Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

A modified version of the ABC algorithm and evaluation of its performance

Kaylash Chand Chaudhary

The University of the South Pacific, Suva, Fiji

ARTICLE INFO

Keywords:
Optimization
Selection
Strategy
Learning
Algorithm

ABSTRACT

The artificial bee colony (ABC) optimization algorithm has been widely used to solve the global optimization problems. Many versions of ABC algorithm exist in the literature intending to achieve optimum solution for problems in different domains. Some modifications of the ABC algorithm are general and can be applied to any problem domain, while some are application dependent. This paper proposes a modified version of the ABC algorithm named as, MABC-SS (modified artificial bee colony algorithm with selection strategy), that can be applied to any problem domain. The algorithm is modified in terms of population initialization and update of a bee position using the old and a new food source equation based on the algorithm's performance in the previous iteration. The selection strategy is measured based on a novel approach called the *rate of change*. The population initialization in any optimization algorithm plays an important role in achieving the global optimum. The algorithm proposed in the paper uses random and an opposition-based learning technique to initialize the population and updates a bee's position after exceeding a certain number of trial limits. The rate of change is based on the average cost and is calculated for the past two iterations and compared for a method to be used in the current iteration to achieve the best result. The proposed algorithm is experimented with 35 benchmark test functions and 10 real world test functions. The findings indicate that the proposed algorithm is able to achieve the optimal result in most cases. The proposed algorithm is compared with the original ABC algorithm, modified versions of the ABC algorithm, and other algorithms in the literature using the test mentioned above. The parameters such as population size, number of iterations and runs were kept same for comparison with non-variants of ABC. In case of ABC variants, ABC specific parameters such as abandonment limit factor (0.6) and acceleration coefficient (1) were kept same. Results indicate that in 40% of the traditional benchmark test functions, the suggested algorithm works better than other variants of ABC (ABC, GABC, MABC, MEABC, BABC, and KFABC), while 30% of the traditional benchmark test functions are comparable. The proposed algorithm was compared to non-variants of ABC as well. The results show that the proposed algorithm achieved the best mean result in 50% of the CEC2019 benchmark test functions and in 94% of the classical benchmark test functions. The result is confirmed by Wilcoxon sum ranked test which shows that MABC-SS achieved statistically significant result in 48% of the classical and 70% of the CEC2019 benchmark test functions when compared with the original ABC. Overall, based on assessment and comparison in benchmark test functions used in this paper, the suggested algorithm is superior to others.

E-mail address: kaylash.chaudhary@usp.ac.fj.

https://doi.org/10.1016/j.heliyon.2023.e16086
Received 21 November 2022; Received in revised form 4 May 2023; Accepted 4 May 2023

Available online 10 May 2023

1. Introduction

Complex optimization problems have become popular due to the rapid development of economy and society [35]. Nowadays, Optimization problems have multiple dimensions, objectives, constraints and optima. The optimization problem is the problem of finding the best solution among all possible solutions. The Optimization algorithms are designed to solve a variety of global optimization problems. Optimization problems can be classified as single or multiobjective, constrained or unconstrained and combinatorial [16]. There are different domains of optimization problems, such as ICT, engineering, mining, and scheduling, to name a few [16]. The objective of any optimization algorithm is to either find a global optimum or find a global best solution. While finding a global optimum can be computationally expensive, global best solution can be picked from a pool of solutions. The complexity of optimization problems is evolving in the modern world. Therefore, optimization algorithms must effectively evolve to address a growing set of problems and their computationally expensive global optimums [7].

There is a continuous effort from the research community to improve the algorithms for different optimization problems [7] [26]. Different types of algorithms can solve optimization problems [6]. These are heuristics and metaheuristics, and this paper concentrates on the metaheuristics algorithm. The metaheuristics algorithm has been further classified as evolutionary, swarm-based and trajectory based. This research adopts the swarm-based algorithm. Some examples of swarm-based algorithms include artificial bee colony optimization (ABC), ant colony optimization (ACO), firefly algorithm (FA), and particle swarm optimization (PSO) [6] [29]. This paper focuses on ABC because it is recent, and its performance is better than any other approach [20]. Kabora developed the ABC algorithm in 2005 [19]. Since then, the ABC algorithm's performance with regard to unconstrained optimization problems and its extension have been thoroughly studied by Karaboga and Basturk [20]. ABC is a popular algorithm and now, there are many different variants available in the literature [3] [12] [11]. A colony of bees is split into three categories according to the ABC algorithm: employed bees (forager bees), onlooker bees (observer bees), and scouts. There is only one employed bee per food supply. In other words, the quantity of food sources and employed bees is identical. The employed bee of a food site that has been abandoned is made to act as a scout, aimlessly looking for new food sources. In order for observer bees to select a food source to scavenge, employed bees in a colony communicate information to onlooker bees.

Although ABC was seen to solve most optimization problems, there were modifications carried out. The modification of the algorithm was to improve the performance in terms of achieving global optimum, faster convergence of the algorithm, convergence of the swarm and the global best [11]. An algorithm is modified based on search equations, strategies and selection techniques [14] [11] [22] [30]. All these techniques deal with the exploration and exploitation capability of the algorithm. ABC has good exploration but suffers from exploitation [12]. The two search equations, exploration and exploitation, need to be properly controlled in the algorithm using some search strategy. The algorithm will only control if it learns about its behaviour, such as if exploration is not working, then change the strategy with exploitation. The change in strategy is possible through a selection technique. The selection technique will select a strategy that eventually selects appropriate search equations. The selection technique is based on the average cost value of the swarm and will be compared with the average cost value of the previous iteration to determine the search equation to be used in the next iteration. While literature shows research based modification of ABC based on search equations and population initialization [14] [11] [22] [30] [12], according to the authors knowledge, there is no such technique in the literature that has been applied on ABC. This research will focus on selecting search equations using a selection technique, modification of search equations and population initialization.

This paper presents a new modification of ABC algorithm to solve problems of different characteristics. Problems can be from any domain and exhibit characteristics like being continuous or discontinuous, separable or nonseparable, convex or non convex, differentiable or nondifferentiable and multimodal or unimodal. The modification is based on population initialization techniques, search equations, strategies and selection techniques. The research objectives of this paper are:

- Modify the standard artificial bee colony optimization algorithm that can effectively that can find better solutions to different types of global optimization problems.
- Compare the proposed algorithm with other variants of artificial bee colony optimization algorithms from the literature
- Compare the proposed algorithm with other optimization algorithms from the literature

The rest of the paper is organized as follows. The next section will present a brief review of the modification of the ABC algorithm. Section 3 presents the contribution. Section 4 lays out the proposed algorithm, strategies, and selection technique. The results of the proposed algorithm on 35 classical and 10 CEC2019 benchmark test functions are discussed in Section 5. Section 6 compares the proposed algorithm with the other variants of ABC and some other optimization algorithms. This paper concludes in Section 7.

2. Literature review

Many different types of metaheuristics algorithms have been proposed in the literature for solving global optimization problems. One such algorithm is artificial bee colony optimization with many variants. Performance enhancement from the standard ABC was the main reason for the evolution of the many kinds of ABC variants in literature [38][33][4]. This section will outline recent variants, particularly describing the changes made with respect to standard ABC and present a brief comparison with the algorithm proposed in this research paper.

The bees can either perform a local search (exploitation) or a global search (exploration) for solutions in the ABC algorithms. Most variants proposed in the literature revolve around modifying the search strategy. Zhu and Kwong [38] improved the exploitation capability by adding a new term, gbest, to search equation. According to the authors, the new term will drive the candidate solution to the global best solution. The experimental findings on a collection of numerical benchmark functions indicate that the GABC algorithm outperforms the ABC algorithm in the majority of tests. Banharnsakun et al. used a new method to calculate a candidate solution by using the best solution and its fitness value [3]. The authors also used the adjustable search radius method to avoid solutions getting trapped in the local optimum [3]. The algorithm shares the best possible solutions globally among the entire population. The algorithm has been tested on numerical benchmark problems and image registration applications and it performed well in comparison with the original ABC. Xue et al. used the global best solution to calculate the candidate solution based on the algorithm's different strategies [34]. Experiments are carried out on 25 benchmark functions. The findings show that SABC-GB outperforms the other algorithms when it comes to handling complex optimization problems. Cui et al. used a depth first search method and modified the search equation to improve the algorithm's performance [5]. The authors used two different search equations for employed and onlooker bees. In terms of solution quality, robustness, and convergence speed, the algorithm outperformed the compared algorithms on the majority of the test functions in contrast to other ABC variants and some non-ABC methods. Xiang et al. introduced a novel search equation inspired by PSO [32]. The search equations consisted of information on the global solution and other best solutions. The algorithm was compared with 5 state-of-the-art swarm based algorithms on CEC05 and BBOB12 benchmark functions and the results showed that the proposed algorithm was comparable to its competitors. Gao et al. revised the search equations to include the best solution and to have the random solution [9]. The authors employed the probability to control the switching frequency from one equation to another. The results show that the proposed algorithm performed better in a suite of unimodal/multimodal benchmark functions. Gao et al. created two separate search equations, one each for employed and onlooker bees [11]. Gao et al. [12], proposed MABC algorithm to maintain a balance between exploration and exploitation. Gao et al. used the genetic algorithm to design a novel search equation [13]. This improved the ability to search for ABC. The qABC algorithm was proposed by Karaboga et al., which used a search equation with the best solution in the onlooker bee phase [21]. A Rosenbrock ABC (RABC) was proposed by Kang et al. where the exploitation phase uses a rotational direction method and exploration is dealt with by ABC [18]. Likewise, other variants of ABC deal with the modification or creation of search equations [25].

Moreover, some other techniques have been used to modify the algorithm. The standard ABC algorithm generates initial solutions randomly. Some studies showed that random initialization slows the convergence process and in order to improve the performance of the algorithm, techniques such as chaotic map, orthogonal learning method and opposition based learning method have been applied. Gao et al. used the orthogonal, chaotic and opposition based learning technique to enhance the performance of their algorithm [12]. Wang et al. used three approaches, random, chaotic and opposition based learning, to get the solutions during initialization and the best solution set will then be used by employed and onlooker bees [30]. Likewise, there are other studies in the literature which used different population initialization techniques [4] [15] [37].

Apart from modifying search equations, some research used new strategies to explore solutions. Kiran et al. used five search strategies to update the solutions using the integration of multiple rules [22]. In [33], the authors used a nested search strategy to update solutions. There are two search strategy pools and each pool has three search strategies. Gao et al. used search strategies to construct solutions and employed the Parzen window method to evaluate each solution's quality [10]. The algorithm performed well in different benchmark test functions.

Some researchers have used a combination of the aforementioned techniques - modification of search equations, multiple selection strategy and population initialization [30]. The algorithm had good performance in different benchmark functions when compared to the standard and other variants of ABC. Each technique has its own advantage and can significantly improve the algorithm's performance if properly used in combinations.

Additionally, some researchers have proposed hybrids with ABC to solve the global optimization problems. For example, a hybrid of ABC and GA [36], PSO and ABC [27], and ABC and DE [17]. Likewise others modified algorithms to solve constrained and unconstrained problems. Kumar et al. proposed a variation of ABC called Arrhenius ABC that intended to balance between diversification and intensification process of the algorithm [23]. Over eight global optimization functions without constraints and two constrained problems were tested using the proposed algorithm. The findings demonstrate that, in contrast to basic ABC and its current variations, the aABC algorithm outperforms them for problems that are low dimensional. Nayyar et al. proposed ABC-based SI technique for solving all types of unconstrained optimization problem [24]. The results show that the algorithm was superior to others in solving such problems.

The literature showed that all research on search equations used either same or different equations for employed and onlooker bees. But the employed and onlooker bees do not have choice for selecting different equations at different times. This research also uses modified search equations for both employed and onlooker bees. The search equations are different in both and will not be the same everytime. This is because the employed and onlooker bees have choice to select either the global best or the local best equation. Also, in most research, the population initialization is through random technique whereas in some the initialization is using opposition based learning or chaotic. This research uses random and opposition based learning for population initialization but it also uses opposition based learning technique for scout bees. One research in the literature used the technique of strategies and ways for its selection. This research also uses selection technique through the use of rate of change technique and different ways of selecting strategies which is new as per the author's knowledge. This research is based on the modification of the ABC algorithm. The three techniques mentioned in the literature (population initialization, modification of search equations and selection strategy) will be used by the authors of this paper to improve the performance of the ABC algorithm.

Table 1
Notations.

Notation	Meaning
$f(S_i)$	The fitness of i^{th} solution
$MaxIt$	The number of maximum iterations
n	Population size
S_{best}	The global best solution
S_i	The i^{th} solution
S_k	The randomly selected solution
SO_i	The i^{th} solution generated by opposition based learning
$trial_i$	The trial number of i^{th} solution
$\emptyset_i$	Random number with range [-1, 1]

3. Contribution

The main contributions of this paper are as follows:

- Novel strategies and their selection technique: the selection strategy is based on the rate of change of the fitness value of the global solution. The algorithm will change the strategy if it learns that the rate of change is greater than the old rate of change. The two strategies used in this paper operate on both the employed and onlooker bees. These strategies and their selection technique improve the algorithm's performance and can also solve different types of problems. According to the authors' knowledge, the literature has different strategies and selection techniques, but none of this kind has been used before. The selection technique can be used to learn and improve the performance of any other metaheuristic algorithm by changing the strategy.
- MABC-SS algorithm: the proposed algorithm uses an opposition based learning technique for population initialization and abundant solution, selection strategies and techniques, and modification of search equations to improve the performance of the algorithm.
- Performance analysis: the proposed algorithm has been tested with 35 classical and 10 CEC2019 benchmark test functions. The results show that the optimal value has been achieved in most functions and near-optimal value in the remaining test functions.
- Comparison: The MABC-SS algorithm is compared with six other variants of ABC and four non-variants. The results show that MABC-SS performs better in most classical and CEC2019 benchmark test functions.

4. Related terminologies

Table 1 shows the notations used in the following subsections to describe the artificial bee optimization and the proposed algorithms.

4.1. *Artificial bee colony optimization algorithm*

Karaboga introduced the artificial bee colony (ABC) algorithm in 2005 for solving numerical problems [19]. It is a metaheuristic algorithm inspired by bees' intellectual food searching behaviour. The food sources are considered as solutions in the algorithm. There are three types of bees in this algorithm. The first is the employed bees, whose purpose is to search for new solutions (food) within the neighbourhood of the current solutions. The second is the onlooker bees, also known as unemployed bees. The solutions found by the employed bees are shared with the onlooker bees. The onlooker bees choose their solutions probabilistically. Scout bees are another set of unemployed bees. Employed bees become scouts if their solutions cannot be further improved in a number of trials. The solutions then become redundant and the scout bees start searching for new solutions. The algorithm has four phases:
Initialization phase: A initial set of solutions is randomly formed using the function:

$$S_i = S_{min} + rand\,(0,1)\,.\left(S_{max} - S_{min}\right), \tag{1}$$

where S_i is the i^{th} solution and S_{max}, S_{min} are the upper and lower boundary of the solutions, respectively.
Employed bee solution search phase: The employed bees search for a new solution around i^{th} solution using the function:

$$New_i = \; S_i + \emptyset_i(S_i - S_j), \tag{2}$$

where New_i is the new i^{th} solution, S_i is the i^{th} solution, S_k is the randomly selected solution from the swarm and $\emptyset_i$ is the random number with range [-1, 1]. The cost of the new solution, New_i, will be evaluated and compared with the old solution, S_i. If the cost of the new solution is less than the old one, the new solution will replace the old one in the swarm. Otherwise, the trial number will be increased which will be used in the last phase for abandoning the solution. All other solutions in the swarm follow the same process.
Onlooker bee solution search phase: The onlooker bees search for solutions probabilistically using the technique known as roulette wheel selection. These bees conduct their search based on better solutions. Each solution in the swarm will have a selection probability, computed by the following (equation (3)).

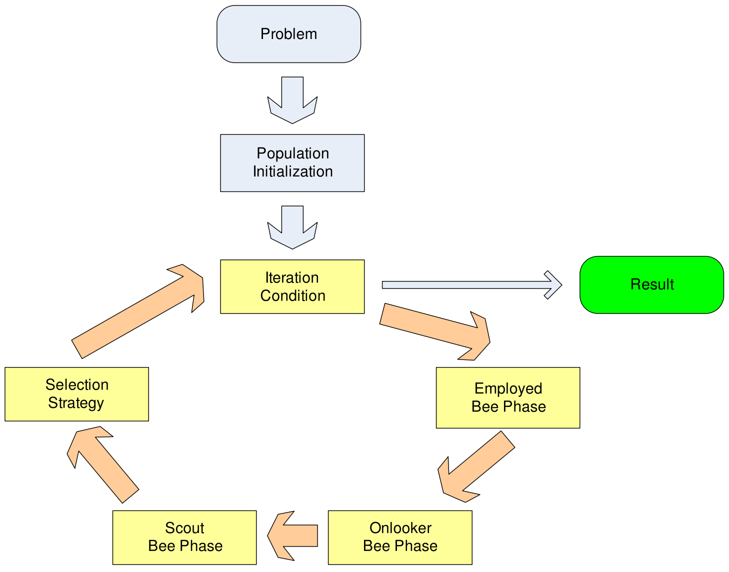


Fig. 1. Model of the proposed algorithm.

$$P\left(S_i\right) = \frac{f(S_i)}{\sum_{i=1}^{n} f(S_i)}, \tag{3}$$

where $P\left(S_i\right)$ is the probability of i^{th} solution and $f(S_i)$ is the fitness value of the i^{th} solution. For a selected solution, the onlooker bees will calculate the new solution using equation (2). The cost of the new solution, New_i, will be evaluated and compared with the old solution, S_i.

Scout bee solution search phase: If the trial number exceeds the limit, the old solution will be removed from the swarm, and the new solution will be computed using equation (1). The swarm will be updated with the new solution.

4.2. *Proposed algorithm – MABC-SS*

The proposed algorithm aims to solve different types of problems using different strategies. The strategy to solve one problem may not work on another problem. The algorithm needs to know the specific strategy that will produce the optimal result to solve a problem. The algorithm will not apriori know this information. Therefore, the algorithm needs to learn whether a given strategy is suitable or not while solving the specific set of problems. If not, then the algorithm should change the strategy. The algorithm to know all this information starts from population initialization. The selection of the strategy is dependent on the swarm that is, the average cost of all solutions.

4.2.1. *MABC-SS model*

Fig. 1 shows the model of the proposed algorithm. The algorithm takes a problem as input and produces the best solution as result. But to get the best solution the algorithm makes loops consisting of employed bee, onlooker bee, scout bee and strategy selection phases. The algorithm will produce an output when the iteration condition will be met.

Population Initialization

A random population initialization may not get the algorithm trapped in local minima but can slow the convergence speed [30]. Opposition based learning (OBL) developed by Tizhoosh [28] provides a significant probability of searching for better solutions. OBL and a random approach are used in the proposed algorithm to initialize the population.

An initial set of solutions is formed using the OBL equation:

$$S_i = S_{min} + \left(S_{max} - S_i\right), \tag{4}$$

where S_i is the i^{th} solution and S_{max}, S_{min} are the upper and lower boundary of the solutions. The initialization procedure is shown in Algorithm 1. A bee's position is first determined by random technique (equation (1)). Then the OBL (equation (4)) is used to calculate the position of a bee. The cost for the both solutions are compared to determine the best position. This process is repeated for the whole swarm of bees.

Strategy Selection Technique

Selecting a suitable strategy is crucial for solving different unimodal and multimodal problems. The algorithm will start with Strategy 1 and change to Strategy 2 if the algorithm learns that Strategy 1 is not suitable for solving the current problem. This change is determined by comparing the current $RateOfChange$ with the previous one. The $RateOfChange$ is the average cost value of the

```
Initialize the range, $S_{max}, S_{min}$
foreach $i = 1: n$ do
    $S_i = S_{min} + rand(0,1).(S_{max} - S_{min})$
    $SO_i = S_{min} + (S_{max} - S_i)$
    $Calculate\ f(S_i)$
    $Calculate\ f(SO_i)$
    if $f(SO_i) \le f(S_i)$ then
        $S_i = SO_i$
    end
    if $f(S_i) \le f(S_{best})$ then
        $S_{best} = S_i$
    end
end
```

Algorithm 1: Initialization.

swarm in one iteration. If the current *RateOfChange* is greater than the previous one, the algorithm will change to Strategy 2. Otherwise, the algorithm will continue with Strategy 1. Every iteration of the algorithm will decide whether to change the strategy or not. The equation for the *RateOfChange* is given below (equation (5):

$$RateOfChange_{it} = \frac{\sum_{i=1}^{n} f(S_i)}{n}, \tag{5}$$

where $RateOfChange_{it}$ is the average fitness value of the swarm in iteration it, $\sum_{i=1}^{n} f(S_i)$ is the sum fitness value of all the solutions in the swarm, S_i is the i^{th} solution, and n is the total number of solutions in the swarm.

This research paper uses two strategies to solve problems. For example, better exploitation speeds up the search process and finds better solutions for unimodal problems [30]. Likewise, a good exploration ability can avoid getting trapped in local minima for multimodal problems. These show that different problems require different strategies or a set of strategies. The two strategies in the research are used by employed and onlooker bees, and do not affect the scout bees. In each algorithm iteration, the strategy for employed bee and onlooker bee can be either 1 or 2 but never both.

Strategy 1: This strategy differs only in terms of the search equation. The employed bees use the following search equation to calculate the position of the new bee:

$$New_{\boldsymbol{i}} = S_{best} + \emptyset_i(S_i - S_j), \tag{6}$$

where New_i is the new i^{th} solution, S_{best} is the global best solution, S_i is the i^{th} solution, S_k is the randomly selected solution from the swarm and $\emptyset_i$ is the random number with range [-1, 1]. Equation (6) supports the local search around the global best solution and provides fast convergence to optimum or near optimum solutions.

The onlooker bees use equation (2) to search around each solution in the swarm and find a new solution.

Strategy 2: This strategy uses both search equations, but the algorithm must decide on which search equation to utilize to find a new solution. Equation (2) will be used if the fitness value of i^{th} solution from the swarm is less than the fitness value of the best solution. This means that the search should be around the i^{th} solution. Otherwise, the search should be local and around the global best solution. In this strategy, both the employed and onlooker bees use the same process. Compared to the first strategy, this strategy gives the option of either searching around a local solution or the global one. There can be four different combinations between employed bees and onlooker bees:

Employed Bee: Equation (2) and the onlooker Bee: Equation (2) – in this combination, the algorithm's behaviour will be exactly like standard ABC. The employed bees search around each solution in the swarm and find a new solution. In contrast, the onlooker bees try to find a better solution by searching around the solution found by the employed bees. The convergence with this combination will be slow, while this combination has an exploration search.

Employed Bee: Equation (2) and onlooker Bee: Equation (6) – the employed bees search around each solution in the swarm and find a new solution. The onlooker bees try to find a better solution by searching around the global best solution. This combination has an exploration search for employed bees and an exploitation search for onlooker bees.

Employed Bee: Equation (6) and onlooker Bee: Equation (2) – this combination is the same as strategy 1. This combination has an exploitation search for employed bees and an exploration search for onlooker bees.

Employed Bee: Equation (6) and onlooker Bee: Equation (6) – the employed and onlooker bees search locally around the global solution. The convergence with this combination will be fast and may find an optimum solution. However, this combination does not offer an exploration search.

4.2.2. *Architecture and working*

The design of MABC-SS is presented in Algorithm 2. The algorithm starts with population initialization and initial solutions. In phase 2, the positions of the employed bees are adjusted based on the two strategies discussed in previous subsection. Then, the fitness of the new solution will be compared with the old solution. If the new solution fitness is better, the old solution in the swarm will be replaced by the new one. Otherwise, the trial will be incremented by one for that solution. Phase 3 is the same as phase 2 except that the employed bees are replaced by onlooker bees. In phase 4, the trial for each solution is compared with the limit, if it is greater, the new solution will be computed using opposition based learning. Also, the rate of change is calculated and compared with the previous iteration rate of change. If the former is greater than the latter, status will be set to 1. Otherwise, it will be set to 0. Assume that the computational time complexity of determining the function value of problem f is $O(f)$. The swarm population is denoted by n. The time complexity of MABC-SS is $O(n.f)$ for each iteration.

```
Phase 1: Initialize bee population, n, using Algorithm 1
while $t < MaxIt$ do
    Phase 2: foreach $i = 1{:}n$ do
        if $Status == 0$ then
            $newbee.pos = Best.pos + \phi(bee_i.pos - bee_j.position)$
        end
        if $Status == 1$ then
            if $f(Best) \le f(bee_i)$ then
                $newbee.position = Best.position + \phi(bee_i.position - bee_j.position)$
            end
            else
                $newbee.position = bee_i.position + \phi(bee_i.position - bee_j.position)$
            end
        end
        if $f(newbee) \le f(bee_i)$ then
            $bee_i = newbee$
        end
        else
            $trial_i = trial_i + 1$
        end
    end
    Phase 3: foreach $i = 1{:}\ n$ do
        if $Status == 0$ then
            $newbee.position = bee_i.position + \phi(bee_i.position - bee_j.position)$
        end
        if $Status == 1$ then
            if $f(Best) <= f(bee_i)$ then
                $newbee.position = Best.position + \phi(bee_i.position - bee_j.position)$
            end
            else
                $newbee.position = bee_i.position + \phi(bee_i.position - bee_j.position)$
            end
        end
        if $f(newbee) \le f(bee_i)$ then
            $bee_i = newbee$
        end
        else
            $trial_i = trial_i + 1$
        end
    end
    Phase 4: foreach $i = 1{:}\ n$ do
        if $trial_i >= limit$ then
            $Calculate\ bee_i.position\ using\ OBL$
            $Calculate\ f(bee_i),$
            $trial_i = 0$
        end
    end
    $Calculate\ RateofChange$
    if $(RateofChange)_t > (RateofChange)_(t-1)$ then
        Status = 1
    end
    else
        Status = 0
    end
end
```

Algorithm 2: MABC-SS.

Fig. 2 shows the logic within the proposed algorithm. The algorithm starts with population initialization phase. Then loops through the employed bees phase where strategies are selected based on the *status* value from the previous iteration. After going through each employed bee from the swarm, the onlooker bee phase starts. Again, the strategies dependent on *status* are used to calculate each bee position in the onlooker bee swarm, if there are scout bees around then new solution for each scout bee is calculated. Lastly, the *rate of change* is calculated and compared with the previous one to set the *status*. If the algorithm meets the termination criteria, then the output will be the best result. Otherwise, the algorithm will loop back to the employed bee phase and will continue from there.

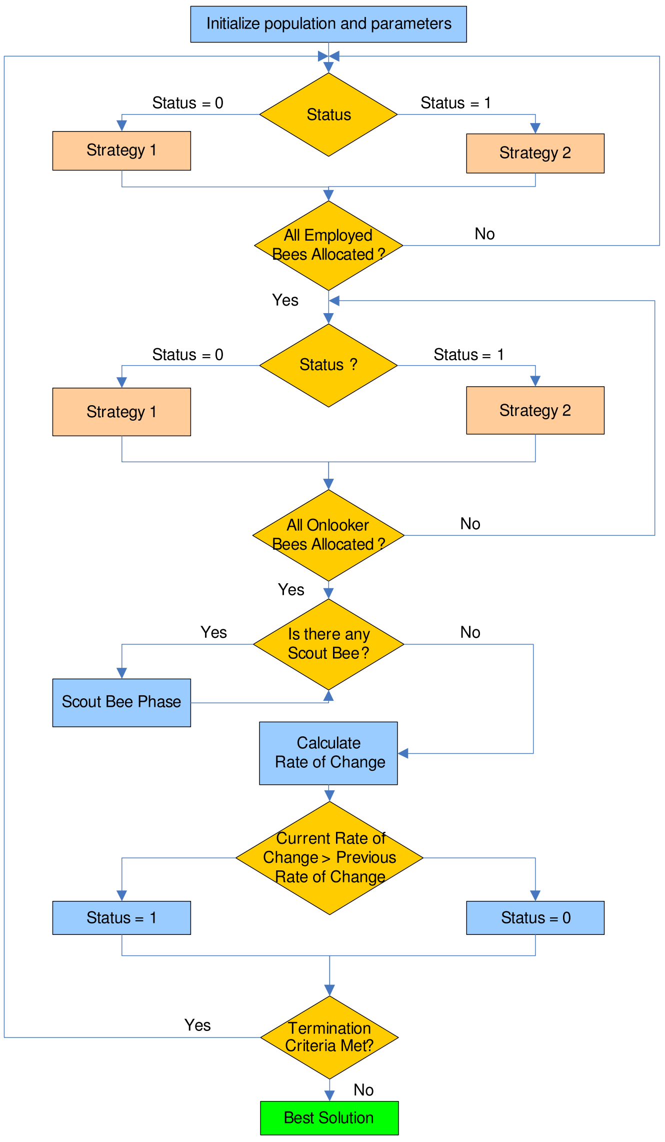


Fig. 2. Flowchart of the proposed algorithm.

5. Results

The proposed algorithm's performance is tested against 35 classical and 10 CEC2019 benchmark test functions. The benchmark functions are a mixture of unimodal and multimodal, and these test functions determine the algorithm's capability. The best, worst, median and mean costs and standard deviation for each test function are recorded.

5.1. Benchmark functions

The benchmark test functions are of different types such as separable (S), non-separable (NS), convex (CN), non-convex (NCN), continuous (C), discontinuous (DC), differentiable (D), non-differentiable (ND), unimodal (U) and multimodal (M). These are the problem categories that are preset in the actual world when a complex issue needs to be resolved.

The two test functions set evaluate different aspects of the algorithm. A unimodal test function tests the convergence and exploitation capability of an algorithm. Likewise, a multimodal test function tests the avoidance of an algorithm's local optima and exploration capability. A separable function is easy to solve because each variable can be optimized independently whereas a non-separable function is hard to solve because variables share strong relationship with each other. A continuous function is one in which a continuous variation (that is, a change without a leap) in the input causes a continuous variation in the function's value. If there is a leap, a function is said be discontinuous. A function is convex if and only if its epigraph—the collection of points on or above the function's graph—is a convex set. Convex functions are significant when researching optimization issues because of their variety of useful characteristics. For instance, there is only one minimum for an absolutely convex function on an open set. Whenever a function has a derivative, it is said to be a differentiable function of one real variable. Since many of these characteristics may be present in real problems from various fields, these various function characteristics are crucial when evaluating an algorithm's capabilities.

Tables 2 and 3 present the 35 benchmark functions used to test the proposed algorithm. Each function in the two tables are labelled from F1 to F35 and this labelling will be used to indicate the results achieved by differ algorithms for a particular function. The proposed algorithm has also been tested with CEC2019 benchmark test marks. Table 4 shows function details such as its range, dimension and optimal value. CEC04 to CEC10 are shifted and rotated functions while others are not. CEC01 to CEC03 have different dimensions and ranges, while CEC04 to CEC10 are 10-dimension problems with range [-100, 100].

5.2. *Experimented results*

Each function in Table 5 is executed 30 times with 100 as the size of the swarm and 10000 iterations [2]. The proposed algorithm achieved the optimal result in 30 out of 35 test functions. For F12 and F20, though the algorithm was unable to find the optimal value, the mean, best, median and worst costs are the same. In 22 out of 35 functions, the standard deviation is 0, which means that the results obtained are the same in every run.

Table 6 shows the simulation results of the MABC-SS algorithm on CEC2019 benchmark functions. The experiment had 30 bees, 500 iterations and 30 runs. The results show that the MABC-SS algorithm could not achieve the optimal result given in Table 4. However, the best result for CEC05, CEC03 and CEC09 was close to optimality. The results will be compared in the discussion section with variants and non-variants of ABC.

6. Discussion

Table 7 shows the best cost, mean cost and standard deviation of MABC-SS and ABC for 35 classical benchmark test functions. MABC-SS was able to achieve the optimal result in 30 out of 35 functions (86%). MABC-SS achieved the best results in the remaining 5 functions when compared to ABC. In comparison, ABC achieved the optimal value in 15 out of 35 functions (43%). Since MABC-SS is a modification of ABC, Table 7 shows that the modification has significantly improved the performance.

The results obtained in Table 8 are based on 30 bees and 500 iterations of CEC2019 test functions. The abandonment factor has been set to 0.6 for both algorithms. The results show that MABC-SS was able to obtain the best and mean value in all 10 problems, while ABC obtained the best result in 2 problems and mean results in 3 problems.

Table 9 presents the mean best values of other ABC variants for selected problems that are common in the literature [30]. The variants that will be compared with MABC-SS are original ABC, ABC with Knowledge infusion (KFABC) [30], Global guided ABC (GABC) [38], Modified ABC (MABC) and Bare bones ABC (BABC) [8]. Each ABC variant is executed for 30 runs with 50 initial solutions. The maximum function evaluations are set to 5000.D, where D is the dimension of the problem fixed to 30. The abandonment limit parameter, ρ, is set to 0.1. The parameters used to measure the mean best values of MABC-SS are kept the same as other ABC variants according to their relevant literature [30] [8] [31] [38]. The results show that MABC-SS achieved the best mean results in 7 out of 10 functions (70%), GABC achieved the best mean result in only 1 function, MABC achieved the best result in 3 functions, MEABC achieved the best mean result in 5 functions and BABC and KFABC achieved the best mean results in 4 test functions.

In Table 10, MABC-SS is compared with fitness dependent optimizer (FDO), dragonfly algorithm (DA), whale optimization algorithm (WOA) and salp swarm algorithm (SSP) in 10 benchmark real-world test functions. The results presented in Table 10 for all algorithms except MABC-SS and FDO are from the literature [1]. The experiment consisted of 500 iterations with 30 search agents and 30 runs. The same has been applied to MABC-SS algorithm for comparison purposes. The results show that MABC-SS achieved the best mean result in 5 out of 10 functions compared with FDO. The latter achieved the best mean result for the 4 test functions.

Table 11 compares the performance of MABC-SS with other algorithms such as butterfly optimization algorithm (BOA), Cuckoo search (CS), particle swarm optimization (PSO), genetic algorithm (GA), firefly algotihm (FA), Monarch butterfly optimization and differential evolution. The results presented in Table 11 except for MABC-SS are all from the literature [2]. The experiment has been set for 10000 iterations, 50 agents and 30 runs. A total of 18 test functions were common between the two research. The MABC-SS algorithm was able to achieve the optimal or best results in 17 out of 18 functions. In comparison, BOA achieved optimal or best result in 12 out of 18 functions, CS achieved best results in 3 out 18 functions, DE and PSO achieved best results in 7 out of 18 functions and GA obtained best results in 8 out of 18 functions. FA and MBO achieved best results in 4 and 5 test functions, respectively. The simulation results show that MABC-SS has generally given the best mean values for different test functions when compared to other algorithms.

6.1. *Statistical tests*

Table 12 shows the Wilcoxon rank sum test results for each classical benchmark test function. This test is conducted to show that the results in Table 6 are statistically significant. MABC-SS is a modification of ABC; therefore, the test is conducted between them. For Functions F4, F5, F6, F9 - F18, F22, F23, F28, F29,and F35, there is no difference between the result distribution of the two algorithms because the P-value is more than 0.05. Both algorithms achieved the same best and mean results in these functions. For the remaining functions (17 out of 35), the results between the two algorithms were statistically significant. Table 13 presents the result of Wilcoxon ranked sum test on CEC2019 benchmark test functions. The result is based on comparing MABC-SS with ABC and

Table 2
Classical benchmark test function (F1 – F16)(Type: C: Continuous, DC: Discontinuous, CN: Convex, NCN: Non-Convex, D: Differentiable, ND: Non-differentiable, M: Multimodal, U: Unimodal, S: Separable, NS: Non-separable).

F	Function Name	Type	Equation	Range	Dim	Opt. Value
F1	Ackley	C, NCN, S, D, M	$f(x) = -20\exp\left(-0.2\times\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\times\sum_{i=1}^{n} \cos(2\pi x_1)\right) + 20 + \exp$	[-32, 32]	30	0
F2	Alpine	DC, NCN, S, D, M	$f(x) = \sum_{i=1}^{n} \lvert x_i\sin(x_i) + 0.1x_i\rvert$	[-10, 10]	30	0
F3	beale	C, NCN, NS, D, M	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	[-4.5, 4.5]	2	0
F4	Bird	C, NCN, NS, D, M	$f(X) = (x_1 - x_2)^2 + \sin(x_1)\cdot e^{[1-\cos(x_2)]^2} + \cos(x_2)\cdot e^{[1-\sin(x_1)]^2}$	[−2 π, 2 π]	2	-106.7645
F5	Bohachevsky1	C, CN, S, D, U	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(3\pi x_2) + 0.7$	[-100,100]	2	0
F6	Bohachevsky2	C, NCN, NS, D, M	$f(X) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)[\cos(4\pi x_2)] + 0.3$	[-100,100]	2	0
F7	Bohachevsky3	C, NCN, NS, D, M	$f(X) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2)[\cos(4\pi x_2)] + 0.3$	[-100,100]	2	0
F8	Booth	C, CN, NS, D, U	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	[-10,10]	2	0
F9	Branin	C, NCN, NS, D, M	$f(\mathbf{x}) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s$	[-5, 10]	2	0.397887
F10	Carromtable	C, D, NS, M	$f_{\text{CarromTable}}(\mathbf{x}) = -\frac{1}{30}e^2\left\lvert 1 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert_{\cos^2(x_1)\cos^2(x_2)}$	[-10,10]	2	24.1568155
F11	crossintray	C, NCN, NS, ND, M	$f(\mathbf{x}) = -0.0001\left(\left\lvert\sin(x_1)\sin(x_2)\exp\left(\left\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)\right\rvert + 1\right)^{0.1}$	[-10,10]	2	-2.06261
F12	Dixonprice	C, CN, NS, D, U	$f(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=2}^{d} i(2x_i^2 - x_{i-1})^2$	[-10, 10]	30	0
F13	Easom	C, NCN, S, D, M	$f(x) = -\cos(x_1)\cos(x_2)\exp[-(x_1 - \pi)^2 - (x_2 - \pi)^2]$	[-100,100]	2	-1
F14	Giunta	C, D, S, M	$f(X) = 0.6 + \sum_{i=1}^{n}\left[\sin^2\left(1 - \frac{16}{15}x_i\right) - \frac{1}{50}\sin\left(4 - \frac{64}{15}x_i\right) - \sin\left(1 - \frac{16}{15}x_i\right)\right]$	[-1,1]	2	0.06447
F15	Goldstein Price	DC, NCN, NS, D, M	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	[-2,2]	2	3
F16	Griewangk	C, NCN, S, D, U	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600,600]	30	0

Table 3
Classical benchmark test function (F17 – F35)(Type: C: Continuous, DC: Discontinuous, CN: Convex, NCN: Non-Convex, D: Differentiable, ND: Non-differentiable, M: Multimodal, U: Unimodal, S: Separable, NS: Non-separable).

F	Function Name	Type	Equation	Range	Dim	Opt. Value
F17	Hartman3	C, D, NS, M	$f(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}\left(x_j - P_{ij}\right)^2\right)$	[0,1]	3	-3.86278
F18	Hartman6	C, D, NS, M	$f(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} A_{ij}\left(x_j - P_{ij}\right)^2\right)$	[0,1]	6	-3.32237
F19	Leon	C, NCN, NS, D, U	$f(x) = 100\left(x_2 - x_1^3\right)^2 + \left(1 - x_1\right)^2$	[-1.2,1.2]	2	0
F20	Levi	C, NCN, NS, D, M	$f(x) = \sin^2\left(3\pi x_1\right) + \left(x_1 - 1\right)^2 [1+$ $\sin^2\left(3x_2\right)] + \left(x_1 - 1\right)^2 \left[1 + \sin^2\left(2\pi x_2\right)\right] +$	[-10,10]	2	0
F21	Matyas	C, CN, NS, D, U	$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2$	[-10,10]	2	0
F22	mccormick	C, CN, NS, D, M	$f(\mathbf{x}) = \sin\left(x_1 + x_2\right) + \left(x_1 - x_2\right)^2 - 1.5x_1 + 2.5x_2 + 1$	X1 = [-1.5,4]X2 = [-3,4]	2	-1.98133
F23	Michalewiz	C, NCN, S, D, M	$f(x) = -\sum_{i=1}^{n} \sin\left(x_i\right)\left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m}$	[0, π]	10	-0.966015
F24	Rastrigin	C, NCN, S, D, M	$f(x) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$	[−5.12, 5.12]	30	0
F25	Rosenbrock	C, NCN, NS, D, M	$f(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$	[-10,10]	30	0
F26	schaffer2	C, NCN, NS, D, U	$f(\mathbf{x}) = 0.5 + \frac{\sin^2\left(x_1^2 - x_2^2\right) - 0.5}{\left[1 + 0.001\left(x_1^2 + x_2^2\right)\right]^2}$	[-100,100]	2	0
F27	Schwefel2.22	C, CN, S, ND, U	$f(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	[-10,10]	30	0
F28	schweffel	C, NCN, S, ND, M	$f(\mathbf{x}) = 418.9829d - \sum_{i=1}^{d} x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	[-500,500]	2	-837.9658
F29	shubert	C, NCN, NS, D, M	$f(x) = \left(\sum_{i=1}^{5} i\cos\left((i+1)x_1 + i\right)\right)$ $\left(\sum_{i=1}^{5} i\cos\left((i+1)x_2 + i\right)\right)$	[−10, 10]	2	-186.7309
F30	Sphere	C, CN, S, ND, U	$f(x) = \sum_{i=1}^{n} x_i^2$	[-100,100]	30	0
F31	Sum Squares	C, CN, S, D, U	$f(x) = \sum_{i=1}^{n} ix_i^2$	[-10, 10]	10	0
F32	three-hump Camel	C, NCN, NS, D, U	$f(\mathbf{x}) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$	[-5,5]	2	0
F33	Wood	C, D, NS, M	$f(X) = \left[100\left(x_2 - x_1^2\right)\right]^2 + \left(1 - x_1\right)^2 + 90\left(x_4 - x_3^2\right)^2 + \left(1 - x_3\right)^2 +$ $10.1\left[\left(x_2 - 1\right)^2 + \left(x_4 - 1\right)^2\right] + 19.8\left(x_2 - 1\right)\left(x_4 - 1\right)$	[-10,10]	4	0
F34	zakharov	C, CN, NS, ND, U	$f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^4$	[-5,10]	10	0
F35	Zettl	C, D, NS, U	$f(X) = \frac{1}{4}x_1 + \left(x_1^2 - 2x_1 + x_2^2\right)^2$	[-1,5]	2	−0.00379

Table 4
CEC2019 benchmark functions.

Function	Range	Dim.	Opt. Value
CEC01	[-8192, 8192]	9	1
CEC02	[-16834, 16834]	16	1
CEC03	[-4, 4]	18	1
CEC04	[-100, 100]	10	1
CEC05	[-100, 100]	10	1
CEC06	[-100, 100]	10	1
CEC07	[-100, 100]	10	1
CEC08	[-100, 100]	10	1
CEC09	[-100, 100]	10	1
CEC10	[-100, 100]	10	1

Table 5
Simulation results of the proposed algorithm.

F	Best	Median	Worst	Mean	SD
F1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F2	6.49E-15	9.88E-15	3.22E-14	1.21E-14	7.07E-15
F3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F4	-1.07E+02	-1.07E+02	-1.07E+02	-1.07E+02	2.21E-14
F5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F9	3.98E-01	3.98E-01	3.98E-01	3.98E-01	0.00E+00
F10	2.42E+01	2.42E+01	2.42E+01	2.42E+01	6.53E-15
F11	-2.06E+00	-2.06E+00	-2.06E+00	-2.06E+00	0.00E+00
F12	6.67E-01	6.67E-01	6.67E-01	6.67E-01	4.89E-10
F13	-1.00E+00	-1.00E+00	-1.00E+00	-1.00E+00	0.00E+00
F14	6.45E-02	6.45E-02	6.45E-02	6.45E-02	3.26E-17
F15	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.30E-16
F16	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F17	−3.86E+02	−3.86E+02	−3.86E+02	−3.86E+02	0.00E+00
F18	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00	3.46E-02
F19	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F20	1.35E-31	1.35E-31	1.35E-31	1.35E-31	0.00E+00
F21	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F22	-1.91E+00	-1.91E+00	-1.91E+00	-1.91E+00	3.44E-06
F23	-9.66E+00	-9.66E+00	-9.66E+00	-9.66E+00	0.00E+00
F24	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F25	3.97E-28	4.71E-11	3.99E+00	7.97E-01	1.68E+00
F26	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F27	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F28	-8.38E+02	-8.38E+02	-8.38E+02	-8.38E+02	0.00E+00
F29	-1.87E+02	-1.87E+02	-1.87E+02	-1.87E+02	3.73E-14
F30	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F31	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F32	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F33	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F34	3.31E-99	3.86E-97	2.91E-95	5.13E-96	8.83E-96
F35	-3.79E-03	-3.79E-03	-3.79E-03	-3.79E-03	1.02E-18

Table 6
Simulation results of the proposed algorithm tested on real world benchmark functions (CEC 2019).

Function	Best	Median	Worst	Mean	SD
CEC01	1.03E+08	5.75E+09	8.12E+10	1.10E+10	1.74E+10
CEC02	1.73E+01	1.73E+01	1.73E+01	1.73E+01	6.92E-15
CEC03	2.70E+00	1.27E+01	1.27E+01	1.27E+01	3.61E-15
CEC04	3.98E+00	2.35E+00	2.37E+00	2.15E+01	6.10E-03
CEC05	1.02E+00	1.09E+00	1.32E+00	1.10E+00	6.52E-02
CEC06	9.42E+00	1.09E+01	1.17E+01	1.08E+01	5.90E-01
CEC07	1.04E+02	3.62E+02	7.09E+02	3.79E+02	1.10E+02
CEC08	3.83E+00	5.29E+00	6.67E+00	5.01+00	1.40E+00
CEC09	2.34E+00	2.35E+00	2.37E+00	2.35E+00	6.10E-03
CEC10	2.05E+01	2.07E+01	2.09E+01	2.04E+01	1.15E-02

Table 7
Comparison of simulation results of MABC-SS and ABC on 35 classical benchmark functions.

F	MABC-SS			ABC		
	Best	Mean	SD	Best	Mean	SD
F1	**0.00E+00**	**0.00E+00**	0.00E+00	2.66E-15	2.66E-15	0.00E+00
F2	**6.49E-15**	**1.21E-14**	7.07E-15	2.40E-02	2.61E-02	9.93E-04
F3	**0.00E+00**	**0.00E+00**	0.00E+00	1.15E-13	1.96E-12	1.91E-12
F4	**-1.07E+02**	**-1.07E+02**	2.21E-14	**-1.07E+02**	**-1.07E+02**	2.21E-14
F5	**0.00E+00**	**0.00E+00**	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00
F6	**0.00E+00**	**0.00E+00**	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00
F7	**0.00E+00**	**0.00E+00**	0.00E+00	2.78E-16	2.11E-13	6.44E-15
F8	**0.00E+00**	**0.00E+00**	0.00E+00	6.42E-19	2.56E-17	2.17E-17
F9	**3.98E-01**	**3.98E-01**	0.00E+00	**3.98E-01**	**3.98E-01**	1.64E-11
F10	**2.42E+01**	**2.42E+01**	6.53E-15	**2.42E+01**	**2.42E+01**	3.35E-06
F11	**-2.06E+00**	**-2.06E+00**	0.00E+00	**-2.06E+00**	**-2.06E+00**	2.25E-10
F12	**6.67E-01**	**6.67E-01**	4.89E-10	**6.67E-01**	**6.67E-01**	3.24E-05
F13	**-1.00E+00**	**-1.00E+00**	0.00E+00	**-1.00E+00**	**-1.00E+00**	0.00E+00
F14	**6.45E-02**	**6.45E-02**	3.26E-17	**6.45E-02**	**6.45E-02**	3.26E-17
F15	**3.00E+00**	**3.00E+00**	3.30E-16	**3.00E+00**	**3.00E+00**	1.26E-15
F16	**0.00E+00**	**0.00E+00**	0.00E+00	**0.00E+00**	**0.00E+00**	0.00E+00
F17	**−3.86E+02**	**−3.86E+02**	0.00E+00	**−3.86E+02**	**−3.86E+02**	1.52E-15
F18	**-3.32E+00**	**-3.32E+00**	3.46E-02	**-3.32E+00**	**-3.32E+00**	1.80E-06
F19	**0.00E+00**	**0.00E+00**	0.00E+00	2.68E-07	1.85E-06	1.71E-06
F20	**1.35E-31**	**1.35E-31**	0.00E+00	4.29E-22	4.60E-20	5.47E-20
F21	**0.00E+00**	**0.00E+00**	0.00E+00	6.82E-16	3.01E-15	2.10E-15
F22	**-1.91E+00**	**-1.91E+00**	3.44E-06	**-1.91E+00**	-1.06E+02	9.29E-01
F23	**-9.66E+00**	**-9.66E+00**	0.00E+00	**-9.66E+00**	**-9.66E+00**	1.52E-03
F24	**0.00E+00**	**0.00E+00**	0.00E+00	1.60E+02	1.72E+02	8.87E+00
F25	**3.97E-28**	**7.97E-01**	1.68E+00	1.83E+01	1.85E+01	1.02E-01
F26	**0.00E+00**	**0.00E+00**	0.00E+00	2.42E-12	2.41E-10	1.19E-10
F27	**0.00E+00**	**0.00E+00**	0.00E+00	1.10E-36	2.43E-35	3.61E-35
F28	**-8.38E+02**	**-8.38E+02**	0.00E+00	**-8.38E+02**	**-8.38E+02**	0.00E+00
F29	**-1.87E+02**	**-1.87E+02**	3.73E-14	**-1.87E+02**	**-1.87E+02**	1.59E-04
F30	**0.00E+00**	**0.00E+00**	0.00E+00	2.88E-45	1.87E-43	3.01E-43
F31	**0.00E+00**	**0.00E+00**	0.00E+00	6.52E-46	4.02E-45	6.03E-45
F32	**0.00E+00**	**0.00E+00**	0.00E+00	7.38E-27	7.69E-26	5.22E-26
F33	**0.00E+00**	**0.00E+00**	0.00E+00	4.82E-08	8.80E-08	3.82E-08
F34	**3.31E-99**	**5.13E-96**	8.83E-96	2.71E-10	3.79E-10	1.08E-10
F35	**-3.79E-03**	**-3.79E-03**	1.02E-18	**-3.79E-03**	**-3.79E-03**	1.02E-18

Table 8
Comparison of simulation results of MABC-SS and ABC on 10 CEC2019 benchmark functions.

F	MABC-SS			ABC		
	Best	Mean	SD	Best	Mean	SD
CEC01	**1.03E+08**	**1.10E+10**	1.74E+10	2.99E+10	6.86E+10	2.92E+10
CEC02	**1.73E+01**	**1.73E+01**	6.92E-15	**1.73E+01**	**1.73E+01**	4.33E-12
CEC03	**2.70E+00**	**1.27E+01**	3.61E-15	1.27E+01	**1.27E+01**	9.90E-07
CEC04	**3.98E+00**	**2.15E+01**	6.10E-03	3.17E+01	4.65E+01	7.40E+00
CEC05	**1.02E+00**	**1.10E+00**	6.52E-02	1.29E+00	1.46E+00	7.74E-02
CEC06	**9.42E+00**	**1.08E+01**	5.90E-01	9.52E+00	1.11E+01	6.04E-01
CEC07	**1.04E+02**	**3.79E+02**	1.10E+02	5.95E+02	9.09E+02	1.77E+02
CEC08	**3.83E+00**	**5.01+00**	1.40E+00	6.08E+00	6.65E+00	2.41E-01
CEC09	**2.34E+00**	**2.35E+00**	6.10E-03	2.49E+00	2.72E+00	1.26E-01
CEC10	**2.03E+01**	**2.04E+01**	1.15E-02	**2.03E+01**	**2.04E+01**	1.11E-01

MABC-SS with FDO. The bold results in the table indicate that there is a significant difference (P-value is less than 0.05) between the two algorithms sample. The best and mean results for CEC02 and CEC10 were the same for MABC-SS and ABC. Hence there was no significant difference in the results of the two algorithms. There is a difference between the results achieved for CEC06 for MABC-SS and ABC but it is not significant. Overall, it can be stated that MABC-SS performed significantly better than ABC.

6.2. Quantitative measure

This section presents the results of the three measures that are used to conduct deep analysis with observation on the proposed algorithm.

The first measure is based on convergence speed. Achieving best or mean results are not the only goal of any optimization algorithm. Convergence speed is also essential. Figs. 4a - 4f show the convergence graphs for MABC-SS and ABC on six functions (F1,

Table 9
Mean results achieved by seven ABC variants.

F	Mean						
	MABC-SS	ABC	GABC	MABC	MEABC	BABC	KFABC
F1	**0.00E+00**	7.21E–05	2.32E–10	4.54E–11	3.26E–14	2.89E–14	1.84E–14
F2	3.41E-14	1.46E–06	4.55E–07	4.31E–13	**1.09E–21**	6.11E–16	1.67E–17
F16	**0.00E+00**	7.97E–14	2.22E–16	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F20	**1.35E–31**	2.16E–13	9.69E–16	4.73E–26	**1.35E–31**	**1.35E–31**	**1.35E–31**
F23	-2.81E+01	–2.93E+01	–2.95E+01	**–2.96E+01**	**–2.96E+01**	**–2.96E+01**	**–2.96E+01**
F24	**0.00E+00**	4.09E–14	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F25	1.40E+01	1.46E–01	**6.84E–02**	6.11E–01	3.01E–01	1.41E+00	1.69E–01
F27	**3.00E-60**	1.58E–10	1.61E–15	4.29E–14	1.62E–21	2.71E–30	4.01E–36
F30	**3.91E-101**	9.49E–16	4.45E–16	3.75E–26	3.12E–40	2.48E–56	6.28E–69
F31	**7.04E-103**	7.22E–16	5.97E–16	3.71E–41	5.26E–41	1.88E–55	2.39E–71

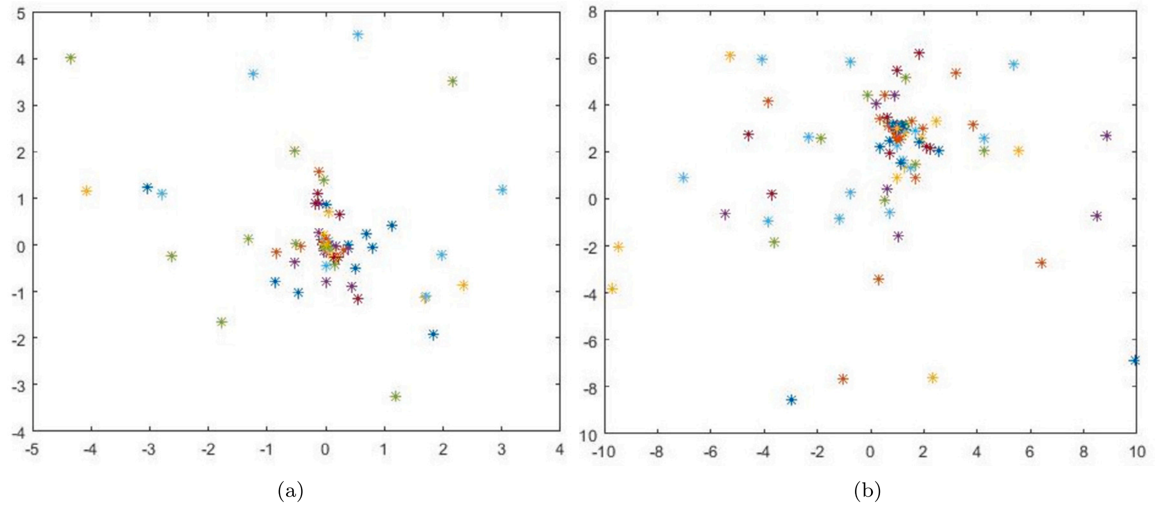


Fig. 3. Bee positions at each iteration in 2-D (a) Three hump camel (f32) and (b) Booth (f8).

F2, F23, F25, F27, F30). It can be seen that MABC-SS converges faster than ABC in F1, F2, F25 and F27, whereas ABC converges faster in f23 and f30 but the convergence speed for MABC-SS increases with increasing iterations. Even though ABC converges faster in F23 and F30, it is not able to obtain the global best solution.

The second quantitative measure is to observe the coverage of the bees in the search space. This experiment was conducted with 10 bees in 200 iterations. The dimension is set to 2 and the range remain the same as before. Fig. 3 shows the positions of the 10 bees for all 200 iterations for two problems F32 and F8. The figure shows that bees start with covering the boundary and then move towards optimality.

The third measure is the convergence of the global best solution. Fig. 5 shows the average and best cost values in each iteration for f34. The experiment consisted 30 bees with 500 iterations. The results show that the best bee continues to converge as the iteration increases. The clear difference between average cost and best cost values is because of the emphasis on local search and exploitation. Compared with Fig. 6, the entire swarm converges to the global best solution. This means that the algorithm improves the cost value of the fittest bee and the swarm as a whole.

The three measures discussed in this section show that the MABC-SS algorithm can explore the search space, fast converge toward the optimality compared to the standard ABC, and improve the fitness of the entire swarm. Even if the whole swarm is not improving, the fittest bee continues to improve.

7. Conclusion

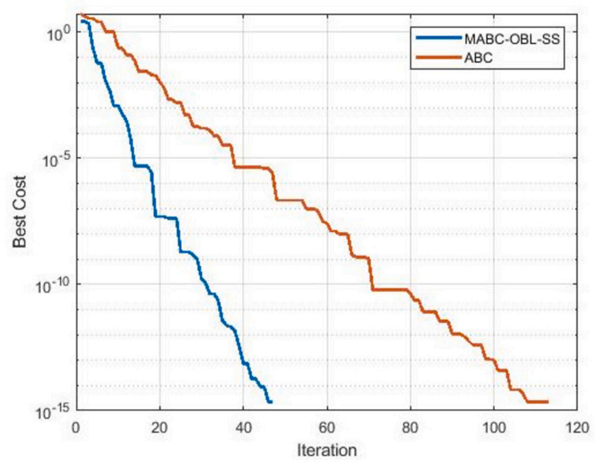
The main reasons for any algorithm modification are to improve its performance and solve different types of problems. This paper proposed an algorithm, MABC-SS, that is modified from the artificial bee colony optimization algorithm. The modification is in terms of population initialization, search equations, strategies and strategy selection technique. The algorithm initializes the population using random and opposition-based learning techniques. The search equations and strategies used in the algorithm focus on exploration and exploitation, that is, searching for solutions around a local or global solution. The two types of strategies focus on search techniques. In The first strategy, the employed bees search for a solution around a global solution. In contrast, the onlooker bees will search for a solution around a local solution. The second strategy provides four options for the algorithm which is dependent

Table 10
Comparison of the simulation results with other algorithms on CEC 2019 benchmark test functions.

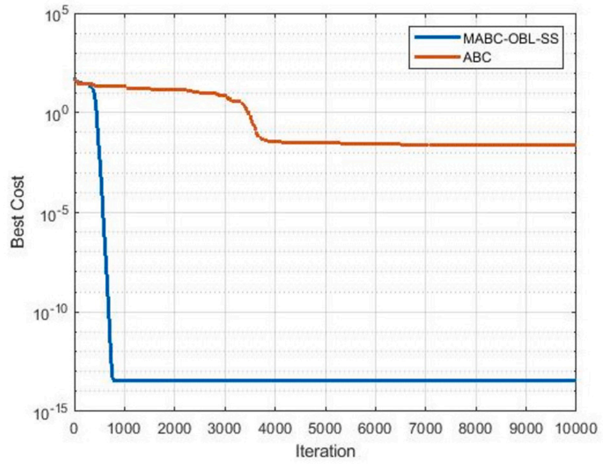
F	MABC-SS		FDO		DA		WOA		SSA	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
CEC01	1.10E+10	1.74E+10	**4.59E+03**	2.07E+04	5.43E+10	6.69E+10	4.11E+08	5.42E+10	6.05E+09	4.75E+09
CEC02	1.73E+01	6.92E-15	**4.00E+00**	1.07E-14	7.80E+01	8.78E+01	1.73E+01	4.50E-03	1.83E+01	5.00E-04
CEC03	**1.27E+01**	3.61E-15	1.37E+01	1.65E-11	1.37E+01	7.00E-04	1.37E+01	0.00E+00	1.37E+01	3.00E-04
CEC04	**2.15E+01**	6.10E-03	3.41E+01	1.65E+01	3.44E+02	4.14E+02	3.95E+02	2.49E+02	4.17E+01	2.22E+01
CEC05	**1.10E+00**	6.52E-02	2.14E+00	8.58E-02	2.56E+00	3.25E-01	2.73E+00	2.92E-01	2.21E+00	1.06E-01
CEC06	1.08E+01	5.90E-01	1.21E+01	6.00E-01	9.90E+00	1.64E+00	1.07E+01	1.03E+00	**6.08E+00**	1.49E+00
CEC07	3.79E+02	1.10E+02	**1.20E+02**	1.36E+01	5.79E+02	3.29E+02	4.91E+02	1.95E+02	4.10E+02	2.91E+02
CEC08	**5.01+00**	1.40E+00	5.64+00	1.14E+00	6.87E+00	5.02E-01	6.91E+00	4.27E-01	6.37E+00	5.86E-01
CEC09	**2.35E+00**	6.10E-03	2.40E+00	2.72E-02	6.05E+00	2.87E+00	5.94E+00	1.66E+00	3.67E+00	2.36E-01
CEC10	2.04E+01	1.15E-02	**1.87E+01**	5.00E+00	2.13E+01	1.72E-01	2.13E+01	1.11E-01	2.10E+01	7.80E-02

Table 11
Comparison of the simulation results of MABC-SS with other algorithms.

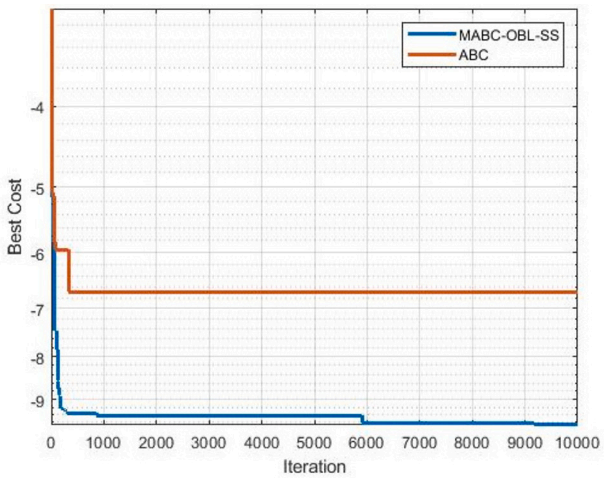
F	MABC-SS		BOA		CS		DE		FA		GA		MBO		PSO	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
F1	**0.00E+00**	0.00E+00	1.72E+00	0.00E+00	1.71E+01	5.46E+00	1.72+00	0.00E+00	7.73E–04	4.81E–05	4.69E–02	1.68E–02	1.31E+01	9.42E+00	1.82E+01	5.48E+00
F2	1.21E-14	7.07E-15	**0.00E+00**	0.00E+00	1.06E+01	4.23E+00	6.90E–16	2.56E–15	1.19E–02	6.43E–03	3.15E–17	3.08E–16	1.97E–01	3.77E–01	4.44E–02	4.53E–01
F3	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	7.69E–03	7.62E–02	**0.00E+00**	0.00E+00	1.07E–10	1.104E–10	**0.00E+00**	0.00E+00	1.87E–02	2.67E–02	**0.00E+00**	0.00E+00
F5	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	4.92E–04	8.67E–04	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	3.04E–02	2.62E–17
F8	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	1.82E–05	1.40E–05	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	8.89E–03	1.51E–02	**0.00E+00**	0.00E+00
F13	**-1.00E+00**	0.00E+00	**-1.00E+00**	0.00E+00	**-1.00E+00**	0.00E+00	**-1.00E+00**	0.00E+00	– 8.50E–01	3.59E–01	**-1.00E+00**	0.00E+00	**-1.00E+00**	0.00E+00	**-1.00E+00**	0.00E+00
F15	**3.00E+00**	3.30E-16	**3.00E+00**	0.00E+00	**3.00E+00**	0.00E+00	**3.00E+00**	0.00E+00	**3.00E+00**	0.00E+00	**3.00E+00**	0.00E+00	**3.00E+00**	2.96E–03	**3.00E+00**	0.00E+00
F16	**0.00E+00**	0.00E+00	1.85E-19	2.69E-20	2.00E+01	4.31E–02	9.22E–17	2.05E–17	1.07E–07	1.31E–08	5.56E–02	2.95E–02	4.31E+02	1.65E+02	4.05E+01	4.13E+01
F19	**0.00E+00**	0.00E+00	1.15E–06	9.47E–07	1.78E–04	1.55E–04	3.14E–03	7.19E–03	4.83E–05	6.48E–05	9.57E–06	3.02E–05	3.27E–01	1.41E–01	**0.00E+00**	0.00E+00
F20	**1.35E-31**	0.00E+00	4.42E–01	5.75E–02	1.62E+01	1.19E+01	5.98E–02	2.08E–01	1.20E+00	1.67E+00	**1.37E–04**	7.66E–05	3.14E+01	6.32E+01	4.93E+00	6.00E+00
F21	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	1.33E–06	2.16E–06	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	8.39E–104	2.52E–103	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F23	**-9.66E+00**	0.00E+00	– 5.34E+00	– 5.61E+00	– 8.06E+00	6.72E–01	– 9.62E+00	4.55E–02	– 8.88E+00	5.98E–01	**-9.66E+00**	0.00E+00	– 8.62E+00	3.82E–01	7.11E-01	1.45E+02
F24	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	1.08E+02	5.00E+01	1.25E+01	3.31E+00	3.47E+01	9.55E+00	1.65E+01	4.11E+00	6.38E+01	4.30E+01	1.45E+02	3.80E+01
F25	**7.97E-01**	1.68E+00	2.88E+01	3.13E–02	3.95E+01	3.34E+01	2.36E+01	4.99E+00	1.84E+01	4.24E+00	3.79E+01	2.56E+01	7.14E+00	1.43E+01	2.57E+02	3.73E+02
F27	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	1.12E+01	2.50E+01	8.15E–91	1.23E–90	1.51E–03	1.13E–04	5.44E–02	1.62E–02	7.02E–03	2.96E–03	7.28E+01	2.27E+01
F30	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	6.49E–61	9.77E–61	2.62E–165	0.00E+00	7.87E–04	1.37E–04	2.20E–02	9.68E–03	2.21E–01	1.41E–01	1.77E–51	1.12E–50
F31	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	2.10E+01	1.16E+01	1.46E–19	1.18E–19	1.46E–06	3.21E–07	3.43E–03	2.63E–03	7.69E+03	1.07E+03	8.27E+02	7.67E+05
F35	**-3.79E-03**	1.02E-18	**-3.79E-03**	0.00E+00	**-3.79E-03**	4.45E–07	**-3.79E-03**	4.57E–19	**-3.79E-03**	4.57E–19	**-3.79E-03**	4.57E–19	2.01E–01	2.23E–01	**-3.79E-03**	0.00E+00



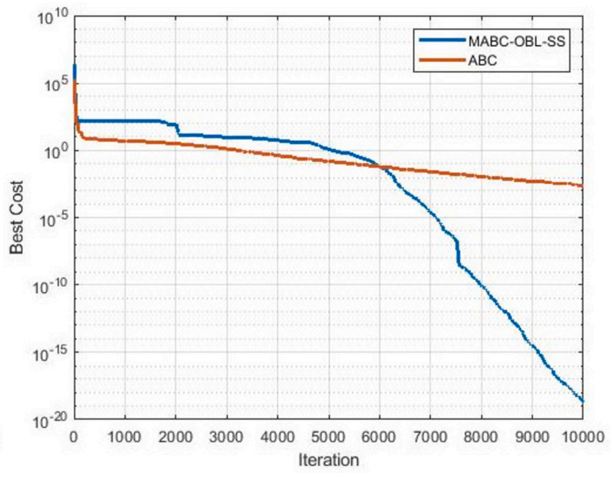
(a) Ackley (F1)



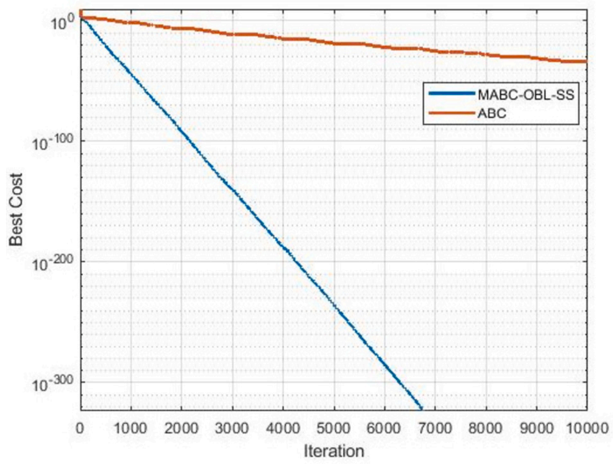
(b) Alpine (F2)



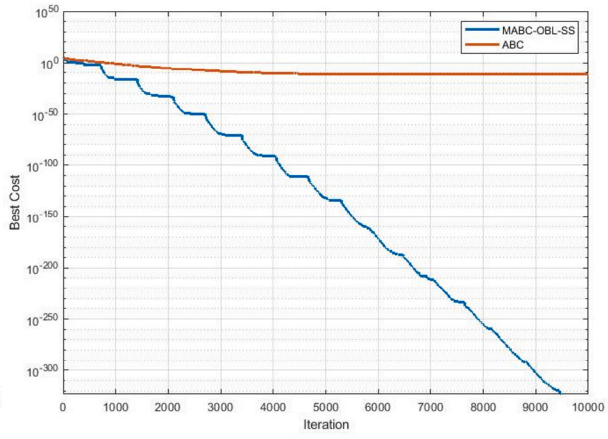
(c) Michalewiz (F23)



(d) Rosenbrock (F25)



(e) Schwefel2.22 (F27)



(f) Sphere (F30)

Fig. 4. Convergence curves of ABC and MABC-SS on six problems.

Table 12
The Wilcoxon rank sum test for classical benchmark test functions.

F	MABC-SS vs ABC (P-value)	F	MABC-SS vs ABC (P-value)
F1	**1.51E-11**	F19	**1.51E-11**
F2	**1.65E-04**	F20	**1.73E-10**
F3	**6.60E-13**	F21	**1.50E-11**
F4	0.00E+00	F22	0.00E+00
F5	0.00E+00	F23	0.00E+00
F6	0.00E+00	F24	**2.58E-03**
F7	**6.05E-13**	F25	**7.50E-03**
F8	**6.06E-13**	F26	**1.51E-11**
F9	0.00E+00	F27	**2.61E-03**
F10	0.00E+00	F28	0.00E+00
F11	0.00E+00	F29	0.00E+00
F12	0.00E+00	F30	**3.87E-09**
F13	0.00E+00	F31	**1.51E-11**
F14	0.00E+00	F32	**1.68E-08**
F15	0.00E+00	F33	**2.04E-09**
F16	0.00E+00	F34	**1.51E-11**
F17	0.00E+00	F35	4.90E-01
F18	0.00E+00		

Table 13
The Wilcoxon rank sum test for CEC2019 benchmark test functions.

F	MABC-SS vs ABC (P-value)
CEC01	**1.44E-09**
CEC02	5.00E-01
CEC03	**1.44E-09**
CEC04	**3.09E-04**
CEC05	**1.38E-18**
CEC06	1.46E-01
CEC07	**4.39E-09**
CEC08	**7.28E-08**
CEC09	**1.44E-09**
CEC10	0.00E+00

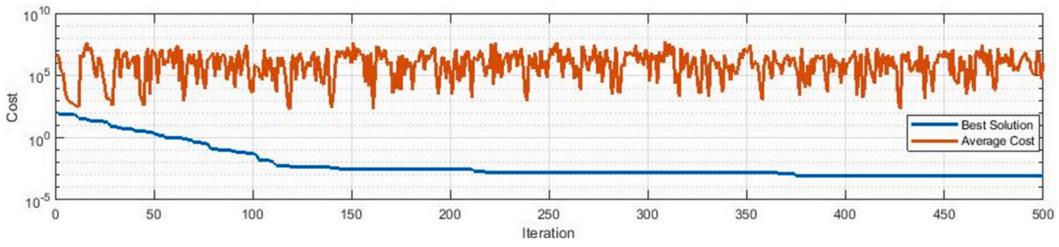


Fig. 5. Best and average solution cost values for F34.

on the result of the comparison between the fitness values of the best solution and a solution from the swarm. The strategy selection is based on a novel technique using the swarm's average fitness value to select the strategy to use in the next iteration.

Since the MABC-SS algorithm is modified from ABC, the paper's first comparison is with ABC on 35 classical benchmark test functions and 10 CEC2019 benchmark functions. The results indicate that MABC-SS achieved either the optimal result or the best result in all benchmark functions. The next comparison is made with other variants (5 variants) of ABC on selected test functions. Since the codes for the other variants were unavailable, the comparison is only made with the functions used in the literature to test those algorithms. The results show that MABC-SS achieved the optimal or best results in 7 out of 10 functions. Then the MABC-SS algorithm is compared with 4 other optimization algorithms on common functions between the two research. MABC-SS algorithm achieved the mean best result in 7 out of 10 functions. The Wilcoxon rank sum test was carried out to see whether the results were statistically significant. In most cases, the results were statistically significant between MABC-SS and ABC.

The quantitative measurement was also conducted to see the search space exploration, convergence speed, convergence of the entire population and the convergence of the global best solution. The proposed algorithm was able to explore the search space and later converges toward optimality. Likewise, MABC-SS converges faster than ABC with the best result. It was also observed that even if the solution quality of the whole swarm is not improving, the algorithm improves the global best solution.

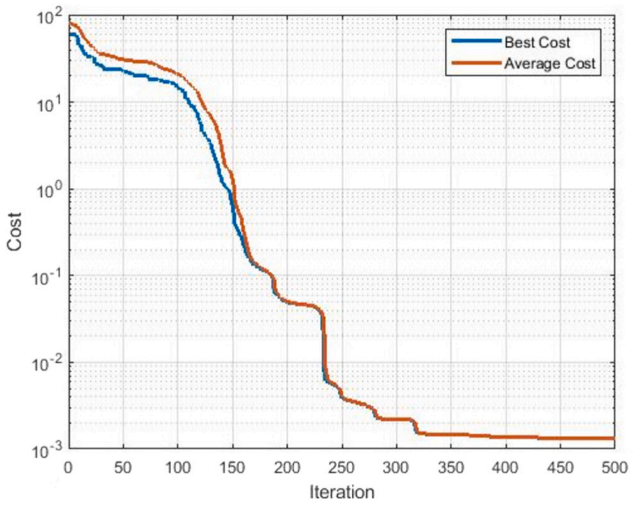


Fig. 6. Best and average solution cost values for F2.

The proposed algorithm has some limitations. While performance has improved, the number of objective function evaluation has increased. In future work, the author of this paper will investigate further increase in convergence and success rates through parameter tuning while having a low algorithm complexity.

CRediT authorship contribution statement

Kaylash Chaudhary: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] J.M. Abdullah, T. Ahmed, Fitness dependent optimizer: inspired by the bee swarming reproductive process, IEEE Access 7 (2019) 43473–43486.
[2] S. Arora, S. Singh, Butterfly optimization algorithm: a novel approach for global optimization, Soft Comput. 23 (3) (2019) 715–734.
[3] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, The best-so-far selection in artificial bee colony algorithm, Appl. Soft Comput. 11 (2) (2011) 2888–2901.
[4] Y. Cao, S. Ji, Y. Lu, Improved artificial bee colony algorithm with opposition-based learning, IET Image Process. 14 (15) (2020) 3639–3650.
[5] L. Cui, G. Li, Q. Lin, Z. Du, W. Gao, J. Chen, N. Lu, A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation, Inf. Sci. 367 (2016) 1012–1044.
[6] A. Das, H.S. Das, H.S. Das, Impact of cuckoo algorithm in speech processing, in: Applications of Cuckoo Search Algorithm and Its Variants, Springer, 2021, pp. 207–228.
[7] C.A. Floudas, C.E. Gounaris, A review of recent advances in global optimization, J. Glob. Optim. 45 (1) (2009) 3–38.
[8] W. Gao, F.T. Chan, L. Huang, S. Liu, Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood, Inf. Sci. 316 (2015) 180–200.
[9] W. Gao, S. Liu, Improved artificial bee colony algorithm for global optimization, Inf. Process. Lett. 111 (17) (2011) 871–882.
[10] W. Gao, Z. Wei, Y. Luo, J. Cao, Artificial bee colony algorithm based on parzen window method, Appl. Soft Comput. 74 (2019) 679–692.
[11] W.-f. Gao, L.-l. Huang, S.-y. Liu, F.T. Chan, C. Dai, X. Shan, Artificial bee colony algorithm with multiple search strategies, Appl. Math. Comput. 271 (2015) 269–287.
[12] W.-f. Gao, S.-y. Liu, A modified artificial bee colony algorithm, Comput. Oper. Res. 39 (3) (2012) 687–697.
[13] W.-f. Gao, S.-y. Liu, L.-l. Huang, A novel artificial bee colony algorithm based on modified search equation and orthogonal learning, IEEE Trans. Cybern. 43 (3) (2013) 1011–1024.
[14] W.-f. Gao, S.-y. Liu, L.-l. Huang, Enhancing artificial bee colony algorithm using more information-based search equations, Inf. Sci. 270 (2014) 112–133.
[15] Z.-a. He, C. Ma, X. Wang, L. Li, Y. Wang, Y. Zhao, H. Guo, A modified artificial bee colony algorithm based on search space division and disruptive selection strategy, in: Mathematical Problems in Engineering, 2014, 2014.
[16] K. Hussain, M.N. Mohd Salleh, S. Cheng, Y. Shi, Metaheuristic research: a comprehensive survey, Artif. Intell. Rev. 52 (4) (2019) 2191–2233.
[17] S.S. Jadon, R. Tiwari, H. Sharma, J.C. Bansal, Hybrid artificial bee colony algorithm with differential evolution, Appl. Soft Comput. 58 (2017) 11–24.
[18] F. Kang, J. Li, Z. Ma, Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions, Inf. Sci. 181 (16) (2011) 3508–3531.
[19] D. Karaboga, An idea based on honey bee swarm for numerical optimization, 2005.

[20] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm, J. Glob. Optim. 39 (3) (2007) 459–471.
[21] D. Karaboga, B. Gorkemli, A quick artificial bee colony (qabc) algorithm and its performance on optimization problems, Appl. Soft Comput. 23 (2014) 227–238.
[22] M.S. Kiran, H. Hakli, M. Gunduz, H. Uguz, Artificial bee colony algorithm with variable search strategy for continuous optimization, Inf. Sci. 300 (2015) 140–157.
[23] S. Kumar, A. Nayyar, R. Kumari, Arrhenius artificial bee colony algorithm, in: S. Bhattacharyya, A.E. Hassanien, D. Gupta, A. Khanna, I. Pan (Eds.), International Conference on Innovative Computing and Communications, Springer, Singapore, 2019, pp. 187–195, Singapore.
[24] A. Nayyar, V. Puri, G. Suseendran, Artificial bee colony optimization—population-based meta-heuristic swarm intelligence technique, in: V.E. Balas, N. Sharma, A. Chakrabarti (Eds.), Data Management, Analytics and Innovation, Springer, Singapore, 2019, pp. 513–525, Singapore.
[25] H. Peng, C. Deng, Z. Wu, Best neighbor-guided artificial bee colony algorithm for continuous optimization problems, Soft Comput. 23 (18) (2019) 8723–8740.
[26] R.V. Rao, V. Patel, Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm, Appl. Math. Model. 37 (3) (2013) 1147–1162.
[27] X. Shi, Y. Li, H. Li, R. Guan, L. Wang, Y. Liang, An Integrated Algorithm Based on Artificial Bee Colony and Particle Swarm Optimization, 2010 Sixth International Conference on Natural Computation, vol. 5, IEEE, 2010, pp. 2586–2590.
[28] H.R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), volume 1, IEEE, 2005, pp. 695–701.
[29] M.N.A. Wahab, S. Nefti-Meziani, A. Atyabi, A comprehensive review of swarm optimization algorithms, PLoS ONE 10 (5) (2015) 1–36.
[30] H. Wang, W. Wang, X. Zhou, J. Zhao, Y. Wang, S. Xiao, M. Xu, Artificial bee colony algorithm based on knowledge fusion, Complex Intell. Syst. 7 (3) (2021) 1139–1152.
[31] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, J.-s. Pan, Multi-strategy ensemble artificial bee colony algorithm, Inf. Sci. 279 (2014) 587–603.
[32] Y. Xiang, Y. Peng, Y. Zhong, Z. Chen, X. Lu, X. Zhong, A particle swarm inspired multi-elitist artificial bee colony algorithm for real-parameter optimization, Comput. Optim. Appl. 57 (2) (2014) 493–516.
[33] S. Xiao, W. Wang, H. Wang, X. Zhou, A new artificial bee colony based on multiple search strategies and dimension selection, IEEE Access 7 (2019) 133982–133995.
[34] Y. Xue, J. Jiang, B. Zhao, T. Ma, A self-adaptive artificial bee colony algorithm based on global best for global optimization, Soft Comput. 22 (9) (2018) 2935–2952.
[35] Z.-H. Zhan, L. Shi, K.C. Tan, J. Zhang, A survey on evolutionary computation for complex continuous optimization, Artif. Intell. Rev. 55 (1) (2022) 59–110.
[36] H. Zhao, Z. Pei, J. Jiang, R. Guan, C. Wang, X. Shi, A hybrid swarm intelligent method based on genetic algorithm and artificial bee colony, in: International Conference in Swarm Intelligence, Springer, 2010, pp. 558–565.
[37] J. Zhao, L. Lv, H. Sun, Artificial bee colony using opposition-based learning, in: Genetic and Evolutionary Computing, Springer, 2015, pp. 3–10.
[38] G. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Appl. Math. Comput. 217 (7) (2010) 3166–3173.