Artificial Intelligence Review
https://doi.org/10.1007/s10462-018-09676-2

From ants to whales: metaheuristics for all tastes

Fernando Fausto[1] · Adolfo Reyna-Orta[2] · Erik Cuevas[1] · Ángel G. Andrade[2] · Marco Perez-Cisneros[1]

Abstract
Nature-inspired metaheuristics comprise a compelling family of optimization techniques. These algorithms are designed with the idea of emulating some kind natural phenomena (such as the theory of evolution, the collective behavior of groups of animals, the laws of physics or the behavior and lifestyle of human beings) and applying them to solve complex problems. Nature-inspired methods have taken the area of mathematical optimization by storm. Only in the last few years, literature related to the development of this kind of techniques and their applications has experienced an unprecedented increase, with hundreds of new papers being published every single year. In this paper, we analyze some of the most popular nature-inspired optimization methods currently reported on the literature, while also discussing their applications for solving real-world problems and their impact on the current literature. Furthermore, we open discussion on several research gaps and areas of opportunity that are yet to be explored within this promising area of science.

Keywords Nature-inspired metaheuristics · Bio-inspired algorithms · Optimization, review

✉ Fernando Fausto
abraham.fausto@academicos.udg.mx

✉ Erik Cuevas
erik.cuevas@cucei.udg.mx

Adolfo Reyna-Orta
adolfo.reyna@uabc.edu.mx

Ángel G. Andrade
aandrade@uabc.edu.mx

Marco Perez-Cisneros
marco.perez@cucei.udg.mx

1 Departamento de Electrónica, Universidad de Guadalajara, CUCEI,, Av. Revolución 1500 44430 Guadalajara, Mexico

2 Facultad de Ingenierías, Universidad Autónoma de Baja California, Blvd. Benito Juárez, 21280 Mexicali, Mexico

Published online: 19 January 2019

1 Introduction

Mathematical optimization is a branch of applied mathematics and computer sciences which deals with the selection of the optimal solution for a particular mathematical function (or problem) with the purpose of either minimizing or maximizing the output of such function. In more simple terms, optimization could be described as the process of selecting of the best element(s) from among a set of available alternatives to get the best possible results when solving a particular problem (Galinier et al. 2013; Cuevas et al. 2016). Optimization is a recurring problem for many different areas of application such as robotics, computer networks, security, engineering design, data mining, finances, economics, and many others (Cuevas et al. 2016). Independently of the area of application, optimization problems are wide-ranging and numerous, so much that the development of methods for solving such problems has remained a hot topic for many years.

Traditionally, optimization techniques can be roughly classified as either deterministic or stochastic (Cavazzuti 2013). Deterministic optimization approaches, which design heavily relies on mathematical formulation and its properties, are known to have some remarkable advantages, such as fast convergence and implementation simplicity (Lin et al. 2012). On the other hand, stochastic approaches, which resort to the integration of randomness into the optimization process, stand as promising alternatives to deterministic methods for being far less dependent on problem formulation and due to their ability to thoroughly explore a problems design space, which in turn allow them to overcome local optima more efficiently (Schneider and Kirkpatrick 2006). While both deterministic and stochastic methods have been successfully applied to solve a wide variety of optimization problems, these classical approaches are known to be subject to some significant limitations; first of all, deterministic methods are often conditioned by problem properties (such as differentiability in the case of gradient-based optimization approaches) (Cuevas et al. 2017a). Furthermore, due to their nature, deterministic methods are highly susceptible to get trapped into local optima, which is something undesirable for most (if not all) applications. As for stochastic techniques, while these are far easier to adapt to most black-box formulations or ill-behaved optimization problems, these methods tend to have a notably slower convergence speed in comparison to their deterministic counterparts, which naturally pose as an important limitation for applications where time is critical. The many shortcomings of classical methods, along with the inherent challenges of real-life optimization problems, eventually lead researchers to the development of heuristics as an alternative to tackle such complex problems (Galinier et al. 2013). Generally speaking, a heuristic could be described as a technique specifically tailored for solving specific problems, often considered too difficult to handle with classic techniques. In this sense, heuristics trade essential qualities such as optimality, accuracy, precision or completeness to, either solve a problem in reasonably less time or to find an approximate solution in situations in which traditional methods fail to deliver an exact solution. However, while heuristic methods have demonstrated to be excellent to handle otherwise hard to solve problems, there are still subject to some issues. Like most traditional approaches, heuristics are usually developed by considering at least some specifications about the target problem, and as such, it is hard to apply them to different problems without changing some or most of their original framework (Díaz-Cortés et al. 2017).

Recently, the idea of developing methodologies that could potentially solve a wide variety of problems in a generic fashion has caught the attention of many researchers, leading to the development of a new breed of "intelligent" optimization techniques formally known as metaheuristics (Yang 2008). A metaheuristic is a particular kind of heuristic-based methodology, devised with the idea of being able to solve many different problems without the need of changing the algorithms basic framework. For this purpose, metaheuristic techniques employ a series of generic procedures and abstractions aimed to improve a set of candidate solution iteratively. With that being said, metaheuristics are often praised due to their ability to find adequate solutions for most problems independently of their structure and properties.

1.1 Nature-inspired metaheuristics

The word "nature" refers to many phenomena observed in the physical world. It comprises virtually everything perceptible to our senses and even some things that are not as easy to perceive. Nature is the perfect example of adaptive problem solving; it has shown countless times how it can solve many different problems by applying an optimal strategy, suited to each particular natural phenomenon. Many researchers around the world have become captivated by how nature can adapt to such an extensive array of situations, and for many years they have tried to emulate these intriguing problem-solving schemes to develop tools with real-world applications. In fact, for the last two decades, nature has served as the most important source of inspiration in the development of metaheuristics. As a result of this, a whole new class of optimization techniques was given birth in the form of the so-called Nature-inspired optimization algorithms. These methods (often referred as bio-inspired algorithms) are a particular kind of metaheuristics, developed with a single idea in mind: mimicking a biological or a physical phenomenon to solve optimization problems. With that being said, depending on their source of inspiration, nature-inspired metaheuristics can be classified in four main categories: evolution-based, swarm-based, physics-based and human-based methods (Binitha and Sathya 2012; Mirjalili and Lewis 2016) (see Fig. 1). Evolution-based methods are developed by drawing inspiration in the laws of natural evolution. From these methods, the most popular is without a doubt the Genetic Algorithms approach, which simulates Darwinian evolution (Mitchell 1995). Other popular methods grouped within this category include Evolution Strategy (Back et al. 1991), Differential Evolution (Storn and Price 1997) and Genetic Programming (Sette and Boullart 2001). On the other hand, swarm-based techniques are devised to simulate the social and collective behavior manifested by groups of animals (such as birds, insects, fishes, and others). The Particle Swarm Optimization (Poli et al. 2007a) algorithm, which is inspired in the social behavior of bird flocking, stands as the most representative and successful example within this category, although other relevant methods include Ant Colony Optimization (Dorigo and Stützle 2004), Artificial Bee Colony (Karaboga and Basturk 2007), Firefly Algorithm (Yang 2010a), Social Spider Optimization (Cuevas et al. 2013a), among others. Also, there are the physics-based algorithms, which are developed with the idea of emulating the laws of physics observed within our universe. Some of the most popular methods grouped within this category are Simulated Annealing (Rutenbar 1989), Gravitational Search Algorithm (Rashedi et al. 2009), Electromagnetism-like Mechanism (Birbil and Fang 2003), States of Matter Search (Cuevas et al. 2013b), to name a few. Finally, we can mention human-based algorithms. These kind of nature-inspired methods are unique due to the fact that they draw inspiration from several phenomena commonly associated with humans' behaviors, lifestyle or perception. Some of the most well-known methods found in the literature include Harmony Search (Geem et al. 2001), Firework Algorithm (Tan and Zhu 2010), Imperialist Competitive Algorithm (Atashpaz-Gargari and Lucas 2007), and many more.

Most nature-inspired methods are modeled as population-based algorithms, in which a group of randomly generated search agents (often referred as individuals) explore different

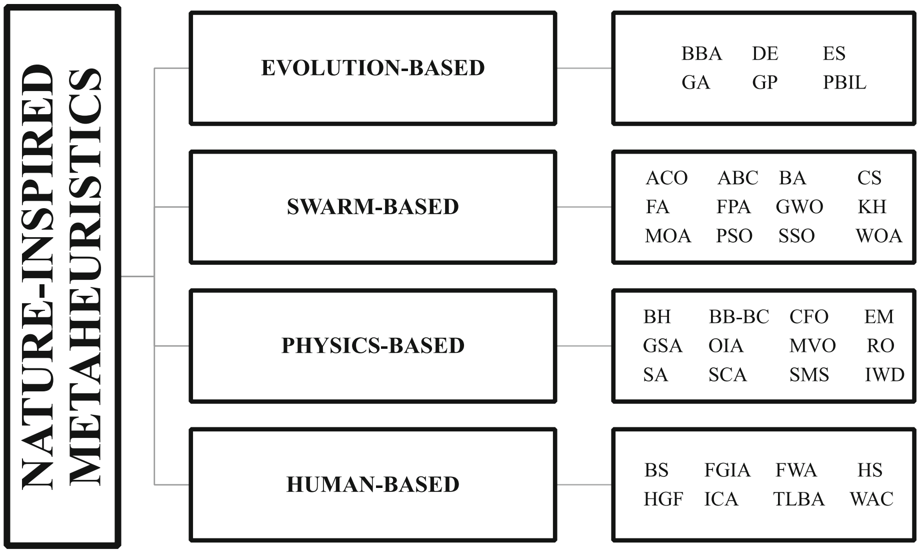

Fig. 1 Classification of nature-inspired metaheuristics

candidate solutions by applying a particular set of rules derived from some specific natural phenomenon. This kind of frameworks offer important advantages in both, the interaction among individuals, which promotes a wider knowledge about different solutions, and the diversity of the population, which is an important aspect on ensuring that the algorithm has the power to efficiently explore the design space while also being able to overcome local optima (Yang 2008). Due to this and many other distinctive qualities, nature-inspired methods have become a popular choice among researchers. As a result, literature related to nature-inspired optimization algorithms and its applications for solving otherwise challenging optimization problems has become extremely vast, with hundreds of new papers being published every year.

In this paper, we present a broad review about nature-inspired optimization algorithms, highlight some of the most popular methods currently reported on the literature as well as some of its applications for solving real-world optimization problems and their impact on the current research. The rest of this paper is organized as follows: in Sect. 2, we analyze the general framework applied by most nature-inspired metaheuristics in terms of design. In Sect. 3, we present nature-inspired methods according to their classification while also reviewing some of the most popular algorithms for each case. In Sect. 4, we open a discussion related to the performance of nature-inspired techniques, emphasizing on several observable design characteristics which have a direct impact on this regard. In Sect. 5, we review some of the most important areas of application where nature-inspired approaches have been applied for solving real-world problems. In Sect. 6, we present a brief study concerning the growth in the number of publications related to nature-inspired methods. In Sect. 7, we discuss some of the current research gaps and areas of opportunity in this rather young area of science. Finally, in Sect. 8, we present our conclusions and final thoughts.

2 General framework of nature-inspired metaheuristics

With some exceptions, most of the nature-inspired metaheuristics currently reported on the literature are modeled as population-based algorithms, which implies that the general framework employed by most of these methods remains almost identical, independently of the natural phenomenon from which the algorithm is inspired (Cuevas et al. 2016).

Usually, the first step of a nature-inspired algorithm involves the definition of a set of N randomly initialized solutions $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ (commonly referred as *population*), and such that:

$$\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}] \tag{1}$$

where the elements $x_{i,n}$ represent the decision variables (parameters) related to a given optimization problem, while d denotes the dimensionality (number of decision variables) of the target solution space.

From an optimization point of view, each set of parameters $\mathbf{x}_i \in \mathbf{X}$ (also known as an *individual*) is considered as a candidate solution for the specified optimization task; as such, each of these solutions is also assigned with a corresponding quality value (or *fitness*) related to the objective function $f(\cdot)$ that describes the optimization task, such that:

$$f_i = f(\mathbf{x}_i) \tag{2}$$

Nature-inspired methods usually follow an iterative search scheme, in which new candidate solutions are generated by modifying currently available individuals; this is achieved by applying some previously specified criteria (usually devised by drawing inspiration from an observed natural phenomenon). For most cases, this process may be illustrated by the following expression:

$$\mathbf{x}'_i = \mathbf{x}_i + \Delta\mathbf{x}_i \tag{3}$$

where $\mathbf{x}'_i$ denotes the candidate solution generated by adding up a specified update vector $\Delta\mathbf{x}_i$ to $\mathbf{x}_i$. It is worth noting that the value(s) adopted by the update vector $\Delta\mathbf{x}_i$ depend on the specific operators employed by each individual algorithm.

Finally, most nature-inspired algorithms include some kind of selection process, in which the newly generated solutions are compared against those in the current population $\mathbf{X}^k$ (with k denoting the current iteration) in terms of solution quality, typically with the purpose of choosing the best individual(s) among them. As a result of this process, a new set of solutions $\mathbf{X}^{k+1} = \{\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}, \ldots, \mathbf{x}_n^{k+1}\}$, corresponding to the following iteration (or *generation*) '$k + 1$', is generated.

This whole process is iteratively repeated until a particular stop criterion is met (i.e., if a maximum number of iterations is reached). Once this happens, the best solution found by the algorithm is reported as the best approximation for the global optimum (Cuevas et al. 2016).

3 Nature-inspired metaheuristics

Nature-Inspired optimization algorithms have become so numerous and so varied that illustrating every single method in existence has become an undoubtedly challenging task. However, several algorithms have become widely popular among researchers, either for their fascinating characteristics or their ease of implementation. In this section, we present some of the most popular nature-inspired optimization techniques currently reported on the literature. The algorithms presented in this section were chosen by considering a balance between both classical and modern approaches. Also, in order to give the reader the facility to understand, analyze and compare each of the described methods in the same terms, we have taken some liberties regarding nomenclature and formulation presented on each case so that it is consistent with the general framework of nature-inspired methods presented in Sect. 2. While the introduced formulations may look slightly different to those reported on their sources, we have made a special effort to keep the essence and particular traits that distinguish each method unaltered; with that being said, the reader is always invited to refer to the original paper(s) in order to get a deeper understanding of these techniques. All approaches described in this section are presented according to the typical classification of to nature-inspired metaheuristics (see Fig. 1): In Sect. 3.1., we center our discussion on evolution-based algorithms; in Sect. 3.2, we dedicate our analysis to swarm-based optimization methods; in Sect. 3.3, we analyze some of the most popular physics-based techniques; finally, in Sect. 3.4, human-based techniques are brought up to discussion.

3.1 Evolution-based methods

Evolution-Based methods comprise a series of optimization algorithms developed by drawing inspiration in the laws of natural evolution. In this kind of techniques, solutions are typically represented by a set of individuals, which compete and combine in ways that allow only the most suitable individuals to prevail. The process for modifying existent solutions in evolution-based techniques often involve the implementation of a series of operators inspired in several processes commonly observed in natural evolution, such as *crossover*, *mutation*, and *selection*.

3.1.1 Differential evolution

The Differential Evolution (DE) approach is an evolutionary algorithm introduced by Storn and Price (1997) and, along with Genetic Algorithms (GA), is one of the most popular optimization approaches inspired in the evolution phenomena.

At each generation 'k', DE applies a series of mutation, crossover and selection operators in order to allow a population of solutions $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N\}$ to "evolve" toward an optimal solution. For DE's mutation operation, new candidate (mutant) solutions $\mathbf{m}_i^k = [m_{i,1}^k, m_{i,2}^k, \ldots, m_{i,d}^k]$ are generated for each individual $\mathbf{x}_i$ as illustrated as follows:

$$\mathbf{m}_i^k = \mathbf{x}_{r_3}^k + F\left(\mathbf{x}_{r_1}^k - \mathbf{x}_{r_2}^k\right) \tag{4}$$

where $r_1, r_2, r_3 \in \{1, 2, \ldots, N\}$ (and with $r_1 \neq r_2 \neq r_3 \neq i$) each denote a randomly chosen solution index, while the parameter $F \in [0, 2]$ is called differential weight, and is used to control the magnitude of the differential variation $\left(\mathbf{x}_{r_1}^k - \mathbf{x}_{r_2}^k\right)$.

Furthermore, for the crossover operation, DE generates a trial solution vector $\mathbf{u}_i^k = [u_{i,1}^k, u_{i,2}^k, \ldots, u_{i,d}^k]$ corresponding to each population member 'i'. The components $u_{i,n}^k$ in such a trial vector are given by combining both the candidate solution $\mathbf{x}_i^k$ and its respective mutant solution $\mathbf{m}_i^k$ as follows:

$$u_{i,n}^k = \begin{cases} m_{i,n}^k & \text{if}(\text{rand} \leq CR) \text{ or } n = n^* \\ x_{i,n}^k & \text{if}(\text{rand} > CR) \text{ otherwise for } n = 1, 2, \ldots, d \end{cases} \tag{5}$$

where $n^* \in \{1, 2, \ldots, d\}$ denotes a randomly chosen dimension index, while rand stand for a random number from within the interval [0, 1]. Furthermore, the parameter $CR \in [0, 1]$ represents the DE's crossover rate which is used to control the probability of an element $u_{i,n}^k$ being given by either a component from the candidate solution $\mathbf{x}_i^k$ ($x_{i,n}^k$) or a component from the mutant solution $\mathbf{m}_i^k$ ($m_{i,n}^k$).

Finally, for DE's selection process, each trail solution $\mathbf{u}_i^k$ is compared against its respective candidate solution $\mathbf{x}_i^k$ in terms of solution quality (fitness) by applying a greedy criterion. This means that, if the trial solution $\mathbf{u}_i^k$ yields to a better fitness value than $\mathbf{x}_i^k$, then the value of the candidate solution for the next generation '$k + 1$' takes the value of $\mathbf{u}_i^k$, otherwise, it remains unchanged. This is:

$$\mathbf{x}_i^{k+1} = \begin{cases} \mathbf{u}_i^k \text{ if } \left(f\left(\mathbf{u}_i^k\right) > \text{if} f\left(\mathbf{x}_i^k\right)\right) \\ \mathbf{x}_i^k \text{ otherwise} \end{cases} \tag{6}$$

As one of the most popular Evolution-based algorithms currently reported on the literature, DE has been extensively studied and applied by researchers in many different areas of science (Opara and Arabas 2017; Padhye et al. 2013; Mohamed et al. 2012; Das and Suganthan 2011; Das et al. 2016; Piotrowski 2017).

3.1.2 Evolution strategies

Evolution strategies (ES) are a series of optimization techniques which draw inspiration from natural evolution (Back et al. 1991). The first ES approach was introduced by Ingo Rechenberg in the early 1960s and further developed during the 1970s. The most straightforward ES approach is the so-called (1 + 1)-ES (or two-membered ES). This approach considers the existence of only a single parent $\mathbf{x} = [x_1, x_2, \ldots, x_d]$, which is assumed to be able to produce a new candidate solution (offspring) $\mathbf{x}' = [x'_1, x'_2, \ldots, x'_d]$ by means of mutation as follows:

$$\mathbf{x}' = \mathbf{x} + \mathbf{N}(0, \sigma) \tag{7}$$

where $\mathbf{N}(0, \sigma)$ denotes a d-dimensional random vector whose values are drawn from a Gaussian distribution of mean 0 and fixed standart deviation σ (although later approaches consider a dynamic value based on the number of successful mutations) (Back et al. 1991).

Furthermore, the (1 + 1)-ES implements a selection operator which allows excluding the individual with the least performance between the parent $\mathbf{x}$ and its respective offspring $\mathbf{x}'$, so that only the best of these solution is considered as the parent for the next generation (iteration).

In later approaches, Rechemberg introduced the concept of population to ES by proposing the first multimembered ES in the form of the so-called (μ + 1)-ES. In such an approach, a population $\mathbf{P} = \{\mathbf{I}_1, \ldots, \mathbf{I}_\mu\}$ consisting on $\mu > 1$ parents $\mathbf{I}_i = \{\mathbf{x}_i, \boldsymbol{\sigma}_i\}$ (with $\mathbf{x}_i = \left[x_{i,1}, x_{i,2}, \ldots x_{i,d}\right]$ and $\boldsymbol{\sigma}_i = [\sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,d}]$) is considered. Furthermore, a discrete recombination mechanism which considers information drawn from a pair of randomly chosen parent is implemented to generate a new offspring $\mathbf{I}' = \{\mathbf{x}', \boldsymbol{\sigma}'\}$ as follows:

$$x'_j = \begin{cases} x_{r_1,n} & \text{if(rand} > 1/2) \\ x_{r_2,n} & \text{otherwise} \qquad \text{for } n = 1, 2, \ldots, d \end{cases} \tag{8}$$

$$\sigma'_j = \begin{cases} \sigma_{r_1,n} & \text{if(rand} > 1/2) \\ \sigma_{r_2,n} & \text{otherwise} \qquad \text{for } n = 1, 2, \ldots, d \end{cases} \tag{9}$$

where $r_1, r_2 \in \{1, \ldots, \mu\}$ denote two randomly chosen solution indexes corresponding to the parents population, while rand stand for a random number from within the interval [0,1].

Similarly to (1 + 1)-ES, (μ + 1)-ES also implements a mutation operator which generate a new offspring solution by perturbing a currently existing parent. Furthermore, a selection operator which allows to choose the best μ solutions from among the population of parents and offspring (generated through recombination and mutation) is also implemented to define a new parents' population for the next generation.

Later approaches, such as the ($\mu + \lambda$)-ES and the (μ, λ)-ES were further proposed to consider the generation of multiple offspring rather than a single one (Back et al. 1991). Furthermore, several variations to the recombination and mutation processes employed on classical ES have also been proposed, giving birth to some interesting variants such as (μ, λ)-MSC-ES and CMA-ES, the latter of which is considered by many authors as the state-of-the-art in ES (Bäck et al. 2013; Beyer and Sendhoff 2008; Auger et al. 2004; Salimans et al. 2017).

3.1.3 Genetic algorithms

Genetic algorithms (GA) is one of the earliest metaheuristics inspired in the concepts of natural selection and evolution and is among the most successful evolutionary algorithms (EA) due to its conceptual simplicity and easy implementation (Mitchell 1996). GA was initially developed by John Henry Holland in 1960 (and further extended in 1975) with the goal to understand the phenomenon of natural adaptation, and how this mechanism could be implemented into computers systems to solve complex problems.

In GA, a population of N solutions $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}]$ is first initialized; each of such solutions (called *chromosomes*) comprises a bitstring (this is, $x_{i,n} \in \{0, 1\}$), which further represents a possible solution for a particular binary problem. At each iteration (also called generation) of GA's evolution process, the chromosome population is modified by applying a set of three evolutionary operators, namely: selection, crossover and mutation. For the selection operation, GA randomly selects a pair of chromosomes $\mathbf{x}_{p_1}$ and $\mathbf{x}_{p_2}$ (with $p_1, p_2 \in \{1, 2, \ldots, N\}$ and $p_1 \neq p_2$) from within the entire chromosome population, based on their individual selection probabilities. The probability P_i for a given chromosome 'i' ($\mathbf{x}_i$) to be selected depends on its quality (fitness value), as given as follows:

$$P_i = \frac{f(\mathbf{x}_i)}{\sum_{j=1}^{N} f(\mathbf{x}_j)} \tag{10}$$

Then, for the crossover operation, the bitstring information of the selected chromosomes (now called parents) is recombined to produce two new chromosomes, $\mathbf{x}_{s_1}$ and $\mathbf{x}_{s_2}$ (referred as *offspring*) as follows:

$$x_{s_1,n} = \begin{cases} x_{p_1,n} & \text{if}(n < l) \\ v & \text{otherwise} \end{cases} \quad x_{s_2,n} = \begin{cases} x_{p_2,n} & \text{if}(n < l) \\ x_{p_1,n} & \text{otherwise for } n = 1, 2, \ldots, d \end{cases}. \tag{11}$$

where $l \in \{1, 2, \ldots, d\}$ is a randomly selected pivot index (usually referred as *locus*).

Finally, for the mutation operation, some elements (bits) of the newly generated offspring are flipped (changed from 1 to 0 or vice versa). Mutation can occur over each bit position in the string with a particular probability P_m (typically as low as 0.001), as given as follows:

$$x_{s_r,n} = \begin{cases} \bar{x}_{s_r,n} & \text{if}(\text{rand} < P_m) \\ x_{s_r,n} & \text{otherwise} \end{cases} \quad \text{for } n = 1, 2, \ldots, d \tag{12}$$

where $x_{s_r,n}$ (with $r \in \{1, 2\}$) stand for the jth element (bit) of the s_rth offspring, while rand stand for a random number form within the interval of 0 and 1.

This process of selection, crossover, and mutation of individuals takes place until a population of N new chromosomes (mutated offspring) has been produced, and then, the N best chromosomes among the original and new populations are taken for the next generation, while the remainder individuals are discarded (Sayed et al. 2017; McCall 2005; Yadav and Prajapati 2012; Pham et al. 2013).

3.1.4 Genetic programming

Genetic programming (GP) is a unique optimization technique proposed by John R. Koza in 1992 (Sette and Boullart 2001). The development of GP is closely related to popularization gained by evolutionary algorithms between the 1960s and 1970s. In essence, GP can be considered an extension of evolutionary methods such as Rechenberg's Evolution Strategies (ES) or Holland's Genetic Algorithms (GA). Different to said traditional methods, however, is the fact that in GP solutions are represented by a set of operations or computer programs; this is that, instead of finding a set of decision variables that optimize a given objective function, the outputs of GP are computer programs specifically tailored (evolved) to perform optimally on predefined tasks. Traditionally, solutions in GP are represented as tree structures which group a set of functions and operands, although other representations are also common. Furthermore, GP is also distinctive due to its variable-length representation of output solutions, which drastically differs to the fixed-length representations adopted by most traditional techniques (Khu et al. 2001; Poli et al. 2007b; Harman et al. 2013; Gerules and Janikow 2016; Vanneschi et al. 2014).

Typically, most GP approaches are comprised of the following four fundamental steps:

1. Generating an initial population of computer programs, composed by the available functions and terminals (operands).
2. Execute each program in the population and assign it a fitness value according to how well it solves a given problem.
3. Generate a new population of programs by:
   a. Copying the current best computer programs (reproduction).
   b. Creating new offspring programs by randomly changing some parts of a program (mutation).
   c. Creating new offspring programs by recombining parts from two existent programs (crossover).
4. If a specified stop criterion is met, return the single best program in the population as the solution for the pre-specified problem. Otherwise, return to step 2.

3.2 Swarm-based methods

Swarm-based optimization algorithms comprise a series of techniques which draw inspiration from the collective behavior manifested by a wide range of living organisms, such as birds, insects, fishes, and others. In this kind of techniques, search agents are modeled by a population of individuals (usually from the same species) which are capable of interacting with each other and the environment that surrounds them. While the movement of search agents in swarm algorithms is often based on simplified behavioral rules abstracted from those observed in nature, the collective manifestation of these individual conducts allows the entire population to exhibit global and complex behavioral patterns, thus allowing them to explore an extensive amount of candidate solutions.

3.2.1 Ant colony optimization

The ant colony optimization (ACO) algorithm is one of the most well-known nature-inspired metaheuristics. The ACO approach was first proposed by Marco Dorigo in 1992 under the name of ant systems (AS) and draws inspiration in the natural behavior of ants (Dorigo and Blum 2005). In nature, ants move randomly while foraging for food, and when an appropriate source is found, they return to their colony while leaving a pheromone trail behind. Ants are able to guide themselves toward previously found food source by following the path traced by pheromones left by them or other ants. However, as time passes, pheromones start to evaporate; intuitively, the more time an ant takes to travel down a given path back and forth, the more time the pheromones have to dissipate; on the other hand, shorter paths are traversed more frequently, promoting that pheromone density becomes higher in comparison to that on longer routes. In this sense, if an ant finds a good (short) path from the colony to a food source, others members are more likely to follow the route traced by said ant. The positive feedback provided by the increase in pheromone density through paths traversed by an increasing number of ants eventually lead all members of the colony to follow a single optimal route (Dorigo and Stützle 2004).

The first ACO approach was conceived as an iterative process devised to handle the task of finding optimal paths in a graph (Dorigo and Blum 2005). For this purpose, ACO considers a population of N ants which move through the nodes and arcs of a graph $G(\mathcal{N}, \mathcal{P})$ (with $\mathcal{N}$ and $\mathcal{P}$ denoting its respective sets of nodes and arcs, respectively). Depending on their current state (node), each ant is able to choose from among a set of adjacent paths (arc) to traverse based on the pheromone density and length associated to each of them. With that being said, at each iteration 'k', the probability for a given ant 'i' to follow a specific path 'xy' (which connects states 'x' and 'y') is given by the following expression:

$$p_{i_{(xy)}}^{k} = \frac{\left(\alpha \cdot \tau_{(xy)}^{k}\right)\left(\beta \cdot \eta_{(xy)}^{k}\right)}{\sum_{z \in \mathbf{Y}_x}\left(\alpha \cdot \tau_{(xz)}^{k}\right)\left(\beta \cdot \eta_{(xz)}^{k}\right)} \tag{13}$$

where $\tau_{(xy)}^{k}$ denotes the pheromone density over the given path 'xy', while $\eta_{(xy)}^{k}$ stand for the preference for traversing said path, which is relative to its distance (cost). Furthermore, $\mathbf{Y}_x$ represent the set of all adjacent states for the given current state 'x'. Finally, α and β are constant parameters used to control the influence of $\tau_{(xy)}^{k}$ and $\eta_{(xy)}^{k}$, respectively.

By applying this mechanism, each ant moves through several paths within the graph until a specific criterion is met (i.e., that a particular destination node has been reached). Once this happens, each ant backtracks its traversed route while releasing some pheromones on each the paths they used. In ACO, the amount of pheromones released by an ant 'i' over any given path 'xy' is given by:

$$\Delta\tau_{i_{(xy)}}^{k} = \begin{cases} Q/L_i & \text{if "the ant used the path } xy \text{ in its tour"} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where L_i denotes the length (cost) associated to the route taken by the ant 'i', while Q stand for a constant value.

Finally, ACO includes a procedure used to update the pheromone density over all paths in the graph for the following iteration ($k + 1$). For this purpose, it considers both, the amount of pheromones released by each ant while backtracking its traced route and the natural dissipation of pheromones which takes place as time passes. This is applied by considering the following expression:

$$\tau_{(xy)}^{k+1} = (1-\rho) \cdot \tau_{(xy)}^{k} + \sum_{i=1}^{N} \Delta\tau_{i_{(xy)}}^{k} \tag{15}$$

where ρ is a constant value known as pheromone evaporation coefficient, while $\Delta\tau_{i_{(xy)}}^{k}$ stand for the amount of pheromones released by an ant 'i' over a specific path 'xy' (as given by Eq. 14).

3.2.2 Artificial Bee Colony

Bees are among the most well-known example of insects which manifest a collective behavior, either for food foraging or mating. Based on this premise, ma researchers have proposed several different swarm intelligence approaches inspired by the behavior of bees. In particular, the Artificial Bee Colony (ABC) approach proposed by Karaboga and Basturk (2008) is known to be among the most popular of these bee-inspired methods.

In the ABC approach, search agents are represented as a colony of artificial honey bees which explore a d-dimentional search space while looking for opmal food (nectar) sources. The locations of these food sources each represent a possible solutions for a given optimization problem and their amount of nectar (quality) is related to the fitness value associated to each of such solutions. Furthermore, the members of the bee colony are divided in three groups: employed bees, onlooker bees and scout bees. Each of these groups of bees has distinctive functions inspired in the mechanics employed by bees while foraging for food. For example, the employed bees comprises the members of the colony which function is to explore the surroundings of individually-known food sources in the hopes of finding places with greater amounts of nectar. In addition, employed bees are able to share the information of currently known food sources with the rest of the members of the colony, so that they can also exploit them. With that being said, at each iteration 'k' of ABC's search process, each employed bee generates a new candidate solution $\mathbf{v}_i$ around a currently known food source $\mathbf{x}_i$ as follow

$$\mathbf{v}_i^k = \mathbf{x}_i^k + \phi\left(\mathbf{x}_i^k - \mathbf{x}_r^k\right) \tag{16}$$

where $\mathbf{x}_i^k$ denotes the location of the food source remembered by a particular employed bee 'i' while $\mathbf{x}_r^k$ (with $r \neq i$) stands for the location of any other randomly chosen food source. Furthermore, ϕ is a random number drawn from within the interval $[-1, 1]$.

On the other hand, onlooker bees can randomly visit any food source known by the employed bees. For this purpose, each available food source is assigned with a certain probability of being visited by an onlooker bee as follows:

$$P_i^k = \frac{f(\mathbf{x}_i^k)}{\sum_{j=1}^{N} f(\mathbf{x}_j^k)} \tag{17}$$

Similarly to the employed bees, once an onlooker bee has decided to visit a particular food source, a new candidate solution $\mathbf{v}_i^k$ is generated around the chosen location $\mathbf{x}_i^k$ by applying Eq. (16). Furthermore, any candidate solution $\mathbf{v}_i^k$ generated by either an employed or an onlooker bee is compared against its originating location $\mathbf{x}_i^k$ in terms of solution quality, and then, the best among them is chosen as the new food source location for the following iteration; this is:

$$\mathbf{x}_i^{k+1} = \begin{cases} \mathbf{v}_i^k & \text{if } \left(f\left(\mathbf{x}_i^k\right) < f\left(\mathbf{v}_i^k\right)\right) \\ \mathbf{x}_i^k & \text{otherwise} \end{cases} \tag{18}$$

Finally, scout bees are the members of the colony whose function is to explore the whole terrain for new food sources randomly. Scout bees are deployed to look for new solutions only if a currently known food source is chosen to be “abandoned” (a thus forgotten by all members of the colony). In ABC, a solution is considered to be abandoned only if it cannot be improved by either the employed or onlooker bees after a determined number of iterations, indicated by the algorithm’s parameter “$limit$”. This mechanism is important for the ABC approach since it allows it to keep the diversity of solutions during the search process.

In general, ABC’s local search performance may be attributed to the neighborhood exploration and greedy selection mechanisms applied by the employed and onlooker bees, while the global search performance is mainly related to the diversification attributes of scout bees.

3.2.3 Bat Algorithm

The Bat Algorithm (BA) is a bio-inspired metaheuristic proposed by Yang (2010b). The BA approach draws inspiration on the behavior manifested by certain species of bats (particularly, microbats). In nature, most bats are equipped with a type of biologic sonar known as echolocation. In simple terms, the echolocation consists of two steps: the emission of loud frequency-modulated sound pulses and the reception (listening) of the echoing sounds that bounce back from surrounding objects, which essentially allows building a three-dimensional scenario of the immediate environment. Typically, bats use this specialized sonar system to assist themselves in several tasks, such as prey detection, obstacle evasion or even locating their roosting places.

In the BA approach, search agents are modeled as a swarm of bats whose position within the d-dimensional search space represent a possible solution for a given optimization problem. Furthermore, each bat is assumed to use echolocation to assist its movement toward a particular prey (modeled as the global best solution). For this purpose, the BA algorithm considers three sets of parameters whose values are constantly adjusted as the search process goes on: *frequencies*, *loudness* and *pulse emission rates*. In the case of the frequencies, these are modeled by a set of d-dimensional vectors, each associated to a given bat ‘i’ and whose values are randomly adjusted at each iteration ‘k’, such that:

$$\mathcal{F}_i^k = \mathcal{F}_{\text{min}} + \beta \cdot (\mathcal{F}_{\text{max}} - \mathcal{F}_{\text{min}}) \tag{19}$$

where the parameters $\mathcal{F}_{\text{min}}$ and $\mathcal{F}_{\text{max}}$ denote the minimum and maximum frequencies, respectively. Finally, β is a vector of random numbers each from within the interval [0, 1].

On the other hand, the loudness and pulse emission rates A_i. and r_i, respectively, comprise a set of parameters whose initial values A_i^0 and r_i^0 are defined during the algorithms initialization. As the search process evolves, the values of these parameters are modified according to the following expression:

$$A_i^{k+1} = \alpha A_i^k, r_i^{k+1} = r_i^0(1 - \exp(-\gamma k)) \tag{20}$$

where $\alpha < 1$ and $\gamma < 1$ are constant parameters.

With regard to the position update operators, the BA algorithm applies the following movement rule to update the location of each bat 'i' for the currentteration 'k':

$$\mathbf{x}_i^k = \mathbf{x}_i^{k-1} + \mathbf{v}_i^k \tag{21}$$

where $\mathbf{x}_i^{k-1}$ represents the position of the ith bat at the previous iterations $(k-1)$, while $\mathbf{v}_i^k$ stands for the velocity of said bat, as given by the following expression:

$$\mathbf{v}_i^k = \mathbf{v}_i^{k-1} + f_i \cdot \left(\mathbf{x}_i^k - \mathbf{x}_{\text{best}}\right) \tag{22}$$

where $\mathbf{x}_{\text{best}}$ denotes for the current global best solution found during the search process, while $\mathcal{F}_i^k$ represents frequency vector associated to bat 'i', as given by Eq. 19.

Furthermore, BA also includes a local search scheme in which, at each iteration, a randomly chosen individual among the current best solutions is further refined by performing a random walk as follows:

$$\mathbf{x}_*^k = \begin{cases} \mathbf{x}_i^k + \varepsilon A_i^k & \text{if } \left(\text{rand} > r_i^k\right) \\ \mathbf{x}_i^k & \text{otherwise} \end{cases} \tag{23}$$

where the parameters A_i^k and r_i^k denote the loudness and pulse emission rates associated to the randomly chosen bat 'i', while rand. stand for a random number drawn from within the uniformly distributed interval [0, 1]. Finally, it is worth noting that the newly generated solution $\mathbf{x}_*^k$ is accepted as the new position for bat 'i' ($\mathbf{x}_i^k$) only if certain conditio met; particularly:

$$\mathbf{x}_*^k = \begin{cases} \mathbf{x}_*^k & \text{if } \left(\text{rand} < A_i^k \text{ and } f\left(\mathbf{x}_i^k\right) < f\left(\mathbf{x}_*^k\right)\right) \\ \mathbf{x}_i^k & \text{otherwise} \end{cases} \tag{24}$$

3.2.4 Crow search algorithm

The Crow Search Algorithm (CSA), as proposed by Askarzadeh (2016), is a nature-inspired optimization method which draws inspiration in the behavior of flocking crows. Crows are considered by many as the most intelligent of birds; they are mainly known for hiding their excess food in specific locations around their environment, and when needed they can accurately remember the location of their hidden food sources. Crows are also known for its tendency to commit thievery. To do so, they observe and follow other crows to find their hiding places, and then, they steal their resources once its owner leaves. Also, from their experience as thieves, crows are able to develop different tactics to prevent their hiding places from being pilfered by other crows, such as moving their hiding places to other locations or even tricking other birds to follow them to a different place.

In CSA, search agents are modeled as a flock comprised by N crows, each with a particular positions $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}]$ within a feasible d-dimentional solution space. Each of these crows is also assumed to have a memory, which allows them to remember the location of their respective food hiding places. For this purpose, CSA assigns each crow 'i' with a memory solution $\mathbf{m}_i = [m_{i,1}, m_{i,2}, \ldots, m_{i,d}]$, which represents the best solution found so far by said crow 'i'. The movement operators employed by CSA to update the position of each crow considers two different possibilities: (1) the case where a crow 'i' follows some randomly chosen crow 'j' to their hiding place, and (2) the situation in which said crow 'i'

is deceived by crow 'j' into moving to a different location. With that being said, at each iteration 'k', CSA updates the position of each crow as follows:

$$\mathbf{x}_i^{k+1} = \begin{cases} \mathbf{x}_i^k + r_i \cdot fl_i^k \cdot \left(\mathbf{m}_j^k - \mathbf{x}_i^k\right) & \text{if } \left(\text{rand} \geq AP_j^k\right) \\ \mathbf{r}_i & \text{if } \left(\text{rand} < AP_j^k\right) \end{cases} \tag{25}$$

where AP_j^k denotes awareness probability of crow 'j' at iteration 'k', whereas rand stand for a random number drawn from a uniform distribution within the interval [0, 1]. The position update operator applied when rand(0, 1) $\geq AP_j^k$ denotes the situation in which a crow 'j' is not aware that a crow 'i' is following it and, as a result, crow 'i' approaches toward the hiding place of sais crow 'j' ($\mathbf{m}_j^k$). In this sense, the parameter fl_i^k denotes the maximum flight length (step size) of crow 'i', while $r_i \in [0, 1]$ stand for a random step factor. On the other hand, if rand $< AP_j^k$, it is assumed that crow 'j' is aware that it is being followed by the crow 'i', and in response, it will fool said pilferer into moving to a different location. With that being said, the position of crow 'i' is updated to $\mathbf{r}_i$, which stand for a randomly generated solution within the feasible decision space.

3.2.5 Cuckoo search

Cuckoo birds are well-known due to they're aggressive, yet fascinating reproduction strategies. In particular, a good number of cuckoo species are known to resort to brood parasitism as part of their life-cycle. Essentially, cuckoos secretly lay their eggs in the nest of other birds (typically of a different species) in the hopes of deceiving the host into thinking that such eggs are their own. Should these alien eggs succeed to be undetected by the host bird, they are almost guaranteed to hatch into new cuckoo chicks; otherwise, the host bird may opt to remove such aliens eggs of their nest or even abandon the nest to build a new one somewhere else. Inspired by this intriguing behavior, in Yang and Deb (2009) proposed the nature-inspired algorithm known as Cuckoo Search (CS).

In the CS approach, solutions are modeled as a set of N host nests. To simulate the situation in which a cuckoo bird chooses a host nest to lay their own eggs, at each iteration 'k', new candidate solutions are generated around randomly selected host nests 'i' by applying a random walk as follows:

$$\mathbf{x}_{\text{new}}^k = \mathbf{x}_i^k + \alpha \cdot \text{Lévy}(\lambda) \tag{26}$$

where Lévy(λ) denotes a random walk via Leví flights (Yang 2008), while $\alpha > 0$ stand for a vector of step sizes related to the scale of the objective solution space.

Once a new candidate solution has been generated, its quality (fitness) is compared to that of the solution represented by the randomly selected host nest. If the quality of a candidate solution $\mathbf{x}_{\text{new}}^k$ is better than $\mathbf{x}_i^k$ related to the host nest 'i', then $\mathbf{x}_{\text{new}}^k$ replaces $\mathbf{x}_i^k$ for the next iteration. This is:

$$\mathbf{x}_i^{k+1} = \mathbf{x}_{\text{new}}^k \tag{27}$$

Furthermore, after new solutions have been generated and evaluated for each randomly selected host nest, a fraction p_a of the worst remaining solutions are eliminated, and then, replaced by an equal amount of randomly generated solutions. This mechanism is intended to simulate the nest abandonment behavior manifested by host birds once they discover the presence of cuckoo eggs within their nest, which further promotes them to build a new nest on a different place.

An important trait about the CS approach is that the definition of host nest can further be modified so that each of them represents a set of solutions (multiple eggs within each nest) instead of a single one which could effectively allow CS to be extended into a type meta-population algorithm, and thus, be applied to solve even more complex optimization problems.

3.2.6 Firefly Algorithm

The Firefly Algorithm (FA), as proposed by Yang (2010a, b), is a swarm intelligence approach inspired by the light-emitting behavior observed in fireflies. In the FA approach, search agents, modeled as a swarm of fireflies, are assumed to be able to interact with each other through its characteristic bioluminescent glowing. In particular, such interactions are modeled as attractions toward other conspecific individuals within the target solution space. The attractiveness of each firefly is considered to be proportional to its light intensity, which in turn is also said to be equivalent to their quality (fitness). Furthermore, it is also assumed that fireflies are only attracted toward brighter individuals; this is, that for any two flashing fireflies, the less bright one will be attracted toward the brighter one (regardless of their gender). Also, it is also assumed that the attractiveness "perceived" by a particular firefly 'i' toward any other individual source 'j' decreases as the distance that separates them increases. Such pha enomenon is simplified by the following expression:

$$\beta_{ij} = \begin{cases} \beta_{0j} \cdot e^{-\gamma r_{ij}^2} & \text{si } f(\mathbf{x}_j) > f(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

where $r_{ij} = \mathbf{x}_j - \mathbf{x}_i$ stand for the Euclidian distance computed with regard to the pair of fireflies 'i' and 'j', while β_{0j} denotes the *attractiveness* of individual 'j'. Furthermore, the parameter γ stand for the so-called *light absorption coefficient* which is used to further vary the overall attractiveness toward the individual 'j'.

Finally, the attraction movement performed by a firefly 'i' toward any given individual 'j' may be given by the following expression:

$$\Delta\mathbf{x}_{ij} = \mathbf{x}_i + \beta_{ij} \cdot (\mathbf{x}_j - \mathbf{x}_i) + \alpha \cdot \boldsymbol{\epsilon}_i \tag{29}$$

where $\boldsymbol{\epsilon}_i$ denotes a d-dimensional vector denoting a random movement, while α stands for a randomization parameter, which values are typically within the interval [0, 1].

As noted by its author, the attraction behaviors represented in FA enables search agents to explore the solution space of a given optimization problem more effectively. Another interesting property of FA is that it can effectively locate both global and local optima, which could be an important advantage on certain implementations.

3.2.7 Flower Pollination Algorithm

The Flower Pollination Algorithm (FPA) is another popular optimization method in Yang's showcase of nature-inspired metaheuristics. First proposed in 2012, the FPA approach is well-known for drawing inspiration in the intriguing pollination process observed in most species of flowers (Yang 2012).

In the FPA approach, solutions are modeled as individual flowers (or pollen gametes). Furthermore, FPA considers two natural pollination methods in order to establish a set of valid movement rules: cross-pollination, which refers to the flower reproduction process in which pollen is carried over long distances by pollinators (such as insects, birds, or other animals) as a way to ensure the reproduction of fittest plants, and self-pollination, which emphasize the fertilization process that occurs among flowers in the absence of viable pollinators. In the context of swarm intelligence, FPA considers cross-pollination as a global search process, while self-pollination is seen as a local search approach. Furthermore, it also assumes that at each iteration 'k' each particular flower is only able to produce a single pollen gamete per iteration. By considering these idealized characteristics, the FPA's position update operators may be illustrated as follows:

$$\mathbf{x}_i^{k+1} = \begin{cases} \mathbf{x}_i^k + L\left(\mathbf{x}_* - \mathbf{x}_i^k\right) & \text{if} \quad (\text{rand} > P) \\ \mathbf{x}_i^k + \epsilon(\mathbf{x}_j^k - \mathbf{x}_q^k) & \text{if} \quad (\text{rand} \leq P) \end{cases} \tag{30}$$

where rand denotes a random number from within the interval [0, 1].

The position update rule applied when rand $> P$ (with P denoting a probability threshold) stand for a global pollination movement rule. In this case $\mathbf{x}_*$ denotes the current global best solution, while L represents a scaling parameter known as pollination strength, which value is given from a Levy distribution, as given by:

$$L \sim \frac{\lambda \Gamma(\lambda) \cdot \sin(\pi \lambda / 2)}{\pi \left(s^{1+\lambda}\right)}, \quad (s \gg s_0 > 0) \tag{31}$$

where $\Gamma(\lambda)$ stand for the standard gamma function (with regard to the constant parameter λ), while s stand for a user-defined step size.

On the other hand, the movement operator corresponding to the case when rand $\leq p$ represents a local pollination movement rule, which can be essentially considered as a random walk. In such a case, $\mathbf{x}_j^k$ and $\mathbf{x}_q^k$ (with $j \neq q$) represent the positions of two randomly chosen flowers (different to $\mathbf{x}_i^k$), while ϵ is a random value drawn from within the interval [0, 1].

3.2.8 Grey Wolf Optimizer

The Grey Wolf Optimizer (GWO) is a nature-inspired metaheuristic proposed by Mirjalili et al. (2014). As its name may imply, GWO's design is inspired by the distinctive hunting behaviors and social hierarchy observed in packs of grey wolves. Grey wolves are known to be organized in a very strict social dominant hierarchy composed of alpha, beta delta, and omega wolves. The alpha wolves (typically comprised by a male and a female member) are the leaders of the pack and are responsible for all decisions related to hunting, resting and moving. The beta wolves, which comprise the second level on the pack's hierarchy, are subordinate to the alpha wolves but can give commands to lower-level wolves. Similarly, delta wolves are subordinate to both alpha and beta members, yet they dominate over the all other wolves. Finally, the omega wolves, which are the lowest-ranked members, play as subordinates to the dominant wolves (alpha, beta, and delta), and as such follow their instructions (especially while hunting). Another important trait observed in grey wolves is related to their cooperative hunting tactics in which wolves start to chase an identified prey until it is encircled, and then they proceeded to attack the prey until it is finally killed. These distinctive traits are mathematically modeled in GWO for the task of solving global optimization problems (Mirjalili et al. 2014).

Analogous to the social hierarchy observed in real grey wolves, GWO identifies each search agent as either alpha, beta, delta or omega wolf depending on their current fitness; in particular, the individual $\mathbf{x}_i$ which represent the fittest solution is considered as the alpha wolf (α). Similarly, the second and third best solutions are designated as beta (β) and delta (δ) respectively, while all remaining wolves are labeled as omega members (ω). Also, WOA considers a hierarchy-based search scheme in which lower-ranked wolves move based on information shared by higher-ranked individuals. As such, at each iteration 'k', each wolf updates their position for the following iteration '$k + 1$' as follows:

$$\mathbf{x}_i^{k+1} = \frac{\boldsymbol{\alpha}_i^k + \boldsymbol{\beta}_i^k + \boldsymbol{\delta}_i^k}{3} \tag{32}$$

where $\boldsymbol{\alpha}_i^k$, $\boldsymbol{\beta}_i^k$ and $\boldsymbol{\delta}_i^k$ each denote the position update applied to wolf 'i' ($\mathbf{x}_i^k$) with regard to the positions occupied by the α, β and δ wolves, respectively, and are given by:

$$\boldsymbol{\alpha}_i^k = \mathbf{x}_\alpha^k - \mathbf{A}^k \cdot \left|\mathbf{C}^k \cdot \mathbf{x}_\alpha^k - \mathbf{x}_i^k\right| \tag{33}$$

$$\boldsymbol{\beta}_i^k = \mathbf{x}_\beta^k - \mathbf{A}^k \cdot \left|\mathbf{C}^k \cdot \mathbf{x}_\beta^k - \mathbf{x}_i^k\right| \tag{34}$$

$$\boldsymbol{\delta}_i^k = \mathbf{x}_\delta^k - \mathbf{A}^k \cdot \left|\mathbf{C}^k \cdot \mathbf{x}_\delta^k - \mathbf{x}_i^k\right| \tag{35}$$

where $\mathbf{x}_\alpha^k$, $\mathbf{x}_\beta^k$ and $\mathbf{x}_\delta^k$ denote the current positions of the α, β and δ wolves, respectively, and where:

$$\mathbf{A}^k = 2 \cdot \mathbf{a}^k \cdot \mathbf{r}_1^k - \mathbf{a}^k \tag{36}$$

$$\mathbf{C}^k = 2 \cdot \mathbf{r}_2^k \tag{37}$$

where $\mathbf{a}^k$ denotes a coefficient vector whose values linearly decreases from 2 to 0 over the course of iterations, while $\mathbf{r}_1^k$ and $\mathbf{r}_2^k$ represent random vector whose values are given by the uniformly distributed interval [0, 1].

3.2.9 Krill Herd Algorithm

The optimization techniques known as Krill Herd Algorithm (KHA) was devised by Gandomi and Alavi (2012). As its name implies, the KHA method is inspired by the natural herding behavior commonly observed on small crustaceans known as krill.

In KHA, solutions are modeled as a group of krill individuals which move through a feasible solution space while herding and foraging for food. Furthermore, KHA considers three distinctive kinds of herding movements: (1) The motion induced by other krill individuals, which represents a tendency to move toward other members of the aggregation aimed to keep a high density of individuals; (2) A foraging movement, which purpose is to guide krill toward the estimated location of a given food source; and (3) A physical diffusion movement, which represents a random process mainly used to keep the diversity of solutions. With that being said, at each time-step 't', the position of each individual krill is updated as follows:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \Delta t \frac{d\mathbf{x}_i^t}{dt} \tag{38}$$

where $\frac{d\mathbf{x}_i^t}{dt}$ denote a speed vector corresponding to the ith krill individual. In the KHA approach, such a speed vector is represented by the following Lagrangian model:

$$\frac{d\mathbf{x}_i^t}{dt} = N_i + F_i + D_i \tag{39}$$

with N_i, F_i and D_i denoting each of the three distinctive krill herding movement, namely: 1. The motion induced by other krill individuals (N_i), 2. Foraging movement (F_i), and 3.

Physical diffusion movement (D_i) (Gandomi and Alavi 2012). Furthermore, the value Δt stand for a speed scale factor, as given by the following expression:

$$\Delta t = C^t \sum_{n=1}^{d} (ub_n - lb_n) \tag{40}$$

where lb_n and ub_n each denote the lower and upper bounds of the nth variable (dimension), respectively, whereas $C^t \in [0, 2]$ is a constant parameter used to control the intensity of the search process.

Another important trait regarding the implementation of the KHA approach is the incorporation of genetic operators (such as crossover and mutation) which, according to the authors, aids to further improve the algorithms overall performance (Gandomi and Alavi 2012).

3.2.10 Moth-Flame Optimization Algorithm

The Moth-Flame Optimization Algorithm (MFO) is novel nature-inspired metaheuristic proposed by Mirjalili (2015). The MFO approach is inspired by the night-navigation mechanism employed by moths in nature. Such a mechanism, known as transversal orientation, consist of moths flying by maintaining a fixed angle to a particular light source. While for distant light sources (such as the moon) this method allow moths to fly in a straight line for very long distances effectively, nearby artificial lights (such as light bulbs or even the flame of a candle) are easily able to hamper such navigation method, causing moths to fly in a spiraling pattern toward such light sources. Such distinctive behaviors are mathematically modeled in MFO to perform global optimization tasks.

In MFO, search agents are represented by a population of N_M moths, each with a particular positions $\mathbf{M}_i = [m_{i,1}, m_{i,2}, \ldots, m_{i,d}]$ within a given solution space. Furthermore, the MFO approach also considers a set of N_F flames (or artificial lights) randomly distributed around such solution space, so that each of them also has a particular position $\mathbf{F}_j = [f_{j,1}, f_{j,2}, \ldots, f_{j,d}]$. Akin to how moths are "attracted" toward nearby light sources, each moth 'i' is assumed to fly in a spiraling pattern toward a given flame 'j'. In this sense, while the positions of both moths and flames represent solutions, only the moths are actual search agents while the flames stand for the best N_F solutions found so far by MFO search process. By considering this, at each iteration 'k', each moth is first assigned to a particular flame, and then, a movement operator modeled after a logarithmic spiral is applied in order to update the position of each search agent as follows:

$$\mathbf{M}_i^{k+1} = \mathbf{D}_{ij}^k \cdot e^{bl} \cdot \cos(2\pi l) + \mathbf{F}_j^k \tag{41}$$

where $\mathbf{D}_{ij}^k = \left|\mathbf{F}_j^k - \mathbf{M}_i^k\right|$. Furthermore, b denotes a constant parameter, while l stand for a random number drawn from within the interval $[r, 1]$ (with r being linearly decreased from -1 to -2 over the course of iterations).

Also, MFO employs a mechanism in which the number of available flames N_F within the search space is reduced as the iterative process goes on. With that being said, at each iteration 'k', the number of available flames is updated by considering the following expression.

$$N_F^{k+1} = \text{round}\left(N_F^0 - k \cdot \frac{N_F^0 - 1}{K} \right) \tag{42}$$

where N_F^0 denotes the initial (maximum) number of flames, while K stand for the maximum number of iterations which comprise the whole search process.

Initially, since moths move around by considering the positions of N_F best solutions (flames), exploration over the search space is highly promoted, while exploitation is minimal. However, as the number of available flames is reduced, exploration intensity is slowly decreased, while exploitation is gradually favored, thus, balancing the exploration and exploitation of solutions.

3.2.11 Particle Swarm Optimization

Devised by Kennedy and Eberhart (1995), the Particle Swarm Optimization (PSO) method draws inspiration in the behavior of flocking birds, collectively foraging for adequate food sources. In PSO, search agents (also referred as *particles*) are each composed by a set of three d-dimensional vectors: the particle's current position, its previous best position and its velocity. Also, each member within the swarm of particles is assumed to have knowledge of the global best position reached by its immediate neighborhood during the search process. With that being said, in the traditional PSO, the position update for each particle 'i' is given by the following expressions:

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1} \tag{43}$$

where k denotes the current iteration. Furthermore, $\mathbf{v}_i^{k+1}$ stand for the velocity of particle 'i' at iteration '$k + 1$', and is given as follows:

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + c_1 \cdot \left(\mathbf{r}_1^k \cdot \left(\mathbf{p}_i^k - \mathbf{x}_i^k\right)\right) + c_2 \cdot \left(\mathbf{r}_2^k \cdot \left(\mathbf{g}_i^k - \mathbf{x}_i^k\right)\right) \tag{44}$$

where $\mathbf{p}_i^k$ denotes the previous best position of particle 'i' (also called the personal best of 'i'), while $\mathbf{g}_i^k$ stand for the current global best position within an specific neighborhood of particles, from which individual 'i' belongs; here, the word "neighborhood" makes reference to a specific subset (topology) of particles (although, in its most simple form, it may refer to the whole swarm of particles) (Poli et al. 2007). Furthermore, $\mathbf{r}_1^k$ and $\mathbf{r}_2^k$ each denote a d-dimensional vector composed by random numbers drawn from the interval of [0, 1], while the values c_1 and c_2 are known as cognitive and social parameters, respectively.

Due to its simplicity and easy implementation, the PSO method (as well as its many variations) has been extensively studied and applied into a wide variety of engineering areas and, as a result, has become one the most popular swarm intelligence approaches currently available for solving complex optimization problems.

3.2.12 Social Spider Optimization

The Social Spider Optimization (SSO) is a swarm intelligence approach proposed by Cuevas et al. (2013a). The SSO approach draws inspiration into several collective behaviors observed within a colony of social spiders. A social spider colony is mainly composed by two components: its members, which may be further distinguished by their gender (either male or female) and a communal web which serves as a medium for both interaction and communication among such members of the colony. One important characteristic of most social spider colonies is the predominance of female spiders within its population. Furthermore, in a social spider colony, each member, depending on their gender, is assigned to cooperate in several different activities, such as building and maintaining the communal web, capturing prey, mating, etc. Another important trait about social spiders lies in their capacity to perceive vibrations transmitted through the communal web. Social spiders employ these vibrations to gain specific information, such as the size of trapped preys or the characteristics of neighboring members.

In the SSO approach, candidate solutions are modeled as a group of N spiders (each with a corresponding position $\mathbf{s}_i = [s_1, s_2, \ldots, s_d]$), interacting within a d-dimensional solution space (properly referred as the communal web). Furthermore, spiders are assumed to be able to communicate through a series of vibrations, emitted by each member and transmitted though the threads which conform the communal web. In this sense, the stimulus perceived by a particular spider 'i' as a result of the vibrations transmitted by some other spider 'j' are modeled as follows:

$$\mathrm{Vib}_{i,j} = w_{\mathbf{s}_j} \cdot e^{-r_{ij}^2} \tag{45}$$

where $r_{ij} = \mathbf{s}_i - \mathbf{s}_j$ denotes for the Euclidian distance between the spiders 'i' and 'j'. Furthermore, $w_{\mathbf{s}_j}$ represents the weight corresponding to the jth spider as given by:

$$w_{\mathbf{s}_j} = \frac{f(\mathbf{s}_j) - f_{\text{worst}}}{f_{\text{best}} - f_{\text{worst}}} \tag{46}$$

where f_{best} and f_{worst} each denote the current best and worst fitness values from among all spiders within the communal web.

Also, each spider within the communal web is designated with a specific gender (either male or female). Depending on their gender, spiders are able to manifest several different behaviors. In the case of female spiders, for example, an attraction or dislike toward other members of the colony is displayed (irrespective of their gender), which depend on several factors such as reproduction cycle, curiosity and other random phenomena. In SSO, such behavior is modeled as either an attraction or repulsion movement toward other prominent individuals within the communal web. With that being said, at each iteration 'k', the position of a given female spider $\mathbf{f}_i^k$ (with $\mathbf{f}_i^k = \mathbf{s}_j^k$, $j \in [1, 2, \ldots, N]$) is updated by applying the following movement rules:

$$\mathbf{f}_i^{k+1} = \begin{cases} \mathbf{f}_i^k + \alpha \cdot \mathrm{Vib}_{c_i}\left(\mathbf{s}_{c_i}^k - \mathbf{f}_i^k\right) + \beta \cdot \mathrm{Vib}_{b_i}\left(\mathbf{s}_b^k - \mathbf{f}_i^k\right) + \delta \cdot \left(\gamma - \frac{1}{2}\right) & \text{if } P > P_f \\ \mathbf{f}_i^k - \alpha \cdot \mathrm{Vib}_{c_i}\left(\mathbf{s}_{c_i}^k - \mathbf{f}_i^k\right) - \beta \cdot \mathrm{Vib}_{b_i}\left(\mathbf{s}_b^k - \mathbf{f}_i^k\right) + \delta \cdot \left(\gamma - \frac{1}{2}\right) & \text{if } P \leq P_f \end{cases} \tag{47}$$

where $\mathbf{s}_{c_i}^k$ denotes the position of the nearest best (nearest heavier) member to the spider 'i', while $\mathbf{s}_b^k$ stand for the position of the best (heaviest) spider within the communal web. Furthermore, Vib_{c_i} and Vib_{b_i} denote the vibrations perceived by the spider 'i' with regard to $\mathbf{s}_{c_i}^k$ and $\mathbf{s}_b^k$, respectively (see Eq. 45), while α, β, δ, γ and P each denote a random number within the interval [0, 1]. Finally, P_f stands for a probability threshold used to define the kind of movement the female spider will perform (either an attraction or a repulsion).

On the other hand, male spiders, which are said to manifest an exclusive attraction toward female members, are further classified as either dominant or non-dominant male spiders. Typically, dominant male spiders have better qualities (i.e., a greater size or weight) in comparison to non-dominant male spiders. Furthermore, while dominant male spiders are usually attracted toward their closest female spider within the communal web, non-dominant male spiders tend to concentrate toward the center of the male population as a strategy to take advantage of resources that are wasted by dominant males. In SSO, these male-characteristic behaviors are modeled by first considering the median weight of all male spiders. At each iteration 'k', the weight of each male spider is compared to such median value in order to classify them as either dominant (weight above the median) or non-dominant (weight equal or lower to the median) male spiders, an then, an appropriate movement rule is applied to update the position $\mathbf{m}_i^k$ (with $\mathbf{m}_i^k = \mathbf{s}_j^k$, $j \in [1, 2, \ldots, N]$) of each of such male spiders, as illustrated as follows:

$$\mathbf{m}_i^{k+1} = \begin{cases} \mathbf{m}_i^k + \alpha \cdot \mathrm{Vib}_{f_i}\left(\mathbf{s}_{f_i}^k - \mathbf{m}_i^k\right) + \delta \cdot \left(\gamma - \frac{1}{2}\right) & \text{if } w_{\mathbf{m}_i}^k > M(\mathbf{w_m}) \\ \mathbf{m}_i^k + \alpha \cdot \left(\frac{\sum_{h=1}^{N_m} \mathbf{m}_h^k \cdot w_{\mathbf{m}_h}^k}{\sum_{h=1}^{N_m} w_{\mathbf{m}_h}^k} - \mathbf{m}_i^k\right) & \text{if } w_{\mathbf{m}_i}^k > M(\mathbf{w_m}) \end{cases} \tag{48}$$

where $w_{\mathbf{m}_i}^k$ denotes the weight of the male spider 'i', whereas $M(\mathbf{w_m})$ denotes the median value with regard to the weights of all male spiders. Furthermore, $\mathbf{s}_{f_i}^k$ stand for the position of the nearest female spider to $\mathbf{m}_i^k$, while Vib_{f_i} stand for the stimulus perceived by the male spider 'i' as a result of the vibrations emitted by its nearest female member (see Eq. 45). Also, the values α, δ, and γ each denote a random number within the interval [0, 1].

Finally, the SSO approach employs a mating mechanism in which female spiders and dominant male spiders are used to construct new candidate solutions. For such a procedure, a dominant male spider is first selected, and then, a set female spiders within a particular "range of mating" (which depends on the size of the target solution space) are further selected to perform a mating operation, in which a new individual is formed by combining the position information of each involved member. In this case, individuals possessing heavier weights (or higher fitness values) are more likely to influence the newly produced solution while those with lower weights tend to be of less relevance Cuevas et al. (2013a).

3.2.13 Whale Optimization Algorithm

In Mirjalili and Lewis (2016) proposed a novel bio-inspired metaheuristic called Whale Optimization Algorithm (WOA). The WOA approach draws inspiration on the predatory behavior observed in humpback whales which, different to other species of whales, distinguish themselves for employing a unique cooperative hunting maneuver known as bubble-net feeding. In such a hunting method, a group of whales (typically formed by two or three individuals) coordinate their efforts to encircle a group of small prey (such as schools of krill or small fishes) by swimming in a spiraling fashion around their quarry. While moving beneath the surface in such a distinctive pattern, each whale starts to exhale a burst of bubbles from their blowhole to form encircling bubble barrier (also referred as bubble-net), which prevents prey from escaping. As prey gets corralled into a tighter circle, one whale sounds a feeding call to the other whales, at which point all individuals simultaneously swim to the surface with their mouths open to feed on the trapped prey.

In WOA, such unique bubble-net feeding behavior is mathematically modeled and applied to solve global optimization problems by first considering a group of search agents, represented by N whales. Moreover, WOA's search process is divided in two phases: exploration and exploitation phase. For the exploration phases, whales are assumed to be randomly searching for prey. This process is modeled as movements toward a randomly chosen member. With that being said, at each iteration 'k', the position of each whale is updated by applying the following equation:

$$\mathbf{x}_i^{k+1} = \mathbf{x}_{\text{rand}}^k - \mathbf{A}^k \cdot \mathbf{D}^k \tag{49}$$

where $\mathbf{x}_{\text{rand}}^k$ denotes the position of a randomly chosen whale and where:

$$\mathbf{A}^k = 2 \cdot \mathbf{a}^k \cdot \mathbf{r}_1^k - \mathbf{a}^k \tag{50}$$

$$\mathbf{D}^k = \left|\left(2 \cdot \mathbf{r}_2^k \cdot \mathbf{x}_{\text{rand}}^k\right) - \mathbf{x}_i^k\right| \tag{51}$$

where $\mathbf{a}^k$ is a coefficient vector whose values linearly decreases from 2 to 0 over the course of iterations, while $\mathbf{r}_1^k$ and $\mathbf{r}_2^k$ denote random vectors whose values are drawn from within the uniformly distributed interval [0, 1].

On the other hand, the exploitation phase emphasizes the situation in which the group of whales has already identified their prey. To represent such behavior, the WOA approach considers two distinctive movement rules: prey encircling and bubble-net attacking method. For the prey encircling behavior, whales are assumed to move to positions around the target prey, while for the bubble-net attacking method step each whale moves in a spiraling fashion around such targeted prey to corral it. These two behaviors are represented by the following position update operators:

$$\mathbf{x}_i^{k+1} = \begin{cases} \mathbf{x}_*^k - \mathbf{A}^k \cdot \mathbf{D}^k & \text{if } p < 0.5 \\ \mathbf{D}_*^k \cdot \left(e^{b \cdot l} \cdot \cos(2\pi l)\right) + \mathbf{x}_*^k & \text{if } p \geq 0.5 \end{cases}. \tag{52}$$

where $\mathbf{x}_*^k$ denotes the position of the current best solution from among all search agents (which further represent the targeted prey), whereas $\mathbf{D}_*^k = \left|\mathbf{x}_*^k - \mathbf{x}_i^k\right|$. Furthremore p stands for a random number drawn from within the interval [0, 1], while the value '0.5' stands for a probability threshold. The operator applied for $p < 0.5$ correspond to the prey encircling movement rule, while the operator applied when $p \geq 0.5$ represent the bubble-net attacking meod operator, fittingly represented by the model of a logarithmic spiral whose shape is controlled by the constant parameters b and a random value $l \in [-1, 1]$.

3.3 Physics-based methods

The nature-inspired techniques known physics-based methods comprise a series of optimization algorithms which design is inspired by the laws of physics that govern our universe. In this sense, the movement that search agents can manifest in this kind of algorithmic structures are usually based on some observable physical phenomenon, such as the movement caused by gravitational forces, the interaction between electrically charged particles, thermodynamical processes, light refraction, among others.

3.3.1 Electromagnetism-like mechanism

In Birbil and Fang (2003) proposed a population-based metaheuristic known as Electromagnetism-like Mechanims (EM) to solve global optimization problems. The EM approach was designed based on the laws of physics which govern the movement of charged particles within a given space (notably, the Coulomb's laws).

In EM, search agents are represented as electrically charged particles, interacting within a feasible solution space. Each of these particles is assigned with an individual charge value, associated to the quality (fitness) of the solution they represent. With that being said, for a given iteration 'k' the charge value associated to a specific particle 'i' is given by the following expression:

$$q_i^k = \exp\left(-d \frac{f\left(\mathbf{x}_i^k\right) - f\left(\mathbf{x}_{\text{best}}^k\right)}{\sum_{j=1}^{N}\left(f\left(\mathbf{x}_j^k\right) - f\left(\mathbf{x}_{\text{best}}^k\right)\right)}\right) \tag{53}$$

where $\mathbf{x}_{\text{best}}^k$ denotes the current best solution found so far by the algorithm's search process, while d stand for the dimensionality of the target solution space.

Furthermore, as illustrated by the Coulomb's law, each of these particles is assumed to be subjected to a series of electrostatic forces, which depend on the magnitude of their individual charges and the distance between them. Furthermore, these electrostatic forces may be either of attraction or repulsive, depending on the sign of each charge. In the EM approach, the total electrostatic force experimented by a given particle 'i' with regard to all other particles is modeled as:

$$\mathbf{F}_i^k = \sum_{j \neq i}^{N} \begin{cases} \left(\mathbf{x}_j^k - \mathbf{x}_i^k\right) \frac{q_i^k q_j^k}{r_{ij}^k} & \text{if } f\left(\mathbf{x}_j^k\right) < f\left(\mathbf{x}_i^k\right) \\ \left(\mathbf{x}_i^k - \mathbf{x}_j^k\right) \frac{q_i^k q_j^k}{r_{ij}^k} & \text{if } f\left(\mathbf{x}_j^k\right) \geq f\left(\mathbf{x}_i^k\right) \end{cases} \tag{54}$$

where $r_{ij}^k = \mathbf{x}_j^k - \mathbf{x}_i^k$ denotes the Euclidian distance between the particles 'i' and 'j.

Intuitively, as a result of such attraction/repulsion forces, each particle is forced to change their position at each time instant. In EM, the position update rule applied to each of such particles is given by the following equation:

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \lambda \cdot \hat{\mathrm{F}}_i^k \cdot \mathbf{m}_i^k \tag{55}$$

where λ denotes a d-dimensional vector of random numbers drawn from within the interval [0, 1], while $\hat{\mathrm{F}}_i^k = \mathbf{F}_i^k / \mathbf{F}_i^k$ stand for the normalized electrostatic force of to the ith particle's at iteration 'k'. Finally, $\mathbf{m}_i^k$ denotes a movement vector whose components $\left(\mathbf{m}_i^k = \left[m_{i,1}^k, m_{i,2}^k, \ldots, m_{i,d}^k\right]\right)$ depend on those of $\hat{\mathbf{F}}_i^k \left(\hat{\mathbf{F}}_i^k = \left[\hat{\mathrm{F}}_{i,1}^k, \hat{\mathrm{F}}_{i,2}^k, \ldots, \hat{\mathrm{F}}_{i,d}^k\right]\right)$, as illustrated as follows:

$$m_{i,n}^k = \begin{cases} ub_n - x_{i,n}^k & \text{if } \hat{F}_{i,n}^k > 0 \\ x_{i,n}^k - lb_n & \text{if } \hat{F}_{i,n}^k \leq 0 \end{cases} \tag{56}$$

with $x_{i,n}^k$ denoting the nth component of the particle's position $\mathbf{x}_i^k$, while lb_n and ub_n stand for the lower and upper fitness function bounds at the nth dimension, respectively.

3.3.2 Gravitational Search Algorithm

The Gravitational Search Algorithm (GSA) is a population-based metaheuristic proposed by Rashedi et al. (2009). The GSA design is mainly inspired by the laws of gravity, which establishes the inherent interaction between different objects (or masses) as a result of the gravitational forces experienced by them.

In GSA, search agents are modeled as N individual masses, subjected to constant interaction within d-dimensional solution space (also referred as a *system*). At each iteration 't' (also referred as *time*), each individual mass 'i' is assigned with a particular mass M_i, which value depends on its current solution quality (fitness), such that:

$$M_i^t = \frac{m_i^t}{\sum_{j=1}^{N} m_j^t} \tag{57}$$

where m_i^t denotes the normalized fitness value corresponding to the ith mass, as given by the following expression:

$$m_i^t = \frac{f(\mathbf{x}_i^t) - f_{\text{worst}}^t}{f_{\text{best}}^t - f_{\text{worst}}^t} \tag{58}$$

where f^t_{best} and f^t_{best} each denote the current best and worst fitness values at time 't', respectively.

Furthermore, akin to the law of gravity, each of these masses is assumed to be in constant interaction with each other as a result of the gravitational forces exerted by each of them. In GSA, the total gravitational force experienced by a particular mass 'i' with regard to all other masses is given by the following equation:

$$\mathbf{F}_i^t = \sum_{j \neq i}^{N} \left(G(t) \frac{M_i^t \cdot M_j^t}{r_{ij}^t + \varepsilon} \left(\mathbf{x}_j^t - \mathbf{x}_i^t \right) \cdot \mathbf{rand} \right) \tag{59}$$

where $r_{ij}^t = \mathbf{x}_j^t - \mathbf{x}_i^t$ stand for the Euclidian distance between masses 'i' and 'j', while ε is a small value used to prevent singularities. Furthermore, **rand** denote a d-dimensional vector of random numbers drawn from within the interval [0, 1]. Finally, $G(t)$ represents the so-called gravitational constant, whose value depends on the current time 't' as follows:

$$G(t) = G_0 e^{\left(-\alpha \frac{t}{T}\right)} \tag{60}$$

with G_0 denoting the initial gravitational constant value, α standing for a constant parameter and T representing the total iteration number which comprises the whole GSA search process.

Finally, as a result of such gravitational interactions, masses are also assumed to be able to experience a movement. With that being said, in GSA, the following position update rule is applied:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{61}$$

where $\mathbf{v}_i^{t+1}$ denotes the velocity of the ith mass at the time '$t+1$', as expressed by the following equation:

$$\mathbf{v}_i^{t+1} = \mathbf{rand} \cdot \mathbf{v}_i^t + \mathbf{a}_i^t \tag{62}$$

where $\mathbf{a}_i^t = \frac{1}{M_i^t} \mathbf{F}_i^t$ stand for the acceleration experienced by the mass 'i' at such time 't'.

3.3.3 Simulated annealing

Simulated annealing (SA) comprises one of the earliest and most successful local search methods specially devised to tackle complex discrete (and to a lesser extend continuous) optimization problems. As first introduced by Kirkpatrick et al. (2007), the SA approach is mainly inspired in the physical process of annealing (a process typically aimed to achieve an alteration on the physical properties of crystalline solids through heat treatment). In particular, a solid is first heated until it reaches a certain temperature, and then it is allowed to cool down slowly; if properly done, this process enables said material's microstructure to achieve a better crystal lattice configuration, and as such, superior structural integrity.

SA is essentially an iterative local search process, in which a given candidate solution is iteratively modified by implementing a computational procedure inspired by the temperature transition scheme modeled on a typical annealing process. As such, SA starts by first defining an initial solution (state) $\mathbf{s}$ and cooling schedule $\mathbf{T} = \{t_0, t_1, \ldots, t_n\}$, where the elements t_k each represent a finite transition temperature in the annealing process and such that:

$$t_i > 0 \text{ and } \lim_{k \to +\infty} t_i = 0 \tag{63}$$

Each temperature t_i in the cooling schedule is applied for a finite number of iterations of the SA's search process. With that being said, SA also defines a repetition schedule

$\mathbf{M} = \{M_0, M_1, \ldots, M_n\}$, where the elements M_i dictate the number of iterations a given temperature t_i will be applied.

Once the SA algorithm completes its initialization step, it starts an iterative search process in which, at each iteration 'k', a neighboring solution $\mathbf{s}'$ around the current best solution $\mathbf{s}^k$ is generated, either randomly or by following a particular criterion. After that, both solutions are compared in terms of fitness value and, depending on the outcome of such a comparison, there is a certain probability to accept said neighbor solution as the current best solution; for a minimization problem, for example, this probability is given by:

$$P^k = \begin{cases} \exp\left(-\frac{(f(\mathbf{s}')-f(\mathbf{s}^k))}{t^k}\right) & \text{if } f(\mathbf{s}') - f(\mathbf{s}^k) > 0 \\ 1 & \text{if } f(\mathbf{s}') - f(\mathbf{s}^k) \leq 0 \end{cases} \tag{64}$$

where $t^k \in \mathbf{T}$ denotes the transition temperature that is applied at given iteration 'k'.

As one of the earliest metaheuristics, the SA algorithm has been a subject of constant study and improvements. Some of the most well-known variants of SA involve changes to either the cooling schedule model, the neighborhood selection method, and even its learning mechanism (Siddique and Adeli 2016).

3.3.4 Sine Cosine Algorithm

The Sine Cosine Algorithm (SCA) is a population-based metaheuristic developed by Mirjalili (2016), which design is based on the properties of sinusoidal functions.

SCA consider a set of N search agents each with a corresponding position $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}]$ within a given d-dimensional solution space. In SCA, each available search agent updates their positions by applying one of two different movement operators, each modeled after some particular sinusoidal function (namely, the sine and cosine functions). With that being said, at each iteration 'k', the position update operator applied to each individual $\mathbf{x}_i^k$ is given by:

$$\mathbf{x}_i^{k+1} = \begin{cases} \mathbf{x}_i^k + r_1 \cdot \sin(r_2) \cdot \left|r_3 \cdot \mathbf{p}_i^t - \mathbf{x}_i^k\right| & \text{if } r_4 < 0.5 \\ \mathbf{x}_i^k + r_1 \cdot \cos(r_2) \cdot \left|r_3 \cdot \mathbf{p}_i^t - \mathbf{x}_i^k\right| & \text{if } r_4 \geq 0.5 \end{cases} \tag{65}$$

where $r_1 = a - k(a/T)$ (with a being a constant value and K denoting the maximum number of iterations for the whole search process), while r_2, r_3 and r_4 are random numbers drawn from within the uniformly distributed intervals $[0, 2\pi]$, $[0, 2]$ and $[0, 1]$, respectively. Furthermore, $\mathbf{p}_i^t$ stand for the current *destination point*, which is given by the position of best solution found so far by SCA's search process.

3.3.5 States of matter search

In Cuevas et al. (2013b) proposed a novel metaheuristic approach coined States of Matter Search (SMS). The SMS approach draws inspiration on the physical principles of thermal-energy motion, manifested by the molecules of a substance as it transitions from one state of matter to another.

In SMS, search agents are represented by individual molecules, in constant motion within the target d-dimentional solution space. A unique trait of SMS is that the whole evolutionary process is divided into three different stages, inspired by the three classic states of matter: (1) a gas state in which molecules are assumed to experiment constant movement and collision with other molecules; (2) a liquid state in which a significant reduction of molecular movement occurs as a result of a descent on the available thermal-energy; and (3) a solid state in which the bonding force among the molecules becomes so strong that their movement is almost completely inhibited. Each of such stages occurs in sequence as the SMS iteration process goes on, with each of them taking place for a finite number iterations. Under this considerations, each molecule moves with a particular "intensity", which depends on the current transitory state (gas, solid or liquid). In general, at each iteration 'k', each molecule updates its position by applying the following movement rule:

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^k \cdot \mathbf{rand} \cdot (\mathbf{ub} - \mathbf{lb}) * \rho \tag{66}$$

where $\mathbf{lb} = [lb_1, lb_2, \ldots, lb_d]$ and $\mathbf{ub} = [ub_1, ub_2, \ldots, ub_d]$ each denote the lower and upper bound vectors of the target solution space, respectively, while **rand** stand for a random vector whose values are drawn from the interval [0, 1]. Furthermore, $\rho \in [0, 1]$ denotes a scalar factor which value depend on the current SMS stage (Cuevas et al. 2013b). Finally, $\mathbf{v}_i^t$ denotes the velocity of the ith molecule at iteration 'k', as given by the following expressions:

$$\mathbf{v}_i^k = v_{init} * \mathbf{d}_i^k \tag{67}$$

where v_{init} denotes the magnitude of the initial velocity, as given as follows:

$$v_{init} = \beta * \frac{\sum_{n=1}^{d}(ub_n - lb_n)}{d} \tag{68}$$

where lb_n and ub_n each denote the lower and upper objective function bounds at the nth dimension, respectively, while $\beta \in [0, 1]$ denotes a scalar factor which, which similarly to ρ in Eq. (66), depends on the current SMS stage (Cuevas et al. 2013b). On the other hand, $\mathbf{d}_i^k$ stand for a direction vector corresponding to the ith, and is given by the following equation:

$$\mathbf{d}_i^k = \mathbf{d}_i^{k-1} * 0.5\left(1 - \frac{k}{K}\right) + \mathbf{a}_i^k \tag{69}$$

with K denoting the maximum number of iterations which compose the whole SMS's search process. Also, $\mathbf{a}_i^k$ denotes a unit vector oriented toward the current global best solution ($\mathbf{x}_{\text{best}}^k$), as given by the following expression:

$$\mathbf{a}_i^k = \frac{(\mathbf{x}_{\text{best}}^k - \mathbf{x}_i^k)}{\mathbf{x}_{\text{best}}^k - \mathbf{x}_i^k} \tag{70}$$

Another important trait present of SMS's is the inclusion of a collision mechanism, which causes molecules to exchange their directions. Essentially, a collision is said to occur among a pair of molecules 'i' and 'j' (with $i \neq j$) if their Euclidian distance $\mathbf{x}_i - \mathbf{x}_j$ is shorter than a given distance threshold. Once a collision occurs, the directions $\mathbf{d}_i$ and $\mathbf{d}_j$ corresponding to such molecules 'i' and 'j' are exchanged, such that:

$$\text{if } \left(\mathbf{x}_i^k - \mathbf{x}_i^k < r\right) \rightarrow \begin{array}{l} \mathbf{d}_i^k = \mathbf{d}_j^k \\ \mathbf{d}_j^k = \mathbf{d}_j^k \end{array} \tag{71}$$

where the distance threshold r (also known as *collision radius*), and is given by the following expression:

$$r = \alpha \cdot \frac{\sum_{n=1}^{d}(ub_n - lb_n)}{d} \tag{72}$$

where $\alpha \in [0, 1]$ is a scalar factor which, similarly to β and ρ in Eqs. (66) and (68), depends on the current SMS stage (Cuevas et al. 2013b).

3.4 Human-based methods

Human-based optimization algorithms are a special class of methodologies which design draws inspiration from several phenomena related to the behavior and the lifestyle of human beings. With that being said, this kind of techniques may be designed based on, either some cognitive process applied by humans to solve problems, or even on certain activities commonly related to the way in which human beings live.

3.4.1 Fireworks Algorithm

In Tan and Zhu (2010) proposed a novel optimization framework known as the Fireworks Algorithm (here referred as FWA). Interestingly, the inspiration for the FWA approach comes from the observation of several properties observed in the flashy explosions created by fireworks. In general, once a firework is set off, the resulting explosion creates a shower of sparks which spreads around a local radius around the denotation's location. In the context of swarm intelligence approaches, the way in which such sparks spread around the firework's explosion radius is seen as a specific case of a local search process around the location in which the firework was set off, and thus, is presented as a useful mechanism to perform optimization.

In the FWA approach, at each iteration 'k', a set of N locations within the feasible d-dimensional search space are chosen to set off individual fireworks. In the context of the fireworks explosion metaphor, explosions can be distinctively identified as either good or bad explosions. In this sense, good explosions create numerous centralized sparks and are assumed to be the result of well manufactured fireworks. On the other hand, bad explosions, which result from badly manufactured fireworks, generate fewer sparks and these scatter around a much larger local space. With that being said, after a firework is set off, several sparks are assumed to be spread within a fixed local area around the explosion´s location. Each generated spark is treated as a local search agent and evaluated with regard to the target objective function. Furthermore, for the given iteration 'k', the number of generated sparks s_i^k and the amplitude of explosion A_i^k for each deployed firework 'i' are said to depend on the quality of its manufacture, which is further represented by the quality (fitness) at its respective explosion location $\mathbf{x}_i$, as given by the following expressions:

$$s_i^k = m \cdot \frac{f_{\text{worst}}^k - f\left(\mathbf{x}_{\text{i}}^k\right) + \xi}{\sum_{i=1}^N \left(f_{\text{worst}}^k - f\left(\mathbf{x}_{\text{i}}^k\right)\right) + \xi} \tag{73}$$

$$A_i^k = \hat{A} \cdot \frac{f\left(\mathbf{x}_i^k\right) - f_{\text{best}}^k + \xi}{\sum_{i=1}^N \left(f\left(\mathbf{x}_i^k\right) - f_{\text{best}}^k\right) + \xi} \tag{74}$$

where m and $\hat{A}$ each denote the fireworks' maximum number of sparks and amplitude of explosion, respectively. Furthermore f_{best}^k and f_{worst}^k each denote the fitness values corresponding to the current best and worst solutions among the N fireworks, respectively, while ξ stand for a small value used to prevent singularities while computing either s_i^k or A_i^k. Furthermore, in order to avoid the overwhelming effects of "splendid" fireworks, bounds are defined for s_i^k with regard to a set of constant parameters a and b as follows:

$$\hat{s}_i^k = \begin{cases} \text{round}(a \cdot m) & \text{if } s_i < a \cdot m \\ \text{round}(b \cdot m) & \text{if } s_i > b \cdot m, \quad a < b < 1 \\ \text{round}\left(s_i^k\right) & \text{if otherwise} \end{cases} \tag{75}$$

Finally, for each firework 'i', each of the $\hat{s}_i^k$ generated sparks are randomly distributed around the local area defined by the amplitude of explosion A_i^k. In FWA, this is achieved by randomly selecting a number of affected directions (dimensions) for each individual spark, and then, a displacement magnitude is calculated within the explosion amplitude A_i^k. This could be effectively seen as a type of random walk, and as such, other similar methods could may be applied to define the locations of the generated sparks.

3.4.2 Harmony Search

The harmony search (HS) method, as proposed by Geem et al. (2001), is a metaheuristic approach inspired on the principles behind the process of harmony improvisation, in which musicians are said to compose a harmony by combining different music pitches stored in their memory, this with the purpose of finding the perfect harmony . In HS, the process of finding the perfect harmony is seen as an analogy to finding the optimal solution in an optimization problem.

HS is initialized by considering a set of N randomly generated solutions, collectively referred as the HS Memory. At each iteration 'k' of the HS search process, a new candidate solution $\mathbf{x}_c = [x_{c,1}, x_{c,2}, \ldots, x_{c,d}]$ is generated (improvised) and then evaluated against currently existing solutions; in particular, if the quality (fitness) of the proposed candidate solution $\mathbf{x}_c$ is better than that of the current worst solution in the HS Memory, then such worst solution is replaced by $\mathbf{x}_c$. Each component $x_{c,n}$ of the improvised solution $\mathbf{x}_c$ is generated depending on the value of the Harmony Memory Considering Rate ($HMCR$), as illustrated as follows:

$$x_{c,n} = \begin{cases} x_{r,n} + \text{rand}(-1, 1) \cdot bw & \text{if rand}(0, 1) < HMCR \\ \text{rand}(lb_n, ub_n) & \text{if rand}(0, 1) \geq HMCR \end{cases} \tag{76}$$

where $x_{r,n}$ denotes the nth component corresponding to a randomly chosen solution $\mathbf{x}_r$ within the HS Memory, whereas the parameter bw represent the so called distance bandwidth (essentially a step size value). Furthermore, rand(a, b) stand for a random number from within the interval $[a, b]$ (i.e., rand(0, 1) correspond to a random number between 0 and 1), while lb_n and ub_n are the objective function's lower and upper bounds at the nth dimension, respectively.

3.4.3 Imperialist Competitive Algorithm

In Atashpaz-Gargari and Lucas (2007) proposed a novel population-based metaheuristic known as Imperialist Competitive Algorithm (ICA). Said optimization technique is inspired by imperialism (or neocolonialism); that is the actions taken by individual countries to extend their power (typically through the acquisition of other territories).

At the initialization step, ICA starts by randomly generating a set of N search agents (called countries), each with an individual solution $\mathbf{x}_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,d}\}$ representing as a set of socio-political characteristics (such as culture, language, economy, religion, etc.). After generating such set of solutions, these are then classified as either imperialists or colonies. For this purpose, the best N_{imp} countries (according to fitness quality) are designated as imperialists countries $(\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_{N_{\text{imp}}})$, while the remaining $N_{\text{col}} = N - N_{\text{imp}}$ countries are labeled as colonies $(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{N_{\text{col}}})$. Afterwards, each of the available N_{col} colonies are proportionally distributed among the N_{imp} imperialists in order to form the empires. The number of colonies assigned to each imperialist is proportional to their respective imperialist power, as given by the following expression:

$$power_i = \left| \frac{cost_i}{\sum_{j=1}^{N_{\mathrm{imp}}} cost_i} \right| \tag{77}$$

where c_i denotes the normalized imperialist cost of the ith imperialist, as given by:

$$cost_i = \max_j \{f(\mathbf{i}_j)\} - f(\mathbf{i}_i) \tag{78}$$

By considering the previous, the total number of colonies that are assigned to each imperialist is given by:

$$NC_i = \mathrm{round}(power_i \cdot N_{\mathrm{col}}) \tag{79}$$

Once ICA's initialization process has been performed, the algorithm starts an iterative search process comprised of four main steps (1) Assimilation, (2) Revolution, (3) Intra-imperialistic competition and (4) Inter-imperialistic competition. For the assimilation process, each colony belonging to an empire is assumed to be influenced by the socio-political elements (culture, economy, religion, etc.) of its respective imperialist country. Said influence is represented as a movement from said colony $\mathbf{c}_{ij}$ toward is respective imperialist $\mathbf{i}_j$, as given as follows:

$$\mathbf{c}_{ij}^* = \mathbf{c}_{ij} + \mathrm{rand}(0, \beta) \cdot (\mathbf{i}_j - \mathbf{c}_{ij}) \tag{80}$$

where the value β is typically set to be between 1 and 2.

Then, in the revolution step, some randomly chosen colonies are assumed to experience some changes in their socio-political characteristics. Akin to the mutation operator in methods such as genetic algorithms (GA), the purpose of the revolution step is to apply sudden changes to some of the colonies in order to favor diversity of solutions, and thus preventing premature convergence or being trapped into local optima.

Furthermore, at the intra-imperialist, colonies that have performed a movement (either by assimilation or revolution) compete for the position of imperialist. Essentially, if any country $\mathbf{c}_{ij}$ within an empire has a better quality than that of their respective imperialist $\mathbf{i}_j$, then $\mathbf{c}_{ij}$ is labeled as the new imperialist, while $\mathbf{i}_j$ is considered as a colony. Finally, for the inter-imperialist step, one of weaker the colonies within the weakest empire is given to another empire based on competition. For this purpose, a probability for possessing said weak colony is first assigned to each empire as follows:

$$P_j = \left| \frac{NTC_j}{\sum_{i=1}^{N_{\mathrm{imp}}} NTC_i} \right| \tag{81}$$

where NTC_j denotes the normalized total cost of the jth empire as given by:

$$NTC_j = \max_i \{TC_i\} - TC_j, TC_j = f(\mathbf{i}_j) + \zeta \cdot \mathrm{mean}\{f(\mathbf{c}_{ij})\} \tag{82}$$

where the value $\zeta \in [0, 1]$ represents the influence that the mean power of the colonies has when determining the empire's total power.

After a possessing probability P_j has been assigned to each empire, an appropriate selection method, (such as the roulette selection approach) is applied to decide which empire will take charge of the disputed colony. This process is applied at each iteration of ICA's search process. As a result of the inter-imperialistic competition, weaker empires suffer a gradually decrease in power as they lose their colonies over time, while stronger empires increase their power as they take possession of said colonies. This eventually lead weaker empires to collapse over time until only a single strongest empire remains.

4 Performance of nature-inspired metaheuristics

Nowadays, nature-inspired metaheuristics have become so numerous and varied in terms of design and applications. From such an abundant variety of techniques, the main question to be addressed is: Which metaheuristic technique performs the best overall? This question, present since the formulation of the first of such algorithms, has been surprisingly hard to answer, and up to this day, it remains as an open concern in this area of science (Neumann and Witt 2010). One widely accepted theory in this research field is that metaheuristic algorithms perform best over a broad spectrum of problems when a proper balance between exploration and exploitation is present in their mechanism. In general terms, exploration refers to the ability of search agents to visit entirely new regions of a search space, while exploitation emphasizes on the capabilities of these agents refine currently known "good" solutions (Črepiňsek et al. 2013). While the importance of such a balance is recognized as essential in most new proposals, it is often loosely implied given the lack of appropriate analysis tools that allow understanding how an algorithm search mechanisms affect it. Another factor that highly affects this balance is the selection of parameters (tuning) designed to control the exploration and exploitation pressure of each algorithm. However, this process is not straightforward, and it highly depends on the algorithm and the problem to solve. Also, there is the so-called 'No-free-lunch (NFL) theorem' proposed by Wolpert and Macready (1997), which states that any algorithm will on average perform equally well as a random search algorithm over all possible functions. For this reason, it is assumed that statistical methods can be applied to find the dominance of one algorithm over others on a specific problem; however, as of today, there is no objective way to point out which algorithm is the overall best and the reasons of such superiority, if existent (Lin et al. 2012). In this section, we open a discussion about the several observable characteristics of metaheuristic algorithms and how these characteristics impact the performance of these methods (see Table 1).

4.1 Computational complexity

An in-depth analysis of the mechanisms implemented on nature-inspired metaheuristics allows pointing on several characteristics which have a direct impact on the expected computational complexity of these methods (Avigad and Donnelly 2004). Depending on their search strategy, for example, some algorithms may require to sort the available candidate solutions with regard to their fitness value, either to find the best members from the population (as in GA), select several good solutions to implement their search strategy (as done in GWO), or even as a tool for efficient implementation (as in the case of FA). Although population sorting can be implemented using several sort algorithms, for most cases it is also important to consider their implied computational cost. In the case of the default sort function employed by MatLab® (QuickSort), for example, the computational complexity of the sorting operation is $O(P\log(P))$ in the average case scenario. The additional complexity these operations add breaks the desired linear complexity that is often sough when developing optimization algorithms, especially when sorting is required on each iteration (although this becomes a real burden only if the population size is too large). On the other hand, some algorithms may require to calculate some kind of population-related measurement(s) as part of their search mechanisms. A measurement commonly computed in several nature-inspired algorithms is the Euclidian distance between individuals within the search space. As seen in the case of GSA, for example, such distance measurements are used to compute a sum of weighted attraction between solutions. On the other hand, in algorithms such as SMS,

Table 1 Observable performance-influencing characteristics on several popular nature-inspired metaheuristics

Algorithms	Exploration/exploitation			Computational complexity			Required additional memory	Parameter Tuning Process	Overall imple-mentation difficulty
	Selection mechanism	Attraction operators	Iteration dependent	Population sorting	Population-related measurements	Variable FFC			
DE	Ind. Greedy	N/A	No	No	No	No	No	Easy	Low
ES	Non-Greedy	Global Best	No	No	No	No	No	Hard	Low
GA	Greddy	N/A	No	Yes	No	No	No	Easy	Low
ABC	Ind. Greedy	Multiple	No	No	No	Yes	Yes	Tuneless	Medium
BA	Ind. Greedy	Global	No	No	No	No	Yes	Easy	Low
CSA	Non-Greedy	Personal	No	No	No	No	Yes	Easy	Low
CS	Ind. Greedy	Global	No	No	No	No	No	Easy	Medium
FA	Non-Greedy	Multiple	No	No	Yes	Yes	Yes	Hard	Medium
FPA	Non-Greedy	Global	No	No	No	No	No	Easy	Low
GWO	Non-Greedy	Multiple	No	Yes	No	No	No	Tuneless	Low
KH	Ind. Greedy	Global Best	No	No	Yes	No	Yes	Hard	Medium
MFO	Greedy	Multiple	Yes	Yes	No	No	Yes	Tuneless	Medium
PSO	Non-Greedy	Multiple	No	No	No	No	Yes	Easy	Low
SSO	Ind. Greedy	Multiple	No	No	Yes	No	Yes	Hard	High
WOA	Non-Greedy	Global Best	No	No	No	No	No	Tuneless	Low
EMO	Non-Greedy	Multiple	No	No	Yes	No	Yes	Tuneless	Medium
GSA	Non-Greedy	Multiple	Yes	No	Yes	No	Yes	Easy	Medium
SCA	Non-Greedy	Global Best	Yes	No	No	No	No	Tuneless	Low
SMS	Non-Greedy	Global Best	Yes	No	Yes	No	No	Hard	High
FWA	Greedy	Multiple	No	Yes	Yes	Yes	Yes	Hard	High
HS	Ind. Greedy	N/A	No	No	No	No	No	Easy	Low
ICA	Non-Greedy	Multiple	No	No	Yes	Yes	Yes	Hard	High

the Euclidian distance between individuals is compared with some sort of distance threshold in order to establish conditions for the movement of search agents. While the Euclidian distance often proves to be useful for the purpose of modeling complex search strategies, the added computational complexity that this implies must always be considered. Under the supposition that Euclidian distances are needed to be calculated with regard to every single pair of individuals in the population (worst case scenario), this will require at least as it requires $O(P^\wedge 2)$ root square calculations (although this can be reduced under certain conditions). Finally, while it is common to find optimization algorithms that perform a fixed number of fitness function computations (given by the product between the population size and the maximum number of iterations that comprise the search process), there are several metaheuristic algorithms that break this rule. In methods such as ABC, for example, scout bees are deployed to explore new solutions within the search space once it is determined that any of the currently known solutions cannot be improved, hence requiring an additional FFC for each abandoned solution. Similarly, in the CS algorithm, a secondary mechanism devised to generate new random solutions is implemented as a mean to increase diversity, adding additional FFCs in the process.

4.2 Memory efficiency

All the population-based metaheuristic required a minimum memory with size $O(N*(D+1))$ (with N denoting the number of elements D their dimensionality) to store each of the available solution vector solution plus their correspondently fitness value, and $(N*(D+1)+a*(D+1))$ (with $a \in \{1, 2\}$) if memorizing the best and/or worst solution(s) is required. Furthermore, some nature-inspired algorithms are known to require extra memory space to store some kind of additional data that is required as part of their search strategy. Some examples of algorithms that required such extra information include SMS, KH and GSA, where measurements such as the Euclidian distances are constantly computed. In these algorithms, such distance measurements are store in memory in order to reduce the number root squares calculations that are needed at each iteration, instead of performing the calculation as they're required (Yang and He 2013). Another algorithm with a more complex demand of memory is the ICA that require the calculation of the distance between colonies and empires, and their proposed costs on each iteration. While these computational requirements may represent no issue for today standard computers, this can severally limit their application on hardware with limited resources (such as those developed with the idea of portability in mind).

4.3 Exploration versus exploitation

Although seemingly trivial, questions about how exploration and exploitation is achieved are still an open subject, and it's often a source of disagreement among researchers (Ghazali et al. 2018; Yang et al. 2015); however, it is commonly accepted among the research community that a good ratio between exploration and exploitation is essential to ensure good performance in metaheuristic search algorithms. The question here is: how can we find a proper balance between these two crucial characteristics? Answering this question is not trivial given that metaheuristic optimization algorithms can be very different in terms of search strategy, so it is necessary to understand first what type of mechanisms are implemented in these methods. In the case of population-based metaheuristics, for example, these usually include selection mechanisms that allow them to collect prominent solutions from among the entire population, either to make them prevail for the next iteration of the search process or to implement some solution update strategy. Although selection operators do not modify or generate new solutions in the population, they play an important role in balancing elitism and diversity (Du et al. 2018). In the case of a greedy selection mechanism, for example, this ensures that the most fitting individuals among candidate solutions remain intact for the next generation. Thus, this type of selection is often used to improve fast convergence on metaheuristic algorithms toward promising solutions. Another type of elitism (individual greedy selection) can be observed in algorithms like DE or HS, where a new solution is accepted only if it directly improves the solution that originates them. This strategy has the appeal of potentially improves the exploration–exploitation ratio by forcing a highly diverse initial population to improve individually from its starting point, and as such this kind of selection method is often preferred in metaheuristic optimization methods, especially when these are meant to deal with multimodal problems (Huang et al. 2017). Also, some algorithms do not implement any kind of selection strategy at all. In the case of methods like PSO or GWO, these do not enforce the selection of prominent individuals, accepting every new solution independently of its quality. In this case, the lack of elitism is preferred to accent exploration over exploitation; however, this methods also implement additional behaviors in order balance the exploration–exploitation rate as is, for example, "attractions" toward the best know solution. Speaking of attractions, it is not uncommon to find some kind of attraction strategy implemented as a part of the search mechanisms of nature-inspired algorithms. As previously implied, attractions are usually applied as an exploitation mechanism that seeks to improve currently known solutions by moving other solutions toward the location of seemingly "good" individuals. Choosing which solutions are to be considered as attractors, and how other solutions will be attracted to these attractors is entirely dependent on the design of the algorithm itself. In the case of PSO, for example, particles are attracted toward the global best solution at the current iteration of the search process; however, individuals modeled in PSO also implement a series of attractions toward the best solutions recorded by each particle as the search process evolves (personal best solution). This approach is usually considered as a more balanced attraction mechanism regarding convergence and solutions diversity; however, implementing this kind of strategy requires the allocation of additional memory, which could be undesired depending on the intended application(s). Also, some algorithms apply attraction mechanisms which consider the composite effect of more than one attractor to discover new potential solutions. These attractors can be comprised by a subset of members from the current population of solutions, or even, by the whole set of available candidate solutions. As previously noted, in PSO both the best-known solution of the population and the best personal record of each solution are considered to modify the velocity of each particle; this could be considered an example of such multiple-attractors phenomenon. On the other hand, there are methods that implement more complex attraction schemes, which consider very specific members of the entire population as well as other particular properties. In FA, for example, the attractiveness between individuals is set to be inversely proportional to the distance that separates them, hence, the longer the distance, the lower the attraction (Yang et al. 2015). Similarly, GSA attractiveness is dictated by "gravitation force" exerted among particles in the available search space, but the magnitude of such attraction also depends on the fitness value of each particular solution. While these mechanisms have proven to be effective for maintaining higher diversity on the population, it is worth noting that modeling these behaviors requires to constantly calculate of the distance between several pairs of solutions, increasing the computational complexity to these the algorithms. Also, it's been observed that this kind of mechanisms tend to slow-down the convergence of the algorithm, a fact that must be considered is execution time is a priority (Yang et al. 2015). Furthermore, some algorithms do not contemplate any type of attraction as part of their search strategy;

instead, these methods generate new solutions by means pure random walks (as in the case of HS) or by considering other criterions (such as the distance between solutions, as in the case of DE). In evolutionary algorithms like GA, for example, some solutions are generated by "mixing" the information of randomly chosen solution (crossover), whereas others are generated by adding perturbations to currently existing solutions (mutation). In this regard, it's also worth mentioning that, while crossover and mutation operators in evolutionary algorithms are often perceived as exploration and exploitation operators, respectively, a deeper observation of crossover behaviors suggest that at the beginning of the evolutionary process (where the population is still diverse), crossover operators favor the exploration of solutions, whereas, toward the end of the search process (where population has lost diversity), exploration capabilities are dramatically reduced. Similarly, mutation operators that consider large amounts of perturbations for modifying existing solution could very easily be considered as exploration mechanisms, as solutions could be generated over much larger proportions of the feasible solution space. Under this considerations, it is hard to roughly classify crossover and mutation as either exploration or exploitation operators, as they intended behavior could be altered by adjusting their crossover and mutation rates, respectively. Finally, it worth noting that there are several algorithms which consider the iterations stop criterion (maximum number of iterations), as well as the iterations progress as part of their search strategy. Such mechanisms are mostly used to adjust the exploration–exploitation rate as the search process evolves with the purpose of avoiding premature convergence. However, this strategy often prevents the algorithm to converge quickly toward currently known best solutions, which could be undesired depending on the situation.

4.4 Implementation

As mention in Sect. 2, most metaheuristic algorithms share a general framework independently of the inspiration. Therefore the main difference is present in their mechanism to generating new solutions and selecting those that will remain to the next iteration. Among the algorithms presented in this work, some have a high level of simplicity that can be translated onto computational code with relative ease, while others may be relatively complex to program given the kind of behaviors and rules these intend to model. In other words, it could be said that the number of lines of code required to program a given metaheuristic algorithm increases as more sophisticated mechanisms are integrated to these methods. In this sense, algorithms such as SCA can be considered among the most straightforward algorithms to code as all of the population is directly attracted toward the best solution, without the need of sorting the population or excessive memory assignation. HS may be as well considered ease to code given how simple is for it to generate new candidate solutions (generating random values, values from known solutions, or slight perturbations of currently existing ones). However, the fact that HS resort to a greedy selection mechanism demands it to perform population sorting, slightly increasing its coding difficulty, and by extension its computational complexity. Leaving aside the performance that a given algorithm could have when applied to solve problems, the degree of coding complexity that relates to these computational approaches is plagued by two particular concerns (Piotrowski and Napiorkowski 2018): (1) A higher risk for some of its elements to introduce structural biases, making the algorithm more prone to explore several parts of the optimization landscape more frequently than others without an actual justification (Piotrowski and Napiorkowski 2016); and (2) The discouragement this coding complexity could cause to its potential users. Besides, most (if not all) metaheuristic algorithms are designed to work as black box models, thus being problem independent. However, regarding implementation, users often require to invest additional time to specify other essential characteristics, such as the formulation of the fitness function that describes the problem, the codification (representation) of the solutions and, if necessary, the parameters tuning for the algorithm in question. Speaking of parameter tuning, it is important to remember that its relevance lies on the fact that it allows to change the exploration–exploitation ratio of the algorithm, potentially allowing it to perform better over certain tasks. The main problem here though, is that there is no universal agreement on how metaheuristic algorithms should be tuned to work optimally over specific optimization problems (which is not surprising given that most algorithms are comparably different); in fact scientists and practitioners are for the most part used to tune metaheuristics by hand, guided only by their experience and by some rules of thumb, hence, tuning metaheuristics is often considered to be more of an art than a science. Although, there are several efforts aimed to provide frameworks to mechanically and objectively select these parameters finding the optimal tuning for these parameters is still an open problem (Yang 2018; Sipper et al. 2018). Another issue related to parameter tuning in metaheuristic algorithms is the number tuning parameters itself. While some algorithms like GA or PSO have only a reduced number of parameters (two in both cases), other methods such as ICA or SMS requires the user to set several more parameters, needed by the algorithm in order to control their behavior. However, as the number of parameters to set increases, understanding how these values influence the performance of the algorithm becomes more complex, thus, making the algorithm harder to analyze overall. Finally, it is worth noting that there are algorithms that do not have any parameters to tune at all. Methods such as WOA and SCA integrate mechanisms devised to automatically adapt the exploration–exploitation rate as they progress over the available iterations, but these do not need the user to set any parameters for it to work. Methods like these offer inexperienced users the ability to easily implement and understand the mechanisms behind metaheuristic algorithms, although these private more experimented users from studying its behavior in a more complex perspective.

5 Nature-inspired metaheuristics and their applications

In recent years, nature-inspired metaheuristics have become popular choices for solving a wide-range of optimization-related problems in many different areas of application such as engineering design, digital image processing, and computer vision, networks and communications, power, and energy management, data analysis and machine learning, robotics, medical diagnosis, and many others (Osman and Laporte 1996). In this section, we will discuss several implementations of nature-inspired algorithms for solving real-world problems in different areas of application.

5.1 Engineering design

Applications of nature-inspired metaheuristics to engineering design are as varied as the multidisciplinary areas of science currently on existence. For example, in the area of networks and communications, a popular design problem is the design of antennas. Notably, in Goudos (2017) the author presented a study where this design problem is tackled by applying several discrete-coded metaheuristics, including GA, DE, and PSO, demonstrating competence in all cases (Goudos et al. 2017). Another representative area of applications is aeronautics, where the most common design problems are related to aircrafts design. In Keshtegar et al. 2017, for example, the authors proposed an optimization framework for the design of aircraft panels based on an adaptive dynamic HS (ADHS). This design approach was compared in terms of performance against those found in other variants of HS, ultimately demonstrating ADHS to be the superior method (Keshtegar et al. 2017). Another common design application is the design of truss structures. As examples, we can mention the works presented in Bekdaş et al. (2015) and Khatibinia and Yazdani (2017), where algorithms such as FPA and GSA were successfully applied to solve this design problem (Bekdaş et al. 2015; Khatibinia and Yazdani 2016). Another interesting application is reported in Shukla and Singh 2016, where the FA algorithm is implemented to aid on the selection of parameters for advanced machining processes with good results (Shukla and Singh 2016). Other classical engineering design problems where nature-inspired algorithms have been successfully applied include the design of tension/compression springs, welded beams, pressure vessels, gear trains, to name a few (Mirjalili and Lewis 2016; Askarzadeh 2016; Mirjalili et al. 2014; Kaveh and Khayatazad 2012; Camarena et al. 2018).

5.2 Digital image processing and computer vision

Nature-inspired metaheuristics have also found interesting applications in the area of digital image processing and computer vision. One typical application in this regard is image segmentation by multilevel thresholding, in which the optimization algorithm is applied to found a set of optimal gray-level threshold that maximizes some kind of image measurement (Mesejo et al. 2016). Prominent examples this kind of applications includes those presented in Horng and Jiang (2010), Ouadfel and Taleb-Ahmed (2016), El Aziz et al. (2017), He and Huang (2017), Khairuzzaman and Chaudhury (2017a, b), were algorithms such as ABC, SSO/FPA, WOA/MFO, FA, GWO have been successfully implemented. Also, in Cuevas et al. (Cuevas et al. 2013a, b), Oliva et al. (2014) and Zhang and Zhou (2017), the task of template matching based on nature-inspired metaheuristics have been explored, where techniques such SMS, EMO, GWO have been implemented to good results (Khairuzzaman and Chaudhury 2017a; Horng and Jiang 2010; Ouadfel and Taleb-Ahmed 2016; El Aziz et al. 2017; He and Huang 2017). Also, in Olague and Trujillo 2012, the authors proposed to use a Multi-Objective GP (MO-GP) approach for the task of synthesizing operators for the detection of interest points in digital images, where the optimization problem is represented regarding three properties (stability, point dispersion, and information content). The experimental results presented by the authors suggest that their proposed approach is able to construct interest point detection operators adapted to different performance criteria, thus making it promising for a wide variety of computer vision applications (Olague and Trujillo 2012). Another interesting application is reported in Kiranyaz et al. 2015, where PSO is applied to assist on the task of perceptual dominant color extraction, presenting promising results when compared to some traditional methods (Kiranyaz et al. 2015).

5.3 Networks and communications

Applications of nature-inspired metaheuristics to this area include the Optimal Sensor Deployment (OSD) for Wireless Sensor Networks (WSNs), a task that consists on finding an optimal distribution for a set of sensing devices designed to collect some kind of physical data. One recent application for this is reported in Zhou et al. 2017, where the authors proposed an optimal sensor deployment scheme based on the SSO algorithm. The reported experiments suggest that the performance of SSO, when applied for OSD in WSNs, is superior to that of methods based on Virtual Force Algorithm (VFA) (Zou et al. 2003), GA, and PSO (Zhou et al. 2017a, b). Another similar application is reported in Deif et al. (2017), where ACO is applied as the optimization method of choice. In this work, there is a special emphasis not only on maximizing the sensor networks' coverage area, but also other important requirements in WSNs, such as minimum level of reliability and deployment cost (Deif et al. 2017). In similar terms, in Alia and Al-Ajouri (2017), the HS algorithm is applied for solving the problem of minimum cost OSD, comparing its performance in this case with that of a random deployment scheme (Alia and Al-Ajouri 2017). Mann and Singh (2017) managed to improve the performance of the ABC algorithm by implementing the Student's t-distribution as a sampling method and applied it to perform energy-efficient clustering in Wireless Sensor Networks, yielding to promising results (Mann and Singh et al. 2016). Other similar applications of nature-inspired methods for OSD in WSNs can be found in Goyal and Patterh (2015) and Cao et al. (2012), where the techniques such as BA and FA have been studied. Community detection in complex networks is another application that has caught the interest of researchers as a candidate to be solved by metaheuristic techniques. In Rahimi et al. (2018), for example, a novel multi-objective community detection scheme based on PSO is proposed. The authors presented experimental results that consider both synthetic and real-world environments and compared their results against those of other similar approaches, showing an outstanding performance (Rahimi et al. 2017). Also, in Guerrero et al. 2017, the problem of adaptive community detection is handled by applying a modification of the GA algorithm coined Generational Genetic Algorithm (GGA+), which involves efficient initialization methods and modularity-guided search operators, is applied and compared against other GA variants, demonstrating superior performance (Guerrero et al. 2017). Another interesting application of nature-inspired methods to this area is documented in Bhardwaj et al. 2014, were the ABC algorithm is implemented for the detection of malicious URLs (Bhardwaj et al. 2014). Also, in Din et al. (2016), the CS algorithm is applied to perform Left Feedback Shift Registers (LFSR) cryptanalysis, a tool mainly employed in information security (Din et al. 2016). Finally, E-mail Spam detection via nature-inspired methods has also been a subject study. In Idris et al. (2015), for example, the authors proposed an e-mail spam detection system based on a combination between Negative Selection Algorithm (NSA) (Johny and Assistant 2017) and PSO, where the latter is applied to improve the random detector generation phase in the former. The statistical data reported in this paper suggest a significant improvement in performance when compared to a framework based exclusively in NSA, thus proving the significance of the proposed modification (Idris et al. 2015).

5.4 Power and energy management

Applications of nature-inspired metaheuristics to the area of energy applications are quite numerous. In Mesbahi et al. (2017), for example, the authors proposed an optimal energy management strategy for hybrid energy storage systems based on PSO and the Nelder-Mead simplex method, which show to have a significant improvement in both battery usage and lifetime when compared to other conventional methods (Mesbahi et al. 2017). In You et al. (2017), a home energy management system based on the SSO algorithm was proposed, showing interesting results when applied for the task of appliances load management. Guha et al. (2016) presented an approach based on the GWO algorithm aimed to solve the problem of load frequency control in interconnected power systems networks by tuning the parameters of a PI/PID controller, showing to outperform schemes based on other similar metaheuristic techniques. In Prasad et al. (2017), a modification of the KHA, known as Chaotic Krill Herd Algorithm (C-KHA) was implemented to solve the problem of Optimal Power Flow (OPF) while also comparing its performance against other state-of-the-art metaheuristics, yielding to favorable results (Prasad et al. 2017). Another interesting application was proposed in Van Sickel et al. (2007), where the DE algorithm is used for the intelligent control of power plants. In this work, DE is applied to both generate a set of optimal points for the monitoring of a reference power governor and the parameter tuning of the same power plant controller. Also in Al-Betar et al. (2018), an approach aimed to solve the problem of Economic Load Dispatch (ELD) was proposed, in which an approach called β-Hill Climbing is applied to minimize the total fuel cost of operation and emission from a set of generating units. The proposed method was evaluated by considering several real-world ELD systems and compared against other simlar approaches (including some based in metaheuristics such as GA, ACO, HS and PSO), demonstrating β-Hill Climbing to be superior for solving this particular problem (Al-Betar et al. 2018). Furthemore, in Babu et al. (2017), the authors proposed the use of PSO for the reconfiguration of photovoltaic (PV) panels aimed to maximize power extraction, presenting promisong results when compared to other similar approaches (Babu et al. 2017). Related to the subject of PV-basesd power generation, another interesting application represented by parameter identification in PV cells/modules. In Oliva et al. (2014), for example, the Artificial Bee Colony (ABC) algorithm was implemented and compared againts other similar techniques in terms of performance, showing promising results (Oliva et al. 2014). Other similar applications for PV cells parameter identification can be found in Askarzadeh and Rezazadeh (2012), Ma et al. (2013), Han et al. (2014), and Sarjila et al. (2016), where algorithms such as FA, CS, AFSA, GSA where successfully implemented. In Valdivia-Gonzalez et al. (2017a, b), an intelligent power allocation scheme for plug-in hybrid vehicles (PHEVs) based on SMS was proposed, in which the objective is to adjust the power distribution provided by PHEV's charging infrastructures by taking into account customer characteristics and restrictions. In Prakash and Lakshminarayana (2016), the authors prosed to tackle the problem of optimal capacitor placement in Radial Distribution Networks by applying an optimization scheme based in WOA, demonstrating to be much more effective than most traditional techniques applied for this purpose. Other applications include those presented in Massan et al. (2015) and Tolabi and Ayob (2014), where nature inspired metaheuristics such FA and a hybridization between SA and GA are applied for the task of optimal wind turbines allocation (Massan et al. 2015) and solar radiation forecasting , respectively, yielding to interesting results in both cases.

5.5 Data analysis and machine learning

As for the area of data analysis and machine learning, some prominent application related to these areas include feature selection. In Mafarja and Mirjalili (2016) proposed a series of wrapper feature selection scheme based on hybridization between the WOA and SA algorithms, where the former is used to promote exploration while the latter is applied to enhance exploitation. The proposed approach was compared in terms of performance against other similar methods based on metaheuristics, such as ALO, PSO, and GA, proving to have the best performance regarding accuracy and average selection size (Mafarja and Mirjalili 2016). In the same year, Hafez et al. presented a feature selection approach based on the SCA algorithm, and compared it in terms of performance against methods based in PSO and GA, leading to favorable results. Similarly, in Emary et al. (2016) the authors published a feature selection methodology based on a binary GWO which was also successful regarding performance against PSO and GA. Also, in Moayedikia et al. (2016) proposed a feature selection algorithm called SYMON which applies both symmetrical uncertainties along with the HS algorithm to develop a strategy to rank features in high dimensional imbalanced class datasets, proving to be superior when compared to other similar techniques. Other interesting feature selection applications can be found in (Wu et al. 2011; Wang et al. 2015), Another application related to data analysis is data clustering, where metaheuristics have also been applied for good results. In Alswaitti et al. (2018) proposed a clustering method based in PSO which integrates a density estimation technique based on Gaussian kernels developed to address premature convergence as well as a method to estimate the best learning coefficients. The proposed approach was compared against other techniques commonly used for data clustering and proved to be significantly better in terms of accuracy. Also, in Zhou et al. (2017a, b) the SSO algorithm was modified to implement a Simplex method in order to improve data clustering on higher dimensions, proving to be better than the original SSO in terms of performance, while also showing its superiority over other similar methods. Furthermore, in Abualigah et al. (2016, 2017a, b) the authors proposed a clustering technique based on a hybrid KHA that integrates the mechanisms of the HS algorithm as a way enhance both exploration and exploitation of solutions. The algorithm was compared regarding performance against traditional clustering methods and approaches based on metaheuristic optimization algorithms demonstrating a higher performance in general (Abualigah et al. 2017a). Other similar works are reported in (Abualigah et al. 2016, 2017b; Mohammad et al. 2015; Abualigah and Khader 2017), where algorithms like the standard KH, GA, and PSO has been applied to improve text document clustering. Then, in Han et al. (2017) proposed a clustering method based on a variant of GSA that integrates an update mechanism devised to increase the diversity of solutions. This variant, called Bird Flocking GSA (BFGSA) was compared against the standard GSA as well as methods based on ABC, PSO, and FA, proving to be much more competent for the aforementioned task (Han et al. 2016). Other approaches for data clustering include those reported in Shukla and Nanda (2016)and Jadhav and Gomathi (2016), where methods such as SSO and a hybrid between GWO and WOA were implemented to good results. Other interesting applications of nature-inspired metaheuristics in this area involve their use as an alternative to train Artificial Neural Networks (ANN). In Sahlol et al. (2009), for example, the authors developed a feedforward ANN based on the SCA algorithm to improve the prediction accuracy of liver enzymes on fishes fed with certain compounds. In this approach the SCA algorithm is applied to find the configuration of weights/biases that allows the proposed NN to achieve optimal performance, yielding to much better results than those of previous prediction models (Sahlol et al. 2016). In Rere et al. (2015), for example, the authors proposed to use the SA algorithm to train a Convolutional Neural Network (CNN) for the classification of handwritten digits. Although the proposed method comes with a significant increase in computation time, it also yields a substantial increase in performance when compared to other methods commonly used to train CNNs (Rere et al. 2015). Another interesting application is reported in Pereira et al. (2016), where the SSO algorithm is used for both, feature selection and parameter tuning for a Support Vector Machine (SVM) designed to aid on energy theft detection. The authors compared the performance of their proposed approach against those based on PSO and a variant of the HS algorithm known as Novel Global Harmony Search (NGHS), highlighting their respective advantages (Pereira et al. 2016).

5.6 Robotics

Implementations of nature-inspired optimization algorithms to the area of robotics usually include tasks such as path planning and trajectory optimization. Perhaps the most popular application in this regard is the autonomous navigation of Unmanned Aerial Vehicles (UAVs). Interesting work is reported in Li and Duan (2012), where an improved GSA (IGSA) approach is applied to develop a path planning strategy for Unmanned Aerial Vehicles (UAVs) devised for military applications. The proposed I-GSA based path planning method was compared against those based on the classic GSA and PSO, demonstrating much better performance (Li and Duan 2012). Similarly, in Oz et al. (2013), path planning strategies for 3D environments based on GA and Hyper-Heuristics (HH) (Burke et al. 2013) that considers technical constraints and mission-specific objectives are proposed. Both algorithms were evaluated by considering other classical studies developed for UAV navigation, yielding to favorable results (Oz et al. 2013). Then, in Behnck et al. (2015), the authors proposed a path planning strategy based on a modified SA algorithm. The reported results demonstrate that the proposed approach can calculate minimum distance paths for a pair of UAVs while also being sufficiently simple to be implemented in an embedded platform (Behnck et al. 2015). Also, in Xie and Zheng (2016), the problem of path planning in UAVs is handled by applying a search strategy based on a hybrid CS algorithm that integrates the mutation and crossover operators of GA, called Improved CS (ICS). As illustrated by the results presented in this work, the ICS notably outperform the standard CS algorithm, demonstrating its competence for the task in question (Xie and Zheng 2016). Outside of UAVs applications, metaheuristic optimization approaches have also been successfully applied to solve the problem of navigation in other kinds of robotic platforms. In Tsai et al. (2016), for example, a path planning scheme based on a multi-objective GWO (MOGWO) approach is applied to aid on robot navigation over environments consisting on fixed positions and obstacles. The proposed path planning scheme was compared regarding performance against that based on a multi-objective GA (MOGA), ultimately proving MOGWO to be slightly superior (Tsai et al. 2016). Furthermore, in Contreras-Cruz et al. (2017), a distributed path planned method for multi-robot systems based on ABC (DPABC) was simulated and compared against a classic priority planner scheme and another approach also based on ABC (PPABC). As suggested by their experimental results, the proposed DPABC approach is favored as the better alternative for solving this problem due to it being able to solve the task in a lower time and with a better performance than that of the other compared methods (Contreras-Cruz et al. 2017). Another interesting application documented in the literature is the development of controllers for robotic platforms. In Silva et al. (2014), for example, GP was applied as an automatic search method for motion primitives in a bipedal robot based on the exploration and exploitation of its particular characteristics. Experimental results demonstrated a significant improvement in performance on the applied robot's locomotion, especially when compared to that of a hard-tuned system (Silva et al. 2014). More recently, in Benkhoud and Bouallègue (2017), the authors proposed a series of metaheuristic-based tuning strategies for a Linear Quadratic Gaussian (LQG) controller, with application to a special class of UAV. The algorithms studied in this work include the HS algorithm, Water Cycle Algorithm (WCA) (Eskandar et al. 2012), and Fractional PSO-based Memetic Algorithm (FPSOMA). Furthermore, comparisons which consider tuning methods based on the standard PSO and ABC algorithms where also developed, yielding to interesting results (Benkhoud and Bouallègue 2017).

5.7 Medical diagnosis

Interestingly, nature-inspired metaheuristics have found a plethora of applications aimed to develop tools to assist on medical diagnosis. Typical applications to this area include diagnostic applications based on the digital image processing medical images. In Ibrahim et al. (2012), for example, the authors proposed an approach to automatically measure ventricular heart volumes on cardiovascular magnetic resonance (CMR) images by applying an ACO-based approach that integrates iterative salient isolated thresholding (ACOISIT) to segment blood-myocardium borders and demonstrated promising results (Ibrahim et al. 2012). Furthermore, in Ouaddah and Boughaci (2016), the authors proposed a methodology to perform image reconstruction from projections based on the HS algorithm. In this approach, the HS algorithm is hybridized with a local search method to enhance its performance and improve the quality of images reconstructed from tomographic images tomographies. Both the original and hybrid HS, as well as two other traditional techniques implemented for image reconstruction, were compared regarding reconstruction quality, demonstrating, in the end, the proficiency of the proposed techniques (Ouaddah and Boughaci 2016). Later, in Chen (2017), an image segmentation approach based on the ACO algorithm is proposed for the detection of Lung Lesions in chest computed tomographies. To validate the proposed systems the authors analyzed its accuracy by considering a specific database of lung patients, obtaining favorable results (Chen 2017). Similarly, in Oliva et al. (2017), the authors proposed an image segmentation method based on both cross entropy thresholding and the CSA algorithm, with applications to the analysis of magnetic resonance brain images. Said approach was compared against cross entropy segmentation techniques based on DE and HS, ultimately demonstrating to be superior in terms of performance (Oliva et al. 2017). Other applications of nature-inspired methods to medical diagnosis include the development of tools aimed to aid on the analysis of specific medical data. As an example, we have the work reported in Wang et al. (2015), where an Improved Electromagnetism-like Algorithm (IEA) is proposed to develop a feature selection method for the prediction of diabetes mellitus. The proposed approach was tested by considering an extensive number of pertinent datasets, and its performance was compared with several other benchmark metaheuristic techniques reported in the literature, yielding to interesting results (Wang et al. 2015). Furthermore, in Kora and Kalva (2015), the authors developed and improved BA approach to extract features from Electrocardiogram (ECG) signals, with the purpose of feeding them to a neural network classifier trained for the prediction of myocardial infarction on heart patients. Both the improved and the standard BA algorithm were compared regarding performance by also considering other possible classifiers, including NN, KNN, SVM, and others (Kora and Kalva 2015). Also, in Nagpal et al. (2017), a feature selection approach based on GSA and k-nearest neighborhood classification is proposed for the task of efficient feature selection. According to the experimental results presented by the authors, the proposed method is able to reduce the number of significant features in the processed data to an average of 66%, while also showing a better performance than technique based on algorithms such as PSO and GA (Nagpal et al. 2017). Then, in Sahoo and Chandra (2017), a variant of the GWO algorithm known as Non-dominated Sorting GWO (NSGWO) and the Multi-Objective GWO (MOGWO) are proposed to address the problem of feature selection to aid on the classification/detection of cervix lesions. The authors compared their proposed feature selection methods with those based on several multi-objective implementations of GA and FA, demonstrating the GWO variants to be superior in all cases (Sahoo and Chandra 2017). Another interesting application is reported in Alshamlan et al. (2015), were the ABC algorithm is applied to aid on the task of gene selection for cancer classification using microarray datasets. The proposed

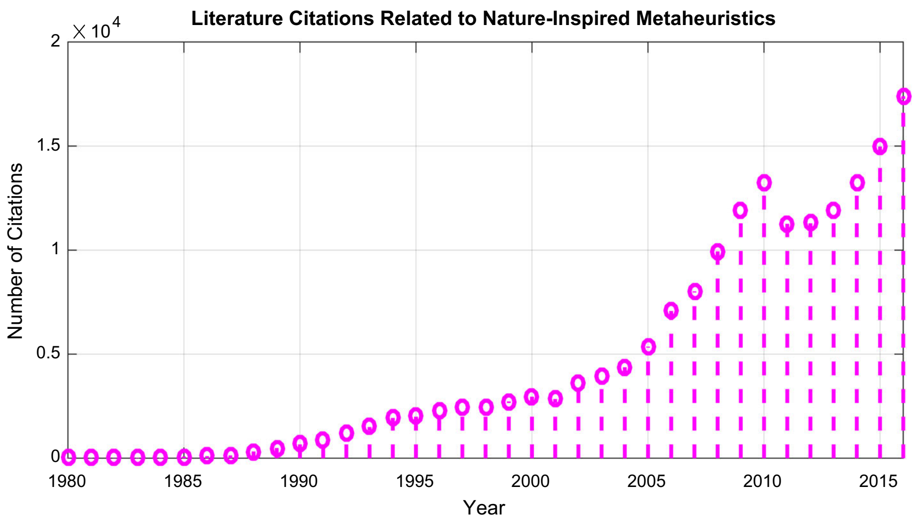

Fig. 2 Annual number of publications in IEEE and Elsevier related nature-inspired metaheuristics

approach combines a filtering stage based on a Minimum Redundancy Maximum Relevancy (mRMR) method and a wrapper method based in both BA and SVM, and it was compared with two other similar approaches based in GA (mRMR-GA) and PSO (mRMR-PSO), showing impressive results (Alshamlan et al. 2015). Similarly, in Alomari et al. (2017), a gene selection methodology based in BA was presented and compared with other popular gene selection methods, demonstrating to be more than competent for the task mentioned above.

6 Nature-inspired metaheuristics on the literature

In the last few years, works related to applications or modifications of nature-inspired metaheuristics for solving a wide range of optimization problems have become so numerous that mentioning every single paper in existence has become an overwhelming task. In Fig. 2, the number annual publications (as reported by IEEE and Elsevier) related to nature-inspired metaheuristics for the last 36 years is shown. As evidenced by the data shown in said figure, the number of publications related to this kind of techniques started to increase at a considerable rate around the 1990s. Nature-inspired optimization methods have become so successful and so well-known on the literature and, as a result, the number of publications per-year related to these techniques has reached astonishing levels, with essentially more than 1000 new papers being published annually since 2010. While nature-inspired techniques have been well received in the literature, some particular methods stand as popular choices among researchers around the world. In Fig. 3, the annual publication statistics (as given by IEEE and Elsevier publishers) related to the ten most cited nature-inspired optimization methods is shown. In particular, some of the earliest nature-inspired methods such as GA, SA, PSO, DE, and ACO stand as the most representative examples among these techniques, while latest methods such as ABC, HS, CS, FA, and GSA have recently experienced an increase on popularity. Furthermore, Table 2 shows specific data regarding each of these methods, including its year of publication and its total number of citations up to the year 2016.

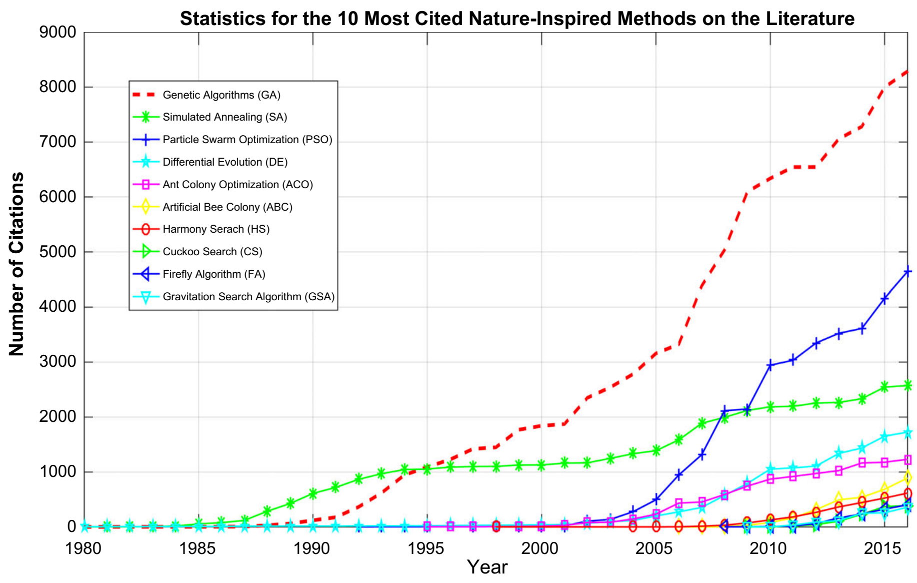

Fig. 3 Annual number of publications in IEEE and Elsevier related to the ten most cited nature-inspired metaheuristics: GA, SA, PSO, DE, ACO, ABC, HS, CS, FA, and GSA

7 Research gaps and future directions

Undoubtedly, research on nature-inspired metaheuristic algorithms and their applications has grown at an accelerated rate. However, while there exists an overwhelming amount of related works reported on the literature, this research area is yet to reach the maturity level that other areas of science currently have. There are still several research gaps and areas of opportunity that are still to be explored by researchers on this rather young area of science. One of this areas of opportunity arise due to the fact that, to this date, there is no single metaheuristic optimization algorithm with the ability to effectively handle all existing problems (Wolpert and Macready 1997); in fact, the literature suggest that there exist several techniques that perform significantly better than other methods when applied to specific problems. In this regard, we can mention the widely known 0–1 knapsack optimization problem (Vocking et al. 2011), on which numerous optimization techniques have been successfully applied with good results; in the case of the work presented in Sapra et al. (2017), for example, a comparative study of several metaheuristic algorithms applied to solve the knapsack problem is presented, where methods such as Tabu Search (Pardalos et al. 2013), Scatter Search (Laguna and Martí 2003) and Local Search (Galinier et al. 2013) are the center of discussion. The experimental results presented in this work suggest that Tabu Search has the least deviation from the best known solution, suggesting it to be more reliable in this regard. However, this work also suggest that Scatter Search presents the least time complexity, being the best option among the three tested methods when execution time is crucial (Sapra et al. 2017). Another example is given by Feng et al. (2017), where a novel Binary Monarch Butterfly Optimization (BMBO) algorithm was proposed to solve this problem, and was further validated by comparing it with binary-coded implementations of ABC, CS, DE, and GA, demonstrating BMBO to have greater accuracy and convergence speed (Feng et al. 2017). Other commonly studied optimization problem is represented by the famous Traveling Salesman Problem (TSP)

Table 2 Total number of citations (up to the year 2016) for some of the most popular nature-inspired metaheuristics

Algorithm	Author(s)	Year of publication	Total number of citations
Genetic Algorithms (GA)	J. H. Holland	1960	102,190
Simulated Annealing (SA)	S. Kirkpatrick, C.D. Gelatt & M.P. Vecchi	1982	44,711
Particle Swarm Optimization (PSO)	J. Kennedy & R. Eberhart	1995	38,634
Differential Evolution (DE)	R. Storn & K. Price	1996	38,632
Ant Colony Optimization (ACO)	M. Dorigo	1992	14,472
Artificial Bee Colony (ABC)	D. Karaboga	2005	4174
Harmony Search (HS)	Z. W. Geem, J. H. Kim & G. V. Loganathan	2001	3466
Cuckoo Search (CS)	X.-S. Yang & S. Deb	2009	1843
Firefly Algorithm (FA)	X.-S. Yang	2008	1807
Gravitational Search Algorithm (GSA)	E. Rashedi, H. Nezamabadi-pour & S. Saryazdi	2009	1639

(Gutin and Punnen 2007). An interesting study can be found in the work present by Saji and Riffi (2016), where a comparison between a novel Discrete Bat Algorithm (DBA) and discrete coded modifications for PSO, CS, and GSA-ACS-PSOT (a hybrid approach based on several metaheuristics) for solving the TSP is presented, demonstrating DBA to be the overall best method to handle this problem (Saji and Riffi 2016). Similarly, in Zhou et al. (2017a, b), the authors propose to solve several spherical TSP by applying a discrete greedy Flower Pollination Algorithm (DGFPA), and compared it with several variants of GA and Tabu Search, finally concluding that DGFPA performs the best in most cases. Finally, we have the vehicle routing problem (VRP) (Pereira and Tavares 2009), a problem that nowadays could be considered a benchmark real-problem for validating the performance of optimization algorithms. While there are several variants to this particular problem, nature-inspired metahueristic methods have been extensively applied for solving each of these (Yurtkuran and Emel 2010; Wei et al. 2017; Marinaki and Marinakis 2016; Potvin 2009); in particular, it is common to see implementations to this kind of problems based on enhanced/improved implementations of well-known bio-inspired algorithms. One example of this is reported in Xu et al.(2018), where Dynamic VRP is handled by applying an Enhanced ACO algorithm (E-ACO). The proposed method was compared in terms of performance against the standard ACO algorithm and another improved variant known as K-means ACO (K-ACO), concluding E-ACO to be slightly superior (Xu et al. 2018). Similarly, In Zhang and Lee (2015), an Improved ABC algorithm is applied to solve Capacitated VRP and was compared in terms of performance against the standard ABC approach, yielding to a superior performance. Another example is presented in Xiang (2016), where an improved PSO algorithim (NPSO) which integrates Gauss Mutation is developed for solving the VRP and compared in terms of performance with the original PSO, virtually outperforming it in both performance and efficiency. From what was previously discussed, it must be concluded that the different degrees of performance on nature-inspired metaheuristics over certain problems is not only heavily influenced by the specific search mechanisms and "adaptations" applied by each applied method, but also by the particular challenges offered by each of these problems, hence there is no way to establish a particular method as the absolute best, and as such there is still a plenty of space for proposing new and innovative methodologies for solving these problems or to modify existing ones with the purpose of enhancing their performance and efficiency. In this sense, many researchers have proposed interesting ideas aimed to improve the performance of nature-inspired metaheuristics. Some researchers, for example, suggest that a hybridization of exact methods and metaheuristic-based techniques could lead to the development of algorithms with enhanced efficiency and convergence capabilities (Jourdan et al. 2009; Puchinger and Raidl 2005). While research in this regard is still somewhat limited, there are several works on the literature which serve as examples of successful applications of both kind of methods (Plateau et al. 2002; Yan et al. 2014; Portmann et al. 1998; Basseur et al. 2004; Gomes and Oliveira 2006). Other interesting ideas can also be found on the work presented by Zelinka (2015), were the author explore the possibility of improving that performance and diversity of search operators by implementing specific control mechanisms (such as deterministic chaos models) to alter the dynamics of swarm and evolutionary algorithms. An example of a metaheuristic-based application that implement this kind of mechanism is presented in Valdivia-Gonzalez et al. (2017a, b), where different chaos models were combined with the search operators of the GSA algorithm in order to enhance its performance. Similarly, in Cuevas et al. (2017a, b), deterministic chaos models where integrated to the search operators of novel swarm-based algorithm known as Locust Search (LS) (Cuevas et al. 2015a, b; González et al. 2017), with the purpose enhancing its performance for the identification of parameters in fractional-order systems (Cuevas et al. 2017b). Also, in Hinojosa et al. (2018), a Multi-Objective implementation of the CSA approach (MOCSA) was modified to integrate chaotic behaviors in order to enhance its solution diversity, demonstrating a notable increase on performance. In the last few years, techniques aimed to improve the efficiency of metaheuristic optimization algorithms by implementing surrogated models are gaining popularity among researchers from this area of science. In Regis (2013), for example, the author proposed a surrogated-assisted optimization framework based on PSO coined Optimization by particle swarm Using Surrogates (OPUS) with the purpose of efficiently solving high-dimensional black-box optimization problems, and implemented it to solve several real world problems, including groundwater bioremediation and watershed calibration. Similarly, in Liu et al. (2016), an optimization approach based on Differential Evolution which implement Gaussian Process (GP) regressions in conjunction with Optimization by Radial Basis functions Interpolation in Trust-regions (ORBIT) (Wild et al. 2008) was developed to efficiently and reliably solve optimization problems. The performance of this method, called Multi-fidelity Gaussian Process and radial basis Memetic Differential Evolution (MGPMDE), was validated by considering a wide set of benchmark test functions, as well as with its applications for data mining (Liu et al. 2016). While all of the previous suggests that there are still plenty of areas of opportunity for the development and application of nature-inspired metaheuristic optimization algorithms, perhaps the greatest gap in this area of science is the absence of theoretical foundations that allows to objectively analyze important characteristics of these techniques, such as convergence rate and efficiency. While there is a general agreement in that the good performance of nature-inspired algorithms is attributed to a proper balance between exploration and exploitation of solutions, the truth is that there is barely a clear definition of what these two concepts truly represent. In fact, classifying the search operators and strategies applied by nature-inspired methods is often an ambiguous task, as many of these seems to contribute in some way to both the exploration and exploitation of solutions, and even then, there is currently no clear way to objectively measure the rate of exploration/exploitation provided by these operators (Črepiňsek et al. 2013). In the absence of mathematical analysis methods that could be applied to measure these properties, the performance of nature-inspired metaheuristics is mostly evaluated by applying ad-hoc methodologies based on the quantitative analysis of certain validation metrics such as error, mean, median, standard deviation, and so on (Boussaïd et al. 2013). A recent attempt to measure these factors over the most popular swarm metaheuristic is presented in (Ghazali et al. 2018). They proposed a modified equation from (Cheng et al. 2014) that calculates the diversity of the population using the distance between the solutions in the search space. This diversity measurement calculates the averaged sum distance between all the solution to the median of each dimension and solutions. Then, the exploration percent of each iteration is calculated by dividing the diversity value between the maximum diversity value encounter in the process, and its inverse is considered as the exploitation rate. However, this idea is oversimplified, as it doesn't provide any information related to the optimization landscape itself, thus making it impossible to get objective conclusions in this regard. Also, it is of common knowledge that the performance of these methods is often evaluated over well-known benchmark test functions, developed to represent in some way important characteristics of real-world problems such as search space scale and imbalance. However, there is no theoretical evidence to prove if these test problems truly reflect on this important characteristics, and as such, these performance evaluation frameworks are often criticized as well (Yang 2011, 2015). Finally, while the absence of theoretical foundations for proving the prowess of metaheuristic algorithms is a major issue, it is also worth noting that the absence of standardized frameworks for the implementation and comparison of this kind algorithms is virtually absent as well (Boussaïd et al. 2013); this means that most of the time, researchers are forced to implement algorithms coded by other researchers or, in the worst case scenario, code the algorithm themselves. Whichever the case, the main problem here is the fact that algorithm coded by different persons tend to be somewhat different in terms of implementation and by extension they also tend to manifest different degrees of performance and efficiency (even in those algorithms meant to represent the same optimization technique); with that being said, for nature-inspired algorithms to be implemented and compared in fair terms, software platforms devised to allow the implementation and evaluation of this methods, all in the same terms, are also necessary.

8 Conclusions and final thoughts

Nature is often praised by researchers as the perfect example of adaptive problem solving, and as such, it is not surprising to see why metaheuristics optimization algorithms inspired in natural phenomena have become so popular. These kinds of methods are designed with the idea of mimicking some biological or physical phenomenon observed in nature with the purpose of developing powerful tools that could be applied to solve optimization problems. The main advantage of nature-inspired metaheuristics over traditional optimization methods, however, lies on their ability to handle a wide variety of problems independently of their structure and properties. Due to this distinctive trait, these methods have become popular choices for solving otherwise complex problems. As a result, these techniques have found application in virtually every single area of science, including robotics, computer networks, security, engineering design, data mining, finances, economics, and many others. In the last few years, literature related to nature-inspired algorithms and its applications for solving optimization problems has experienced an almost exponential increase, with tons of new papers being published every year. Methods such as GA, SA, PSO, DE, and ACO are among the most successful and most cited optimization approaches currently reported on the literature that are widely applied to solve numerous real-world problems. While there are still several research gaps that remain to be explored for this area of science to reach maturity, these have served as inspiration to researchers for the development of better techniques, suited to solve an ever-increasing amount of complex real-world problems. In any case, there is no doubt that nature-inspired metaheuristics have rightfully earned their place as powerful tools for optimization, and as such, this line of investigation is expected to keep growing in the near future.

Appendix

In Table 3, we present a list comprised of several nature-inspired metaheuristics currently proposed on the literature. A total of 168 different algorithms, along with their respective abbreviations, authors and its year of proposal, have been documented for this table (note that some of the algorithms abbreviations might be repeated). Further information about these methods may be found in (Reyna et al. 2017).

Table 3 List of nature-inspired metaheuristics

Algorithm	Abbreviation	Authors	Year
Amoeboid Organism Algorithm	AOA	Y. J. Zhang, Z.L. Zhang & Y. Deng	2011
Ant Colony Optimization	ACO	M. Dorigo	1992
Ant Lion Optimizer	ALO	S.Mirjalili	2014
Artificial Bee Colony	ACO	D. Karaboga & B. Basturk	2007
Artificial Beehive Algorithm	ABA	M. A. Muñoz, J. A. López & E. Caicedo	2009
Artificial Chemical Process	LARES	R. Irizarry	2005
Artificial Cooperative Search Algorithm	ACS	P. Civicioglu	2013
Artificial Fish Swarm Algorithm	AFSA	X. Li et al.	2002
Artificial Immune System	AIS	J.D. Farmer, N. Packard & A. Perelson	1986
Artificial Physics Optimization	APO	L. Xie, J.ianchao Zeng & Z. Cui	2009
Artificial Reaction Algorithm	ARA	L. Astudillo	2013
Artificial Searching Swarm Algorithm	ASSA	T. Chen	2009
Artificial Swarm Intelligence	ASI	L. Rosenberg	2014
Artificial Tribe Algorithm	ATA	T. Chen, Y.Wang & J. Li	2012
Atmosphere Clouds Model Optimization	ACMO	Y. Gao-Wei & H. Zhanju	2012
Backtracking Search Algorithm	BSA	P. Civicioglu	2013
Bacterial Colony Chemotaxis	BCC	S.D. Muller, J. Marchetto, S. Airaghi & P. Koumoutsakos	2002
Bacterial Colony Optimization	BCO	B. Niu & H. Wang	2012
Bacterial Foraging Algorithm	BFA	Kevin M. Passino	2002
Bar Systems	BS	E. Del-Acebo & J. L. De-la Rosa	2008

Table 3 continued

Algorithm	Abbreviation	Authors	Year
Bat Algorithm	BA	X.S. Yang	2010
Bat Intelligence	BI	B. Malakooti, H. Kim & S. Sheikh	2012
Bean Optimization Algorithm	BeOA	X. Zhang	2010
Bee Colony Optimization	BCO	D. Teodorovic, P. Lucic, G. Markovic & M. D. Orco	2006
Bee Colony-inspired Algorithm	BCiA	S. Häckel & P. Dippold	2009
Bee Swarm Optimization	BSO	R. Akbari, S. A. Mohammadi & K. Ziarati	2009
Bee System	BS	T. Sato & M. Hagiwara	1997
Bees Algorithm	BA	D.T. Pham, A. Ghanbarzadeh, et al.	2006
Bees Life Algorithm	BLA	S. Bitam	2012
Bees Optimization	BO	S. Nakrani & C. Tovey	2004
Big Bang–Big Crunch	BB–BC	O. K. Erol & I. Eksin	2006
Biogeography-based Optimization	BBO	D. Simon	2008
Bioluminescent Swarm Optimization	BSO	D. R. de Oliveira, R. S. Parpinelli & H. S. Lopes	2011
Bionic Optimization	BO	R. Steinbuch & S. Gekeler	2011
Black Hole Algorithm	BHA	A. Hatamlou	2013
Blind, Naked Mole-Rats	BNMR	M. Taherdangkoo, M. H. Shirzadi & M. H. Bagheri	2012
Brain Storm Optimization	BSO	Y. Shi	2011
Bumblebees Algorithm	BA	F.P. Comellas-Padró & J. Martínez-Navarro	2009
Cat Swarm Optimization	CSO	S. C. Chu, P. Tsai & J.S. Pan	2006
Central Force Optimization	CFO	R. A. Formato	2009
Chaos Optimization Algorithm	COA	B. Li & W. S. Jiang	1998
Charged System Search	CSS	A. Kaveh & S. Talatahari	2010
Chemical-Reaction Optimization Algorithm	CRO	A. Y. S. Lam	2010
Clonal Selection Algorithm	CSA	L.N. de Castro & F. V. Zuben	2000
Cloud Model-Based Differential Evolution	CMDE	C. Zhu & J. Ni	2012
Cockroach Swarm Optimization	CSO	L. Cheng	2010
Collective Animal Behaviour	CAB	E. Cuevas, M. González, D. Zaldivar, M. Pérez-Cisneros & G. García	2012
Cricket Behaviour-based Evolutionary	CBBE	M. Canayaz & A. Karci	2016
Cuckoo Optimization Algorithm	COA	R. Rajabioun	2011
Cuckoo Search	CS	X.S. Yang & S. Deb	2009
Cultural Algorithm	CA	R. G. Reynolds	1994
Cuttlefish Optimization Algorithm	CFA	A. S. Eesa, Z. Orman & A. M. Abdulazeez-Brifcani	2014
Differential Evolution	DE	R. Storn & K. Price	1997
Differential Search Algorithm	DSA	P. Civicioglu	2012
Dove Swarm Optimization	DSO	Su, et al.	2009

Table 3 continued

Algorithm	Abbreviation	Authors	Year
Dragonfly Algorithm	DA	Se. Mirjalili	2016
Eagle Strategy	ES	X.S. Yang & S. Deb	2010
Electromagnetism-like Mechanism	EM	S. Ilker Birbil & S. C.Fang	2003
Elephant Herding Behaviour	EHO	G.G.Wang, S. Deb & L. S. Coelho	2015
Evolution Strategies	ES	I. Rechenberg & H. P. Schwefel	1964
Extremal optimization	EO	S. Boettcher & A. Percus	2000
Firefly Algorithm	FA	X.S. Yang	2008
Fireworks Algorithm	FA	Y. Tan & Y. Zhu	2010
Fish School Search	FSS	C. J. A. Bastos-Filho, F. B. De-Lima-Neto, A. J. C. C. Lins, et al.	2009
Flock by Leader	FL	A. Bellaachia & A. Bari	2012
Flocking-based Algorithm	FBA	X. Cui, J. Gao & T. E. Potok	2006
Flower Pollination Algorithm	FPA	X.S. Yang	2012
Football Game Inspired Algorithm	FGIA	E. Fadakar & M. Ebrahimi	2016
Frog Calling Algorithm	FCA	A. Mutazono, M. Sugano & M. Murata	2009
Fruit Fly Optimization Algorithm	FFOA	W.T. Pan	2012
Galaxy-based Search Algorithm	GbSA	H. Shah-Hosseini	2011
Gases Brownian Motion Optimization	GBMO	M. Abdechiri, M. R. Meybodi & H. Bahrami	2013
Genetic Algorithm	GA	J. H. Holland	1975
Glowworm Swarm Optimization	GSO	K. N. Krishnanand & D. Ghose	2009
Goose Optimization Algorithm	GTO	J. Wang & D. Wang	2008
Gravitational Clustering Algorithm	GCA	S. Kundu	1999
Gravitational Emulation Local Search	GELS	B. Barzegar, A. M. Rahmani, K. Zamanifar & A. Divsalar	2009
Gravitational Field Algorithm	GFA	M. Zheng, G. Liu, C. Zhou, et al.	2010
Gravitational Interactions Optimization	GIO	J. J. Flores, R. López & J.Barrera	2011
Gravitational Search Algorithm	GSA	E. Rashedi, H. Nezamabadi-pour & S. Saryazdi	2009
Great Deluge Algorithm	GDA	G. Dueck	1993
Grenade Explosion Method	GEA	A. Ahrari & A. A. Atai	2010
Grey Wolf Optimizer	GWO	S. Mirjalili, S. M. Mirjalili and A. Lewis	2014
Group Escaping Algorithm	GEA	H. Min & Z. Wang	2011
Group Leaders Optimization Algorithm	GLOA	A. Daskin & S. Kais	2011
Group Search Optimizer	GSO	S. He, Q. H. Wu & J. R. Saunders	2009
Harmony Elements Algorithm	HEA	Y. H. Cui	2009
Harmony Search	HS	Z. W. Geem, J. H. Kim, & G. V. Loganathan	2001
Honey Bee Behavior	HBB	H.F. Wedde, M. Farooq & Y. Zhang	2004
Honey-bee Mating Optimization	HBMO	O. B. Haddad, A. Afshar & M. A. Mariño	2006
Honeybee Social Foraging	HSF	N. Quijano, K.M. Passino	2007
Human Group Formation	HGF	A. Thammano & J. Moolwong	2010

Table 3 continued

Algorithm	Abbreviation	Authors	Year
Hunting Search	HuS	R. Oftadeh, M.J.Mahjoob & M.Shariatpanahi	2010
Hysteretic Optimization	HO	G. Zaránd, F. Pázmándi, K. F. Pál, G. T. Zimányi	2002
Immune-Inspired Computational Intelligence	ICI	P. Cortés, J. M. García, L.Onieva, et al.	2008
Imperialist Competitive Algorithm	ICA	E. A. Gargari & C. Lucas	2007
Integrated Radiation Optimization	IRO	C.L. Chuang & J.A. Jiang	2007
Intelligent Water Drops	IWD	H. Shah-Hosseini	2008
Invasive Weed Optimization	IWO	A.R. Mehrabian & C. A Lucas	2006
Krill Herd	KHA	A. H. Gandomi & A. H. Alavi	2012
Leaders and Followers Algorithm	LFA	Y. Gonzalez-Fernandez & S. Chen	2015
League Championship Algorithm	LCA	A. H. Kashan	2009
Light Ray Optimization	LRO	J. H. Shen & J. L. Li	2010
Magnetic Optimization Algorithm	MFO	M. H. Tayarani & M. R. Akbarzadeh	2008
Melody Search	MS	S. M. Ashrafi	2011
Membrane Algorithm	MA	T. Y. Nishida	2005
Method of Musical Composition	MMC	R.A. Mora-Gutiérrez, J. Ramírez-Rodríguez & E.A. Rincón-García	2014
Migrating Birds Optimization	MBO	E. Duman, M. Uysal & A. F. Alkaya	2012
Mine Blast Algorithm	MBA	A. Sadallah, A. Bahreininejad & M. Hamdi	2012
Monkey Search Algorithm	MS	A. Mucherino & O. Seref	2007
Mosquito Host-Seeking Algorithm	GMHSA	X. Feng, X. Liu & H. Yu	2008
Moth-flame Optimization Algorithm	MFO	S. Mirjalili	2015
Multi-Verse Optimizer	MVO	S. Mirjalili, S. M. Mirjalili & A. Hatamlou	2016
Natural Aggregation Algorithm	NAA	F. Luo, Z.Y. Dong, Y. Chen & J. Zhao	2016
OptBees	OB	R. D. Maia, L. De-Castro & W.M. Caminhas	2013
Optics Inspired Optimization	OIO	A. H. Kashan	2015
Oriented Search Algorithm	OSA	X.Zhang, W. Chen	2008
Paddy Field Algorithm	PFA	U. Premaratne, J. Samarabandu & T. Sidhu	2009
Particle Collision Algorithm	PCA	W. F. Sacco & C. R.E. de Oliveira	2005
Particle Swarm Optimization	PSO	J. Kennedy & R. C. Eberhart	1995
Photosynthetic Algorithm	PA	H. Murase & A. Wadano	1999
PoPMuSiC Algorithm	PoPMuSiC	E. Taillard & S. VoB	1999
Population Migration Algorithm	PMA	Y. Zhang	2009
Prey-Predator Algorithm	PPA	S. L. Tilahun & H. C. Ong	2015
Ray Optimization Algorithm	ROA	A. Kaveh	2012
River Formation Dynamics Algorithm	RFDA	P. Rabanal, I. Rodríguez & F. Rubio	2007
Roach infestation Optimization	RIO	T. C. Havens, C. J. Spain, N. G. Salmon & J. M. Keller	2008

Table 3 continued

Algorithm	Abbreviation	Authors	Year
Sapling Growing Up Algorithm	SGA	A. Karci	2007
Search group algorithm	SGA	M. S. Gonçalves, R. H.Lopez & L. F. Fadel-Miguel	2015
Seeker Optimization Algorithm	SOA	C. Dai, Y. Zhu & W. Chen	2006
Self-defense Techniques of the Plants	SDTP	C. Caraveo, F. Valdez, O. Castillo & P. Melin	2016
Self-Organizing Migrating Algorithm	SOMA	I. Zelinka	2004
Shark Smell Optimization	SSO	O. Abedinia, N. Amjady & A. Ghasemi	2014
Shark-Search Algorithm	SSA	M.Hersovici, M. Jacovi, et al.	1998
Sheep Flock Heredity Mode	SFHM	K. Nara, et al.	1999
Shuffled Frog Leaping Algorithm	SFLA	M. E. Muzaffar & K. E. Lansey	2003
Simple Optimization	SOPT	O. Hasancebi, K.S. Azad	2012
Simulated Annealing	SA	S. Kirkpatrick, C. D. Gelatt & M. P. Vecchi	1983
Simulated Bee Colony	SBC	J. D. McCaffrey	2009
Sine Cosine Algorithm	SCA	S. Mirjalili	2016
Slime Mould Algorithm	SMA	M. Shann	2008
Social Cognitive Optimization Algorithm	SCOA	Z. Wei, Z. Cui & J.C. Zeng	2010
Social Spider Algorithm	SSA	James J.Q. Yu & Victor O.K. Li	2015
Social Spider Optimization Algorithm	SSO	E. Cuevas, M. Cienguegos, D. Zaldivar & M. Perez	2013
Society and Civilization Algorithm	SCA	T. Ray & K. M. Liew	2003
Space Gravitational Optimization	SGO	Y.T. Hsiao, C.L. Chuang, J.A. Jiang & C.C. Chien	2005
Spiral Optimization Algorithm	SpOA	G. G. Jin	2010
States of Matter Search	SMS	E. Cuevas, A. Echavarría and M.A. Ramírez-Ortegón	2014
Stem Cells Algorithm	SCA	M. Taherdangkoo, M. Yazdi & M. H. Bagheri	2012
Stochastic Focusing Search	SFS	W. Weibo, F. Quanyuan & Z. Yongkang	2008
Superbug Algorithm	SuA	C. Anandaraman, A. V. M. Sankar & R.Natarajan	2012
Swallow Swarm Optimization	SSO	M. Neshat, G. Sepidnam & M. Sargolzaei	2013
Tabu Search	TS	F. Glover	1986
Teaching–learning-based Optimization	TLBO	R.V. Rao, V.J.Savsani & D.P.Vakharia	2011
Termite-hill Algorithm	TA	A. M. Zungeru, L.M. Ang & K. P. Seng	2012
Unconscious Search	US	E. Ardjmand & M. R. Amin-Naseri	2012
Virtual Bees Algorithm	VBA	X.S. Yang	2005
Virus Optimization Algorithm	VOA	Y.C. Liang & J. R. Cuevas-Juarez	2009
Vortex Search algorithm	VS	B. Dogan & T. Ölmez	2015
Wasp Swarm Optimization	WSO	G. Theraulaz, S. Goss, J. Gervet, and J. L. Deneubourg	1991

Table 3 continued

Algorithm	Abbreviation	Authors	Year
Water Cycle Algorithm	WCA	H. Eskandar, A. Sadollah, A. Bahreinineja & M. H. Abd-Shukor	2012
Water Flow Algorithm	WFA	S. Basu, C. Chaudhuri, M. Kundu & M. Nasipuri	2007
Water Flow-like Algorithm	WFA	F.C. Yang & Y.P. Wang	2007
Water Wave Optimization	WWO	Y.J. Zheng	2015
Whale Optimization Algorithm	WOA	S. Mirjalili & A. Lewis	2016
Wisdom of Artificial Crowds	WAC	R. V. Yampolskiy, L. H. Ashby & L. Hassan	2011
Wolf Pack Search	WPS	C. Yang, X. Tu & J. Chen	2007

References

Abualigah LM, Khader AT (2017) Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering. J Supercomput 73(11):4773–4795

Abualigah LM, Khader AT, Al-Betar MA, Awadallah MA (2016) A krill herd algorithm for efficient text documents clustering. In: ISCAIE 2016—2016 IEEE symposium on computer applications & industrial electronics, pp 67–72

Abualigah LM, Khader AT, Hanandeh ES, Gandomi AH (2017a) A novel hybridization strategy for krill herd algorithm applied to clustering techniques. Appl Soft Comput J. 60:423–435

Abualigah LM, Khader AT, Al-Betar MA, Alomari OA (2017b) Text feature selection with a robust weight scheme and dynamic dimension reduction to text document clustering. Expert Syst Appl 84:24–36

Al-Betar MA, Awadallah MA, Abu Doush I, Alsukhni E, ALkhraisat H (2018) A non-convex economic dispatch problem with valve loading effect using a new modified β-Hill Climbing Local Search Algorithm. Arab J Sci Eng 43:7439–7456

Alia OM, Al-Ajouri A (2017) maximizing wireless sensor network coverage with minimum cost using Harmony Search Algorithm. IEEE Sens J 17(3):882–896

Alomari OA, Khader AT (2017) MA Al Betar, and LM Abualigah (2017) Gene selection for cancer classification by combining minimum redundancy maximum relevancy and bat-inspired algorithm. Int J Data Min Bioinform 19(1):32

Alshamlan H, Badr G, Alohali Y (2015) MRMR-ABC: a hybrid gene selection algorithm for cancer classification using microarray gene expression profiling. Biomed Res. Int. 2015:1–15

Alswaitti M, Albughdadi M, Isa NAM (2018) Density-based particle swarm optimization algorithm for data clustering. Expert Syst Appl 91:170–186

Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: Crow Search Algorithm. Comput Struct 169:1–12

Askarzadeh A, Rezazadeh A (2012) Parameter identification for solar cell models using harmony search-based algorithms. Sol Energy 86(11):3241–3249

Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: 2007 IEEE congress on evolutionary computation, CEC 2007, pp 4661–4667

Auger A, Schoenauer M, Vanhaecke N (2004) {LS-CMA-ES}: a second-order algorithm for covariance matrix adaptation. Parallel Probl Solving Nat PPSN VIII 3242(1):182–191

Avigad J, Donnelly K (2004) Formalizing O notation in Isabelle/HOL. Springer, Berlin, pp 357–371

Babu TS, Ram JP, Dragicevic T, Miyatake M, Blaabjerg F, Rajasekar N (2017) Particle Swarm Optimization based solar PV array reconfiguration of the maximum power extraction under partial shading conditions. IEEE Trans Sustain Energy 9:74–85

Back T, Hoffmeister F, Schwefel HP (1991) A survey of evolution strategies. In: Proceedings of the fourth international conference on genetic algorithms, vol 9, p 8

Bäck T, Foussette C, Krause P (2013) Contemporary evolution strategies, vol 47. Springer, Berlin

Basseur M, Lemesre J, Dhaenens C, Talbi EG (2004) Cooperation between branch and bound and evolutionary approaches to solve a bi-objective flow shop problem, vol 2632. Springer, Berlin

Behnck LP, Doering D, Pereira CE, Rettberg A (2015) A modified simulated annealing algorithm for SUAVs path planning. IFAC-PapersOnLine 28(10):63–68

Bekdaş G, Nigdeli SM, Yang XS (2015) Sizing optimization of truss structures using flower pollination algorithm. Appl Soft Comput J 37:322–331

Benkhoud K, Bouallègue S (2017) Dynamics modeling and advanced metaheuristics based LQG controller design for a Quad Tilt Wing UAV. Int J Dyn Control 6(2):630–651

Beyer HG, Sendhoff B (2008) Covariance matrix adaptation revisited—the CMSA evolution strategy. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol 5199, LNCS, pp 123–132

Bhardwaj T, Sharma TK, Pandit MR (2014) Social engineering prevention by detecting malicious URLs using Artificial Bee Colony Algorithm. In: 3rd international conference on soft computing for problem solving, advances in intelligent systems, pp 355–363

Binitha S, Sathya SS (2012) A survey of bio inspired optimization algorithms. Int J Soft Comput Eng 2(2):137–151

Birbil ŞI, Fang SC (2003) An electromagnetism-like mechanism for global optimization. J Glob Optim 25(3):263–282

Boussaïd I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inf Sci (Ny) 237:82–117

Burke EK et al (2013) Hyper-heuristics: a survey of the state of the art. J Oper Res Soc 64(12):1695–1724

Camarena O, Cuevas E, Pérez-cisneros M, Fausto F, González A, Valdivia A (2018) Ls-II: an improved locust search algorithm for solving constrained optimization problems

Cao S, Wang J, Gu X (2012) A wireless sensor network location algorithm based on Firefly Algorithm. Asia Simul Conf 2012:18–26

Cavazzuti M (2013) Optimization methods: from theory to design. Springer, Berlin

Chen C (2017) Image segmentation for lung lesions using ant colony optimization classifier in chest CT. In: Advances in intelligent information hiding and multimedia signal processing, pp 283–289

Cheng S, Shi Y, Qin Q, Ting TO, Bai R (2014) Maintaining population diversity in brain storm optimization algorithm. In: Proceedings 2014 IEEE congress on evolutionary computation CEC 2014, pp 3230–3237

Contreras-Cruz MA, Lopez-Perez JJ, Ayala-Ramirez V (2017) Distributed path planning for multi-robot teams based on artificial bee colony. In: Proceeding on IEEE congress on evolutionary computation CEC 2017 pp 541–548

Črepiňsek M, Liu S-H, Mernik M (2013) Exploration and exploitation in evolutionary algorithms: a survey. ACM Comput Surv 45(33):1–33

Cuevas E, Cienfuegos M, Zaldívar D, Pérez-cisneros M (2013a) A swarm optimization algorithm inspired in the behavior of the social-spider. Expert Syst Appl 40(16):6374–6384

Cuevas E, Echavarría A, Ramírez-Ortegón MA (2013b) An optimization algorithm inspired by the states of matter that improves the balance between exploration and exploitation. Appl Intell 40(2):256–272

Cuevas E, González A, Fausto F, Zaldívar D, Pérez-Cisneros M (2015a) An optimisation algorithm based on the behaviour of locust swarms. Int. Bio Inspir Comput 7(6):402

Cuevas E, González A, Fausto F, Zaldívar D, Pérez-Cisneros M (2015b) Multithreshold segmentation by using an algorithm based on the behavior of locust swarms. Math Probl Eng 2015:26

Cuevas E, Díaz Cortés MA, Oliva Navarro DA (2016) Advances of evolutionary computation: methods and operators, 1st edn. Springer, Berlin

Cuevas E, Osuna V, Oliva D (2017a) Evolutionary computation techniques: a comparative perspective, vol 686. Springer, Berlin

Cuevas E, Gálvez J, Avalos O (2017b) Parameter estimation for chaotic fractional systems by using the locust search algorithm. Comput y Sist 21(2):369–380

Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31

Das S, Mullick SS, Suganthan PN (2016) Recent advances in differential evolution—an updated survey. Swarm Evol Comput 27:1–30

Deif DS, Member S, Gadallah Y, Member S (2017) "An Ant Colony Optimization approach for the deployment of reliable wireless sensor networks. IEEE Access 5:10744–10756

Díaz-Cortés M-A, Cuevas E, Rojas R (2017) Engineering applications of soft computing. Springer, Berlin

Din M, Pal SK, Muttoo SK, Jain A (2016) Applying Cuckoo Search for analysis of LFSR based cryptosystem. Perspect Sci 8:435–439

Dorigo M, Blum C (2005) Ant colony optimization theory: a survey. Theor Comput Sci 344(2–3):243–278

Dorigo M, Stützle T (2004) Ant colony optimization. Springer, Berlin

Du H, Wang Z, Zhan WEI (2018) Elitism and distance strategy for selection of evolutionary algorithms. IEEE Access 6:44531–44541

El Aziz MA, Ewees AA, Hassanien AE (2017) Whale Optimization Algorithm and Moth-Flame Optimization for multilevel thresholding image segmentation. Expert Syst Appl 83:242–256

Emary E, Zawbaa HM, Hassanien AE (2016) Binary grey wolf optimization approaches for feature selection. Neurocomputing 172:371–381

Eskandar H, Sadollah A, Bahreininejad A, Hamdi M (2012) Water cycle algorithm—a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 110–111:151–166

Feng Y, Wang GG, Deb S, Lu M, Zhao XJ (2017) Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization. Neural Comput Appl 28(7):1619–1634

Galinier P, Hamiez JP, Hao JK, Porumbel D (2013) Handbook of optimization, vol 38. Springer, Berlin

Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845

Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68

Gerules G, Janikow C (2016) A survey of modularity in genetic programming. In: 2016 IEEE congress on evolutionary computation CEC 2016, pp 5034–5043

Ghazali R, Deris MM, Nawi NM, Abawajy JH (2018) Recent advances on soft computing and data mining, vol 700. Springer, Berlin

Gomes AM, Oliveira JF (2006) Solving Irregular Strip Packing problems by hybridising simulated annealing and linear programming. Eur J Oper Res 171(3):811–829

González A, Cuevas E, Fausto F, Valdivia A, Rojas R (2017) A template matching approach based on the behavior of swarms of locust. Appl Intell 47(4):1087–1098

Goudos SK (2017) Antenna design using binary differential evolution. In: IEEE antennas and propagation magazine

Goyal S, Patterh MS (2015) Performance of BAT algorithm on localization of wireless sensor network. Wirel Pers Commun 6(3):351–358

Guerrero M, Montoya FG, Baños R, Alcayde A, Gil C (2017) Adaptive community detection in complex networks using genetic algorithms. Neurocomputing 266:101–113

Guha D, Roy PK, Banerjee S (2016) Load frequency control of interconnected power system using grey Wolf Optimization. Swarm Evol Comput 27:97–115

Gutin G, Punnen AP (2007) The traveling salesman problem and its variations. Springer, US

Han W, Wang H, Chen L (2014) Parameters identification for photovoltaic module based on an Improved Artificial Fish Swarm Algorithm

Han X, Quan L, Xiong X, Almeter M, Xiang J, Lan Y (2017) A novel data clustering algorithm based on modified gravitational search algorithm. Eng Appl Artif Intell 61:1–7

Harman M, Langdon WB, Weimer W (2013) Genetic programming for reverse engineering. In: 20th working conference on reverse engineering, WCRE 2013, pp 1–10

He L, Huang S (2017) Modified firefly algorithm based multilevel thresholding for color image segmentation. Neurocomputing 240:152–174

Hinojosa S, Oliva D, Cuevas E, Pajares G, Avalos O, Gálvez J (2018) Improving multi-criterion optimization with chaos: a novel Multi-Objective Chaotic Crow Search Algorithm. Neural Comput Appl 29(8):319–335

Horng M-H, Jiang T-W (2010) Multilevel image thresholding selection using the Artificial Bee Colony Algorithm. Artif Intell Comput Intell 6320:318–325

Huang T, Jia XD, Yuan HQ, Jiang JQ (2017) Niching community based differential evolution for multimodal optimization problems. In: IEEE, Piscataway

Ibrahim E, Birchell S, Elfayoumy S (2012) Automatic heart volume measurement from CMR images using ant colony optimization with iterative salient isolated thresholding. J Cardiovasc Magn Reson 14(1):1–2

Idris I et al (2015) A combined negative selection algorithm-Particle Swarm Optimization for an email spam detection system. Eng Appl Artif Intell 39:33–44

Jadhav AN, Gomathi N (2016) WGC: hybridization of exponential grey wolf optimizer with whale optimization for data clustering. Alex Eng J 57:1569–1584

Johny DC, Assistant AJS (2017) Negative selection algorithm : a survey. Int J Sci Eng Technol Res 6(4):711–715

Jourdan L, Basseur M, Talbi EG (2009) Hybridizing exact methods and metaheuristics: a taxonomy. Eur J Oper Res 199(3):620–629

Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39(3):459–471

Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput J 8(1):687–697

Kaveh A, Khayatazad M (2012) A new meta-heuristic method: ray optimization. Comput Struct 112–113:283–294

Kennedy J, Eberhart R (1995) Particle Swarm Optimization. IEEE Int Conf Neural Netw 4:1942–1948

Keshtegar B, Hao P, Wang Y, Li Y (2017) Optimum design of aircraft panels based on adaptive dynamic harmony search. Thin-Walled Struct 118(May):37–45

Khairuzzaman AKM, Chadhury S (2017) Moth-Flame Optimization Algorithm based multilevel thresholding for image segmentation. Int J Appl Metaheuristic Comput 8(4):58–83

Khairuzzaman AKM, Chaudhury S (2017) Multilevel thresholding using grey wolf optimizer for image segmentation. Expert Syst Appl 86:64–76

Khatibinia M, Yazdani H (2017) Accelerated multi-gravitational search algorithm for size optimization of truss structures. Swarm Evol Comput 1:1. https://doi.org/10.1016/j.swevo.2017.07.001

Khu ST, Liong SY, Babovic V, Madsen H, Muttil N (2001) Genetic programming and its application in real-time runoff forecasting. J Am Water Resour Assoc 37(2):439–451

Kiranyaz S, Uhlmann S, Ince T, Gabbouj M (2015) Perceptual dominant color extraction by multidimensional Particle Swarm Optimization. EURASIP J Adv Signal Process 2009:451638

Kirkpatrick S, Gelatt CD, Vecch MP (2007) Optimization by simulated annealing. Science 220(4598):671–680

Kora P, Kalva SR (2015) Improved Bat algorithm for the detection of myocardial infarction. Springerplus 4(1):666

Laguna M, Martí R (2003) Scatter Search, Methodology and Implementations in C. Springer, New York

Li P, Duan H (2012) Path planning of unmanned aerial vehicle based on improved gravitational search algorithm. Sci China Technol Sci 55(10):2712–2719

Lin M, Tsai J, Yu C (2012) A review of deterministic optimization methods in engineering and management. Math Probl Eng 2012:1–15

Liu B, Koziel S, Zhang Q (2016) A multi-fidelity surrogate-model-assisted evolutionary algorithm for computationally expensive optimization problems. J Comput Sci 12:28–37

Ma J, Ting TO, Man KL, Zhang N, Guan SU, Wong PWH (2013) Parameter estimation of photovoltaic models via cuckoo search. J Appl Math 2013:10–12

Mafarja MM, Mirjalili S (2016) Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. Neurocomputing 260:302–312

Mann PS, Singh S (2017) Improved metaheuristic based energy-efficient clustering protocol for wireless sensor networks. Eng Appl Artif Intell 57(2016):142–152

Marinaki M, Marinakis Y (2016) A Glowworm Swarm Optimization algorithm for the vehicle routing problem with stochastic demands. Expert Syst Appl 46(4):145–163

Massan SUR, Wagan AI, Shaikh MM, Abro R (2015) Wind turbine micrositing by using the firefly algorithm. Appl Soft Comput J 27:450–456

McCall J (2005) Genetic algorithms for modelling and optimisation. J Comput Appl Math 184(1):205–222

Mesbahi T, Rizoug N, Bartholomeus P, Sadoun R, Khenfri F, Lemoigne P (2017) Optimal energy management for a Li-ion battery/supercapacitor hybrid energy storage system based on Particle Swarm Optimization incorporating Nelder-Mead simplex approach. IEEE Trans Intell Veh 2(2):1

Mesejo P, Ibáñez Ó, Cordón Ó, Cagnoni S (2016) A survey on image segmentation using metaheuristic-based deformable models: state of the art and critical analysis. Appl Soft Comput J 44:1–29

Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst 89:228–249

Mirjalili S (2016) SCA : a sine cosine algorithm for solving optimization problems. Knowl Based Syst 96:120–133

Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67

Mirjalili S, Mirjalili SM, Lewis A (2014) Grey Wolf Optimizer. Adv Eng Softw 69:46–61

Mitchell M (1995) Genetic algorithms: an overview. Complexity 1(1):31–39

Mitchell M (1996) An introduction to genetic algorithms. The MIT Press, Cambridge

Moayedikia A, Ong K-L, Boo YL, Yeoh WG, Jensen R (2017) Feature selection for high dimensional imbalanced class data using harmony search. Eng Appl Artif Intell 57(2016):38–49

Mohamed AW, Sabry HZ, Khorshid M (2012) An alternative differential evolution algorithm for global optimization. J Adv Res 3(2):149–165

Mohammad L, Abualigah Q, Hanandeh ES (2015) Applying Genetic Algorithms to information retrieval using vector space model. Int J Comput Sci Eng Appl 5(1):19–28

Nagpal S, Arora S, Dey S, Shreya (2017) Feature selection using Gravitational Search Algorithm for biomedical data. Procedia Comput Sci 115:258–265

Neumann F, Witt C (2010) Bioinspired computation in combinatorial optimization—algorithms and their computational complexity. Springer, Berlin

Olague G, Trujillo L (2012) Interest point detection through multiobjective genetic programming. Appl Soft Comput J 12(8):2566–2582

Oliva D, Cuevas E, Pajares G (2014) Parameter identification of solar cells using artificial bee colony optimization. Energy 72:93–102

Oliva D, Hinojosa S, Cuevas E, Pajares G, Avalos O, Gálvez J (2017) Cross entropy based thresholding for magnetic resonance brain images using Crow Search Algorithm. Expert Syst Appl 79:164–180

Opara KR, Arabas J (2018) Differential evolution: a survey of theoretical analyses. Swarm Evol Comput. https://doi.org/10.1016/j.swevo.2018.06.010

Osman IH, Laporte G (1996) Metaheuristics: a bibliography. Ann Oper Res 63(5):511–623

Ouaddah A, Boughaci D (2016) Harmony search algorithm for image reconstruction from projections. App Soft Comput J 46:924–935

Ouadfel S, Taleb-Ahmed A (2016) Social spiders optimization and flower pollination algorithm for multilevel image thresholding: a performance study. Expert Syst Appl 55:566–584

Oz I, Topcuoglu HR, Ermis M (2013) A meta-heuristic based three-dimensional path planning environment for unmanned aerial vehicles. Simulation 89(8):903–920

Padhye N, Mittal P, Deb K (2013) Differential evolution: performances and analyses. In: 2013 IEEE congress on evolutionary computation (CEC), pp 1960–1967

Pardalos PM, Du D-Z, Graham RL (2013) Handbook of combinatorial optimization. Springer, US

Pereira FB, Tavares J (2009) Bio-inspired algorithms for the vehicle routing problem. Springer, US

Pereira DR et al (2016) Social-Spider Optimization-based support vector machines applied for energy theft detection. Comput Electr Eng 49:25–38

Pham DT, Huynh TTB, Bui TL (2013) A survey on hybridizing genetic algorithm with dynamic programming for solving the traveling salesman problem. In: 2013 international conference soft computer pattern recognition, SoCPaR 2013, pp 66–71

Piotrowski AP (2017) Review of differential evolution population size. Swarm Evol Comput 32:1–24

Piotrowski AP, Napiorkowski JJ (2016) Searching for structural bias in Particle Swarm Optimization and differential evolution algorithms. Swarm Intell 10(4):307–353

Piotrowski AP, Napiorkowski JJ (2018) Some metaheuristics should be simplified. Inf Sci (NY) 427:32–62

Plateau A, Tachat D, Tolla P (2002) A hybrid search combining interior point methods and metaheuristics for 0–1 programming. Int Trans Oper Res 9(6):731–746

Poli R, Kennedy J, Blackwell T (2007a) Particle Swarm Optimization. Swarm Intell 1(1):33–57

Poli R, Langdon WB, McPhee NF, Koza JR (2007b) Genetic programming an introductory tutorial and a survey of techniques and applications. Technical report CES475, vol 18, Oct 2007, pp 1–112

Portmann MC, Vignier A, Dardilhac D, Dezalay D (1998) Branch and bound crossed with GA to solve hybrid flowshops. Eur J Oper Res 107(2):389–400

Potvin JY (2009) A review of bio-inspired algorithms for vehicle routing. Stud Comput Intell 161(July):1–34

Prakash DB, Lakshminarayana C (2016) Optimal siting of capacitors in radial distribution network using Whale Optimization Algorithm. Alex Eng J 56:499–509

Prasad D, Mukherjee A, Mukherjee V (2017) Application of chaotic krill herd algorithm for optimal power flow with direct current link placement problem. Chaos Solitons Fractals 103:90–100

Puchinger J, Raidl GR (2005) Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. In: Mira J, Álvarez JR (eds) Artificial intelligence and knowledge engineering applications: a bioinspired approach. IWINAC 2005. Lecture notes in computer science, vol 3562. Springer, Berlin.

Rahimi S, Abdollahpouri A, Moradi P (2018) A multi-objective Particle Swarm Optimization algorithm for community detection in complex networks. Swarm Evol Comput 39:297–309

Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci (NY) 179(13):2232–2248

Regis RG (2013) Particle swarm with radial basis function surrogates for expensive black-box optimization. J Comput Sci 5(1):1–12

Rere LMR, Fanany MI, Arymurthy AM (2015) Simulated annealing algorithm for deep learning. Procedia Comput Sci 72:137–144

Reyna A, Fausto F (2017) AISearch. Nature-inspired Metaheuristic Optimization Algorithms. https://aisearch.github.io. Accessed 01 Jan 2017

Rutenbar RA (1989) Simulated annealing algorithms: an overview. IEEE Circuits Dev Mag 5(1):19–26

Sahlol AT, Ewees AA, Hemdan AM, Hassanien AE (2009) Training of feedforward neural networks using sine-cosine algorithm to improve the prediction of liver enzymes on FIsh farmed on nano-selenite. In: 2016 12th international conference computer engineering conference (ICENCO), pp 35–40

Sahoo A, Chandra S (2017) Multi-objective Grey Wolf Optimizer for improved cervix lesion classification. Appl Soft Comput J 52:64–80

Saji Y, Riffi ME (2016) A novel discrete bat algorithm for solving the travelling salesman problem. Neural Comput Appl 27(7):1853–1866

Salimans T, Ho J, Chen X, Sidor S, Sutskever I (2017) Evolution strategies as a scalable alternative to reinforcement learning, pp 1–13. arXiv:1703.03864v2

Sapra D, Sharma R, Agarwal AP (2017) Comparative study of metaheuristic algorithms using Knapsack Problem. In: 7th International conference on cloud computing, data science & engineering, pp 134–137

Sarjila R, Ravi K, Edward JB, Kumar KS, Prasad A (2016) Parameter extraction of solar photovoltaic modules using Gravitational Search Algorithm

Sayed GI, Hassanien AE, Nassef TM (2017) Genetic and evolutionary computing, vol 536. Springer, Berlin

Schneider JJ, Kirkpatrick S (2006) Stochastic optimization. Springer, Berlin

Sette S, Boullart L (2001) Genetic programming: principles and applications. Eng Appl Artif Intell 14(6):727–736

Shukla UP, Nanda SJ (2016) Parallel social spider clustering algorithm for high dimensional datasets. Eng Appl Artif Intell 56:75–90

Shukla R, Singh D (2016) Selection of parameters for advanced machining processes using firefly algorithm. Eng Sci Technol Int J 20(1):1–10

Siddique N, Adeli H (2016) Simulated annealing, its variants and engineering applications. Int J Artif Intell Tools 25(06):1630001

Silva P, Santos CP, Matos V, Costa L (2014) Automatic generation of biped locomotion controllers using genetic programming. Rob Auton Syst 62(10):1531–1548

Sipper M, Fu W, Ahuja K, Moore JH (2018) "Investigating the parameter space of evolutionary algorithms. BioData Min 11(1):2

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

Tan Y, Zhu Y (2010) Fireworks algorithm for optimization. In: ICSI 2010—proceedings first international conference, part I, 2010, June, pp 355–364

Tolabi HB, Ayob SM (2014) New technique for global solar radiation forecasting by simulated annealing and genetic algorithms using. Appl Sol Energy 50(3):202–206

Tsai P, Nguyen T, Dao T (2016) Genetic and evolutionary robot path planning optimization based on multiobjective grey wolf optimizer. In: Genetic and evolutionary computing proceedings of the tenth international conference on genetic and evolutionary computing, pp 166–173

Valdivia-Gonzalez A, Zaldívar D, Fausto F, Camarena O, Cuevas E, Perez-Cisneros M (2017a) A states of matter search-based approach for solving the problem of intelligent power allocation in plug-in hybrid electric vehicles. Energies 10(1):92

Valdivia-Gonzalez A, Zaldívar D, Fausto F, Camarena O, Cuevas E, Perez-Cisneros M (2017b) A states of matter search-based approach for solving the problem of intelligent power allocation in plug-in hybrid electric vehicles. Energies 10(1):92

Van Sickel JH, Lee KY, Heo JS (2007) Differential evolution and its applications to power plant control. In: 14th international conference on intelligent systems applications to power systems, no 2, pp 560–565

Vanneschi L, Castelli M, Silva S (2014) A survey of semantic methods in genetic programming. Genet Program Evolv Mach 15(2):195–214

Vocking B et al (2011) Algorithms unplugged. Springer, Berlin Heidelberg

Wang KJ, Adrian AM, Chen KH, Wang KM (2015) An improved electromagnetism-like mechanism algorithm and its application to the prediction of diabetes mellitus. J Biomed Inform 54:220–229

Wild SM, Regis RG, Shoemaker CA (2008) ORBIT: optimization by radial basis function interpolation in trust-regions. SIAM J Sci Comput 30(6):3197–3219

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Wu J, Qiu T, Wang L, Huang H (2011) An approach to feature selection based on Ant Colony Optimization and Rough Set, pp. 466–471

Xiang T (2016) Vehicle routing problem based on particle Swarm Optimization Algorithm with gauss mutation. Am J Softw Eng Appl 5(1):1

Xie C, Zheng H (2016) Application of improved cuckoo search algorithm to path planning unmanned aerial vehicles. In: 12th international conference intelligent computing theories and application, ICIC 2016, pp 722–729

Xu H, Pu P, Duan F (2018) Dynamic vehicle routing problems with enhanced ant colony optimization. Discret Dyn Nat Soc 2018:1–13

Wei L, Zhang Z, Zhang D, Leung SCH (2017) A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. Eur. J. Oper. Res. https://doi.org/10.1016/j.ejor.2017.08.035

Yadav PK, Prajapati NL (2012) An overview of genetic algorithm and modeling. Int J Sci Res Publ 2(9):1–4

Yan L, Yujuan Q, Zujian W, Wang L, Yan J (2015) A hybrid method combining genetic algorithm and Hooke—Jeeves method for 4PLRP. In: 2014 IEEE/CIC international conference on communication China—Work. CIC/ICCC 2014, vol 10, no. 4, pp 36–40

Yang X (2008) Nature-inspired metaheuristic algorithms, 2nd edn. Luniver Press, Beckington

Yang X (2010a) Firefly algorithm, Lévy flights and global optimization. Springer, Berlin

Yang X-S (2010b) A new metaheuristic bat-inspired algorithm. Stud Comput Intell 284:65–74

Yang XS (2011) Metaheuristic optimization: algorithm analysis and open problems. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence lecture notes in bioinformatics), vol 6630 LNCS, pp 21–32

Yang XS (2012) Flower pollination algorithm for global optimization. In: Lecture notes in computer science, vol 7445, LNCS, pp 240–249

Yang XS, He X (2013) Firefly algorithm: recent advances and applications. Int J Swarm Intell 1(1):1–14

Yang XS (2015) Nature-inspired algorithms: success and challenges. Comput Methods Appl Sci 38:129–143

Yang X-S (2018) Swarm-based metaheuristic algorithms and no-free-lunch theorems. Intech Open 2:64

Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: Proceedings 2009 world congress on nature and biologically inspired computing, NABIC 2009, pp 210–214

Yang XS, Deb S, Hanne T, He X (2015) Attraction and diffusion in nature-inspired optimization algorithms. Neural Comput Appl 19:1–8

You I, Yim K, Barolli L (2017) A Social Spider Optimization based home energy management system. In: Advances in network-based information systems, 20th international conference on network-based information systems, pp 771–778

Yurtkuran A, Emel E (2010) A new hybrid electromagnetism-like algorithm for capacitated vehicle routing problems. Expert Syst Appl 37(4):3427–3433

Zelinka I (2015) A survey on evolutionary algorithms dynamics and its complexity—mutual relations, past, present and future. Swarm Evol Comput 25:2–14

Zhang SZ, Lee CKM (2016) An improved artificial bee colony algorithm for the capacitated vehicle routing problem. In: Proceedings—2015 IEEE international conference on systems, man, and cybernetics SMC 2015, pp 2124–2128

Zhang S, Zhou Y (2017) Template matching using grey wolf optimizer with lateral inhibition. Opt-Int J Light Electron Opt 130:1229–1243

Zhou Y, Zhao R, Luo Q, Wen C (2017a) "Sensor deployment scheme based on Social Spider Optimization Algorithm for wireless sensor networks. Neural Process Lett 48:71–94

Zhou Y, Wang R, Zhao C, Luo Q, Metwally MA (2017b) Discrete greedy flower pollination algorithm for spherical traveling salesman problem. Neural Comput Appl. https://doi.org/10.1007/s00521-017-3176-4

Zou Y, Chakrabarty K (2003) Sensor deployment and target localization based on virtual forces. In: Twenty-second annual joint conference of the IEEE computer and communications, vol 2, no. C, pp 1293–1303