Natural Computing
https://doi.org/10.1007/s11047-020-09837-9

Similarity in metaheuristics: a gentle step towards a comparison methodology

Jesica de Armas[1] · Eduardo Lalla-Ruiz[2] · Surafel Luleseged Tilahun[3,4] · Stefan Voß[5]

Accepted: 16 December 2020

Abstract
Metaheuristics are found to be efficient in different applications where the use of exact algorithms becomes short-handed. In the last decade, many of these algorithms have been introduced and used in a wide range of applications. Nevertheless, most of those approaches share similar components leading to a concern related to their novelty or contribution. Thus, in this paper, a pool template is proposed and used to categorize algorithm components permitting to analyze them in a structured way. We exemplify its use by means of continuous optimization metaheuristics, and provide some measures and methodology to identify their similarities and novelties. Finally, a discussion at a component level is provided in order to point out possible design differences and commonalities.

Keywords Metaheuristics design · Comparison methodology · Pool template · Algorithm similarity

1 Introduction

Challenging problems arising from different complex applications often require solutions in a reasonable computational time, usually because of their hardness and practical relevance. This has resulted in the development of a large number of optimization techniques. In this context, when addressing an optimization problem, it can be tackled by means of exact algorithms that guarantee the optimality of the solution at the expense of execution time. Moreover, computing a (globally) optimal solution may not be an option in some cases, either due to the dimension of the problem instances or due to user requirements in terms of solving time and memory. This leads to the other main possibility which consists of changing the optimality advantage provided by the exact algorithms for the time efficiency given by metaheuristics. This way, the guarantee of finding an optimal solution is switched to a competitive performance allowing decision-makers to get good solutions with smaller computational effort.

A metaheuristic is '*an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method*' (Voß et al. 1999). More recently, Sörensen and Glover (2013) define a metaheuristic as '*a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms*'. Unlike classical or deterministic methods,

✉ Eduardo Lalla-Ruiz
e.a.lalla@utwente.nl

Jesica de Armas
jesica.dearmas@upf.edu

Surafel Luleseged Tilahun
surafelaau@yahoo.com

Stefan Voß
stefan.voss@uni-hamburg.de

1 Department of Economics and Business, University Pompeu Fabra, Barcelona, Spain

2 Department of Industrial Engineering and Business Information Systems, University of Twente, Enschede, The Netherlands

3 Addis Ababa Science and Technology University, Addis Ababa, Ethiopia

4 Department of Mathematical Sciences, University of Zululand, Richards Bay, South Africa

5 Institute of Information Systems, University of Hamburg, Hamburg, Germany

Published online: 10 February 2021

these methods are not much affected by the structure of the problem. This is because they try to improve the solution set based on the experience or on a '*trial and error*' approach using defined components (often stochastic) throughout the process. Many of these algorithms start with randomly or pseudo-randomly generated feasible solutions and update them by conducting systematic changes by means of a given solution structure or in a sampling fashion. Even though they do not guarantee optimality, they are found to be effective in different applications.

Since their development – starting from early metaheuristics in the middle of the 1960s – different research has been conducted along a wide spectrum of journals specifically dedicated to the presentation, application, and analysis of them. In this context, the introduction and application of emerging metaheuristics started to increase slowly at the beginning of the seventies while nowadays there is an alarmingly increasing rate.

Furthermore, within metaheuristics, there is a category called *nature-inspired* (also known as *bio-inspired*) that mimics natural processes and events from the universe to develop a well-defined and structured procedure. In literature, this type of algorithms has attracted a lot of attention from researchers and practitioners due to the appealing association of their components with nature, *e.g.* a mosquito (solution) seeking for a host (objective). The aforementioned advantage is commonly supported by a competitive performance when tackling real-world problems. Among nature-inspired algorithms, those based on the swarm organization, as well as the collective living and traveling of animals, accumulate a considerable interest in the scientific community (Xing and Gao 2014; Yang 2008). Thus, nowadays we are witnesses of a rapid growth of new nature-inspired metaheuristics. This growth, without rigorous in-depth analysis, may cause an overlap between already proposed algorithms and prospective novel ones in terms of components when extracting algorithmic procedures from nature. This raises some questions – from a methodological perspective – about their real contribution and novelty. In this realm, Sörensen (2015) remarks this trend in his influential paper, while indicating that some algorithms besides adapting the used terminology into a nature-inspired one, did not perform a proper analysis of their ruling components. Additionally, Laguna (2016) emphasizes the need to establish some policies in order to 'avoid the publication of articles that repackage and embed old ideas in methods that are claimed to be based on metaphors of natural or man-made systems and processes'. He also criticizes the use of quirky names and similes that do not match reality.

The previously observed circumstance is especially noticeable in continuous optimization, where in the last decade a considerable number of so-called 'new' nature-inspired metaheuristics have been 'invented'. This aroused some critics from the scientific community related to the novelty and contribution of those approaches. In fact, as discussed in Weyland (2010, 2015), Sörensen (2015), and Duarte et al. (2018) some of those new algorithms work in a similar search paradigm but are presented, by mimicking different processes, as new algorithms. Besides metaphor-related metaheuristics, there are also variants or hybrid metaheuristic approaches that are metaphor-less and whose components are not clearly stated or delimited from their original approaches. This might cause difficulty when advancing on the investigation on a given technique or hybridization scheme.

Considering the above discussion, the research contributions of this work are listed as follows:

- As indicated in a short note by Swan et al. (2015), a clear template dedicated to providing a purely functional description of metaheuristics for separating any metaphors from their defining components is necessary. Hence, in the same spirit as the work by Watson et al. (2006), we propose a classification method that supports the identification, description, and analysis of a given metaheuristic considering its defining components. The above procedure constitutes a way for providing insights about the contribution of their components as well as increasing the transparency of their algorithm design. It is worth noticing that this classification method is aimed at a design level; therefore it does not predict the performance of the algorithms nor substitutes the empirical analysis to measure the contribution of the components' differences.
- Bearing the proliferation of nature-inspired algorithms in mind, in this paper, we investigate the novelty of some of them in terms of their algorithmic components. The main goal of this selection is indicating and providing additional insights on what previous authors (Weyland 2010; Sörensen 2015) have already questioned, that is, their real 'novelty' from a methodological viewpoint besides their friendly and popular nature association. Thus, in doing so, by means of the selected sample, we analyze and determine which algorithm contributes with novelty to the field, and in which level their components are similar or a special case of other algorithms. The study can be extended to any other group of metaheuristics by applying the same methodology covered in this work.
- Finally, to provide a degree of novelty of new metaheuristics regarding older ones, we propose a method for evaluating the novelty of the proposed algorithm or determining if the studied algorithm coincides in some components with others already proposed in the literature. We exemplify the use of this

measure by means of the algorithms studied in this paper.

The rest of this paper is organized as follows. Section 2 describes the pool template which is later used to categorize different metaheuristics according to their structure. Afterwards, in Sect. 3, a selection of algorithms dealing with continuous optimization problems is presented in terms of the pool template components. Analysis and discussion of the selected metaheuristics are provided in Sect. 4. Finally, Sect. 5 draws the main conclusions extracted from this work and provides some lines for further research.

2 A pool template for metaheuristics comparison

The functioning of most metaheuristics can be described based on their intensification and diversification capabilities as well as the manipulation of their starting conditions. A dynamic balance between diversification and intensification is crucial to obtain a good performance of these optimization methods (Blum and Roli 2003).

As Greistorfer and Voß (2005) pointed out, some well-known procedures such as simulated annealing, tabu search, or genetic algorithms can be explained by means of an established structure. With that idea, they proposed a pool template as a solution framework that uses and registers solutions in a pool (or set) of solutions. Afterwards, some are selected for further recombination and improvement before putting them back to the pool in a feedback fashion. Inspired by that work, in this paper, a pool template as a framework for decomposing and analyzing metaheuristics is proposed. The pool template structure is presented in Fig. 1 and its components are described below.

- **Generation method** (*GM*)**:** This component defines the method providing the initial solution/s by means of a given procedure. Therefore, the produced solution/s is/are the input of the pool later used by the metaheuristic.
- **Pool** (*P*)**:** It is a set composed of solutions initially constructed by means of the *GM*. The pool is represented by P having a cardinality of p, i.e., $|P| = p$. For example, for the case of single-based approaches $p = 1$, while in population-based approaches $p > 1$.
- **Archive** ($\overline{P}$)**:** This component serves as a memory to guide the search by storing information about the solution features which are promising. For example, in some algorithms, elite solutions are stored and used to direct the search of the algorithm. That idea is handled by the archive which is like a memory of the algorithm. The cardinality of $\overline{P}$ is $\overline{p}$, which needs not to be constant throughout the iterations.
- **Selected pool** (*S*)**:** The selected solutions from the pool (to be updated) and the archive (to guide the search) are processed in this structure, being sorted if necessary and adjusted for the updating mechanism.
- **Updating mechanism** (*UM*)**:** This function handles the solutions in the selected pool S. Thus, an updated solution set referred to as updated pool S' is produced by means of different rules, algorithmic changes, and procedures such that $UM : S \rightarrow S'$ with $|S'| = S_{upd}$.
- **Updated pool** (S')**:** The output solutions from the updating mechanism compose the updated pool which is later handled by the input function and by the archiving function to update the solutions in P and $\overline{P}$, respectively.
- **Archiving functions** (*AF*)**:** These functions add solutions to the archive $\overline{P}$. There are two archiving functions. The first one, $AF_1(P)$, directly archives solutions from the pool P without any manipulation from other components. The second one, $AF_2(S')$, archives solutions from S', the set of updated solutions called updated pool obtained through the updating mechanism[1].
- **Output functions** (*OF*)**:** These functions are responsible for selecting the solutions from the pool P and the archive $\overline{P}$ to be handled by the updating mechanism. Hence, from the pool P and the archive $\overline{P}$, solutions in S_1 and S_2 will be selected by using $OF_1(P)$ and $OF_2(\overline{P})$, respectively. The set of selected solutions is referred to as the selected pool S, as explained below.
- **Input function** (*IF*)**:** This function is involved in the selection and addition of the resulting solutions from the updating stage into the pool P. Hence, it takes the processed set of updated solutions in the updated pool S' and uses its rule to determine how to update the pool of solutions.

As can be observed in Fig. 1, there is a cyclical structure starting from the pool P with initial solution/s and handling them along the process by means of the defined functions. Notation *Pool*, $Pool_{sel}$ and $Pool_{upd}$ is introduced in the figure to ease the identification of the different sets of solutions. Thus, a given metaheuristic can be decomposed following this structure. It is important to note that the flow process of the solutions along the pool frame is performed

[1] We should note that the archiving function may be extended in the sense of distinguishing long-term memory and short-term memory functions, the first accounting for recent solutions or solution attributes and the latter accounting for measures supporting to memorize an overall history of a search, e.g., by using frequency-based memory to count for occurrences of properties of solutions.

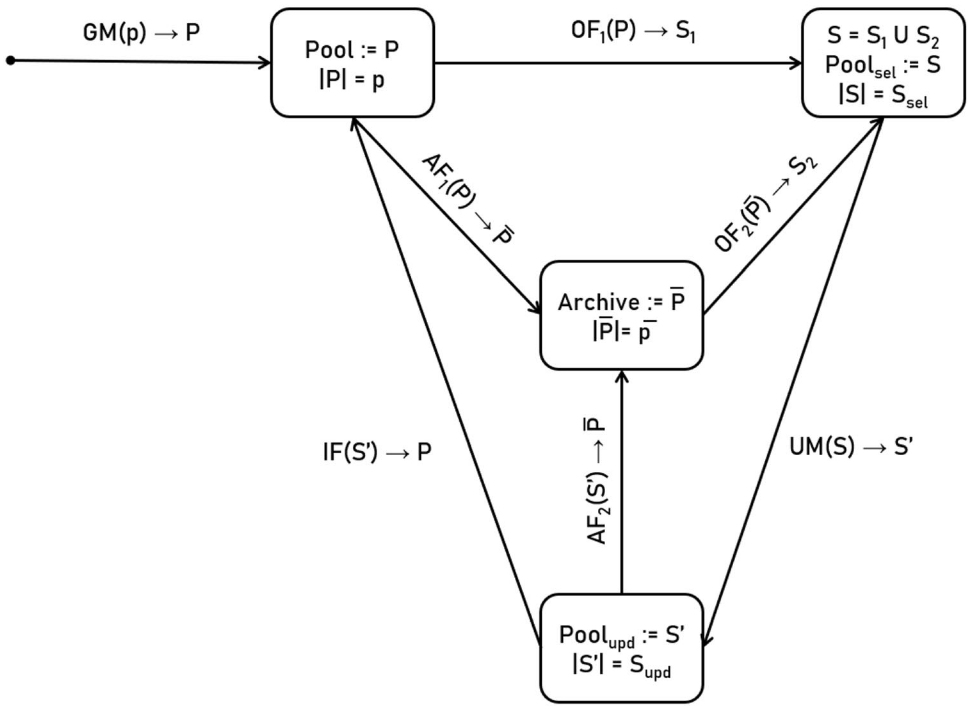


Fig. 1 Pool template scheme

until a termination criterion is met, and the output solutions are selected from the final pool and/or the archive.

An example application of the pool template for a genetic algorithm (GA) is illustrated in Fig. 2. This type of algorithm usually works starting with a randomly generated set of N solutions (i.e., a population) and improving it through iterations. In the context of the pool template, the initial set of solutions is defined by a pool of size $|P| = N = p$ with $p > 1$. In this case, GM provides these initial solutions either at random or through a given procedure. According to Whitley (1994), no archive is used by this metaheuristic and hence, $S_2 = \emptyset$. Consequently archiving functions (AF) and the second output function OF_2 are not needed (dashed lines in Fig. 2). Thus, the solutions that are going to be used in the next search step, S, are selected only by $OF_1(P)$, which in this case is an objective-function-based (or, if modified to include other means of heuristic measure, in GA language, fitness-based) process.

In the GA scope, the *recombination* of solutions (i.e., parents) producing other solutions (i.e., offspring) is known as crossover. Moreover, the *alteration* process whereby some existing solutions are changed is called mutation, which is commonly defined as a procedure for introducing diversity in the population in the form of new solutions. These two methods are considered in the updating mechanism which provides updated solutions $UM(S) \rightarrow S'$. The resulting solutions are stored in the updated pool S' and they are used by the input function IF for determining how and in which quantity they will compose the updated solutions belonging to P. In this step, strategies such as *replication* (i.e., reproduction or keeping given solutions) takes place. In general, a certain number of solutions with the best objective function (or fitness) values (elite $E \in P$) in the current iteration are transferred to the next iteration. Thus, the solutions belonging to the next pool of solutions are those resulting from the recombination, replication, and alteration.

From the previous example, it can be mentioned that obtaining the pool template of the Harmony Search (HS) algorithm will result in the same pool template structure provided in Fig. 2, i.e., functions OF_2, AF_1 and AF_2 are not used. This is aligned, in terms of algorithm design, with the indications provided by Weyland (2015) concerning similarities between HS and evolutionary algorithms. Moreover, if the pool template components (i.e., GM, UM, IF, and OF) from the algorithm provided in Weyland (2015) are compared with those corresponding to HS, it can be seen indeed that both approaches are the same[2]. Nevertheless, it is relevant to indicate that some degree of novelty can be justified when two or more components from different algorithms but not yet combined in one are proposed in a new algorithm. In such cases, it is recommendable to provide a systematic study (e.g., through the pool template) to contextualize and support this contribution.

[2] For this comparison, we rely on Algorithm 3 presented Weyland (2015).

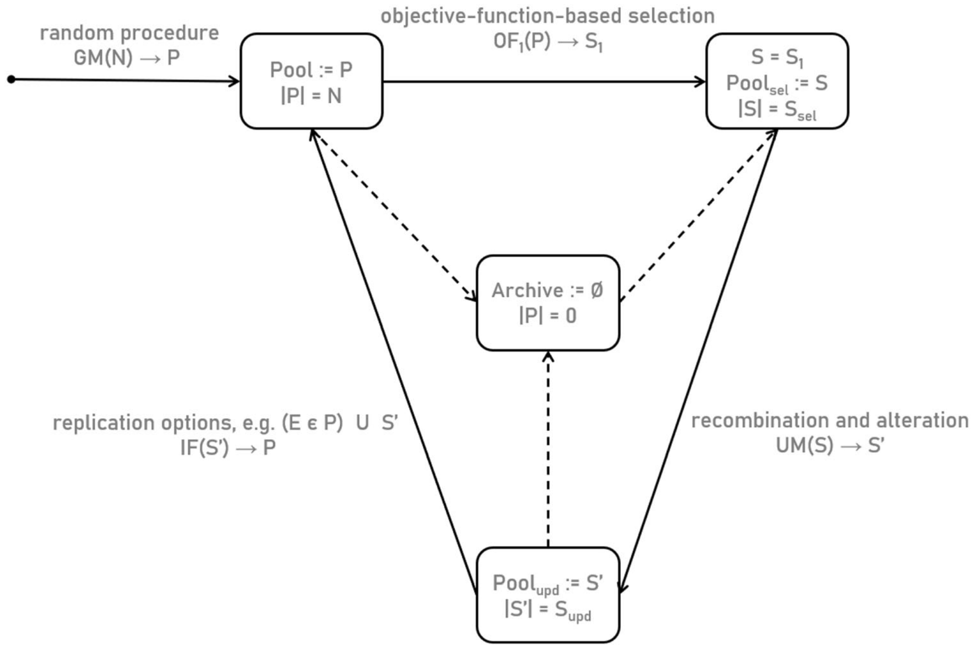


Fig. 2 Description of a GA based on the pool template. Dashed lines indicate that the corresponding functions OF_2, AF_1 and AF_2 are not used

3 Metaheuristics under analysis

Numerous and diverse metaheuristics have been proposed in the literature. Thus, to maintain the longitude of this work up to a comprehensive dimension without losing details and considering the large number of metaheuristics algorithms, our focus is restricted to several swarm-based algorithms applied in continuous optimization. In doing so, we have selected a sample of 15 algorithms, trying to get a set of diverse algorithms which allows us to test the performance and flexibility of our framework. On that selection, we have paid particular attention to some modern nature-inspired algorithms with the aim of analyzing their contribution and design besides their already provided metaphor-based algorithmic scheme and definitions. For maintaining the same studying baseline when analyzing the selected metaheuristics, we consider their application to a maximization problem as given in (1).

$$\begin{aligned}&\max_{x}\ f(x)\\&\ s.t.\ \ x \in S \subseteq \Re^{n}\end{aligned} \tag{1}$$

where x is the decision variable, $f(x)$ is the objective function, and S is the feasible search space. Furthermore, since we discuss the similarities and differences among the metaheuristics, below we briefly describe the algorithms based on the pool template components. The notation used throughout the descriptions is summarized as follows:

- *rand*() represents a random vector of appropriate dimension with entries of *rand*, where *rand* is a random number between 0 and 1 generated from a uniform distribution.
- r represents a random number between -0.5 and 0.5, i.e $r = rand - 0.5$, and similarly $r()$ is a random vector with entries of r.
- *randn* stands for a random number from a normal distribution with a given mean and standard deviation.
- x_i^t represents the i^{th} solution at iteration t.
- x_b is the best performing solution in the given iteration.
- x_i^{best} refers to the best historic location of the solution x_i in previous iterations.

Finally, the procedure followed for collecting the works of the selected metaheuristics was based on the selection of the papers where the metaheuristics were initially proposed. In this regard, a guideline for applying the pool template is to include the citations of the seminal works related to the metaheuristic(s). This permits, on the one hand, to proper link the studied methods and, on the other hand, to have an incremental analysis with regard to the studied approaches. It is worth mentioning that in the case the pool template is used when proposing a metaheuristic, it is also relevant to include, besides traditional metaheuristics, those novel variants that might also be related to the proposed method.

3.1 Random search (RS)

Random search is a direct search method introduced in the 1950's (Brooks 1958; Rastrigin 1963). This approach samples solutions from across the search space by using (2) for a given number of iterations. The best solution known is updated if a sampled solution results in an improvement. The corresponding pool structure details are the following. The algorithm starts with random solutions, and a singleton solution is allowed, i.e., $GM(p)$ is random with $p > 0$. No archive is used and each solution is selected by $OF_1(P)$ for the updating mechanism, i.e., $S = S_1 = P$. The updating mechanism $UM(S)$ is a random local search and is given by (2).

$$x_i^{t+1} = x_i^t + \lambda r() \tag{2}$$

where λ is an algorithm parameter for the step length of the move. The output S' obtained by this mechanism is used by the input function $IF(S')$, which updates the pool P using the best solution from $P \cup S'$.

3.2 Particle swarm optimization (PSO)

This metaheuristic, proposed by Kennedy and Eberhart (1995), is inspired by the collective behavior of some animal species. The algorithm randomly distributes solutions along the search space that are *moved* (i.e., modified) considering (3) and (4) until a given stopping criterion is met. In terms of its pool structure, the algorithm starts with a random set of paired solutions, i.e., $GM(p, v)$ is random with $p = v > 1$. Each solution x_i has a corresponding additional component v_i to guide the search. Initially, the first archiving function $AF_1(P)$ saves all the p solutions as the best performance of each individual solution and also determines the best solution x_b, i.e., $\overline{P} = P$. All the solutions in P and $\overline{P}$ are selected by $OF_1(P)$ and $OF_2(\overline{P})$ for updating, i.e., $S_1 = P$ and $S_2 = \overline{P}$. The updating mechanism $UM(S)$ is given in (3) and (4).

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

where

$$v_i^{t+1} = wv_i^t + \lambda_1 \cdot rand \cdot (x_i^{best} - x_i^t) + \lambda_2 \cdot rand \cdot (x_b - x_i^t)$$

for the best solution x_b, the best configuration of solution x_i represented by x_i^{best}, and algorithm parameters w, λ_1, and λ_2.

$$x_i^{t+1} = x_i^t + \lambda_1 \cdot rand \cdot (x_i^{best} - x_i^t) + \lambda_2 \cdot rand \cdot (x_b - x_i^t) + w \sum_{k=1}^{t+1} v_i^k \tag{4}$$

All the updated solutions go to the updated pool S'. From there, the archive $\overline{P}$ is updated by using $AF_2(S')$, i.e., if better solutions are found, when applicable the best configuration of solution x_i so far (i.e., x_i^{best}) and/or the global best x_b are updated. The input function $IF(S')$ replaces the solution in the pool P by the updated solutions.

3.3 Invasive weed optimization (IWO)

This metaheuristic was proposed by Mehrabian and Lucas (2006) and mimics the colonizing behavior of weeds. Its functioning consists of generating an initial population of solutions at random. Then, each solution will generate a number of randomly distributed solutions based on (5). The number of new solutions that can be generated by each solution is linearly determined considering the best and worst solution in the population and the solution itself. When the population exceeds an upper limit on the number of solutions, the worst solutions are excluded. This process is repeated until a termination criterion is met. Details of its pool structure are the following. The generation method $GM(p)$ is random with $p > 1$. No archive is used. The output function $OF_1(P)$ takes all the solutions from the pool P, so $S = S_1 = P$. The updating mechanism $UM(S)$ is given by (5).

$$x_i^{t+1} = x_i^t + \lambda \tag{5}$$

where λ is drawn from a normal distribution with mean zero and a standard deviation σ which will decrease from σ_{max} to σ_{min} through iteration. The input function IF constructs the new pool P by taking the best p solutions from $P \cup S'$.

3.4 Wolf pack search (WPS)

This is a metaheuristic proposed by Yang et al. (2007) which abstracts the behavior of wolves when searching for food. WPS starts with a number of randomly generated solutions. Over iterations, new solutions are generated using (6) while updating the best solution known. Its translation into the pool structure is the following. The initial solutions are generated randomly, i.e., $GM(p)$ is random with $p > 1$. The best solution is archived, i.e., $AF_1(P)$ takes x_b from P. The output functions $OF_1(P)$ and $OF_2(\overline{P})$ take all the solutions from the pool P and the best solution from the archive $\overline{P}$, i.e., $S_1 = P$ and $S_2 = x_b$. The solutions are updated by using (6).

$$x_i^{t+1} = x_i^t + \lambda \frac{x_b - x_i^t}{||x_b - x_i^t||} \tag{6}$$

where λ is an algorithm parameter for the step length.

The second archiving function $AF_2(S')$ updates the archive $\overline{P}$ if an updated solution is better than an archived one, and the input function $IF(S')$ replaces the solutions in the pool P by the updated solutions.

3.5 Firefly algorithm (FA)

This algorithm was introduced by Yang (2008) inspired by the flashing behavior of lighting bugs also called fireflies. FFA considers a set of solutions that are initially generated at random and where each solution is updated according to the below-described equation. Thus, its pool template is as follows. The initial solutions are generated randomly, i.e., $GM(p)$ is random with $p > 1$. There is no archiving function as it is a memory-less algorithm, i.e., $\overline{p} = 0$. The first output function $OF_1(P)$ takes all the solutions in the pool P, that is $S = S_1 = P$. Since there is no archiving, $S = S_1$. For each solution x_i from S the updating mechanism $UM(S)$ uses (7).

$$x_i^{t+1} = \begin{cases} x_i^t + \lambda_2 r(), & if\ \forall k, f(x_i^t) > f(x_k^t), \\ x_i^t + \lambda_1(x_j^t - x_i^t) + \lambda_2 r(), & \forall j | f(x_j^t) > f(x_i^t), otherwise \end{cases} \tag{7}$$

where λ_1 is a step length based on the performance of the solution and the distance between the solutions, and λ_2 is another step length.

The input function $IF(S')$ replaces the solutions in the pool P by the updated solutions.

3.6 Oriented search algorithm (OSA)

This is a population-based algorithm proposed by Zhang et al. (2008) that simulates the human random search behavior. The set of solutions is generated at random and solutions generate new ones by means of (8). At each iteration the best solution found is updated in case of an improvement. In the context of the pool template, random multiple solutions are generated to construct the initial pool P, i.e., $GM(p)$ is random with $p > 1$. No archive method is used and hence $S_2 = \emptyset$. The first output function $OF_1(P)$ takes all the solutions in the pool P for updating, i.e., $S = S_1 = P$. The updating mechanism $UM(S)$ is based on (8).

$$x_i^{t+1} = x_i^t + rand \cdot (x_b(1 + w \cdot randn) - x_i^t) \tag{8}$$

where $randn$ is a random number from a normal distribution between zero and one, and w is an algorithm parameter which will decrease from w_{max} to w_{min} during the performed iterations.

Developing this equation we obtain $x_i^{t+1} = x_i^t + rand \cdot x_b - rand \cdot x_i^t + rand \cdot randn \cdot w \cdot x_b$. Hence the updating formula can be written as given in Eq. (9).

$$x_i^{t+1} = x_i^t + rand \cdot (x_b - x_i^t) + \lambda_2 x_b \tag{9}$$

where λ_2 is a step length equal to $rand \cdot randn \cdot w$. The input function $IF(S')$ replaces solutions in the pool P by S'.

3.7 Roach infestation optimization (RIO)

This metaheuristic was proposed by Havens et al. (2008) and emulates the behavior of cockroaches. Initial solutions are generated at random and updated using a given equation for a certain number of iterations. In terms of the pool structure, it starts with a random set of paired solutions, i.e., $GM(p, v)$ is random with $p = v > 1$. Each solution x_i has a corresponding component v_i used for updating the solution x_i. The first archiving function $AF_1(P)$ archives all the solutions in P, i.e., initially $\overline{P} = P$. The first output function $OF_1(P)$ takes all the solutions in P. Similarly, the second output function $OF_2(\overline{P})$ takes all the solutions from the archive $\overline{P}$. The updating of each solution x_i is conducted using the equation given in (10).

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{10}$$

where

$$v_i^{t+1} = C_0 v_i^t + \lambda_{max} \cdot rand(). * (x_i^{best} - x_i^t) + \lambda_{max} \cdot rand(). * (x_b^l - x_i)$$

, x_i^{best} is the best position of x_i in previous iterations, x_b^l is the best solution in the radius of d from x_i, C_0 and λ_{max} are algorithm parameters, and $.*$ gives an element-wise multiplication of vectors of the same dimension, i.e., $X. * Y = [x_i y_i]$. A random update is also suggested after a threshold.

Hence, the updating equation can be reduced to (11).

$$x_i^{t+1} = x_i^t + \lambda_{max} \cdot rand(). * (x_i^{best} - x_i^t) + \lambda_{max} \cdot rand(). * (x_b^l - x_i) + C_0 \sum_{k=1}^{t} v_i^k \tag{11}$$

The second archiving function $AF_2(S')$ updates the archive $\overline{P}$ by replacing the solutions if better performance is recorded, and the input function $IF(S')$ replaces the solutions in the pool P by the updated values.

3.8 Gravitational search algorithm (GSA)

This metaheuristic was proposed by Rashedi et al. (2009) and it is based on the law of gravity. It considers an initial set of solutions generated at random that generates new solutions by means of (12) through iterations until a stopping criterion is met. Its translation into the pool structure is the following. Random initial paired solutions are used as initial members of the pool, i.e., $GM(p, v)$ is random with $p = v > 1$. There is no archive, i.e., $\overline{p} = 0$. The output function $OF_1(P)$ takes every solution in P, i.e., $S = S_1 = P$. The updating mechanism $UM(S)$ is developed in the same way as it is done in PSO by means of (3), i.e.,

$x_i^{t+1} = x_i^t + v_i^{t+1}$. Additionally, the solution component v_i is computed using (12).

$$v_i^{t+1} = rand \cdot v_i^t + \sum_{x_j^t \in K, j \neq i} \lambda_{ij}(x_j^t - x_i^t) \tag{12}$$

where K is the set of best solutions with its cardinality initially set to the algorithm parameter K_{best}. This set K decreases in cardinality during iterations with its last element being x_b. λ_{ij} is a step length which depends on the distance between the solutions and the performance of the solutions. This can be reduced to (13).

$$x_i^{t+1} = x_i^t + \sum_{x_j^t \in K, j \neq i} \lambda_{ij}(x_j^t - x_i^t) + \sum_{k=1}^{t+1} rand \cdot v_i^k \tag{13}$$

The input function $IF(S')$ replaces the solutions in the pool P by the updated solutions.

3.9 Bee swarm optimization (BSO)

This metaheuristic, proposed by Akbari et al. (2009), considers the foraging behavior of honey bees. It starts with a set of solutions that is divided into three categories according to the objective function value. The solutions generate new solutions through iterations until a termination condition is met. Its pool structure is detailed as follows. It starts the pool with random solutions, i.e., $GM(p)$ is random with $p > 2$. The first archiving function $AF_1(P)$ initializes the archive $\overline{P}$ with the best $\overline{p}$ solutions from the pool P, for $\overline{p} < \frac{p}{2}$. The first output function $OF_1(P)$ copies the solution in the pool P to S. The solutions in S_1 are categorized into three levels before the updating mechanism: the best $\overline{p}$ solution as X_1 (elite), the worst $p - 2\overline{p}$ solutions as X_3, and the rest as X_2. The updating mechanism $UM(S)$ is then given in (14).

$$x_i^{t+1} = \begin{cases} x_i^t + \lambda_1(x_i^{best} - x_i^t) + \lambda_2(x_b - x_i^t), & if\ x_i^t \in X_1 \\ x_i^t + \lambda_2(x_b - x_i^t), & if\ x_i^t \in X_2 \\ x_i^t + \lambda_3 r(), & otherwise \end{cases} \tag{14}$$

The updating mechanism given in (14) can be written as given in (15).

$$x_i^{t+1} = x_i^t + \lambda_1(x_i^{best} - x_i^t) + \lambda_2(x_b - x_i^t) + \lambda_3 r() \tag{15}$$

where

$$\lambda_1 = \begin{cases} 1, & x_i^t \in X_1 \\ 0, & otherwise \end{cases}$$

,

$$\lambda_2 = \begin{cases} 0, & x_i^t \in X_3 \\ 1, & otherwise \end{cases}$$

and

$$\lambda_3 = \begin{cases} 1, & x_i^t \in X_3 \\ 0, & otherwise \end{cases}$$

.

The second archiving function $AF_2(S')$ updates the archive $\overline{P}$ by replacing solutions if better ones have been obtained. The input function $IF(S')$ replaces all the solutions in the pool P.

3.10 Bat algorithm (BA)

This algorithm introduced by Yang (2010) mimics the echolocation of bats. In this algorithm, for a set of randomly generated solutions, two measures are defined for each individual and later used in the generation of solutions. The details of its pool structure are the following. Multiple random-paired solutions construct the pool P, i.e., $GM(p, v)$ is random with $p = v > 1$. There is no archive in the algorithm. The first output function $OF_1(P)$ takes every solution in P, i.e., $S = S_1 = P$. The updating of the solution is conducted using (16).

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{16}$$

for $v_i^{t+1} = v_i^t + (x_i^t - x_b)\lambda_i$ where $\lambda_i = \lambda_{min} + (\lambda_{max} - \lambda_{min})\beta$ with the algorithm parameter $\beta \in [0, 1]$ and prefixed parameter values λ_{min} and λ_{max}. According to this updating formula, the solution moves away from the best solution. However, the author also mentions that it resembles particle swarm optimization under certain conditions (Yang 2010). Hence, v_i should be $v_i^{t+1} = v_i^t + (x_b - x_i^t)\lambda_i$. Then, the updating formula can also be expressed as in (17).

$$x_i^{t+1} = x_i^t + \lambda_i(x_i^t - x_b) + \sum_{k=1}^{t} v_i^k \tag{17}$$

After this formula a random movement is also considered under certain probability. The updated solutions replace all the solutions in the pool P through the input function $IF(S')$.

3.11 Simple optimization algorithm (SOPT)

This algorithm is a population-based metaheuristic proposed by Hasançebi and Azad (2012) for engineering design optimization problems. It starts with a population of random solutions and, through iterations, individuals are generated considering (18). When generating the new individuals, this approach considers the normal distribution with a standard deviation calculated considering solutions' structure. The details of its pool structure are the following. Random multiple feasible solutions are generated to construct the initial pool P, i.e., $GM(p)$ is random with $p > 1$.

No archive is used. The first output function OF_1 takes all the solutions from the pool P, i.e., $S = S_1 = P$. This algorithm considers that the generation of new candidate solutions is more probable in the vicinity of the best solution. Thus, the updating is done using the mechanism given in (18).

$$x_i^{t+1} = x_b + \lambda R \tag{18}$$

where λ is a step length, R is a vector of random numbers from a normal distribution of mean zero and standard deviations calculated considering the corresponding solution component for all solutions.

The updating mechanism of this algorithm is a two-phase approach that iterates updating the pool S'. Thus, an additional formula for the second phase is used by doubling λ as given in (19)

$$x_i^{t+1} = x_b + 2\lambda R \tag{19}$$

In the input function $IF(S')$, only the best solutions among the pool P and the updated solutions S' are used to update the solutions in the pool P while preserving the pool size, i.e., the best p solutions are selected from $P \cup S'$.

3.12 Collective animal behavior (CAB)

This algorithm was proposed by Cuevas et al. (2012) and emulates a group of animals which interact with each other based on the biological laws of collective motion. The set of solutions are sorted and used to update two memory structures through iterations. The generation of individuals considers (20), (21) and (22). In the context of the pool template the generation method uses a random procedure, i.e., $GM(p)$ is random with $p > 1$. Some of the best solutions are archived, i.e., B of the p solutions from the pool P using the first archiving function $AF_1(P)$. The output functions take all the solutions in the pool P and in the archive $\overline{P}$, thus, $S_1 = P$ and $S_2 = \overline{P}$. As a result, $|S|$ is $p + \overline{p} = p + B$. The updating mechanism of this algorithm is a three-phase approach that uses (20), (21) and (22) in the given sequence order.

$$x_i^{t+1} = x_h + \lambda r() \tag{20}$$

where x_h is the solution from the archive ($\overline{P}$) with the objective value closest to x_i^t, and λ is an algorithm parameter.

$$x_i^{t+1} = \begin{cases} x_i^t \pm \lambda_1 (x_h - x_i^t), & rand < H \\ x_i^t \pm \lambda_2 (x_b - x_i^t), & otherwise \end{cases} \tag{21}$$

where H is an algorithm parameter in [0,1], x_h is the solution from the archive with objective value closest to x_i^t, x_b is the best solution among the current solutions, and λ_1 and λ_2 are the other algorithm parameters in [-1,1].

$$x_i^{t+1} = \begin{cases} x_{min}() + rand() \cdot (x_{max}() - x_{min}()), & rand < threshold \\ x_i^t, & otherwise \end{cases} \tag{22}$$

where *threshold* is an algorithm parameter, and $x_{min}()$ and $x_{max}()$ are vectors containing the lower and upper bounds for each component of a solution. After this, the archive $\overline{p}$ is updated by $AF_2(S')$ using the rule of replacing weak solutions (by means of the quality of the objective function). The updated solution replaces the solutions in the pool P.

3.13 Artificial tribe algorithm (ATA)

This metaheuristic presented by Chen et al. (2012) operates by simulating the existent skill of natural tribes. In the algorithm, a set of solutions is generated at random and updated through iterations based on (23). Regarding the pool structure, the algorithm starts with random multiple solutions, i.e., $GM(p)$ is random with $p > 1$. The archive $\overline{P}$ is initialized with the solutions from the pool P using the first archiving function $AF_1(P)$. The output functions use the solutions from the pool P and the archive $\overline{P}$, hence $S = P \cup \overline{P}$. The updating mechanism $UM(S)$ uses the equation given in (23).

$$x_i^{t+1} = \begin{cases} x_i^t + rand \cdot |x_i^{best} - x_i^t| + \lambda \cdot rand \cdot |x^* - x_i^t|, & if\ \Delta f(x^*) < Tol \\ rand \cdot x_i^t + (1 - rand) x_j, & otherwise \end{cases} \tag{23}$$

where x^* is the best solution found so far, x_j is a solution randomly selected, and, in each iteration, $\Delta f(x^*)$ is the change in the best functional value from the previous iteration, and λ and Tol are algorithm parameters.

The second archiving function $AF_2(S')$ updates the archive $\overline{P}$ replacing the solutions by better solutions from S'. Finally, the input function $IF(S')$ replaces the pool solutions by the updated solutions.

3.14 Black hole (BH)

This optimization method proposed by Hatamlou (2013) is inspired by the black hole phenomenon occurring in the universe. It starts with an initial set of candidate solutions with their corresponding objective function value calculated. At each iteration, the best candidate is selected to be the guiding solution. Then, solutions are perturbed according to (24). If a solution is below a given ratio provided by the guiding solution and the perturbed solution, then a new solution is randomly generated. In terms of the pool structure, the algorithm can be described as follows. Multiple random solutions are generated to be the members of the initial pool P, i.e., $GM(p)$ is random with

$p > 1$. The best solution x_b is archived using the first archiving function $AF_1(P)$. S is composed of $S_1 = P$ and $S_2 = \{x_b\}$. The p solutions are updated using (24).

$$x_i^{t+1} = \begin{cases} x_i^t + rand \cdot (x_b - x_i^t), & if\ d(x_i^t, x_b) \geq \dfrac{f(x_b)}{\sum_{k=1}^{N} f(x_k^t)} \\ generate\_random\_sol & otherwise \end{cases} \tag{24}$$

where $d(x_i, x_j)$ is a distance measure between x_i and x_j, $\frac{f(x_b)}{\sum_{k=1}^{N} f(x_k^t)}$ is the ratio determining if a solution is used or a new one has to be generated, and f represents the objective function value. The archive is updated if a better solution is found, and the updated solutions in S' replace all the solutions in the pool P.

3.15 Animal migration optimization (AMO)

This metaheuristic proposed by Li et al. (2014) mimics the animal migration behavior. The algorithm can be described as follows. Random multiple solutions are generated as initial solutions, i.e., $GM(p)$ is random with $p > 1$. The best solution is archived using the first archiving function $AF_1(P)$. S is composed of $S_1 = P$ and $S_2 = \{x_b\}$. Then, all the solutions in S are used by a two-phase updating mechanism following 25 and 26.

$$x_i = x_i + \lambda(x_j - x_i) \tag{25}$$

for a solution x_j in the neighborhood of x_i, and an algorithm parameter λ.

$$x_i = x_{j_1} + rand \cdot (x_b - x_i) + rand \cdot (x_{j_2} - x_i) \tag{26}$$

for two random solution x_{j_1} and x_{j_2}, with $j_1 \neq j_2 \neq i$ from S. In this process only improving updates are accepted. If a solution better than the stored best is found, then it will replace the archived solution using the second archiving function $AF_2(S')$. Finally, the updated solutions replace the solutions in the pool P by means of the input function $IF(S')$.

4 Analysis and discussion on the selected algorithms

This section is devoted to analyzing the algorithms listed in Sect. 3, in terms of their pool template components. Moreover, a similarity index aimed at providing a measure of the likeness among the updating mechanism component of the different approaches is proposed. Finally, a discussion regarding the relation and grade of similarity among algorithms is presented.

4.1 Analysis of the components constituting the algorithms

In order to develop this analysis, each algorithm component is studied and categorized according to its definition and function. For instance, an updating mechanism involving that a solution is randomly moved is categorized as random move. Moreover, it should be noted that, excluding the random search metaheuristic, all the algorithms considered in this work are population-based.

The first component to consider is the generation method $GM(p)$, i.e., the starting conditions of the algorithms. Although it can be classified as *random*, *pseudo-random* or *deterministic*, all the algorithms described in this work generate the initial set of solutions at *random*. The number of solutions p in the pool P depends on the algorithm and its implementation, but $p > 0$, $p > 1$, and $p > 2$ are the most common constraints. Additionally, the archive size $\overline{p}$ ranges from 0 (when no archive is used) to p. When the archive $\overline{P}$ is considered, the first archiving function $AF_1(P)$ might take the *whole set* P, or *a subset of* P, such as the best solution x_b, or the $\overline{p}$ best solutions from P.

The first and second output functions of the algorithms usually take all the solutions in P and $\overline{P}$, respectively, to create S. Sometimes a categorization of solutions is performed before starting the updating mechanism, distinguishing between elite and non-elite solutions.

Regarding the updating mechanism component, the following classification can be obtained considering the algorithms provided:

- *Random move*: This strategy moves the solution from its current position within a given neighborhood to another one depending on λ. That is, for a given solution x_i and a step length λ, the random movement is defined by (27).

$$x_i = x_i + \lambda r() \tag{27}$$

  This random movement can also be written as $x_i = x_j + rand(x_i - x_j)$ for a random solution x_j, or equivalently, it can also be expressed by $rand \cdot x_i + (1 - rand)x_j$ as given in (23).
- *Follow the best*: This updating mechanism promotes moving towards the best solution found so far, x_b. (28) expresses its updating formula.

$$x_i = x_i + \lambda(x_b - x_i) \tag{28}$$

- *Follow elite*: This movement promotes that new solutions move towards a solution or a set of solutions having a better objective function value. This is defined in (29).

$$x_i = x_i + \lambda(x_j - x_i), \forall j \in p \; s.t. \; f(x_j) \geq f(x_i) \qquad (29)$$

Notice that along this work, the term 'elite solutions' is used to refer to the subset of solutions with best objective values, but other strategies can be considered as well.

- *Follow its own best*: Each solution keeps track of its best configuration through iterations. By means of it, this strategy takes into account that position termed as x_i^{best} when performing the movement. The updating mechanism is formally expressed in (30).

$$x_i = x_i + \lambda(x_i^{best} - x_i) \qquad (30)$$

- *Follow the neighbor*: This updating mechanism heads the movement towards a solution x_j in the neighborhood of x_i. (31) expresses the updating formula.

$$x_i = x_i + \lambda(x_j - x_i) \qquad (31)$$

- *Alteration*: Different kinds of alteration operators can be introduced and used. For the algorithms discussed above, the alteration operators can be classified into two: (i) replacing: a solution x_i or a part of it is replaced by a newly generated one, (ii) randomly moving: the solution is moved towards randomly selected solution(s). Although it is not present in the studied algorithms, other alteration strategies can be defined such as partially changing a solution as done in the mutation process in genetic algorithms.
- *Guided move*: In this updating mechanism the current solution is influenced by a certain direction; it could be the sum of previous updating directions as in the case of PSO, RIO, GSA, and BA, or based on the location of the best or better solutions and in their neighborhood like in the case of OSA.

Once the updating mechanism finishes, the input function $IF(S')$ component of the pool template updates P using the set of updated solutions in S'. Particularly, for the algorithms considered in this work, this function updates P selecting *the best p from* $P \cup S'$, or *the complete set* S'. On the other hand, the archiving function $AF_2(S')$ component replaces solutions in the archive $\overline{P}$ using some rules. These rules can be different: *replace if better*, *replace with a given probability*, or *replace all*. However, the complete set of selected algorithms uses the same rule, which is *replace if better*.

Considering the previous algorithm descriptions, Table 1 provides a summary of the studied metaheuristics based on the pool template (see Section 2). Notice that S and S' are not specified, since these pools are always constituted by the outputs of the first and second output function and by the updating mechanism, respectively. Additionally, the sizes of P and $\overline{P}$ are considered.

4.2 Analysis of similarity

With the aim of analyzing the similarity between the metaheuristics described above, a comparison of their pool template components is considered. This analysis is divided into three parts:

- The first part is devoted to obtaining a one-to-one relationship comparison of the 15 algorithm components in terms of the proposed pool template. This is done by using a binary code, where 0 means that those components are different, while 1 means that they are equal.
- Considering Table 1 information, it can be seen that the updating mechanism plays a defining role in the functioning of the selected metaheuristics. Therefore, in the second part, a similarity index enabling to compare this component among the different algorithmic approaches is proposed.
- Finally, the last part of this analysis discusses the algorithms case by case and provides overall insights considering all metaheuristics components. This last part provides a degree of novelty that besides pool template components also considers the time frame when the different approaches have been proposed.

4.2.1 Relationship between metaheuristics' pool template components

For eight of nine components of the pool template depicted in Table 1 this similarity index is quite simple to obtain, since we just contemplate if they are different (0) or equal (1). Tables 2, 3, 4, 5, 6 depict these indexes for p, $\overline{p}$, $AF_1(P)$, $AF_2(S')$, and $IF(S')$. Each of these tables is symmetric, and to improve readability we only show the upper part. Notice that tables for $GM(p)$, $OF_1(P)$, and $OF_2(\overline{P})$ are not shown since all the selected algorithms use the same strategies, and therefore each element in these tables is 1. However, in contrast to the previous pool template components, the updating mechanism can be composed by a different number of elements, hence, a more elaborated index needs to be considered. Thus, an updating similarity index aimed at measuring the degree of similarity between algorithms' updating mechanism is provided below.

4.2.2 Quantifying the updating mechanism similarity

This section aims at determining how the degree of similarity between metaheuristic updating mechanism is. In doing so, an updating similarity index is proposed.

Definition 1 Updating similarity index, χ

Table 1 Summary of algorithms based on the pool template

Alg.	p	$\overline{p}$	$GM(p) \rightarrow P$	$OF_1(P) \cup OF_2(\overline{P}) \rightarrow S$	$UM(S) \rightarrow S'$	$AF_1(P) \rightarrow \overline{P} / AF_2(S') \rightarrow \overline{P}$	$IF(S') \rightarrow P$
RS	> 0	0	random	$OF_1(P) = P$	random move	–	the best p from $P \cup S'$
PSO	> 1	p	random	$OF_1(P) = P$	follow the best/follow its own best/	$AF_1(P) = P$	S'
				$OF_2(\overline{P}) = \overline{P}$	guided move based on previous moves	$AF_2(S')$ replace if better	
IWO	> 1	0	random	$OF_1(P) = P$	random move	–	the best p from $P \cup S'$
WPS	> 1	1	random	$OF_1(P) = P$	follow the best	$AF_1(P) = \{x_b\}$	S'
				$OF_2(\overline{P}) = \overline{P} = \{x_b\}$		$AF_2(S')$ replace if better	
FA	> 1	0	random	$OF_1(P) = P$	$\begin{cases} \text{random move,} & \text{if best} \\ \text{follow elite/random move,} & \text{otherwise} \end{cases}$	–	S'
OSA	> 1	0	random	$OF_1(P) = P$	follow the best/guided move based on x_b	–	S'
RIO	> 1	p	random	$OF_1(P) = P$	follow neighborhood elite[3]/follow its own best/	$AF_1(P) = P$	S'
				$OF_2(\overline{P}) = \overline{P}$	guided move based on previous moves	$AF_2(S')$ replace if better	
GSA	> 1	0	random	$OF_1(P) = P$	follow elite/guided move based on previous moves	–	S'
BSO	> 2	$<p/2$	random	$OF_1(P) = P$		$AF_1(P) = \{\overline{p}$ best solutions $\in P\}$	S'
				$OF_2(\overline{P}) = \overline{P}$	$\begin{cases} \text{follow its own best/follow the best,} & \text{if among best solutions} \\ \text{follow the best,} & \text{if middle class solution} \\ \text{random move,} & \text{otherwise} \end{cases}$	$AF_2(S')$ replace if better	
BA	> 1	0	random	$OF_1(P) = P$	follow the best/guided move based on previous moves/random move	–	S'
SOPT	> 1	0	random	$OF_1(P) = P$	guided move based on x_b	–	the best p from $P \cup S'$
CAB	> 1	$<p$	random	$OF_1(P) = P$		$AF_1(P) = \{\overline{p}$ best solutions $\in P\}$	S'
				$OF_2(\overline{P}) = \overline{P}$	$\begin{cases} \text{random move} & \text{if it is elite solution} \\ \text{follow the best or follow the nearest elite [3],} & \text{otherwise} \end{cases}$	$AF_2(S')$ replace if better	
ATA	> 1	p	random	$OF_1(P) = P$		$AF_1(P) = P$	S'
				$OF_2(\overline{P}) = \overline{P}$	$\begin{cases} \text{follow the best/follow its own best} & \text{if below a limit} \\ \text{alteration (follow a random solution),} & \text{otherwise} \end{cases}$	$AF_2(S')$ replace if better	

Table 1 (continued)

Alg.	p	$\overline{p}$	$GM(p) \to P$	$OF_1(P) \cup OF_2(\overline{P}) \to S$	$UM(S) \to S'$	$AF_1(P) \to \overline{P} / AF_2(S') \to \overline{P}$	$IF(S') \to P$
BH	> 1	1	random	$OF_1(P) = P$ $OF_2(\overline{P}) = \overline{P} = \{x_b\}$	$\begin{cases} \text{follow the best,} & \text{if distance to the best} \geq \text{parameter} \\ \text{alteration,} & \text{otherwise} \end{cases}$	$AF_1(P) = \{x_b\}$ $AF_2(S')$ replace if better	S'
AMO	> 1	1	random	$OF_1(P) = P$ $OF_2(\overline{P}) = \overline{P} = \{x_b\}$	follow the neighbor/alteration based on 2 randomly chosen solutions	$AF_1(P) = \{x_b\}$ $AF_2(S')$ replace if better	S'

[3]Particular case of *follow elite solution*

The updating similarity index of a metaheuristic i with respect to another metaheuristic j is the number of elements in the updating mechanism used in i which are similar to the elements in the updating mechanism used in j divided by the total number of elements in the updating mechanism used in i.

For instance, if an algorithm A_1 uses n elements in its updating mechanism and another algorithm A_2 uses m elements, then the similarity index of A_1 compared with A_2 is given by (32).

$$\chi_{A_1 A_2} = \frac{|CA_1 \cap CA_2|}{n} \tag{32}$$

where CA_i is a set of elements in the updating mechanism used by A_i for $i = 1, 2$ and $|CA_1 \cap CA_2|$ is the number of common elements for the two algorithms. Note that $\chi_{A_1 A_2}$ is not necessarily equal to $\chi_{A_2 A_1}$, it is not commutative. Furthermore, if $\chi_{A_1 A_2} = 1$ it means all the updating elements of algorithm A_1 exist in the second algorithm; hence it is possible to say A_1 is at least a special case of A_2.

Theorem 1 *If A_1 is a special case of A_2 and $\chi_{A_1 A_2} = \chi_{A_2 A_1}$, then the two algorithms have the same updating mechanism.*

Proof Suppose the number of updating elements in A_1 and A_2 are n and m, respectively.

Since $\chi_{A_1 A_2} = \chi_{A_2 A_1} \Rightarrow \frac{|CA_1 \cap CA_2|}{n} = \frac{|CA_2 \cap CA_1|}{m}$
$\Rightarrow n = m$, intersection of sets is a commutative operator.

However, since A_1 is a special case of A_2, $\frac{|CA_1 \cap CA_2|}{n} = 1$ implying $|CA_1 \cap CA_2| = n$

Hence, the two algorithms have the same number of updating elements and these elements are the same.

Therefore, the algorithms have the same updating mechanism. □

Theorem 2 *The special case relation is transitive.*

Proof Suppose A_1 is a special case of A_2 and A_2 is a special case of A_3.

A_1 is a special case of $A_2 \Rightarrow \chi_{A_1 A_2} = \frac{|CA_1 \bigcap CA_2|}{|CA_1|} = 1.$
$\Rightarrow |CA_1 \bigcap CA_2| = |CA_1|$, hence $A_1 \bigcap A_2 = A_1$.

A_2 is a special case of $A_3 \Rightarrow \chi_{A_2 A_3} = \frac{|CA_2 \bigcap CA_3|}{|CA_2|} = 1.$
$\Rightarrow |CA_2 \bigcap CA_3| = |CA_2|$, hence $A_2 \bigcap A_3 = A_2$.

From these $A_1 \bigcap A_3 = A_1$, this implies $\chi_{A_1 A_3} = \frac{|CA_1 \bigcap CA_3|}{|CA_1|} = 1.$ □

Note that commutativity of the updating similarity index holds for two algorithms if they have the same number of updating elements.

The updating similarity index helps to see the degree of similarity on the updating mechanism of a given algorithm compared to another algorithm. Table 7 shows the updating similarity index of the algorithms studied. For instance, if we compare WPS against PSO, WPS has one updating element which is following the best solution and that updating element is the one used on PSO. Hence the similarity index is 1. However, if WPS is compared with RS, the similarity index will be 0. Similarly, when comparing RIO against BSO, RIO uses three updating elements from which one is in common with BSO, hence the similarity index of RIO compared to BSO is 0.33. Then, if we compare BSO with RIO, BSO has also three elements from which one is in common with RIO, so the updating similarity index of BSO regarding RIO is 0.33 again.

The last step consists of condensing every single index corresponding to a component of the pool template and obtaining a composed similarity index that depicts the similarity of every pair of metaheuristics. With this aim, the nine indexes can be summed and normalized. However, it is worth mentioning that the performance of any metaheuristic is highly influenced by its diversification and intensification capabilities. That in turn depends on the updating mechanism as it is the component involved in the movement and treatment of a solution resulting into a new one. Hence, in our analysis, we consider the similarity among metaheuristics when more weight is put on the updating mechanism of the algorithms than in the remaining components of the pool template. In this regard, the updating mechanism has been weighted with about a half of the total weight (55.5%) and the remaining components with the remaining weight (5.5% each one). Thus, Table 8 has been obtained to define the similarity of the set of metaheuristics.

Based on the reported values, three different similarity ranges can be defined. Values belonging to the range [0.7, 0.8) have been marked with boldunderlined, values in the range [0.8, 0.9) with italic, and values in the range [0.9, 1] with boldoverlined. Notice that we have rounded numbers to one decimal to make this classification. These ranges ease the visual identification of relevant, quite relevant, and very relevant similarities, respectively. Values below these ranges are not considered enough relevant to extract reliable conclusions, and the diagonal values are not considered since they represent the relation between each algorithm and itself.

Like in the individual indexes tables, in Table 8 the algorithm in the row is compared with the algorithm in the column. Note that, the algorithms are ordered in their year of introduction starting from RS introduced in 1958, and finalizing with AMO, introduced in 2014. Also, it is worth indicating that when a new metaheuristic method needs to be incorporated afterwards, the current values does not change. Only a new row and column related to the method to be incorporated have to be added. With regards to the table entries, a value close to 1 in row p and column q implies that the algorithm in the row is similar or a special case of the algorithm in the column. On the other hand, if the entry in row q and column p is also close to 1, the two algorithms can be considered practically equal. The higher the values, the more similar they are. For example, looking at the $3rd$ row and the $1st$ column the value is 0.94 and also at the $1st$ row and column 3. Hence, the two algorithms (RS and IWO) are practically the same. If we look at row 4 and column 2 the entry is 0.89 but in row 2 and column 4 the

Table 2 Similarity index regarding the size of the pool of solutions P (i.e., p)

	RS	PSO	IWO	WPS	FA	OSA	RIO	GSA	BSO	BA	SOPT	CAB	ATA	BH	AMO
RS	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PSO		1	1	1	1	1	1	1	0	1	1	1	1	1	1
IWO			1	1	1	1	1	1	0	1	1	1	1	1	1
WPS				1	1	1	1	1	0	1	1	1	1	1	1
FF					1	1	1	1	0	1	1	1	1	1	1
OSA						1	1	1	0	1	1	1	1	1	1
RIO							1	1	0	1	1	1	1	1	1
GSA								1	0	1	1	1	1	1	1
BSO									1	0	0	0	0	0	0
BA										1	1	1	1	1	1
SOPT											1	1	1	1	1
CAB												1	1	1	1
ATA													1	1	1
BH														1	1
AMO															1

Table 3 Similarity index regarding the size of the archive $\overline{P}$ (i.e., $\overline{p}$)

	RS	PSO	IWO	WPS	FA	OSA	RIO	GSA	BSO	BA	SOPT	CAB	ATA	BH	AMO
RS	1	0	1	0	1	1	0	1	0	1	1	0	0	0	0
PSO		1	0	0	0	0	1	0	0	0	0	0	1	0	0
IWO			1	0	1	1	0	1	0	1	1	0	0	0	0
WPS				1	0	0	0	0	0	0	0	0	0	1	1
FA					1	1	0	1	0	1	1	0	0	0	0
OSA						1	0	1	0	1	1	0	0	0	0
RIO							1	0	0	0	0	0	1	0	0
GSA								1	0	1	1	0	0	0	0
BSO									1	0	0	0	0	0	0
BA										1	1	0	0	0	0
SOPT											1	0	0	0	0
CAB												1	0	0	0
ATA													1	0	0
BH														1	1
AMO															1

Table 4 Similarity index regarding the first archiving function (i.e., $AF_1(P)$)

	RS	PSO	IWO	WPS	FA	OSA	RIO	GSA	BSO	BA	SOPT	CAB	ATA	BH	AMO
RS	1	0	1	0	1	1	0	1	0	1	1	0	0	0	0
PSO		1	0	0	0	0	1	0	0	0	0	0	1	0	0
IWO			1	0	1	1	0	1	0	1	1	0	0	0	0
WPS				1	0	0	0	0	0	0	0	0	0	1	1
FF					1	1	0	1	0	1	1	0	0	0	0
OSA						1	0	1	0	1	1	0	0	0	0
RIO							1	0	0	0	0	0	1	0	0
GSA								1	0	1	1	0	0	0	0
BSO									1	0	0	1	0	0	0
BA										1	1	0	0	0	0
SOPT											1	0	0	0	0
CAB												1	0	0	0
ATA													1	0	0
BH														1	1
AMO															1

entry is 0.52. Thus, WPS is a special case of PSO but PSO is different from WPS.

In order to ease the drawing of conclusions regarding the relationship among the algorithms, Figs. 3, 4 and 5 represent them. The first focuses on the highest similarity values, i.e., very relevant similarities. The second depicts the next range of values considering quite relevant similarities, and the third represents the last range of values, i.e., the relevant similarities. Pointed arrows represent that one algorithm is a special case of the other. Double pointed arrows indicate that both algorithms are special cases of each other with the same degree of similarity. Moreover, if the degree of similarity is near to one, then we can consider them equivalent or almost. This is the case of IWO and RS as shown in Fig. 3.

Considering Fig. 3, it is possible to come to the following conclusions:

- SOPT is a special case of the OSA
- WPS is a special case of PSO, ATA, BH, and CAB
- BH is a special case of ATA

Table 5 Similarity index regarding the second archiving function (i.e., $AF_2(S')$)

	RS	PSO	IWO	WPS	FA	OSA	RIO	GSA	BSO	BA	SOPT	CAB	ATA	BH	AMO
RS	1	0	1	0	1	1	0	1	0	1	1	0	0	0	0
PSO		1	0	1	0	0	1	0	1	0	0	1	1	1	1
IWO			1	0	1	1	0	1	0	1	1	0	0	0	0
WPS				1	0	0	1	0	1	0	0	1	1	1	1
FA					1	1	0	1	0	1	1	0	0	0	0
OSA						1	0	1	0	1	1	0	0	0	0
RIO							1	0	1	0	0	1	1	1	1
GSA								1	0	1	1	0	0	0	0
BSO									1	0	0	1	1	1	1
BA										1	1	0	0	0	0
SOPT											1	0	0	0	0
CAB												1	1	1	1
ATA													1	1	1
BH														1	1
AMO															1

Table 6 Similarity index regarding the input function (i.e., $IF(S')$)

	RS	PSO	IWO	WPS	FA	OSA	RIO	GSA	BSO	BA	SOPT	CAB	ATA	BH	AMO
RS	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
PSO		1	0	1	1	1	1	1	1	1	0	1	1	1	1
IWO			1	0	0	0	0	0	0	0	1	0	0	0	0
WPS				1	1	1	1	1	1	1	0	1	1	1	1
FA					1	1	1	1	1	1	0	1	1	1	1
SOPT						1	1	1	1	1	0	1	1	1	1
RIO							1	1	1	1	0	1	1	1	1
GSA								1	1	1	0	1	1	1	1
BSO									1	1	0	1	1	1	1
BA										1	0	1	1	1	1
SOPT											1	0	0	0	0
CAB												1	1	1	1
ATA													1	1	1
BH														1	1
AMO															1

- IWO can be considered nearly equivalent to RS, and they are special cases of FA and BA

It can be observed from the resulting relationship scheme that PSO and BH are not strongly related. This may be seen contradictory with regards to the findings provided by Piotrowski et al. (2014), where PSO and BH were compared to investigate the novelty of the second. In that regard, it can be firstly remarked that we considered the first PSO presented by Kennedy and Eberhart (1995), while Piotrowski et al. (2014) analyzed a later one with inertia weights included. Moreover, for the restarting procedure considered in BH, Piotrowski et al. (2014) referenced other PSO variants that considered restart but no direct comparison is provided. Since the pool template heavily relies on complete algorithms in terms of components, it can be said that although both algorithms PSO and BH share some similarities, they are not the same as can be distinguished from Table 1. In this sense, a further study including the PSO variants components mentioned in Piotrowski et al. (2014) should be conducted for being able to quantitatively

Table 7 Updating similarity index

	RS	PSO	IWO	WPS	FA	OSA	RIO	GSA	BSO	BA	SOPT	CAB	ATA	BH	AMO
RS	1	0	1	0	1	0	0	0	1	1	0	1	0	0	0
PSO	0	1	0	0.33	0	0.33	0.67	0.33	0.67	0.67	0	0.33	0.67	0.33	0
IWO	1	0	1	0	1	0	0	0	1	1	0	1	0	0	0
WPS	0	1	0	1	0	1	0	0	1	1	0	1	1	1	0
FA	0.50	0	0.50	0	1	0	0.50	0.50	0.50	0.50	0	1	0	0	0
OSA	0	0.50	0	0.50	0	1	0	0	0.50	0.50	0.50	0.50	0.50	0.50	0
RIO	0	0.67	0	0	0.33	0	1	0.67	0.33	0.33	0	0.33	0.33	0	0
GSA	0	0.50	0	0	0.50	0	1	1	0	0.50	0	0.50	0	0	0
BSO	0.33	0.67	0.33	0.33	0.30	0.33	0.33	0	1	0.67	0	0.67	0.67	0.33	0
BA	0.33	0.67	0.33	0.33	0.33	0.33	0.33	0.33	0.67	1	0	0.67	0.33	0.33	0
SOPT	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
CAB	0.33	0.33	0.33	0.33	0.67	0.33	0.33	0.33	0.67	0.67	0	1	0.33	0.33	0
ATA	0	0.67	0	0.33	0	0.33	0.33	0	0.67	0.33	0	0.33	1	0.67	0.33
BH	0	0.50	0	0.50	0	0.50	0	0	0.50	0.50	0	0.50	1	1	0.50
AMO	0	0	0	0	0	0	0	0	0	0	0	0	0.50	0.50	1

Table 8 Final similarity index with weights. Boldunderlined corresponds to values belonging to the range [0.7, 0.8), Bold to values in the range [0.8, 0.9), and Boldoverlined to values in the range [0.9, 1], identifying relevant, quite relevant, and very relevant similarities, respectively

	RS	PSO	IWO	WPS	FA	OSA	RIO	GSA	BSO	BA	SOPT	CAB	ATA	BH	AMO
RS	1	0.17	$\overline{\mathbf{0.94}}$	0.17	$\overline{\mathbf{0.89}}$	0.33	0.17	0.33	$\underline{\mathbf{0.72}}$	$\overline{\mathbf{0.89}}$	0.39	$\underline{\mathbf{0.72}}$	0.17	0.17	0.17
PSO	0.17	1	0.22	0.52	0.28	0.46	*0.82*	0.46	0.65	0.65	0.22	0.52	**0.82**	0.52	0.33
IWO	$\overline{\mathbf{0.94}}$	0.22	1	0.22	$\overline{\mathbf{0.94}}$	0.39	0.22	0.39	$\underline{\mathbf{0.72}}$	$\overline{\mathbf{0.94}}$	0.44	**0.78**	0.22	0.22	0.22
WPS	0.17	$\overline{\mathbf{0.89}}$	0.22	1	0.28	**0.83**	0.33	0.28	**0.83**	**0.83**	0.22	$\overline{\mathbf{0.89}}$	$\overline{\mathbf{0.89}}$	$\overline{\mathbf{1}}$	0.44
FA	0.61	0.28	0.67	0.28	1	0.44	0.56	$\underline{\mathbf{0.72}}$	0.50	$\underline{\mathbf{0.72}}$	0.39	**0.83**	0.28	0.28	0.28
OSA	0.33	0.56	0.39	0.56	0.44	1	0.28	0.44	0.50	$\underline{\mathbf{0.72}}$	0.67	0.56	0.56	0.56	0.28
RIO	0.17	**0.82**	0.22	0.33	0.46	0.28	1	0.65	0.46	0.46	0.22	0.52	0.63	0.33	0.33
GSA	0.33	0.56	0.39	0.28	$\underline{\mathbf{0.72}}$	0.44	**0.83**	1	0.22	$\underline{\mathbf{0.72}}$	0.39	0.56	0.28	0.28	0.28
BSO	0.35	0.65	0.35	0.46	0.39	0.41	0.46	0.22	1	0.59	0.17	$\underline{\mathbf{0.71}}$	0.65	0.46	0.28
BA	0.52	0.65	0.57	0.46	0.63	0.63	0.46	0.63	0.59	1	0.39	0.65	0.46	0.46	0.28
SOPT	0.39	0.22	0.44	0.22	0.39	$\overline{\mathbf{0.94}}$	0.22	0.39	0.17	0.39	1	0.22	0.22	0.22	0.22
CAB	0.35	0.52	0.41	0.52	0.65	0.46	0.52	0.46	$\underline{\mathbf{0.71}}$	0.65	0.22	1	0.52	0.52	0.33
ATA	0.17	**0.82**	0.22	0.52	0.28	0.46	0.63	0.28	0.65	0.46	0.22	0.52	1	$\underline{\mathbf{0.71}}$	0.52
BH	0.17	0.61	0.22	$\underline{\mathbf{0.72}}$	0.28	0.56	0.33	0.28	0.56	0.56	0.22	0.61	$\overline{\mathbf{0.89}}$	1	$\underline{\mathbf{0.72}}$
AMO	0.17	0.33	0.22	0.44	0.28	0.28	0.33	0.28	0.28	0.28	0.22	0.33	0.61	$\underline{\mathbf{0.72}}$	1

comment on BH novelty and its relationship with other PSO variants components.

Considering the next level of relevance regarding similarity (Fig. 4), it is possible to come to the following conclusions:

- IWO and FA are special cases of CAB
- WPS is a special case of OSA, BSO and BA
- GSA is a special case of RIO
- PSO is a special case of ATA and vice-versa
- PSO is a special case of RIO and vice-versa

Finally, if the last level of relevance regarding similarity is considered (Fig. 5), it is possible to come to the following conclusions:

- RS is a special case of CAB and BSO
- IWO is a special case of BSO
- CAB is a special case of BSO and vice-versa
- AMO is a special case of BH and vice-versa
- ATA is a special case of BH

- BH is a special case of WPS
- GSA, FA, and OSA are special cases of BA
- FA is a special case of GSA and vice-versa

As can be observed in Figs. 3, 4 and 5, a global view of the similarity among algorithms permits determining which algorithms are related to which. This could be interesting when deciding the algorithm to solve a particular problem. For instance, if a given algorithm has been tried on an optimization problem without success, applying equivalent or a sub-type of a it may not lead to better results (unless there are random components with different seeds).

Furthermore, these algorithms have been proposed at different dates and it is relevant to assess the novelty of newer proposals. Therefore, the year of publication has also been put into consideration. Thus, each algorithm is compared against older ones considering the lower half (under the diagonal) of the table and using the other half to check either the algorithm is a special case or the same with the algorithm in the column. Figures 6, 7 and 8 depict the relationships between newer and older algorithms, which are summarized as follows:

- BH is a special case of ATA with a very relevant similarity
- IWO can be considered equivalent to RS
- WPS is a special case of PSO with a very relevant similarity
- SOPT is a special case of OSA with a very relevant similarity
- ATA and RIO are a special cases of PSO with a quite relevant similarity
- GSA is a special case of RIO with a quite relevant similarity
- GSA is a special case of FA with a relevant similarity
- AMO is a special case of BH with a relevant similarity
- BH is a special case of WPS with a relevant similarity
- CAB is a special case of BSO with a relevant similarity

According to the very relevant similar cases, BH, IWO, WPS, and SOPT did not contribute (in terms of bringing novelty) to a high degree considering the previously proposed algorithms. If quite relevant similarities are considered, then we can extend the list of algorithms not contributing to a high degree with ATA, GSA and RIO. Finally, if relevant similarities are taken into account, AMO and CAB can be aggregated in the list. Details are discussed in the next section.

4.3 Case-by-case discussion

Considering the previous analysis and the algorithms' development along time from old to recent, it is possible to assess their novelty in terms of their components. In this sense, besides the novelty of considering a nature process inspiration, some approaches are similar to older ones and, hence, not so algorithmically novel.

From the set of discussed algorithms, the oldest proposal is RS. This uses a *random move* to update solutions. Later, PSO introduced *follow its own best* and *follow the best* operators, and a *guided move* in which previous updating directions are considered. Additionally, it introduced the use of memory, i.e., the archive component and their management functions.

IWO proposed a *random move* as updating component which was used previously in RS and the remaining pool template components do not differ from previous ones. This is the reason why IWO is considered nearly equivalent to RS. In the case of the WPS algorithm, it reused the *follow the best* updating mechanism previously used by PSO. The difference regarding previous algorithms was the size of the memory, which is always a unique solution (the best solution) and its initialization to the best solution obtained by the generation method component.

FA reused the *random move* and introduced the *follow elite* solutions operator, involving a certain novelty (within continuous optimization and the described set of swarm-based metaheuristics) as this was the first time to be proposed. The remaining pool template components are not new, although the combination of not using memory and the updating of the solutions in the pool with all the new

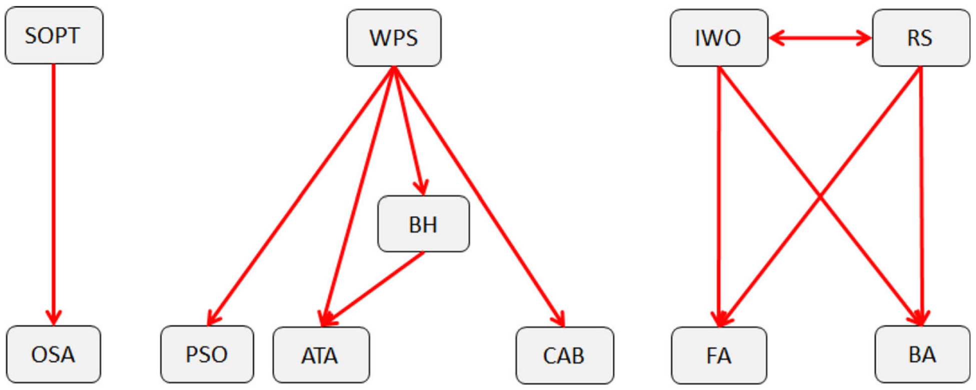
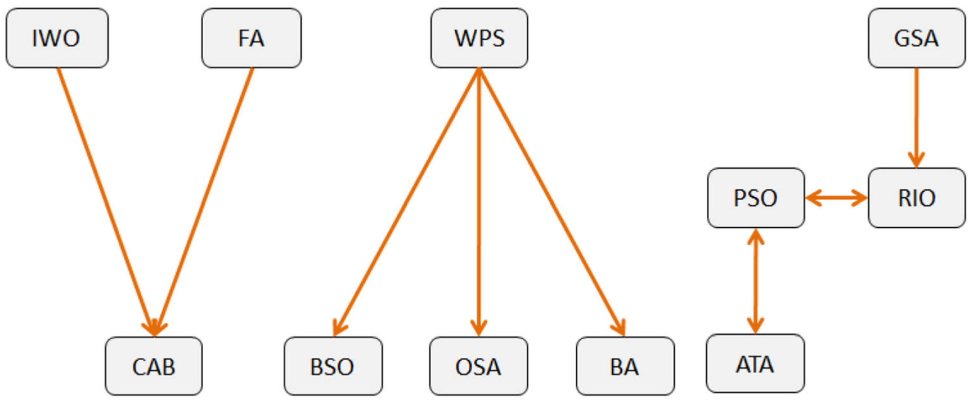


Fig. 3 Algorithms' relationship based on the very relevant similarity indexes

Fig. 4 Algorithms' relationship based on quite relevant similarity indexes



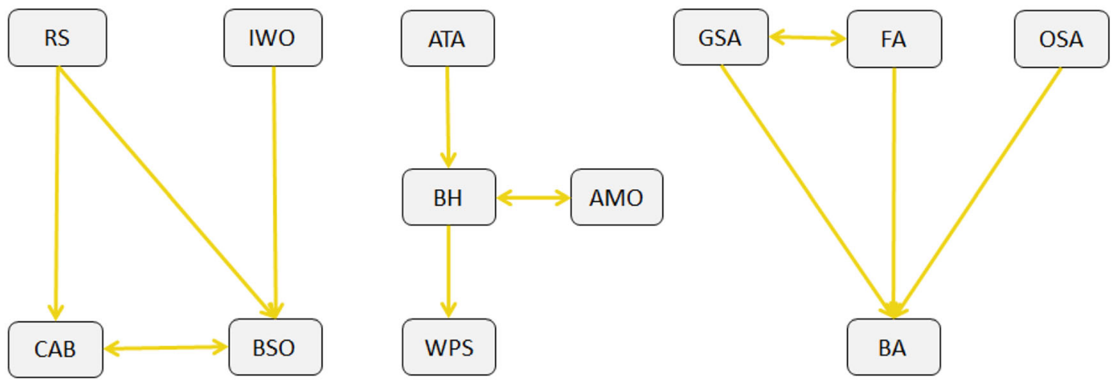


Fig. 5 Algorithms' relationship based on relevant similarity indexes

solutions is distinctive (regarding previously presented algorithms) at the publication date.

OSA uses *follow the best* solution and *guided move* in the direction of the best solution from the origin. This is a new updating movement while the remaining pool template components are combinations of previous algorithms. In this sense, this metaheuristic seems to introduce some degree of novelty. However, the guided search introduced does not have a strong motivation and is not clear if it is helping for intensification or diversification. Furthermore, the suitability of the move depends on the relative location of the current best solution from the solution which is going to be updated, x_i. Hence, we should consider carefully the novelty of this algorithm. If we consider the introduced guided search marginal, then OSA will be a special case of PSO.

The RIO algorithm uses a similar updating mechanism to the one used by PSO, but changes *follow the best* operator by *following elite* solution in the neighborhood. The remaining pool template components are equal to PSO components. Moreover, the *following elite* solution movement has been applied before by FA, hence, to some extent, RIO could be considered as a hybrid of PSO and FA. Another algorithm with a similar updating mechanism is GSA. This uses *follow elite* solutions and *guided move*. However, it does not use an archive. As indicated in Fig. 7, it is a special case of RIO and therefore, also can be considered a hybrid of PSO and FA. Actually, it has a relevant similarity with FA as shown in Fig. 8.

BSO uses a data structure where solutions are put into different categories. The minimum number of solutions in P, the size of the archive, and the way it is initialized are different from the previous algorithms. However, its updating mechanism elements are *random move*, *follow its own best* and *follow the best* solution, which are movements previously proposed. Hence, it introduced some novelties regarding P, $\overline{P}$, and $AF_1(P)$, although its updating mechanism elements are a combination of the ones in PSO and RS.

BA uses a *guided move* based on previous updating directions and *follow the best*, like PSO, although it also applies a *random move* and does not use any memory. Thus, its pool template components are previously proposed by PSO and RS.

Regarding SOPT, it moves the solution to the neighborhood of the best solution. This is a *guided move* based on the best solution, previously used in OSA. Actually, it is a special case of OSA with a very relevant similarity and does not introduce any novelty considering the previous algorithms proposed.

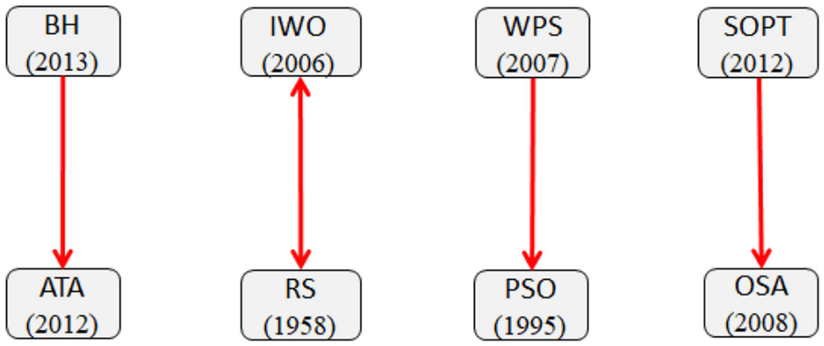


Fig. 6 Algorithms' relationship considering publication year and the very relevant similarity indexes

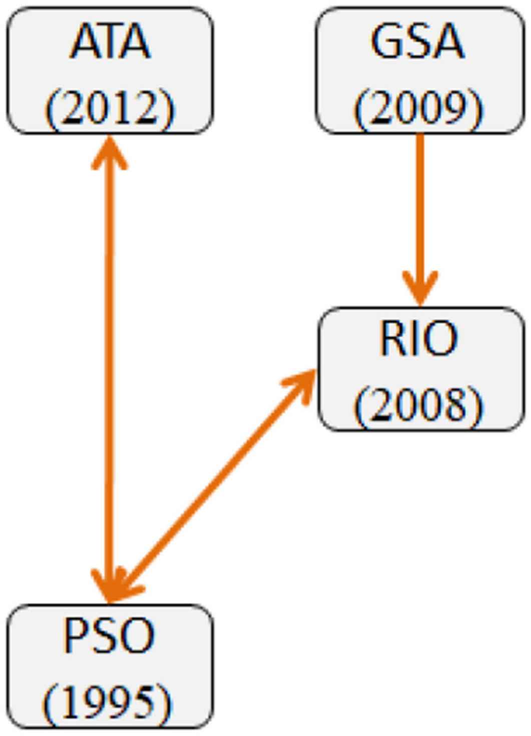


Fig. 7 Algorithms' relationship considering publication year and the quite relevant similarity indexes

Later, CAB was proposed, which is a particular case of BSO with relevant similarity because their updating mechanism implementations are based on similar concepts and also their archiving and input functions are the same. Nevertheless, the size of the archive is different from previous algorithms, so it introduced a very slight novelty.

In ATA, an additional *alteration* operator is introduced for relocating the solution to another neighborhood. This is a novelty, although the remaining pool template components are equal to PSO, reason why it has a quite relevant similarity index with PSO. Like ATA, BH is an algorithm with *alteration* operator in the updating mechanism. Actually, it is considered a special case of ATA with a very relevant similarity index since they share the remaining pool template components and also the *follow the best* element of the updating mechanism. Alteration aside, BH is a special case of WPS.

Algorithm AMO also uses an *alteration* operator and introduces a new element in the updating mechanism, *follow the neighbor*. That is the difference regarding BH, which uses a *follow the best* element instead of *alteration*. This makes AMO a special case of BH with a relevant similarity.

4.4 Overall insights

Despite the similarities found and presented, the number of works proposing these metaheuristics and citing the corresponding similar algorithm is quite reduced. Only BA, ATA and RIO cite PSO.

In order to establish a factual measure of the novelty introduced by each algorithm along the publication years, a degree of novelty can be defined. Notice that we are always in the framework of the 15 metaheuristics selected and, therefore, the novelty is with respect to this set of algorithms. Thus, for each algorithm and each pool template component, we check if the elements used within those components are novel. If any other previous algorithm has used it before, then the component is set to 0. Otherwise, a 1 is assigned. For the updating mechanism, the proportion of new movements is considered (see Table 9). Finally, for each algorithm, the same weights used in Sect. 4 are assigned to its components, i.e., 55.5% to the updating mechanism and 5.5% for the remaining components. Thus, a final degree of novelty can be calculated as the weighted sum. Figure 9 depicts this degree for each algorithm.

Because of the reuse of existing algorithms under new names, the field has been invaded with many proposals where useful studies can be conducted instead. Interesting research issues, like the description or definition of the metaheuristic components have not been explored enough whereas the focus of many researchers has been put on mimicking a given scenario to introduce a 'new' algorithm. One of the reasons is that the researchers who proposed new algorithms did not do sufficient literature study.

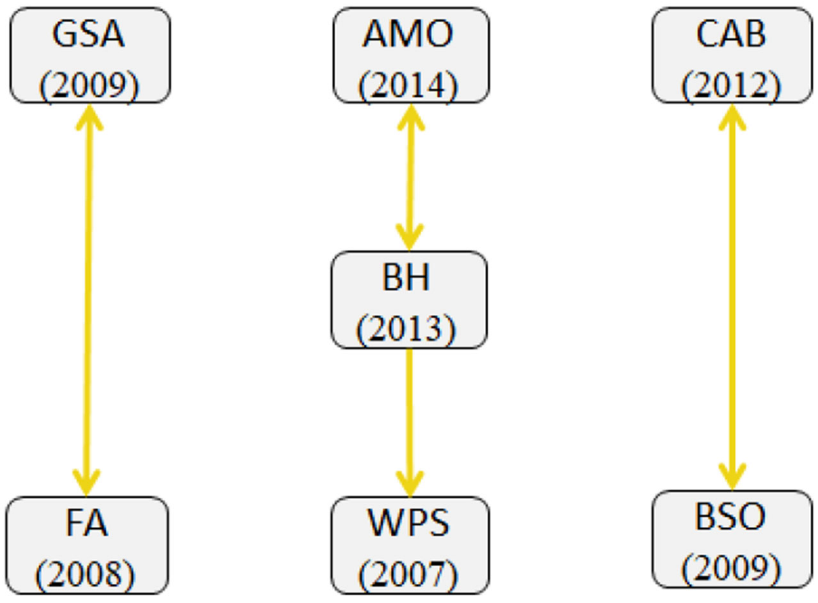


Fig. 8 Algorithms' relationship considering publication year and the relevant similarity indexes

Hence, if a researcher wants to introduce a new metaheuristic, a good bibliography study of existing approaches needs to be done. Additionally, the use of appropriate or standard terminology for explaining and describing a metaheuristic is advisable.

Different applications used these 'new' algorithms successfully. However, a good performance on benchmark problems does not necessarily demonstrate the novelty of a given algorithm. If an algorithm is composed of other algorithms' elements it may outperform others, but that makes it a hybrid algorithm, not a new one.

5 Conclusions

The novelty introduced by a new metaheuristic or a variant is a research issue that requires relating and analysing its (possibly incremental) contribution with regards to the state-of-the-art. Despite of this observation, many new algorithms are proposed and used persistently but lack of design analysis. Even though a new metaheuristic can be a good contribution, researchers should be able to define its components appropriately. In order to do that, they should consider previous related approaches in the literature while assessing the algorithmic novelty of their contribution. Thus, in this paper, a pool template is proposed for

Table 9 Novelty of each pool template component for each algorithm considering year of proposal

Algorithm	p	$\overline{p}$	GM(p)	$OF_1(P)$	$OF_2(\overline{P})$	UM(S)	$AF_1(P)$	$AF_2(S')$	IF(S')
RS	1	1	1	1	1	1	1	1	1
PSO	1	1	0	0	0	1	1	1	1
IWO	0	0	0	0	0	0	0	0	0
WPS	0	1	0	0	0	0	1	0	0
FA	0	0	0	0	0	0.50	0	0	0
OSA	0	0	0	0	0	0.50	0	0	0
RIO	0	0	0	0	0	0.33	0	0	0
GSA	0	0	0	0	0	0	0	0	0
BSO	1	1	0	0	0	0	1	0	0
BA	0	0	0	0	0	0	0	0	0
SOPT	0	0	0	0	0	0	0	0	0
CAB	0	1	0	0	0	0	0	0	0
ATA	0	0	0	0	0	0.33	0	0	0
BH	0	0	0	0	0	0	0	0	0
AMO	0	0	0	0	0	0.50	0	0	0

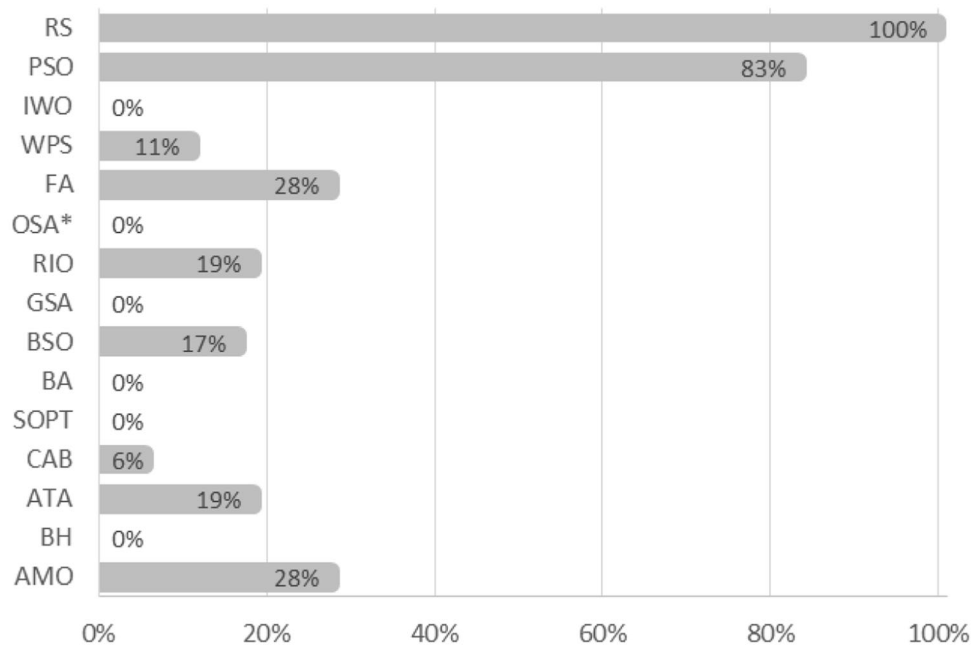


Fig. 9 Degree of novelty. Asterisks (*) indicates that even though there is one new updating element, it hinders the search process

identifying, classifying, and analyzing metaheuristics at a component level.

To exemplify its use, several recent swarm-based metaheuristics applied in continuous optimization have been compared and analysed by means of our approach. The results confirm that indeed there is an issue of novelty at a design level where the majority of these algorithms are similar in terms of components and some of them can be defined as a special case of previous algorithms. Thus, future algorithms regardless of their inspirational source can be accompanied by a proper algorithmic breakdown study that permits identifying the design contribution of them. Moreover, in cases where two or more algorithms present a design difference already identified through the pool template, an empirical study can be developed to measure the contribution of such design difference.

A methodology is proposed to measure the similarity and relationship among algorithms. This is useful when deciding and designing the algorithm to solve a particular problem. Additionally, the novelty of an algorithm regarding previously proposed metaheuristics is quantified using some terms and approaches based on the pool template. This methodology and analysis can be extended to any other group of metaheuristics.

Furthermore, it is also noteworthy to indicate that some novelty can come by the combination of already defined components in other algorithms but not yet put together in an algorithm. In this regard, it is necessary to analyse if that novelty is enough to justify a new name for the algorithm (nature-inspired or not) instead of a variant from the other algorithms where the components exist. For example, Camacho Villalón et al. (2020) compare the Grey Wolf Algorithm, Firefly Algorithm, and Bat Algorithm with different parts of different algorithms (PSO, variants of PSO, and Simulated Annealing). This way, they indicate that the analyzed algorithms can be built by taking components of already existing algorithms and putting them together. This, therefore, motivates (as in Piotrowski et al. (2014)) the use of a methodology, like the pool template proposed in this work, to systematically analyse and compare metaheuristic algorithms in terms of their defining components, as well as quantify, in case there is a combination, how each algorithm component contributes/relates to the new proposed algorithm.

In the same vein, some possible future works include:

- Developing a similar study for metaheuristics applied to discrete optimization problems or for specific types of metaheuristics such as single-point-based ones, among others.
- Preparing a compiled set of updating mechanism elements categorization needs to be done. Based on that it will be easy for researchers to see what mechanisms exist (especially for learners).
- A mathematical analysis of updating mechanism implementations based on algorithms' convergence and similarity.

Acknowledgements This work was partially supported by the Ministry of Science, Innovation and Universities of the Government of Spain AEI/FEDER through the project RTI2018-095197-B-I00. Surafel Luleseged Tilahun would like to acknowledge the TWAS-DFG Cooperation Visits Programme for the support to visit University of Hamburg for a research collaboration.

References

Akbari R, Mohammadi A, Ziarati K (2009) A powerful bee swarm optimization algorithm. In: IEEE 13th international multitopic conference (INMIC), pp 1 – 6, https://doi.org/10.1109/INMIC.2009.5383155

Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Computing Surv 35(3):268–308. https://doi.org/10.1145/937503.937505

Brooks SH (1958) A discussion of random methods for seeking maxima. Op Res 6(2):244–251. https://doi.org/10.1287/opre.6.2.244

Camacho Villalón CL, Stützle T, Dorigo M (2020) Grey wolf, firefly and bat algorithms: three widespread algorithms that do not contain any novelty. In: Dorigo M, Stützle T, Blesa MJ, Blum C, Hamann H, Heinrich MK, Strobel V (eds) Swarm intelligence. Springer International Publishing, Cham, pp 121–133

Chen T, Wang Y, Li J (2012) Artificial tribe algorithm and its performance analysis. J Softw 7:651–656. https://doi.org/10.4304/jsw.7.3.651-656

Cuevas E, González M, Zaldivar D, Pérez-Cisneros M, G G, (2012) An algorithm for global optimization inspired by collective animal behavior. Discrete Dyn Nat Soc 2012638275:24. https://doi.org/10.1155/2012/638275

Duarte A, Laguna M, Martí R (2018) Introduction to spreadsheet modeling and metaheuristics. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-68119-1_1

Greistorfer P, Voß S (2005) Controlled pool maintenance for metaheuristics. In: Rego C, Alidaee B (ed) Metaheuristic optimization via memory andevolution. Kluwer, Boston, pp 387—424. https://doi.org/10.1007/0-387-23667-8_18

Hasançebi O, Azad SK (2012) An efficient metaheuristic algorithm for engineering optimization: SOPT. Int J Optim Civ Eng 2:479–487

Hatamlou A (2013) Black hole: a new heuristic optimization approach for data clustering. Inform Sci 222:175–184. https://doi.org/10.1016/j.ins.2012.08.023

Havens TC, Spain CJ, Salmon NG, Keller JM (2008) Roach infestation optimization. In: IEEE Swarm Intelligence Symposium, September 21–23, 2008, St. Louis MO, USA, pp 1–7, https://doi.org/10.1109/SIS.2008.4668317

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Neural Networks, 1995. Proceedings., IEEE International Conference on, vol 4, pp 1942–1948 vol.4, https://doi.org/10.1109/ICNN.1995.488968

Laguna M (2016) Editor's note on the MIC 2013 special issue of the Journal of Heuristics (Volume 22, Issue 4, August 2016). J Heuristics 22(5):665–666. https://doi.org/10.1007/s10732-016-9318-5

Li X, Zhang J, Yin M (2014) Animal migration optimization: an optimization algorithm inspired by animal migration behavior. Neural Computing Appl 24:1867–1877. https://doi.org/10.1007/s00521-013-1433-8

Mehrabian AR, Lucas C (2006) A novel numerical optimization algorithm inspired from weed colonization. Ecol Inform 1:355–366. https://doi.org/10.1016/j.ecoinf.2006.07.003

Piotrowski AP, Napiorkowski JJ, Rowinski PM (2014) How novel is the "novel" black hole optimization approach? Inform Sci 267:191–200

Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inform Sci 179:2232–2248. https://doi.org/10.1016/j.ins.2009.03.004

Rastrigin LA (1963) The convergence of the random search method in the extremal control of a many parameter system. Autom Remote Control 24:1337–1342

Sörensen K (2015) Metaheuristics - the metaphor exposed. Int Trans Op Res 22:3–18. https://doi.org/10.1111/itor.12001

Sörensen K, Glover F (2013) Metaheuristics. In: Gass SI, Fu M (eds) Encyclopedia of operations research and management science. Springer, New York, pp 960–970. https://doi.org/10.1007/978-1-4419-1153-7

Swan J, Adriaensen S, Bishr M, Burke EK, Clark JA, De Causmaecker P, Durillo J, Hammond K, Hart E, Johnson CG, et al. (2015) A research agenda for metaheuristic standardization. In: Proceedings of the XI metaheuristics international conference

Voß S, Martello S, Osman I, Roucairol C (eds) (1999) Meta-Heuristics: advances and trends in local search paradigms for optimization. Kluwer, Boston. https://doi.org/10.1007/978-1-4615-5775-3

Watson JP, Howe AE, Whitley LD (2006) Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem. Computers Op Res 33(9):2623–2644. https://doi.org/10.1016/j.cor.2005.07.016

Weyland D (2010) A rigorous analysis of the harmony search algorithm - how the research community can be misled by a "novel" methodology. Int J Appl Metaheuristic Computing 1–2:50–60. https://doi.org/10.4018/jamc.2010040104

Weyland D (2015) A critical analysis of the harmony search algorithm–how not to solve sudoku. Op Res Perspecti 2:97–105

Whitley D (1994) A genetic algorithm tutorial. Stat Computing 4(2):65–85. https://doi.org/10.1007/BF00175354

Xing B, Gao WJ (2014) Innovative computational intelligence: a rough guide to 134 clever algorithms. Springer, Berlin. https://doi.org/10.1007/978-3-319-03404-1

Yang C, Tu X, Chen J (2007) Algorithm of marriage in honey bees optimization based on the wolf pack search. In: IEEE International conference on intelligent pervasive computing (IPC), pp 462 – 467, https://doi.org/10.1109/IPC.2007.104

Yang XS (2008) Nature-inspired metaheuristic algorithms. Luniver Press, UK

Yang XS (2010) A new Metaheuristic Bat-Inspired algorithm. In: González JR, Pelta DA, Cruz C, Terrazas G, Krasnogor N (eds) Nature inspired cooperative strategies for optimization (NICSO 2010). Studies in computational intelligence, vol 284. Springer, Berlin, Heidelberg, pp 65–74. 10.1007/978-3-642-12538-6_6

Zhang X, Chen W, Dai C (2008) Application of oriented search algorithm in reactive power optimization of power system. In: Third international conference on electric utility deregulation and restructuring and power technologies, DRPT 2008, Nanjing, China, pp 2856 – 2861, https://doi.org/10.1109/DRPT.2008.4523896