Cuckoo Search $\equiv (\mu + \lambda)$–Evolution Strategy

A Rigorous Analysis of an Algorithm That Has Been Misleading the Research Community for More Than 10 Years and Nobody Seems to Have Noticed

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

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2021-006

March 2021
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/2021-006

Revision history:

TR/IRIDIA/2021-006.001	March 2021
TR/IRIDIA/2021-006.002	March 2021

Cuckoo search $\equiv (\mu + \lambda)$–evolution strategy

A rigorous analysis of an algorithm that has been misleading the research community for more than 10 years and nobody seems to have noticed

Christian Leonardo Camacho Villalón ·
Thomas Stützle · **Marco Dorigo**

Abstract It has been more than 10 years since the first version of *cuckoo search* was proposed by Yang and Deb and published in the proceedings of the World Congress on Nature & Biologically Inspired Computing, in 2009. The two main articles on *cuckoo search* have now been cited almost 8 000 times (according to `Google scholar`), there are books and chapters published about this algorithm, and even this special issue is celebrating its first decade of existence. Given the popularity of the algorithm and its widespread use, it is quite surprising that no one has ever noticed that *cuckoo search* is an evolutionary algorithm. In this article, we conduct a rigorous analysis of *cuckoo search* in which we identify the concepts used in the algorithm and provide compelling evidence that these are the exact same concepts as those proposed in the $(\mu + \lambda)$–evolution strategy, a well-known evolutionary algorithm introduced originally in 1981. We analyze the "cuckoos' parasitic behavior" metaphor that inspires the algorithm according to four criteria (usefulness, novelty, dispensability, and sound motivation) that allow to clarify whether the use of the metaphor is justified or not. The result is that *cuckoo search* does not comply with any of these criteria. Surprisingly, we found that the algorithm proposed for *cuckoo search* in the original paper does not match the publicly available implementation of the algorithm, which was provided by the authors to show readers how to correctly implement the algorithm; moreover, neither of them follow precisely the metaphor of the cuckoos.

Keywords Cuckoo search · Evolution strategy · Continuous optimization · Metaheuristic optimization · Metaphor-based algorithm

Christian L. Camacho Villalón[0000−0002−0182−3469] ·
Thomas Stützle[0000−0002−5820−0473] · Marco Dorigo[0000−0002−3971−0507]
E-mail: {ccamacho,stuetzle,mdorigo}@ulb.ac.be
IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

1 Introduction

In computational intelligence, metaphor-based algorithms comprise a large group of methods that have been developed taking inspiration from the behavior of natural or artificial systems. Traditionally, the metaphors used in these kinds of algorithm were mostly optimization processes observed in nature that had been studied by scientists working in the fields from which the metaphors were taken. The goal of the metaphor-based algorithm designers was to devise new ways to solve hard optimization problems by carefully translating a metaphor into efficient design choices in an algorithm. Additionally, it used to be always the case that the motivation to use the metaphor had a sound, scientific ground, allowing to easily identify what new concepts were brought by the metaphor to the field of optimization and to understand why they were useful to solve optimization problems.

Indeed, the list of positive examples of metaphor-based algorithms is long [9,18, 23]. Among the best known and most widely used are *evolutionary algorithms* (EAs) [14,35,20,37,19]—inspired by natural evolution and the survival of the fittest; *simulated annealing* (SA) [27,26,44]—inspired by the changes that particles in some solid bodies experience when they are subject to high temperatures; *ant colony optimization* (ACO) [12,11,13]—inspired by the foraging behavior of some species of ants that using indirect communication are capable of finding shortest paths between their nest and food sources; and *particle swarm optimization* (PSO) [25,39]—inspired by the social interactions and collective behavior of a flock of birds.

In recent years, it became extremely popular to use new metaphors to propose "new" optimization algorithms [41,8,29]. However, recently these algorithms have started to become the object of some rigorous analyses [45,46,40,33,42,6,7]. These analyses have shown that in a number of cases a so-called "new" algorithm is nothing else than an old, well-known method presented using new terminology; in other words, the ideas proposed in the "new" algorithm are the same as those already presented in the past with the only difference being the natural metaphor and the associated terminology used to describe the algorithmic design. This trend has created confusion in the literature of stochastic optimization and hindered our understanding of metaphor-based algorithms because it has become increasingly hard to know what is actually new and what is not. There is also the fact that the majority of these algorithms are characterized by a profound lack of scientific motivation to use new metaphors—which are often justified by the fact that they are "beautiful" or "interesting" to the authors—and of scientific rigor in the testing and comparison of the proposed algorithms with other methods [41,43,16].

In this article, we analyze the popular and highly-cited *cuckoo search* algorithm [50,51] considering four criteria that allow to evaluate if the use of the metaphor of "cuckoos laying eggs in the nests of other birds" is justified or not. These criteria are:

- **usefulness**: does the metaphor bring useful concepts to solve optimization problems?
- **novelty**: were the concepts brought by the metaphor new in the field of stochastic optimization at the time when they were proposed?
- **dispensability**: can the design choices in the algorithm be understood and justified once the metaphor is removed from the algorithm?

– **sound motivation**: is there a sound, scientific motivation to use the metaphor?

In our view, only when the metaphor complies with these four criteria—as, for example, in EAs, SA, ACO, PSO, etc.—we can consider that its usage is justified, and therefore, that the metaphor-based algorithm should be added to the set of useful techniques in stochastic optimization.

According to our analysis, *cuckoo search* does not comply with any of the criteria listed above. Particularly worrying is the fact that the concepts used in *cuckoo search* were originally proposed by the evolutionary computation community 30 years before the first publication of *cuckoo search*. We provide compelling evidence that *cuckoo search* is the same as the $(\mu + \lambda)$–evolutionary strategy [37,4]. Surprisingly, while carrying out our analysis, we found that the algorithmic procedure proposed for *cuckoo search* in [50,51] and the implementation provided by its authors in Matlab in [49] are very different. We analyze one-by-one these differences and show that neither of them follow consistently the description of the metaphor that inspired the algorithm.

The rest of the article is organized as follows. In Section 2, we describe the three components that define cuckoo search: the metaphor (Sect. 2.1), the algorithm (Sect. 2.2), and the implementation (Sect. 2.3). In Section 3, we present one-by-one the differences that exist between the metaphor and the algorithm, and between the algorithm and its implementation. In Section 4, we discuss whether the metaphor complies with the criteria of usefulness, novelty, dispensability, and sound motivation. We conclude the article in Section 5 by summarizing our findings.

2 The *three* (inconsistent) components of cuckoo search

2.1 The **metaphor** of the cuckoo search algorithm

In the first two articles proposing cuckoo search [50,51], which are the ones typically cited to reference the algorithm[1], the authors describe the cuckoo search algorithm using as a metaphor the "parasitic breeding behavior of cuckoos", a behavior that, according to them and to the reference that they cite in their article, some species of cuckoos practice. In the words of the authors:

> *Cuckoos are fascinating birds, not only because of the beautiful sounds they can make, but also because of their aggressive reproduction strategy. Some species such as the ani and guira cuckoos lay their eggs in communal nests, though they may remove others' eggs to increase the hatching probability of their own eggs (Payne et al., 2005). Quite a number of species engage the obligate brood parasitism by laying their eggs in the nests of other host birds (often other species). There are three basic types of brood parasitism: intraspecific brood parasitism, cooperative breeding and nest takeover. Some host birds can engage direct conflict with the intruding cuckoos. If a host bird discovers the eggs are not its own, it will either throw these alien eggs away or simply abandons its nest and builds a new nest elsewhere. Some cuckoo species such as the new world brood-parasitic Tapera have evolved in such a way that female parasitic cuckoos are often very specialised in the mimicry in colour and pattern of the eggs of a few chosen host species (Payne et al., 2005). This reduces the probability of their eggs being abandoned and thus increases their*

[1] [50]: 5553 citations; and [51]:2265 citations. Source: Google Scholar. Retrieved: March 17, 2021.

reproductivity.
[50, p. 210] and [51, pp. 331,332]

The timing of egg-laying of some species is also amazing. Parasitic cuckoos often choose a nest where the host bird just laid its own eggs. In general, the cuckoo eggs hatch slightly earlier than their host eggs. Once the first cuckoo chick is hatched, the first instinct action it will take is to evict the host eggs by blindly propelling the eggs out of the nest, which increases the cuckoo chick's share of food provided by its host bird (Payne et al., 2005). Studies also show that a cuckoo chick can also mimic the call of host chicks to gain access to more feeding opportunity.
[50, p. 210] and [51, p. 332]

2.2 The **proposed** cuckoo search algorithm

To translate the metaphor above into an algorithm, the authors simplified the process into three idealized rules—again, in the words of the authors:

For simplicity in describing our new Cuckoo Search (Yang and Deb 2009), we now use the following three idealized rules:
– Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;
– The best nests with high quality of eggs (solutions) will carry over to the next generations;
– The number of available host nests is fixed, and a host can discover an alien egg with a probability $p_a \in [0,1]$. *In this case, the host bird can either throw the egg away or abandon the nest so as to build a completely new nest in a new location.*
For simplicity, this last assumption can be approximated by a fraction p_a *of the n nests being replaced by new nests (with new random solutions at new locations).*
[51, p. 3]

In addition to these rules, the description of cuckoo search is limited to one equation that is used to generate new solutions as follows:

When generating new solutions $\mathbf{x}^{(t+1)}$ *for, say, a cuckoo i, a Lévy flight is performed.*

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^t + \alpha \otimes \text{Lévy}(\lambda) \tag{1}$$

[50, p. 211] and [51, p. 2]

The authors of cuckoo search refer to Eq. 1 as “Lévy flights” because it makes use of the Lévy distribution to sample random numbers. Note that, because of the use of the metaphor of cuckoos, the authors introduced new terminology, in particular, they used three different words (*eggs*, *nests* and *cuckoos*) to refer to a candidate solution to the problem. However, this terminology is not clear and the authors are not consistent with their use.

If we consider the first rule: “*Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest*” that is modeled using Eq. 1, it is understood that $\mathbf{x}_i^t$ is the position of the *cuckoo*, the term $\alpha \otimes \text{Lévy}(\lambda)$ that is added represents the distance the *cuckoo* flew, and $\mathbf{x}_i^{(t+1)}$ is the nest to which the *cuckoo* arrived and deposited the *egg*. However, in [50, p. 211], the terminology seems to be used differently:

For simplicity, we can use the following simple representations that each egg in a nest represents a solution, and a cuckoo egg represent a new solution, the aim is to use the new and potentially better solutions (cuckoos) to replace a not-so-good solution in the nests.

According to this "representation", what the authors refer to as an *egg* and a *cuckoo* is inverted with regard to their first rule and Eq. 1, that is, in the excerpt above, the *egg* represents the initial solution $\mathbf{x}_i^t$ and the *cuckoo* represents the new and potentially better solution $\mathbf{x}_i^{(t+1)}$.

The most serious problem with the metaphor comes from the second rule, which says that "*The best nests with high quality of eggs (solutions) will carry over to the next generations*". While the metaphor of the *cuckoos' parasitic behavior* describes a process in which cuckoos lay their eggs in the nest of other birds and some of these eggs survive and some others do not (as specified in the third rule), there is no mention of a selection mechanism to get rid of the low quality eggs that were laid in the different nests. However, by including the second rule as part of the rules that define the cuckoo search algorithm and saying that these rules are taken from their metaphor of cuckoos, the authors implied that selection is part of the cuckoos metaphor when it is not.

2.3 The **implemented** cuckoo search algorithm

Cuckoo search is an iterative, population-based algorithm to tackle continuous optimization problems. In a continuous optimization problem, the goal is to minimize a d-dimensional continuous objective function $f : S \subseteq \mathbb{R}^d \to \mathbb{R}$ by finding a vector $\mathbf{o} \in S$ such that $\forall \mathbf{x} \in S, f(\mathbf{o}) \le f(\mathbf{x})$. The search space S is a subset of $\mathbb{R}^d$ in which a solution is represented by a real-valued vector $\mathbf{x}$, and each component x^k of $\mathbf{x}$ is constrained by a lower and upper bound such that $lb^k \le x^k \le ub^k$, for $k = 1,\dots,d$. The vector $\mathbf{o}$ represents the solution for which the objective function $f(\cdot)$ returns the minimum value. For a maximization problem, the obvious adaptation consists in using $-f(\cdot)$ instead of $f(\cdot)$.

According to the publicly available implementation of the algorithm in Matlab [49], in plain computational terms, the cuckoo search algorithm consists of the following four steps.

Step 1 (initialization). Create a set of n initial solutions $\mathbf{x}_t^i$ randomly distributed in the search space using the following equation:

$$x_{t=0}^{i,k} = \mathscr{U}(lb^k, ub^k), \text{ for } i = 1,\dots,n \text{ and } k = 1,\dots,d, \tag{2}$$

where t is the iteration number, $\mathscr{U}$ is a random uniform distribution, lb^k and ub^k are the lower and upper limit of dimension k, and d is the number of dimensions in the problem.

Step 2 (perturbation). Perturb all n solution $\mathbf{x}_t^i$ by adding a random vector $\mathbf{r}_t^i$ as follows:

$$\mathbf{x}_t^{i'} = \mathbf{x}_t^i + \alpha\,\mathbf{r}_t^i, \text{ for } i = 1,\dots,n, \tag{3}$$

where $\mathbf{x}_t^{i'}$ is the perturbed solution, $\mathbf{r}_t^i$ is a random vector whose components are sampled from a Lévy distribution $\mathscr{L}_\lambda$ with scale parameter λ, and α is a parameter that controls the magnitude of the perturbation.

Step 3 (selection). Compare each pair $(\mathbf{x}_t^i, \mathbf{x}_t^{i'})$ on the basis of the objective function $f(\cdot)$ and select the one that has higher quality. This is formally done as follows:

$$\mathbf{x}_{t'}^i = \begin{cases} \mathbf{x}_t^{i'}, & \text{if } f(\mathbf{x}_t^{i'}) \text{ is better than } f(\mathbf{x}_t^i) \\ \mathbf{x}_t^i, & \text{otherwise} \end{cases}. \tag{4}$$

Step 4 (recombination). With probability $1 - p_a$, apply recombination to the k^{th} component of vector $\mathbf{x}_{t'}^i$ using two randomly selected solutions $\mathbf{x}_{t'}^{l^i}$ and $\mathbf{x}_{t'}^{m^i}$ as follows:

$$x_{t+1}^{i,k} = \begin{cases} x_{t'}^{i,k} + \mathscr{U}[0,1] \cdot (x_{t'}^{l^i,k} - x_{t'}^{m^i,k}), & \text{if } \mathscr{U}[0,1] \geq p_a \\ x_{t'}^{i,k}, & \text{otherwise} \end{cases}, \forall k, \forall i, \tag{5}$$

where $\mathbf{x}_{t'}^{l^i}$ and $\mathbf{x}_{t'}^{m^i}$ are two solutions taken from sets L_t and M_t that will be recombined to modify probabilistically the components of $\mathbf{x}_{t'}^i$. Sets L_t and M_t contain each a copy of the population after executing step 3 (selection), i.e., a copy of $\mathbf{x}_{t'}^i$ for $i = 1, \ldots, n$. Since, in every iteration, a solution is used once as $\mathbf{x}_{t'}^l$ and once as $\mathbf{x}_{t'}^m$, solutions are removed from L_t or M_t, according to the case, after they have been used. Note that it can be the case that $\mathbf{x}_t^{l^i} = \mathbf{x}_t^{m^i} = \mathbf{x}_t^i$, in which case vector $\mathbf{x}_t^i$ is not modified. After finishing the process of recombination, solutions are evaluated once again.

The implementation of cuckoo search consists in applying step 1 (random initialization) once and repeating step 2 (perturbation), step 3 (selection) and step 4 (recombination) iteratively until a termination criterion is met.

3 Differences among the metaphor, the algorithm and the implementation of cuckoo search

Based on what we presented in Sect. 2.1, the metaphor of *cuckoos' parasitic reproduction* that inspired the algorithm can be summarized as the strategy that some species of cuckoos practice that consists in laying their eggs in the nest of other birds instead of creating their own nest. It also includes the idea that cuckoos' eggs laid in the nests of other birds are sometimes identified by those other birds, that can either remove the cuckoos' eggs from the nest or abandon the nest and build a new one. As we presented in Sect. 2.2, in [50,51] this metaphor was translated into a set of rules as follows:

i at each iteration, each cuckoo lays one egg in a randomly chosen nest;
ii the number of nests is fixed and each nest can host only one egg;
iii the nests with the better quality at the end of iteration t are used as eggs in the iteration $t+1$;
iv with probability p_a, an egg is removed from the nest and replaced by a new one in a new location.

As we also discussed in Sect. 2.1, the metaphor of the cuckoos does not include any selection mechanism between old eggs and new eggs or old eggs and cuckoos, but the authors presented the algorithm in such a way that they made it look as if it was part of it. According to rules (i) and (iii) and to the cuckoo search algorithm implementation the authors provided in [49], in every iteration, a solution is selected between the solution represented by the *parent cuckoo* and the solution represented by its *egg* depending on their quality. This specific type of selection was originally introduced by the evolutionary computation community and, as we describe in detail in Sect. 4.1, it is the same concept used in the $(\mu + \lambda)$–evolution strategy [37].

In the following, we explain the many differences we found between the cuckoo search algorithm as published in [50,51] (reported in Alg. 1) and what its authors provided as an example of correct implementation in [49]. We do so by comparing Alg. 1 to steps 1–4 that correspond to the implementation of the algorithm in Matlab—see [49] for details.[2]

Algorithm 1 Cuckoo search algorithm as published in [50,51]

```
 1: begin
 2:     Objective function $f(\mathbf{x}), x = (x_1, \ldots, x_d)^T$
 3:     Initial population of $n$ hosts nests $\mathbf{x}_i (i = 1, 2, \ldots, n)$
 4:     while ($t <$ MaxGenerations) or (stop creiterion) do
 5:         Get a cuckoo (say $i$) randomly by Lévy flights
 6:         Evaluate its quality/fitness $F_i$
 7:         Choose a nest among $n$ (say $j$) randomly
 8:         if ($F_i > F_j$) then
 9:             Replace $j$ by the new solution
10:         end if
11:         Abandon a fraction ($p_a$) of the worse nests [and build new ones at new locations via Lévy
    flights]
12:         Keep the best solutions (or nests with quality solutions)
13:         Rank the solutions and find the current best
14:     end while
15:     Postprocess results and visualization
16: end
```

The first difference to note between steps 1–4 and what is depicted in Alg. 1, is that, in Alg. 1, there is not a **for** loop to iterate over all the n solutions in the population. Therefore, differently from step 2 (perturbation), in Alg. 1, the Eq. 3 is applied only to one solution i randomly selected from the population at every iteration (line 5 of Alg. 1).

The second difference has to do with step 3 (selection). In this step, after a solution $\mathbf{x}_t^i$ has been perturbed using Eq. 3, either the perturbed solution $\mathbf{x}_t^{i'}$ or the initial solution $\mathbf{x}_t^i$ is accepted as $\mathbf{x}_{t'}^i$ depending on its quality—see Eq. 4. However, in Alg. 1 this is done differently. In Alg. 1, the condition in the **if** statement (line 8 of Alg. 1) says that **if** $f(\mathbf{x}_t^{i'})$ is better than $f(\mathbf{x}_t^j)$, where $\mathbf{x}_t^j$ is a randomly chosen solution, **then** $\mathbf{x}_t^j$ is

[2] We remind the reader that the code of the algorithm is also publicly available in the *Appendix: Demo Implementation* of [51] in the version published in the arXiv repository: `arXiv:1005.2908`.

replaced by the perturbed solution $\mathbf{x}_t^{i'}$. Clearly, since j is chosen randomly, it may or may not correspond to i.

The last and most important difference we found concerns step 4 (recombination) and the original corresponding algorithm instruction, as indicated in line 11 of Alg. 1. First, according to the rules derived from the metaphor by the authors (see Sect. 2.2), solutions are supposed to be removed randomly, but in line 11 of Alg. 1 this is done deterministically. Second, although the authors do not give precise directions on how to implement line 11 of Alg. 1, from what it is written in this line, it is understood that the solutions are first ranked (otherwise it is not possible to know which ones are the worst) and then, Eq. 3—that is, the equation of the "Lévy flights"—is applied to the worst $(p_a \times n)$ solutions. However, in the Matlab implementation, line 11 of Alg. 1 is implemented using Eq. 5, that recombines two randomly selected solutions and uses them to perturb the solution vector probabilistically and dimension-wise.

Although it is unclear whether the authors really intended cuckoo search as in Alg. 1 or as in the Matlab implementation, in [51, p. 3][3], they mention:

A demo version is attached in the Appendix (this demo is not published in the actual paper, but as a supplement to help readers to implement the cuckoo search correctly).

4 Is the metaphor of cuckoo search justified?

4.1 Usefulness and novelty

To know if the metaphor of cuckoo search has brought useful and novel concepts on how to solve optimization problems, we compare cuckoo search to a particular kind of evolutionary algorithm, called *evolution strategies*. Evolution strategies (ES) [34, 35, 37, 36, 3, 4] are among the oldest and best known evolutionary algorithms to solve continuous optimization problems. In ES, as in the rest of evolutionary algorithms, the idea is to simulate the process of natural evolution in order to evolve one or several solutions by iteratively applying the mechanisms of *parental selection*, *variation*, *evaluation* and *survival selection* [31].

The specific ES algorithm the authors of cuckoo search reintroduced is the so-called $(\mu + \lambda)$–ES [37]. To show that cuckoo search is the same as the $(\mu + \lambda)$–ES, in Alg. 2, we show the algorithm of cuckoo search as it is defined using steps 1–4 and, in Alg. 3, the algorithm of the $(\mu + \lambda)$–ES.

The $(\mu + \lambda)$–ES is an algorithm in which there is a population of μ parents (solutions) that produce λ offspring (new solutions) and the population reduces again to μ parents that pass to the next generation. As we show in Alg. 3, to instantiate a $(\mu + \lambda)$–ES, it is necessary to choose the specific *recombination*, *mutation* and *selection* operators to be used. Among the different ways in which the $(\mu + \lambda)$–ES can be implemented, one possibility is to set $\lambda = \mu$, that is, one parent produce one single offspring at each iteration [4, 2]. *Recombination*, although it is often considered an optional component in the $(\mu + \lambda)$–ES, is part of the ES from the beginning and there

[3] This quote is taken from the version of [51] that is published in the arXiv repository in `arXiv: 1005.2908`.

Algorithm 2 Cuckoo search

```
1:  begin
2:      $t \leftarrow 0$
3:      initialize solutions (cuckoos) using Eq. 2
4:      evaluate cuckoos
5:      while not termination-condition do
6:          $t \leftarrow t+1$
7:          apply perturbation to the cuckoos using Eq. 3
8:          evaluate the new created solutions (eggs)
9:          select a cuckoo or an egg using Eq. 4
10:         recombine the selected solutions using Eq. 5 and use them as cuckoos for the next iteration
11:     end while
12: end
```

Algorithm 3 $(\mu+\lambda)$-evolution strategy

```
1:  begin
2:      $t \leftarrow 0$
3:      initialize population of $\mu$ individuals
4:      evaluate population
5:      while not termination-condition do
6:          $t \leftarrow t+1$
7:          recombine individuals (optional)
8:          apply mutation operator to the $\mu$ parents
9:          evaluate the $\lambda$ newborn offspring
10:         select $\mu$ individuals for survival
11:     end while
12: end
```

are several types of recombination operators proposed, such as discrete, intermediate (which is the one used in cuckoo search), global–discrete, global–intermediate, etc.

In all variants of the $(\mu+\lambda)$–ES [35,3], the *selection* operates over parent-offspring couplings (called cuckoo–egg couplings in cuckoo search), which means that parents will pass from one generation to another until they are superseded by an offspring with better fitness. The *mutation* operator in ES, which is a type of perturbation, is the same shown in Eq. 3, with the only difference that it was originally defined using the Gaussian instead of the Lévy distribution. In the evolution strategies literature, the idea of using other distributions for the *mutation* operator, such as the Cauchy distribution, was introduced later by [24], and the first works proposing specifically to use the Lévy distribution were published a few years later [21,28].

From the description of the $(\mu+\lambda)$–ES and the cuckoo search algorithm, it is clear that the two algorithms are the same, with the minor difference that in cuckoo search *recombination* is applied at the end of the **while** loop (line 11 of Alg. 2), whereas in $(\mu+\lambda)$–ES it is applied at the beginning of the loop, line 7 of Alg. 3. It is easy to see that, from iteration 2, the two algorithms are equivalent, that is, *recombination*, followed by *mutation* and then by *selection*. As cuckoo search does not follow the normal order in which the three main components of evolutionary algorithms are used, solutions have to be evaluated twice at each iteration of the **while** loop, wasting computational time.

4.2 Dispensability and sound motivation

Dispensability refers to the fact that one can remove the metaphor from the algorithm and the design choices in the algorithm still make sense and they can be justified on the basis of concepts that are used in stochastic optimization, such as perturbation, neighborhood, local search, etc. Clearly, it should always be the case that the algorithm is, in fact, following the metaphor, otherwise this criterion is failed because the ideas that are supposed to be used in the algorithm are not. In cuckoo search, as we have shown in Sect. 2.2 and Sect. 3, the metaphor, the algorithm and the implementation are three different things, and therefore, cuckoo search fails the evaluation of this criterion. First of all, the algorithm is described using three rules, but the second rule is not part of the metaphor presented by the authors. Second, removing the metaphor of cuckoos and the terminology specific to it, the cuckoo search algorithm—as it is presented in Alg. 1—consists in (i) perturbing one single solution randomly taken from the population at each iteration; (ii) using the perturbed solution to replace another random selected solution; and (iii) perturbing a fraction of the worst solutions in the population. It is impossible to justify these design choices using the metaphor of the cuckoos or the rules derived from the metaphor by the authors of the algorithm. Third, the implementation of cuckoo search provided by the authors, which is different from the metaphor and the algorithm, is in fact the same as the well-known $(\mu+\lambda)$–ES proposed 30 years before cuckoo search.

Sound motivation is intended to evaluate whether the motivation to use the metaphor has a sound, scientific basis. In this sense, it should always be possible (i) to understand what is the natural or artificial optimization process used by the authors as a source of inspiration, and (ii) to clearly identify the components in this observed optimization process that can be used as effective design choices in the algorithm. It is important to stress that even if nobody has ever thought before to use the metaphor as a source of inspiration, points (i) and (ii) have to be verified in order to consider its usage valid. In the metaphor of cuckoos there is no optimization process, and therefore, it does not comply with point (i); as a consequence, it is not possible to identify the components to be used in the algorithm, and therefore, also point (ii) is not verified.

5 Conclusions

Based on the criteria that we established to analyze *cuckoo search*, we can conclude that neither the metaphor nor the algorithm can be considered as part of the set of useful techniques in stochastic optimization. On the contrary, *cuckoo search* can now be added to the list of metaphor-based algorithms that do not contain any novelty, such as *harmony search* [45] and *black holes optimization* [33]. As we have shown in this article, using the metaphor of "cuckoos' parasitic reproduction", *cuckoo search* reintroduced the exact same concepts as those proposed in 1981 in an algorithm called $(\mu+\lambda)$–ES. It is quite surprising, and indeed worrying, that this fact has been overlooked for more than 10 years. As other cases like *cuckoo search* are being recognized more and more by scientific journals, some of them have already established editorial policies where this issue is acknowledged [22, 10, 1] and steps are being taken to avoid the publication of papers presenting this type of algorithms (see, e.g., [45, 46, 40, 30, 33, 15, 42, 29, 6, 7]).

Since a few years, we have taken the enterprise to show that many of the novel metaphor-based metaheuristics that are being proposed in the literature are just a reiteration of already well-known algorithms and that the only novelty is in the terminology being used to describe the algorithms. We have done so with the *intelligent water drops* [38] and with the *grey wolf* [32], *firefly* [47] and *bat* [48] algorithms, which were shown to be special cases of ant colony optimization and of particle swarm optimization, respectively. However, it is virtually impossible to discuss in detail (as done in this paper and in [5, 6, 7]) all the new proposed algorithms; they are simply too many and new ones are being published with alarming regularity (see, e.g., [8, 29]). We hope that our work can help the research community to understand that the approach based on (i) finding an "interesting" natural or artificial behavior, (ii) extracting the terminology from the behavior, and (iii) using the new, non-standard terminology to define an optimization algorithm whose components are based on well-known concepts and differ from what we already know only in the terminology, should not be pursued as it is a waste of time and resources.

Declarations

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

Conflict of interest: The authors declare that there is no conflict of interest.

Availability of data and material: Not applicable

Code availability: Not applicable

References

1. ACM Transactions on Evolutionary Learning and Optimization. Guidelines for Authors. `https://dl.acm.org/journal/telo/author-guidelines` (2021). Version visited last on March 26, 21
2. Bäck, T., Fogel, D.B., Michalewicz, Z.: Handbook of evolutionary computation. IOP Publishing (1997)
3. Bäck, T., Hoffmeister, F., Schwefel, H.P.: A survey of evolution strategies. In: Proceedings of the fourth international conference on genetic algorithms, pp. 2–9. Morgan Kaufmann (1991)
4. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimization. Evolutionary computation **1**(1), 1–23 (1993)
5. Camacho-Villalón, C.L., Dorigo, M., Stützle, T.: 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, *Lecture Notes in Computer Science*, vol. 11172, pp. 302–314. Springer (2018)
6. 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). DOI 10.1007/s11721-019-00165-y

7. Camacho-Villalón, C.L., Stützle, T., Dorigo, M.: Grey wolf, firefly and bat algorithms: Three widespread algorithms that do not contain any novelty. In: International Conference on Swarm Intelligence, pp. 121–133. Springer (2020)
8. Campelo, F.: Evolutionary computation bestiary. `https://github.com/fcampelo/EC-Bestiary` (2021). Version visited last on 26 March 2021
9. Corne, D., Dorigo, M., Glover, F., Dasgupta, D., Moscato, P., Poli, R., Price, K.V.: New ideas in optimization. McGraw Hill (1999)
10. Dorigo, M.: Swarm intelligence: A few things you need to know if you want to publish in this journal. `https://www.springer.com/cda/content/document/cda_downloaddocument/Additional_submission_instructions.pdf` (2016). Version visited last on March 26, 2021
11. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: An autocatalytic optimizing process. Tech. Rep. 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Italy (1991)
12. Dorigo, M., Maniezzo, V., Colorni, A.: Positive feedback as a search strategy. Tech. Rep. 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy (1991)
13. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA (2004)
14. Fogel, D.B., Owens, A.J., Walsh, M.J.: Artificial Intelligence Through Simulated Evolution. John Wiley & Sons (1966)
15. Fong, S., Wang, X., Xu, Q., Wong, R., Fiaidhi, J., Mohammed, S.: Recent advances in metaheuristic algorithms: Does the makara dragon exist? The Journal of Supercomputing **72**(10), 3764–3786 (2016)
16. García-Martínez, C., Gutiérrez, P.D., Molina, D., Lozano, M., Herrera, F.: Since CEC 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness. Soft Computing **21**(19), 5573–5583 (2017)
17. Gass, S.I., Fu, M.C. (eds.): Encyclopedia of Operations Research and Management Science. Springer Verlag (2010)
18. Gendreau, M., Potvin, J.Y. (eds.): Handbook of Metaheuristics, *International Series in Operations Research & Management Science*, vol. 146, 2 edn. Springer, New York, NY (2010)
19. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Boston, MA, USA (1989)
20. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
21. Iwamatsu, M.: Generalized evolutionary programming with levy-type mutation. Computer Physics Communications **147**(1-2), 729–732 (2002)
22. Journal of Heuristics. Policies on Heuristic Search Research. `https://www.springer.com/journal/10732/updates/17199246` (2015). Version visited last on March 26, 2021
23. Kacprzyk, J., Pedrycz, W. (eds.): Springer Handbook of Computational Intelligence. Springer, Berlin, Heidelberg, Germany (2015)
24. Kappler, C.: Are evolutionary algorithms improved by large mutations? In: International Conference on Parallel Problem Solving from Nature, pp. 346–355. Springer (1996)
25. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)
26. Kirkpatrick, S.: Optimization by simulated annealing: Quantitative studies. Journal of Statistical Physics **34**(5-6), 975–986 (1984)
27. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
28. Lee, C.Y., Yao, X.: Evolutionary programming using mutations based on the lévy probability distribution. IEEE Transactions on Evolutionary Computation **8**(1), 1–13 (2004)
29. Lones, M.A.: Mitigating metaphors: A comprehensible guide to recent nature-inspired algorithms. SN Computer Science **1**(1), 1–12 (2020)
30. 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)
31. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms. In: Gass and Fu [17], pp. 517–527
32. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Advances in Engineering Software **69**, 46–61 (2014)
33. Piotrowski, A.P., Napiorkowski, J.J., Rowinski, P.M.: How novel is the "novel" black hole optimization approach? Information Sciences **267**, 191–200 (2014)
34. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Ph.D. thesis, Department of Process Engineering, Technical University of Berlin (1971)
35. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart, Germany (1973)

36. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In: J.J. Grefenstette (ed.) ICGA, pp. 93–100. Lawrence Erlbaum Associates (1985)
37. Schwefel, H.P.: Numerical optimization of computer models. John Wiley & Sons, Inc. (1981)
38. Shah-Hosseini, H.: Problem solving by intelligent water drops. In: Proceedings of the 2007 Congress on Evolutionary Computation (CEC 2007), pp. 3226–3231. IEEE, IEEE Press, Piscataway, NJ (2007)
39. Shi, Y., Eberhart, R.: 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), pp. 69–73. IEEE Press, Piscataway, NJ (1998)
40. Simon, D., Rarick, R., Ergezer, M., Du, D.: Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms. Information Sciences **181**(7), 1224–1248 (2011)
41. Sörensen, K.: Metaheuristics—the metaphor exposed. International Transactions in Operational Research **22**(1), 3–18 (2015). DOI 10.1111/itor.12001
42. 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)
43. Sörensen, K., Glover, F.: Metaheuristics. In: Gass and Fu [17], pp. 960–970
44. Černý, V.: A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications **45**(1), 41–51 (1985)
45. 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)
46. Weyland, D.: A critical analysis of the harmony search algorithm: How not to solve Sudoku. Operations Research Perspectives **2**, 97–105 (2015)
47. Yang, X.S.: Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms, pp. 169–178. Springer (2009)
48. 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)
49. Yang, X.S.: Cuckoo search (cs) algorithm. `https://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm` (2021). MATLAB Central File Exchange. Retrieved March 12, 2021.
50. Yang, X.S., Deb, S.: Cuckoo search via lévy flights. In: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), pp. 210–214 (2009)
51. Yang, X.S., Deb, S.: Engineering optimisation by cuckoo search. International Journal of Mathematical Modelling and Numerical Optimisation **1**(4), 330–343 (2010)