Grey Wolf, Firefly and Bat Algorithms: Three Widespread Algorithms that Do Not Contain Any Novelty

C.L. Camacho Villalón, T. Stützle, and M. Dorigo

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2020-002

May 2020
Last revision: March 2021

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:

IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2020-002

Revision history:

TR/IRIDIA/2020-002.001	May 2020
TR/IRIDIA/2020-002.002	March 2021

Grey Wolf, Firefly and Bat Algorithms: Three Widespread Algorithms that Do Not Contain Any Novelty

Christian Leonardo Camacho Villalón[0000−0002−0182−3469], Thomas Stützle[0000−0002−5820−0473], and Marco Dorigo[0000−0002−3971−0507]

IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
{ccamacho,stuetzle,mdorigo}@ulb.ac.be

Abstract. In this paper, we carry out a review of the grey wolf, the firefly and the bat algorithms. We identify the concepts involved in these three metaphor-based algorithms and compare them to those proposed in the context of particle swarm optimization. We provide compelling evidence that the grey wolf, the firefly, and the bat algorithms are not novel, but a reiteration of ideas introduced first for particle swarm optimization and reintroduced years later using new natural metaphors. These three algorithms can therefore be added to the growing list of metaphor-based algorithms—to which already belong algorithms such as harmony search and intelligent water drops—that are nothing else than repetitions of old ideas hidden by the usage of new terminology.

1 Introduction

Algorithms inspired by natural or artificial metaphors have become a common place in the stochastic optimization literature [4]. Despite being invariably presented as novel methods, many of these algorithms do not seem to be proposing novel ideas; rather, they reintroduce well-known concepts already proposed in previously published algorithms [22]. That is, the same ideas developed in the context of local search (LS) heuristics, evolutionary algorithms (EAs), and ant colony optimization (ACO), to mention a few, appear in these "novel" algorithms, although presented using new terminology. In addition to this, it is often the case that rather than clearly expressing new ideas in plain algorithmic terms and highlighting differences with what has already been proposed in the literature, authors of these algorithms focus on aspects such as the novelty and beauty of the new inspiring source.

Several are the undesirable consequences of this practice. Perhaps the most detrimental one is that it has generated a lot of confusion in the literature, since using different terminologies for referring to concepts already defined makes it difficult to compare algorithms—both conceptually and experimentally—hindering our understanding. Additionally, presenting ideas using unconventional terminology instead of the normal one used in optimization, adds an unnecessary extra effort to distinguish between what is novel and what is not.

A number of studies [25,11,16,26,2,3,23] have shown that the use of metaphors has served the sole purpose of hiding the similarities among different methods, thus allowing copies of well-known approaches to be misrepresented as new. One of the first examples of this was provided in 2010 by Weyland [25,26] in a rigorous analysis of the harmony search (HS) algorithm. In this analysis, Weyland found that this seemingly new method was the same as a particular evolutionary algorithm, called "evolution strategy $(\mu+1)$" [19], which was proposed about 30 years before the HS algorithm. Similarly, other studies in the same direction have shown that the black holes optimization algorithm is a simplification of particle swarm optimization (PSO) [16], and, more recently, that the intelligent water drops algorithm is a special case of ACO [2,3]. Even though it has been shown that a number of these algorithms are a reiteration of well-known ideas and it has been suggested that the whole trend is unhealthy and damaging for the entire research field [22,23], new algorithms and new variants of metaphor-based algorithms continue to be published with alarming high frequency.

In this paper, we review the concepts utilized in three highly-cited algorithms proposed for continuous optimization problems: the grey wolf, the firefly and the bat algorithms. We provide evidence that (i) *using new metaphors*, (ii) *changing the terminology*, and (iii) *presenting the algorithm in a confusing way* allowed to overlook the fact that they were all PSO variants.

The rest of the paper is organized as follows. In Section 2, we briefly review the basic concepts of PSO and some of its most popular variants. In Section 3, we describe the three metaphor-based algorithms that we analyze using standard PSO terminology. In this way, it becomes immediately apparent the fact that these algorithms are indeed PSO variants. In Section 4, we conclude the paper by highlighting some aspects that make the analysis of these 'novel' metaphor-inspired algorithms a challenging endeavor.

2 Particle Swarm Optimization

Particle swarm optimization is arguably the most popular swarm intelligence algorithm to tackle continuous optimization problems. It was introduced by Kennedy and Eberhart [7,9] in 1995, and is inspired by the observation of social dynamics in bird flocks. The optimization capabilities of PSO come from the repeated application of a set of rules that allow particles, which represent problem solutions, in the swarm to identify and move in promising directions in the search space that they estimate from locations that they previously visited and that correspond to good solutions [?]. Particles employ simple memory structures to keep track of important information collected during the search process, such as their own position and velocity and the position of the best among neighboring particles.

In the standard PSO (SPSO) [20,21] algorithm, each particle i knows at every iteration t its current position $\boldsymbol{x}_t^i$, its velocity $\boldsymbol{v}_t^i$, its personal best position $\boldsymbol{p}_t^i$, and the position $\boldsymbol{l}_t^i$ of its local best particle, where the local best particle is the particle j in the neighborhood of particle i that has the best personal best position $\boldsymbol{p}_t^j$. Many different types of neighborhood are possible among which star, ring, and lattices are typical options; when the neighborhood consists of all the particles in the swarm, then the local best particle is called global best and its position, which is the same for all particles i, is indicated by $\boldsymbol{g}_t$.

The rules to update a particle's position and velocity in SPSO are:

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

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

where ω is the inertia weight that controls the effect of the velocity at time t, φ_1 and φ_2 are the acceleration coefficients that weigh the relative influence of the personal best and local best position, $\boldsymbol{a}_t^i$ and $\boldsymbol{b}_t^i$ are two random vectors used to provide diversity to particle's movement, and $\odot$ indicates the Hadamard (pointwise) product between two vectors.

Over the years, many variants of PSO have been proposed to improve the optimization capabilities of the algorithm. One of them is SPSO-2011 [5,29], that was developed with the goal of preventing the issue of rotation variance that affects SPSO. In this variant, the velocity update rule—Eq. 2 above—was modified as follows:

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

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

The center $\boldsymbol{c}_t^i$ is computed as

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

where

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

The two key concepts proposed in SPSO-2011 consist of (i) the definition of the new point $\boldsymbol{c}_t^i$ in the search space defined as a function of the position of three particles in the swarm: $\boldsymbol{x}_t^i$, $\boldsymbol{p}_t^i$ and $\boldsymbol{l}_t^i$ (Eq. 4), and (ii) the use of $\boldsymbol{c}_t^i$ to generate a point $\boldsymbol{x'}_t^i$ that is then used to update the particles' velocities; the point $\boldsymbol{x'}_t^i$ is randomly selected from a hyper-spherical distribution with center $\boldsymbol{c}_t^i$ and radius $|\boldsymbol{c}_t^i - \boldsymbol{x}_t^i|$, which is invariant to rotation (Eq. 3).

Another interesting variant is the fully informed PSO (FiPSO) [12], in which the authors propose to use the whole swarm to influence particles' new velocities:

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

where $\chi = 0.7298$ is a constant value called constriction coefficient and defined in [6], T_t^i is the set of particles in the neighborhood of i, and φ is a parameter. The main innovation in this variant is a generalization of some of the algorithmic ideas used in PSO. In particular, while in most previous PSO variants only one particle (the local best) contributed to update the particles' velocities, here the number of particles that influence the velocity update becomes a design choice typically referred as model of influence.

One last example of a PSO variant relevant for our analysis in the next section is the "bare-bones" PSO [8]. This variant belongs to a class of PSO algorithms typically referred to as velocity-free variants in which the rule to update the position of particles does not include a velocity vector:

$$\boldsymbol{x}_{t+1}^{i} = \mathcal{N}\left(\frac{\boldsymbol{p}_t^i + \boldsymbol{l}_t^i}{2}, |\,\boldsymbol{p}_t^i - \boldsymbol{l}_t^i|\right) \tag{7}$$

Although in this variant the particles' new position is obtained by sampling values from a normal distribution whose center $(\boldsymbol{p}_t^i + \boldsymbol{l}_t^i)/2$ and dispersion $|\boldsymbol{p}_t^i - \boldsymbol{l}_t^i|$ are computed as a function of $\boldsymbol{p}_t^i$ and $\boldsymbol{l}_t^i$, many other ways have been proposed to compute $\boldsymbol{x}_{t+1}^i$ in velocity-free PSOs. For example, in [14,15], it was proposed a way of instantiating various velocity-free variants from a generalized position update rule given by

$$\boldsymbol{x}_{t+1}^{i} = \boldsymbol{x}_t^i + \epsilon(\boldsymbol{y} - \boldsymbol{x}_t^i) \tag{8}$$

where ϵ is an acceleration coefficient and $\boldsymbol{y}$ is a vector which is obtained combining information from other particles, for example:

$$\boldsymbol{y} = \frac{u_1 \boldsymbol{p}_t^1 + u_2 \boldsymbol{p}_t^2}{u_1 + u_2} \tag{9}$$

where u_1 and u_2 are random values drawn from $\mathcal{U}[0,1]$, and $\boldsymbol{p}_t^1$ and $\boldsymbol{p}_t^2$ are the personal best positions of two neighbor particles of i chosen according to some criterion. Velocity-free PSO variants are relatively less common in the literature of PSO, and they vary in all kind of aspects, including the way in which the current position of a particle is taken into account in the computation of $\boldsymbol{x}_{t+1}^i$.

3 The Grey Wolf, Firefly, and Bat Algorithms—Explained

The three algorithms that we analyze in this paper—grey wolf, firefly, and bat algorithms—are taken from the evolutionary computation bestiary [4]. We chose them not only because they were amenable to the analysis that we present in this paper, but also because they were highly cited[1], which gives a reasonable indication of the impact these algorithms have had on the research community.

In the reminder of this section, we present a detailed description of the grey wolf, firefly and bat algorithms using terminology and concepts belonging to PSO. The idea is to recreate together with the reader these three algorithms from concepts that he/she is already familiar with, and to give the details of the metaphor employed by the authors only after the algorithm has been described

[1] Grey Wolf Optimizer [13]: 3656 citations; Firefly Algorithm [27]: 3018 citations; and Bat Algorithm [28]: 3549 citations. Source: Google Scholar. Retrieved: July 10, 2020.

in plain algorithmic terms. By doing so, it will be easier to understand if the metaphor was necessary (or useful) to understand the proposed algorithm and if the introduced concepts were indeed new or just hidden by the unconventional terminology used.

3.1 Grey Wolf Optimizer (GWO)

GWO as a PSO algorithm. The grey wolf optimizer (GWO) [13] is an algorithm in which, in PSO terms, the three best particles in the swarm are used to bias the movement of the remaining swarm particles. This idea is implemented in GWO by defining three vectors $\boldsymbol{s}_t^k$ as follows:

$$\begin{aligned} \boldsymbol{s}_t^1 &= \boldsymbol{g}_t^1 - (2\,\varphi_t - 1)\boldsymbol{r}_t^1\,\left|2\,\boldsymbol{q}_t^1 \odot \boldsymbol{g}_t^1 - \boldsymbol{x}_t^i\right| \\ \boldsymbol{s}_t^2 &= \boldsymbol{g}_t^2 - (2\,\varphi_t - 1)\boldsymbol{r}_t^2\,\left|2\,\boldsymbol{q}_t^2 \odot \boldsymbol{g}_t^2 - \boldsymbol{x}_t^i\right| \\ \boldsymbol{s}_t^3 &= \boldsymbol{g}_t^3 - (2\,\varphi_t - 1)\boldsymbol{r}_t^3\,\left|2\,\boldsymbol{q}_t^3 \odot \boldsymbol{g}_t^3 - \boldsymbol{x}_t^i\right| \end{aligned} \tag{10}$$

where $\boldsymbol{g}_t^1$, $\boldsymbol{g}_t^2$ and $\boldsymbol{g}_t^3$ indicate the position of the three best particles in the swarm at iteration t, $\boldsymbol{r}_t^k, \boldsymbol{q}_t^k$ ($k = 1, 2, 3$) are random vectors with values drawn from $\mathcal{U}[0, 1]$ that will induce perturbation to the components of $\boldsymbol{s}_t^k$, and φ_t is a decreasing acceleration coefficient that goes from 2 to 0.

The position update rule combining the information of the three best particles $\boldsymbol{s}_t^k$ is defined as follows:

$$\boldsymbol{x}_{t+1}^i = (\boldsymbol{s}_t^1 + \boldsymbol{s}_t^2 + \boldsymbol{s}_t^3)/3 \tag{11}$$

How does GWO compare to PSO? The values $\boldsymbol{s}_t^k$ in Eq. 10 are defined in a very similar way to $\boldsymbol{P}_t^i$ and $\boldsymbol{L}_t^i$ of Eq. 5. The main difference is that instead of defining the vectors in terms of $\boldsymbol{x}_t^i$ (as in SPSO-2011), in GWO they are defined in terms of the three values $\boldsymbol{g}_t^k$ (see Eq. 10). The kind of perturbation induced by $\boldsymbol{q}_t^k$ in Eq. 10 is equivalent to the one induced by vectors $\boldsymbol{a}_t^i$ or $\boldsymbol{b}_t^i$ in PSO (see Eq. 2); however, the one induced by $\boldsymbol{r}_t^k$ is different because the entries of $\boldsymbol{r}_t^k$ are multiplied by $(2\,\varphi_t - 1)$ producing both positive and negative values. Computing the Euclidean norm of $(2\,\boldsymbol{q}_t^k \odot \boldsymbol{g}_t^k - \boldsymbol{x}_t^i)$ to generate new random points in a radius $|2\,\boldsymbol{q}_t^k \odot \boldsymbol{g}_t^k - \boldsymbol{x}_t^i|$ is the same idea as proposed in SPSO-2011 to generate a random point around the hypersphere center $\boldsymbol{c}_t^i$ (see Eq. 3). To compute a φ_t that linearly decreases from 2 to 0, GWO uses the same mechanism proposed in the "self-organizing hierarchical PSO with time-varying acceleration coefficients" [18] for computing φ_1, with the only difference that the lower bound in GWO is set to 0 instead of 0.5 as done in [18].

The position update rule introduced in Eq. 11 is an extension of the recombination rule in velocity-free PSOs (Eqs. 8 and 9) which uses the three best particles in the swarm. Similarly to how it was done in bare-bones PSO (Eq. 7), where the authors employed the recombination operator shown in Eq. 9 assuming $u_1 = u_2$ for computing the center of the normal distribution, the authors of GWO employed the same recombination operator (also assuming $u_1 = u_2 = u_3$) for computing the particles' position.

The metaphor of grey wolves hunting. The authors of GWO say in their original paper published in 2014 [13] that they were inspired by the way in which grey wolves organize their hunting following a strict social hierarchy in which they divide—from top to bottom—their pack: α, β, δ and ω wolves. 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* towards the prey. However, GWO does not consider 5 of the 7 steps mentioned, and seems to take inspiration only from two steps respectively in phase (i) and (iii): *encircling* and *attacking.*

In GWO, a solution to the problem being tackled is called a "wolf", the optimum of the problem is referred to as the "prey" that the wolves are hunting, and the three best solutions and the remaining particles are named as $\boldsymbol{\alpha}_t$, $\boldsymbol{\beta}_t$, $\boldsymbol{\delta}_t$ and $\boldsymbol{\omega}_t$ respectively, in analogy to the levels in the wolves social hierarchy. The GWO algorithm then consists in the "wolves encircling and attacking the prey".

To model encircling the authors used Eq. 11 while attacking the prey was modeled by linearly decreasing the value of φ from 2 to 0 in Eq. 10. In fact, in the imaginary of the metaphor, when φ_t is lower than 1, wolves can concentrate around the prey (therefore attacking it); and when it is greater than 1, they search for other preys.[2] The authors mentioned that the use of $\boldsymbol{q}_t^k$, as done in Eq. 10, emphasizes the *search* behavior of wolves in a similar way in which $\varphi_t > 1$ does it, although, in their view, $\boldsymbol{q}_t^k$ represents "the effect of obstacles when wolves approach a prey in nature."

Unfortunately, as it should be clear to the reader by now, the wolf hunting metaphor is neither necessary nor useful to the definition and understanding of the way GWO works. In fact, it is not at all clear what is the optimization process in the wolf hunting that is translated in effective choices in the design of the optimization algorithm. While there is not a PSO variant that exactly matches GWO, as we have shown above, all the concepts introduced in GWO are related to existing concepts already proposed in the PSO literature and the only contribution given by the use of novel terms such as "wolf", "prey", "attacking", and so on is to create confusion and to hinder understanding.

3.2 Firefly Algorithm (FA)

FA as a PSO algorithm. The firefly algorithm (FA) [27] is, in PSO terminology, an algorithm in which the swarm of particles is fully-connected and the particles movement is influenced only by those other particles in the swarm that have a higher quality. This means that the movement of the best particle is not influenced by any other particle. In FA, at each iteration particles are sorted according to their quality; the particle position update is then applied iteratively starting with worst quality particle and ending with the best quality particle.

[2] Although search is not an activity in the hunting phases of wolves, the authors explain it as "the divergence among wolves during hunting in order to find a *fitter prey*" [13, p. 50].

When particle i updates its position, it has to determine the set $W_t^i \subseteq T_t^i$ (where T_t^i is the set of particles in the neighborhood of i) that contains the $|W_t^i|$ particles with quality higher than its own. Updating a particle's position for the next iteration $t+1$ requires $|W_t^i|$ movements of the particle (one for each particle in W_t^i), where the position of the particle obtained in movement $s-1$ (indicated by $\boldsymbol{m}_{t,s-1}^i$) is the starting position for the next one ($\boldsymbol{m}_{t,s}^i$). The initial position of the particle is set to $\boldsymbol{m}_{t,s=0}^i = \boldsymbol{x}_t^i$, $\forall\, i\ \forall\, t$.

The position update rule of FA is given by the following two equations:

$$\boldsymbol{x}_{t+1}^i = \boldsymbol{m}_{t,s=|W_t^i|}^i \tag{12}$$

$$\boldsymbol{m}_{t,s}^i = \boldsymbol{m}_{t,s-1}^i + \varphi_t^{\boldsymbol{w}_{t,s}^i,\boldsymbol{m}_{t,s-1}^i}\big(\boldsymbol{w}_{t,s}^i - \boldsymbol{m}_{t,s-1}^i\big) + \xi \boldsymbol{r}_{t,s}^i \tag{13}$$

where $\boldsymbol{w}_{t,s}^i$ is an element of the ordered set W_t^i, $\varphi_t^{\boldsymbol{w}_{t,s}^i,\boldsymbol{m}_{t,s-1}^i}$ is an acceleration coefficient[3] whose value depends on the Euclidean distance between the two intermediate points $\boldsymbol{w}_{t,s}^i$ and $\boldsymbol{m}_{t,s-1}^i$, and $\boldsymbol{r}_{t,s}^i$ is a vector whose components are random numbers drawn from the uniform distribution $\mathcal{U}[0,1]$ multiplied by a real scalar ξ.

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

$$\varphi^{\boldsymbol{w},\boldsymbol{m}} = \alpha \cdot e^{-\gamma|\boldsymbol{w}-\boldsymbol{m}|^2} \tag{14}$$

where $|\boldsymbol{w}-\boldsymbol{m}|$ is the Euclidean distance between the position of two particles $\boldsymbol{w}$ and $\boldsymbol{m}$, γ is a parameter that allows to control the weight given to $|\boldsymbol{w}-\boldsymbol{m}|^2$, and α a parameter that controls the weight of the exponential function. Because of the way $\varphi^{\boldsymbol{w},\boldsymbol{m}}$ is computed, solutions have larger displacements when they are located close to each other and smaller ones when they are far away.

How does FA compare to PSO? To better understand how FA is a combination of known PSO concepts, we consider the case in which $|W_t^i| = 1$. In this case, particle i updates its position performing only one movement. This allows us to rewrite Eqs. 12 and 13 as follows:

$$\boldsymbol{x}_{t+1}^i = \boldsymbol{x}_t^i + \varphi_t^{i,\boldsymbol{w}_t^i}\big(\boldsymbol{w}_t^i - \boldsymbol{x}_t^i\big) + \xi \boldsymbol{r}_t^i \tag{15}$$

Eq. 15 can be obtained from Eq. 8 by setting $\epsilon = 1$, $\boldsymbol{y} = \boldsymbol{w}_t^i$ and by adding $\xi \boldsymbol{r}_t^i$ at the end of the equation. While setting the value of ϵ and adding $\xi \boldsymbol{r}_t^i$ are typical design choices for velocity-free PSO variants, using the current position $\boldsymbol{w}_t^i$ of a neighbor instead of the neighbor's personal best position is not a common design choice for an implementation using Eqs. 8 and 9. In practice, using the neighbor's current position may increase the diversity of the solutions in the algorithm since a particle's position changes more often than its personal best position, which is updated only when a new better quality position is found. The last term $\xi \boldsymbol{r}_t^i$ in Eq. 15—a random perturbation—is used to increase the exploration of the algorithm and also allows the global best solution to move from its initial position in the search space.

[3] Note that in the following we will use the shorter notation $\varphi_t^{\boldsymbol{w},\boldsymbol{m}}$ when the meaning is clear from the context.

The metaphor of fireflies flashing. The author of the FA algorithm, first published in 2009 [27], says he was inspired by the flashing behavior of fireflies. Because of the metaphor used, he introduced the following terms: "fireflies" to indicate solutions of the considered problem, and "brightness" to indicate a function that computes the value of the acceleration coefficient $\varphi^{\boldsymbol{w},\boldsymbol{m}}$. The acceleration coefficient $\varphi^{\boldsymbol{w},\boldsymbol{m}}$ weighs the distance between two solutions according to their positions in the search space—in the context of the fireflies flashing metaphor, this is meant to model the fact that fireflies are attracted to other "brighter" fireflies.

Most of the metaphor of fireflies flashing is explained in terms of the different behaviors that can be obtained varying the value of parameter γ, for which the author considered two limit cases: $\gamma \to 0$ and $\gamma \to \infty$. When $\gamma \to 0$, the value of $\varphi^{\boldsymbol{w},\boldsymbol{m}} \to 1$ and the attraction among fireflies becomes constant regardless of their distance in the search space. In the metaphor of fireflies flashing, this is the case when "the light intensity does not decay in an idealized sky" and "fireflies can be seen anywhere in the domain" [27, p. 174].

For the other limiting case, when $\gamma \to \infty$, the value of $\varphi^{\boldsymbol{w},\boldsymbol{m}} \to 0$ (making the attractiveness among fireflies negligible) and new solutions can only be created by means of the random vector $\xi \boldsymbol{r}^i_{t,s}$ (see Eq. 13). According to the metaphor, 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) "they feel almost zero attraction to other fireflies."

As can be seen from the explanations given for the use of the metaphor, its usefulness in describing and understanding the proposed algorithm is very doubtful. The only contribution of the metaphor of fireflies flashing seems to be the idea of using an exponential function based on the distance between two particle to compute the value of φ. However, this ideas was also explored before in the context of PSO in a variant called extrapolation PSO (ePSO) [1], published around 2 years before FA, in late 2007.

In ePSO, a particle i experiences a stronger attraction toward $\boldsymbol{g}_t$ when $f(\boldsymbol{g}_t) \ll f(\boldsymbol{x}^i_t)$, where $f(\cdot)$ is the objective function of a minimization problem, and a weak attraction when $f(\boldsymbol{g}_t) \approxeq f(\boldsymbol{x}^i_t)$. Note that, although it is the same idea, it is applied with opposite goals in the two algorithms, that is, in ePSO particles are more attracted towards particles that are far away while in FA they are attracted more to particles that are closer. Also, the distance is defined differently, since ePSO uses the distance with regard to the function evaluation and FA uses the Euclidean distance.

3.3 Bat Algorithm (BA)

BA as a hybrid PSO and simulated annealing algorithm. The bat algorithm (BA) [28] is an algorithm in which (i) particles in the swarm move by identifying good search directions exploiting the location of the global best particle, and (ii) there is the occasional introduction of new random solutions around the global best solution that are accepted using a simulated annealing like criterion. Using PSO terminology, the BA algorithm can be explained as follows.

Each particle employs two parameters: the probability ρ^i_t—increasing over time—of randomly generating a solution around $\boldsymbol{g}_t$, and the probability ζ^i_t—decreasing over time—to accept the new solution generated. At each iteration t and with probability ρ^i_t, a particle generates a random point around $\boldsymbol{g}_t$ and keeps it in a variable $\boldsymbol{z}^i_t$, which will be accepted as the new position of the particle if two conditions are verified: (i) the quality of $\boldsymbol{z}^i_t$ must be higher than that of $\boldsymbol{g}_t$, that is, $f(\boldsymbol{z}^i_t) < f(\boldsymbol{g}_t)$, where $f(\cdot)$ is the objective function;[4] (ii) $\boldsymbol{z}^i_t$ is accepted with probability ζ^i_t. Therefore, for $\boldsymbol{z}^i_t$ to be accepted the following variable `Accept` must be true: $\texttt{Accept} = ((f(\boldsymbol{z}^i_t) < f(\boldsymbol{g}_t)) \wedge (\mathcal{U}[0,1] < \zeta^i_t))$.

If the random particle around $\boldsymbol{g}_t$ is not generated (this happens with probability $(1 - \rho^i_t)$) or when `Accept` is false[5] (i.e., $\boldsymbol{z}^i_t$ was rejected), particles update their position by adding a velocity vector to their current position.

The process described above is mathematically modeled as follows:

$$\boldsymbol{x}^i_{t+1} = \begin{cases} \boldsymbol{g}_t + \hat{\zeta}_t \, \boldsymbol{r}^i_t, & \text{if } \texttt{Generate} \wedge \texttt{Accept} \\ \boldsymbol{x}^i_t + \boldsymbol{v}^i_{t+1} & \text{if } (\texttt{Generate} \wedge (\neg\texttt{Accept})) \vee (\neg\texttt{Generate}) \end{cases} \tag{16}$$

where $\hat{\zeta}_t$ is the average of the parameters ζ^i_t of all the particles in the swarm, $\boldsymbol{r}^i_t$ is a vector with values randomly distributed in $\mathcal{U}[-1,1]$, and `Generate` is a logical variable which is true when the algorithm decides, with probability ρ^i_t, to create a random solution around $\boldsymbol{g}_t$.

The equations to update the probabilities ρ^i_t and ζ^i_t are:

$$\begin{aligned} \rho^i_{t+1} &= \rho_{t=0}(1 - e^{-\beta_1 t'}) \\ \zeta^i_{t+1} &= \begin{cases} \beta_2 \, \zeta^i_t & \text{if } \texttt{Generate} \wedge \texttt{Accept} \\ \zeta^i_t & \text{otherwise} \end{cases} \end{aligned} \tag{17}$$

where $\beta_1 > 0$ and $0 < \beta_2 < 1$ are parameters, t' is an iteration counter that is updated every time $\texttt{Generate} \wedge \texttt{Accept} = \texttt{TRUE}$ in Eq. 16, and $\rho_{t=0}$ is the initial value of parameter ρ. As it is clear from Eq. 17, the value of ρ^i_t tends to $\rho_{t=0}$ and the value of ζ^i_t tends to 0. Also, note that since the value ζ^i_t decreases with the number of iterations, so does $\hat{\zeta}_t$; this means that for increasing t values the solutions generated in the first case of Eq. 16 will be closer and closer to $\boldsymbol{g}_t$.

As mentioned above and indicated in Eq. 16, when $(\texttt{Generate} \wedge (\neg\texttt{Accept})) \vee (\neg\texttt{Generate})$ the particles update their positions by adding a velocity vector to their current position, which is defined as follows:

$$\boldsymbol{v}^i_{t+1} = \boldsymbol{v}^i_t + \boldsymbol{d}^i_t \odot (\boldsymbol{g}_t - \boldsymbol{x}^i_t) \tag{18}$$

[4] In this paper, we consider minimization problems; the obvious adaptation should be made in case of maximization problems.

[5] Due to the constraint that both conditions have to be met, it may be the case that $\boldsymbol{z}^i_t$ is rejected even when its quality is higher than that of $\boldsymbol{g}_t$.

where, except for $\boldsymbol{d}_t^i$, the components of Eq. 18 are the same as those of SPSO (see Eq. 2). The vector $\boldsymbol{d}_t^i$ is computed as follows:

$$\boldsymbol{d}_t^i = \varphi_1 + \boldsymbol{a}_t^i(\varphi_2 - \varphi_1) \tag{19}$$

where φ_1 and φ_2 are parameters and $\boldsymbol{a}_t^i$ is the same random vector as in Eq. 2.

How does BA compare to PSO and simulated annealing?[6] The velocity update rule of BA—Eq. 18—is a special case of SPSO—Eq. 2. It can be easily seen that, if we set $\omega = 1$ and $\varphi_1 = 0$ in the velocity update rule of SPSO, it simplifies to the BA velocity update rule in Eq. 18. The only difference is that, in BA, the magnitude of the random vector $\boldsymbol{a}_t^i$ depends on the value of parameters φ_1 and φ_2.

The parameter ζ_t^i is very similar to the temperature acceptance criterion T first introduced in simulated annealing (SA) [10]. Two minor differences are that (i) in BA, the value of ζ_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; and (ii) that BA only accepts solutions with better quality than that of the global best solution, while SA can accept both improving and worsening solutions.

The metaphor of bats echolocation. BA introduces a rather technical terminology, in which "bats" are the solutions to the considered problem, the range of "frequencies" in which bats emit their sound are φ_1 and φ_2 (defined in Eq. 19), the "loudness" of bats' sound is the acceptance criterion (ζ_t^i), and the "pulse emission rate" of their sound is the probability ρ_t^i of starting the process in which new solutions are generated around $\boldsymbol{g}_t$.

The author of BA says he was inspired by the echolocation that some bat species use to find their way in the dark by producing sound waves that echo when they are reflected off an object. In order to develop the bat algorithm, the author strongly simplified several aspects of this process. In the words of the author, it was assumed that: (i) "bats are able to differentiate in some magical way between food/prey and background barriers"; (ii) "bats can automatically adjust the frequency and rate in which they are emitting sound"; and (iii) "the loudness of their sound can only decrease from a large value to almost 0."

The author imagined that bats have two different flying modes, which correspond to the two cases in Eq. 16. In the first flying mode, bats fly randomly adjusting their "pulse emission rate" ρ_t^i and "loudness" ζ_t^i. According to the metaphor, bats decrease the pulse emission rate and produce louder sounds when they are randomly searching for a prey; and vice-versa when they have found one. Bats adjusting their pulse emission rate and loudness was modeled using Eq. 17. In the second flying mode, modeled by Eqs. 18 and 19, bats control their

[6] Note that, although in this paper we compared BA with PSO and SA, BA could also be interpreted as a variant of differential evolution (DE) [24]. This is because the probability $\boldsymbol{\rho}_t^i$ and the `Accept` criterion in BA are used in the same way as the mutation probability and the acceptance between donor and trial vectors in DE [17].

step size and range of movement by adjusting their sound frequency (vector $\boldsymbol{d}_t^i$ in Eq. 19) and by moving towards the best bat in the swarm.

As it should be clear to the reader at this point, the metaphor of bats echolocation seems to be an odd and confusing way of explaining the algorithm. This is because there are so many simplifications and unrealistic assumptions in the way in which the metaphor was translated into algorithmic terms that metaphor and algorithm seem to be two completely different things. In fact, except for generalities, the metaphor described by the author in his article does not provide a convincing basis for the choices made in the design of the resulting algorithm.

4 Conclusions

In this article, we have rewritten three highly-cited, metaphor-based algorithms in terms of PSO. We have shown that, perhaps contrary to the goal of the authors, the metaphors of *grey wolves hunting*, *fireflies flashing*, and *bats echolocation* do not facilitate understanding the corresponding GWO, FA and BA algorithms; rather, they create confusion because they hide their strong similarities with existing PSO algorithms. Even though with the help of imagination it is possible to vaguely understand how some of the ideas coming from the metaphor were used to match the corresponding algorithms, it is hard to see how such metaphors are useful at all.

After reviewing the GWO, FA and BA algorithms, we found that none of them propose truly novel ideas. In fact, they can all be seen as variants of existing PSO algorithms. Therefore, we conclude that these three algorithms are unnecessary since they do not add anything new to the tools that can be used to tackle optimization problems. In future work, we intend to experimentally compare GWO, FA and BA with other PSO variants and to analyze the impact that the specific design choices used in these algorithms have on their performance.

The problem of well-known concepts being reintroduced using new terminology has been spreading in the literature for over 15 years and is currently one of the main criticisms of metaphor-based algorithms. Rigorous analyses [25,11,16,23,3] have shown that a number of these algorithms are equivalent, or differ minimally, from well-known methods. Yet, instead of being proposed as variants of existing algorithms, they are often introduced as completely novel approaches—just as it was the case for the three algorithms studied in this paper.

Acknowledgments. Christian Leonardo Camacho Villalón, Thomas Stützle and Marco Dorigo acknowledge support from the Belgian F.R.S.-FNRS, of which they are, respectively, research fellow and research directors.

References

1. Arumugam, M.S., Murthy, G.R., Rao, M., Loo, C.X.: A novel effective particle swarm optimization like algorithm via extrapolation technique. In: 2007 International Conference on Intelligent and Advanced Systems. pp. 516–521. IEEE (2007)
2. Camacho-Villalón, C.L., Dorigo, M., Stützle, T.: Why the intelligent water drops cannot be considered as a novel algorithm. In: Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Reina, A., Trianni, V. (eds.) Swarm Intelligence, 11th International Conference, ANTS 2018. Lecture Notes in Computer Science, vol. 11172, pp. 302–314. Springer (2018)
3. Camacho-Villalón, C.L., Dorigo, M., Stützle, T.: The intelligent water drops algorithm: why it cannot be considered a novel algorithm. Swarm Intelligence **13**(3–4), 173–192 (2019). https://doi.org/10.1007/s11721-019-00165-y
4. Campelo, F.: Evolutionary computation bestiary. `https://github.com/fcampelo/EC-Bestiary` (2021), version visited last on 26 March 2021
5. Clerc, M.: Standard particle swarm optimisation from 2006 to 2011. open archive HAL hal-00764996, HAL (2011)
6. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation **6**(1), 58–73 (2002)
7. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. pp. 39–43 (1995)
8. Kennedy, J.: Bare bones particle swarms. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706). pp. 80–87. IEEE (2003)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks. vol. 4, pp. 1942–1948. IEEE (1995)
10. Kirkpatrick, S.: Optimization by simulated annealing: Quantitative studies. Journal of Statistical Physics **34**(5-6), 975–986 (1984)
11. Melvin, G., Dodd, T.J., Groß, R.: Why 'GSA: a gravitational search algorithm' is not genuinely based on the law of gravity. Natural Computing **11**(4), 719–720 (2012)
12. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: simpler, maybe better. IEEE Transactions on Evolutionary Computation **8**(3), 204–210 (2004)
13. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Advances in Engineering Software **69**, 46–61 (2014)
14. Peña, J.: Simple dynamic particle swarms without velocity. In: Dorigo, M., et al. (eds.) Ant Colony Optimization and Swarm Intelligence, 6th International Conference, ANTS 2008. Lecture Notes in Computer Science, vol. 5217, pp. 144–154. Springer, Heidelberg, Germany (2008)
15. Peña, J.: Theoretical and empirical study of particle swarms with additive stochasticity and different recombination operators. In: Ryan, C. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2008. pp. 95–102. ACM Press, New York, NY (2008)
16. Piotrowski, A.P., Napiorkowski, J.J., Rowinski, P.M.: How novel is the "novel" black hole optimization approach? Information Sciences **267**, 191–200 (2014)
17. Price, K., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer, New York, NY (2005)

18. Ratnaweera, A., Halgamuge, S.K., Watson, H.C.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation **8**(3), 240–255 (2004)
19. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart, Germany (1973)
20. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Simpson, P.K., Haines, K., Zurada, J., Fogel, D. (eds.) Proceedings of the 1998 IEEE International Conference on Evolutionary Computation (ICEC'98). pp. 69–73. IEEE Press, Piscataway, NJ (1998)
21. Shi, Y., Eberhart, R.: Empirical study of particle swarm optimization. In: Proceedings of the 2009 Congress on Evolutionary Computation (CEC 2009). pp. 1945–1950. IEEE Press, Piscataway, NJ (2009)
22. Sörensen, K.: Metaheuristics—the metaphor exposed. International Transactions in Operational Research **22**(1), 3–18 (2015). https://doi.org/10.1111/itor.12001
23. Sörensen, K., Arnold, F., Palhazi Cuervo, D.: A critical analysis of the "improved clarke and wright savings algorithm". International Transactions in Operational Research **26**(1), 54–63 (2019)
24. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**(4), 341–359 (1997)
25. Weyland, D.: 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 (2010)
26. Weyland, D.: A critical analysis of the harmony search algorithm: How not to solve Sudoku. Operations Research Perspectives **2**, 97–105 (2015)
27. Yang, X.S.: Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms. pp. 169–178. Springer (2009)
28. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Studies in Computational Intelligence, vol. 284, pp. 65–74. Springer, Berlin, Germany (2010)
29. Zambrano-Bigiarin, M., Clerc, M., Rojas, R.: Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. In: Proceedings of the 2013 Congress on Evolutionary Computation (CEC 2013). pp. 2337–2344. IEEE Press, Piscataway, NJ (2013)