Computers & Operations Research 142 (2022) 105747

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

An analysis of why cuckoo search does not bring any novel ideas to optimization

Christian L. Camacho-Villalón [*], Marco Dorigo, Thomas Stützle

IRIDIA, Université Libre de Bruxelles, Avenue Franklin Roosevelt 50, 1050, Brussels, Belgium

ARTICLE INFO

Keywords:
Evolution strategies
Differential evolution
Continuous optimization
Metaheuristic algorithm

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 8 700 times (according to `Google scholar`), there are books and chapters published about this algorithm, and a special issue has been organized to celebrate its first decade of existence. Given the popularity of the algorithm and its widespread use, it is quite surprising that no one has ever formally investigated its obvious resemblance to evolutionary algorithms. In this article, we conduct a rigorous analysis of *cuckoo search* in which we identify the concepts used in the algorithm and show that these are the exact same concepts proposed in the $(\mu + \lambda)$–evolution strategy, a well-known evolutionary algorithm introduced originally in 1981, and in the classic differential evolution algorithm introduced in 1997. We analyze the "cuckoos' parasitic behavior" metaphor that inspired the algorithm according to three criteria—usefulness, novelty 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 the authors proposed for *cuckoo search* does not match the publicly available implementation they provided; moreover, neither of them really follows the metaphor of the cuckoos that inspired the algorithm.

1. Introduction

In computational intelligence, metaphor-based algorithms comprise a large group of methods that have been developed by taking inspiration from natural and artificial behaviors. Originally, the behaviors used to inspire these algorithms consisted of a relatively small number of optimization processes observed in nature, such as the phenomenon of natural evolution used in *evolutionary algorithms* (EAs) (Fogel et al., 1966; Rechenberg, 1973; Holland, 1975; Schwefel, 1981; Goldberg, 1989), the thermodynamics of some metals used in *simulated annealing* (SA) (Kirkpatrick et al., 1983; Černý, 1985), and the foraging of some ants used in *ant colony optimization* (ACO) (Dorigo et al., 1991, 1996; Dorigo and Stützle, 2004). The goal of algorithm designers back then was to take inspiration from natural optimization processes to devise new ways to solve hard optimization problems; the way to do so consisted in a careful translation of these processes into new algorithm designs. Also, since it used to be always the case that the choice to use these behaviors had a sound motivation, it was easy to identify what new concepts were brought to the field by these behaviors and to understand how they could be useful to solve optimization problems.

In recent years, hundreds of self-proclaimed "novel" metaphor-based optimization algorithms inspired by all kinds of behaviors have been published in the literature (Campelo, 2021) with the unfortunate particularity that, in almost all cases, it is unclear whether they introduce any novelty apart from new metaphors (Sörensen, 2015; Campelo, 2021; Lones, 2020). The relatively small number of rigorous analyses performed on some of the most widespread algorithms of this kind (Weyland, 2010, 2015; Simon et al., 2011; Piotrowski et al., 2014; Sörensen et al., 2019; Camacho-Villalón et al., 2019, 2020) have shown that the ideas proposed in these algorithms are the same as those already proposed in the past with the only difference being the metaphor and the associated terminology used to describe the algorithm. The appearance of hundreds of metaphor-based algorithms presented using a non-standard terminology (all of them claiming to be novel approaches without clearly showing what is novel about them) has hindered our understanding of stochastic optimization algorithms and created confusion in the literature of the field. In addition to this, the majority of these algorithms are also characterized by both the lack of scientific motivation for the use of a new metaphor—which is often justified by the fact that the novel metaphor is considered to be

[*] Corresponding author.
E-mail address: ccamacho@ulb.ac.be (C.L. Camacho-Villalón).

https://doi.org/10.1016/j.cor.2022.105747
Received 23 October 2021; Received in revised form 29 January 2022; Accepted 2 February 2022
Available online 12 February 2022

"beautiful" or "interesting" by the authors—and the lack of scientific rigor in testing and comparing the proposed algorithms with other methods (Sörensen, 2015; Sörensen and Glover, 2013; García-Martínez et al., 2017).

In this article, we analyze the popular and highly-cited *cuckoo search* algorithm (Yang and Deb, 2009, 2010) considering three criteria that allow us 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 the following:

- **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?
- **Sound motivation.** Is there a sound motivation to use the metaphor?

In our view, only when the metaphor complies with these three criteria—as, for example, in the mentioned EAs, SA and ACO algorithms—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 all the concepts used in *cuckoo search* were originally proposed by the evolutionary computation community—most of them 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 (Schwefel, 1981; Bäck and Schwefel, 1993) using the type of recombination proposed for differential evolution (Storn and Price, 1997).

While carrying out our analysis, we also surprisingly found that the algorithmic procedure proposed for *cuckoo search* in Yang and Deb (2009, 2010) and the implementation provided by its authors in Matlab in Yang (2021) are quite different. We analyze one-by-one these differences and show that none of them follows 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 (Section 2.1), the algorithm (Section 2.2), and the implementation (Section 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, 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 (Yang and Deb, 2009, 2010), 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 specialized in the mimicry in color 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 reproductivity.*
>
> [*Yang and Deb (2009, p. 210) and Yang and Deb (2010, 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.*
>
> [*Yang and Deb (2009, p. 210) and Yang and Deb (2010, p. 332)*]

2.2. The **proposed** cuckoo search algorithm

To translate the metaphor described in Section 2.1 into an algorithm, the authors simplified the process into three idealized rules. 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).*
>
> [*Yang and Deb (2010, p. 332)*]

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 $\vec{x}^{(t+1)}$ for, say, a cuckoo i, a Lévy flight is performed.*
>
> $$\vec{x}_i^{(t+1)} = \vec{x}_i^{\,t} + \alpha \otimes \text{Lévy}(\lambda) \quad (1)$$
>
> [*Yang and Deb (2009, p. 211) and Yang and Deb (2010, p. 333)*].

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—see the three idealized rules above. However, this terminology is not clear and the authors are not consistent with its use.

[1] Yang and Deb (2009): 6291 citations; and Yang and Deb (2010): 2494 citations. Source: Google Scholar. Retrieved: January 20, 2022.

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 $\vec{x}_i^t$ is the position of the *cuckoo*, the term $\alpha \otimes \text{Lévy}(\lambda)$ that is added represents the distance the *cuckoo* flew, and $\vec{x}_i^{(t+1)}$ is the nest to which the *cuckoo* arrived and deposited the *egg*. However, in Yang and Deb (2009, 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 to Eq. (1); that is, in the excerpt above, the *egg* represents the initial solution $\vec{x}_i^t$ and the *cuckoo* represents the new and potentially better solution $\vec{x}_i^{(t+1)}$.

The most serious problem with the metaphor, however, 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 cuckoos metaphor, the authors implied that selection is part of the cuckoos metaphor when it is not.

2.3. The **implemented** cuckoo search algorithm

According to the publicly available implementation in Matlab (Yang, 2021), cuckoo search is an iterative, population-based algorithm for the approximate solution of continuous optimization problems. In continuous optimization problems, the goal is to minimize a d-dimensional continuous objective function $f : S \subseteq \mathbb{R}^d \to \mathbb{R}$ by finding a vector $\vec{o} \in S$ such that $\forall\, \vec{x} \in S,\ f(\vec{o}) \leq f(\vec{x})$. The search space S is a subset of $\mathbb{R}^d$ in which a solution is represented by a real-valued vector $\vec{x}$, and each component x^k of $\vec{x}$ is constrained by a lower and upper bound such that $lb^k \leq x^k \leq ub^k$, for $k = 1, \ldots, d$. The vector $\vec{o}$ 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 $f(\cdot)$.

The implemented cuckoo search algorithm (Yang, 2021), which is reported in Alg. 2, consists of the following four steps.

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

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

where t is the iteration number, $\mathcal{U}$ 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 μ solution $\vec{x}_t^i$ by adding a random vector $\vec{r}_t^i$ as follows:

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

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

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

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

Step 4 (recombination). With probability $1 - p_a$, apply recombination to the kth component of vector $\vec{x}_{t'}^i$ using two randomly selected solutions $\vec{x}_{t'}^{l^i} \in L_t$ and $\vec{x}_{t'}^{m^i} \in M_t$, where sets L_t and M_t contain each a copy of the population after executing step 3 (selection), i.e., a copy of $\vec{x}_{t'}^i$ for $i = 1, \ldots, n$. Step 4 (recombination) is computed as follows:

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

Solutions $\vec{x}_{t'}^{l^i}$ and $\vec{x}_{t'}^{m^i}$ are selected from sets L_t or M_t without replacement, that is, each solution is used once as $\vec{x}_{t'}^l$ and once as $\vec{x}_{t'}^m$. Note that it can be the case that $\vec{x}_t^{l^i} = \vec{x}_t^{m^i} = \vec{x}_t^i$, in which case vector $\vec{x}_t^i$ 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 then repeating step 2 (perturbation), step 3 (selection) and step 4 (recombination) iteratively until a termination criterion is met.

3. Differences between the metaphor, the algorithm and the implementation of cuckoo search

There are three important components that should help understanding how cuckoo search works: the metaphor, the algorithm description and its implementation. Unfortunately, we discovered that the concepts brought forward by the metaphor are hardly used in the algorithm and that the algorithmic procedure proposed for *cuckoo search* in Yang and Deb (2009, 2010) and the implementation provided by its authors in Matlab in Yang (2021) are quite different.

Differences between metaphor and algorithm. As we discussed in Section 2.1, the authors of cuckoo search say that they were inspired by the *cuckoos' parasitic reproduction* behavior, that is, by the cuckoos' strategy of laying eggs in the nest of other birds; and by the fact that cuckoos' eggs laid in the nests of other birds are sometimes identified by those other birds, that can either remove them from the nest or abandon them in the nest. They translated this behavior (see Section 2.2) 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 best quality eggs at the end of iteration t will pass to 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.

However, in the cuckoos' parasitic reproduction behavior there is no mechanism that allows the cuckoos to select the best "quality" eggs that survive and therefore it cannot be used to "inspire" rule iii. Indeed, by including this rule as part of the algorithm description, the authors made the cuckoos' parasitic reproduction behavior look as an optimization process, when this is not the case. Also note that the central idea in rule iii is no other than the evolutionary concept of "the survival of the fittest", which was originally introduced to the field of stochastic optimization by the evolutionary computation community and, as we describe in detail in Section 4.1, it is one of several concepts used in cuckoo search that belong to the $(\mu + \lambda)$–evolution strategy (Schwefel, 1981).

Differences between algorithm description and algorithm implementation. In Yang and Deb (2009, 2010), the authors of cuckoo search provided the pseudocode of the algorithm (reported in Alg. 1) and, in Yang (2021), Yang and Deb (2010), an example of its correct implementation. In the following, we present the many differences we found between these two components. We do so by comparing Alg. 1 to steps 1--4 that correspond to the implementation of the algorithm in Matlab—see Yang (2021) for details.[2]

Algorithm 1 Cuckoo search algorithm as published in Yang and Deb (2009, 2010)

```
 1: begin
 2:     Objective function $f(\vec{x})$, $x = (x_1, \dots, x_d)^T$
 3:     Initial population of $n$ hosts nests $\vec{x}_i (i = 1, 2, \dots, n)$
 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 μ solutions in the population. Therefore, differently from step 2 (perturbation), in Alg. 1, 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 $\vec{x}_t^i$ has been perturbed using Eq. (3), either the perturbed solution $\vec{x}_t^{i'}$ or the initial solution $\vec{x}_t^i$ is accepted as $\vec{x}_{t'}^i$ 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** $f(\vec{x}_t^{i'})$ is better than $f(\vec{x}_t^j)$, where $\vec{x}_t^j$ is a randomly chosen solution, **then** $\vec{x}_t^j$ is replaced by the perturbed solution $\vec{x}_t^{i'}$. 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 indicated in line 11 of Alg. 1. First, according to the rules derived from the metaphor by the authors (see Section 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.

It is therefore unclear whether the authors definition of cuckoo search is the one presented in the paper (i.e., Alg. 1) or in the Matlab implementation. Indeed, in Yang and Deb (2010, p. 3),[3] they write:

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).

[2] We remind the reader that the code of the algorithm is also publicly available in the *Appendix: Demo Implementation* of Yang and Deb (2010) in the version published in the arXiv repository: arXiv:1005.2908.

[3] This quote is taken from the version of Yang and Deb (2010) that is published in the arXiv repository in arXiv:1005.2908.

4. Is the metaphor of cuckoo search justified?

To better understand whether it makes sense to use the "cuckoo's parasitic behavior" as a metaphor for a new optimization algorithm, we consider the criteria of *usefulness*, *novelty* and *sound motivation* to evaluate the metaphor. These criteria aim to give answers to the following questions: (i) Are there concepts in the metaphor that can be useful to solve optimization problems? And in case of positive answer, are these concepts novel in the field of optimization at the moment when they were proposed? (ii) Is the metaphor describing an existing natural optimization process?

4.1. Usefulness and novelty

To evaluate the criteria of usefulness and novelty, we compare cuckoo search to a particular kind of evolutionary algorithms, called *evolution strategies*. Evolution strategies (ES) (Rechenberg, 1971, 1973; Schwefel, 1981; Schaffer, 1985; Bäck et al., 1991; Bäck and Schwefel, 1993) are among the oldest and best known evolutionary algorithms for the solution of 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[4] of *parental selection*, *recombination*, *mutation* and *survival selection* (Michalewicz and Schoenauer, 2013). The specific ES algorithm the authors of cuckoo search reintroduced is the so-called $(\mu + \lambda)$–ES (Schwefel, 1981) with two minor modifications that we explain in detail in this section. To do so, in Alg. 2, we report the cuckoo search algorithm as implemented by its authors (see steps 1--4 of Section 2.3) and, in Alg. 3, the $(\mu + \lambda)$–ES algorithm.

Algorithm 2 Cuckoo search

```
 1: begin
 2:     $t \leftarrow 0$
 3:     initialize $\mu$ cuckoos (solutions)                        ▷ Eq. (2)
 4:     evaluate $\mu$ cuckoos
 5:     while not termination-condition do
 6:         $t \leftarrow t + 1$
 7:         apply mutation to $\mu$ cuckoos to create $\mu$ eggs (new solutions)
                                                                    ▷ Eq. (3)
 8:         evaluate eggs
 9:         select $\mu$ solutions from the set of ($\mu$-cuckoos + $\mu$-eggs)
                                                                    ▷ Eq. (4)
10:         apply recombination to the $\mu$ selected solutions     ▷ Eq. (5)
11:         evaluate the $\mu$ selected solutions and use them as cuckoos for
            the next iteration
12:     end while
13: end
```

The $(\mu + \lambda)$–ES is an evolutionary algorithm in which a population of μ solutions (parents) produce λ new solutions (offspring) to generate a population of $\mu + \lambda$ individuals. The population is then reduced again to μ solutions that constitute the next generation. As we show in Alg. 3, in order to instantiate a $(\mu + \lambda)$–ES, it is necessary to choose the specific operators to be used. In the following, we describe the operators proposed for the $(\mu + \lambda)$–ES and compare them with those used in the implementation of cuckoo search following the order in which they are used in the implementation of cuckoo search, that is, *parental selection*, *mutation*, *survival selection*, and *recombination*; we then discuss the impact of using a different order.

[4] These mechanisms are often referred to as *evolutionary operators* (or just *operators*) in the jargon of EAs.

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

```
1:  begin
2:      $t \leftarrow 0$
3:      initialize $\mu$ parents (solutions)
4:      evaluate $\mu$ parents
5:      while not termination-condition do
6:          $t \leftarrow t + 1$
7:          apply recombination to $\mu$ parents to create $\lambda$ offspring (new
            solutions)                                          ▷ Opt.
8:          apply mutation to offspring
9:          evaluate offspring
10:         select $\mu$ solutions from the set of ($\mu$-parents + $\lambda$-offspring) and
            use them as parents for the next iteration
11:     end while
12: end
```

Parental selection. It refers to the way individuals that will be used to generate a new set of solutions are selected; the selection can be deterministic (one or more specific individuals) or probabilistic (individual are selected based on a probability distribution constructed based on their fitness, ranking, etc.). In the $(\mu + \lambda)$–ES, the parental selection operator can be implemented in many different ways (Bäck and Schwefel, 1993; Bäck et al., 1997). One option that has been used is to let each parent generate one single offspring at each iteration, which results in $\lambda = \mu$. As it can be seen in lines 3, 7 and 9 of Alg. 2 and lines 3, 7 and 10 of Alg. 3, both algorithms use the same parental selection mechanism.

Survival selection. As opposed to parental selection, survival selection refers to the mechanism used to choose the solutions that will be eliminated from the population. In $(\mu + \lambda)$–ES, survival selection operates over parent–offspring couplings, which means that parents will pass from one generation to another until they are replaced by an offspring with better fitness. Cuckoo search uses the same survival mechanism—see Alg. 2, line 9 and Eq. (4)—of the $(\mu + \lambda)$–ES to select between cuckoo–egg couplings.

Mutation. The goal of mutation, which is a type of perturbation, is to induce small random variations to all the variable encoded in the solutions. Both ES and cuckoo search generate mutations by sampling a random distribution. While cuckoo search uses the Lévy distribution, ES strategies have used a number of different types of distribution: the Gaussian distribution, that was used in the original algorithm (Schwefel, 1981); the Cauchy distribution, that was introduced in Kappler (1996) in 1994; and the Lévy distribution introduced in Iwamatsu (2002) and Lee and Yao (2004) in 2002. Therefore, the mutation operator used by cuckoo search is exactly the same as in ES.

Recombination. The goal of recombination, which is also a type of perturbation, is to create one new solution by combining the information of two or more solutions taken from the current population. Recombination is often regarded as an optional component in the $(\mu + \lambda)$–ES; however, it was used in the early variants of ES and it can be applied in a variety of ways. Some of the most common implementations are discrete, intermediate, global–discrete and global–intermediate recombination (Bäck et al., 1997). In cuckoo search, recombination—see Alg. 2, line 10 and Eq. (5)—differs in two aspects from the way this operator is traditionally implemented in ES. First, the specific recombination mechanism implemented in cuckoo search was not defined in the context of ES, but in the one of *differential evolution* (DE) (Storn and Price, 1997; Price et al., 2005). Second, it is applied at the end of the algorithm and not at the beginning as it is normally done in ES.

Regarding the first difference, in DE, a so-called trial vector $\vec{u}$ is created by means of *differential mutation* and *crossover* as follows:

$$u_t^{i,k} = \begin{cases} x_t^{a,k} + \beta \cdot (x_t^{b,k} - x_t^{c,k}), & \text{if } \mathcal{U}[0,1] \geq p_a \\ x_t^{i,k}, & \text{otherwise,} \end{cases} \quad \forall k, \forall i, \tag{6}$$

where $\vec{x}_t^a$, $\vec{x}_t^b$ and $\vec{x}_t^c$ are three different vectors chosen from the population and β is a parameter. Note that, if we set $\beta = \mathcal{U}[0,1]$ in Eq. (6), recombination as proposed in Eq. (5) becomes the same as the creation of the trial vector in Eq. (6).

The second difference is that in cuckoo search recombination is applied at the end of the **while** loop—line 10 of Alg. 2; differently, in $(\mu + \lambda)$–ES, recombination is applied at the beginning of the loop—line 7 of Alg. 3. Similarly to ES, the recombination operator is optional in cuckoo search, since it can be removed from the algorithm implementation by setting parameter $p_a = 0$ in Eq. (5); in this case cuckoo search is exactly the same as the $(\mu+\lambda)$–ES. However, when parameter $p_a > 0$, the two algorithms become equivalent starting from iteration 2; in fact, in Alg. 2 *parental selection* and *recombination* (line 10) are followed by *mutation* (line 7) and then by *survival selection* (line 9). As cuckoo search does not follow the normal order in which the four main components of evolutionary algorithms are used, solutions have to be evaluated twice at each iteration of the **while** loop, which results in wasting computational time.

4.2. Sound motivation

To meet the sound motivation criterion, the authors proposing a new metaphor-based algorithm should show that: (1) the behavior they use in the metaphor is either an optimization behavior or a behavior in which there are components that can be used as effective design choices in the algorithm, and (2) all the components involved in the behavior are well represented in the algorithm. The reason why cuckoo search does not meet point (1) is that neither the metaphor of *cuckoos' parasitic reproduction* is an optimization behavior, nor there are components in this metaphor that can be used to design an effective optimization algorithm. In cuckoo search, the one idea that can make the metaphor of cuckoos to resemble an optimization behavior is the selection mechanism introduced by the authors as one of the rules used to develop the algorithm; however, as we discussed in Section 3, this mechanism is not part of the cuckoos' reproduction strategy.

In order to verify point (2), it has to be the case that the algorithm is in fact using the concepts brought forward by the metaphor that inspired it, otherwise it makes no sense to introduce a new metaphor that has no relation to the algorithm. In cuckoo search, as we have shown in Sections 2 and 3, the metaphor, the algorithm and the implementation are quite different. If we remove the metaphor of cuckoos and the terminology specific to it, the cuckoo search algorithm—as it is presented in Alg. 1—consists of (i) perturbing one single solution randomly taken from the population at each iteration; (ii) using the perturbed solution to replace another randomly selected solution; and (iii) perturbing a fraction of the worst solutions in the population. It is therefore impossible to find a correspondence between the behavior of the cuckoos (Section 2.1), the rules derived from the cuckoos metaphor (Section 2.2), and the resulting cuckoo search algorithm and implementation (Section 2.3).

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 unnecessary metaphor-based algorithms that have been shown to contain no novelty, such as *harmony search* (Weyland, 2010), *black holes optimization* (Piotrowski et al., 2014), etc. 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 an algorithm called $(\mu+\lambda)$–ES, in 1981, and in differential evolution, in 1997. Given the widespread use of *cuckoo search*, as evidenced by its more than 8 700 citations, it is as surprising as it is worrying that this fact has not been formally investigated 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 (Anon, 2015; Dorigo, 2016; Anon, 2021) and steps are being taken to avoid the publication of papers presenting this type of algorithms (see, e.g., Weyland, 2010, 2015; Simon et al., 2011; Melvin et al., 2012; Piotrowski et al., 2014; Fong et al., 2016; Sörensen et al., 2019; Lones, 2020; Camacho-Villalón et al., 2019, 2020; Aranha et al., 2021).

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* (Shah-Hosseini, 2007) and with the *grey wolf* (Mirjalili et al., 2014), *firefly* (Yang, 2009) and *bat* (Yang, 2010) 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 Camacho-Villalón et al., 2018, 2019, 2020) all the metaphor-based algorithms published in the literature; they are simply too many and new ones are being published with alarming regularity (see, e.g., Campelo, 2021; Lones, 2020). We hope that our work can help the research community to see that finding an interesting metaphor with a sound basis and whose usage can bring useful and truly novel ideas on how to solve optimization problems is difficult.

CRediT authorship contribution statement

Christian L. Camacho-Villalón: Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review & editing. **Marco Dorigo:** Conceptualization, Investigation, Writing – review & editing. **Thomas Stützle:** Conceptualization, Investigation, Writing – review & editing.

Funding

Christian L. 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

Anon, 2015. Policies on heuristic search research. J. Heuristics https://www.springer.com/journal/10732/updates/17199246, version visited last on March 26, 2021.

Anon, 2021. Acm transactions on evolutionary learning and optimization. Guidelines for authors. https://dl.acm.org/journal/telo/author-guidelines, version visited last on March 26, 21.

Aranha, C., Camacho-Villalón, C.L., Campelo, F., Dorigo, M., Ruiz, R., Sevaux, M., Sörensen, K., Stützle, T., 2021. Metaphor-based metaheuristics, a call for action: the elephant in the room. Swarm Intelligence 16 (1), 1–6.

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

Bäck, T., Hoffmeister, F., Schwefel, H.-P., 1991. A survey of evolution strategies. In: Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann, pp. 2–9.

Bäck, T., Schwefel, H.-P., 1993. An overview of evolutionary algorithms for parameter optimization. Evol. Comput. 1 (1), 1–23.

Camacho-Villalón, C.L., Dorigo, M., Stützle, T., 2018. 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. In: 11172 of Lecture Notes in Computer Science, Springer, 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 Intell. 13 (3–4), 173–192. http://dx.doi.org/10.1007/s11721-019-00165-y.

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

Campelo, F., 2021. Evolutionary computation bestiary. https://github.com/fcampelo/EC-Bestiary, version visited last on 26 2021.

Černý, V., 1985. A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. J. Optim. Theory Appl. 45 (1), 41–51.

Dorigo, M., 2016. 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, version visited last on March 26, 2021.

Dorigo, M., Maniezzo, V., Colorni, A., 1991. The Ant System: An Autocatalytic Optimizing Process. Tech. Rep. 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Italy.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. – B 26 (1), 29–41.

Dorigo, M., Stützle, T., 2004. Ant Colony Optimization. MIT Press, Cambridge, MA.

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

Fong, S., Wang, X., Xu, Q., Wong, R., Fiaidhi, J., Mohammed, S., 2016. Recent advances in metaheuristic algorithms: Does the makara dragon exist? J. Supercomput. 72 (10), 3764–3786.

García-Martínez, C., Gutiérrez, P.D., Molina, D., Lozano, M., Herrera, F., 2017. Since CEC 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness. Soft Comput. 21 (19), 5573–5583.

Gass, S.I., Fu, M.C. (Eds.), 2013. Encyclopedia of Operations Research and Management Science, 3 Springer Verlag.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Boston, MA, USA.

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

Iwamatsu, M., 2002. Generalized evolutionary programming with levy-type mutation. Comput. Phys. Comm. 147 (1–2), 729–732.

Kappler, C., 1996. Are evolutionary algorithms improved by large mutations? In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 346–355.

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

Lee, C.-Y., Yao, X., 2004. Evolutionary programming using mutations based on the lévy probability distribution. IEEE Trans. Evol. Comput. 8 (1), 1–13.

Lones, M.A., 2020. Mitigating metaphors: A comprehensible guide to recent nature-inspired algorithms. SN Comput. Sci. 1 (1), 1–12.

Melvin, G., Dodd, T.J., Groß, R., 2012. Why 'GSA: a gravitational search algorithm' is not genuinely based on the law of gravity. Nat. Comput. 11 (4), 719–720.

Michalewicz, Z., Schoenauer, M., 2013. Evolutionary algorithms, In: Gass and Fu (Gass and Fu, 2013), pp. 517–527.

Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61.

Piotrowski, A.P., Napiorkowski, J.J., Rowinski, P.M., 2014. How novel is the novel black hole optimization approach? Inform. Sci. 267, 191–200.

Price, K., Storn, R.M., Lampinen, J.A., 2005. Differential Evolution: A Practical Approach To Global Optimization. Springer, New York, NY.

Rechenberg, I., 1971. Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Evolution (Ph.D. 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.

Schaffer, J.D., 1985. Multiple objective optimization with vector evaluated genetic algorithms. In: Grefenstette, J.J. (Ed.), ICGA. Lawrence Erlbaum Associates, pp. 93–100.

Schwefel, H.-P., 1981. Numerical Optimization of Computer Models. John Wiley & Sons Inc..

Shah-Hosseini, H., 2007. Problem solving by intelligent water drops. In: Proceedings of the 2007 Congress on Evolutionary Computation. CEC 2007, IEEE, IEEE Press, Piscataway, NJ, pp. 3226–3231.

Simon, D., Rarick, R., Ergezer, M., Du, D., 2011. Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms. Inform. Sci. 181 (7), 1224–1248.

Sörensen, K., 2015. Metaheuristics—the metaphor exposed. Int. Trans. Oper. Res. 22 (1), 3–18. http://dx.doi.org/10.1111/itor.12001.

Sörensen, K., Arnold, F., Palhazi Cuervo, D., 2019. A critical analysis of the improved clarke and wright savings algorithm. Int. Trans. Oper. Res. 26 (1), 54–63.

Sörensen, K., Glover, F., 2013. Metaheuristics, In: Gass and Fu (Gass and Fu, 2013), pp. 960–970.

Storn, R., Price, K., 1997. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11 (4), 341–359.

Weyland, D., 2010. A rigorous analysis of the harmony search algorithm: How the research community can be misled by a novel methodology. Int. J. Appl. Metaheuristic Comput. 12 (2), 50–60.

Weyland, D., 2015. A critical analysis of the harmony search algorithm: How not to solve Sudoku. Oper. Res. Perspect. 2, 97–105.

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

Yang, X.-S., 2010. A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization. NICSO 2010, In: 284 of Studies in Computational Intelligence, Springer, Berlin, Germany, pp. 65–74.

Yang, X.-S., 2021. Cuckoo search (cs) algorithm. https://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm, mATLAB Central File Exchange. Retrieved March 12, 2021.

Yang, X.-S., Deb, S., 2009. Cuckoo search via lévy flights. In: 2009 World Congress on Nature & Biologically Inspired Computing. NaBIC, pp. 210–214.

Yang, X.-S., Deb, S., 2010. Engineering optimisation by cuckoo search. Int. J. Math. Modell. Numer. Optim. 1 (4), 330–343.