International Transactions in Operational Research / Early View

Article | Full Access

# Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: six misleading optimization techniques inspired by *bestial* metaphors

Christian L. Camacho-Villalón, Marco Dorigo, Thomas Stützle

First published: 26 July 2022
**https://doi.org/10.1111/itor.13176**
Citations: 1

**ViewIt@ULB**

## Abstract

We present a rigorous, component-based analysis of six widespread metaphor-based algorithms for tackling continuous optimization problems. In addition to deconstructing the six algorithms into their components and relating them with equivalent components proposed in well-established techniques, such as *particle swarm optimization* and *evolutionary algorithms*, we analyze the use of the metaphors that inspired these algorithms to understand whether their usage has brought any novel and useful concepts to the field of metaheuristics. Our result is that the ideas proposed in the six studied algorithms have been in the literature of metaheuristics for years and that the only novelty in these self-proclaimed *novel* algorithms is six different terminologies derived from the use of new metaphors. We discuss the reasons why the metaphors that inspired these algorithms are misleading and ultimately useless as a source of inspiration to design effective optimization tools. Finally, we discuss the rationale often presented by the authors of metaphor-based algorithms as their motivation to propose more algorithms of this type, which is based on a wrong understanding of the no-free-lunch theorems for optimization.

## 1 Introduction

Algorithms inspired by natural, artificial, and sometimes supernatural behaviors have become commonplace in the metaheuristics literature (Lones, 2014, 2020; Sörensen, 2015; Campelo and Aranha, 2021). From *intelligent water drops* to *musicians* and even the *COVID-19 pandemic*, it seems that virtually anything can be used as a source of inspiration to develop "novel" optimization techniques. Although taking inspiration from natural processes has played an important role in the history of metaheuristics (Sörensen and Glover, 2013; Sörensen et al., 2017; Gendreau and Potvin, 2019), in the last two decades far too many self-proclaimed "novel" metaphor-based algorithms have been published in the literature. Unfortunately, in the great majority of cases, it is not clear why the proposed metaphors are being used and what novel ideas they bring to the field of metaheuristics.

Among the most problematic aspects in papers proposing self-proclaimed "novel" metaphor-based techniques are the following. First, they use a metaphor to introduce new terminology to redefine concepts that are already known in the field of optimization. Second, they derive trivial mathematical models from the proposed metaphor that are only very loosely based on the metaphor itself, so that the models do not reflect correctly the metaphors; additionally, often the proposed algorithms do not match the mathematical models derived from the metaphor. Third, they motivate the use of a new metaphor on reasons such as "it has never been used before" or "the mathematical models are different from those used in the past"—rather than motivating it on a sound, scientific basis and explaining what is the optimization process represented in the metaphor and how it was used to make effective design choices in the proposed algorithm. Finally, they present biased evaluations and comparisons with other methods, such as an experimental evaluation based on a small number of low complexity problems and/or a comparison of the proposed algorithms with old techniques whose performance is far from the state of the art.

In this paper, we present a component-based analysis of the *grey wolf optimizer* (GWO) (Mirjalili et al., 2014), *moth-flame algorithm* (MFA) (Mirjalili, 2015b), *whale optimization algorithm* (WOA) (Mirjalili and Lewis, 2016), *firefly algorithm* (FA) (Yang, 2009), *bat algorithm* (BA) (Yang, 2010), and *antlion optimizer* (ALO) (Mirjalili, 2015a), which are among the most widespread "novel" metaphor-based algorithms published in the literature. These algorithms were chosen from the evolutionary computation-bestiary (Campelo and Aranha, 2021) using as sole criteria that they were proposed for the approximate solution of continuous optimization problems and that they were highly cited (data from Google Scholar retrieved on 6 June 2022, shows the following citation counts—GWO: 8468 citations; MFA: 2376 citations; WOA: 5407 citations; FA: 4317 citations; BA: 5201 citations; and ALO: 2096 citations.) In this paper, we analyze these algorithms in order to (i) clarify what concepts are proposed in them, and (ii) understand whether the concepts that come from the use of new metaphors are useful and novel in optimization. For each of the metaphors used in these algorithms, we evaluate whether the following criteria are fulfilled:

*Usefulness*—Does the metaphor bring useful ideas on how to solve optimization problems?

*Novelty*—Were the ideas brought by using the new metaphor novel in the field of metaheuristics when they were proposed?

The main finding of our analysis is that, despite presenting all these algorithms as *novel* approaches, none of them proposes a single new idea; rather, their authors used variations of the same algorithm components that have been for years in the literature of metaheuristics, framed them into new metaphors, and published them as new. In particular, we found that the *grey wolf* algorithm, MFA, WA, FA, and BA use the concepts proposed in the context of *particle swarm optimization* (PSO), while the *antlion* algorithm uses those of *evolution strategies* (ESs).

From the analysis of the metaphors that inspired the six algorithms, we found that they cannot be used to explain the majority of the design choices in them. The description of the metaphors in the articles are vague for the most part; important aspects are not mentioned at all or lack sufficient detail (such as what exactly is being optimized in the behavior that inspired the algorithm), while those that are irrelevant for the purpose of designing an optimization algorithm are abundant and overemphasized (e.g., how many species of fireflies/moths/whales exist and how amazing/fancy/unique they are in the opinion of the authors). Additionally, we observed that it is a common practice for the authors of this type of publications to relate metaphors and mathematical models using (i) ideas that do not belong to the metaphor originally described, and/or (ii) ideas that radically change the behavior that reputedly inspired the algorithm.

The rest of this paper is structured as follows. In Section 2, we present a number of variants of PSO (Section 2.1) and ESs (Section 2.2) that are relevant for our analysis. In Section 3, we deconstruct the six metaphor-based algorithms into their components and compare them, one by one, to those proposed in the context of PSO and ESs. Also, in this section, after analyzing each algorithm, we present the metaphors that inspired the algorithms and investigate whether they meet the criteria of usefulness and novelty. In Section 4, we discuss why (i) finding a novel metaphor that is both sound and useful to devise new optimization techniques is very difficult, and (ii) why motivating the introduction of “novel” algorithms on the no-free-lunch (NFL) theorems for optimization is irrelevant and shows a poor understanding of the theoretical research done in the field. Last, in Section 5, we summarize our findings and conclude the paper.

## 2 A few well-established metaheuristics techniques

### Particle swarm optimization

PSO (Kennedy and Eberhart, 1995) is a population-based algorithm proposed for the approximate solution of continuous optimization problems. In PSO, a swarm of particles, each representing a solution to the problem at hand, try to identify the most promising areas of the search space by applying a set of rules that take into account the locations of good solutions that they and their neighboring particles have visited in the past. To do so, each particle *i* knows, at every iteration *t*, its current position $\vec{x}_t^i$, velocity $\vec{v}_t^i$, and personal best position $\vec{p}_t^i$, as well as the best position $\vec{l}_t^i$ of the best particle in its neighborhood. The goal of using vectors $\vec{p}_t$ (called *cognitive* influence) and vectors $\vec{l}_t$ (called *social* influence) is to combine the knowledge acquired by each particle during the search with the knowledge of the best-informed individual in the neighborhood of the particle.

Depending on the way the neighborhood is defined in the algorithm, it is possible to create many different population topologies. For example, if the neighborhood of a particle consists of the two adjacent (i.e., closest) particles we have the so-called *ring* topology, while assigning all particles to the neighborhood of all other particles creates a *fully connected* or *g*best topology. In the latter, the local best particle is called the *global best* and its position is indicated by $\vec{g}_t$. When using the ring topology, the information about where the best-so-far solution is located spreads slowly among particles, while with the fully connected or *g*best topology the entire swarm knows immediately the position of the best-so-far solution at each iteration. Along with these two topologies, many others have been studied in the literature, including *wheels*, *lattices*, *stars*, and *randomly assigned edges* (Mendes et al., 2004).

Since its initial publication, PSO has been extensively studied and applied to many problems, resulting in a plethora of variants that range from little refinements of the original algorithm to new versions of the algorithm that contain quite elaborate changes and novel ideas. In the remaining of this section, we present a number of PSO variants that are relevant to our study.

*Standard PSO (Shi and Eberhart, 1998, 1999)—StdPSO*. In StdPSO, particles update their positions using the following rule:

$$\vec{x}_{t+1}^i = \vec{x}_t^i + \vec{v}_{t+1}^i, \tag{1}$$

$$\vec{v}_{t+1}^i = \omega \vec{v}_t^i + \varphi_1 \vec{a}_t^i \odot (\vec{p}_t^i - \vec{x}_t^i) + \varphi_2 \vec{b}_t^i \odot (\vec{l}_t^i - \vec{x}_t^i), \tag{2}$$

where ω is called inertia weight and controls the effect of the velocity vector at time *t*, $\phi_1$ and $\phi_2$ are called acceleration coefficients and weigh the relative importance given to the cognitive and social influence, $\vec{a}_t^i$ and $\vec{b}_t^i$ are two random vectors used to provide diversity to the particles' movement, and ⊙ indicates the Hadamard (entrywise) product between two vectors.

*Standard PSO 2011 (Clerc, 2011; Zambrano-Bigiarin et al., 2013)—SPSO-2011*. This variant is a modified version of StdPSO that prevents the issue of rotation variance. The velocity update rule of StdPSO was revised in SPSO-2011 as follows:

$$\vec{v}_{t+1}^{i} = \omega \vec{v}_t^i + \vec{x}_t'^i - \vec{x}_t^i, \tag{3}$$

where $\vec{x}_t'^i$ is a randomly generated point in the hypersphere $\mathscr{H}_i(\vec{c}_t^i, |\vec{c}_t^i - \vec{x}_t^i|)$ with center $\vec{c}_t^i$ and radius $|\vec{c}_t^i - \vec{x}_t^i|$, and $|\cdot|$ indicates the vector's L2 norm.

The computation of the center $\vec{c}_t^i$ is defined as follows:

$$\vec{c}_t^i = (\vec{L}_t^i + \vec{P}_t^i + \vec{x}_t^i) / 3, \tag{4}$$

where

$$\begin{aligned} \vec{P}_t^i &= \vec{x}_t^i + \varphi_1 \vec{a}_t^i \odot (\vec{p}_t^i - \vec{x}_t^i) \\ \vec{L}_t^i &= \vec{x}_t^i + \varphi_2 \vec{b}_t^i \odot (\vec{l}_t^i - \vec{x}_t^i) \end{aligned}. \tag{5}$$

*Fully informed PSO (Mendes et al., 2004)—FiPSO*. In FiPSO, the velocity update rule is as follows:

$$\vec{v}_{t+1}^i = \chi \left( \vec{v}_t^i + \sum_{k \in T_t^i} \varphi \vec{a}_{kt}^i \odot (\vec{p}_t^k - \vec{x}_t^i) \right), \tag{6}$$

where $\chi = 0.7298$ is a constant value called constriction coefficient (Clerc and Kennedy, 2002), $T_t^i$ is the set of particles in the neighborhood of *i*, and ϕ is a parameter. As opposed to the existing PSO variants that use the *best-of-neighborhood* model, in which the *social* influence of a particle comes from either $\vec{l}_t^i$ or $\vec{g}_t$, in FiPSO, a particle is influenced by all of its neighbors. The *social* influence model proposed in FiPSO is referred to as *fully informed* in the literature of PSO.

*Simple dynamic particle swarms (Peña, 2008a, 2008b)—SDPSs*. This class of PSO algorithms does not use the randomness induced by vectors $\vec{a}_t$ and $\vec{b}_t$ in the position update rule of the particles. Also, they are often implemented without a velocity vector $\vec{v}_t^i$. Most SDPSs can be instantiated from the following generalized position update rule:

$$\vec{x}_{t+1}^i = \vec{x}_t^i + \varepsilon (\vec{y} - \vec{x}_t^i), \tag{7}$$

where ε is a parameter and $\vec{y}$ is a vector obtained by combining the information of two or more particles in the swarm. Examples of how vector $\vec{y}$ can be computed in SDPSs include:

$$\text{Standard:}\vec{y} = \frac{u_1\,\vec{x}_t^{\prime 1} + u_2\,\vec{x}_t^{\prime 2}}{u_1 + u_2}, \tag{8}$$

$$\text{Normal:}\vec{y} = \mathscr{N}\left(\frac{\vec{x}_t^{\prime 1} + \vec{x}_t^{\prime 2}}{2}, |\vec{x}_t^{\prime 1} - \vec{x}_t^{\prime 2}|\right), \tag{9}$$

where $\vec{x}_t^{\prime 1}$ and $\vec{x}_t^{\prime 2}$ are position vectors chosen according to some criterion, and $u_1$ and $u_2$ are two real parameters whose value is typically set in the range (0,1]. SDPSs algorithms vary in all types of aspects, including the number of particles participating in the computation of $\vec{y}$, which can be from 1 to all particles and the way in which the current position of a particle is taken into account in the computation of $\vec{x}_{t+1}^{i}$.

*Extrapolation PSO (Arumugam et al., 2007, 2009)—ePSO*. In *e*PSO, particles do not have a *cognitive* influence component (i.e., vector $\vec{p}_t$) and parameters $\varphi_{1t}$ and $\varphi_{2t}$ are replaced by two so-called "extrapolation" coefficients. The position update rule proposed for *e*PSO is as follows:

$$\vec{x}_{t+1}^{i} = \vec{g}_t + \varphi_{1t}\,\vec{g}_t + \varphi_{2t}\,(\vec{g}_t - \vec{x}_t^{i}), \tag{10}$$

where $\varphi_{1t} = \mathscr{U}\,[0,1]\,k_1$, $\varphi_{2t} = k_1\,e^{k_2 \Lambda_t^i}$, $k_1 = k_2 = e^{-t/t_{max}}$, $\Lambda_t^i = |(f\,(\vec{g}_t) - f\,(\vec{x}_t^i))\,/\,f\,(\vec{g}_t)|$, and $f\,(\,\cdot\,)$ refers to the objective function of a minimization problem. This position update rule combines the information of the global best solution ($\vec{g}_t$) with the current position of the particles ($\vec{x}_t^i$) and adjusts the displacement of the particle in terms of the difference between $f\,(\vec{g}_t)$ and $f\,(\vec{x}_t^i)$. Because of the way $\varphi_{1t}$ and $\varphi_{2t}$ are computed, a particle will experience a strong attraction toward $\vec{g}_t$ when its quality is much lower than that of $f\,(\vec{g}_t)$, and a weak attraction toward $\vec{g}_t$ when its quality is similar to $f\,(\vec{g}_t)$.

## Evolution strategies

ESs (Rechenberg, 1971, 1973; Schwefel, 1977, 1981) are among the first evolutionary algorithms (Fogel et al., 1966; Holland, 1975) (EAs) mainly proposed to solve continuous optimization problems. ESs use real-valued vectors to represent solutions and, similar to other EAs, they iteratively apply a number of *evolutionary operators* (or just *operators*) to stochastically sample new solutions. The operators typically used in ESs are as follows:

*parental selection*—choice of the solutions (*parents*) at the beginning of each iteration that will be used to create new solutions (*offspring*);

*recombination*—mechanism used to combine the information of two or more *parents* in order to create one or more *offspring*;

*mutation*—the process of applying a small perturbation to *offspring*;

*survival selection*—choice of the solutions that will pass to the next iteration (also known as *generation*).

The *parental selection* operator can be implemented using a *deterministic* scheme (one or more specific individuals are selected) or a *stochastic* scheme (individuals are selected according to a probability distribution based on their quality value). The quality value is usually called *fitness* value of an individual. One of the first stochastic parental selection schemes proposed in the literature is *fitness proportional* (Holland, 1975; Bäck et al., 1997) and consists in assigning to each individual a probability of being selected that is proportional to its solution quality. Among the deterministic schemes, a typical option is the so-called *fitness-based* selection, in which only individuals with similar solution quality are matched together to produce offspring (Hansen et al., 2015). Since *fitness-based* selection drives the evolution process toward the best individuals, it is often used when survival selection is *nonelitist* (see below).

The *recombination* operator can be implemented in many different ways. Among the most used recombination operators are *discrete recombination*, where the *k*th component of the offspring is taken from either of the parents, *intermediate recombination*, where the offspring is the result of computing the arithmetic average of the parents' *k*th component, and *weighted recombination*, which is similar to *intermediate recombination* but parents can have different weights. Although recombination was widely used in early ES variants, it is considered optional in most recent ESs implementations.

In ESs, the *mutation* operator is the most important algorithm component and it is commonly implemented by adding a point symmetric perturbation (e.g., random numbers drawn from a multivariate Gaussian/Cauchy/Lévy distribution) to the result of recombination or, if recombination is not used, to the result of parental selection. In practice, most ESs employ the Gaussian distribution to create a perturbed vector $\vec{u}$msup, such as the well-known *spherical/isotropic* mutation, which is defined as follows:

$$\vec{u}\ \text{msup} = \vec{u} + \mathscr{N}(0, \mathbf{C}), \tag{11}$$

where $\vec{u}$ is a vector representing an individual and $\mathscr{N}(0, \mathbf{C})$ is the Gaussian distribution with zero mean and covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$. In the *spherical/isotropic* mutation, $\mathbf{C}$ is proportional to the identity matrix $\mathbf{I}$, and therefore, the Gaussian mutation component is often indicated as $\mathscr{N}(0, \mathbf{I})$.

ESs use the mnemonic notation $(\mu \overset{+}{,} \lambda)$ to indicate the way in which *survival selection* will be implemented in the algorithm, where μ and λ are positive integers that represent, respectively, the number of parents at the beginning of the iteration and the number of offspring generated at each iteration. The symbols “+” and “, ” are used to specify whether survival selection is *elitist*

$(\mu + \lambda)$ or *nonelitist* $(\mu, \lambda)$. In the $(\mu + \lambda) -$ ES, the next generation is generated by selecting the best μ solutions from the set of $\mu + \lambda$ individuals, whereas in the $(\mu, \lambda) -$ ES the next generation is generated by selecting the best μ solutions from the set of λ offspring.

## 3 Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms

In this section, we analyze the *grey wolf* algorithm, MFA, WA, FA, BA, and *antlion* algorithm. For each of them, we present (i) the algorithm using the standard optimization terminology, (ii) a component-based comparison with existing techniques, and (iii) the metaphor that inspired the algorithm and discussion on whether it meets the criteria of *novelty* and *usefulness*. For (i) and (ii), we avoid using the vocabulary introduced by the authors of these algorithms because, as we will show after presenting each of them, it is unnecessary and misleading in many ways. In our view, one of the main reasons why these algorithms have not been immediately recognized as minor variants of well-established techniques is that they were presented using metaphor-based terminologies that obfuscated their similarities with existing approaches. Therefore, by explaining each algorithm in plain computational terms, we intend to make clearly visible what ideas are being proposed in them and whether they are truly novel or not. Also, we believe that presenting first (i) and (ii) and leaving (iii) at the end allows the reader to better appreciate whether the metaphor contributes at all to the design of the proposed algorithm. Finally, we would like to mention to the reader that all six algorithms discussed here have publicly available implementations. *Grey wolf algorithm*, MFA, WA, and *antlion* algorithm can be downloaded from https://seyedalimirjalili.com/ and FA and BA from https://nl.mathworks.com/matlabcentral/profile/authors/2652824.

### Grey wolf optimizer

The GWO (Mirjalili et al., 2014) is an algorithm in which the three iteration-best solutions in the population are used to bias the movement of the remaining solutions. This idea is implemented in GWO by defining three vectors $\vec{s}_t^k$ (for $k = 1, 2, 3$) as follows:

$$\begin{aligned}\vec{s}_t^1 &= \vec{x}_t^{\text{best1}} - \varphi_t\left(2\vec{r}_t^1 - \vec{1}\right) \odot \left(2\vec{q}_t^1 \odot \vec{x}_t^{\text{best1}} - \vec{x}_t^i\right)^{\text{abs}}\\ \vec{s}_t^2 &= \vec{x}_t^{\text{best2}} - \varphi_t\left(2\vec{r}_t^2 - \vec{1}\right) \odot \left(2\vec{q}_t^2 \odot \vec{x}_t^{\text{best2}} - \vec{x}_t^i\right)^{\text{abs}}, \forall i,\\ \vec{s}_t^3 &= \vec{x}_t^{\text{best3}} - \varphi_t\left(2\vec{r}_t^3 - \vec{1}\right) \odot \left(2\vec{q}_t^3 \odot \vec{x}_t^{\text{best3}} - \vec{x}_t^i\right)^{\text{abs}}\end{aligned} \quad (12)$$

where $\vec{x}_t^{\text{best1}}$, $\vec{x}_t^{\text{best2}}$, and $\vec{x}_t^{\text{best3}}$ are the three best solutions at iteration *t*, $\vec{r}_t^k$, and $\vec{q}_t^k$ are two random vectors with values drawn from $\mathscr{U}[0, 1]$ that induce perturbation to the components of $\vec{s}_t^k$, $\varphi_t$ is a parameter that decreases linearly from 2 to 0, $\vec{1}$ is a vector of all ones, and $(\cdot)^{\text{abs}}$ indicates the entrywise absolute value of a vector. The entrywise absolute value of a vector is a transformation that can be formally defined as $(\vec{u})^{\text{abs}} = (|u_i|, \cdots |u_d|)^{\mathrm{T}}$. The position update rule combining the information of vectors $\vec{s}_t^k$ is defined as follows:

$$\vec{x}_{t+1}^i = (\vec{s}_t^1 + \vec{s}_t^2 + \vec{s}_t^3)/3. \tag{13}$$

### 3.1.1 The grey wolf optimizer is PSO

The mathematical model of GWO is a variant of the one proposed for SPSO-2011. To better explain how GWO compares to SPSO-2011, we consider first the mathematical model of GWO without the perturbation component $\varphi_t(2\vec{r}_t^k - \vec{1})$, which is as follows:

$$\begin{aligned} \vec{s}_t^1 &= \vec{x}_t^{\text{best1}} - \left(2\vec{q}_t^1 \odot \vec{x}_t^{\text{best1}} - \vec{x}_t^i\right)^{\text{abs}} \\ \vec{s}_t^2 &= \vec{x}_t^{\text{best2}} - \left(2\vec{q}_t^2 \odot \vec{x}_t^{\text{best2}} - \vec{x}_t^i\right)^{\text{abs}}, \ \forall i. \\ \vec{s}_t^3 &= \vec{x}_t^{\text{best3}} - \left(2\vec{q}_t^3 \odot \vec{x}_t^{\text{best3}} - \vec{x}_t^i\right)^{\text{abs}} \end{aligned} \tag{14}$$

Both SPSO-2011 (Equation (5)) and GWO (Equation (14)) are based on the idea of (i) defining, for each particle *i* in the population, a hypertriangle in the search space, whose vertices are a function of positions known as *i*, and (ii) using the centroid of the hypertriangle in the computation of *i*'s new position. The main difference between the two algorithms is in the way in which the vertices of the hypertriangle are computed. While in GWO all particles compute the vertices using the same three iteration-best solutions, that is, $\vec{x}_t^{\text{best1}}$, $\vec{x}_t^{\text{best2}}$, and $\vec{x}_t^{\text{best3}}$, in SPSO-2011, each particle uses its local information, that is, vectors $\vec{x}_t^i$, $\vec{p}_t^i$, and $\vec{l}_t^i$.

In PSO, the goal of using vectors $\vec{v}_t^i$, $\vec{l}_t^i$, and $\vec{p}_t^i$, where $\vec{l}_t^i$ is different for each neighborhood and $\vec{p}_t^i$ is different for each particle, is to include components that allow to balance the relation between *exploration* and *exploitation* of the search space. Adding particles' previous velocity $\vec{v}_t^i$ to their new positions promotes exploration, whereas attracting particles toward known good solutions, such as $\vec{l}_t^i$ and $\vec{p}_t^i$, promotes exploitation. In contrast, in GWO, as seen in Equation (14), the entire swarm is attracted toward the same three solutions $\vec{x}_t^{\text{best1}}$, $\vec{x}_t^{\text{best2}}$, and $\vec{x}_t^{\text{best3}}$, which can be useful for intensifying the search in the area defined by these vectors, but prevents particles from exploring other regions.

Not surprisingly, the authors of GWO found that their first version of the algorithm (which is, as far as it is understood in their article, the one using Equation (14)) resulted in a poor performing implementation that "is prone to stagnation in local solutions" (Mirjalili et al., 2014, p. 50). To remediate this issue, the authors added a second perturbation component to the computation of vectors $\vec{s}_t^k$ (Equation (12)), which includes a random vector $\vec{r}_t^k$ multiplied by a linearly decreasing parameter $\varphi_t$. This additional perturbation component produces both positive and negative random values in the range [ – 2, 2], determining a much stronger perturbation to the particles movement than the one initially defined with vector $\vec{q}_t^k$. To avoid particles divergence outside the search space, the impact of $\varphi_t\,(2\vec{r}_t^k - \vec{1})$ in the computation of vectors $\vec{s}_t^k$ is controlled by parameter $\varphi_t$ whose value decreases linearly from 2 to 0, allowing particles to move closer and closer to $\vec{x}_t^{\text{best1}}$, $\vec{x}_t^{\text{best2}}$, and $\vec{x}_t^{\text{best3}}$ toward the end of the algorithm's execution.

In addition to the computation of vectors $\vec{s}_t^k$, the position update rule of GWO introduced in Equation (13) is same as the computation of the center $\vec{c}_t^i$ in SPSO-2011—see Equation (4). However, in SPSO-2011, vector $\vec{c}_t^i$ is the center of a hyperspherical distribution from which a random vector is generated, whereas in GWO the computed center becomes the new position of the particle. Because of this difference, GWO's position update rule can also be compared to the standard recombination rule proposed for SDPSs (Equation (8)) extended to three particles, where vectors $\vec{x}_t'^1$, $\vec{x}_t'^2$, and $\vec{x}_t'^3$ correspond to vectors $\vec{s}_t^k$, and parameters $u_1$, $u_2$, and $u_3$ are set to 1.

### 3.1.2 The metaphor of grey wolves hunting

The authors of GWO say in their original paper (Mirjalili et al., 2014) that they were inspired by the way in which "grey wolves organize for hunting following a strict social hierarchy," where the pack is divided, from top to bottom, into α, β, δ, and ω wolves. According to their description of grey wolves hunting behavior, while α wolves usually take part in hunting and they are in charge of guiding the rest of wolves participating in this activity, the β and δ wolves only take part in hunting occasionally. In GWO, vector $\vec{x}_t^{\text{best1}}$ represents the α wolf, $\vec{x}_t^{\text{best2}}$ represents the β wolf, $\vec{x}_t^{\text{best3}}$ represents the δ wolf, and the rest of solutions in the swarm represent the ω wolves. However, since it is false that the entire pack always participates in hunting every time, saying that solutions $\vec{x}_t^{\text{best1}}$, $\vec{x}_t^{\text{best2}}$, and $\vec{x}_t^{\text{best3}}$ represent the α, β and δ is inaccurate, as it does not follow the description of the "strict social hierarchy" of grey wolves that inspired the authors when proposing the algorithm.

The authors of GWO mention that there are three phases during hunting, each one composed of a number of steps: (i) *tracking*, *chasing*, and *approaching* the prey; (ii) *pursuing*, *encircling*, and *harassing* the prey until it stops moving; and (iii) *attacking* toward the prey. However, GWO considers only two of the seven steps mentioned: *encircling*, which is modeled using Equation (13), and *attacking*, which is modeled by linearly decreasing the value of $\varphi_t$ from 2 to 0 in Equation (12). In the imagery of the metaphor, when $\varphi_t$ is less than 1, wolves concentrate around the prey (therefore attacking it); and when it is greater than 1, they *search* for other prey. Note that, despite *search* being not an activity in the hunting phases of wolves, the authors added this step to the metaphor and explained it as "the divergence among wolves during hunting in order to find a *fitter prey*" (Mirjalili et al., 2014, p. 50).

Based on the description of the "grey wolves hunting" behavior presented by the authors of

GWO and the way this behavior is used as a metaphor to develop the proposed algorithm, it is clear that the only contribution of the metaphor is to create confusion and to hide the similarities of the “novel” GWO with PSO. In particular, GWO does not satisfy the criterion of usefulness because there are no components in the behavior of grey wolves that can be used as effective design choices in an optimization algorithm, as evidenced by the fact that the mathematical model originally derived from this behavior resulted in a poor performing technique that stagnated prematurely. Also, as it was shown in the previous section, GWO does not satisfy the novelty criterion because all the algorithm components of GWO correspond to particular cases of algorithm components previously proposed for SPSO-2011 and SDPSs.

## Moth-flame algorithm

In the MFA (Mirjalili, 2015b), each solution in the population is assigned a ranking ($rank_t^i$) based on the quality of its current position ($\vec{x}_t^i$), which determines the specific neighbor of the population that will take part in the computation of its position update rule; that is, the solution with $rank_t^i = 1$ will compute its new position using the best overall solution $\vec{g}_t^1$, the one with $rank_t^i = 2$ will use the second best solutions $\vec{g}_t^2$, and so on. Note that the set of vectors $\vec{g}_t$ is same as the set of vectors $\vec{p}_t$ in PSO, but ordered according to their quality, so that $\vec{g}_t^1$ corresponds to the vector $\vec{p}_t$ with the highest quality and $\vec{g}_t^n$ to the one with the lowest quality. The equation modeling this process is

$$\vec{x}_{t+1}^i = \vec{g}_t^{rank_t^i} + \varphi_t \left(\vec{g}_t^{rank_t^i} - \vec{x}_t^i\right)^{abs}, \tag{15}$$

with

$$\varphi_t = e^{\delta} \cos\left(2\pi\delta\right), \tag{16}$$

where $(\,\cdot\,)^{abs}$ denotes the entrywise absolute value, $\delta = \left(\frac{-t}{t_{max}} - 2\right) \mathscr{U}\left[0, 1\right] + 1$, and $t_{max}$ is the iteration number at which the algorithm stops. There are two things to note about Equation (16). First, the range in which the value of δ is computed spans from $[-1 \frac{1}{t_{max}}, 1]$ to ( – 2, 1] as the value of *t* grows; second, when $\delta \to -\infty$, the value of $\varphi_t \to 0$. Therefore, the probability of computing large values for $\varphi_t$ decreases toward the end of the execution of the algorithm, allowing solutions to move closer and closer to their respective vector $\vec{g}_t^{rank_t^i}$.

In MFA, every $t_{max}/n$ iterations, the variable $rank_t^i$ of the worst $n - m$ solutions is set to $rank_t^i = m$, where *m* is computed as $m = \textbf{round}\left(n - \frac{t(n-1)}{t_{max}}\right)$, *n* is the population size, and **round** $(\,\cdot\,)$ indicates the round to the nearest integer function. The goal of doing this is to stop using the $\vec{p}_t$ vector of the $n - m$ solutions to influence the position update of other solutions. Because of the way the value of *m* is computed, the number of solutions influencing the position update rule of other solutions will decrease over the course of iterations, until only the global best solution ($\vec{g}_t^1$) is used to influence the movement of the entire population. It is worth mentioning that, in Mirjalili (2015b), the equation used to compute the value of *m* is given by **round** $\left((n-t)\,\frac{(n-1)}{t_{max}}\right)$; however, this equation is wrong as it produces negative values when $t > n$.

## 3.2.1 The moth-flame algorithm is PSO

The MFA is a variant of *e*PSO, where the only difference is that MFA uses a model of influence (MoI) that assigns each particle with the personal best position of one specific neighbor depending on its ranking. Therefore, with the exception of the MoI, the comparison between MFA and *e*PSO can be done directly, since it is possible to obtain the mathematical model of MFA (Equation (15)) just by setting $\varphi_{1t} = 0$ and $k_1 = 1$ in the position update rule of *e*PSO (Equation (10)).

In PSO, the MoI (also known as *selection of social influence* (Mendes, 2004) or *graph of influence* (Clerc, 2010)) refers to the way in which particles select other members of the swarm to influence their movement and it is equivalent to the concept of parental selection in EAs. In Section 2.1, we describe the two most popular MoIs, which are the *best-of-neighborhood* and the *fully informed* models. However, there are many other models proposed in the literature of PSO, including *random*, in which particles are influenced by a random neighbor (Kennedy, 1999), *ranked-fully informed* (Jordan et al., 2008), in which the contribution of each neighbor is weighted according to its rank—see Mendes (2004) and Montes de Oca (2011) for a detailed review.

Although to the best of our knowledge, there is no PSO variant using exactly the same MoI implemented in the MFA, the idea of using a rank-based selection is not new at all in the metaheuristics literature. For example, the *ranking* selection mechanism used in EAs assigns to each individual a ranking based on its fitness value that determines the probability of selecting it for the next generation (Bäck et al., 1997; Grefenstette, 2000). In the deterministic version of the ranking selection, the best individuals are selected from the population to pass to the next generation with the goal of both always preserving the best solutions found at any iteration and of biasing the creation of new solutions through recombination and mutation—see Section 2.2. In MFA, the best solution found by each particle is kept in its personal best vector ($\vec{p}_t^i$) and the only goal of using rankings is to create a mapping between a particle and one of its neighbors to bias its movement.

A PSO variant that is similar to MFA in this regard is the rank-based PSO with dynamic adaptation ($\mathrm{PSO}_{\mathrm{rank}}$) (Akbari and Ziarati, 2011), where each particle receives influence from multiple neighbors and the contribution of each neighbor is weighted according to three criteria: ranking, Euclidean distance, and the total number of neighbors. Similar to MFA, in

$\mathrm{PSO}_{\mathrm{rank}}$, the number of neighbors influencing the particles at each iteration is controlled using a parameter that decreases linearly according to the number of iterations, so that all particles are eventually influenced only by the global best solution.

### 3.2.2 The metaphor of moths navigation

As we showed in the previous section, the MFA is same as the *e*PSO algorithm except for the deterministic rank-based MoI component, which is an idea originally proposed in the context of EAs and that is implemented in MFA in a similar way to $\mathrm{PSO}_{\mathrm{rank}}$. Therefore, the MFA does not meet the criterion of novelty. In this section, we analyze the behavior of moths that inspired the algorithm to check whether it has any component that can be useful from the point of view of designing an optimization algorithm.

The author of MFA says that the inspiration for this algorithm is the "navigation method of moths in nature" that allows them to move in a straight line by maintaining a fixed angle with respect to the moon—a mechanism known as *transverse orientation* according to Mirjalili (2015b). For developing MFA, the author considered the case when moths are attracted to artificial lights, not to the moon. In this case, moths engage in what the author referred to as "useless or dead spiral fly path," which happens because the transverse orientation method is only useful to fly in a straight line when the light source is very far. The flight of moths around artificial lights is modeled using Equations (15) and (16), where the set of current solutions ($\vec{x}_t$) represent "moths," their personal best positions ($\vec{p}_t$), called "flames," represent artificial light sources, and the "useless or dead spiral fly path" behavior is represented by computing the value of $\varphi_t$ using a logarithmic spiral function.

Although the behavior that inspired MFA is moths' inability to escape from artificial light sources due to transverse orientation, in the algorithm, the author prevents this behavior by assigning "moths" (current solutions) to specific "flames" (personal best solutions) and by gradually stopping the use of the worst "flames" as the number of iteration grows. This is because, if "moths" and "flames" are defined as fixed couplings in the algorithm (as they are in the metaphor), a solution located in a poor quality region of the search space will most likely be incapable of moving away from that region because the only influence the solution has is its personal best solution. The author intended to avoid this problem by changing the specific personal best solution to which a current solution is assigned. However, this modification puts into question the motivation to use the metaphor of moths in the first place, since the algorithm following the "useless or dead spiral fly path" behavior of moths is, in the words of the author, prone "to be trapped in local optima quickly" (Mirjalili, 2015b, p. 232).

After analyzing the "useless or deadly spiral fly path" behavior of moths that inspired MFA, the mathematical model derived from it, and the resulting algorithm that is "prone to stagnation quickly," it is obvious that the metaphor of "moths navigation" does not meet the criterion of usefulness. Also, considering the modification of assigning “moths” to specific “flames” that the author introduced to make the algorithm actually be able to perform optimization, radically changing the behavior of moths that inspired him to proposed MFA, the use of this metaphor of “moths navigation” in the context of optimization is rather counterproductive.

## Whale optimization algorithm

WOA (Mirjalili and Lewis, 2016) is a combination of the mathematical models of GWO and MFA algorithms (all of them proposed by the same authors), in which solutions are updated using one of three possible position update rules (the three cases of Equation (17)) that is chosen on the basis of stochastic criteria. The mathematical model of WOA is as follows:

$$\vec{x}_{t+1}^{i} = \begin{cases} \vec{x}_t^k - \varphi_{1t}\left(2\vec{r}_t - \vec{1}\right) \odot \left(2\vec{q}_t \odot \vec{x}_t^k - \vec{x}_t^i\right)^{\text{abs}} & \text{if} \texttt{Random \_ Neighbor} \\ \vec{x}_t^{\text{best}} + \varphi_{2t}\left(\vec{x}_t^{\text{best}} - \vec{x}_t^i\right)^{\text{abs}} & \text{if } \neg \texttt{Random \_ Neighbor} \wedge \texttt{Exp \_ Coefficient} \\ \vec{x}_t^{\text{best}} - \varphi_{1t}\left(2\vec{r}_t - \vec{1}\right) \odot \left(2\vec{q}_t \odot \vec{x}_t^{\text{best}} - \vec{x}_t^i\right)^{\text{abs}} & \text{if } \neg \texttt{Random \_ Neighbor} \wedge \neg \texttt{Exp \_ Coefficient} \end{cases}, \quad (17)$$

where $(\cdot)^{abs}$ denotes the entrywise absolute value, $\vec{x}_t^k$ indicates the current position of a randomly chosen neighbor *k* at iteration *t*, and vectors $\vec{x}_t^{\text{best}}$, $\vec{r}_t$, $\vec{q}_t$, $\vec{x}_t^i$, and parameter $\varphi_{1t}$ are the same ones defined for computing vector $\vec{s}_t^1$ in GWO (see Equation (12)). In the remainder, we will refer to the three cases of Equation (17) as *first rule*, *second rule*, and *third rule*, respectively, to simplify our analysis.

The computation of $\varphi_{2t}$ in Equation (17) is given by

$$\varphi_{2t} = e^{\delta}\cos\left(2\pi\delta\right), \quad (18)$$

with $\delta = \left(\frac{-t}{t_{max}} - 2\right)\mathscr{U}\left[0,1\right] + 1$, which is exactly the same as Equation (16) proposed for MFA.

The specific position update rule that a particle will use depends on the value of the logical variables $\texttt{Random \_ Neighbor}$ and $\texttt{Exp \_ Coefficient}$, which are defined as follows:

$$\begin{aligned} \texttt{Random \_ Neighbor} &:= \begin{cases} \texttt{TRUE} & \text{if} \varphi_{1t}\left(2r_{1,t} - 1\right) > 1 \\ \texttt{FALSE} & \text{otherwise} \end{cases}, \\ \texttt{Exp \_ Coefficient} &:= \begin{cases} \texttt{TRUE} & \text{if } \mathscr{U}\left[0,1\right] < 0.5 \\ \texttt{FALSE} & \text{otherwise} \end{cases}, \end{aligned} \quad (19)$$

where $r_{1,t}$ indicates the first element of vector $\vec{r}_t$. Because of the way the logical variables $\texttt{Random \_ Neighbor}$ and $\texttt{Exp \_ Coefficient}$ are used in Equation (17), the algorithm will select with higher probability the *first rule* during the first half of its execution, and either the *second rule* or the *third rule* (but not the *first* one) with the same probability during the second half. The reason why the *first rule* is not selected by the algorithm during the second half of its execution is that the value of $\varphi_{1t}$ is less than 1 and the probability of $\mathtt{Random\_Neighbor}$:= TRUE when $\varphi_{1t} < 1$ is 0.

### 3.3.1 The whale optimization algorithm is PSO

WOA is a PSO algorithm that combines the mathematical models of SPSO-2011 and *e*PSO. To explain how WOA compares with SPSO-2011, let us consider initially WOA's *first rule* and *third rule*. These two rules differ only in the vector that is used to bias the particles' movement: $\vec{x}_t^k$ (*first rule*) and $\vec{x}_t^{\text{best}}$ (*third rule*). In Section 3.1.1, where we compared GWO with PSO, we discussed the fact that the computation of vector $\vec{s}_t^1$ in GWO, which is same as WOA's *third rule*, is defined in the same way as vectors $\vec{L}_t^i$ and $\vec{P}_t^i$ in SPSO-2011 (Equation (5)), the only difference is that there is an additional perturbation component—$\varphi_{1t}(2\vec{r}_t - \vec{1})$—that was introduced to avoid the premature stagnation issue that affects GWO. Similarly, WOA's *first rule* is a variant of the mathematical model shown in Equation (5), but in this case there is also the difference that the local best particle ($\vec{l}_t^i$) is replaced by a randomly chosen neighbor ($\vec{x}_t^k$).

Unlike most PSO variants, WOA does not make use of a velocity vector ($\vec{v}_t^i$), which is an algorithm component that, among others, allows particles to diverge from moving exactly toward $\vec{l}_t^i$ or $\vec{p}_t^i$ and to explore other areas of the search space. To compensate for the lack of a component such as $\vec{v}_t^i$, WOA lets particles to be occasionally biased by $\vec{x}_t^k$ (*first rule*) instead of by $\vec{x}_t^{\text{best}}$ (*second* and *third rules*). Therefore, although WOA's *first rule* is defined in the same way as vectors $\vec{L}_t^i$ and $\vec{P}_t^i$ in SPSO-2011, the purpose of having this rule in the algorithm is rather similar to the one of using vector $\vec{v}_t^i$ in PSO algorithms—as it allows to introduce diversity in the solutions. To control the impact of the *first rule* in the optimization process and let particles converge toward $\vec{x}_t^{\text{best}}$, the authors of WOA defined the value of the logical variable $\mathtt{Random\_Neighbor}$ as a function of the linearly decreasing parameter $\varphi_{1t}$, which results in $\mathtt{Random\_Neighbor}$:= FALSE in the second half of the algorithm's execution when $\varphi_{1t} < 1$.

Let us now consider the *second rule*—that is, $\vec{x}_t^{\text{best}} + \varphi_{2t}(\vec{x}_t^{\text{best}} - \vec{x}_t^i)^{\text{abs}}$—that allows particles to explore the area of the search space around the iteration-best solution and that uses the exponential function to compute the value of $\varphi_{2t}$. The ideas involved in the *second rule* of WOA are the same ideas introduced in *e*PSO (Equation (10)) described in Section 2.1, where particles do not use vectors $\vec{p}_t$ and the value of the acceleration coefficients is computed using the exponential function. In fact, it is easy to see that the *second rule* of WOA can be obtained from the position update rule of *e*PSO (Equation (10)) just by setting $\varphi_{1t} = 0$ and $k_1 = 1$, where the only difference is that, in WOA, the displacement of a particle toward $\vec{x}_t^{\text{best}}$ is adjusted using a random value (see Equation (16)), whereas in *e*PSO it is adjusted based on the difference between $f(\vec{x}_t^i)$ and $f(\vec{g}_t)$.

### 3.3.2 The metaphor of humpback whales' bubble net

The authors of WOA say that the inspiration for this algorithm is the “bubble-net strategy” that humpback whales use for hunting (Mirjalili and Lewis, 2016). According to the description they provided in their paper, this strategy involves performing two different *maneuvers*. The first maneuver is called “upward spirals” and consists in creating an upward spiral path of bubbles in the water, while the second maneuver, called “double loops,” is composed of three different stages: *coral loop*, *lobtail*, and *capture loop*. For developing WOA, the authors considered only the “upward-spiral” maneuver. The “double-loop” maneuver and its three stages are not described in the WOA paper.

The authors of WOA modeled the “spirals path of bubbles” created by whales during the “upward-spiral” maneuver by computing parameter $\varphi_{2t}$ in WOA's *second rule* (Equation (17)) using the exponential function. Since $\varphi_{2t}$ is the only component in the algorithm that can be justified in terms of the metaphor of whales, the authors added two more maneuvers to the metaphor originally presented, one called “shrinking encircling” and the other called “search for prey.” The “shrinking encircling” maneuver, whose mathematical model and description are exactly similar to the “encircling” step in the GWO (see Section 3.1) published by the same authors two years before WOA, is used to justify the linearly decreasing value of $\varphi_{1t}$ in WOA's *first* and *third rules* and it is based on the idea that “whales can recognize the location of prey and encircle them”; whereas the “search for prey” maneuver is used to justify using the position vector of a randomly chosen neighbor *k* in WOA's *first rule* and it is based on the idea that “humpback whales search randomly according to the position of each other” (Mirjalili and Lewis, 2016, pp. 53–54).

There are several reasons why the metaphor of “humpback whales hunting” that inspired WOA does not meet the criteria of usefulness and novelty. First, since the authors do not provide a complete description of the humpback whales' “bubble-net strategy” that inspired them, it is impossible to know what exactly is the optimization behavior they observed in it, if any. Second, the only component taken from the behavior of humpback whales' “bubble-net strategy” presented in the WOA paper is the idea of computing the value of parameter $\varphi_{2t}$ using the exponential function; however, as we discussed in the previous section, this idea was already proposed before in the PSO literature in a variant called *e*PSO. Third, the mathematical model of WOA is nothing but a combination of the ones used in the GWO (Section 3.1) and MFA (Section 3.2) algorithms, which were proposed by the same authors in 2014 and 2015, respectively, and that are variants of PSO.

Indeed, concerning the third point, in the WOA paper the authors mention as follows: “The main difference between the current work [*whale optimization algorithm*] and the recently published papers by the authors (particularly GWO (Mirjalili et al., 2014)) is the simulated hunting behavior with random or the best search agent to chase the prey and the use of a spiral to simulate bubble-net attacking mechanism of humpback whales” (Mirjalili and Lewis, 2016, pp. 52). Although the authors acknowledge that there are similarities between WOA and GWO, they make only a vague mention of the connection between the two algorithms and fail to mention that the equation that models the spiral in the whales' bubble-net mechanism is exactly the same equation they used to model the "useless or deadly spiral fly path of moths" in the MFA (Mirjalili, 2015b).

## Firefly algorithm

In the FA (Yang, 2009), at each iteration *t*, each solution *i* in the population updates its position in the search space by moving toward every other solution that has higher quality than its own. The process of updating solutions in this algorithm is carried out in two steps. In the first step, the population is sorted bottom-up according to the solutions' quality, so that the first solution to be updated is the one with the worst quality and the last one to be updated is the one with the best quality. In the second step, following the bottom-up order established before, each solution *i* determines the set $W_t^i$ of solutions with better quality than its own; sets its initial position $\vec{m}_{t,s_0}^i = \vec{x}_t^i$; and applies the following two equations:

$$\vec{x}_{t+1}^i = \vec{m}_{t,s_{|W_t^i|}}^i, \tag{20}$$

$$\vec{m}_{t,s}^i = \vec{m}_{t,s-1}^i + \varphi_t^{\vec{w}_{t,s}^i,\vec{m}_{t,s-1}^i}\left(\vec{w}_{t,s}^i - \vec{m}_{t,s-1}^i\right) + \xi \vec{r}_{t,s}^i, \tag{21}$$

where $\vec{w}_{t,s}^i$ is an element of the ordered set $W_t^i$, $\varphi_t^{\vec{w}_{t,s}^i,\vec{m}_{t,s-1}^i}$ is an acceleration coefficient whose value is computed as a function of the Euclidean distance between the two intermediate points $\vec{w}_{t,s}^i$ and $\vec{m}_{t,s-1}^i$, $\vec{r}_{t,s}^i$ is a vector whose components are random numbers drawn from the uniform distribution $\mathscr{U}[0,1]$, and ξ is a parameter. (In the remainder of this analysis, we will use the shorter notation $\varphi_t^{\vec{w},\vec{m}}$ as the meaning is clear from the context.) As it can be observed by the way Equation (21) is defined, a solution updates its position by performing $|W_t^i|$ movements, one for each solutions in $W_t^i$, where the position obtained in movement $s-1$ (indicated by $\vec{m}_{t,s-1}^i$) is the starting position for the next one ($\vec{m}_{t,s}^i$). Also, it is worth noticing that, since the best quality solution has an empty set $W_t^i$, the new position of this solution is obtained by adding a random vector $\xi \vec{r}_{t,s}^i$ to its current position.

The acceleration coefficient $\varphi^{\vec{w},\vec{m}}$ is computed as follows:

$$\varphi^{\vec{w},\vec{m}} = \iota \cdot e^{-\gamma|\vec{w}-\vec{m}|^2}, \tag{22}$$

where $|\vec{w}-\vec{m}|$ is the Euclidean distance between solutions $\vec{w}$ and $\vec{m}$, and γ and ι are two parameters that allow to control, respectively, the weight given to $|\vec{w}-\vec{m}|^2$ and to the exponential function. Because of the way in which $\varphi^{\vec{w},\vec{m}}$ is defined, solutions have larger displacements when they are located close to each other and smaller displacements when they are located far away.

### 3.4.1 The firefly algorithm is PSO

FA is a variant of the SDPSs proposed by Peña (2008a, 2008b) using the extrapolation coefficients of *e*PSO and the fully informed model of FiPSO. In order to explain why FA is a combination of these PSO algorithms, we will consider two cases: $|W_t^i| = 1$ and $|W_t^i| > 1$.

In the first case (i.e., when $|W_t^i| = 1$), a particle updates its position in only one movement, which allows to combine Equations (20) and (21) into one single equation as follows:

$$\vec{x}_{t+1}^i = \vec{x}_t^i + \varphi_t^{i,\vec{w}_t^i}\left(\vec{w}_t^i - \vec{x}_t^i\right) + \xi\vec{r}_t^i. \quad (23)$$

As it can be easily seen, it is possible to obtain Equation (23) from the position update rule of SDPSs (Equation (7)) by setting $\vec{y} = \vec{w}_t^i$, $\varepsilon = \varphi_t^{i,\vec{w}_t^i}$, and by computing the value of $\varphi_t^{i,\vec{w}_t^i}$ using the strategy proposed in *e*PSO (Equation (10)). The only difference is that, in FA (Equation (22)), the value of $\varphi_t^{i,\vec{w}_t^i}$ is adjusted in terms of particles' Euclidean distance, while in *e*PSO (Equation (10)) it is adjusted in terms of the difference between their objective function evaluation. Except for this minor difference, when $|W_t^i| = 1$, the position update rule of FA is a special case of the one used in SDPSs with extrapolation coefficients (*e*PSO).

In the second case (i.e., when $|W_t^i| > 1$), the mathematical model of FA (i.e., Equations (20) and (21)) involves the recursive addition of $|W_t^i|$ exponentially weighted vectors. In this case, the mathematical model of FA is the result of using the same algorithm components mentioned before (i.e., the position update rule of SDPSs Equation (7) and the extrapolation coefficients of *e*PSO (Equation (10)) with a particular case of the fully informed model of FiPSO (Equation (6)). Different from FiPSO, where particles add as many vectors as their number of neighbors, in the version of the fully informed model used in FA, particles only add the vectors of those neighbors with better quality than their own.

### 3.4.2 The metaphor of fireflies flashing/brightening

Although the author of FA says that the algorithm is inspired by the “flashing behavior of fireflies,” which consists of *short*, *rhythmic flashes* that fireflies produce (Yang, 2009, p. 171), the only idea the author used to develop the algorithm is that “fireflies are attracted towards other brighter fireflies.”

Most of the metaphor of “fireflies brightening” (as opposed to the one of “fireflies flashing”) is explained in terms of the different set of values that can be obtained by varying the value of parameter γ (Equation (22)), for which the author considered two limit cases. The first case is when γ goes to 0 and the value of $\varphi^{\vec{w},\vec{m}}$ goes to 1, making the attraction among fireflies constant regardless of their distance in the search space. In the imagery of the fireflies metaphor, this is the case when “the light intensity does not decay in an idealized sky” and “fireflies can be seen anywhere in the domain” (Yang, 2009, p. 174). The second case is when γ goes to ∞ and the value of $\varphi^{\vec{w},\vec{m}}$ goes to 0, which makes the attractiveness among fireflies negligible and new solutions are created only by means of the random perturbation $\xi\vec{r}^{\,i}_{t,s}$ (see Equation (21)). According to the author, this is the case when fireflies are either “short-sighted because they are randomly moving in a very foggy region,” or (for reasons not explained in the paper) “fireflies feel almost zero attraction to other fireflies.”

As we mention before, FA is not really inspired by the behavior of “fireflies flashing,” but on the phenomenon of light attenuation (i.e., the reduction in intensity of a light beam as the beam propagates in matter due to the joint action of the absorption and scattering of light) and the connection that the author makes between light intensity and the objective function of an optimization problem. For the author of FA, since fireflies produce light through bioluminescence, they can represent candidate solutions for an optimization problem, and since the quality of the solution can be associated with the intensity of the light it emits, it follows that the “brighter” the “firefly,” the better the solution it represents and the more “attractive” it becomes to other “fireflies.”

The reasons why FA does not meet the criterion of usefulness are as follows: (i) it is unclear what is the optimization behavior that the author observed in the behavior of “fireflies flashing” that can be used to develop a new optimization algorithm; (ii) the central elements in the “fireflies flashing” behavior (i.e., *short* and *rhythmic* light flashes) are not considered in the algorithm design of FA; and (iii) in the metaphor of “fireflies brightening,” the association between “fireflies” and “brightness” adds only a new, unnecessary terminology to refer to the concepts of *solution* and *solution quality*. In addition to this, FA does not fulfill the criterion of novelty because, as we showed in the previous section, FA uses the same ideas that were proposed before in SDPSs, *e*PSO, and FiPSO.

## Bat algorithm

BA (Yang, 2010) is a population-based algorithm in which new solutions are generated in two possible ways: (i) by identifying good search directions that are estimated based on the position of $\vec{g}_t$, or (ii) by generating a random point around $\vec{g}_t$ and accepting it on the basis of stochastic criteria. In BA, each solution has two parameters associated: $\rho^i_t$, which is the probability of randomly generating a solution around $\vec{g}_t$ that increases over time, and $\zeta^i_t$, which is the probability of accepting the new solution that decreases over time. The position update rule in BA is the following:

$$\vec{x}^i_{t+1} = \begin{cases} \vec{g}_t + \hat{\zeta}_t \vec{r}^i_t, & \text{if} \texttt{Generate} \wedge \texttt{Accept} \\ \vec{x}^i_t + \vec{v}^i_{t+1} & \text{if} \left(\texttt{Generate} \wedge (\neg \texttt{Accept})\right) \vee (\neg \texttt{Generate}) \end{cases}, \tag{24}$$

where $\hat{\zeta}_t$ is the average of the parameters $\zeta^i_t$ of all the solutions in the population, and $\vec{r}^i_t$ is a vector with values randomly distributed in $\mathscr{U}[-1,1]$. The logical variables **Generate** and **Accept** are defined as follows:

$$\begin{aligned} \texttt{Generate} \; &:= \begin{cases} \texttt{TRUE} & \text{if } \rho^i_t > \mathscr{U}[0,1] \\ \texttt{FALSE} & \text{otherwise} \end{cases}, \\ \texttt{Accept} \; &:= \begin{cases} \texttt{TRUE} & \text{if } \left(f(\vec{z}^i_t) < f(\vec{g}_t)\right) \wedge \left(\mathscr{U}[0,1] < \zeta^i_t\right) \\ \texttt{FALSE} & \text{otherwise} \end{cases}, \end{aligned} \tag{25}$$

where $f(\cdot)$ refers to the objective function of a minimization problem.

In BA, at each iteration *t* and with probability $\rho^i_t$, a solution *i* generates a random point around $\vec{g}_t$ that it keeps in a variable $\vec{z}^i_t$. The newly generated point $\vec{z}^i_t$ is accepted as the new position of *i* only when **Accept**:= TRUE, which happens when two conditions are met: first, the quality of $\vec{z}^i_t$ is higher than that of $\vec{g}_t$, and second, $\vec{z}^i_t$ is accepted with probability $\zeta^i_t$.

In the case when either **Generate**:= FALSE (i.e., the random solutions was never generated) or **Accept**:= FALSE (i.e., $\vec{z}^i_t$ was rejected), solution *i* generates a velocity vector ($\vec{v}^i_t$) that is added to its current position $\vec{x}^i_t$, as shown in the second case of Equation (24). Vector $\vec{v}^i_{t+1}$ is computed as follows:

$$\vec{v}^i_{t+1} = \vec{v}^i_t + \vec{d}^i_t \odot (\vec{g}_t - \vec{x}^i_t) \tag{26}$$

with

$$\vec{d}^i_t = \varphi_{min} + \vec{a}^i_t(\varphi_{max} - \varphi_{min}), \tag{27}$$

where $\varphi_{min} < \varphi_{max}$ are two parameters and $\vec{a}^i_t$ is a vector whose values are sampled from $\mathscr{U}[0,1]$.

The equations to update the probabilities $\rho^i_t$ and $\zeta^i_t$ are as follows:

$$\begin{aligned} \rho^i_{t+1} &= \rho_0(1 - e^{-\beta_1 t'}) \\ \zeta^i_{t+1} &= \begin{cases} \beta_2 \zeta^i_t & \text{if} \texttt{Generate} \wedge \texttt{Accept} \\ \zeta^i_t & \text{otherwise}, \end{cases} \end{aligned} \tag{28}$$

where $\beta_1 > 0$ and $0 < \beta_2 < 1$ are parameters, $t'$ is an iteration counter that is updated every time Generate$\wedge$ Accept $:= \texttt{TRUE}$, and $\rho_0$ is the initial value of parameter ρ. Note that, in Equation (28), the value of $\rho_t^i$ tends to $\rho_0$ and the value of $\zeta_t^i$ tends to 0. Also, note that, as the value $\zeta_t^i$ decreases with the number of iterations, so does the value of $\hat{\zeta}_t$; therefore, for increasing *t* values, the solutions generated in the first case of Equation (24) will be closer and closer to $\vec{g}_t$.

### 3.5.1 The bat algorithm is PSO

BA is a simplified variant of the StdPSO algorithm combined with a simulated annealing (SA) acceptance criterion (Kirkpatrick et al., 1983; Černý, 1985). First, in order to show that BA is a simplified variant of StdPSO, we compare Equations (24), (26), and (27) of BA with the position (Equation (1)) and velocity (Equation (2)) update rules of StdPSO.

In BA, the second case of Equation (24) is exactly same as the position update rules in StdPSO (Equation (1)), which consist in adding a velocity vector to a particle's current position. Also, the velocity vector is computed in the same way in both algorithms. By setting $\omega = 1$ and $\varphi_1 = 0$, the velocity update rule of StdPSO (Equation (2)) simplifies to the one of BA (Equation (26)). The perturbation component $\vec{d}_t^i$ in BA (Equation (27)) is equivalent to the term $\varphi_2 \vec{b}_t^i$ in StdPSO (Equation (2)). The only difference is that the value of the acceleration coefficient ϕ in Equation (27) is computed in the range $[\varphi_{min}, \varphi_{max}]$ with the goal of varying the magnitude of the perturbation induced by $\vec{a}_t^i$. One of the first PSO variants using the idea of varying the value of the control parameters of PSO is the “time-varying acceleration coefficient PSO” (Ratnaweera et al., 2004), in which the value of $\phi_2$ linearly increases from $\varphi_{min}$ to $\varphi_{max}$.

Now, we will compare BA with SA (Kirkpatrick et al., 1983), which is a single solution based algorithm for solving combinatorial optimization problems proposed in the early 1980s. As shown in Equation (27), BA uses the concept of generating new solutions around $\vec{g}_t$ and accepting them on the basis of a decreasing probability (parameter $\zeta_t^i$). This idea comes originally from SA, where a parameter called *temperature* (*T*) that decreases over time is used to decided when to accept a worsening solution. In the context of SA, the so-called *cooling scheme* is used to control the updates of parameter *T* iteration after iteration (Franzin and Stützle, 2019). In BA, parameter $\zeta_t^i$ plays the same role as parameter *T* in SA. Also, the model used in BA to update the value of $\zeta_t^i$ (Equation (28)) is the same as the well-known *geometric cooling scheme*, which was proposed in the very first version of SA by Kirkpatrick et al. (1983). One minor difference is that, in BA, the value of $\zeta_t^i$ is updated only when a solution is accepted, while in SA the value of *T* is typically updated at the end of each iteration.

### 3.5.2 The metaphor of bats echolocation

The author of BA says that the inspiration for this algorithm is “the behavior of echolocation that some bats species use to find preys, avoid obstacles and discriminate between different objects" (Yang, 2010, p. 66). For developing BA, the author "idealized" several aspects of this behavior. In the words of the author, it was assumed that (i) "all bats use echolocation for sensing distance," (ii) "bats are able to differentiate in some magical way between food/prey and background barriers," (iii) "bats can automatically adjust the frequency and rate in which they are emitting sound," and (iv) "the loudness of their sound can only decrease from a large value to a minimum constant" (Yang, 2010, pp. 67–68).

To explain BA, the author considered that bats have two different flying modes, which correspond to the two cases in Equation (24). In the first mode, bats fly randomly adjusting their "pulse emission rate" $\rho_t^i$ and "loudness" $\zeta_t^i$. According to the author, small values for parameter $\rho_t^i$ and large values for $\zeta_t^i$ represent when "bats are randomly searching for a prey," while the opposite case (i.e., large values for $\rho_t^i$ and small ones for $\zeta_t^i$) represent when "bats have found a prey and temporarily stop emitting any sound" (Yang, 2010, p. 70). The mathematical model of bats adjusting their "pulse emission rate" and "loudness" is given in Equation (28). For the second flying mode, which is modeled using Equations (26) and (27), the authors considered that bats control their step size and range of movement by adjusting their "sound frequency" (modeled by vector $\vec{d}_t^i$ in Equation (27)) and by moving toward the best bat in the swarm; parameters $\varphi_{min}$ and $\varphi_{max}$ represent "the range of frequencies in which bats emit their sound."

As there are so many simplifications and unrealistic assumptions in the behavior of "bats echolocation" as idealized by Yang (2010), it is impossible to understand how this behavior was taken into account at all for developing BA. Consider, for example, the unlikely ideas that bats use the location of the global best bat while hunting, or that the loudness of their sound can only decrease, or that they have a "magical" ability to differentiate between food/prey and background barriers. For this reason, BA does not meet the criterion of usefulness. Also, as we showed in the previous section, BA uses concepts originally proposed in StdPSO and SA that were published, respectively, in 1995 and 1983, thereby failing also in the criterion of novelty. Unfortunately, it seems evident to us that the only goal of using the metaphor of "bats echolocation" has been to hide the fact that the algorithm is unnecessary as its only contribution is a bats-inspired terminology to refer to well-known concepts.

## Antlion optimizer

In ALO (Mirjalili, 2015a), a population of μ solutions is randomly selected for recombination with the global best solution ($\vec{g}_t$) in order to produce λ new solutions, with $\lambda = \mu$. In ALO, at the beginning of each iteration, each solution in the population *P* is given a probability of being selected for recombination with $\vec{g}_t$ as follows:

$$\Pr(\vec{x}_t^k) = \frac{\texttt{fit}(\vec{x}_t^k)}{\sum_{z=1}^{|P|} \texttt{fit}(\vec{x}_t^z)}, \tag{29}$$

where $\Pr(\vec{x}_t^k)$ is the probability of selecting solution $\vec{x}_t^k \in P$ and $\texttt{fit}(\cdot)$ is a function that maps the quality of a solution with a real positive value, so that the better the quality of a solution, the greater the value returned by $\texttt{fit}(\cdot)$. It is worth noting that, since $\vec{g}_t$ is an element of *P*, the solution with higher probability of being selected for recombination with $\vec{g}_t$ is the solution $\vec{g}_t$ itself.

The equation used to recombine solutions in order to create the set of λ new solutions is

$$\vec{x}_{t+1}^i = \frac{\vec{x}_t^{\prime k} + \vec{g}_t^{\prime i}}{2} \text{for} i = 1, \cdots, \lambda, \tag{30}$$

where $\vec{x}_t^{\prime k}$ is a vector generated by perturbing solution $\vec{x}_t^k$ that has been randomly selected with repetitions from *P* based on the probabilities computed using Equation (29), and $\vec{g}_t^{\prime i}$ is a vector generated by perturbing $\vec{g}_t$.

The computation of vectors $\vec{x}_t'^k$ and $\vec{g}_t'^i$ is done using an elaborated procedure that involves performing a random walk of *s*-steps for each dimension of each vector. This procedure is defined as follows:

$$u^j = \frac{\left(\mathscr{R}^j_{t,s=\varrho} - \mathtt{min}\left(\mathscr{R}^j_t\right)\right) \times \left(\alpha^j_t - \eta^j_t\right)}{\mathtt{max}\left(\mathscr{R}^j_t\right) - \mathtt{min}\left(\mathscr{R}^j_t\right)} + \eta^j_t, \mathrm{for} j = 1, \cdots, d; \mathrm{for} \vec{u} \in \{\vec{x}^k_t, \vec{g}_t\}, \tag{31}$$

where vector $\vec{u}$ is a variable used to iterate between vectors $\vec{x}^k_t$ and $\vec{g}_t$, $\mathscr{R}^j_t$ is a sequence of ϱ values computed using a one-dimensional random walk on $\mathbb{Z}$ that starts at 0 and moves to +1 or −1 with equal probability at each step *s*, $\mathscr{R}^j_{t,s=\varrho}$ indicates the last value in the sequence $\mathscr{R}^j_t$, and functions $\mathtt{min}(\cdot)$ and $\mathtt{max}(\cdot)$ return, respectively, the minimum and maximum values in $\mathscr{R}^j_t$. Vectors $\vec{\alpha}_t$ and $\vec{\eta}_t$, which are computed as

$$\vec{\alpha}_t = \begin{cases} (x^j_u / \psi_t) + u^j & \mathrm{if} \mathscr{U}(0,1) > 0.5 \\ (-x^j_u / \psi_t) + u^j & \mathrm{otherwise} \end{cases}, \forall j \tag{32}$$

and

$$\vec{\eta}_t = \begin{cases} (x^j_l / \psi_t) + u^j & \mathrm{if} \mathscr{U}(0,1) > 0.5 \\ (-x^j_l / \psi_t) + u^j & \mathrm{otherwise} \end{cases}, \forall j, \tag{33}$$

allow to use a fraction $1/\psi_t$ of the upper ($x^j_u$) and lower ($x^j_l$) bounds of each dimension of the vector being perturbed to normalize the values of the random walk $\mathscr{R}^j_t$ around them. Finally, to create the population that will be used in the next iteration, the best μ solutions are selected from the set of $\mu + \lambda$ solutions at the end of each iteration.

### 3.6.1 The antlion optimizer is an ES

As the reader familiar with EAs must have realized by now, the algorithm components proposed for ALO are similar to those proposed in the context of ESs—see Section 2.2. In particular, Equation (29) corresponds to the *fitness proportional* parental selection, Equation (30) corresponds to the *intermediate* recombination, and Equation (31) corresponds to the *spherical/isotropic* mutation. In fact, as ALO uses the *elitist* $(\mu + \lambda)$ survival selection mechanism, it is virtually same as the $(\mu + \lambda) - $ ES (Schwefel, 1981), where the only difference is that, in ALO, mutation is applied before recombination, while in the $(\mu + \lambda) - $ ES it is applied after recombination.

Although most of the components in ALO can be easily recognized as evolutionary operators, this may not be necessarily the case for the *spherical/isotropic* mutation. The mutation operator implemented in ALO (Equation (31)) uses a complex procedure that involves computing a random walk (Equation (31)) and two other equations (Equations (32) and (33)) to mutate each dimension of a vector, whereas the procedure implemented in ESs (Equation (11)) requires sampling normally distributed values. The one-dimensional random walk over $\mathbb{Z}$ implemented in ALO (that starts at 0 and adds +1 or −1 with equal probability at each time step *s*) is commonly known in the literature as the *simple isotropic random walk* (Codling et al., 2008) and its diffusion model is given by the Gaussian distribution with mean 0 and variance *s* for a sufficiently large number of time steps *s*. Using this fact, the complex procedure proposed for ALO in Equation (31) to compute vectors $\vec{x}_t'^k$ and $\vec{g}_t'^i$ can be reformulated in a much simpler and computationally efficient way as follows:

$$\vec{x}_t'^k = \vec{x}_t^k + \mathscr{N}(0, \psi_t \mathbf{I}), \tag{34}$$

$$\vec{g}_t'^i = \vec{g}_t + \mathscr{N}(0, \psi_t \mathbf{I}), \tag{35}$$

where $\psi_t$ is the time-varying parameter that the authors of ALO introduced in Equations (32) and (33).

### 3.6.2 The metaphor of antlions hunting

The author of ALO says that the inspiration for the algorithm is the "intelligent behavior of antlions in hunting ants in nature" (Mirjalili, 2015a, p. 81). According to the description of this behavior provided in the ALO paper, the strategy that antlions use for hunting consists in the following steps: (i) digging a cone-shaped pit in the sand, (ii) hiding underneath the sand at the bottom of the pit, and (iii) waiting for prey to fall in the pit so that they can catch it.

In ALO, the μ solutions at the beginning of each iteration represent "antlions" and the fitness value of the solutions—computed using $\texttt{fit}(\cdot)$ in Equation (29)—represent "the size of the pit in which the antlion constructed and is hidden at the bottom." Accordingly, in the imagery of the metaphor, the "fitter" the "antlion," the larger is its pit and the higher are its chances of catching an ant. To model the behavior of ants randomly moving in the sand, the author used Equations (31)–(33) that involve performing a random walk on $\mathbb{Z}$. For Mirjalili (2015a), the computation of vectors $\vec{x}_t'^k$ and $\vec{g}_t'^i$ in Equation (31) represent the "influence of antlions pits on the random walk of ants," and computing the arithmetic average of those two vectors (Equation (30)) represents "the final position of the ant." Also, when a new solution has higher quality than the one used to create it—that is, when $f(\vec{x}_{t+1}^i)$ is better than $f(\vec{x}_t'^k)$—this represents the case when "an ant reaches the bottom of the pit and is caught in the antlion's jaw" (Mirjalili, 2015a, p. 83). Finally, the fact that vector $\vec{g}_t$ is always used when recombining two solutions (Equation (30)) represents the fact that "each ant can be caught by an antlion in each iteration and the elite (fittest antlion)" (sic) (Mirjalili, 2015a, p. 82).

As the reader should have realized by now, the metaphor of "antlions hunting" is nothing but a far-fetched way to explain the proposed ALO. The "novel" ALO proposed in 2015 does not meet the criterion of novelty because, as we showed in the previous section, it is a variant of the $(\mu + \lambda) -$ ES proposed roughly 35 years before ALO. Also, the mathematical model derived from the metaphor of "antlions hunting" resulted in an inefficient procedure to perturb solutions (Equations (31)–(33)) that produces the same type of perturbation as the simple *spherical/isotropic* mutation (Equation (11)). Therefore, ALO does not meet the criterion of usefulness. Finally, it is worth pointing out that, similar to what we observed for the other metaphors analyzed in this article, many of them proposed by the same author, it is impossible to understand what is the optimization process/component observed by the author in the behavior of antlions that inspired him to propose this algorithm.

## 4 Discussion

### Why some metaphors work while others do not?

In the period comprised between the early 1980s and the beginning of 2000s, the field of metaheuristics was characterized by a rapid growth in the number of new methods proposed to tackle complex optimization problems and by many experimental and theoretical studies of such methods investigating their properties and ways to extend their capabilities (Corne et al., 1999; Sörensen et al., 2017; Gendreau and Potvin, 2019). The field also witnessed the introduction of a few very successful techniques inspired by natural phenomena such as EAs (Fogel et al., 1966; Rechenberg, 1971; Holland, 1975; Schwefel, 1977, 1981), where the inspiration is the phenomenon of evolution by natural selection and survival of the fittest; SA (Kirkpatrick et al., 1983; Černý, 1985), inspired by the metal annealing process whereby atoms reorganize themselves to minimize an energy function; *ant colony optimization* (Dorigo et al., 1991; Dorigo, 1992), inspired by the foraging behavior of ants; and PSO (Kennedy and Eberhart, 1995), inspired by the dynamics and social interaction of bird flocks. These methods not only attracted the attention of scientists and practitioners interested in solving relevant problems for which other methods fail to provide satisfactory results but also led to one of the most widespread beliefs in the field, that is, "nature is a never ending source of inspiration" to tackle complex optimization problems.

While there are examples of natural behaviors that have been useful to design new and efficient algorithms, such as those that inspired the metaheuristics mentioned above, the truth is that finding a natural behavior that leads to *useful* and *novel* ideas on how to solve optimization problems turns out to be very difficult. Indeed, in many cases the authors of these *novel* metaheuristics are blinded by their desire of introducing a successful novel algorithm and do not realize that the behaviors they take inspiration from are irrelevant for the design of an optimization algorithm. A clear example is given by the six algorithms analyzed in this paper. They are all based on interesting behaviors exhibited by different species of animals; however, when these behaviors are used as a source of inspiration to devise optimization algorithms, they turn out to be ultimately useless or to lead to the same ideas that were explored in already published algorithms. Indeed, as shown by the growing list of metaphor-based algorithms for which it was demonstrated that they are either same or have little variations of well-known techniques (Weyland, 2010; Piotrowski et al., 2014; Weyland, 2015; Camacho-Villalón et al., 2018, 2019, 2020, 2022; Sörensen et al., 2019), it is common that introducing metaphors of new behaviors results in algorithms that lack significant novelty. Even worse, these algorithms are very often "inspired" by such unusual metaphors—for example, *football*, *reincarnation*, *chickens*, and *zombies* (Campelo and Aranha, 2021)—that they look more like jokes than serious, scientifically motivated algorithms.

Even in those cases in which a behavior is original, scientifically motivated and abstracted in a proper way to design an optimization algorithm, there is still the problem that the algorithm derived from the new metaphor may be equivalent to one that has already been published in the literature. An example of this is the *biogeography-based optimization* (Simon, 2008) that, as its author showed in a later study (Simon et al., 2011), is "a generalization of a genetic algorithm with global uniform recombination," which was proposed in the 1960s.

## Metaphor-based algorithms motivated on theoretical research

In many papers proposing "novel" metaphor-based algorithms for the approximate solution of continuous problems, a common thread that is as unfortunate as it is exasperating is the mention of the NFL theorems formulated by Wolpert and Macready (1997) for finite search spaces as the theoretical foundation that motivates their authors, such as in the following examples:

> "Obviously, the No Free Lunch theorem makes this field of study highly active which results in enhancing current approaches and proposing new meta-heuristics every year. This also motivates our attempts to develop a new meta-heuristic with inspiration from grey wolves.
> (Mirjalili et al., 2014, pp. 46–47)"

> Some of the most popular algorithms in this field are: Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Differential Evolution (DE), Evolutionary Programming (EP). Although these algorithms are able to solve many real and challenging problems, the so-called No Free Lunch theorem allows researchers to propose new algorithms.
> (Mirjalili, 2015a, p. 81)

> According to the "no-free-lunch" (NFL) theory, it is difficult to employ a single meta-heuristic algorithm in striving to solve all possible optimization problems. … This has been a motive for the researchers in this field, as well as ourselves, to look for new and innovative nature-inspired methods to solve and show superior scores on the current and new hard real-life problems. The door is still open, and here we present a novel meta-heuristic algorithm based on human behavior with the very famous tale of Ali Baba and the forty thieves, as our inspiration targeting numerical optimization problems.
> ("A novel meta-heuristic algorithm for solving numerical optimization problems: Ali Baba and the forty thieves" published in 2021 in *Neural Computing and Applications*, pp. 1–47.)

While the main implication of the NFL is that "all search heuristics have the same performance when averaged over the uniform distribution over all possible objective functions," the assumptions that lead to this implication have been criticized as *restrictive* and *unrealistic*. Indeed, it has been demonstrated that the NFL is unrealistic in black-box optimization scenarios (i.e., where the objective function can be queried but its definition remains unknown) and that the theorems do not hold for multi-objective problems and continuous domains —being this latter the case of three algorithms in the above quotations. A detailed discussion of the implications of the NFL in problems that are relevant in practice can be found in (Auger and Teytaud, 2007, 2010) and in the references cited in these papers.

However, beyond the real implications of the NFL, it is obvious that faster and more performing optimization methods are necessary to tackle the ever more challenging optimization problems arising in all fields and disciplines. This necessity has stressed the urgency of establishing higher scientific standards that allow us to develop new techniques on the basis of the understanding of the weaknesses and strengths of the existing techniques. Unfortunately, rather than contributing to this goal, metaphor-based algorithms are leading the field astray by filling it with noise, creating confusion in the literature, and making the use of unscientific practices to seem *normal* in a scientific field. Furthermore, as it is illustrated in the quotations above, the authors of metaphor-based algorithms seek to convey the idea that advancing the field means finding more metaphors to develop more “novel” algorithms, despite there is growing evidence (as the one provided in this paper) pointing to the fact that metaphors are useless for this purpose and they only serve to hide the lack of novelty of these algorithms.

## 5 Conclusions

In this paper, we perform a rigorous, component-based analysis of six widespread metaphor-based algorithms—GWO, MFA, WOA, FA, BA, and ALO—in which we first identify the ideas proposed in each of them, and then, we compare them with those that have been proposed in the context of PSO and ESs. We show that, although the six algorithms were proclaimed as “novel” approaches by their authors, they lack any novelty, as GWO, MFA, WOA, FA, BA are variants of PSO, and ALO is a variant of ESs. Also, we evaluate the metaphors that inspired the six metaphor-based algorithms according to the criteria of usefulness and novelty that we defined, and found that none of them have been useful to develop a novel optimization algorithm, nor is there a sound motivation that justifies their use. Finally, we discuss (i) the reasons why finding a natural/artificial behavior that can be useful to design a new optimization algorithms is something rather difficult and mostly rare; and (ii) a series of statements frequently presented as motivation by the authors of metaphor-based algorithms, which are based on a wrong understanding of the NFL theorems and/or a lack of knowledge of the theoretical advances in the field.

As we have discussed here and in several other papers (Camacho-Villalón et al., 2018, 2019, 2020, 2022; Aranha et al., 2022), the publication of metaphor-based algorithms has detrimental effects to the field of metaheuristics, such as creating confusion in the literature, hindering our understanding of the existing metaheuristic algorithms, and making it troublesome to compare algorithms both conceptually and experimentally. Despite the ample evidence suggesting that the only purpose of framing algorithms into new metaphors is to conceal their similarities with other techniques published before and to make it difficult to see that there is nothing really novel in them, proving that this is the case for all (or a great majority) of them would be very challenging due to the myriad of algorithms of this type already published and the fact that more appear all too often. Therefore, we consider it urgent that metaphor-based algorithms are not published anymore unless their authors (i) present the algorithms using the standard optimization terminology, and (ii) are able to show that the new behavior leads to new ideas that are useful in optimization.

## Acknowledgments

Christian Leonardo Camacho Villalón, Marco Dorigo, and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are, respectively, FNRS Aspirant and Research Directors.

## References

Akbari, R., Ziarati, K., 2011. A rank based particle swarm optimization algorithm with dynamic adaptation. *Journal of Computational and Applied Mathematics* **235**, 8, 2694– 2714.

Aranha, C., Camacho-Villalón, C.L., Campelo, F., Dorigo, M., Ruiz, R., Sevaux, M., Sörensen, K., Stützle, T., 2022. Metaphor-based metaheuristics, a call for action: the elephant in the room. *Swarm Intelligence* **16**, 1, 1– 6.

Arumugam, M.S., Murthy, G.R., Rao, M., Loo, C.X., 2007. A novel effective particle swarm optimization like algorithm via extrapolation technique. 2007 International Conference on Intelligent and Advanced Systems, IEEE, Piscataway, NJ, pp. 516– 521.

Arumugam, M.S., Rao, M.V.C., Tan, A.W., 2009. A novel and effective particle swarm optimization like algorithm with extrapolation technique. *Applied Soft Computing* **9**, 1, 308– 320.

Auger, A., Teytaud, O., 2007. Continuous lunches are free! Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, Association for Computing Machinery, New York, pp. 916– 922.

Auger, A., Teytaud, O., 2010. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica* **57**, 1, 121– 146.

Bäck, T., Fogel, D.B., Michalewicz, Z., 1997. *Handbook of Evolutionary Computation*. IOP Publishing, Bristol.

Camacho-Villalón, C.L., Dorigo, M., Stützle, T., 2018. Why the intelligent water drops cannot be considered as a novel algorithm. In M. Dorigo, M. Birattari, C. Blum, A.L. Christensen, A. Reina, V. Trianni (eds) *Swarm Intelligence*, 11th International Conference, ANTS 2018, Springer, Berlin, pp. 302–314.

Camacho-Villalón, C.L., Dorigo, M., Stützle, T., 2019. The intelligent water drops algorithm: why it cannot be considered a novel algorithm. *Swarm Intelligence* **13**, 3–4, 173– 192.

Camacho-Villalón, C.L., Dorigo, M., Stützle, T., 2022. An analysis of why cuckoo search does not bring any novel ideas to optimization. *Computers & Operations Research* **142**, 105747.

Camacho-Villalón, C.L., Stützle, T., Dorigo, M., 2020. Grey wolf, firefly and bat algorithms: three widespread algorithms that do not contain any novelty. International Conference on Swarm Intelligence, Springer, Berlin, pp. 121– 133.

Campelo, F., Aranha, C., 2021. Evolutionary computation bestiary. Available at https://github.com/fcampelo/EC-Bestiary (accessed 26 March 2021).

Černý, V., 1985. A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications* **45**, 1, 41– 51.

Clerc, M., 2010. *Particle Swarm Optimization*, Vol. **93**. John Wiley & Sons, Hoboken, NJ.

Clerc, M., 2011. *Standard Particle Swarm Optimisation from 2006 to 2011*. Independent Consultant, Jersey City, NJ.

Clerc, M., Kennedy, J., 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation* **6**, 1, 58– 73.

Codling, E.A., Plank, M.J., Benhamou, S., 2008. Random walk models in biology. *Journal of the Royal Society Interface* **5**, 25, 813– 834.

Corne, D., Dorigo, M., Glover, F., Dasgupta, D., Moscato, P., Poli, R., Price, K.V., 1999. *New Ideas in Optimization*. McGraw Hill, New York.

Dorigo, M., 1992. Optimization, learning and natural algorithms. Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy.

Dorigo, M., Maniezzo, V., Colorni, A., 1991. The ant system: an autocatalytic optimizing process. Technical Report 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Italy.

Fogel, D.B., Owens, A.J., Walsh, M.J., 1966. *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, Hoboken, NJ.

Franzin, A., Stützle, T., 2019. Revisiting simulated annealing: a component-based analysis. *Computers & Operations Research* **104**, 191– 206.

Gendreau, M., Potvin, J.Y., 2019. *Handbook of Metaheuristics*, *International Series in Operations Research & Management Science*. Vol. 272. Springer, Berlin.

Grefenstette, J., 2000. Rank-based selection. In: T. Back, D.B. Fogel, T. Michalewicz (eds) *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, Bristol, pp. 187– 194.

Hansen, N., Arnold, D.V., Auger, A., 2015. Evolution strategies. In *Springer Handbook of Computational*

*Intelligence*. Springer, Berlin, pp. 871– 898.

Holland, J.H., 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Jordan, J., Helwig, S., Wanka, R., 2008. Social interaction in particle swarm optimization, the ranked FIPS, and adaptive multi-swarms. GECCO'08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, Association for Computing Machinery, New York, pp. 49– 56.

Kennedy, J., 1999. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. Proceedings of the 1999 Congress on Evolutionary Computation—CEC99 (Cat. No. 99TH8406), Vol. **3**, IEEE, Piscataway, NJ, pp. 1931– 1938.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. Proceedings of ICNN'95—International Conference on Neural Networks, Vol. **4**, IEEE, Piscataway, NJ, pp. 1942– 1948.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. *Science* **220**, 671– 680.

Lones, M.A., 2014. Metaheuristics in nature-inspired algorithms. In C. Igel, D.V. Arnold (eds) *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2014*, ACM Press, New York, NY, pp. 1419– 1422.

Lones, M.A., 2020. Mitigating metaphors: a comprehensible guide to recent nature-inspired algorithms. *SN Computer Science* **1**, 1, 1– 12.

Mendes, R., 2004. Population topologies and their influence in particle swarm performance. PhD final dissertation, Departamento de Informática Escola de Engenharia Universidade do Minho.

Mendes, R., Kennedy, J., Neves, J., 2004. The fully informed particle swarm: simpler, maybe better. *IEEE Transactions on Evolutionary Computation* **8**, 3, 204– 210.

Mirjalili, S., 2015a. The antlion optimizer. *Advances in Engineering Software* **83**, 80– 98.

Mirjalili, S., 2015b. Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. *Knowledge-Based Systems* **89**, 228– 249.

Mirjalili, S., Lewis, A., 2016. The whale optimization algorithm. *Advances in Engineering Software* **95**, 51– 67.

Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. *Advances in Engineering Software* **69**, 46– 61.

Montes de Oca, M.A., 2011. Incremental social learning in swarm intelligence systems. PhD thesis, IRIDIA, École polytechnique, Université Libre de Bruxelles, Belgium.

Peña, J., 2008a. Simple dynamic particle swarms without velocity. In M. Dorigo et al. (eds) *Ant Colony Optimization and Swarm Intelligence*, 6th International Conference, ANTS 2008, Springer, Heidelberg, pp. 144– 154.

Peña, J., 2008b. Theoretical and empirical study of particle swarms with additive stochasticity and different recombination operators. GECCO'08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, Association for Computing Machinery, New York, pp. 95– 102.

Piotrowski, A.P., Napiorkowski, J.J., Rowinski, P.M., 2014. How novel is the “novel” black hole optimization approach? *Information Sciences* **267**, 191– 200.

Ratnaweera, A., Halgamuge, S.K., Watson, H.C., 2004. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation* **8**, 3, 240– 255.

Rechenberg, I., 1971. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. PhD thesis, Department of Process Engineering, Technical University of Berlin.

Rechenberg, I., 1973. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, Germany.

Ryan, C., 2008. Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, July 12–16, 2008. ACM Press, New York, NY.

Schwefel, H.P., 1977. *Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie*. Birkhäuser, Basel.

Schwefel, H.P., 1981. *Numerical Optimization of Computer Models*. John Wiley & Sons, Hoboken, NJ.

Shi, Y., Eberhart, R., 1998. A modified particle swarm optimizer. In P.K. Simpson, K. Haines, J. Zurada, D. Fogel (eds) *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation (ICEC'98)*, IEEE Press, Piscataway, NJ, pp. 69– 73.

Shi, Y., Eberhart, R., 1999. Empirical study of particle swarm optimization. Proceedings of the 1999 Congress on Evolutionary Computation (CEC 1999), IEEE Press, Piscataway, NJ, pp. 1945– 1950.

Simon, D., 2008. Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation* **12**, 6, 702– 713.

Simon, D., Rarick, R., Ergezer, M., Du, D., 2011. Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms. *Information Sciences* **181**, 7, 1224– 1248.

Sörensen, K., 2015. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research* **22**, 1, 3– 18.

Sörensen, K., Arnold, F., Palhazi Cuervo, D., 2019. A critical analysis of the “improved clarke and wright savings algorithm”. *International Transactions in Operational Research* **26**, 1, 54– 63.

Sörensen, K., Glover, F., 2013. Metaheuristics. In S.I. Gass, M.C. Fu (eds) *Encyclopedia of Operations Research and Management Science* ( 3rd edn). Springer, Berlin, pp. 960– 970.

Sörensen, K., Sevaux, M., Glover, F., 2017. A history of metaheuristics. Available at https://hal.archives-ouvertes.fr/hal-01496372 .

Weyland, D., 2010. A rigorous analysis of the harmony search algorithm: how the research community can be misled by a “novel” methodology. *International Journal of Applied Metaheuristic Computing* **12**, 2, 50– 60.

Weyland, D., 2015. A critical analysis of the harmony search algorithm: how not to solve Sudoku. *Operations Research Perspectives* **2**, 97– 105.

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

Yang, X.S., 2009. Firefly algorithms for multimodal optimization. International Symposium on Stochastic Algorithms, Springer, Berlin, pp. 169– 178.

Yang, X.S., 2010. A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), *Studies in Computational Intelligence*, Vol. **284**. Springer, Berlin, pp. 65–74.

Zambrano-Bigiarin, M., Clerc, M., Rojas, R., 2013. Standard particle swarm optimisation 2011 at CEC-2013: a baseline for future PSO improvements. Proceedings of the 2013 Congress on Evolutionary Computation (CEC 2013), IEEE Press, Piscataway, NJ, pp. 2337– 2344.

## Citing Literature

Download PDF