithm. In Section 5, We discuss setting the parameters of the ABFIA and analyze its convergence behavior. We then perform comprehensive computational experiments to assess the performance of the ABFIA, in which we compare the ABFIA and some state-of-the-art methods. The paper concludes in Section 6.

2. Related work and background

In 1986, Fred Glover invented the term "metaheuristic" [22] to represent a heuristic approach that is not problem-specific. Randomness is the key feature of these algorithms to escape from the local optimal solutions and plays a crucial role in solving most nonlinear optimization problems, which are typically NP-hard [1]. However, the randomized structure of such methods can make them unreliable in producing a consistent result in each run [23]. The "no free lunch" theorem proves that one metaheuristic cannot be claimed as the best optimization algorithm in solving all types of problems [24]. According to this theorem, one algorithm may be best suited for a set of problems while showing poor results on another group of test problems. As a result, the research area of metaheuristics has been highly active, competitive, and challenging [25].

In this section, we first discuss various groups of metaheuristic algorithms. Then, we focus on the artificial bee colony metaheuristic algorithm and its variants.

2.1. Four classes of metaheuristics

The metaheuristic algorithms can be classified into the following four groups [26]:

Physical-based methods. Examples include simulated annealing (SA) [27], gravitational search algorithm (GCA) [28] and galaxy-based search algorithm (GbSA)[29]. In these methods, rules from physical phenomena are used to guide the algorithm towards obtaining quality solutions. These algorithms provide promising results in solving different optimization problems. However, they are more efficient in optimizing unimodal problems, where they show their strong local search capability [30]. The unimodal optimization problems consist of objective functions with only one minimum or one maximum value.

Evolutionary algorithms. These algorithms are inspired by laws of evolution in nature, are conceptually simple, and are easy to understand. These algorithms are flexible in solving various types of optimization problems, including nonlinear and NP-hard problems. A huge number of publications in this area indicates the attractiveness of evolutionary algorithms [31]. The evolutionary algorithms use a set of random populations to start searching

the feasible area. By combining the best individuals to produce new generations, they obtain new solutions and continue the search. The genetic algorithm (GA) [32], differential evolution (DE) [33] and the evolutionary strategy (ES) [34] are among the most popular evolutionary algorithms in solving various types of optimization problems. In comparison with the GA in solving nonlinear optimization problems, the ES has the difficulty of attaining an optimal convergence rate if there is a large number of constraints [35]. The DE may be chosen over the GA in solving multi-objective optimization problems because the DE explores the search space more efficiently [36].

Swarm intelligence methods. These methods, which are based on the collaboration of simple agents to find quality solutions, are commonly inspired by the colonies and folks in nature. The ant colony optimization (ACO) [37], the particle swarm optimization (PSO) [38] and the ABC algorithm [39], which is inspired by the foraging behavior of honey bees, are some of the most applied swarm intelligence methods. The ACO algorithm is well suited for discrete optimization problems [40], and the PSO is used in multi-objective, dynamic optimization, and constraint handling settings [41]. Compared with the ABC, the PSO is a faster algorithm for solving nonlinear and high-dimensional optimization problems. The ABC, however, has a robust global search ability and is suitable for problems with many local optima [42]. Other recently proposed swarm intelligence algorithms such as Dragonfly algorithm (DA) [43] Salp swarm algorithm (SSA) [44], the ant lion optimizer (ALO) [45], the grey wolf optimizer (GWO) [25], and the whale optimization algorithm (WOA) [26] have their own limitations.

For example, the ALO, which is inspired by the foraging behavior of ant lion's larvae, demands long run times because of the random walk process incorporated in the algorithm [46]. The main drawback of the WOA, which is inspired by the hunting behavior of humpback whales, is the poor exploration capability [47]. The GWO is motivated by the hunting behavior of grey wolves in nature. The main shortcomings of the GWO algorithm are slow convergence rate and low accuracy of the algorithm [48].

Human-based algorithms. These algorithms are inspired by human behavior and are another category for metaheuristic algorithms. The harmony search (HS) [49] and tabu search (TS) [50] are the most popular human-based algorithms. The HS is influenced by musicians' improvisational processes and

offers many appealing features, including ease of implementation and a small number of parameters. The main weakness in this algorithm is its premature convergence [51]. Another human-based algorithm, the learner performance based behavior algorithm (LPB), is inspired by the process of accepting graduates from high schools at universities[52]. The recently published FIA [21] is another human-based metaheuristic algorithm. The FIA was inspired by the Fibonacci retracement, a characteristic that the stock market traders use to predict the best moment to speculate in the market. Fluctuations in stock prices in a time interval lead to a large number of local minima and maxima. In the FIA, the objective function value is considered as the price of a stock, and the aim is to predict the local optimum values of the objective function.

2.2. The artificial bee colony algorithm

The ABC algorithm was proposed by [39] to optimize numerical problems. The operation of the ABC is based on the behavior of honey bee swarms in nature. A bee colony consists of "employed" bees, "onlooker" bees, and "scout" bees. The onlooker and scout bees may be referred to as the "unemployed" bees. The "employed" bees evaluate food sources based on the quality and distance to the hive and share that information with the unemployed bees within the dancing areas in the hive. Around 90-95% of the unemployed bees are onlooker bees, who aim to improve the food sources. The onlooker bees watch many dances in the hive to choose the best food source and then employ themselves on that. The scout bees randomly search the environment.

2.2.1. Operation of the ABC algorithm

There are four main phases in the ABC algorithm. In the first phase, which is also called the "initialization" phase, the ABC creates a population of SN solutions. Each solution $x^i = (x_1^i, x_2^i, \ldots, x_D^i)$, $i = 1, 2, \ldots, SN$ is called a food source with fitness value $F(x^i)$. The j^{th} dimension of a food source x^i is generated by

$$x_j^i = l_j + rand[0,1] \cdot (u_j – l_j), j = 1, \ldots, D \tag{2}$$

In the second phase, which is called the "employed bee" phase, new solutions are generated based on the current solutions. Let V_j^i be the j^{th} dimension of the i^{th} candidate solution. The ABC generates V_j^i from the previous solutions as follows:

$$V_j^i = x_j^i + \psi(x_j^i - x_j^k), j = 1, 2, \ldots, D, \tag{3}$$

where $k \neq i$ is randomly selected from $\{1, 2, \ldots, SN\}$, and ψ is a uniform random number in the range $[-1, 1]$. If the best solution is improved, the employed bees update their position, i.e., the area they search, by replacing x^i by V^i.

In the third phase of the ABC, which is called the "onlooker bee" phase, solutions are analyzed and selected by the onlooker bees: A roulette wheel selection mechanism is employed to select solution x^i with probability.

$$P_i = 1 - \frac{F(x^i)}{\sum_{j=1}^{SN} F(x^j)}. \tag{4}$$

According to Equation (4), the better the fitness value of the i^{th} solution, the greater chance of x^i to be used to generate a new solution. Next, a local search is performed around each of the selected solutions. Given the i^{th} solution is selected, the algorithm uses Equation (3) to perform the local search around that solution, and once a superior solution (with an improved fitness value) is obtained, the position of the solution is updated accordingly. The local search is performed for a predetermined number of trials, and if the local search fails to deliver a new solution, meaning that no updated position is recorded, the ABC algorithm starts the fourth phase.

In the fourth phase, which is called the "scout bee" phase, the solution is replaced with a new random solution generated according to Equation (2). The pseudo-code of the ABC [6] is summarized in Algorithm 1.

2.2.2. Improving the ABC algorithm

This section discusses certain improvements to overcome the major limitations of the ABC algorithm, which include poor searching strategy in the onlooker bee phase and lack of an effective strategy to avoid being trapped in low quality solutions in the scout bee phase.

Firstly, the ABC algorithm converges very slowly, and that the local and global search abilities are not balanced [6, 53]. The poor exploitation and slow convergence of the ABC roots in the execution of the scout bee phase right after the local search performed by the onlooker bees, which disrupts the convergence [54]. In the ABC algorithm, the onlooker bees aim to improve the solutions by focusing on better solutions, i.e., the onlooker bees select solutions according to their quality (see Equation 3).

Algorithm 1: The ABC algorithm

Input: Objective function $F(x)$, SN
Output: A feasible solution
Phase 1: Initialization
Use Equation 2 to create SN initial solution (food sources); set $trial_i = 0$ for each solution.
while *the stopping criterion is not met* **do**
    Phase 2: The employed bee phase
    for $i = 1, 2, \ldots, SN$ **do**
        Generate a new candidate solution V^i using Equation (3);
        if $F(V^i) \leq F(x_i)$ **then** $x^i = V^i$;
        if $F(V^i) > F(x^i)$ **then** $trial_i = trial_i + 1$;
        else $trial_i = 0$;
    Phase 3: The onlooker bee phase
    Calculate the probability P_i for solution x^i using Equation (4);
    for $i = 1, 2, \ldots, SN$ **do**
        if $rand(0, 1) \leq P_i$ **then**
            Generate a new candidate solution V^i using Equation (3) for the onlooker bees;
            if $F(V^i) \leq F(x^i)$ **then** $x^i = V^i$;
            if $F(V^i) > F(x^i)$ **then** $trial_i = trial_i + 1$;
            else $trial_i = 0$;
    Phase 4: The scout bee phase
    for $i = 1, 2, \ldots, SN$ **do**
        if $trial_i > limit$ **then**
            Replace x^i by a new randomly produced candidate solution using Equation (2);
Return: x;

Because most onlooker bees concentrate on the neighborhood of high-quality solutions, the remainder of the solutions might not be sought, implying that the ABC may not effectively perform the search around other solutions. Consequently, the exploration capability of the ABC algorithm is adversely impacted [55], and this becomes worse when the algorithm finds a solution with significantly superior fitness value during the onlooker bee phase.

Secondly, the scout bee phase of the ABC algorithm performs a random search. A random search usually demands a large number of trials and has been documented not to be a successful strategy to escape from local optima [56, 57].

To overcome those limitations, various techniques and hybridization methods were proposed to improve the ABC algorithm. For example, in what is named the ABCx, the artificial bee colony algorithm was improved by employing novel search techniques in the employed and onlooker bee phases [58]. The improved ABC algorithm (IABC) uses the mutation and crossover operations of a differential evolution (DE) algorithm to enhance the ABC's exploitation capability [59], and the ABC based fitness weighted search (ABCFWS) algorithm considers weights for fitness values of the food source and selected neighbor food sources to obtain a more balanced exploration and exploitation [60]. The global-best-solution guided ABC (GABC) by [61] aims to improve the convergence rate and exploitation ability of the ABC towards obtaining global optimum solutions. In that method, certain information of the best solution is used during the search operation. Gao and Liu proposed learning strategies in the modified ABC (MABC) algorithm [62]. They also proposed novel search strategies to balance the original ABC algorithm's exploration and exploitation capabilities. Based on the solution update rule of DE algorithm, ABC-best has been proposed by [53].

In general, optimization algorithms can be hybridized to use the algorithm's advantages and eliminate the disadvantages of each algorithm. For example, in [42], the PSO was used in the employed and onlooker bee phases for continuous optimization problems. The resulting hybrid algorithm showed promising results in unimodal problems compared to the stand-alone ABC and PSO algorithms. In another attempt, the PSO and ABC were used in the particle-bee algorithm [63] to solve the construction site layout optimization problem. In what is called the velocity-based ABC algorithm, which is proposed for high-dimensional continuous optimization problems, the PSO algorithm was employed to guide the search in the onlooker bee

phase [64]. In another attempt, the crossover operator of the GA was applied after the employed bee phase is performed to improve the solutions before the onlooker bee phase starts [65].

To escape from being trapped in low-quality local optima, the simulated annealing (SA) was applied to the employed bee phase to enhance the exploitation of the ABC algorithm [66]. That hybrid algorithm was also shown to outperform the stand-alone ABC algorithm. In [9], a combination of the ABC with SA was introduced in which the acceptance rule allows for accepting inferior solutions with a controlled probability. During the algorithm's operation, the probability of accepting an inferior candidate solution is dynamically reduced.

The aforementioned studies show that hybridizing the ABC with existing heuristics and metaheuristics is an effective strategy in improving the performance of the ABC algorithm. In the present paper, we propose to hybridize the ABC with the FIA, which has been shown to be an efficient and effective algorithm in solving a wide range of optimization problems [21]. We will discuss the FIA in Section 3 and the hybridization of the ABC with FIA in Section 4.

3. The Fibonacci indicator algorithm

The FIA is a newly published metaheuristic optimization method [21], which is inspired by the Fibonacci retracement method, a well-known technique used by stock market traders to predict highly volatile stock prices.

In the stock market, technical analysis support and resistance levels are fixed levels for a stock price where it is assumed that the price would begin to stop and reverse. At a support level, the stock price is likely to find support when it falls, which means that rather than breaking through this support level and falling again, the price is more likely to bounce off it. On the contrary, as the price increases, it encounters opposition at the resistance level. The Fibonacci retracement method forecasts a support level, which acts as a local minimum for the stock price, and the resistance level, which presents a local maximum for the stock price.

The above discussion follows that the FIA may predict the local minima in the line search algorithm instead of performing the expensive complete search. Assume that x and x_{best} are two different solutions, and we need to perform a line search on the line defined by these two solutions. Let a fixed number of five solutions be generated as

$$x_i = x + \tau_i(x_{best} - x), \quad i = 1, \ldots, 5, \tag{5}$$

where $\{\tau_i\} = \{50\%, 73.6\%, 88.2\%, 111.8\%, 150\%\}$ is the set of Fibonacci ratios suggested in [21].

Figure 1 illustrates how the five candidate solutions are generated using x and x_{best}. The solution with the best fitness value is selected as the result of the line search. It can be seen that this method, which is called the Fibonacci indicator (FI) method, focuses the search more around x_{best} than x.

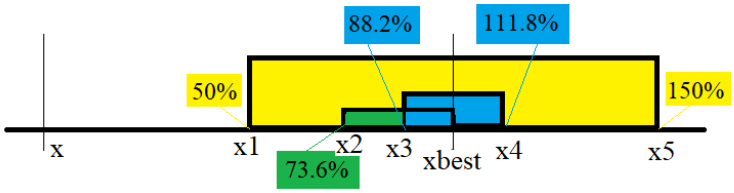

Figure 1: Generating solutions $(x_1, x_2, x_3, x_4, x_5)$ using the FI method.

3.1. Operation of FIA

The FIA is a population-based metaheuristic algorithm, where an initial population is comprised of N randomly generated solutions. Solutions in a population are sorted in non-decreasing order of their fitness value (we solve a minimization problem). The solution with the best fitness value is referred to x_{best} with fitness value g_{best}.

The FIA generates a new solution by selecting two solutions, as parents, from the current population and applying a line search to parent solutions. The current best solution is consistently chosen as a parent, and the next best solution in the population that has not yet been selected is considered the second parent. If the fitness value of the new solution is superior to both parents, we update the current population by adding the new solution to the population and removing the worst solution from the population. Whenever the current best solution is updated, all solutions in the population are treated as "never have been chosen". A new population is obtained when all the solutions in the current population have been replaced. We refer to the update of a population as Step 1.

If the population cannot be updated, i.e., at least one solution in the population cannot be removed after all solutions in the population have been attempted as parents, a new population is randomly generated.

After obtaining a new population in Step 1, the FIA performs Step 2. The FIA picks two solutions, as parents, to generate a new solution using the FI method. The best solution is always selected as one parent. With probability $1 - p$, where p is a given parameter, the second parent is selected from the remaining solutions in the order of non-increasing objective function values. With probability p, the second parent x will be generated by the crossover operation defined as

$$x = (x_1^{r_1}, x_2^{r_2}, \ldots, x_D^{r_D}), \tag{6}$$

where r_i, $i = 1, \ldots, D$, is an integer number randomly selected between 1 and N. Step 2 is indeed designed to maintain the diversification of the population by randomly generating some solutions using (6) while keeping all solutions generated by line search. This mechanism ensures a good trade-off between global exploration and local exploitation of the FIA.

If the best solution is not improved after $C \in Z^+$ consecutive trials of Steps 1 and 2, the search is restarted by generating a new population comprised of the current best solution x_{best} and $N - 1$ randomly generated solutions.

The algorithm terminates if the number of function evaluations reaches a threshold, which is a parameter of the algorithm. The pseudo-code of the FIA is shown in Algorithm (2).

Algorithm 2: The FIA [21].

```
Input: Objective function F(x), Parameters N, p, C;
Output: A feasible solutions.
Set I = 0;
Randomly generate a population of size N;
while stopping criterion is not met do
    while I < C do
        Search phase:
        Perform Step 1: Improve the population focusing on
         exploitation;
        Perform Step 2: Improve the population with a balance of
         exploration and exploitation;
        I = I + 1;
        if the current best solution was updated then I = 0;
    Set I = 0, and generate a population of size N, including the
     current best solution and N − 1 random solutions;
Return: x;
```

3.2. Motivations to hybridize FIA with ABC

The FIA has been shown superior performance on convergence speed, and solution quality over an extensive range of optimization problems [21]. However, we will later show (in Section 5) that the FIA converges very slowly in solving high-dimensional multimodal problems compared to the ABC and its variants.

The reason could be that after a restart, the population is randomly generated without taking advantage of the information from previous iterations other than the current best solution. On the contrary, when the ABC restarts the search in the scout bee phase, only one solution is randomly generated while the others are retained.

Another possible reason is that the best solution is consistently selected as one parent in the FIA. However, in the ABC algorithm, the onlooker bees randomly select the candidate solutions for the local search, leading to a more robust global convergence.

The strong exploration ability of the ABC algorithm and the fastness of the FIA motivates us to hybridize the FIA with the ABC algorithm. We expect that hybridization can enhance the performance of FIA to solve challenging problems such as the high-dimensional multimodal problems, while

at the same time, the FIA can guide the ABC to attain a faster convergence. Next, we are discussing hybridizing the ABC with the FIA.

4. Hybridizing the ABC with FIA: The ABFIA

We propose to hybridize the ABC and FIA by integrating the FIA with the ABC in the exploitation process, i.e., during the onlooker bee phase and in the scout bee phase, as an effective strategy to escape from local optima. We name the resulting algorithm the ABFIA. The scheme to hybridize the ABC with FIA is illustrated in Figure 2. The details of each component of ABFIA are explained in the following sections.

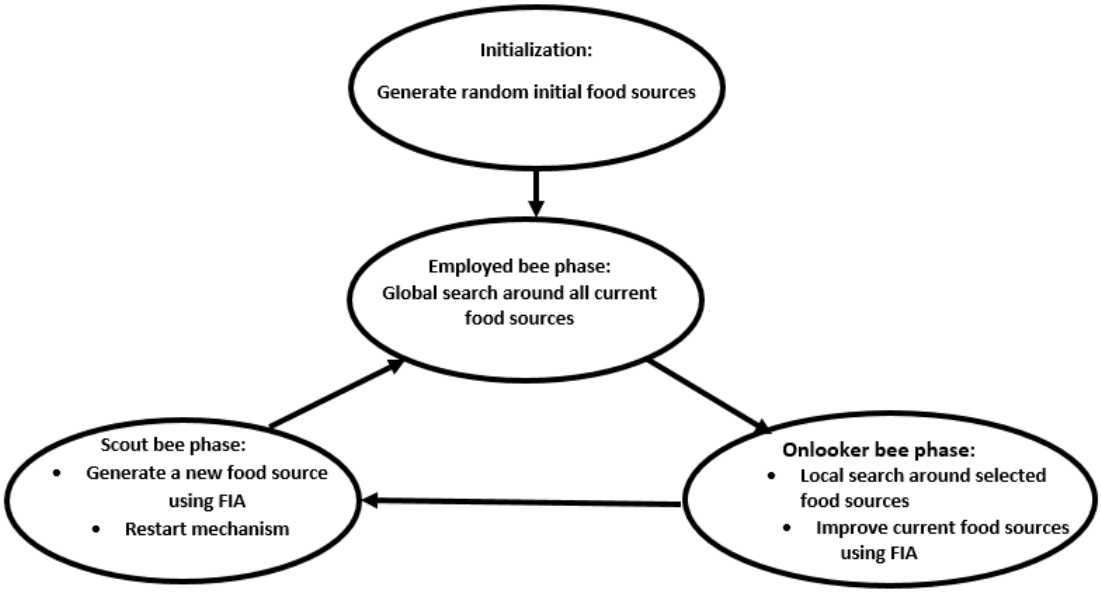

Figure 2: The conceptual diagram of the ABFIA hybrid algorithm.

4.1. Initialization

Initialization of the ABFIA is the same as in the original ABC algorithm. Accordingly, in the initialization phase, the ABFIA generates SN random solutions using Equation (2).

4.2. The employed bee phase

The employed bee phase of the ABFIA is the same as in the original ABC algorithm. In the employed bee phase, each employed bee is sent to one solution (food source). If an employed bee obtains a better solution in the neighborhood of the current solution, the current solution is updated according to Equation (3).

4.3. The onlooker bee phase

The onlooker bee phase consists of two different strategies. In the first strategy, for a given parameter $P_{Onlooker}$ percent of iterations, the onlooker bee phase is the same as in the original ABC algorithm. However, in the second strategy, for $1 - P_{Onlooker}$ percent of iterations, the onlooker bees use the FIA to search the feasible space.In the first strategy, the neighborhood of each of the selected solutions by the roulette wheel mechanism (see Equation 4) is searched (see Equation (3)). The food source is updated when a solution with better fitness is obtained within its neighborhood. Otherwise, the trial counter is increased by 1.
In the second strategy, the solution will be attempted by the FIA. Here, an initial population of the FIA is comprised of the target solution and $N - 1$ solutions, which are randomly selected from the remaining $SN - 1$ solutions. The ABC suffers from getting trapped in a local optimum due to the roulette wheel mechanism, which favors solutions with low fitness values. The ABFIA overcomes this shortcoming by randomly selecting $N-1$ solutions and the search methods of the FIA. The methods of selecting the worst of remaining solutions as a parent and performing a crossover between solutions to generate a parent improve the diversification of the solutions. The improved diversification is not at the expense of solution quality because strong local search capability is embedded within the FIA.

4.4. The scout bee phase

The main goal of the scout bee phase in the ABFIA is to escape from local optima. In this phase, each solution with more than *limit* trials, for P_{Scout} percent of iterations, is replaced by a solution found by the FIA. The FIA is initialized with a random population to maintain the diversity of the current population.
The scout bee phase in the ABC is a restart strategy that replaces the current solutions with randomly generated solutions. Therefore, the scout bee phase is not an effective mechanism for escaping from local optima. That leads to

slow convergence of the ABC [56]. The ABFIA employs the original scout bee phase for $1 - P_{Scout}$ percent of iterations and for other iterations employs the original FIA in the scout bee phase to find a better solution than randomly generating a solution, which leads to improving the convergence rate of the ABFIA against the ABC. On the other hand, the FIA in the scout bee phase employs far fewer iterations than the original FIA [21].

5. Computational experiments

To assess the performance of the proposed ABFIA, we compare the performance of the ABFIA over several benchmark functions, and that of ABC and certain variants of the ABC, such as the GABC [61], two modified ABC algorithms, namely the ABC-Best 1 and ABC-Best 2 [53], the MABC [62], the ABCx [58], and the ABCFWS [60], and with FIA, the grey wolf optimizer (GWO) [25], the whale optimization algorithm (WOA) [26], the hybrid of GWO with WOA (WOAGWO) [67], the dragonfly algorithm (DA) [43], the salp swarm algorithm (SSA) [44], and the learner performance based behavior algorithm (LPB) [52]. We use Matlab R2020a version 9.8.0.1323502 under Windows 10 operating system, and all experiments are conducted on a personal machine with an Intel(R) Core$^{\text{TM}}$ i7 clocked at 2 GHz, and 6 GB of memory. All tested functions are of type minimization.

Next, we explain the benchmark functions, followed by tuning the parameters of algorithms in Section 5.2 and the convergence analysis of the ABC, FIA and ABFIA in Section 5.3. In Sections 5.4 to 5.7, we detail outcomes of the computational experiments.

5.1. Benchmark problems

It is important to benchmark the performance of heuristic and metaheuristic optimization algorithms on problems with different characteristics. The main characteristics that shape the problem's landscape are modality, dimensionality, separability, and basins.

The peaks in the landscape, which represent global or local minimum areas (for a minimization problem), are defined by modality. The unimodal functions with one global minimum are suitable to test the exploitation and local search abilities of an optimization algorithm. However, multimodal functions contain many local minimum areas and are commonly used to test the exploration ability of the algorithms [68]. Both unimodal and multimodal functions are often rotated to add complexity in the landscape [69]. Table 1

Algorithm 3: The ABFIA.

Input: Objective function $F(x)$, Parameters $SN, limit_1, limit_2, N, p, C$.
Output: A feasible solution.
Phase 1: Initialization
Generate random solutions x^i, $i = 1, 2, \ldots, SN$ by using Equation (2);
Set $trial_i = 0$, $i = 1, 2, \ldots, SN$;
while *stopping criterion for the ABFIA is not met* **do**
    Phase 2: The employed bee phase (from the ABC algorithm)
    Search around all existing solutions and update *trial* accordingly;
    Phase 3: The onlooker bee phase
    if $rand < P_{Onlooker}$ **then**
        First strategy: The onlooker bee phase (from the ABC algorithm):
        Search around solutions selected by the roulette wheel mechanism and update *trial* accordingly;
    else
        Second strategy:
        for $i = 1, 2, \ldots, SN$ **do**
            Generate a population $\mathcal{T}$ of size N including x^i and $(N-1)$ solutions randomly selected from remaining solutions;
            Generate a new candidate solution V^i using FIA with $\mathcal{T}$ as the initial population;
            if $F(V^i) \leq F(x^i)$ **then** $x^i = V^i$;
            if $F(V^i) > F(x^i)$ **then** $trial_i = trial_i +$ Number of function evaluations within FIA;
            else $trial_i = 0$;
    Phase 4: The scout bee phase
    for $i = 1, 2, \ldots, SN$ **do**
        if $trial_i > limit$ **then**
            if $rand < P_{Scout}$ **then**
                x^i=FIA(Objective function $F(x)$,N, p, C);
                $trial_i = 0$;
            else
                The scout bee phase (from the ABC algorithm):
                Replace x^i by a new randomly produced candidate solution using Equation (2);
Return: x;

presents 20 well-known basic benchmark functions including 8 unimodal and 12 multimodal functions that we investigated in the present study.

Table 1: The basic numeric functions

Test Function	Name	Search Range	C	Function
F1	Sphere	[-100,100]	US	$f(x_1 \cdots x_n) = \sum_{i=1}^{n} x_i^2$
F2	Elliptic	[-100,100]	UN	$f(x_1 \cdots x_n) = \sum_{i=1}^{n} (10^6)^{(i-1)/(n-1)} x_i^2$
F3	Sumsquars	[-10,10]	US	$f(x_1 \cdots x_n) = \sum_{i=1}^{n} i x_i^2$
F4	SumPower	[-10,10]	MS	$f(x_1 \cdots x_n) = \sum_{i=1}^{n} \lvert x_i \rvert^{i+1}$
F5	schwefel2.22	[-10,10]	UN	$f(x_1, ..., x_n) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F6	Quartic	[-1.28,1.28]	US	$f(x_0 \cdots x_n) = \sum_{i=0}^{n} i x_i^4 + \text{random}[0, 1)$
F7	Rosenbrock	[-10,10]	UN	$f(\mathbf{x}) = f(x_1, ..., x_n) = \sum_{i=1}^{n-1} [b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$
F8	Rastrigin	[-5.12,5.12]	MS	$f(x_1 \cdots x_n) = 10d + \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i))$
F9	Griewank	[-600,600]	MN	$f(x_1 \cdots x_n) = 1 + \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$
F10	schwefel2.26	[-500,500]	MN	$f(\mathbf{x}) = f(x_1, x_2, ..., x_n) = 418.9829n - \sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i \rvert})$
F11	Ackley	[-32,32]	MN	$f(x_1 \cdots x_n) = -20exp(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}) -$ $exp(\frac{1}{n} \sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$
F12	Alpine	[-10,10]	MS	$f(x_1 \cdots x_n) = \sum_{i=1}^{n} x_i sin(x_i) + 0.1x_i$
F13	Schaffer	[-100,100]	MN	$f(x_1 \cdots x_n) = 0.5 + \frac{sin^2 \sqrt{\sum_{i=1}^{n} x_i^2} - 0.5}{(1 + 0.001(\sum_{i=1}^{n} x_i^2))^2}$
F14	Himmelblau	[-5,5]	MS	$f(x_1 \cdots x_n) = \frac{1}{n} \sum_{i=1}^{n} (x_i^4 - 16x_i^2 + 5x_i)$
F15	Shifted Rastrigin	[-5.12,5.12]	MS	$f(x_1 \cdots x_n) = 10d + \sum_{i=1}^{n} (z_i^2 - 10cos(2\pi z_i)), z = x - o$
F16	Shifted Griewank	[-600,600]	MN	$f(x_1 \cdots x_n) = 1 + \frac{1}{4000} \sum_{i=1}^{n} z_i^2 - \prod_{i=1}^{n} cos(\frac{z_i}{\sqrt{i}}), z = x - o$
F17	Shifted Ackely	[-32,32]	MN	$f(x_1 \cdots x_n) = -20exp(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^{n} z_i^2}) -$ $exp(\frac{1}{n} \sum_{i=1}^{n} cos(2\pi z_i)) + 20 + e, z = x - o$
F18	Shifted Alpine	[-10,10]	MN	$f(x_1 \cdots x_n) = \sum_{i=1}^{n} z_i sin(z_i) + 0.1z_i, z = x - o$
F19	Discus	[-5.12,5.12]	US	$f(x_1 \cdots x_n) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2$ $exp(\frac{1}{n} \sum_{i=1}^{n} cos(2\pi z_i)) + 20 + e, z = x - o$
F20	Schwefel2.20	[-10,10]	US	$f(x_1, ..., x_n) = \sum_{i=1}^{n} \lvert x_i \rvert$

The other main characteristic of benchmark functions is separability. Optimization problems in which each variable is independent of other variables are called separable. Separable benchmark functions are relatively easy to solve. The optimum value for each variable can be found by solving an independent optimization process. Non-separable functions, e.g., Griewangk, Ackley, and Alpine function, include variables related to each other, and as such, the variables cannot be optimized independently.

The other characteristic that shapes the problem's landscape is basins. A basin is a sharp falling area surrounding a wide region. Optimization algorithms are typically attracted to such regions, and it is challenging for those algorithms to escape from there. In multimodal problems, e.g., in the Ackley function, which contains a narrow, deep basin in the middle [69],

Table 2: The 100-Digit Challenge Test Functions

Test Function	Name	C	D	Search Range
F21	Storn's Chebyshev Polynomial Fitting Problem	MN	9	[-8192,8192]
F22	Inverse Hilbert Matrix Problem	MN	16	[-16384,16384]
F23	Lennard-Jones Minimum Energy Cluster Problem	MN	18	[-4,4]
F24	Shifted and Rotated Rastrigin's Function	MN	10	[-100,100]
F25	Shifted and Rotated Griewank's Function	MN	10	[-100,100]
F26	Shifted and Rotated Weierstrass Function	MN	10	[-100,100]
F27	Shifted and Rotated Schwefel's Function	MN	10	[-100,100]
F28	Shifted and Rotated Expanded Schaffer's F6 Function	MN	10	[-100,100]
F29	Shifted and Rotated Happy Cat Function	MN	10	[-100,100]
F30	Shifted and Rotated Ackley Function	MN	10	[-100,100]

finding the basin of global minima and avoiding the basin of local minima is usually challenging for metaheuristic algorithms.

Increasing dimensionality of the instances exponentially enlarges the landscape, which leads to an increase in local optimum areas. Low-dimensional instances are relatively easy to solve, and as such highly dimensional instances are suitable for evaluating the performance of optimization algorithms. Our experiments consider various functions representing the those characteristics, including 30-, 60- and 200-dimensional benchmark functions. As an additional evaluation on the ABFIA, a set of 10 modern CEC2019 benchmark test functions, which designed to be used in 2019 annual optimization contest is used [70]. Table 2 shows these test functions, which are known as "The 100-Digit Challenge" and include multimodal, unimodal, expanded multimodal, and hybrid composition functions. In the table, functions F21 to F23 have different dimensions, and function F24 to F30 are shifted and rotated. We refer the interested reader to [70] for details.

5.2. Parameter settings

There are three groups of parameters in the hybrid ABFIA algorithm, i.e., parameters for the FIA component, parameters for the ABC component, and parameters due to the hybridization. In this section, we discuss setting the value of those parameters.

Parameters setting for FIA

There are three parameters in this group, i.e., C (the search restart control), N (the population size) and p (the probability for crossover) in Algorithm 2. In the present study, the value of all parameters for the FIA are as suggested by [21]. More precisely, we set $p = 0.25$. As for the stopping criterion, we note that setting a large value for the stopping criterion of the FIA within the ABFIA means that the FIA component is preferred heavily (compared to the ABC component). Following some preliminary experiments, for each iteration, we set the stopping criterion for the FIA within ABFIA equal to 12 function evaluations and $C = 5$ which is less than half of the stopping criterion.

Parameters setting for ABC

The best outcome in the ABC algorithm is obtained when the number of solutions is equal to the number of employed and onlooker bees[71]. In all experiments conducted in the present paper, the same number of employed and onlooker bees are used, both of which are equal to the number of solutions (denoted as SN) and are set to 100. The parameter $limit$, which determines the maximum number of trials for a solution to be abandoned, is set according to sixty percent of the problem dimensions multiplied by the number of employed bees, i.e., $0.6 \times SN \times D$.

Parameters setting for ABFIA

The maximum number of iterations, run time, and the maximum number of function evaluations (FEs) are commonly used as stopping criteria for heuristic algorithms.

To have a fair comparison when reporting the results of ABFIA, the stopping criterion changed according to the study that we report their results. In experiments 1 and 2, the maximum number of function evaluations is considered as the stopping criterion, and the maximum number of iterations is used in experiment 3.

The parameters $P_{Onlooker}$ and P_{Scout} in the ABFIA are set to balance between the first and second strategies of the onlooker bee phase and restart procedure in the scout bee phase. Parameter $P_{Onlooker}$ in the ABFIA is the probability of using the original onlooker bee phase from ABC and increasing this parameter up to 1, decreases the probability of using FIA within this phase. Parameter P_{Scout} determines the probability of using FIA in the scout bee phase. To tune parameters $P_{Onlooker}$ and P_{Scout} we use F20, which is a

Table 3: The impact of parameter $P_{Onlooker}$ on different benchmark functions. $P_{Scout} = 0$.

$P_{Onlooker}$	30D	60D	200D	Average
0	1.02E+01	7.45E+01	7.87E+06	2.62E+06
0.10	2.22E-03	2.78E-01	1.22E+00	5.01E-01
0.20	1.20E-04	2.85E-01	1.76E-01	1.54E-01
0.30	7.12E-07	9.29E-04	4.83E-02	1.64E-02
0.40	1.09E-07	1.00E-04	1.40E-02	4.69E-03
0.50	4.09E-08	4.65E-05	2.03E-03	6.93E-04
0.60	1.29E-08	7.08E-06	9.30E-04	3.12E-04
0.70	**3.33E-09**	2.24E-05	3.52E-04	1.25E-04
0.80	3.27E-09	**4.20E-06**	**4.00E-05**	**1.47E-05**
0.90	4.60E-09	9.89E-06	2.27E-04	7.89E-05
1.00	7.79E-09	3.52E-05	2.95E-04	1.10E-04

unimodal function, and F9 and F12, which are two multimodal benchmark functions. For different values of $P_{Onlooker}$ between zero and one, we report the average of the results of solving those three test functions for each 30-, 60-, and 200-dimension instances. The maximum number of function evaluations is set to $1000 \times D$. Table 3 shows the impact of $P_{Onlooker}$ on the results while $P_{Scout} = 0$ which means the scout bee phase is the same as original ABC.

The parameter P_{Scout} is then tuned while using the best value for the $P_{Onlooker}$. the average of three independent runs is considered per instance, meaning that we run the ABFIA three times, and we report the average results over those three runs. According to Table 3, we set $P_{Onlooker} = 0.8$ for all instances and solve the same test functions for different values of P_{Scout} between zero and one. Table 4 shows that for 30-dimensional problems $P_{Scout} = 0.1$ provides the best results and for 60- and 200-dimensional test functions $P_{Scout} = 0.2$ leads to the best value. Based on these experiments, in this research we set $P_{Onlooker} = 0.8$ and $P_{Scout} = 0.2$ for all of the experiments.

5.3. Convergence comparison

In order to assess the convergence behavior of the ABFIA, we compare the ABFIA and the ABC and FIA on the 100-dimensional Griewangk and 10-dimensional Discus functions. The details of setting the parameters are explained in section 5.2.

Table 4: The impact of parameter P_{Scout} on different benchmark functions. $P_{Onlooker} = 0.8$.

P_{Scout}	30D	60D	200D	Average
0	1.60E-12	1.53E-09	4.84E-09	2.12E-09
0.1	**5.06E-13**	1.05E-08	4.71E-08	1.92E-08
0.2	6.34E-13	**7.61E-11**	**4.80E-10**	**1.86E-10**
0.3	6.64E-13	8.24E-10	2.20E-08	7.59E-09
0.4	1.91E-13	9.31E-10	7.39E-09	2.77E-09
0.5	9.72E-13	1.77E-09	1.07E-08	4.16E-09
0.6	2.88E-13	4.06E-09	4.98E-09	3.01E-09
0.7	3.50E-13	3.14E-09	2.20E-08	8.39E-09
0.8	3.22E-13	5.76E-09	4.66E-09	3.47E-09
0.9	1.82E-13	3.73E-10	1.36E-08	4.67E-09
1	2.62E-13	3.37E-09	7.89E-09	3.75E-09

Figure 3 demonstrates that the ABC algorithm converges slowly in comparison to the FIA and ABFIA on both functions. In solving the Discus function, the FIA converges faster than other algorithms. However, the ABFIA obtains better solutions. In the Griewangk function, ABFIA converges faster than other algorithms and delivers solutions closer to the global optima. As shown in Figure 3, the FIA converges faster than the ABC but gets trapped in poorer local optima.
Next, we detail the outcomes of the computational experiments.

5.4. Experiment 1: Unimodal benchmark test functions

In this section, the performance of ABFIA is evaluated on various unimodal 30- and 60-dimensional test functions.

As shown in Tables 5 and 6 we compare the outcomes of ABFIA, and those of FIA, ABC, and basic ABC variants which taken from [58] and [60]. In all algorithms, the stopping criterion is set to $5000 \times D$ number of function evaluations, and the objective function values of less than $1E-60$ are set to zero. For each function, the first and second rows of the tables present the mean and standard deviation of the best values of ten independent runs. The third row shows the rank of each algorithm in this experiment. The algorithms are ranked according to the average objective function values for each test function. For 30-dimensional functions of F1, F5, F6, and F7, the ABFIA outperforms all other algorithms. The ABCx and ABCFWS in

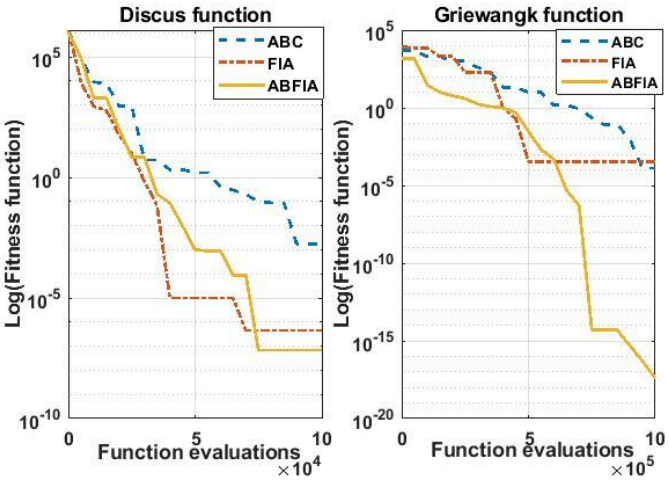

Figure 3: Convergence curves for the ABC, FIA and ABFIA.

F2 and F3 functions, and ABC-Best 1 in F3 provide better solutions than ABFIA.

For solving 60-dimensional instances, the ABFIA outperforms all other algorithms, and ABCFWS and ABCx are ranked second and third, respectively (see Table 6). The results clearly show that the ABFIA outperforms the stand-alone ABC and FIA in all unimodal test functions, as well as the tested algorithms. To show the scalability of our algorithm, we also increased the dimension of the test functions up to 200. Table 9 includes the results of 200-dimensional unimodal test functions and shows the ability of the ABFIA in dealing with high-dimensional problems. We note that we use the maximum number of function evaluations as the stopping criterion for a fair comparison and not the computation time.

5.5. Experiment 2: Multimodal benchmark test functions

In the second experiment, the performance of ABFIA is investigated over a set of 30-, 60- and 200- dimensional multimodal test functions.

Tables 7 and 8 represent the results of 30- and 60-dimensional instances, respectively. The maximum number of function evaluations in all runs is set to $5000 \times D$. Results are averaged over ten independent runs, and values less than $1E-60$ are set to zero. In the tables, for each test function the first and second rows show the average and standard deviation of the best results, and the third row represents the rank of each algorithm. According

Table 5: Average results of ten runs for 30-dimensional unimodal functions

Test Function		ABFIA	FIA	ABC	GABC	ABCBest1	ABCBest2	MABC	ABCX	ABCFWS
	Ave	**0.00E+00**	6.92E-29	5.10E-16	4.62E-16	3.11E-47	5.96E-35	9.43E-32	**0.00E+00**	**0.00E+00**
F1	Std	**0.00E+00**	7.56E-29	8.40E-17	7.12E-17	3.44E-47	3.61E-35	6.67E-32	**0.00E+00**	**0.00E+00**
	Rank	**1.00E+00**	7.00E+00	9.00E+00	8.00E+00	4.00E+00	5.00E+00	6.00E+00	**1.00E+00**	**1.00E+00**
	Ave	2.59E-31	3.14E-17	4.79E-16	3.62E-16	5.35E-24	1.70E-28	3.66E-28	**0.00E+00**	**0.00E+00**
F2	Std	6.49E-31	5.44E-17	9.88E-17	7.88E-17	4.91E-24	2.35E-28	5.96E-28	**0.00E+00**	**0.00E+00**
	Rank	3.00E+00	7.00E+00	9.00E+00	8.00E+00	6.00E+00	4.00E+00	5.00E+00	**1.00E+00**	**1.00E+00**
	Ave	6.40E-38	3.25E-16	5.06E-16	4.55E-16	6.50E-48	5.55E-36	2.10E-32	**0.00E+00**	**0.00E+00**
F3	Std	9.05E-38	5.12E-16	9.20E-17	7.00E-17	6.04E-48	3.36E-36	1.56E-32	**0.00E+00**	**0.00E+00**
	Rank	4.00E+00	7.00E+00	9.00E+00	8.00E+00	3.00E+00	5.00E+00	6.00E+00	**1.00E+00**	**1.00E+00**
	Ave	**0.00E+00**	9.37E-03	1.28E-15	1.35E-15	2.10E-25	1.36E-18	2.40E-17	1.84E-05	1.85E-03
F5	Std	**0.00E+00**	4.55E-03	1.44E-16	1.36E-16	9.08E-26	4.27E-19	9.02E-18	6.62E-06	2.53E-04
	Rank	**1.00E+00**	9.00E+00	5.00E+00	6.00E+00	2.00E+00	3.00E+00	4.00E+00	7.00E+00	8.00E+00
	Ave	**0.00E+00**	2.54E-17	2.01E-16	1.21E-16	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F6	Std	**0.00E+00**	4.13E-16	4.74E-17	3.99E-17	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	**1.00E+00**	7.00E+00	9.00E+00	8.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	**8.53E-11**	3.43E-06	4.32E-02	3.21E-01	1.49E+01	5.45E+00	6.11E-01	3.08E+01	2.89E+00
F7	Std	**8.95E-11**	3.24E-06	4.71E-02	8.21E-01	2.87E+01	8.40E+00	4.55E-01	2.12E+01	2.02E+00
	Rank	**1.00E+00**	2.00E+00	3.00E+00	4.00E+00	8.00E+00	7.00E+00	5.00E+00	9.00E+00	6.00E+00
	Average Rank	1.83E+00	6.50E+00	7.33E+00	7.00E+00	4.00E+00	4.17E+00	4.50E+00	3.50E+00	3.00E+00

Table 6: Average results of ten runs for 60-dimensional unimodal functions

Test Function		ABFIA	FIA	ABC	GABC	ABCBest1	ABCBest2	MABC	ABCx	ABCFWS
	Ave	**0.00E+00**	3.21E-02	1.24E-15	1.06E-15	3.92E-44	4.82E-33	6.03E-29	**0.00E+00**	**0.00E+00**
F1	Std	**0.00E+00**	5.21E-02	1.16E-16	1.21E-16	2.64E-44	2.59E-33	4.31E-29	**0.00E+00**	**0.00E+00**
	Rank	**1.00E+00**	9.00E+00	8.00E+00	7.00E+00	4.00E+00	5.00E+00	6.00E+00	**1.00E+00**	**1.00E+00**
	Ave	**0.00E+00**	4.11E-16	1.15E-15	8.97E-16	1.70E-41	5.86E-27	3.51E-25	**0.00E+00**	**0.00E+00**
F2	Std	**0.00E+00**	7.49E-16	1.50E-16	9.29E-17	9.16E-42	1.13E-26	2.72E-25	**0.00E+00**	**0.00E+00**
	Rank	**1.00E+00**	7.00E+00	9.00E+00	8.00E+00	4.00E+00	5.00E+00	6.00E+00	**1.00E+00**	**1.00E+00**
	Ave	**0.00E+00**	9.81E-12	1.21E-15	1.04E-15	2.06E-44	9.10E-34	1.39E-29	**0.00E+00**	**0.00E+00**
F3	Std	**0.00E+00**	5.37E-12	1.59E-16	1.27E-16	1.83E-44	3.87E-34	8.84E-30	**0.00E+00**	**0.00E+00**
	Rank	**1.00E+00**	9.00E+00	8.00E+00	7.00E+00	4.00E+00	5.00E+00	6.00E+00	**1.00E+00**	**1.00E+00**
	Ave	**4.40E-57**	5.76E-01	2.80E-15	2.96E-15	8.48E-24	1.58E-17	6.96E-16	5.71E-02	1.62E-01
F5	Std	**6.22E-57**	6.21E-01	2.40E-16	1.85E-16	2.31E-24	3.32E-18	1.20E-16	1.08E-02	2.14E-02
	Rank	**1.00E+00**	9.00E+00	5.00E+00	6.00E+00	2.00E+00	3.00E+00	4.00E+00	7.00E+00	8.00E+00
	Ave	**0.00E+00**	7.21E-09	4.89E-16	3.73E-16	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F6	Std	**0.00E+00**	2.61E-09	6.20E-17	6.67E-17	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	**1.00E+00**	9.00E+00	8.00E+00	7.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	**1.41E-16**	6.47E-02	9.28E-02	3.30E+00	5.04E+01	5.10E+01	1.51E+00	7.17E+01	4.34E+01
F7	Std	**1.92E-16**	7.43E-02	1.37E-01	1.28E+01	5.46E+01	3.77E+01	1.34E+00	2.54E+01	1.32E+01
	Rank	**1.00E+00**	2.00E+00	3.00E+00	5.00E+00	7.00E+00	8.00E+00	4.00E+00	9.00E+00	6.00E+00
	Average Rank	1.00E+00	7.50E+00	6.83E+00	6.67E+00	3.67E+00	4.50E+00	4.50E+00	3.33E+00	3.00E+00

to Table 7, and Table 8 except for functions F9, F12, and F16, the ABFIA outperforms all other algorithms in both 30- and 60-dimensional test functions. According to the average rank of the algorithms, the ABFIA shows the best performance, which proves the advantage of hybridizing the ABC with FIA. The results of tests on 200-dimensional benchmark functions are also provided in Table 9, which further show the ability of our proposed algorithm in solving high dimensional multimodal problems.

Table 7: Average results of ten runs for 30-dimensional multimodal functions

Test Function		ABFIA	FIA	ABC	GABC	ABCBest1	ABCBest2	MABC	ABCX	ABCFWS
	Ave	**0.00E+00**	4.13E-07	2.85E-17	1.64E-17	**0.00E+00**	3.00E-46	**0.00E+00**	**0.00E+00**	**0.00E+00**
F4	Std	**0.00E+00**	3.21E-07	9.69E-18	8.07E-18	**0.00E+00**	1.07E-45	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	**1.00E+00**	9.00E+00	8.00E+00	7.00E+00	**1.00E+00**	6.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	**0.00E+00**	3.13E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F8	Std	**0.00E+00**	7.89E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	**1.00E+00**	9.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	3.56E-20	5.72E-19	7.62E-11	3.70E-17	**0.00E+00**	1.81E-08	**0.00E+00**	**0.00E+00**	**0.00E+00**
F9	Std	4.94E-20	2.66E-19	4.18E-10	5.32E-17	**0.00E+00**	6.29E-08	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	5.00E+00	6.00E+00	8.00E+00	7.00E+00	**1.00E+00**	9.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	**4.44E-15**	5.64E-05	1.09E-12	9.42E-02	1.33E-12	1.76E-12	1.21E-13	5.23E-09	7.07E-06
F10	Std	**5.95E-15**	3.11E-05	9.06E-13	5.16E-01	8.18E-13	3.32E-13	4.53E-13	2.86E-08	2.38E-05
	Rank	**1.00E+00**	8.00E+00	3.00E+00	9.00E+00	4.00E+00	5.00E+00	2.00E+00	6.00E+00	7.00E+00
	Ave	**2.18E-24**	4.32E-12	3.79E-14	3.20E-14	3.01E-24	3.07E-14	4.13E-14	1.04E-14	2.38E-14
F11	Std	**1.03E-24**	2.44E-12	3.99E-15	3.36E-15	2.91E-15	3.43E-15	2.17E-15	3.08E-15	3.38E-15
	Rank	**1.00E+00**	9.00E+00	7.00E+00	6.00E+00	2.00E+00	5.00E+00	8.00E+00	3.00E+00	4.00E+00
	Ave	2.78E-26	9.33E-10	8.82E-10	3.41E-09	4.74E-16	9.47E-16	1.58E-16	**9.68E-58**	**1.57E-32**
F12	Std	3.95E-27	6.70E-10	2.19E-09	1.13E-08	1.80E-15	2.46E-15	2.48E-16	**1.52E-57**	**5.57E-48**
	Rank	3.00E+00	8.00E+00	7.00E+00	9.00E+00	5.00E+00	6.00E+00	4.00E+00	**1.00E+00**	2.00E+00
	Ave	**9.70E-04**	9.10E-03	3.27E-01	2.66E-01	2.39E-01	2.81E-01	2.95E-01	8.36E-02	1.81E-01
F13	Std	**1.30E-05**	3.50E-04	4.44E-02	4.39E-02	6.13E-02	3.92E-02	3.17E-02	2.23E-02	3.72E-02
	Rank	**1.00E+00**	2.00E+00	9.00E+00	6.00E+00	5.00E+00	7.00E+00	8.00E+00	3.00E+00	4.00E+00
	Ave	**-7.83E+01**	-7.70E+01	**-7.83E+01**	**-7.83E+01**	**-7.83E+01**	**-7.83E+01**	**-7.83E+01**	**-7.83E+01**	**-7.83E+01**
F14	Std	**5.11E-12**	4.01E-02	**1.02E-10**	**2.94E-14**	**6.68E-12**	**4.86E-09**	**2.06E-07**	**-1.07E-01**	**8.25E-14**
	Rank	**1.00E+00**	9.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	**0.00E+00**	2.45E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F15	Std	**0.00E+00**	5.32E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	**1.00E+00**	9.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	6.55E-20	9.53E-10	8.17E-10	3.33E-17	8.81E-16	1.46E-07	**0.00E+00**	**0.00E+00**	1.54E-04
F16	Std	6.32E-20	3.43E-10	4.47E-09	5.17E-17	3.38E-15	7.78E-07	**0.00E+00**	**0.00E+00**	2.57E-04
	Rank	3.00E+00	7.00E+00	6.00E+00	4.00E+00	5.00E+00	8.00E+00	**1.00E+00**	**1.00E+00**	**9.00E+00**
	Ave	**9.64E-15**	5.92E-12	3.73E-14	3.20E-14	2.89E-14	3.01E-14	4.92E-14	1.39E-14	1.61E-14
F17	Std	**6.74E-15**	1.89E-11	4.45E-15	2.80E-15	2.59E-15	3.70E-15	5.31E-15	2.16E-15	2.23E-15
	Rank	**1.00E+00**	9.00E+00	7.00E+00	6.00E+00	4.00E+00	5.00E+00	8.00E+00	2.00E+00	3.00E+00
	Ave	**2.89E-19**	9.43E-08	1.35E-09	6.65E-08	1.50E-16	1.33E-13	1.38E-16	1.09E-16	8.30E-06
F18	Std	**7.12E-18**	5.77E-08	3.48E-09	2.39E-07	2.48E-16	4.89E-13	8.11E-17	7.76E-17	5.84E-06
	Rank	**1.00E+00**	8.00E+00	6.00E+00	7.00E+00	4.00E+00	5.00E+00	3.00E+00	2.00E+00	9.00E+00
	Average Rank	1.67E+00	7.75E+00	5.33E+00	5.33E+00	2.83E+00	4.92E+00	3.25E+00	1.92E+00	3.58E+00

5.6. Experiment 3: CEC2019

In experiment 3, the ABFIA is examined on CEC2019 test functions. These benchmark functions allow investigating the exploration and exploitation abilities of the algorithms.

Table 10 reports computational results of DA, WOA, LPB, WOAGWO, GWO, and also the average of the best results for ten runs of ABFIA and the stand-alone FIA for each benchmark function. To have a fair comparison with the results of other algorithms, the maximum number of iterations in each run is set to 500. As the table shows, the ABFIA produces superior solutions in 7 out of 10 tested functions, demonstrating the ABFIA's ability to balance exploration and exploitation. Comparing the results of the ABFIA with ABC and FIA shows the impact of hybridization of these two algorithms,

Table 8: Average results of ten runs for 60-dimensional multimodal functions

Test Function		ABFIA	FIA	ABC	GABC	ABCBest1	ABCBest2	MABC	ABCx	ABCFWS
	Ave	**0.00E+00**	1.23E-01	4.31E-17	2.85E-17	0.00E+00	7.53E-39	3.00E-59	0.00E+00	0.00E+00
F4	Std	**0.00E+00**	9.34E-02	1.47E-17	1.01E-17	0.00E+00	3.95E-38	3.87E-59	0.00E+00	0.00E+00
	Rank	**1.00E+00**	9.00E+00	8.00E+00	7.00E+00	**1.00E+00**	6.00E+00	5.00E+00	**1.00E+00**	**1.00E+00**
	Ave	**0.00E+00**	2.56E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F8	Std	**0.00E+00**	4.25E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	**1.00E+00**	9.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	1.17E-19	9.46E-11	3.74E-16	2.47E-04	**0.00E+00**	3.96E-09	**0.00E+00**	**0.00E+00**	**0.00E+00**
F9	Std	1.65E-19	6.73E-11	2.92E-16	1.35E-03	**0.00E+00**	2.04E-08	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	5.00E+00	7.00E+00	6.00E+00	9.00E+00	**1.00E+00**	8.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	**4.44E-14**	4.25E-04	9.55E-01	3.97E+01	3.99E-11	3.95E+00	3.56E-11	7.90E+00	4.72E-05
F10	Std	**5.85E-14**	3.43E-04	5.23E+00	6.47E+01	3.64E-12	2.16E+01	2.18E-12	3.00E+01	7.73E-05
	Rank	**1.00E+00**	5.00E+00	6.00E+00	9.00E+00	3.00E+00	7.00E+00	2.00E+00	8.00E+00	4.00E+00
	Ave	**1.04E-21**	8.33E-11	8.68E-14	7.31E-14	6.93E-14	7.47E-14	1.37E-13	2.74E-14	5.44E-14
F11	Std	**1.44E-21**	5.33E-11	8.48E-15	5.57E-15	5.00E-15	4.12E-15	1.24E-14	3.86E-15	3.86E-15
	Rank	**1.00E+00**	9.00E+00	7.00E+00	5.00E+00	4.00E+00	6.00E+00	8.00E+00	2.00E+00	3.00E+00
	Ave	9.41E-22	3.20E-09	2.31E-07	7.34E-07	5.29E-16	2.23E-11	8.20E-16	2.04E-17	**1.83E-26**
F12	Std	1.31E-22	9.20E-09	7.68E-07	1.70E-06	1.25E-15	3.77E-11	4.69E-16	1.11E-16	**2.04E-26**
	Rank	2.00E+00	7.00E+00	8.00E+00	9.00E+00	4.00E+00	6.00E+00	5.00E+00	3.00E+00	**1.00E+00**
	Ave	**8.30E-03**	3.99E-02	4.76E-01	4.62E-01	4.61E-01	4.68E-01	4.84E-01	2.87E-01	3.93E-01
F13	Std	**4.90E-04**	4.52E-02	7.84E-03	1.79E-02	1.15E-02	9.17E-03	3.62E-03	2.67E-02	6.73E-02
	Rank	**1.00E+00**	2.00E+00	8.00E+00	6.00E+00	5.00E+00	7.00E+00	9.00E+00	3.00E+00	4.00E+00
	Ave	**-7.83E+01**	-7.52E+01	**-7.83E+01**	**-7.83E+01**	**-7.83E+01**	**-7.83E+01**	**-7.83E+01**	**-7.83E+01**	**-7.83E+01**
F14	Std	**4.65E-09**	9.31E-01	**5.77E-09**	**4.89E-14**	**3.71E-11**	**1.76E-08**	**2.40E-07**	**8.85E-02**	**2.11E-13**
	Rank	**1.00E+00**	9.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	**0.00E+00**	4.79E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F15	Std	**0.00E+00**	1.29E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	**1.00E+00**	9.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	5.54E-19	1.14E-10	2.63E-16	6.66E-17	**0.00E+00**	1.44E-08	**0.00E+00**	**0.00E+00**	**0.00E+00**
F16	Std	3.66E-19	6.73E-11	2.71E-16	1.08E-16	**0.00E+00**	7.17E-08	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	5.00E+00	8.00E+00	7.00E+00	6.00E+00	**1.00E+00**	9.00E+00	**1.00E+00**	**1.00E+00**	**1.00E+00**
	Ave	**1.42E-14**	7.76E-11	8.38E-14	7.54E-14	6.90E-14	7.39E-14	2.00E-13	3.45E-14	3.88E-14
F17	Std	**6.32E-14**	2.94E-11	7.20E-15	5.00E-15	4.82E-15	3.54E-15	3.07E-14	3.97E-15	2.27E-15
	Rank	**1.00E+00**	9.00E+00	7.00E+00	6.00E+00	4.00E+00	5.00E+00	8.00E+00	2.00E+00	3.00E+00
	Ave	**2.66E-17**	2.54E-08	1.04E-07	1.24E-05	1.80E-16	2.53E-10	9.71E-16	3.80E-16	4.59E-05
F18	Std	**3.41E-17**	4.32E-08	2.24E-07	5.65E-05	1.17E-16	1.17E-09	5.70E-16	3.09E-16	2.78E-05
	Rank	**1.00E+00**	6.00E+00	7.00E+00	8.00E+00	2.00E+00	5.00E+00	4.00E+00	3.00E+00	9.00E+00
	Average Rank	1.75E+00	7.42E+00	5.58E+00	5.67E+00	2.33E+00	5.17E+00	3.83E+00	2.25E+00	2.50E+00

and according to the average rank of the algorithms, ABFIA outperforms all other methods.

5.7. Statistical analysis

We conduct non-parametric statistical test, including Friedman test [72] and then Holm's post-hoc test[73] as suggested in [74], to show the efficiency of the ABFIA in comparison to other methods tested in the present paper in solving the challenging CEC2019 test functions (see Section 5.6).

We employ the Friedman statistical test with the null hypothesis (H_0) of no significant differences among the compared algorithms, and the alternative hypothesis (H_1) of existence of a significant difference among the algorithms' performance. We choose a typical significant level of 5%, i.e., $\alpha = 0.05$.

The Friedman test is a multiple comparison test that aims to detect signif-

Table 9: Average results of ten runs for 200-dimensional functions

Test Function	Unimodal functions		Test Function	Multimodal functions	
F1	Ave	2.55E-43	F4	Ave	4.31E-55
	Std	3.54E-43		Std	6.71E-55
F2	Ave	2.37E-41	F8	Ave	0.00E+00
	Std	2.61E-41		Std	0.00E+00
F3	Ave	3.47E-23	F9	Ave	2.45E-18
	Std	5.23E-23		Std	3.52E-18
F5	Ave	3.99E-34	F10	Ave	4.94E-08
	Std	4.93E-34		Std	8.85E-08
F6	Ave	4.68E-15	F11	Ave	5.39E-15
	Std	6.65E-15		Std	6.61E-15
F7	Ave	4.68E-09	F12	Ave	5.22E-18
	Std	5.42E-09		Std	4.32E-18
			F13	Ave	1.10E-02
				Std	2.50E-03
			F14	Ave	-7.83E+01
				Std	7.74E-09
			F15	Ave	3.99E-12
				Std	5.11E-12
			F16	Ave	9.73E-23
				Std	4.65E-23
			F17	Ave	3.23E-06
				Std	4.34E-06
			F18	Ave	4.11E-15
				Std	5.72E-15

icant differences between the results of two or more algorithms. The average rank of algorithms and p-value obtained from the Friedman test are shown in Table 11.

The p-value of the Friedman statistical test is equal to $3.00E-06$, which suggests rejecting the null hypothesis, showing therefore significant differences among the tested algorithms. The Friedman statistical test, however, can only conclude significant differences over multiple comparisons and cannot establish superiority of a particular algorithm. Therefore, we use Holm's post-hoc test method to compare ABFIA and other algorithms. Table 11 shows that in all experiments except for the LPB algorithm, the null hypothesis is rejected (p-value < 0.005). Therefore, hybridizing ABC and FIA is superior than the stand-alone ABC and FIA. Also, the results show that the ABFIA is significantly superior to all tested algorithms except the LPB. For the LPB algorithm ,test did not reject the null hypothesis.

Table 10: Average results of ten runs for CEC2019 functions

Test Function		ABFIA	ABC	FIA	DA	WOA	SSA	LPB	WOAGWO	GWO
	Ave	2.92E+05	6.35E+07	1.25E+08	5.43E+10	4.11E+10	6.05E+10	7.49E+09	**4.76E+04**	2.13E+08
F21	Std	8.42E+05	5.49E+07	1.21E+08	6.69E+10	5.42E+10	4.75E+10	8.14E+09	**5.19E+03**	3.07E+08
	Rank	2.00E+00	3.00E+00	4.00E+00	8.00E+00	7.00E+00	9.00E+00	6.00E+00	**1.00E+00**	5.00E+00
	Ave	**1.73E+01**	1.83E+01	1.73E+01	7.80E+01	1.73E+01	1.83E+01	1.76E+01	1.83E+01	1.83E+01
F22	Std	**7.14E-15**	4.64E-03	1.88E-03	8.78E+01	4.50E-03	5.00E-04	3.19E-01	4.72E-04	3.04E-04
	Rank	**1.00E+00**	5.00E+00	1.00E+00	9.00E+00	1.00E+00	5.00E+00	4.00E+00	5.00E+00	5.00E+00
	Ave	1.27E+01	1.37E+01	1.37E+01	1.37E+01	1.37E+01	1.37E+01	**1.27E+01**	1.37E+01	1.37E+01
F23	Std	1.88E-15	5.47E-07	3.92E-09	7.00E-04	0.00E+00	3.00E-04	**0.00E+00**	1.83E-05	1.92E+00
	Rank	1.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	**1.00E+00**	3.00E+00	3.00E+00
	Ave	**2.08E+01**	3.24E+02	5.35E+02	3.44E+02	3.95E+02	4.17E+01	7.79E+01	2.54E+02	3.01E+02
F24	Std	**1.07E+01**	4.35E+02	1.10E+02	4.14E+02	2.49E+02	2.22E+01	2.99E+01	5.39E+02	6.87E+02
	Rank	**1.00E+00**	6.00E+00	9.00E+00	7.00E+00	8.00E+00	2.00E+00	3.00E+00	4.00E+00	5.00E+00
	Ave	**1.08E+00**	1.17E+00	1.91E+00	2.56E+00	2.73E+00	2.21E+00	1.19E+00	2.43E+00	2.43E+00
F25	Std	**4.38E-02**	4.53E-01	7.18E-02	3.25E-01	2.92E-01	1.06E-01	1.09E-01	2.62E-01	2.52E-01
	Rank	**1.00E+00**	2.00E+00	4.00E+00	8.00E+00	9.00E+00	5.00E+00	3.00E+00	6.00E+00	7.00E+00
	Ave	6.46E+00	1.12E+01	2.33E+01	9.90E+00	1.07E+01	6.08E+00	**3.74E+00**	1.14E+01	1.19E+01
F26	Std	3.71E+00	5.42E+00	5.69E+00	1.64E+00	1.03E+00	1.49E+00	**8.23E-01**	1.64E+00	7.31E-01
	Rank	3.00E+00	6.00E+00	9.00E+00	4.00E+00	5.00E+00	2.00E+00	**1.00E+00**	7.00E+00	8.00E+00
	Ave	**4.03E+01**	2.77E+02	3.66E+02	5.79E+02	4.91E+02	4.10E+02	1.45E+02	5.88E+02	5.35E+02
F27	Std	**1.17E+02**	1.56E+02	1.64E+02	3.29E+02	1.95E+02	2.91E+02	1.78E+02	3.49E+02	2.92E+02
	Rank	**1.00E+00**	3.00E+00	4.00E+00	8.00E+00	6.00E+00	5.00E+00	2.00E+00	9.00E+00	7.00E+00
	Ave	**4.36E+00**	6.46E+00	4.94E+00	6.87E+00	6.91E+00	6.37E+00	4.89E+00	5.59E+00	5.40E+00
F28	Std	**5.45E-01**	1.56E+00	3.65E+00	5.02E-01	4.27E-01	5.86E-01	6.79E-01	1.02E+00	9.94E-01
	Rank	**1.00E+00**	7.00E+00	3.00E+00	8.00E+00	9.00E+00	6.00E+00	2.00E+00	5.00E+00	4.00E+00
	Ave	**2.35E+00**	3.15E+00	6.84E+00	6.05E+00	5.94E+00	3.67E+00	2.89E+00	5.67E+00	1.47E+01
F29	Std	**1.02E-01**	3.66E+00	4.35E-01	2.87E+00	1.66E+00	2.36E-01	2.31E-01	8.81E-01	5.00E+01
	Rank	**1.00E+00**	3.00E+00	8.00E+00	7.00E+00	6.00E+00	4.00E+00	2.00E+00	5.00E+00	9.00E+00
	Ave	**1.58E+01**	2.11E+01	2.14E+01	2.13E+01	2.13E+01	2.10E+01	2.00E+01	2.16E+01	2.15E+01
F30	Std	**9.07E+00**	5.64E-02	5.04E-02	1.72E-01	1.11E-01	7.80E-02	2.33E-03	9.22E-02	6.85E-02
	Rank	**1.00E+00**	4.00E+00	7.00E+00	5.00E+00	6.00E+00	3.00E+00	2.00E+00	9.00E+00	8.00E+00

6. Concluding remarks

In this paper, a hybrid optimization algorithm called ABFIA was proposed. This algorithm is a combination of the FIA and ABC algorithms and uses the powerful exploration capability of ABC and the fast convergence ability of FIA. The proposed ABFIA was tested on various benchmark functions with up to 200 dimensions. According to the outcomes of the comprehensive experimental tests, the hybridization of ABC and FIA ensures an excellent trade-off between exploration and exploitation of the ABC, improves the global exploration of the FIA, and reinforces the local exploitation ability of the ABC. Our results demonstrate that the ABFIA is a leading method for solving the tested functions.

Future research directions may include utilizing learning methods to improve parameter tuning of the proposed ABFIA. Also, binary and multi-

Table 11: Results of the statistical test while comparing ABFIA and other optimization algorithms on CEC2019 functions

Algorithm	Average rank	Holm p-value (vs. ABFIA)
ABFIA	1.30E+00	-
LPB	2.70E+00	0.1224985
ABC	4.20E+00	0.0035477
SSA	4.60E+00	0.0012617
FIA	5.30E+00	0.0001049
WOAGWO	5.60E+00	3.66E-05
WOA	6.00E+00	7.50E-06
GWO	6.20E+00	3.52E-06
DA	6.80E+00	2.41E-07
Friedman test p-value	3.00E-06	-

objective versions of the ABFIA method can be investigated. Finally, combining the FIA with other metaheuristics to further examine the added benefits of the FIA to these metaheuristics is of particular research interest.

References

[1] M. Bellare, P. Rogaway, The complexity of approximating a nonlinear program, Mathematical Programming 69 (1995) 429–441. URL: https://doi.org/10.1007/BF01585569. doi:10.1007/BF01585569.

[2] V. Gergel, A Global Optimization Algorithm for Multivariate Functions with Lipschitzian First Derivatives, Journal of Global Optimization 10 (1997) 257–281.

[3] T. Nguyen, Z. Li, S. Zhang, T. Truong, A hybrid algorithm based on particle swarm and chemical reaction optimization, Expert Systems with Applications: An International Journal 41 (2014) 2134–2143. doi:10.1016/j.eswa.2013.09.012.

[4] H. T. Reiner Horst, Global Optimization, 3rd ed., Springer-Verlag Berlin Heidelberg, 1996.

[5] Immanuel Bomze, Tibor Csendes, R. Horst, Panos Pardalos, Developments in Global Optimization, 1st ed., Springer US, 1997.

[6] Z. Li, W. Wang, Y. Yan, Z. Li, Ps–abc: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems, Expert Systems with Applications 42 (2015) 8881 – 8895. URL: http://www.sciencedirect.com/science/article/pii/S0957417415005035. doi:https://doi.org/10.1016/j.eswa.2015.07.043.

[7] D. S. W. G. G. Rizk-Allah Rizk M., EI-Sehiemy R A., Hybridizing sine cosine algorithm with multi-orthogonal search strategy for engineering design problems, A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor 73 (2016) 1–22.

[8] R. M. Rizk-Allah, R. A. El-Sehiemy, G.-G. Wang, A novel parallel hurricane optimization algorithm for secure emission/economic load dispatch solution, Applied Soft Computing 63 (2018) 206 – 222. URL: http://www.sciencedirect.com/science/article/pii/S1568494617307160. doi:https://doi.org/10.1016/j.asoc.2017.12.002.

[9] E. López-Rubio, A. Yurtkuran, E. Emel, An enhanced artificial bee colony algorithm with solution acceptance rule and probabilistic multisearch, Computational Intelligence and Neuroscience 2016 (2016). URL: https://doi.org/10.1155/2016/8085953. doi:10.1155/2016/8085953.

[10] H. M. Alshamlan, G. H. Badr, Y. A. Alohali, Genetic bee colony (gbc) algorithm: A new gene selection method for microarray cancer classification, Computational Biology and Chemistry 56 (2015) 49 – 60. URL: http://www.sciencedirect.com/science/article/pii/S147692711500047X. doi:https://doi.org/10.1016/j.compbiolchem.2015.03.001.

[11] K. Premalatha, A. Natarajan, Hybrid pso and ga for global maximization, Int J Open Prob Compt Math 2 (2009).

[12] S. Fidanova, M. Paprzycki, O. Roeva, Hybrid ga-aco algorithm for a model parameters identification problem, in: 2014 Federated Conference on Computer Science and Information Systems, FedCSIS 2014, 2014, pp. 413–420. doi:10.15439/2014F373.

[13] Z. Meng, J.-S. Pan, H. Xu, Quasi-affine transformation evolutionary (quatre) algorithm: A cooperative swarm based algorithm for global optimization, Knowledge-Based Systems 109 (2016) 104 – 121. URL: http://www.sciencedirect.com/science/article/pii/S0950705116302003. doi:https://doi.org/10.1016/j.knosys.2016.06.029.

[14] A. Sharma, A. Sharma, S. Choudhary, R. Pachauri, A. Shrivastava, D. Kumar, A review on artificial bee colony and it's engineering applications, Journal of Critical Reviews 7 (2020) 2020. doi:10.31838/jcr.07.11.558.

[15] B. T. Tezel, A. Mert, A cooperative system for metaheuristic algorithms, Expert Systems with Applications 165 (2021) 113976. URL: http://www.sciencedirect.com/science/article/pii/S0957417420307545. doi:https://doi.org/10.1016/j.eswa.2020.113976.

[16] J. Nocedal, S. J. Wright (Eds.), Line Search Methods, Springer New York, New York, NY, 1999, pp. 34–63. URL: $https://doi.org/10.1007/0-387-22742-3_3$. doi:$10.1007/0-387-22742-3_3$.

[17] E. Chong, S. Zak, An Introduction to Optimization, Wiley Series in Discrete Mathematics and Optimization, Wiley, 2013. URL: https://books.google.nl/books?id=iD5s0iKXHP8C.

[18] S. Das, P. Koduru, Min Gui, M. Cochran, A. Wareing, S. M. Welch, B. R. Babin, Adding local search to particle swarm optimization, in: 2006 IEEE International Conference on Evolutionary Computation, 2006, pp. 428–433.

[19] J. Wang, Y. Zhou, Quantum-behaved particle swarm optimization with generalized local search operator for global optimization, in: D.-S. Huang, L. Heutte, M. Loog (Eds.), Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 851–860.

[20] Junying Chen, Zheng Qin, Yu Liu, Jiang Lu, Particle swarm optimization with local search, in: 2005 International Conference on Neural Networks and Brain, volume 1, 2005, pp. 481–484.

[21] A. Etminaniesfahani, A. Ghanbarzadeh, Z. Marashi, Fibonacci indicator algorithm: A novel tool for complex optimization problems, Engineering Applications of Artificial Intelligence 74 (2018) 1 – 9. URL: http://www.sciencedirect.com/science/article/pii/S0952197618300915. doi:https://doi.org/10.1016/j.engappai.2018.04.012.

[22] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers & Operations Research 13 (1986) 533–549. URL: https://www.sciencedirect.com/science/article/pii/0305054886900481. doi:https://doi.org/10.1016/0305-0548(86)90048-1, applications of Integer Programming.

[23] S. Kirkpatrick, Optimization by simulated annealing: Quantitative studies, Journal of Statistical Physics 34 (1984) 975–986. URL: https://doi.org/10.1007/BF01009452. doi:10.1007/BF01009452, applications of Integer Programming.

[24] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1997) 67–82.

[25] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, Advances in Engineering Software 69 (2014) 46 – 61. URL: http://www.sciencedirect.com/science/article/pii/S0965997813001853. doi:https://doi.org/10.1016/j.advengsoft.2013.12.007.

[26] S. Mirjalili, A. Lewis, The whale optimization algorithm, Advances in Engineering Software 95 (2016) 51 – 67. URL: http://www.sciencedirect.com/science/article/pii/S0965997816300163. doi:https://doi.org/10.1016/j.advengsoft.2016.01.008.

[27] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680. doi:10.1126/science.220.4598.671.

[28] B. Webster, P. Bernhard, A local search optimization algorithm based on natural principles of gravitation, in: IKE, 2003, pp. 671–680.

[29] H. Shah-Hosseini, Principal components analysis by the galaxy-based search algorithm: A novel metaheuristic for continuous optimisation, International Journal of Computational Science and Engineering 6 (2011) 132–140. doi:10.1504/IJCSE.2011.041221.

[30] B. Vahidi, A. Foroughi Nematolahi, Physical and physic-chemical based optimization methods: A review, Journal of Soft Computing in Civil Engineering 3 (2019) 12–27. doi:10.22115/*scce*.2020.214959.1161.

[31] B. Alatas, U. Can, Physics based metaheuristic optimization algorithms for global optimization, American Journal of Information Science and Computer Engineering (2015) 1–14.

[32] J. H. Holland, Genetic algorithms, Scientific American (1992).

[33] R. Storn, K. Price, Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces, Journal of Global Optimization 23 (1995).

[34] I. Rechenberg, Evolutionsstrategien, in: B. Schneider, U. Ranft (Eds.), Simulationsmethoden in der Medizin und Biologie, Springer Berlin Heidelberg, Berlin, Heidelberg, 1978, pp. 83–114.

[35] D. Vasiljevic, Comparison of optimization algorithms, Springer US, Boston, MA, 2002, pp. 83–88. URL: *https* : *//doi.org*/10.1007/978 − 1 − 4615 − 1051 − 2_5. doi:10.1007/978 − 1 − 4615 − 1051 − 2_5.

[36] T. Tušar, B. Filipič, Differential evolution versus genetic algorithms in multiobjective optimization, in: Evolutionary Multi-Criterion Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 257–271.

[37] M. Dorigo, C. Blum, Ant colony optimization theory: A survey, Theoretical Computer Science 344 (2005) 243 – 278. URL: `http://www.sciencedirect.com/science/article/pii/S0304397505003798`. doi:*https* : *//doi.org*/10.1016/*j.tcs*.2005.05.020.

[38] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.

[39] D. Karaboga, An idea based on honey bee swarm for numerical optimization, technical report - tr06, Technical Report, Erciyes University (2005).

[40] M. Dorigo, G. Di Caro, L. M. Gambardella, Ant algorithms for discrete optimization." artificial life 5, 137-172, Artificial Life 5 (1999) 137–172. doi:10.1162/106454699568728.

[41] V.Selvi, R.Umarani, Comparative analysis of ant colony and particle swarm optimization techniques, International Journal of Computer Applications 5 (2010) 1–6. doi:10.5120/908-1286.

[42] M. El-Abd, A hybrid abc-spso algorithm for continuous function optimization, in: 2011 IEEE Symposium on Swarm Intelligence, 2011, pp. 1–6. doi:10.1109/SIS.2011.5952576.

[43] S. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, Neural Computing and Applications 27 (2015) 1053–1073.

[44] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, S. M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, Advances in Engineering Software 114 (2017) 163–191. URL: https://www.sciencedirect.com/science/article/pii/S0965997816307736. doi:https://doi.org/10.1016/j.advengsoft.2017.07.002.

[45] S. Mirjalili, The ant lion optimizer, Advances in Engineering Software 83 (2015) 80–98. URL: https://www.sciencedirect.com/science/article/pii/S0965997815000113. doi:https://doi.org/10.1016/j.advengsoft.2015.01.010.

[46] K. C. Kilic Haydar., Yuzgec Ugur., A novel improved antlion optimizer algorithm and its comparative performance, Neural Computing and Applications 32 (2020) 3803–3824.

[47] S. Mostafa Bozorgi, S. Yazdani, Iwoa: An improved whale optimization algorithm for optimization problems, Journal of Computational Design and Engineering 6 (2019) 243–259. URL: https://www.sciencedirect.com/science/article/pii/S2288430018301994. doi:https://doi.org/10.1016/j.jcde.2019.02.002.

[48] L. S.-X. Wang Jie-Sheng., An Improved Grey Wolf Optimizer Based on Differential Evolution and Elimination Mechanism, Scientific Reports 9 (2019) 71–81.

[49] Z. W. Geem, J. Kim, G. V. Loganathan, A new heuristic optimization algorithm: Harmony search, Simulation 76 (2001) 60 – 68.

[50] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, USA, 1997.

[51] P. F. Pai, W. Sun, X. Chang, Comparative analysis of ant colony and particle swarm optimization techniques, Journal of Electrical and Computer Engineering 2015 (2015) 712–753. doi:10.1155/2015/753712.

[52] C. M. Rahman, T. A. Rashid, A new evolutionary algorithm: Learner performance based behavior algorithm, Egyptian Informatics Journal 22 (2021) 213–223. URL: https://www.sciencedirect.com/science/article/pii/S1110866520301419. doi:https://doi.org/10.1016/j.eij.2020.08.003.

[53] W. Gao, S. Liu, L. Huang, A global best artificial bee colony algorithm for global optimization, Journal of Computational and Applied Mathematics 236 (2012) 2741 – 2753. URL: http://www.sciencedirect.com/science/article/pii/S0377042712000246. doi:https://doi.org/10.1016/j.cam.2012.01.013.

[54] K. Hussain, M. N. Mohd Salleh, S. Cheng, Y. Shi, R. Naseem, Artificial bee colony algorithm: A component-wise analysis using diversity measurement, Journal of King Saud University - Computer and Information Sciences 32 (2020) 794–808. URL: https://www.sciencedirect.com/science/article/pii/S1319157818302039. doi:https://doi.org/10.1016/j.jksuci.2018.09.017.

[55] S. Imamura, T. Kaihara, N. Fujii, D. Kokuryo, , A. Kitamura, Characteristic analysis of artificial bee colony algorithm with network-structure, Journal of Advanced Computational Intelligence and Intelligent Informatics 21 (2017) 496–506. doi:10.20965/jaciii.2017.p0496.

[56] D. Yazdani, M. R. Meybodi, A novel artificial bee colony algorithm for global optimization, in: 2014 4th International Conference on Computer and Knowledge Engineering (ICCKE), 2014, pp. 443–448. doi:10.1109/ICCKE.2014.6993393.

[57] S. Anuar, A. Selamat, R. Sallehuddin, A modified scout bee for artificial bee colony algorithm and its performance on optimization problems, Journal of King Saud University - Computer and Information Sciences 28 (2016) 395 – 406. URL: http://www.sciencedirect.com/science/article/pii/S1319157816300039. doi:https://doi.org/10.1016/j.jksuci.2016.03.001.

[58] H. Hakli, M. Kıran, An improved artificial bee colony algorithm for balancing local and global search behaviors in continuous optimization, International Journal of Machine Learning and Cybernetics 11 (2020) 2051–2076. doi:10.1007/s13042-020-01094-7.

[59] X. He, W. Wang, J. Jiang, L. Xu, An improved artificial bee colony algorithm and its application to multi-objective optimal power flow, Energies 8 (2015) 2412–2437. doi:10.3390/en8042412.

[60] Y. Celik, An enhanced artificial bee colony algorithm based on fitness weighted search strategy, Automatika 62 (2021) 300–310. URL: https://doi.org/10.1080/00051144.2021.1938477. doi:10.1080/00051144.2021.1938477. arXiv:https://doi.org/10.1080/00051144.2021.1938477.

[61] G. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Applied Mathematics and Computation 217 (2010) 3166–3173. URL: https://www.sciencedirect.com/science/article/pii/S0096300310009136. doi:https://doi.org/10.1016/j.amc.2010.08.049.

[62] W. feng Gao, S. yang Liu, A modified artificial bee colony algorithm, Computers & Operations Research 39 (2012) 687–697. URL: https://www.sciencedirect.com/science/article/pii/S0305054811001699. doi:https://doi.org/10.1016/j.cor.2011.06.007.

[63] L.-C. Lien, M.-Y. Cheng, A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization, Expert Systems with Applications 39 (2012) 9642 – 9650. URL: http://www.sciencedirect.com/science/article/pii/S0957417412003971. doi:https://doi.org/10.1016/j.eswa.2012.02.134.

[64] N. Imanian, M. E. Shiri, P. Moradi, Velocity based artificial bee colony algorithm for high dimensional continuous optimization problems, Engineering Applications of Artificial Intelligence 36 (2014) 148 – 163. URL: http://www.sciencedirect.com/science/article/pii/S0952197614001808. doi:https://doi.org/10.1016/j.engappai.2014.07.012.

[65] S. Kumar, V. Kumar Sharma, R. Kumari, A Novel Hybrid Crossover based Artificial Bee Colony Algorithm for Optimization Problem, International Journal of Computer Applications 82 (2013) 18–25. doi:10.5120/14136-2266. arXiv:1407.5574.

[66] S.-M. Chen, A. Sarosh, Y.-F. Dong, Simulated annealing based artificial bee colony algorithm for global numerical optimization, Applied Mathematics and Computation 219 (2012) 3575–3589. URL: https://www.sciencedirect.com/science/article/pii/S0096300312009514. doi:https://doi.org/10.1016/j.amc.2012.09.052.

[67] H. Mohammed, T. Rashid, A novel hybrid gwo with woa for global numerical optimization and solving pressure vessel design, Neural Computing and Applications (2020) 14701–14718. doi:10.36227/techrxiv.11916369.v1.

[68] M. Jamil, X. Yang, A literature survey of benchmark functions for global optimisation problems, ArXiv abs/1308.4008 (2013).

[69] K. Hussain, M. Salleh, S. Cheng, R. Naseem, Common benchmark functions for metaheuristic evaluation: A review, International Journal on Informatics Visualization 1 (2017) 218–223. doi:10.30630/joiv.1.4-2.65.

[70] K. V. Price, N. H. Awad, M. Z. Ali, P. N. Suganthan, The 100-Digit Challenge: Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization, Technical Report, School Elect. Electron. Eng., Nanyang Technol. Univ., Singapore, 2018.

[71] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Applied Mathematics and Computation 214 (2009) 108 – 132. URL: http://www.sciencedirect.com/science/article/pii/S0096300309002860. doi:https://doi.org/10.1016/j.amc.2009.03.090.

[72] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Annals of Mathematical Statistics 11 (1940) 86–92.

[73] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.

[74] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (2011) 3–18. URL: https://www.sciencedirect.com/science/article/pii/S2210650211000034. doi:https://doi.org/10.1016/j.swevo.2011.02.002.