Artificial Intelligence Review (2021) 54:4967–4996
https://doi.org/10.1007/s10462-021-10015-1

Bee-inspired metaheuristics for global optimization: a performance comparison

Ryan Solgi[1] · **Hugo A. Loáiciga**[2]

Published online: 20 May 2021

Abstract
Metaheuristics are widely applied to solve optimization problems. Numerous metaheuristic algorithms inspired by natural processes have been introduced in the past years. Studying and comparing the convergence of metaheuristics is helpful in future algorithmic development and applications. This study focuses on bee-inspired metaheuristics and identifies seven basic or root algorithms applied to solve continuous optimization problems. They are the bee system, mating bee optimization (MBO), bee colony optimization, bee evolution for genetic algorithms (BEGA), bee algorithm, artificial bee colony (ABC), and bee swarm optimization. The algorithms' performances are evaluated with several benchmark problems. This study's results rank the cited algorithms according to their convergence efficiency. The strengths and shortcomings of each algorithm are discussed. The ABC, BEGA, and MBO are the most efficient algorithms. This study's results show the convergence rate among different algorithms varies, and evaluates the causes of such variation.

Keywords Metaheuristics · Swarm intelligence · Evolutionary algorithms · Optimization · Bee inspired algorithms

1 Introduction

The task of optimization arises in a wide range of problems relevant to machine learning (Darwish et al. 2019; Xu et al. 2016), robotics and animation (Starke et al. 2019; Dereli and Koker et al. 2019), supply chains and logistics (Rabe and Deininger 2012), transportation (Moradipari and Alizadeh 2018), urban infrastructures (Solgi et al. 2016; Bozorg-Haddad et al. 2016a) and other fields. Optimization is simply defined as finding the best feasible solution based on some criteria and constraints among a set of plausible alternatives. Yet, optimization problems cannot always be easily solved. A variety of techniques have been introduced to solve different kinds of optimization problems

✉ Ryan Solgi
Solgi@ucsb.edu

[1] University of California Santa Barbara (UCSB), Santa Barbara, CA, USA

[2] Department of Geography, University of California Santa Barbara (UCSB), Santa Barbara, CA, USA

including linear programming, dynamic programming, non-linear convex optimization and so forth (Hillier and Lieberman 1995; Boyd and Vandenberghe 2004).

Metaheuristics are other kinds of optimization algorithms that are useful to find a near optimal solution when the problem is not approachable by other techniques (Bozorg-Haddad et al. 2017a, b). Metaheuristic are defined as "*a system of adaptive heuristics which are adjusted through concurrent learning*". Many metaheuristic algorithms have been applied successfully in a broad range of domains, from computer science to healthcare and across all engineering fields (Abualigah and Hanandeh 2015; Bozorg-Haddad et al. 2016b; 2017a; b; Abualigah et al. 2017, 2018a, b; Ashghari and Navimipour, 2019a, b; Hajimirzaei and Navimipour 2019, Panahi and Navimipour 2019; Zanbouri and Navimipour 2019, Abualigah 2019, Abualigah and Diabat 2020).

Sorensen et al. (2017) reported that formal studies on techniques used in modern metaheuristic algorithms can be traced back to world war II with the advent of operations research long before the term metaheuristic emerged. Hooke and Jeeves (1961) introduced pattern search (PS), which is one of the early and well-known metaheuristics. The emergence of metaheuristics has gone hand in hand with the revolutionary emergence of evolutionary computation (EC). De Jong et al. (1997) presented a recent history of EC starting from the mid 1950s, citing the endeavors of pioneers like Box (1957), Friedberg (1958), Barker (1958), and Bremermann (1962), who implement evolutionary processes in enhancing industrial productivity, mathematical programming, and machine learning. After a relatively short period of skepticism the power of EC was demonstrated by the notable works of Rechenberg (1965), Fogel et al. (1966), and Holland (1967) who invented evolutionary strategies (ESs), evolutionary programming (EP), and genetic algorithms (GAs), respectively. A variety of metaheuristics were introduced inspired by different metaphors following the introduction of the well-known GA by Holland (1975). Simulated annealing (SA) by Kirkpatrick et al. (1983), tabu search by Glover (1986), ant colony optimization (ACO) by Dorigo et al. (1991, 1996), particle swarm optimization (PSO) by Kennedy and Eberhart (1995), and differential evolution (DE) by Storn and Price (1997) are among the early metaheuristics. Since then the metaheuristics and EC gained popularity and many other algorithms have been developed (Abualigah 2020; Abualigah et al. 2020a, b).

One of the metaphors inspiring metaheuristics is bees' behavior. Bees' social and mating rituals are unique and constitutes a kind of swarm intelligence that has survived for nearly 100 million years. Normally these species live in well-organized colonies which maximize efficiency by division of labor for tasks like foraging, and for reproduction. Numerous algorithms have been recently inspired from bee swarming behavior (Karaboga and Akay 2009). The first metaheuristic explicitly claimed to be inspired by bees was the bee system (BS) developed by Sato and Hagiwara (1997). Next, a number of bee inspired metaheuristics were proposed including but not limited to mating bee optimization (MBO) by Abbass (2001), bee colony optimization (BCO; Lucic 2002), bee evolution for genetic algorithms (BEGA; Jung 2003), BeeHive (Wedde et al. 2004), bee algorithm (BA; Pham et al. 2005), artificial bee colony (ABC; Karaboga 2005), honey bee social foraging (Quijano and Passino 2010), bee swarm genetic algorithm (BSGA; Xu et al. 2008), bumblebees (Comellas and Martinez-Navarro 2009), virtual bee algorithm (VBA; Khan et al. 2010), bee swarm optimization (BSO; Akbari et al. 2010), and queen honey bee migration (QHBM; Jong et al. 2017). The aforementioned list cites a few algorithms which have undergone modifications and hybridizations after development, and which have been implemented in solving a variety of problems that cannot be recounted in a length-limited paper.

Our study of bee inspired algorithms revealed that among an extensive number of bee-inspired metaheuristics, many of which have been tied to a specific kind of problem, only a few basic ideas have been proposed for global optimization in the continuous domain. Yet, there is no comparison of these algorithmic ideas. This work presents an experimental study of the performance of bee-inspired metaheuristic algorithms. Hence, the bee system (BS), mating bee optimization (MBO), bee colony optimization (BCO), bee evolution for genetic algorithms (BEGA), bee algorithm (BA), artificial bee colony (ABC), and bee swarm optimization (BSO) are selected and their performance has been compared under several benchmark function. This work contributes to algorithmic development by highlighting pathways for future studies in the field of metaheuristics.

In the following, first the methodology of this study is presented. Next, the bees' characteristics that have been used for algorithmic development are briefly described. This is followed by a description of the metaheuristic algorithms studied in this work and the steps of each algorithm. The performances of the algorithms are evaluated with continuous-domain problems and the results are presented in the section "experiments and results". A discussion on the algorithms and pathways for future developments closes this paper.

2 Methodology

This work presents an experimental study of the performance of bee-inspired metaheuristic algorithms for global optimization. A literature review was conducted to find the bee-inspired metaheuristics that solve continuous problems, and which can be rated as original or basic (root) ideas. Our review went beyond reliance titles or keywords and considered the conceptual contexts of published works. Moreover, the characteristics and steps of the algorithms in numerous articles were meticulously evaluated. Our review compiled all bee-inspired metaheuristics. Hundreds of journals, book chapters, and conference proceedings were found in which either a new algorithm under a new title was developed or former algorithms were modified or applied. Our search was narrowed down to only bee-inspired metaheuristics which were capable of solving continuous optimization problems.

Once an algorithm is introduced it may undergo modifications seeking to improve its efficiency. Sometimes these modifications amount to changing or adding a function or an operator, but they rarely constitute a completely new search strategy. Therefore, this work reviewed modified algorithms looking for novel metaheuristics even if the designers themselves did not present their idea under a new name. Some of the bee inspired algorithms available in the literature were initially presented for combinatorial problems, and many of them were developed to solve specific problems. The search operators in such algorithms are only appropriate for the specific kind of problems which they were designed for and not for continuous functions. However, there are some algorithms which initially were developed for combinatorial problems, such as the BCO, and were later modified to solve continuous problems. Another example is the MBO that initially was designed for the traveling salesman problem (TSP) but later was applied to continuous functions. This work selected bee-inspired metaheuristics that have been used for solving continuous problems.

The list of selected algorithms for analysis consists of the bee system (BS), mating bee optimization (MBO), bee colony optimization (BCO), bee evolution for genetic algorithms (BEGA), bee algorithm (BA), artificial bee colony (ABC), and bee swarm optimization (BSO). There are modified versions for some of these algorithms, yet this work evaluated their first versions because many of the modifications, hybridizations, and adaptations rely

heavily on the basic characteristics or ideas. Also, the selected algorithms seem to be constructively different from each other, whereas their modifications commonly replace one or two heuristic functions, or adjust probabilities involved in the search process, and introduce other minor changes. In addition, some of the modified versions are devoted to solving specific kind of problems, or their modifications are not inspired by bees thus making modification not pertinent to this study. It is worthy of notice that numerous algorithmic modifications seem to be more efficient algorithms because they reach a better solution or find an optimum with fewer function evaluations. However, many of the modifications have been made at the expense of adding extra user-defined parameters that increased the difficulty of application. Therefore, this work focused on the eight cited, basic but distinct, and original bee-inspired metaheuristics.

There were some challenges encountered while completing this review that may affect the completeness of the selected list of metaheuristic algorithms. For example, some of the names of the algorithms were used interchangeably, and some algorithms were given more than one name by different authors. It is possible that algorithms that qualify as basic and novel were not evaluated in our study. This work, therefore, does not claim to be an exhaustive survey of all the bees-inspired metaheuristics. Nevertheless, this study presents a representative review and evaluation of bee-inspired, metaheuristic algorithms.

The codes of the evaluated algorithms were herein written in Python version 3.7.0 based on the published papers. All the codes are available online at [https://github.com/rmsolgi/bee-inspired-metaheuristics] for those who may wish to reproduce or validate our findings.

Each algorithm herein studied has a number of parameters that must be adjusted usually by conducting a sensitivity analysis. Parameter setting is a key disadvantage of metaheuristics algorithms that hinder their application. In many previous studies algorithmic comparisons resorted to meticulous parameter adjustments. Such careful parameter adjustments may render the comparative results biased because commonly it is not reported how much time and effort is spent in parameter setting. This is pertinent in the realm of metaheuristics because the algorithms are expected to be clever and self-adjusting and thus capable of removing the burden of parameter calibration to users of metaheuristics. For this reason, one of the primary goals of this study is determining the power of basic ideas in metaheuristics in a truly unguided search. This means this work's algorithmic analysis did not involve parameter setting. Rather, algorithmic parameters were set equal to the values reported by their original developers. This is consistent with this work's intent to evaluate basic search ideas in terms of their inherent capabilities, such as ease of implementation and search efficiency. Therefore, for each algorithm a single set of parameters is used in solving all the benchmark problems.

This work begins with an overview of bee-inspired metaheuristics, followed by the presentation of results from the implementation of the algorithms, and it ends with a discussion of the algorithms' performances, highlighting pathways for future related studies in the field of metaheuristics.

3 Bee inspired metaheuristics

Metaheuristics consists mainly of two repetitive phases: generation of new tentative solutions, and selection of a set of solutions to advance the search towards a global optimum. Every algorithm initially generates a set of possible or tentative solutions randomly or deterministically. The generated solutions are evaluated based on some predefined criteria

and some of them are selected to form the basis for generating newly improved solutions. Bee metaheuristics commonly generate initial solutions randomly. Next, selection and generation of new solutions are repeated iteratively until some termination criteria are met. The algorithms generally converge to a near optimum eventually. But the convergence rate and accuracy of the algorithms differ, which has led to the development of a variety of algorithms seeking the highest search efficiency. Exploration and exploitation are two important factors in population-based metaheuristics, which must be balanced well otherwise the algorithm would experience either premature convergence (trapping in a local optimum) or stagnation. This balance is implicitly provided by probabilistic functions applied for selecting and generating new solutions. Primarily the major differences between algorithms are the functions or procedures implemented in selecting and generating new solutions. Proportionate selection and simulated annealing are among the selection techniques. Random walk and genetic operators such as crossover and mutation are among the functions used for generating new solutions. A mixture of these techniques and other novel heuristic functions lead to a variety of algorithms.

This study identified eight metaheuristics with basic characteristics that clearly differentiate one from each other as distinct, bee-inspired, metaheuristics. The identified algorithms are: the bee system (BS), mating bee optimization (MBO), bee colony optimization (BCO), bee evolution for genetic algorithms (BEGA), bee algorithm (BA), artificial bee colony (ABC), and bee swarm optimization (BSO). It is noteworthy that there are two distinct sources of inspiration for these algorithms, namely, bee foraging behavior and bee mating rituals.

Most of the bee inspired metaheuristics were developed based on the foraging behavior of bees including the ABC, BCO, BSO, and BA. There exist in nature three types of bee foragers responsible for collecting tasks and feeding a bee colony. These are employed (experienced), onlookers (unemployed), and scouts each of which has a unique way of searching for new food sources. Employed foragers are those who already discovered a good food source and return to that frequently. Due to randomness they may visit nearby neighborhood points and occasionally discover a better source. The employed bees may share their information to unemployed bees through a waggle dance and attract some unemployed bees to follow them. These followers are called onlookers who watch the dance of some experienced foragers and follow them. The number of unemployed bees attracted by each experienced forager is proportional to the quality of their dance. Onlookers following an experienced forager search neighboring location of the already discovered source. The third groups of foragers are scouts which look for food resources totally unguided and search randomly all across the space. An experienced forager may be initially an onlooker or a scout, and after finding a good food source it may stick to it and advertise it through waggle dancing.

A solution to the optimization problem achieved by bee-foraging metaheuristics is represented by an N dimensional vector (where N denotes the number of control variables) defining the position of an agent (bee forager). Each point in the decision space has a degree of desirability with respect to a criterion. That degree of desirability is measured by the fitness function.[1] Agents are looking for a location which has the most food (the global optimum). Starting from initial random locations the agents update their positions through search procedures and gradually move toward the best position.

[1] Fitness function refers to a penalized objective function, that is, the objective function with constraints added to it as penalty.

Experienced foragers, onlookers, and scouts are all agents that update their positions with different heuristic functions. The selection phase in these algorithms generally refers to tasks about dividing populations of bees (i.e., solutions) into the three categories of foragers, selecting experienced foragers by onlookers, and deciding whether or not agent must move to a new position.

Bees' mating and reproduction processes have also inspired other metaheuristics such as MBO and BEGA. From mating ritual and reproduction perspectives a colony of bees is composed of a queen, drones, and workers. The queen and drones are responsible for reproduction so that the queen is the mother of the hive and the drones are the fathers. Workers are primarily responsible for brood caring. The broods made by a queen bee can be both haploid and diploid. Diploid broods stem from fertilized eggs produced by the mating of the queen with drones. However, the queen can also lay eggs without any mating giving birth to diploid broods which inherit their genotype only from the queen. During the mating season a queen mates with several drones to fill its spermatheca through a mating dance. Among many drones vying for queen only a few succeed in mating and passing their sperms to its spermatheca. The more attractive the drone, the better the chance it has to mate with the queen. The drones which mate with the queen usually die immediately after mating and fall to the ground. The queen goes back to the hive after its spermatheca is full or when it has lost her mating strength. The rest of the drone population is forbidden from entering the hive at the end of mating season and die from exposure and famine. Only the productive queen and the workers remain in the hive until the queen makes new broods.

In the metaheuristics inspired by mating rituals each solution is represented as the genotype of a bee. The hive's queen is the best solution that can be found; drones are other trial or random solutions. Workers in these algorithms are related to specific heuristic functions. They do not concern the solution, but, rather, become functions for generating a new solution. This resembles the role of worker bees which feed and grow the broods. In these algorithms new solutions (offspring) are usually generated by genetic operators such as crossover and mutation. Selection refers to the process by which the queen (female bee) selects some drones (male bees) for reproduction.

In the following a summary of the development and later modifications of the studied algorithms is provided. This is followed by a brief description of the algorithmic steps. For brevity, if a function or step is the same between two algorithms, it is referred to the first time it is presented whenever it is used again. This summary is helpful to highlight the differences, similarities, and the performances of the reviewed algorithms. For a complete description of the underlying techniques the reader is referred to the cited works and to the Python codes provided online.

3.1 Bee system (BS)

The Bee system was introduced by Sato and Hagiwara (1997). The BS is inspired by the foraging behavior of bees. Yet, considering the other algorithms inspired by bees' foraging behavior it is debatable to state that this algorithm actually resembles the foraging behavior of bees. This work reviews this algorithm because BS constitutes one of the first efforts to design a metaheuristic based on bees' behavior. The algorithm consists of a global search by means of a simple GA and a local search which is designed based on the sharing information by a swarm of bees.

Steps of the BS:

1. A global search is executed by running a simple GA G_{sc} times, and the best solution of each run is memorized. To produce a population (set) of possible solution vectors M ($M = G_{pop}$) N-dimensional vectors are generated randomly as follows:

$$X_i = (x_{i,1}, \ldots, x_{i,N}) \ \forall i = 1,2,\ldots,M \tag{1}$$

$$x_{i,d} = r\left(l_d, u_d\right) \ \forall d = 1,2,\ldots,N \tag{2}$$

where $r\left(l_d, u_d\right)$ = a random number in $[l_d, u_d]$, l_d and u_d = lower boundary an upper boundary, respectively, of the d-dimensional solution space.

Each vector or member X_i of the population is known as a chromosome and assigned a probability using Eq. (3):

$$Prob_i = \frac{f_i}{\sum_{i=1}^{m} f_i} \tag{3}$$

where f_i = normalized fitness function and $m = M$.

A roulette wheel is implemented based on the calculated probabilities. P vectors ($P < M$) are selected[2] as the parent population using the roulette wheel, and the rest of the vectors are removed from the population. M-P new vectors are generated to reconstitute the population. Hence, first a mating pool (the active parent population) is constructed so that every vector in the parent population has a chance equal to the crossover probability, P_c ($0 < P_c < 1$, a predefined parameter) of entering the mating pool. Next, two vectors are repetitively selected from the mating pool with the uniform distribution, and two new vectors are generated applying the crossover and mutation functions successively.

Let two parent vectors or solutions $X = (x_1, x_2, \ldots, x_N)$ and $X' = (x_1', x_2', \ldots, x_N')$ be selected for crossover.[3] Two new offspring are generated as follows:

$$X_2^{new} = (x_1', x_2', \ldots, x_C', x_{C+1}, x_{C+2}, \ldots, x_N) \tag{4}$$

$$X_1^{new} = \left(x_1, x_2, \ldots, x_C, x_{C+1}', x_{C+2}', \ldots, x_N'\right). \tag{5}$$

where C denotes the crossover point.

By mutation each component of a solution has a chance equal to P_m ($0 < P_m < 1$ a predefined parameter) to be replaced by a random value within the feasible space.

The M-P newly generated vectors as offspring and the parent population constitute the new population. The steps of selecting P parents followed by reproduction continues until a termination criterion is met. The best achieved solution is saved after the GA is terminated in a set called superior chromosomes (SC). This memorized set of solutions each of which is a vector in N dimensional space is applied for a local search which starts in step 2 below. A new GA is implemented until G_{sc} superior chromosomes are achieved and the SC set is full.

[2] In this manuscript selections always are done with replacement.

[3] This work applies uniform crossover in all algorithms in which the crossover function is used. The number and place of crossover points are random and uniformly distributed (Bozorg-Haddad et al. 2017a, b).

2. For each member of set SC a separate (local) population is constructed. Each local population consists of M ($M = L_{pop}$) vectors in an N dimensional space which are initially generated randomly using Eq. (1).
3. For each local population:

3.1. First the population is modified by a concentrated crossover function between each member of the population and the superior chromosome, and the new generated solution replaces the previous one. This is called concentrated crossover because one side of the crossover is always the superior chromosome.

3.2. The three best vectors of each local population are saved as $X_1^{'}$,$X_2^{'}$, and $X_3^{'}$. Next, the procedure of the (standard) GA is applied to each local population to execute parent selection, by roulette wheel, and reproduction. Reproduction generates M-P-2 new vectors as offspring by genetic operators. Next, a conditional statement called migration criterion is applied. If the migration criterion is met the algorithm goes to step 3.3. Otherwise, it goes to step 3.4. The migration criterion is simply a predefined number such as G_{mig} whereby the migration procedure is activated after G_{mig} iterations of the GA.

3.3. Two vectors are exchanged between two neighboring local populations so that the local nth population gives a vector to the $n+1$st population and receives one vector in turn. This step resembles the exchange of information among bees.

3.4. Two new vectors are generated based on the simplex method with the three already selected vectors $X_1^{'}$,$X_2^{'}$, and $X_3^{'}$ as follows:

$$X_{ref} = (1+\alpha)X_0^{'} + \alpha X_3^{'} \tag{6}$$

$$X_{cont} = (1-\beta)X_0^{'} + X_3^{'} \tag{7}$$

$$X_0^{'} = \frac{X_1^{'} + X_2^{'}}{2} \tag{8}$$

X_{ref} and X_{cont} = two new solutions added to the population, α and β = values in the range [0,1], which are parameters of the algorithm.

4. The algorithm stops once the termination criterion is met. Otherwise, it goes to step 3.2 (Sato and Hagiwara 1997).

The user-defined parameters of the BS are G_{sc}, G_{pop}, L_{pop}, P_{mg} (P_m for global search), P_{ml} (P_m for local search), G_{mig}, P_c (probability of crossover), P_{pop} (parent population), α, β, and the total NFE (TNFE, termination criterion). Sato and Hagiwara (1997) did not cite some of these parameters (e.g., P_c) and underestimated the number of the BS's parameters. This algorithm applies the GA, which features its own parameters, thus adding to the number of required parameters to be set.

3.2 Mating bee optimization (MBO)

Abbass (2001) proposed mating bee optimization (MBO) inspired by the mating ritual of honey bees. The algorithm implemented simulated annealing function for selection and genetic operators for the reproduction phases. Abbass and Teo (2003) replaced the original annealing function used in the selection phase of the MBO with a more conventional simulated annealing method. The MBO has been modified as honey bee mating optimization (HBMO, Bozorg-Haddad et al. 2006). Yang et al. (2007) introduced

the fast marriage in honey bee optimization (FMBO) in which the annealing process was removed and all randomly generated solutions were accepted in the selection process without any filtering. Poolsamran and Thammano (2011) proposed new heuristic functions to generate new solutions for the MBO. Celik and Ulker (2013) applied the levy flight algorithm in the mating step (selection) of the MBO and called it improved marriage in honey bees optimization (IMBO). Solgi et al. (2017) reported a modified HBMO in which the simulated annealing function is replaced by proportionate selection and a new heuristic function was introduced.

Steps of the MBO:

1. M vectors (solutions) in an N dimensional space are generated randomly (Eq. (1)) and form a population of possible solution vectors.
2. The best vector in the population is saved as Q. The Q refers to a queen bee.
3. The population is discarded. It sets $Energy = 1$; $Counter = 0$; and $\lambda = 1$.
4. One random vector is generated. The new solution is saved in the memory called queen's spermatheca probabilistically based on the following annealing function:

$$Prob = e^{-\Delta f/\lambda} \tag{9}$$

   where Δf = absolute difference between the fitness of new random vector and the best vector in the current population. After the annealing function is applied it sets $Energy = Energy - \omega$ and $\lambda = \mu \times \lambda$; if the new vector is successful it is added to the queen's spermatheca, $Counter = Counter + 1$. $0 < \mu \leq 1$ and $0 \leq \omega \leq 1$ are predefined parameters of the algorithm and respectively are called the queen speed parameter and energy decay rate.
5. If $Energy > 0$ and $Counter < SC$ the algorithm goes to step 4. Otherwise, it goes to step 6. Step 4 and 5 resemble the mating flight of a queen bee during which the queen mates with several drones successively until its spermatheca fills with drones' sperm or its energy is completely depleted. SC denotes the size of the queen's spermatheca, a predefined parameter of the algorithm. During the mating flight the speed of the queen is gradually reduced by parameter μ; which increase the selection pressure (makes the queen to be pickier).
6. In this step M new vectors are generated as broods. If $Counter > 0$ (i.e., the spermatheca is not empty) broods are generated with the crossover (Eq. (4)) and mutation functions, successively. Notice that in the MBO, one side of the crossover is always Q and the other side is randomly selected from the vectors saved in the queen's spermatheca with the uniform distribution. Mutation adds more diversity to the generated vectors by crossover. However, if $Counter = 0$ (this signals the queen's energy is used up and no drone is selected during the mating flight) new vectors are generated by mutation applied to the vector Q.
7. The best newly generated vector replaces Q if is better than Q.
8. The algorithm goes to step 3 if the termination criterion is not met; otherwise, it ends (Abbass 2001).

The parameters of the MBO which must be set are M (population size), P_m (probability of mutation), μ, ω, SC, and the number of algorithmic iterations (denoted by $TNIt$).

3.3 Bee evolution for genetic algorithms (BEGA)

Jung (2003) introduced the queen bee evolution for genetic algorithm (BEGA). Ming et al (2010) applied a self-adaptive selection operator resulting in the improved bee evolutionary genetic algorithm (IBEGA). The BEGA can be seen as a modification on the GA by modifying the selection and reproduction steps.

Steps of the BEGA:

1. M vectors in an N dimensional space are generated randomly (Eq. (1)) and form a population of tentative solution vectors.
2. P vectors must be selected and form a parent population in the GA. The BEGA selects half of the parents using a roulette wheel described by Eq. (3). The other half of the parent population is filled by copying the best vectors of the current population.
3. $M - P$ new vectors are generated by crossover and mutation successively as is done in the standard GA (see step 1 of the BS). The BEGA has two types of mutation functions: normal and strong mutations. A portion θ of the population, which is a predefined parameter, undergoes normal mutation and the rest is subjected to strong mutation. The difference is that in the strong mutation functions the probability of changing the value of an element in a vector is higher than in normal mutation.
4. If the termination criterion is not met the algorithm goes to step 2; otherwise, it stops (Jung 2003).

The parameters of the BEGA are M (the population size), θ, P_{mn} (normal mutation probability), P_{ms} (strong mutation probability), P_c (crossover probability), P_{pop} (the number of parents), and the total number of iterations ($TNIt$).

3.4 Artificial bee colony (ABC)

The Ant Bee Colony (ABC) algorithm was developed by Karaboga (2005). Karaboga et al. (2012) presented a literature review on the implementation of the ABC algorithm in a wide range of problems. Later on, researchers continued to apply the ABC algorithm for solving numerous problems (Mernik et al. 2015). Tsai et al. (2009) applied the Newtonian law of gravitation to update the position of onlookers in the ABC. Zhu and Kwong (2010) enhanced the ABC algorithm by implementing the information achieved from the best solution in the population of solutions to generate new solutions, and called it gbest guided ABC (GABC). Banharnsakun et al. (2011) updated the way new solutions are generated by onlooker bees that steers the search towards the best solution more rapidly. Wang and Wang (2012) reasoned that the GABC may increase the probability of trapping at a local optimum and proposed a pbest guided ABC (PABC). Gao et al. (2012) implemented chaotic system and opposition-based learning method for generating the initial population in a global best guided ABC algorithm. Xiang and An (2013) modified initialization, selection, and generating new solutions in the ABC. Gao et al. (2013a, b) proposed new functions and orthogonal learning for generating new neighborhood solutions for the ABC. Gao et al. (2015) proposed a multi-population strategy for the ABC. Qin et al. (2015) applied a time-varying strategy to balance the ratio between the number of onlookers and employed bees. Cui et al. (2018) applied a dual population framework in which greedy search by

employed bees was done near a better portion of the population (and not on the whole of population as is done in the original ABC). Aslan (2019) modified the selection process of the ABC applying a transition control mechanism in which not all employed bees advertise the food source they found; only a portion of employed bees attempt to attract onlookers. Aslan et al. (2019) proposed improved artificial bee colony (iqABC) in which a new exploitation technique has been applied. Chen et al. (2019) applied differential search strategies to update more variables using combination of crossover and mutation operators and introduced self-adaptive differential ABC (sdABC). Gupta and Deep (2019) used sin cosine algorithm (SCA) to improve the performance of the ABC.

Steps of the ABC:

1. M vectors in an N dimensional space are generated randomly (Eq. (1)) and form a population of possible solution vectors; Define n_e which is the number of employed (experienced) bees, as a parameter of the algorithm.
2. For every vector i, another vector j is randomly selected ($i \neq j$ and $1 \leq j \leq n_e$) using the uniform distribution. Next, a neighborhood point is generated by changing the magnitude of vector i in a randomly selected direction d:

$$x_{i,d}^{new} = x_{i,d} + (r - 0.5) \times \left| x_{i,d} - x_{j,d} \right| \tag{10}$$

   in which $x_{i,d}^{new}$ = new value of the d^{th} element of the i^{th} vector and r = a random value in [0,1]. The new vector replaces the old vector if the former has a better fitness than the latter. This process is analogous to the movement of actual employed bees to a new position where there is more food.
3. A roulette wheel is constructed using probabilities calculated by the Eq. (3) for the employed bee's population already modified in step 2. $M - n_e$ vectors from the population of employed bees are selected with the constructed roulette wheel. For each selected vector i another employed bee j is selected using the uniform distribution, and a new neighbor point is generated with Eq. (10).

   If the new vector's fitness function is improved it replaces the old vector. This step resembles actual onlooker bees following the employed bees and exploring the neighboring locations of food sources already found by employed bees. The actual experienced agents which discover better food sources attract more onlookers; therefore, this step implements a roulette wheel selection function. The numbers of onlookers and experienced foragers (n_e and n_o, respectively) are usually selected to be the same and are equal to one half of the population size ($M/2$). Each experienced agent examines only one neighborhood point in its position, whereas the frequency of search near the positions of onlookers is proportional to the desirability of their positions.
4. S vectors of employed bees which have not been improved by neighborhood search for at least a predetermined number of iterations (*limit*) are replaced by a vector randomly generated. This step resembles scouts in a bee swarm (unlike onlookers, scout bees move randomly and do not follow experienced foragers).
5. The algorithm goes to step 2 if the termination criteria are not met. Otherwise, the algorithm stops (Karaboga and Basturk 2008).

The parameters of the ABC are M (the population size), *limit*, n_e, S (the number of scouts), and the total number of iterations (*TNIt*).

3.5 Bee algorithm (BA)

The bee algorithm (BA) was introduced by Pham et al. (2005). Koc (2010) studied the effect of several modifications of the BA named dynamic recruitment, proportional shrinking, and abandonment strategies. Hussein et al. (2016) studied several variants of the BA including basic, standard, and shrinking-based BAs. Pham et al. (2011) presented a modified version of the BA that implemented several operators for generating new solutions and also a slight modification on the criteria for initializing scout bees. Yuce et al. (2013) proposed an adaptive neighborhood size change and site abandonment (ANSSA) for the BA. Pham and Darwish (2008) proposed a fuzzy greedy selection to adjust the parameters of the algorithm automatically. Nasrinpour et al. (2017) also developed a grouped version of the BA (GBA) for reducing the number of parameters that must be adjusted.

Steps of the BA:

1. M vectors in an N dimensional space are generated randomly (Eq. (1)) and form a population of possible solution vectors. The parameters n_{elit}, n_b,n_{en}, n_{bn} are set so that $n_{bn} < n_{en}$ and $n_e + n_b < M$.
2. The population of solutions is sorted in descending order according to the solutions' fitness functions' desirability.
3. For each of the first n_{elit} vectors n_{en} new vectors are generated as follows:

$$x_{i,d}^{j} = \mathcal{R}(\tau_d, x_{i,d}) \ \forall \ \ 1 \le i \le n \ \ 1 \le d \le N \ 1 \le j \le \ n_n \tag{11}$$

$$\tau_d = r \times (\varepsilon/2) \tag{12}$$

   where $\mathcal{R}(\tau_d, x_{i,d})$ = symmetric random walk starting from $x_{i,d}$ with step τ_d; and r is a random value uniformly distributed in [0,1]; ε is a predefined parameter of the algorithm that refers to the maximum distance a bee can fly away from its original position; $n_n = n_{en}$ (number of trail vectors); and$n = n_{elit}$. If the best vector among the n_n newly trial vectors is better than the current one the best one replaces the current one.
4. For each of the next n_b vectors in the population n_{bn} new vectors are generated with Eq. (11) where $n_n = n_{bn}$ and $n \le i \le n + n_b$. The best trial vector replaces the current one if its fitness function is better than the current one's. Thus, the only difference between step 3 and 4 is the number of neighborhood points n_{bn} and n_{en} that are evaluated.
5. The rest of the population $(M - n_b)$ is replaced by random vectors as scouts.
6. If the termination criterion is not met the algorithm goes to step 2; otherwise, it stops (Pham et al. 2005).

The parameters of the BA are M, n_{elit}, n_b,n_{en}, n_{bn}, ε, and the total number of iterations (*TNIt*).

3.6 Bee swarm optimization (BSO)

Bee swarm optimization (BSO) developed by Akbari et al. (2010) is an optimization algorithm inspired by the foraging strategies of honey bees.

Steps of the BSO:

1. M vectors in an N dimensional space are generated randomly (with Eq. (1)) and form a population of possible solution vectors. The population of solutions is sorted in descending order based on the solutions' fitness functions' desirability.
3. n_f new vectors are generated as follows:

$$x_{i,d}^{new} = x_{i,d} + \omega_b r_b\left(b_{i,d} - x_{i,d}\right) + \omega_e r_e(e_d - x_{i,d}) \quad \forall\ 1 \le d \le N \quad 1 \le i \le n_f \tag{13}$$

   where ω_b and ω_e = a predetermined parameter in [0,1]; r_b and r_e random values in [0,1]; $b_{i,d} = d^{th}$ component of the best ever experienced position by vector i (the best experienced positions and corresponding fitness functions are updated each iteration and saved); and e_d = the best vector in the current population. These new vectors refer to the new positions of actual foragers in nature.
4. A roulette wheel using probabilities calculated by Eq. (3) is applied to the n_f newly generated vectors, where $m = n_f$.
5. n_l new vectors are generated as follows:

$$x_{i,d}^{new} = x_{i,d} + \omega_e r_e(g_{i,d} - x_{i,d}) \quad \forall\ 1 \le d \le N \quad n_f < i \le n_f + n_l \tag{14}$$

   where $g_{i,d}$ = the d^{th} component of a vector selected from n_f previously generated vectors in step 3. The selection is done by replacement and using the already constructed roulette wheel function in step 4. This step resembles actual onlooker bees who follow an experienced forager. Obviously, vectors with a relatively superior fitness function have more chance to be selected by roulette wheel selection.
6. $M - n_f - n_l$ new vectors are generated as follows:

$$x_{i,d}^{new} = x_{i,d} + \mathcal{R}(\tau, x_{i,d}) \quad \forall\ 1 \le d \le N \quad n_f + n_l < i \le M \tag{15}$$

$$\tau = \rho_t(u_d - l_d) \quad \forall\ 1 \le d \le N \tag{16}$$

   in which $\mathcal{R}(\tau, x_{i,d})$ = a symmetric random walk from $x_{i,d}$ with step τ. $\rho_t = \rho_{max}$ initially and is reduced to ρ_{min} linearly through iterations where ρ_{max} and ρ_{min} are in [0,1].
7. The new population is sorted in descending order based on the solutions' fitness functions' desirability.
8. The fitness of the newly generated vectors are compared with their previous values. The new vectors replace the old ones if their fitness functions are better than those of the old vectors.
9. The algorithm stops if the termination criterion is met; otherwise, it proceeds to step 3, (Akbari et al. 2010).

The parameters of the BSO are M, ω_b, ω_e, n_f, n_l, ρ_{max}, ρ_{min}, and the total number of iterations (*TNIt*).

3.7 Bee colony optimization (BCO)

The bee colony optimization (BCO) was introduced by Lucic (2002) to solve transportation problems. The algorithm was initially named bee system but was later renamed BCO. This algorithm is different from the BS developed by Sato and Hagiwara (1997). The original BCO was designed to solve combinatorial problems. Nikolic and

Teodorovic (2013) presented a version of BCO for solving continuous problems; yet, the continuous version did not become ubiquitous in the manner the combinatorial version did.

Steps of the BCO:

1. M vectors in an N dimensional space are generated randomly (Eq. (1)) and form a population of possible solution vectors.
2. Based on probabilities calculated by Eq. (3) a roulette wheel is applied and one vector from the population is selected. The whole population is updated as follows:

$$X_i = V \ \forall \ 1 \le i \le M \tag{17}$$

where V = selected vector. This is step is executed because this algorithm was primarily designed for combinatorial problems so that all artificial bees (agents) start moving and gradually build a path to a solution. The algorithm is designed in a way that at the start of each iteration the population consists of equal solutions. The algorithm also sets $\sigma = (u_d - l_d)$ initially and $Counter = 0$.

3. For every vector i in the population a single dimension d from N is randomly selected using the uniform distribution and the d^{th} component of the vector i is modified as follows:

$$x_{i,d}^{new} = x_{i,d} + \mathcal{R}(\tau, x_{i,d}) \ \forall \ 1 \le i \le M \tag{18}$$

$$\tau = r \times Min(\sigma, \Delta) \tag{19}$$

where $\mathcal{R}(\tau, x_{i,d})$ = symmetric random walk starting from $x_{i,d}$ with step τ; r = a random number in [0,1]; and Δ = the distance from the current value of the dth component of vector i from either the lower or upper boundary based on the direction of the random walk. It sets $Counter = Counter + 1$.

4. The algorithm updates $\sigma = \eta \times \sigma$ where η is a positive value less than one. If σ is less than a predefined value γ, it sets $\sigma = (u_d - l_d)$ again.
5. While $Counter < n_p$: If $Counter$ modulo n_c is not zero the algorithm goes to step 3; otherwise, the algorithm continues to step 5.1. If $Counter = n_p$ then the algorithm proceeds to step 6.

5.1. For every vector i in the population the algorithm decides if the vector must keep its current value and search in the neighborhood of the current position (a recruiter bee) or must abandon the current position (becoming an uncommitted bee) and follows a recruiter (experienced forager). For this, probabilities are calculated as follows:

$$Prob_i = e^{-(f_e - f_i)} \ \forall \ 1 \le i \le M \tag{20}$$

where f_e = fitness function of the best vector in the population, and f_i = fitness function of vector i. A random number in [0,1] is generated for each vector and a vector remains recruiter if $Prob_i$ is larger than the random number; otherwise, it abandons the current position and becomes an uncommitted agent.

5.2. Once recruiters and uncommitted vectors are determined a recruiter is selected for each uncommitted vector (onlooker) by applying a roulette wheel function based on Eq. (3) in which m = number of recruiters. The algorithm goes to step 3.

6. If the termination criterion is met the algorithm terminates; otherwise, it goes to step 2 (Nikolic and Teodorovic, 2013).

Table 1 List of the test functions

Function's name	Function's equation	Domain
Ackley	$f_1(X) = -20EXP\left(-0.2\sqrt{\frac{\sum_{d=1}^{N} x_d^2}{N}}\right) - EXP\left(\sqrt{\frac{\sum_{d=1}^{N} Cos(2\pi x_d)}{N}}\right) + 20 + EXP(1)$	$-32.768 < x_d < 32.768$
Griewank	$f_2(X) = 1 + \frac{1}{4000}\sum_{d=1}^{N} x_d^2 - \prod_{d=1}^{N} Cos(\frac{x_d}{\sqrt{d}})$	$-600 < x_d < 600$
Michalewicz	$f_3(X) = -\sum_{d=1}^{N} Sin(x_d)\left(Sin\left(\frac{dx_d^2}{\pi}\right)\right)^{20}$	$0 < x_d < \pi$
Restrigin	$f_4(X) = \sum_{d=1}^{N}(x_d^2 - 10Cos(2\pi x_d) + 10)$	$-5.12 < x_d < 5.12$
Rosenbrock	$f_5(X) = \sum_{d=1}^{N-1} 100(x_{d+1} - x_d^2)^2 + (x_d - 1)^2$	$-50 < x_d < 50$
Schaffer	$f_6(X) = 0.5 + \frac{(Sin(\sum_{d=1}^{N} x_d^2))^2 - 0.5}{(1+0.001(\sum_{d=1}^{N} x_d^2))^2}$	$-100 < x_d < 100$
Sphere	$f_7(X) = \sum_{d=1}^{N} x_d^2$	$-100 < x_d < 100$
Schwefel	$f_8(X) = 418.9829N - \sum_{d=1}^{N} x_d Sin(\sqrt{\lvert x_d \rvert})$	$-500 < x_d < 500$
Weierstrass	$f_9(X) = \sum_{d=1}^{N}\left[\sum_{k=0}^{20} 0.5^k Cos(2\pi 3^k(x_d + 0.5)) - N\sum_{k=0}^{20} 0.5^k Cos(\pi 3^k)\right]$	$-0.5 < x_d < 0.5$

The parameters of the BCO are M, η, γ, n_p, n_c, and the total number of iterations (*TNIt*).

4 Experiments and Results

The studied algorithms were implemented for solving nine benchmark functions listed in Table 1 (Molga and Smutnicki, 2005), where $X = (x_1, x_2, \ldots, x_d, \ldots x_N)$ and N denotes the decision variable and the dimension of the decision space, respectively. The minima of all of these functions equal zero, except for the optimal solution of the Michalewicz function which changes depending on its dimensionality. The 5-dimensional Michalewicz' function has optimal value equal to (-4.687658). The parameters of the algorithms are listed in Table 2. It is noteworthy that algorithms' parameters were mostly set equal to the values recommended by the developers of the algorithms. The number of iterations in all the algorithms corresponds to about 500,000 functional evaluations executed in solving the same problems for the purpose of comparison. The number of functional evaluations (NFE) is widely used for algorithmic comparison because estimating the fitness function for one trial solution may be as computationally intense as tens of iterations of a metaheuristic algorithm. Many previous studies multiply the number of iterations by the average number of generated solutions in each iteration. This work counted exactly the number of times the fitness function is executed. The fitness function is called when generating a new solution, and if the fitness is going to be used later its value must be saved, thus avoiding calling it again. These precautions might seem obvious, but they are in fact necessary to ensure the underestimation or overestimation of functional evaluations (Mernick et al. 2015).

The algorithms first were used to solve the problems in 5-dimensional space. Each algorithm was run five times and the average and standard division of each algorithm are listed in Tables 3 and 4, respectively. Figure 1 shows the average of the runs of the algorithms for problems with 5-dimensional space. Figure 2 shows the number of times each problem

Table 2 Algorithmic parameters

BS		BSO		BA		BEGA		MBO		BCO		ABC	
G_{sc}	3	M	100	M	500	M	100	M	100	M	50	M	100
G_{pop}	100	ω_b	0.5	n_{elit}	5	θ	0.8	P_m	0.3	η	0.998	$limit$	$n_e \times N$
L_{pop}	100	ω_e	0.5	n_b	15	P_{mn}	0.05	μ	0.99	γ	0.001	n_e	50
P_{mg}	0.05	n_f	48	n_{en}	50	P_{ms}	1	ω	0.01	n_p	1	S	1
P_{ml}	0.5	n_l	48	n_{bn}	30	P_c	0.6	SC	25	n_c	50	TNIt	5,000
G_{mig}	5	ρ_{max}	1	ε	0.01	P_{pop}	0.3	TNIt	2,500	TNIt	10,000		
P_c	0.5	ρ_{min}	0.1	TNIt	500	TNIt	10,000						
P_{pop}	0.3	TNIt	10,000										
α, β	0.4												
TNFE	500,000												

Table 3 Average of five runs of the algorithms for the test functions with 5-dimensional decision space (absolute zeros refer to any number less than 1.0E-17)

Function	ABC	BA	BCO	BS	BSO	MBO	BEGA
Ackley	− 4.4E-16	7.5E+00	6.4E+00	2.1E-01	7.9E-01	6.3E-04	3.2E-04
Griewank	0.0E+00	1.7E+00	3.6E-01	2.9E-01	2.9E-01	1.8E-02	2.1E-02
Michalewicz	− 4.7E+00	− 4.7E+00	− 4.4E+00	− 4.7E+00	− 4.3E+00	− 4.7E+00	− 4.7E+00
Rastrigin	0.0E+00	2.6E+00	2.2E+00	1.9E+00	2.2E+00	1.4E-04	1.4E-04
Rosenbrock	3.6E-02	8.1E+02	1.3E+01	1.9E+01	8.9E+00	1.9E+00	1.6E+01
Schaffer	3.9E-03	1.5E-01	3.3E-01	9.8E-02	9.7E-03	2.1E-02	2.6E-02
Sphere	0.0E+00	1.1E+02	2.6E-02	1.2E+00	8.3E-03	4.0E-04	5.4E-04
Schwefel	6.4E-05	2.5E+02	2.5E+01	1.5E-01	4.9E+02	5.1E-04	9.6E-04
Weierstrass	0.0E+00	9.0E-01	8.2E-01	2.1E-01	3.6E-01	3.9E-02	4.6E-02

Table 4 Standard deviation of five runs of the algorithms for test functions with 5-dimensional decision space (absolute zeros refer to any number less than 1.0E-17)

Function	ABC	BA	BCO	BS	BSO	MBO	BEGA
Ackley	0.0E+00	2.4E+00	7.3E+00	1.4E-01	4.6E-01	5.3E-04	1.6E-04
Griewank	0.0E+00	4.9E-01	5.2E-02	8.2E-02	1.7E-01	1.1E-02	9.1E-03
Michalewicz	7.9E-16	1.7E-02	8.1E-02	1.4E-02	3.0E-01	4.1E-06	7.5E-06
Rastrigin	0.0E+00	1.2E+00	3.9E-01	6.8E-01	7.4E-01	1.2E-04	1.0E-04
Rosenbrock	2.5E-02	1.0E+03	1.9E+01	7.3E+00	1.0E+01	2.2E+00	1.6E+01
Schaffer	4.8E-03	4.0E-02	7.8E-02	1.6E-02	0.0E+00	1.3E-02	1.3E-02
Sphere	0.0E+00	4.2E+01	1.9E-02	2.2E+00	1.1E-02	2.0E-04	4.3E-04
Schwefel	0.0E+00	3.1E+01	7.9E+00	1.5E-01	8.5E+01	3.3E-04	4.4E-04
Weierstrass	0.0E+00	4.8E-01	3.6E-01	5.9E-02	2.9E-01	7.4E-03	1.4E-02

solved successfully in nine runs. The ABC, MBO and BEGA were selected based on the result of these runs, and their performance was studied further in higher-dimensional spaces. The NFE and the number of runs remained the same as those used in 5-dimentional space. The results of these runs are reported in Tables 5, 6, 7, and 8, and in Figs. 3 and 4.

It is seen in Table 3 that the best performance belongs to the ABC. The solutions found by the ABC for all the test functions are significantly better than those calculated by the other algorithms. The ABC also found a near global optimal solution for all functions, whereas each of the other algorithms did not achieve good convergence to the global optimum in at least one of the problems. MBO and BEGA had good performances but were inferior to the ABC. Their solutions are not as good as those of the ABC but they approached the global optimum except for Rosenbrock function, which all of the algorithms failed to solve, except the ABC. The accuracies of the solutions found by the MBO and BEGA were also better than those of the BA, BCO, BS, and BSO. The performances of the MBO and BEGA were almost identical, while for some problems one of them found a slightly better solution than the other. Table 4 establishes the smaller standard deviation (STD) in most cases belongs to the ABC. We also see the same pattern when comparing the MBO and BEGA with respect to their STDs.

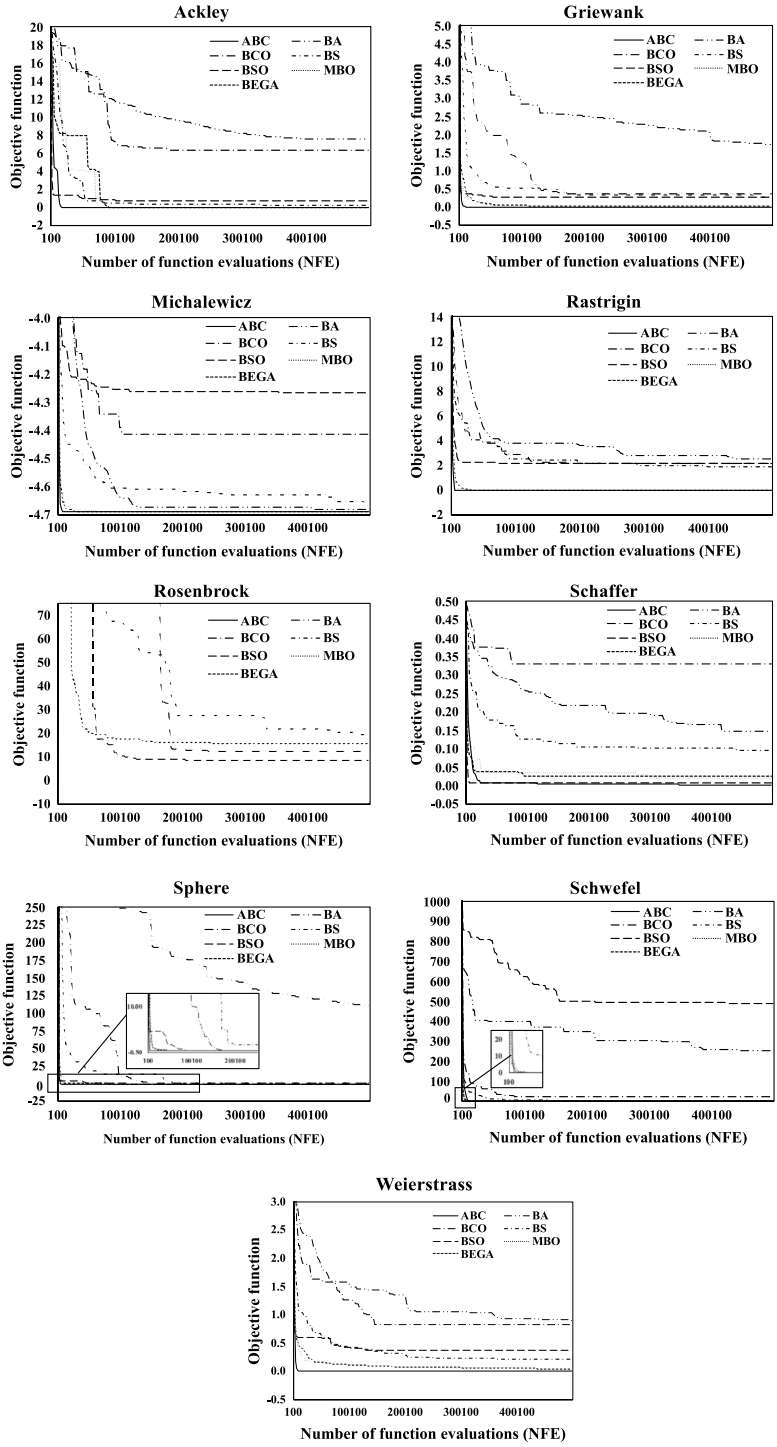

Fig. 1 Average objective function of runs of all algorithms for 5-dimensional test functions

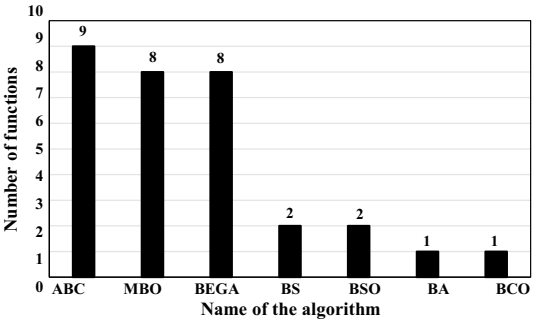

Fig. 2 The number of 5-dimensional test functions among nine test functions for which each algorithm found a near global optimum within a margin of 2.00E-01 with 500,000 functional evaluations (NFE)

Table 5 Average of five runs of the algorithms for test functions with 10-dimensional decision space (absolute zeros refer to any number less than 1.0E-17)

Function	ABC	MBO	BEGA
Ackley	4.00E+00	2.00E+01	1.60E+01
Griewank	0.00E+00	1.16E-01	6.54E-02
Michalewicz	− 9.66E+00	− 9.66E+00	− 9.66E+00
Rastrigin	0.00E+00	5.54E-03	7.23E-04
Rosenbrock	2.55E-02	2.47E+01	1.30E+01
Schaffer	1.52E-02	2.20E-01	8.95E-02
Sphere	0.00E+00	2.02E-02	3.08E-03
Schwefel	1.27E-04	4.51E-02	6.07E-03
Weierstrass	0.00E+00	2.62E-01	1.25E-01

Table 6 Standard deviation of five runs of the algorithms for test functions with 10-dimensional decision space (absolute zeros refer to any number less than 1.0E-17)

Function	ABC	MBO	BEGA
Ackley	8.00E+00	4.00E-03	8.00E+00
Griewank	0.00E+00	2.20E-02	2.94E-02
Michalewicz	7.94E-16	1.50E-03	5.56E-05
Rastrigin	0.00E+00	1.42E-03	2.42E-04
Rosenbrock	1.66E-02	1.67E+01	1.30E+01
Schaffer	1.10E-02	5.32E-02	3.41E-02
Sphere	0.00E+00	7.80E-03	1.53E-03
Schwefel	0.00E+00	2.95E-02	2.09E-03
Weierstrass	0.00E+00	5.20E-02	1.92E-02

Table 7 Average of five runs of the algorithms for test functions with 50-dimensional decision space (absolute zeros refer to any number less than 1.0E-17)

Function	ABC	MBO	BEGA
Ackley	2.00E+01	2.09E+01	2.00E+01
Griewank	0.00E+00	1.54E+02	1.01E+00
Michalewicz	− 4.95E+01	− 3.16E+01	− 4.92E+01
Rastrigin	0.00E+00	2.58E+02	1.44E+00
Rosenbrock	1.59E-01	8.83E+07	4.48E+02
Schaffer	4.57E-01	5.00E-01	4.42E-01
Sphere	0.00E+00	1.74E+04	2.38E+00
Schwefel	6.36E-04	6.58E+03	8.63E+00
Weierstrass	1.42E-14	4.05E+01	2.88E+00

Table 8 Standard deviation of five runs of the algorithms for test functions with 50-dimensional decision space (absolute zeros refer to any number less than 1.0E-17)

Function	ABC	MBO	BEGA
Ackley	3.92E-05	2.45E-02	1.65E-05
Griewank	0.00E+00	2.42E+01	2.29E-02
Michalewicz	1.44E-02	7.99E-01	6.38E-02
Rastrigin	0.00E+00	1.58E+01	1.26E-01
Rosenbrock	1.12E-01	2.33E+07	1.18E+02
Schaffer	9.84E-03	1.18E-05	1.52E-02
Sphere	0.00E+00	9.98E+02	7.46E-01
Schwefel	1.46E-12	4.74E+02	1.71E+00
Weierstrass	8.99E-15	7.90E-01	7.50E-02

Defining a margin equal to 2.0E-1 in the objective space about the exact global optimum as a near global optimum implied the BA, BCO, BS, and BSO did not converge to within the marginal space so defined for several functions. The BA and BCO approached the global optima of the Michalewicz and Sphere functions, respectively. The BS and BSO did slightly better and approached the global optima of the two problems. Also, even for the functions for which these algorithms approached the global optima correctly, they did not find a competitive solution in comparison to the ABC, MBO, and BEGA. Figure 2 depicts the number of problems solved correctly by each algorithm.

Tables 5 and 7 list the average of the five runs for the algorithms for 10- and 50- dimensional spaces, respectively. It is seen in Tables 5 and 7 that the performance of the ABC was the best and approached the global optima better in comparison to the other algorithms. The MBO and BEGA did not show similar performance in higher dimensional spaces as they did in 5-dimensional space. BEGA performed more accurately and faster than the MBO. By comparing the STD of the runs in Table 6 and 8 it follows that the MBO had a lower (better) STD, yet this does not mean it had a better performance. For example, it is seen in Table 6 that the STD of the ABC and BEGA for function Ackley is larger than the MBO's. As seen in Table 5 the averages of the ABC and BEGA were closer to the global optima. In fact, this implies the MBO was trapped in a local optimum whereas the BEGA and ABC converged to the global optimum in several but not all runs. This obviously reduces the average obtained but increases the STD. Table 8 indicates that none of the algorithms approached the global optimum for the Ackley function; at the same time the STD was low for all of them. For most of the functions even in higher dimensions MBO, BEGA, and ABC approached well the global optima with the same NFE like the 5-dimensional decision space. However, this was not the case for the Ackley function. The results confirm that larger search spaces require more computational effort. For example, if instead of NFE = 500,000 the algorithms conduct at least 1,000,000 NFE, then the average of five runs of the ABC, MBO, and BEGA are 9.12E-04, 2.00E+01, and 2.00E+01, respectively. The STD of ABC, MBO, and BEGA are in this case 1.73E-03, 2.32E-03, and 6.12E-08, respectively. Small STDs demonstrate the convergence of the algorithm to common optima. However, not all of them converge to the global optimum. A larger NFE results in ABC being able to converge to the global optimum well, but the MBO and BEGA were trapped at local optima even after doubling the number of NFE, and experienced premature convergence. The lesson here is that although all of these algorithms could be shown to eventually converge to the global optimum (Rudolph 2012), in practice, however, it may take them an impractically long computational time to escape from a local optimum.

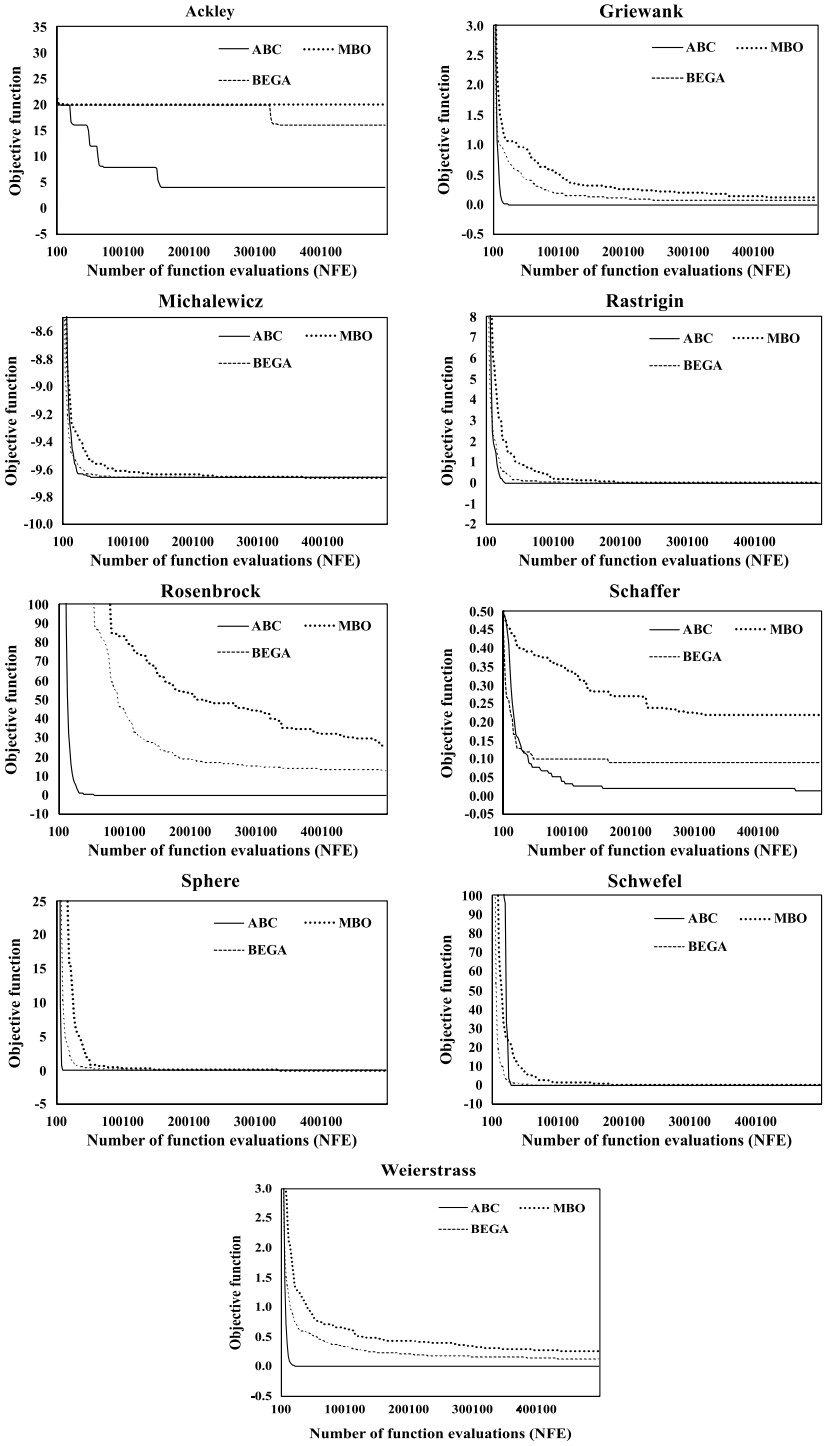

Fig. 3 Average objective function of runs of ABC, MBO, and BEGA for 10-dimensional test functions

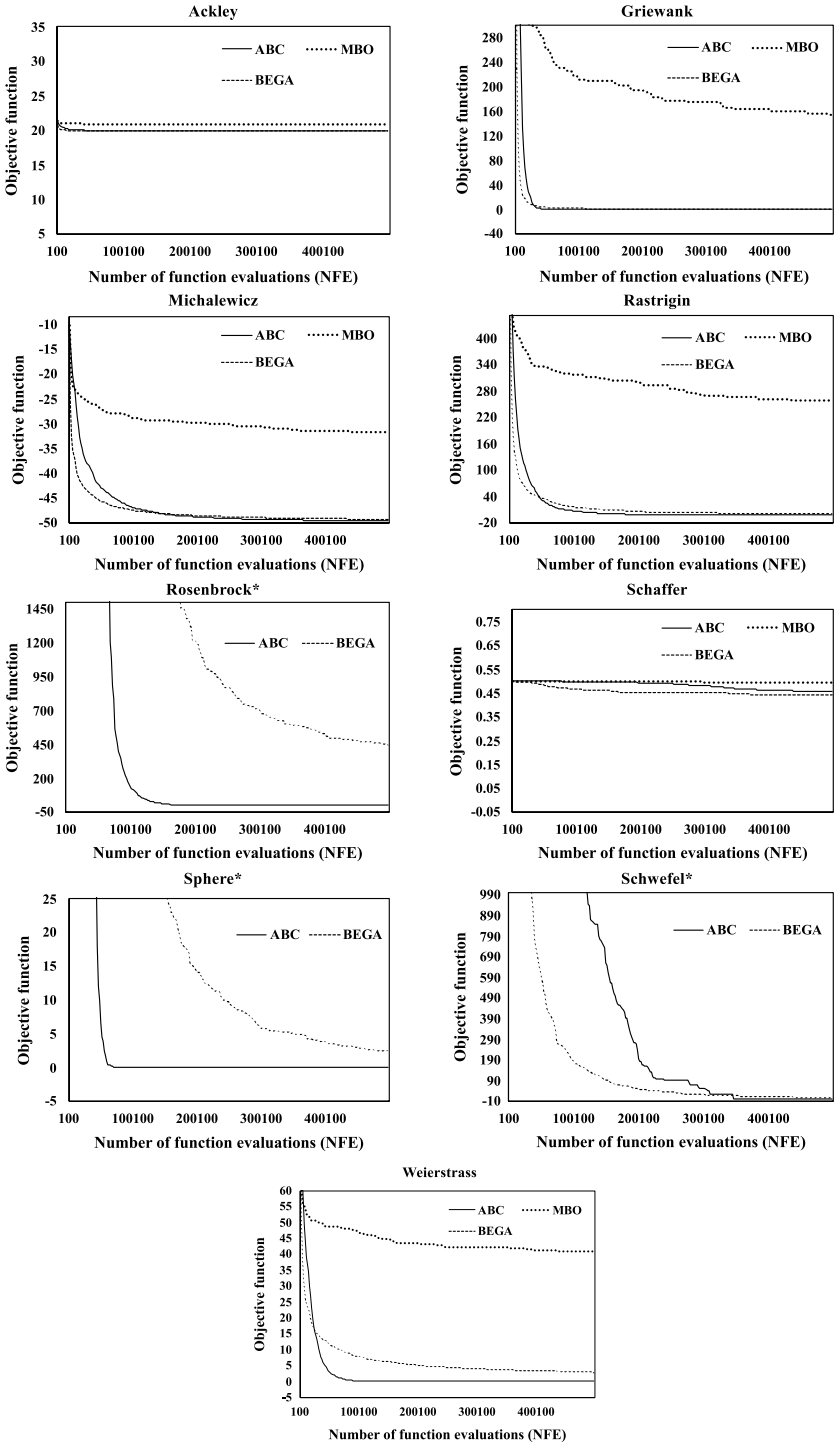

Fig. 4 Average objective function of runs of ABC, MBO, and BEGA for 50-dimensional test functions (* refers to functions for which MBO did not approach the optimum and is out of the range of the graph)

Overall, the ABC has been shown by our evaluation to be a powerful algorithm that solved the problems with different dimensions accurately even without any parameter adjustment. MBO and the BEGA performed fairly and were successful in solving the majority of problems. MBO and the BEGA can be categorized as efficient algorithms although they did not perform as well as ABC in the test problems. The other metaheuristics herein evaluated failed in solving most of the test problems.

5 Discussion

This study's results confirm the findings of previous works except for the BCO and BSO, in which case differences may arise by the fact that no parameter adjustment was affected in this study. It is common for the algorithms' developers to conduct a meticulous parameter adjustment for the algorithm. This work presented an unbiased comparison of several metaheuristics insofar as parameters adjustment is concerned. This work's results are implicitly supported by the relevant literature, wherein the ABC and MBO (or later called HBMO) have been reported more frequently than the other algorithms. Other versions of these algorithms such as the BCO may exhibit good performance for special problems of a combinatorial nature, for example. This work's purpose is not to judge previous works; rather, its primary goal is to implement several algorithms and compare their performances emphasizing the user perspective.

The comparison of foraging-inspired algorithms with those inspired by mating behavior of bees revealed that all mating inspired-algorithms provide a fair solution, whereas the algorithms developed based on the foraging behavior of bees were herein found to be more likely to fail in the search for optimal solutions. The better performance of mating-inspired algorithms may be explained by the fact that BSO, BA, and BCO apply a random walk in which the length of the steps is a parameter of the algorithm, directly or indirectly. On the other hand, the ABC generates a trial point between two selected solutions. Recall all metaheuristics can be divided into two phases of generating new solutions and selection. Functions used in these phases determine the success or failure of the search. The BSO and BA do not have any established and firm strategy for generating new solutions. Therefore, the result of the algorithm heavily depends on parameters adjustment that determines the length of the random walk. In the BCO this is worse because the length of the steps is dynamic, which makes the algorithm more complex when the gradual adjustment rate is determined by user-defined parameters. The mating-inspired algorithms on the other hand, implement the crossover operator when generating new solutions. The method used in the ABC is also a crossover function in the continuous domain. However, why does the ABC work better than its mating-inspired rivals, the MBO and BEGA? An initial guess is that the MBO and the BEGA conduct a concentrated search near the best solution found. MBO features one side of the crossover function as the best solution; the BEGA features half of the parent population of a simple GA as copies of the best solution. The ABC, on the other hand, performs a more scattered search: it gives more attention to the better solutions but does not dedicate the majority of its search capacity to the best solution. Therefore, the superiority of the ABC on its mating-inspired rivals seems to stem from the selection functions applied by rival algorithms. The ABC in general balances exploitation and exploration quite well in comparison to other algorithms. Table 9 provides a summary of the strengths and shortcomings of each algorithms.

Table 9 Summary of the strengths and weaknesses of the studied algorithms

Algorithm	Summary of the results
ABC	ABC is fast, efficient, and easy to implement
MBO	MBO has a high capability of approaching the global optimum but suffers from some inefficiency due to simulated annealing applied in its selection method
BEGA	BEGA is simple and effective but has more user defined parameters in comparison to its rivals
BS	BS is slow and inefficient but introduced the idea of sharing information among batches of solutions
BSO	BSO suffers from the lack of established strategy for generating new solutions
BA	BA has several serious design flaws
BCO	Unlike its combinatorial version, the continuous version of BCO is far from a mature and efficient algorithm

The BS's performance was herein found not to be exceptional among the evaluated algorithms. However, this algorithm has an interesting trait. The BS is one of the first algorithms, developed in 1997, and introduced the idea of sharing information among batches of solutions. An idea very similar to what later inspired well-known shuffled frog-leaping algorithm (SFLA; Eusuff and Lansey 2003). BS has not been widely applied according to our literature review. The main reason could be its complexity. Coding this algorithm was the most difficult among all the bee-inspired metaheuristics herein evaluated. Its number of parameters is also relatively large.

The MBO was the first algorithm that accurately simulated the bees' behavior to solve optimization problems. Its performance is competitive although is not the best among the evaluated algorithms. The MBO's selection method is computationally inefficient due to the application of simulated annealing. The simulated annealing selection method generates a random solution. If that random solution's fitness is good enough to pass a random criterion it enters the search; otherwise, the trial solution is deleted and another random solution is generated until the energy of the queen vanishes or a predefined number of solutions is selected. This is inefficient because whenever a totally random solution is generated its fitness function is evaluated, whereas most of the trial solutions are not successful in entering the search due to their low fitness and are eliminated immediately. Thus, a large amount of computational effort is invested in evaluating the fitness function of unsuccessful drones whose characteristic never participate in the search procedure. Yang et al. (2007) presented the fast marriage in honeybee optimization (FMBO) in which the simulated annealing process is eliminated, and all randomly generated solutions are accepted in the selection process without any filtering. Solgi et al. (2016) modified the MBO (or HBMO) and replaced the simulated annealing with a Boltzmann proportionate selection function.

The BEGA is a modified GA based on bee mating rituals. There are two differences between the standard GA and the BEGA. First, half of the parent population in the BEGA consists of copies of the queen (the best solution) and the remaining half is selected using a selection function, whereas in the simple GA all parents are selected randomly. The second difference is that part of the population is modified with strong mutation rather than normal or weak mutation. The BEGA is simple and effective. Its main disadvantage is determining the rates of normal and strong mutation and their probabilities, which are parameters that determines the BEGA's performance.

The BA is straightforward, and its basic idea is searching near the best and other good solutions, and replacing previous solutions with newly found, superior, solutions. Also, it devotes part of the population to search the decision space randomly. The main shortcoming with this algorithm is that it rests on the assumption that generating random solutions ensures good population diversity. This assumption is not true because randomly generated solutions does not ensure improvement from one generation to the next if they do not mix with the existed solutions appropriately. The probability of finding an improved solution by generating totally random vectors is very low. Diversity is enhanced by mixing random solutions with other solutions that exist in the population, which allows the algorithm search subspaces that are impossible to access with only resorting to combinations of existing solutions. The randomly generated solutions in the BA usually are not mixed with existing ones and thus have a very low chance of improving the new solutions. Another BA disadvantage is assigning a step for neighborhood search. This is problem-dependent, and the step specification is difficult to adjust properly without good knowledge of the decision space, which is usually unavailable. Another BA disadvantage is that a roulette wheel or other selection functions are not applied to assign onlookers to the good solutions in this algorithm. It is non-trivial for the user to decide the number of neighborhood solutions assigned to the elite and good solutions. Also, repetitive solutions in the population cause elite and good solutions to be nearly identical. Therefore, searching near them with different ratios is not effective.

The BCO leaves defining the strategy of generating new solutions almost entirely to the user. Kruger et al. (2016) proved the convergence of the BCO, but they also stated that generating new solutions with this algorithm is problem dependent. The primary question concerns the capacities of this algorithm. The reason metaheuristics are used is mostly for solving problems with unknown domains. Finding the best way to generate new solutions in unknown domains may be as difficult as finding the optimal solution. The BCO has been used in solving combinatorial problems frequently. Our results show the continuous version is far from being an effective and reliable algorithm.

Surprisingly, the best performance in our evaluation belongs to the algorithms which have the least number of parameters (i.e., the ABC, MBO, and BEGA). Parameters specification has a significant effect on algorithmic performance and hinders their applicability and their desirability. Most new varieties of algorithms added new user-defined parameters, and a few featured automatic parameter adjustments. From a coding perspective the ABC is the simplest to code among the evaluated algorithm. One of the appealing features of the ABC reported frequently in the literature is its simplicity. Most of the modified versions of the ABC have added extra parameters left to the user to adjust, and generally have rendered it more complex. This runs contrary to the fact that the ABC's popularity is partly explained by its relative simplicity. This work recommends that new versions of the algorithm must improve its performance while reducing the number of parameters and the algorithmic complexity. This recommendation is consistent with the law of parsimony (or Occam's philosophical razor) which posits that entities should not be multiplied without necessity.

Last but not least, slow convergence and trapping in local optima are common problems found in several evaluated metaheuristics. The convergence of metaheuristics has been assessed in several studies (i.e. Yang 2011; Rudolph, 2012; Kruger, et al. 2016); yet, the hitting time (convergence rate) of the algorithms is not easily predictable. This issue has rendered the metaheuristic development akin to an art more than a science. One phenomenon, bees' behavior, has inspired multiple metaheuristics. Our analysis of such metaheuristics suggests that the gained experience in algorithmic development

played a greater role in the appearance of new algorithms than newly gained knowledge of natural phenomena. Therefore, it appears timely to further study the metaheuristics' convergence rate theoretically and empirically.

6 Concluding remarks

This work studied bee-inspired metaheuristics for global optimization. Several basic search strategies have been developed claiming to be inspired from one single metaphor among which seven algorithms herein evaluated were the bee system (BS), mating bee optimization (MBO), bee colony optimization (BCO), bee evolution for genetic algorithms (BEGA), bee algorithm (BA), artificial bee colony (ABC), and bee swarm optimization (BSO). The latter algorithms feature original or basic (root) ideas or characteristics that have been shown to solve continuous problems. The performances of the algorithms were evaluated with nine benchmark functions. Strengths and shortcomings of each algorithm were discussed. The ABC had the best performance among all the bee-inspired algorithms and found a near global optimum for all problems, whereas other algorithms did not solve at least one problem each. The BEGA and MBO, each of which solved eight problems among nine problems, were the best performing algorithms after the ABC. The BEGA showed a better performance than the MBO when the dimensionality of the test problems increased. The ABC is the simplest algorithm with the smallest number of user-defined parameters (i.e., it is parsimonious). The ABC's simplicity and its good performance make it notable among its rivals. The BEGA and MBO are also relatively simple algorithms to code and execute. Dividing the source of inspiration of the bees' metaheuristics into the foraging behavior of bees and the mating rituals of bees revealed that all the latter algorithms yielded reliable performances, whereas the former features only one successful algorithm (i.e., the ABC). Our study of several bee-inspired algorithms indicates that successive development did not arise from a better understanding of natural phenomena, but, rather, from the cumulative experience gained from years of algorithmic trials and human ingenuity. Algorithm developers continue to relate their intuitions to natural phenomena. Yet, it seems timely to investigate more profoundly the reasoning behind the metaheuristics' convergence rate.

Funding Not relevant.

Availability of data and material Not relevant.

Code availability The codes are available at https://github.com/rmsolgi/bee-inspired-metaheuristics.

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

References

Abbass HA (2001) MBO: marriage in honey bees optimization a haplometrosis polygynous swarming approach. In: Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No.01TH8546), 27–30 May, Seoul, South Korea

Abbass H, Teo J (2003) A true annealing approach to the marriage in honey-bees optimization algorithm. Int J Comput Intell Appl 3(2):199–211

Abualigah LMQ (2019) Feature selection and enhanced krill herd algorithm for text document clustering. Studies in computational intelligence. Springer, Berlin

Abualigah L (2020) Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications. Neural Comput Appl 32:12381–12401

Abualigah L, Diabat A (2020) A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. Clust Comput. https://doi.org/10.1007/s10586-020-03075-5

Abualigah LMQ, Hanandeh ES (2015) Applying genetic algorithms to information retrieval using vector space model. Int J Comput Sci Eng Appl 5(1):19

Abualigah LM, Khader AT, Hanandeh ES (2017) A new feature selection method to improve the document clustering using particle swarm optimization algorithm. J Comput Sci 25:456–466

Abualigah LM, Khader AT, abd Hanandeh ES (2018a) Hybrid clustering analysis using improved krill herd algorithm. Appl Intell 48:4047–4071

Abualigah LM, Khader AT, Hanandeh ES (2018b) A combination of objective functions and hybrid krill herd algorithm for text document clustering analysis. Eng Appl Artif Intell 73:111–125

Abualigah L, Diabat A, Geem ZW (2020a) A comprehensive survey of the harmony search algorithm in clustering applications. Appl Sci 10(11):3827

Abualigah L, Shehab M, Alshinwan M, Mirjalili S, Elaziz MA (2020b) Ant lion optimizer: a comprehensive survey of its variants and applications. Arch Comput Methods Eng. https://doi.org/10.1007/s11831-020-09420-6

Akbari R, Mohammadi A, Ziarati K (2010) A novel bee swarm optimization algorithm for numerical function optimization. Commun Nonlinear Sci Number Simulat 15:3142–3155

Ashghari S, Jafari Navimipour N (2019a) Cloud service composition using an inverted ant colony optimization algorithm. Int J Bio-Inspir Comput 13(4):257

Ashghari S, Jafari Navimipour N (2019b) Resource discovery in the peer to peer networks using an inverted ant colony optimization algorithm. Peer Peer Netw Appl 12:129–142

Aslan S (2019) A transition control mechanism for artificial bee colony (ABC) algorithm. Comput Intell Neurosci 2019:5012313

Aslan S, Badem H, Karaboga D (2019) Improved quick artificial bee colony (iqABC) algorithm for global optimization. Soft Comput 23:13161–13182

Banharnsakun A, Achalakul T, Sirinaovakul B (2011) The best-so-far selection in artificial bee colony algorithm. Appl Soft Comput 11:2888–2901

Barker JSF (1958) Simulation of genetic systems by automatic digital computers. Aust J Biol Sci 11(4):603–612

Box GEP (1957) Evolutionary operation: a method for increasing industrial productivity. Appl Stat 6(2):81–101

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge Uni. Press, Cambridge

Bozorg-Haddad O, Afshar A, Marino MA (2006) Honey-bees mating optimization (HBMO) algorithm: a new heuristic approach for water resources optimization. Water Resour Manag 20:661–680

Bozorg-Haddad O, Hoseini-Ghafari S, Solgi M, Loaiciga HA (2016a) Intermittent urban water supply with protection of consumer's welfare. J Pipeline Syst Eng Pract 7(3):04016002

Bozorg-Haddad O, Ghajarnia N, Solgi M, Loaiciga HA (2016b) A DSS based honey bee mating optimization (HBMO) algorithm for single- and multi-objective design of water distribution networks. In: Metaheuristic and optimization in civil engineering. Springer, Cham, pp 199–233

Bozorg-Haddad O, Ghajarnia N, Solgi M, Loaiciga HA, Marino MA (2017a) Multi-objective design of water distribution systems based on the fuzzy reliability index. J Water Supply Res Technol 66(1):36–48

Bozorg-Haddad O, Solgi M, Loaiciga HA (2017b) Meta-heuristic and evolutionary algorithms for engineering optimization. Wiley, New York

Bremermann HJ (1962) Optimization through evolution and recombination. In: Yovits MC, Jacobi GT, Goldstein GD (eds) Self-organized systems. Spartan Books, Washington

Celik Y, Ulker E (2013) An improved marriage in honey bees optimization algorithm for single objective constrained optimization. Sci World J 2013:370172

Chen X, Tianfield H, Li K (2019) Self-adaptive differential bee colony algorithm for global optimization problem. Swarm Evol Comput 45:70–91
Comellas F, Mrtinez-Navaro J (2009) Bumblebees: a multiagent combinatorial optimization algorithm inspired by social insect behavior. In: Proceedings of the first ACM/SIGEVO summit on genetic evolutionary computation, 12–14 June, Shanghai, China
Cui L, Li G, Luo Y, Chen F, Ming Z, Lu N, Lu J (2018) An enhanced artificial bee colony algorithm with dual-population framework. Swarm Evol Comput 43:184–206
Darwish A, Hassanien AE, Das S (2019) A survey of swarm and evolutionary computing approaches for deep learning. Artif Intell Rev 53:1767–1812
De Jong K, Fogel DB, Schwefel HP (1997) A history of evolutionary computation. In: Back T, Fogel DB, Michalewicz Z (eds) Handbook of evolutionary computation. IOP publishing Ltd and Oxford University Press, Oxford
Dereli S, Koker R (2019) A metaheuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm. Artif Intell Rev 53:949–964
Dorigo M, Maniezzo V, Colorni A (1991) Positive feedback as a search strategy. Dipartimento di Elettronica, Politecnico di Milano, Milano, Technical Report No 91-016
Dorigo M, Maniezzo V, Colorni A (1996) The ant system: Optimization by a colony of cooperating ants. IEEE Trans Syst Man Cybern Part B 26(1):29–42
Eusuff MM, Lansey KE (2003) Application of the shuffled frog leaping algorithm for the optimization of a general large-scale water supply system. Water Resour Manag 23(4):797–823
Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence through simulated evolution. Wiley, New York
Friedberg RM (1958) A learning machine: part I. IBM J Res Dev 2(1):2–13
Gao WF, Liu SY, Huang LL (2013a) A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. IEEE Trans Cybern 43(3):1011
Gao W, Liu S, Huang L (2013b) A global best artificial bee colony algorithm for global optimization. J Comput Appl Math 236:2741–2753
Gao WF, Huang LL, Liu SY, Dai C (2015) Artificial bee colony algorithm based on information learning. IEEE Trans Cybern 45(12):2827
Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13:533–549
Gupta S, Deep K (2019) Hybrid sin cosine artificial bee colony algorithm for global optimization and image segmentation. Neural Comput Appl 32:9521–9543
Hajimirzaei B, Jafari Navimipour N (2019) Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. ICT Express 5(1):56
Hillier FS, Liberman GJ (1995) Introduction to operations research, 6th edn. McGraw-Hill, New York
Holland JH (1967) Nonlinear environments permitting efficient adaptation. Computer and information sciences II. Academic Press Inc, New York
Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor
Hooke R, Jeeves TA (1961) Direct search solution of numerical and statistical problems. J ACM 8(2):212–229
Hussein WA, Sahran S, Sheikh Abdullah SNH (2016) The variants of the bees algorithm (BA): s survey. Artif Intell Rev 47(1):67
Jong GJ, Horng GJ (2017) A novel queen honey bee migration (QHBM) algorithm for sink repositioning in wireless sensor network. Wirel Pers Commun 95:3209–3232
Jung SH (2003) Queen-bee evolution for genetic algorithms. Electron Lett 39(6):575
Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Erciyes University, Technical Report-TR06, Kayseri, Turkey
Karaboga D, Akay B (2009) A survey: algorithms simulating bee swarm intelligence. Artif Intell Rev 31:61–85
Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8:687–698
Karaboga D, Gorkemli B, Ozturk C, Karaboga N (2012) A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif Intell Rev 42:21–57
Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceeding of international conference on neural networks, Perth, Australia, November 27 to December 1, Institute of Electrical and Electronics Engineers (IEEE), Piscataway, NJ, pp 1942–1948
Khan L, Ullah I, Saeed T, Lo KL (2010) Virtual bees algorithm based design of damping control system for TCSC. Aust J Basic Appl Sci 4(1):1–18
Kirkpatrick S, Gelatte CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4589):671–680

Koc (2010) The bees algorithm theory, improvements and applications. PhD thesis, Cardiff University, Cardiff, UK
Kruger TJ, Davidovic T, Teodorovic D, Selmic M (2016) The bee colony optimization algorithm and its convergence. Int J Bio Inspir Comput 8(5):340
Lucic P (2002) Modeling transportation problems using concepts of swarm intelligence and soft computing. PhD thesis, Virginia Polytechnic Institute and State University, Virginia, USA
Mernik M, Liu SH, Karaboga D, Crepinsek M (2015) On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation. Inf Sci 291:115–127
Ming H, Baohui J, Xu L (2010) An improved bee evolutionary genetic algorithm. In: IEEE international conference on intelligent computation and intelligent systems, 29–31 October, Xiamen, China
Molga M, Smutnicki C (2005) Test functions for optimization needs. http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf. Accessed Nov 2020
Moradipari A, Alizadeh M (2018) Pricing differentiated services in an electric vehicle public charging station network. In: 57th IEEE conference on decision and control (CDC), December 17–19, FL, USA
Nasrinpour HR, Bavani MA, Teshnehlab M (2017) Grouped bees algorithm: a grouped version of the bees algorithm. Computers 6(1):5
Nikolic M, Teodorovic D (2013) Empirical study of the bee colony optimization (BCO) algorithm. Expert Syst Appl 40:4609–4620
Panahi V, Jafari Navimipour N (2019) Join query optimization in the distributed database system using an artificial bee colony algorithm and genetic operators. Concurr Comput Pract Exp 31(17):e5218
Pham DT, Darwish AH (2008) Fuzzy selection of local search sites in the bees algorithm. In: Pham DT, Eldukhri EE, Soroka AJ (eds) Innovative production machines and systems. Cardiff University, Cardiff
Pham DT, Ghanbarzadeh A, Koc E, Otri S, Rahim S, Zaidi M (2005) Bee algorithm a novel approach to function optimization. Technical Note: MEC 0501, Cardiff University, Cardiff, UK
Pham QT, Pham DT, Castellani M (2011) A modified bees algorithm and a statistics-based method for tuning its parameters. Proc Inst Mech Eng Part I J Syst Control Eng 226:287–301
Poolsamran P, Thammano A (2011) A modified marriage in honey-bee optimization for function optimization problems. Procedia Comput Sci 6:335–342
Qin Q, Cheng S, Zhang Q, Li L, Shi Y (2015) Artificial bee colony algorithm with time varying strategy. In: Discrete Dynamics in Nature and Society, 2015, 674595
Quijano N, Passino KM (2010) Honey bee social foraging algorithms for resource allocation: theory and Application. Eng Appl Artif Intell 23(6):845
Rabe M, Deininger M (2012) State of art and research demands for simulation modeling of green supply chains. Int J Autom Technol 6(3):296
Rechenberg I (1965) Cybernetic solution path of an experimental problem. Royal Aircraft Establishment Library Translation 1122
Rudolph G (2012) Stochastic convergence. In: Rozenberg G, Back T, Kok JN (eds) Handbook of natural computing. Springer, Berlin, pp 847–869
Sato T, Hagiwara M (1997) Bee system: finding solution by a concentrated search. In: IEEE international conference on systems, man, and cybernetics. computational cybernetics and simulation, 12–15 October, Orlando, FL, USA
Solgi M, Bozorg-Haddad O, Seifollahi Aghmiuni S, Ghasemi-Abiazani P, Loaiciga HA (2016) Optimal operation of water distribution networks under water shortage considering water quality. J Pipeline Syst Eng Pract 7(3):04016005
Solgi M, Bozorg-Haddad O, Loaiciga HA (2017) The enhanced honey-bee mating optimization algorithm for water resources optimization. Water Resour Manag 31:885–901
Sorensen K, Sevaux M, Glover F (2017) A history of metaheuristics. In: Marti R, Pardalos P, Resende M (eds) Handbook of heuristics. Springer, Berlin
Starke S, Hendrich N, Zhang J (2019) Memetic evolution for genetic full-body inverse kinematics in robotics and animation. IEEE Trans Evol Comput 23(3):406
Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
Tsai P, Chu SC, Pan JS (2009) Enhanced artificial bee colony optimization. Int J Innov Comput Inf Control 5(12):5081
Wang B, Wang L (2012) A novel artificial bee colony algorithm for numerical function optimization. In: Fourth international conference on computational and information sciences, 17–19 August, Chongqing, China

Wedde HF, Farooq M, Zhang Y (2004) BeeHive: an efficient fault-tolerant routing algorithm inspired by honey bee behavior. In: Dorigo M, Birattari M, Blum C, Gambardella LM, Mondada F, Stutzle Th (eds) Ant colony optimization and swarm intelligence. Springer, Berlin

Xiang W, An M (2013) An efficient and robust artificial bee colony algorithm for numerical optimization. Comput Oper Res 40:1256–1265

Xu C, Zhang Q, Li J, Zhao X (2008) A bee swarm genetic algorithm for the optimization of DNA encoding. In: The 3rd international conference on innovative computing information and control (ICICIC'08), 18–20 June, Dalian, China

Xu B, Zhang M, Browne WM, Yao X (2016) A survey on evolutionary computation approached to feature selection. IEEE Trans Evol Comput 20(4)

Yang XS (2011) Metaheuristic optimization: algorithm analysis and open problems. In: Pardalos PM, Rebennack S (eds) SEA 2011, LNCS 6630. Springer, Berlin

Yang C, Chen J, Tu X (2007) Algorithm of fast marriage in honey bees optimization and convergence analysis. In: Proceedings of IEEE international conference on automation and logistics, August 18–21, Jinan, China

Yuce B, Packianather MS, Mastrocinque E, Pham DT, Lambiase A (2013) Honey bees inspired optimization method: the bees algorithm. Insects 4:646–662

Zanbouri K, Jafari Navimipour N (2019) A cloud service composition method using a trust-based clustering algorithm and honeybee mating optimization algorithm. Int J Commun Syst 33:e4259

Zhu G, Kwong S (2010) Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl Math Comput 217:3166–3172