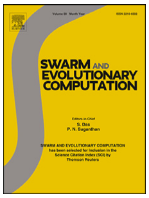
Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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



Survey Paper

A prescription of methodological guidelines for comparing bio-inspired optimization algorithms

Antonio LaTorre[a,*], Daniel Molina[b,b,*], Eneko Osaba[c], Javier Poyatos[b], Javier Del Ser[c,d], Francisco Herrera[b]

[a] Center for Computational Simulation, Universidad Politécnica de Madrid, Spain
[b] DaSCI: Andalusian Institute of Data Science and Computational Intelligence, University of Granada, Spain
[c] TECNALIA, Basque Research and Technology Alliance (BRTA), Spain
[d] University of the Basque Country (UPV/EHU), Spain

ARTICLE INFO

Keywords:
Bio-inspired optimization
Benchmarking
Parameter tuning
Comparison methodologies
Statistical analysis
Recommendations review
Guidelines

ABSTRACT

Bio-inspired optimization (including Evolutionary Computation and Swarm Intelligence) is a growing research topic with many competitive bio-inspired algorithms being proposed every year. In such an active area, preparing a successful proposal of a new bio-inspired algorithm is not an easy task. Given the maturity of this research field, proposing a new optimization technique with innovative elements is no longer enough. Apart from the novelty, results reported by the authors should be proven to achieve a significant advance over previous outcomes from the state of the art. Unfortunately, not all new proposals deal with this requirement properly. Some of them fail to select appropriate benchmarks or reference algorithms to compare with. In other cases, the validation process carried out is not defined in a principled way (or is even not done at all). Consequently, the significance of the results presented in such studies cannot be guaranteed. In this work we review several recommendations in the literature and propose methodological guidelines to prepare a successful proposal, taking all these issues into account. We expect these guidelines to be useful not only for authors, but also for reviewers and editors along their assessment of new contributions to the field.

1. Introduction

Bio-inspired algorithms in the field of optimization is a mature research area. The number of contributions submitted to conferences and journals of this area increases sharply every year [1]. However, a major fraction of these proposals do not prove the goodness of new algorithms appropriately. It is often the case that a work presenting a new bio-inspired algorithm raises doubts in regards to the true contribution of the new proposal. As a result, these concerns put at risk its acceptance by the research community or, alternatively, its capacity to assess the true contribution and significance of the proposed research.

There are a number of reasons for this noted fact, ranging from low-quality papers to works lacking originality [2]. In those cases, there is little to do but to continue investigating toward reaching better results. On the other hand, there is a number of works whose contribution appear to be significant, but that can not be accepted for several reasons. These include, but are not limited to, experimental flaws, questionable/insufficient validation efforts, or a weak discussion of the results. Although these practices could be easily avoided, their repeated occurrence makes them a crucial problem: rigorous experimental practices are needed so that the community could embrace the conclusions drawn from a research work, eventually leading to meaningful advances in this research field.

Several papers can be found with suggestions about important issues found in experimental benchmarks and comparison among meta-heuristics. Each of them focuses on a specific aspect, such as how to design the experiments [3] or how to select and interpret statistical tests to assess the relative differences among algorithms [4]. However, to the best of our knowledge, there is no prior work that deals, at the same time, with different relevant issues that could ultimately jeopardize the fairness assumed in the performance comparisons among techniques. When the goal is to discriminate which algorithm performs best among a set of possible choices, fairness should be an unwavering principle. This includes the design of the benchmark, the selection of performance metrics, and the analysis and discussion of the results. Without this principle being guaranteed along the experimental workflow, conclusions extracted from these studies will remain in doubt.

[*] Corresponding authors.
E-mail addresses: a.latorre@upm.es (A. LaTorre), dmolina@decsai.ugr.es (D. Molina), eneko.osaba@tecnalia.com (E. Osaba), jpoyatosamador@ugr.es (J. Poyatos), javier.delser@tecnalia.com (J. Del Ser), herrera@decsai.ugr.es (F. Herrera).

https://doi.org/10.1016/j.swevo.2021.100973
Received 18 April 2020; Received in revised form 23 June 2021; Accepted 13 August 2021
Available online 20 August 2021

The main objective of this manuscript aligns with the above remarks. Specifically, we review the literature background and provide a set of useful guidelines intended for researchers to avoid common mistakes in experiments with bio-inspired meta-heuristics. Those bad practices could eventually generate doubts about the fairness of the comparisons reported therein. Our methodological approach is pragmatic, mainly aimed at making it easier for new researchers entering this area to prepare an experimental section under high quality standards. We start our study by exploring current approaches for algorithms' analysis. We pay special attention to bad practices, identified not only in this work but also in previous literature. All this information can be found in Section 2.

Then, we propose 4 guidelines that authors introducing a new algorithmic proposal should take into account to boost their chances to get their work embraced by the community. We provide a brief motivation for each of them in the following paragraphs, whereas Sections 3–6 present detailed methodological guidelines for the elaboration of successful proposals:

- **Guideline #1: Benchmarks.** Sometimes the benchmark is a real-world problem. In these cases, the benchmark gauges how the proposed algorithm tackles the problem at hand. By contrast, in other cases the proposal is compared against other reference algorithms by using a benchmark specially designed to test their performance. In any case, the selection of the appropriate benchmark is an important issue, since the conclusions that can be extracted from the study depend deeply on the test bed. Unfortunately, the chosen benchmarks frequently present some features that might favor algorithms with a particular bias. This is, of course, not desirable for the sake of fairness in the subsequent comparisons. Thus, the results obtained by the newly proposed solver must be analyzed by taking into account the different characteristics of the test problems covered by the benchmark at hand.
- **Guideline #2: Validation of the results.** The presentation of raw results arranged in full-page tables is, today, not enough. A proper validation of the results from a statistical point of view should always be provided along with the aforementioned tables. In this sense, it is important that not only statistical tests are used, but also that the correct ones are applied. It is quite usual to find parametric tests that are used without ensuring that the assumptions required for those tests are met by the obtained results. In addition, we also recommend visualization techniques for comparative analysis. They can summarize a significant amount of information in a condensed representation that can be quickly grasped and interpreted by the reader.
- **Guideline #3: Components analysis and parameter tuning of the proposal.** The hypotheses of the proposal must be clearly stated at the beginning of the paper, and discussed once the results have been validated. Moreover, the authors should conduct a thorough analysis of the results considering, at least, the following aspects: search phases identification (balance between exploration and exploitation), components and complexity analysis (individual analysis of the contribution of each of the components of the overall method, and their complexity), parameter tuning of the algorithm, and statistical comparison with state-of-the-art algorithms (as described in Guideline # 2).
- **Guideline #4: Why is my algorithm useful?** Finally, prospective contributors should clearly state why their proposed algorithm should be considered relevant by the rest of the community. In this guideline we discuss this issue in depth from different points of view. We also suggest several reasons for which a new proposal poses an advance in knowledge (i.e. it is found to be competitive against state-of-the-art methods, it presents methodological contributions that stimulate further research, or other reasons later elaborated).

In order to illustrate each of the problems discussed in this contribution, we will resort to different use-cases coming from our previous experience or especially tailored for the purposes of this study. Data utilized for each of these exemplifying problems may vary, as not all situations can be clearly explained with one single example. We also provide several case studies in Section 7 that describe the process of designing and evaluating new algorithms according to the methodology proposed herein. Each of them embraces all the methodological guidelines by, first, selecting a standard benchmark , a performance measure (ranking) and the reference algorithms. Then, we conduct a proper statistical validation of the results compared with those of the reference algorithms. We also use visualization techniques to offer a more clear view of the results. To continue, the contribution of each component of the new algorithm is analyzed to ensure that all of them contribute to the results of the overall method. Each case study finishes with a discussion on the usefulness of the new proposal. As can be seen, it properly covers all the methodological guidelines proposed in this contribution.

As a summary, the main key elements of this work are:

1. A literature review, with an emphasis on the identification of bad practices in the analysis of new algorithmic proposals.
2. Four methodological guidelines to help authors achieve contributions adopted by the community.
3. Several case studies, as described in the previous paragraph, that simulate the process of proposing a new algorithm by following the aforementioned four methodological guidelines.

The remainder of this paper is organized as follows. Section 2 discusses several previous useful guidelines and recommendations in the literature. Sections 3 through 6 present and discuss the guidelines proposed in this work, whereas in Section 7 we provide several case studies covering some of these guidelines. Finally, Section 8 concludes the study.

2. Relevant issues for the proposal of methodological guidelines

In any field of science, it is crucial to work under correct and unbiased experimental conditions, and to conduct a rigorous and adequate analysis of the obtained results. However, sometimes there are small aspects that can lead to inadvertently biased comparisons, partially benefiting a certain type of algorithms over the others.

In this section we review prior work in the literature, advising constructively against issues that could generate objective doubts about the strength of the experimental claims. Specifically, we revisit several particular topics of relevance for the current study: an inadequate or incomplete description of an algorithm (Section 2.1), the presence of bias in the search process (Section 2.2), relevant features that should be taken into account when selecting benchmarks (Section 2.3), prior studies focused on the validation of experimental results (Section 2.4), and existing works on component analysis and parameters tuning (Section 2.5). Topics tackled in this first background analysis have a straightforward connection with our methodological guidelines given in Sections 3 to 6.

2.1. Inappropriate description of the algorithm

This is a common issue in many proposals, especially in those of more advanced methods. Reproducibility of scientific results should always be required and this is not possible if some implementation details are missing [5,6]. This includes not only a high-level description of the algorithm, but also implementation details, dependencies, parameters values, etc. Furthermore, if the proposal is based on a pre-existing method, differences with regards to the base algorithm should be stressed. Finally, and although this is still not mandatory in most journals and conferences, we encourage authors to freely distribute the source code of their algorithms, to allow other users to better replicate their results.

2.2. Bias in the search process

One of the most critical decisions when evaluating an algorithmic proposal is the selection of the benchmark used to show its goodness. Unfortunately, for many papers the testbed was proposed by the same authors, and is normally a combination of well-known synthetic theoretical functions. Moreover, the only measure of performance is often the benchmark proposed. The design of a good benchmark is not an easy task, and they can be used for benefiting newly proposed methods by exploiting any bias in the search algorithm [7].

- *Optima close to the center of the domain search:* one of the most typical sources of bias is the tendency of some algorithms to explore with more intensity in the surroundings of the center of the domain search. This issue was first discussed and termed *structural bias* in [8]. Many versions of Genetic Algorithms (GA), Particle Swarm Optimization (PSO), or Differential Evolution (DE) [9], have exploited this characteristic, as it has been traditionally where the optimum of the problem under analysis was located. Those algorithms, on the other hand, tend to exhibit a bad performance near to the bounds of the domain search [10]. In [11] and [12] a detailed experimentation about the structural bias in search algorithms is given. Avoiding this kind of bias in the design of algorithms is not easy, but at least they should not be evaluated on benchmarks favored by these biases during exploration. One popular approach to avoid having the optimum into the center of the domain search is shifting.
- *Sensitivity to the coordinate system:* another possible source of bias emerges when the exploration of the search domain is done mainly by moving along the directions of the coordinate system. In this regard, some algorithms have proven to be very sensitive to the coordinate system [13]. Some benchmark functions are rotated to test the invariance of the algorithms to such transformations. Ideally, the algorithm should be invariant to these rotations, such as Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [14]) and Black-Box DE [13].

In order to compare algorithms designed to avoid these sources of bias, several benchmarks taking into account these issues have been proposed. This allows a fairer comparison between algorithmic proposals, that can be compared on the same testbed. In particular, in real-parameter optimization, several benchmarks have been proposed since 2005 to date [15–19]. All these benchmarks try to avoid the first source of bias by shifting the location of the global optimum. Furthermore, in the more recent testbeds an increasing number of functions have been rotated, featuring more complex landscapes. A more detailed view of the different benchmarks proposed and their evolution can be found in [20].

On a closing note in this matter, benchmarks should definitely evaluate the sensitiveness of the algorithms to different sources of bias as the ones mentioned previously. However, when moving from the academic realm towards real-world scenarios, it is important to bear in mind that the goal is not to find a good *meta-heuristic* approach, but rather to solve a given problem efficiently. This being said, the availability of a priori information on any bias of the problem to be solved should be exploited. Indeed, if a solver exploiting a certain source of bias known to exist in the problem, then this solver should be preferred. Actually, benchmarks are useful to identify algorithms with a good performance on problems with similar sources of bias. These thoughts concur with recent work around the exploitation of problem-specific knowledge when designing an optimization algorithm aimed to solve it efficiently (grey-box optimization, see e.g. [21] and references therein). In summary, if the objective of the study is to solve a given problem, taking advantage of possible sources of bias when designing the algorithm is convenient and advisable.

2.3. Relevant features of benchmarks

Different properties related to the landscape of the functions composing the benchmark should be addressed for a fair analysis of the behavior of the proposed solver(s). Among them:

- **Separability of the components:** some functions can be easily solved by optimizing each dimension individually, so it is crucial not to use only separable functions in the benchmark. This is the case, for example, of the CEC'2008 LSGO benchmark proposed in [22], in which many functions are of this type. In more recent benchmarks, especially in the field of large-scale optimization, the focus is on evaluating the capability of the algorithms to identify existing subcomponents in the functions. If the new proposal deals with this kind of problem, it should be tested on a benchmark that allows evaluating this characteristic, such as e.g. the one proposed in [23].
- **Dimensionality of the problems:** another important issue is the dimensions of the benchmark, because some algorithms are designed to work properly only for very low-dimensional problems. As the size of the search domain increases exponentially with the dimensionality of the problem, the so-called *curse of dimensionality* [24] poses a significant computational challenge. This is particularly the case for well-known algorithms such as PSO [25] and DE [26]. Actually, as the performance of most algorithms degrades when the dimension grows, the current trend is to develop specific algorithms for problems with higher dimensionality. Nonetheless, it is increasingly important to offer robust performance for a medium range of dimension values. Some benchmarks are designed to evaluate the performance of algorithms in problems of small to moderate size [15–19], whereas others aim at problems of a much larger size [22,23,27]. These are problems of a very different nature and, normally, algorithms with an outstanding performance over one type of problems do not perform as such when applied to other types of problems. For example, strategies such as computing the covariance matrix of solutions as done by [14] do not scale up nicely for problems of larger size. Furthermore, these latter problems require increased exploration abilities of the algorithm to cover a much broader search space, which has a significant cost in terms of fitness evaluations.
- **Number of optima of the problems:** There are functions that have only one optimum, and other that have several optima called multi-modal functions. Multi-modal functions can have several global optima, with the same fitness value, and also multiple local optima, with different (worse) fitness values. The presence of local optima increases the difficulty of the optimization process because the algorithm can be stuck in them. This is the case of algorithms with a strong elitist behavior [28]. The difficulty increase is due to the fact that the black-box optimization algorithms are not aware of the number of local optima, although several works have proposed methods to estimate their number [29].
- **High-conditioning problems:** A high-conditioning or ill-conditioning function is one in which a small change in the variables of the solution implies a large change in its fitness value. This means that the correct solution/answer to the problem becomes hard to find for optimization algorithms [30,31].
- **Noisy functions:** finally, noise is another important factor that has not been widely considered in the literature. However, this is changing recently in several recent benchmarks that also consider this issue. These benchmarks, such as the BBOB [32] or the Nevergrad [33] benchmarks, explicitly include functions with different degrees of noise that resemble real-world scenarios in which noise can be a very important issue. Despite this recent interest, very few studies have hitherto dealt with noisy functions [34,35].

2.4. Validation of the results

Selecting competitive algorithms to be included in the comparison is another crucial aspect in benchmarking. In the current literature many different algorithms can be found and chosen to be reference algorithms for a given benchmark. Unfortunately, there is no clear criterion to make such a selection. Although good practices usually suggest comparing against the most recent state-of-the-art algorithms, it is often the case that authors only compare their proposal against basic or very similar versions of other algorithms, as was spotted in [36].

In this context, studies comparing different algorithms are scarce, and the results reported therein strongly depend on the problem(s). In [37] a benchmark of classic functions was used to compare among Cuckoo Search (CS), PSO, DE, and Artificial Bee Colony (ABC). The study concludes that the best results were obtained by CS and DE, and the worst ones were those rendered by ABC. On the other hand, for a different problem [38], ABC was found to perform best, followed by DE and PSO.

For the methodological part of the comparisons, there are far more studies. Statistical tests, for instance, lay at the core of prior contributions on this matter. However, such contributions are frequently written from a statistical point of view – like the one by Demšar [39] – making it difficult for researchers in this field to embrace their methodological recommendations. More recently, some tutorials have tried to bring together the fields of meta-heuristics and inferential statistics [4]. Some examples can be found in [40], in which a statistical treatment is suggested for distinguishing between measurements of performance in adaptive evolutionary algorithms. Another good example is [41], which shows that in a popular real-parameter benchmark (CEC'2005), conditions needed for running parametric hypothesis tests did not hold, and non-parametric tests were thus recommended. More recently, in [42], some recommendations for the comparison of evolutionary algorithms are provided, which can be even extrapolated to machine learning benchmarks.

Another important issue from a methodological point of view is the assessment of the performance of bio-inspired algorithms from the perspective of the experimental design. Some studies [5] provide general recommendations to design experiments for the comparison of algorithms in a similar way to what we do in this contribution. However, these recommendations are far more general as it targets a broader scope (the design of algorithms and not bio-inspired optimization methods, specifically). This difference in the target of the proposed guidelines makes it miss some important issues inherent to bio-inspired optimization methods that we cover in this contribution. Other works, although focused on optimization methods, are more specific in their recommendations, targeting specific issues such as the selection of problems and performance measures [43–45]. Finally, other studies are more oriented to the analysis and definition of experimental frameworks such as those used in CEC special sessions and the COCO framework [6]. While this is also a very relevant contribution to the field of experimental design in the context of evolutionary algorithms, it deals with the problem from a different perspective and should be considered as a complement to the guidelines here presented.

2.5. Components analysis and parameter tuning

New proposals are frequently based on previously existing algorithms to which certain components have been added/replaced. However, the addition/replacement of new components is not always adequately justified. This is a fundamental design flaw that can contribute to make more and more complex algorithms in which the new additions only report a marginal contribution to the overall performance of the method. Authors should clearly discuss about the contribution of each new component in order for the new proposal to be considered significant.

Another important research topic in the design of meta-heuristic algorithms is the selection of the values of their parameters. Indeed, parameters can be a double-edged sword. On the one hand, they grant flexibility to control the search behavior of the algorithm. On the other hand, finding the parameter values that lead to the best search performance is another optimization problem itself [46]. For this reason, there is a long literature record of studies dealing with the best parameter values for different meta-heuristic algorithms, such as GA [47], PSO [48], or DE [26,49].

The tuning of parameters can be carried out by means of different automatic tuning tools. There are several consolidated tools of this nature, with different features [50]. F-RACE [51] and I-RACE [52] are iterative models, which, at every step, evaluate a set of candidate parameter configurations, discarding several of them along the search. These methods remove candidates upon the result of statistical comparisons, e.g. two-way analysis of variance. I-RACE is an implementation of Iterative F-RACE that includes several extensions, such as a restart using normal distributions. REVAC [53], on the contrary, relies on an Estimation of Distribution Algorithm (EDA). For each parameter, REVAC starts by sampling an uniform distribution of values. Then, at each step it reduces the value range of each parameter by using specially designed transformation operators, considering an entropy measure. ParamILS [54] is an iterative local search algorithm that, from a default parameter configuration, applies a local search with random perturbations to improve the configurations.

3. Guideline #1: Benchmarks

The first decision that a researcher must face when preparing a new contribution in the field of optimization is the selection of the benchmark to test the newly proposed algorithm(s). Once this is done, the authors must identify relevant algorithms to compare the obtained results and guarantee the significance of conclusions drawn therefrom. This first guideline deals with these two crucial aspects: the selection of the benchmark (Section 3.1), and the reference algorithm(s) to which the proposal is compared (Section 3.2).

3.1. Selection of the benchmark

As mentioned before, this first decision is one of the most important factors to prove the quality and performance of an algorithm. In real-world problems, this is not a decision at all, because the benchmark is the problem to tackle. In contrast, when designing and improving meta-heuristic techniques, the selection of a benchmark is an important decision to take. This issue is common for all types of optimization. During the last years, benchmarks have been proposed for several types of optimization (such as combinatorial and numerical optimization) with the main goal of becoming a standard for future comparisons. Without loss of generality, during the following section we will focus on numerical optimization, yet all conclusions and recommendations given hereafter are applicable regardless of the domain.

In the field of numerical optimization, special sessions devoted to benchmarking have taken place in reference events such as the IEEE Congress on Evolutionary Computation or the Genetic and Evolutionary Computation Conference. In these events, participants could compare their algorithms in a controlled environment with a homogeneous set of functions [20,55]. Nonetheless, the efforts towards providing standard benchmarks and tools for the comparison of bio-inspired optimization methods is not limited to special sessions and competitions, as shown below:

- Special Sessions and Workshops:
  - IEEE Real-Coding Special Session (since 2005)
  - Black-Box Optimization Benchmarking (BBOB) Workshops at GECCO and PPSN (since 2009)
  - Black-Box Optimization Competition (BBComp) (since 2015)
  - IEEE Large-scale Global Optimization Special Session (since 2008)
  - Benchmarking Workshop at GECCO and PPSN (since 2020)
  - Other (more specific) special sessions: constrained, multimodal, real-world, etc.
- Tools:
  - Comparing Continuous Optimizers (COCO) framework[1]
  - IOHProfiler[2]

[1] https://github.com/ttusar/coco/

[2] https://iohprofiler.github.io/

  - Nevergrad[3]
  - TACO: Toolkit for Automatic Comparison of Optimizers[4]
  - P. N. Suganthan's CEC benchmarks[5]
  - IEEE CEC LSGO Benchmark[6]
- Other initiatives:
  - Benchmarking Network[7]
  - Cost Action CA15140 (ImAppNio)[8]

However, some works simply overlook these standard benchmarks and tools. Instead, they rather choose their own subset of functions to evaluate their proposal. This is problematic for several reasons. First, as follows from Section 2.2, it is very hard to know whether there is any bias in the selection of the functions from the point of view of the performance of the algorithm under consideration. Secondly, many different benchmark functions exist (or can be defined). Therefore, it becomes very difficult to appraise the different characteristics of all such benchmark functions. Finally, comparisons with other reference methods usually imply running them by the same authors of the new method, as it is very unlikely that multiple studies would have focused on the same selected functions or experimental conditions. As a result, assessing the quality of a new contribution that does not use standard benchmarks gets almost impossible to accomplish, and therefore should not be given credit by the community.

However, there are two special situations in which researchers have no other option but to use *ad-hoc* generated problem instances: i) when the problem to be solved has never been tackled in the previous literature, and hence no benchmark can be found; or ii) when a real-world problem is under consideration, with specific requirements and constraints. In these cases, the instance generation process must be deeply detailed, and all the instances generated should be shared for other researchers to replicate and improve upon the presented results. Furthermore, for any of these two alternatives, practitioners should generate a benchmark as realistic and general as possible.

On the other hand, one should also be very careful when selecting a benchmark for the experimentation to be carried out. Benchmarks in the literature have been conceived with some objectives in mind, and are appropriate to test certain characteristics of the algorithms under evaluation. A non-exhaustive list of these characteristics follows:

- **Bias avoidance of the search algorithm:** In order to avoid the problems described in Section 2.2, it is highly advisable that the optimum is not located at the center of the domain search (e.g. by shifting). Furthermore, rotation should be enforced to test the sensibility of the algorithm to the coordinate system.
- **Sensitivity to the number of local optima:** The number of local optima of a function is another important characteristic of a test problem. Unless properly considered in the algorithmic design, search spaces with multiple local optima may negatively affect the convergence of metaheuristics towards the global optimum. In this kind of problems, multiple local optima can act as basins of attraction, preventing the algorithm from reaching the global optima if it does not correctly balance its exploration/exploitation ratio.

Additionally, benchmarks normally establish the experimental conditions under which algorithms should be evaluated. In particular, it is very frequent that a benchmark selects a common:

- **Measure of performance:** In evolutionary algorithms it is usual to use the mean error obtained for the different runs. However, there are more adequate measures and indicators of performance for dynamic optimization [56], and multi-objective optimization [57]. Additionally, other performance indicators, such as running time of memory footprint, could also provide interesting insights on the behavior of the algorithms (especially in real-world scenarios). Nonetheless, this should be carefully considered as those measures might be biased depending on external factors such as the programming language of choice and not the algorithm itself, which can hinder the comparison.
- **Stopping criterion:** In order for a comparison of several algorithms to be fair, all of them must conduct a similar effort in finding a solution. This is normally achieved by establishing a common stopping criterion. If this were not the case, one algorithm could be stopping when a predefined precision is reached whereas other algorithm could run until a maximum budget of fitness evaluations is exhausted. The results of both algorithms are not comparable and this is why most benchmarks define a common stopping criterion. It also allows grasping the full picture over the performance of algorithms, as different methods may yield different convergence rates, and the results of the comparison might differ depending on the checkpoint at which they are evaluated. This issue will arise and will be discussed in the use cases presented in Section 7.

Finally, there is another relevant characteristic from a design perspective that is not specifically linked to the functions themselves, but to the algorithms that solve those functions. The algorithms use the fitness function to guide the search process, but sometimes only the ranking of the solutions computed from fitness values is actually used. In this sense, some recent algorithms such as the Firefly algorithm [58] or the Grasshopper optimization algorithm [59] use the quantitative information provided by the fitness function to guide the search process, whereas others propose mutation and/or selection methods [60] that only require to know whether one solution is better than the other. In some real-world scenarios, the last approach can be very beneficial as it simplifies the process of defining the fitness function.

3.2. Selection of the reference algorithms

Another important issue, which is actually related to the previous guideline, is the selection of the reference algorithms to include in the comparison. On the one hand, if the proposed algorithm relies on some other basic algorithms, these should be included in the comparison to check the individual contribution of each of them, as we will discuss in detail in Section 5. On the second hand, once the benchmark has been selected, the best-so-far methods for that particular benchmark should be also considered in the comparison. Unfortunately, many papers fail to compare their proposed algorithm against competitive counterparts [36].

A well-informed experimentation should, at least, include the best algorithms in the special session where the benchmark was originally proposed (as it is usually the case). We refer to [20] for an updated review on special sessions and competitions on continuous optimization.

Finally, authors should also consider similar algorithms, not only from the same *family* (e.g., PSO, DE, GA...) but also the base algorithm, if the proposal is an improvement over a previous algorithm, or other similar approaches (for example, different memetic algorithms with a common local search). Our claim in this regard is to stop comparing new methods with classic algorithms that have been clearly outperformed by newer ones. Comparisons adopting this misleading strategy should be avoided for the questionable scientific contribution of their proposal.

4. Guideline #2: Validation of the results

Just as important as a correct experimentation design (see Guideline #1) is a principled validation procedure for the benchmark. For this purpose, we emphasize on two different tools: statistical analysis and comparative visual analysis. Both approaches are covered in the following two subsections: statistical analysis (Section 4.1), and visualization techniques for comparing meta-heuristics (Section 4.2).

[3] https://github.com/facebookresearch/nevergrad
[4] https://tacolab.org/
[5] https://github.com/P-N-Suganthan
[6] http://www.tflsgo.org/special_sessions/cec2019.html#original-code
[7] https://sites.google.com/view/benchmarking-network/
[8] https://imappnio.dcs.aber.ac.uk/

Table 1
Recommended tests according to data characteristics.

Conditions	Equal variances	Unequal variances
Normally distributed	Paired Student's *t*-test	Paired Welch's *t*-test
Not normally distributed	Wilcoxon signed-rank test	

4.1. Statistical analysis: Non-parametric tests and beyond

Statistical comparison of results should be deemed mandatory in current benchmarks among bio-inspired algorithms. However, even if statistical comparisons are made in studies reported nowadays, they are not always carried out properly. In inferential hypothesis testing, there are some popular methods, such as the *t*-test or the ANOVA family of tests. However, these tests are referred to as *parametric tests* because they assume a series of hypotheses on the data on which they are applied (i.e., on the parameters of the underlying distribution of the data). If such assumptions do not hold (for example, the normality assumption for the results), the reliability of the tests is not guaranteed, and alternative approaches should be considered. Thus, either these conditions are checked to be true (i.e., by using a normality test to prove the normality in the distribution), or another type of tests that do not make these assumptions should be used instead. This is the case of *non-parametric* tests, which do not assume particular characteristics for the underlying data distribution. This non-parametric nature must be seen as an advantage over the aforementioned parametric tests (for their independence with respect to data) but also as a limitation (non-parametric tests are less powerful) imposed by the nature of the underlying data, that do not satisfy the requirements to use the more powerful parametric methods.

As a result of this, parametric tests should be preferred whenever they can be safely used (i.e. whenever the hypotheses on the underlying data distribution are met). Unfortunately, this often fails to be the case when comparing the results of bio-inspired algorithms. Consequently, non-parametric tests should be used instead [4]. A common error (less frequent in current research) is to apply parametric tests without checking if the required hypotheses are satisfied.

In the following paragraph we provide a workflow to decide which kind of test to choose. It consists of the following steps:

1. Check the conditions required for the application of the parametric test of choice (normally, Student's *t*-test).
   (a) Normality: Shapiro-Wilk test [61] or Kolmogorov-Smirnov. Shapiro-Wilk should be used with smaller sample sizes [62].
   (b) Homocedasticity (equal variances): Levene's test [63].
2. If both conditions are satisfied, apply Student's *t*-test [64].
3. If only normality can be guaranteed, then Welch's *t*-test alternative is considered [65].
4. If none of the assumptions on the underlying distribution holds, then the non-parametric Wilcoxon signed-rank test is used [66].

Once the appropriate test has been selected, the comparison can be carried out. First, the ranking of each algorithm over the whole benchmark must be computed, and the significance of the differences in the ranking values must be tested. Friedman rank-sum test can serve for this purpose [67]. If differences are declared to be statistically significant by the Friedman test, then we proceed to the pairwise comparisons with the test selected from Table 1. In those pairwise comparisons a reference algorithm is compared against all the other methods selected for validation. Normally, the reference algorithm is selected to be the one with the best average ranking or, alternatively, the new proposal presented in the work.

Another typical oversight noted in the literature is to neglect the accumulated error. A statistical test for two samples, like the Wilcoxon's test, has an estimated error, but this error increases with each pair of comparisons. Thus, when simultaneously comparing the results of our proposal with those attained by several other algorithms, the application of Wilcoxon's test (or others such as the *t*-test) is totally discouraged, because it cannot ensure that the proposal is statistically better than all the other reference algorithms. Thus, once the pairwise p-values have been computed, a correction method must be used to counteract the effect of multiple comparisons, by controlling either the family-wise error rate, or the false discovery error rate [68]. Several procedures have been proposed to this end, among which Bonferroni-Dunn [69], Holm-Bonferroni [70], Hochberg [71] and Hommel [72] are the most widely used [4].

Also linked to statistical validation, another recommendation is to provide the p-values of tests carried out in the experimentation. However, we note that p-value, as such, is not a totally objective measure, as it is highly dependent on the sample size [73]. Section 7 presents several examples of comparisons that goes through the methodological steps prescribed in this second guideline.

Traditionally, the standard null hypothesis testing methods we just described have been used for comparing the performance of different metaheuristic algorithms. This has produced over the years a large number of different post-hoc tests and graphical representations that ease the process of evaluating which algorithm performs best on average, with statistical significance (e.g. critical distance plots). However, much criticism has arisen lately in different aspects of these tests that suggest that a step further should be made towards alternative means to assess the statistical relevance in performance comparison studies. To begin with, the use of the so-called significance parameter (often denoted as α) clashes with its lack of interpretability, and does not link directly to the performance differences observed among the counterparts in the benchmark. Furthermore, conclusions drawn from non-statistical hypothesis testing are largely sensitive with respect to the number of samples used for their computation, as well as the number of algorithms and problems over which the study is made. The work by Benavoli et al in [74] lit a light on this matter, and proposed the use of Bayesian analysis for comparison analysis. The Bayesian paradigm makes statements about the distribution of the difference between the two algorithms under comparison, which can be of help when the null hypothesis significance test (NHST) does not find significant differences between them. The rNPBST package [75] and the jMetalPy framework [76] are useful tools to apply these tests.

The use of different tests can help putting the results in context. As it is mentioned in [77], authors encourage the joint use of non-parametric and Bayesian tests in order to obtain a complete perspective of the comparison of the algorithms' results: *"While non-parametric tests can provide significant results when there is a difference between the compared algorithms, in some circumstances these tests do not provide any valuable information and Bayesian tests can help to elucidate the real difference between them"* [77]. Practitioners must consider this possibility to complement well-known non-parametric tests when they do not provide a full difference among algorithms.

4.2. Visualization techniques for comparative analysis

Visualization techniques are other useful methods to report results when comparing several bio-inspired algorithms. The main advantage of these approaches over reporting raw data in tables is that they can be much easily interpreted by the reader. They have also the ability to summarize the information covered by one or even multiple tables.

In Fig. 1 we provide an example of some visualizations that illustrate the performance of several algorithms on the CEC'2013 LSGO benchmark. Fig. 1(d) uses a radar chart to visualize the average ranking of each bio-inspired algorithm on different groups of functions. Each group has been defined according to some common characteristics present in many state-of-the-art benchmarks: degree of separability, modality, etc. Figs. 1(c)-1(b) provide an alternative view on the same data: it does not depict their average behavior, but instead the number of times in which one algorithm obtained the best overall results for problems belonging to each of the previously defined categories. In Fig. 1(c), the

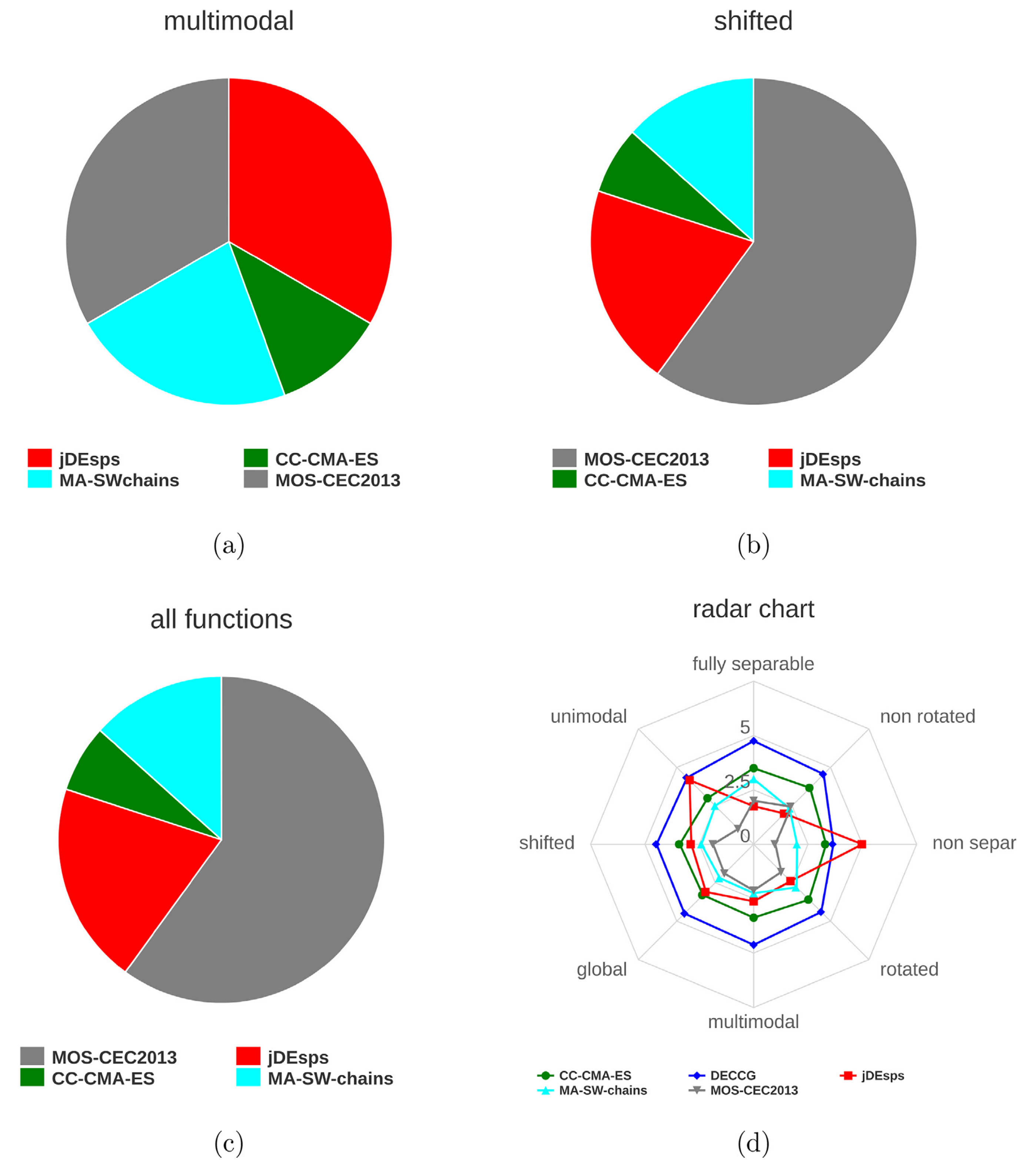
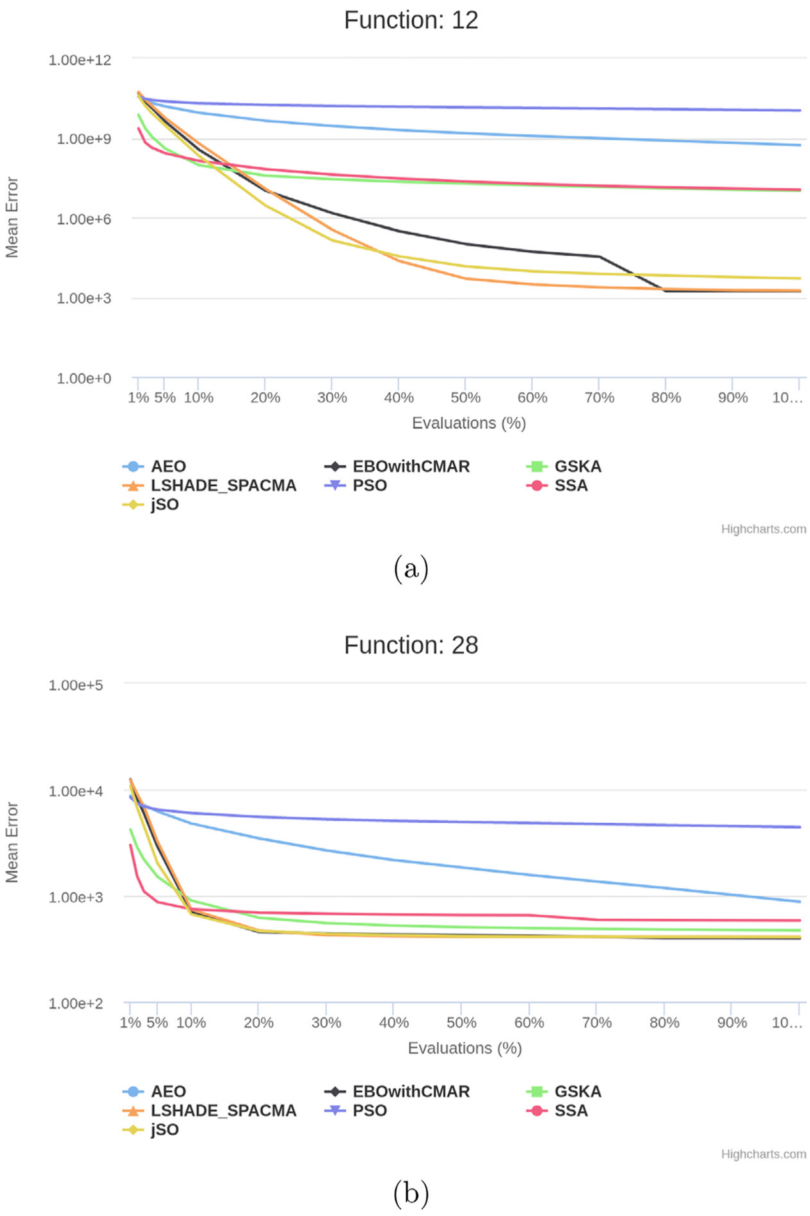


Fig. 1. Different visualizations for the comparison of the performance of several algorithms on the CEC'2013 LSGO benchmark: (a) Average ranking of algorithms on different types of functions; (b) Fraction of functions for which a specific algorithm obtained the best results; (c) Fraction of multimodal functions for which a specific algorithm obtained the best results; (d) Fraction of shifted functions for which a specific algorithm obtained the best results.

whole benchmark is considered, whereas Figs. 1(a) and 1(b) show the results for multimodal and shifted functions, respectively.

Another typical visual representation is that of the convergence of an algorithm. In this case, the variable being discussed is the convergence speed of the methods involved in the comparison. An example of this kind of plot is provided in Fig. 2. As can be seen in Fig. 2(a), which depicts the convergence speed of multiple algorithms on F11 of the CEC'2017 benchmark, although EBOwithCMAR is able to reach the minimum error among the competing algorithms, it is not until the end



(a)

(b)

Fig. 2. Convergence curves for several functions of the CEC'2017 benchmark and dimension 10: (a) function 12; (b) function 28.

of the run that it overcomes other methods. For the majority of the time, jSO yields better results.

With a similar visual idiom, we can represent very different data. For instance, it could be interesting to measure the ratio of problems solved as the number of fitness evaluations increases. These plots not only evince which algorithm is able to reach the optima in more problems, but also how much effort is required to accomplish it. An example of this visualization is provided in Fig. 6, which can be found in Section 7.1.

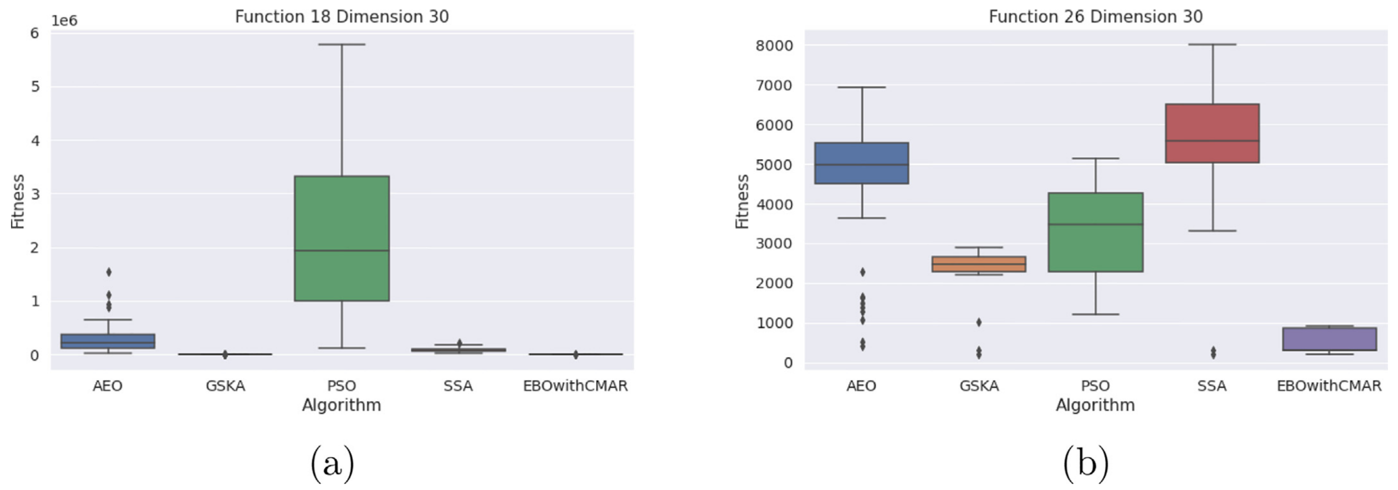
Finally, even common visual representations such as boxplots can be very useful. In the context of bio-inspired optimization, it is a good way to show not only the mean error, but also the error for the different runs. Fig. 3 provides two examples comparing the results of different algorithms for the case study in Section 7.1.

The visual idioms suitable to represent information in this kind of comparisons are not limited to the two examples given in this section. There are many other alternative representations that can help to gain insight in the results under discussion.

What should be clear is that different idioms support different types of analyses, and that visualization techniques should be carefully selected in order to present the results in a summarized yet insightful fashion. All in all, our recommendation at this point is not only to visualize the results of the comparison, but also to use these techniques to complement and/or summarize the information provided by other means.

5. Guideline #3: Components analysis and parameter tuning of the proposal

This third guideline could be seen as a check-list for the discussion section. It covers the full proposal analysis, from the statement of the hypotheses to be proved by the experimentation to the presentation of the results of the different comparisons needed to assess the contribution of the work at hand. This section elaborates on this list by pausing at the following aspects: origin and work hypotheses that motivate the proposal (Section 5.1), the identification of the search phases, with claims on them solidly informed by empirical evidences (Section 5.2), the individual analysis of algorithmic components of the proposal (Section 5.3), and parameter tuning and analysis (Section 5.4).



(a)

(b)

Fig. 3. Box-plots for the CEC'2017 benchmark and dimension 30: (a) function 18; (b) function 26..

5.1. *Origin, hypotheses and proposal*

Before starting to discuss about the benefits of the novel approach(es) proposed in the work, it is necessary to have a clear perspective of the expected results, i.e., the work hypotheses of the present study. Furthermore, authors should clearly describe how the proposal helps to attain the targeted objectives. Most of the times we assume that the main contribution of a work is a new algorithm or a new modification capable of improving the state of the art in some particular benchmark (properly chosen as already discussed in Section 3.1). While this assumption often holds in practice, any other contribution to be taken into account must be clearly highlighted at this point for the work to be considered relevant. We will discuss further on this issue in Section 6.

5.2. *Search phases identification*

A key issue when solving optimization problems with complex fitness landscapes is to keep an appropriate balance between exploration and exploitation [78,79]. This is a recurrent statement in many contributions, especially when the obtained results seem to support that hypothesis (at least in terms of overall accuracy). However, most of these studies fail to provide evidence on how the exploration/exploitation balance is maintained. It is normally not enough to state that *"algorithm A is better than algorithm B because it properly maintains the exploration/exploitation balance"*. This type of statements requires an empirical analysis to check to which extent this claim is supported by evidence.

The work reported in [80] analyzes this issue from a dual perspective. First, they inspect how different authors measure the exploration/exploitation balance, to conclude that this is mainly carried out by means of indirect measures (e.g., the diversity of the solutions). Second, they propose a taxonomy of methods that aim to promote population diversity. Authors eager to include an analysis of this type in their research works should conduct a quantitative experimental study to justify the kind of statements that we have already mentioned. This can be achieved either by using evolutionary operators that explicitly enhance this balance [81–83] or by updating the algorithm or the operators to take into account some kind of indirect measure [84,85].

5.3. *Components analysis and simplicity/complexity*

It is common in recent literature, especially in those papers where the proposal is evaluated on a well-known benchmark, that new proposals are built upon previous existing algorithms. Those new methods normally i) improve previous algorithms by updating or adding new characteristics to their baseline search procedure; or ii) combine existing methods to create a hybrid algorithm of some kind. However, few of these works analyze individually each of the improvements/components of the new proposal. This is an important issue from an algorithmic design perspective. It is true that *powerful* algorithms are usually sought (in terms of their ability to find solutions as close as possible to the global optimum). But it is not less true that *simplicity* should be considered as another preferential aspect in the design of new optimization techniques. Simplicity in algorithmic design has a number of advantages:

- Simple algorithms normally have less parameters to adjust (or, at least, less sensitive ones).
- Their behavior is more predictable, as there are less components involved.
- They can be described and implemented more easily.
- It is less likely that the algorithm overfits one particular benchmark.

All these reasons are important enough to pay attention to the complexity of the new algorithm. For this reason, it is mandatory to provide an in-depth analysis of the contribution of each of the components of the new method to its overall performance. Every change or addition on top of the original algorithm must be supported by a significant contribution to the improved behavior of the novel method. Furthermore, if this contribution is shown to be small, this improvement should be considered for removal in the interest of simplicity. In this sense, we encourage the authors to use the same statistical validation methods described in Section 4.1 to compare each of the individual components of the algorithm. This is the same procedure that we recommend to compare the proposal with other state-of-the-art algorithms (see Section 7).

An illustrative example in this direction is [86]. This work analyzes one of the best performing algorithms of the IEEE CEC'2016 competition on real-parameter single objective optimization, namely, L-SHADE-EpSin [87]. One of the conclusions of this analysis is that only one of the multiple additions to the base L-SHADE algorithm provides some significant improvement in the results (the initialization of the F parameter to 0.5 during the first half of the search). The other modifications (materialized through the inclusion of several local search strategies) were found no significantly better. Moreover, they favored a bias in the search towards solutions around the origin of the search space, as also buttressed by [86]. This means that, even for competitive algorithms, the contribution of each component should be carefully evaluated. This is so, since a simplified version of the algorithm will always be easier to maintain, and can even lead to better results.

A second example aligned with our recommendations at this point emerges from the results of the MOS-SOCO2011 algorithm presented in [79]. By virtue of the Multiple Offspring Sampling (MOS) framework, this optimization technique combines two well-known algorithms: DE

Table 2
Results for MOS-SOCO2011, DE and MTS-LS1 over different 1000-D functions [79].

Benchmark function	MOS-SOCO2011	DE	MTS-LS1
Sphere	0.00e+00	3.71e+01	1.15e-11
Schwefel 2.21	4.25e-01	1.63e+02	2.25e-02
Rosenbrock	6.15e+01	1.59e+05	2.10e+02
Rastrigin	0.00e+00	3.47e+01	1.15e-11
Griewank	0.00e+00	7.36e-01	3.55e-03
Ackley	0.00e+00	8.70e-01	1.24e-11
Schwefel 2.22	0.00e+00	0.00e+00	0.00e+00
Schwefel 1.2	1.94e+05	3.15e+05	1.23e+05
Extended f10	0.00e+00	6.26e-02	1.99e+03
Bohachevsky	0.00e+00	1.67e-01	0.00e+00
Schaffer	0.00e+00	4.42e-02	1.99e+03
f12	0.00e+00	2.58e+01	5.02e+02
f13	8.80e+01	8.24e+04	8.87e+02
f14	0.00e+00	2.39e+01	2.23e+03
f15	0.00e+00	2.11e-01	0.00e+00
f16	0.00e+00	1.83e+01	1.00e+03
f17	2.25e+01	1.76e+05	1.56e+03
f18	0.00e+00	7.55e+00	1.21e+03
f19	0.00e+00	2.51e-01	0.00e+00
Solved functions	14	1	4

and the first local search method (MTS-LS1) of the Multiple Trajectory Search (MTS) algorithm [88]. The MOS-SOCO2011 hybrid algorithm was evaluated on the benchmark proposed for the Soft Computing Special Issue on the scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems [89]. MOS-SOCO2011 obtained the best overall results among all the participants in the special issue. In [79], authors reported not only the results for the proposed MOS-SOCO2011 algorithm, but also those of each of the independent components, DE and MTS-LS1, which are shown in Table 2. This threefold compilation of results allows for a direct comparison on the number of functions solved to the maximum precision (14, 1 and 4 for MOS-SOCO2011, DE and MTS-LS1, respectively), and sheds light on the synergy of both algorithms: except for functions Schwefel 2.21 and Schwefel 1.2, for which the MTS-LS1 algorithm obtained the best results, the hybrid method was able to reach the performance of the best one of its composing algorithms, normally outperforming them.

Additionally, a statistical comparison was also carried out, reporting significant p-values for both comparisons (MOS-SOCO2011 versus DE, and MOS-SOCO2011 versus MTS-LS1). These two comparisons altogether provide enough confidence to support the superiority of the hybrid method over each of its composing algorithms. This is not the only example of such a comparison. Authors of the L-SHADE algorithm follow a similar approach in [90], comparing the new version of their algorithm to previous ones, and evaluating the addition of new components to prove their benefits.

5.4. Parameter tuning and analysis

One important problem in the design of an algorithm is the number of free parameters that can be adjusted to modify its behavior. In general, the more flexibility, the more control parameters to adjust. Very often, the values selected for these parameters are so determinant in the search that even a well designed algorithm can yield bad results with the wrong parameter values. As a consequence, the selection of the values for these internal parameters is a critical decision that should not be underestimated nor overseen.

Unfortunately, the selection of the right parameters values or the adaptation mechanism is not an easy task, because most of the times there is not a clear criterion to guide that selection, and it could be considered an optimization problem itself. Parameter tuning can be tackled in different ways:

- **Offline vs. online tuning**. *Offline* tuning approaches search the optimal EA parameter settings for several representative problems. Then, these values are applied to new problems [91], which means that the tuning is carried out before the algorithm is run on the problem to be solved. The main drawback of this approach is that the values obtained do not necessarily need to be optimal for new problems, but, in practice, robust parameters configurations can be obtained. An alternative approach would be to conduct the parameter tuning during the run. This strategy, in theory, can obtain better adapted values to each problem, but it is a more complex method to implement.
- **Static, adaptive and self-adaptive parameter control.** In [92], parameter control approaches (those carried out during the run of the algorithm) are divided into one of the three following categories: deterministic, adaptive and self-adaptive. Deterministic parameter control means that the value of the strategy parameter is updated following some deterministic rules, without feedback from the search. Adaptive parameter control takes place when some feedback from the search process is used to determine the direction and/or the magnitude of the adjustment of the parameters. Finally, self-adaptive parameter control happens when the values of the parameters are embedded within the solution to optimize. Thus the parameter setting is implicit during the search. Several successful examples of self-adaptation are discussed below.

The most usual approach to deal with the tuning problem is to do it offline and experimentally, comparing the results obtained on multiple combinations of the parameters values. However, there are several pitfalls in which a researcher may fall when conducting a manual tuning:

- *Test each parameter on a small number of predefined values, without taking any feedback into consideration*: some works only test extreme values of the parameter(s) to be tuned (either minimum or maximum over its range). Under these circumstances, the number of values to test should be extended to check whether a better result can be obtained with other values over the range of the parameter(s).
- *Tune only a small part of their parameters*: in this case, the values of the remaining parameters are guessed or initialized to fixed values, without analyzing which parameters influence most on the performance of the algorithm. This analysis would eventually justify which parameters should be selected and carefully tuned, but it is rarely done in the literature.
- *Try to tune each parameter independently, keeping the others fixed*: this widely adopted approach poses many problems. The results obtained by varying just one parameter depend largely on the values given to the others, since it is not uncommon that multiple parameters influence on each other. Therefore, all value combinations of parameters should be optimized altogether. While it is true that exploring all the possible combinations can lead to a combinatorial explosion, there are several techniques in the field of experimental design [3] that can alleviate this problem, such as the fractional design [93] or the alternative Taguchi methods [94].
- *Use parameter values tuned by other authors in previous experiments*: these parameters were usually tuned for a different problem, and thus their values might not be appropriate for the problem/benchmark under consideration. Of course, these values can still be used, but they should never be considered to be the optimal ones, as their competitiveness on a different benchmark can not be guaranteed [95,96].
- *Lack of statistical tests in the comparisons to tune the parameters values of the proposal*: due to the stochastic nature of meta-heuristic algorithms, selecting parameter values based on average fitness values is not enough. The same statistic procedures used to compare multiple techniques should be used when comparing different configurations of an algorithm. A choice of the most promising values is at discretion of prospective authors, but the use of statistical tests should be always enforced, no matter which parameter tuning approach is be-

ing carried out. In fact, as shown in [97], the use of statistical tests for this kind of comparisons can be straightforward.

An alternative approach to parameter tuning is the self-adaptation of the parameters [46]. Under this approach, parameters are not given a fixed value. Instead, a process is devised to automatically adjust their value according to the feedback obtained from the optimization process. This type of adaptation mechanism has been successfully applied to DE, which is very sensitive to its parameters [26,98]. The work in [99] showed the convenience of self-adaptive values versus fixed parameter settings. In [49] the first DE with self-adaptive parameters was presented, drawing parameter values by sampling a distribution which mean is updated by considering new solutions entering the population. Since then, new algorithms improving the self-adaptation mechanism [100,101] have been proposed. In [90], the previous self-adaptive parameters were complemented with an adaptive population size that linearly decreases along the search process. In [102] a tuning of parameters was applied, improving further the results of the overall algorithm.

Although adaptive parameters are a clear improvement over fixed parameters, both in terms of ease of usage and robustness, not all the parameters can be self-adjusted in this way (in particular, some more internal parameters). Thus, even in self-adaptive algorithms, there are fixed parameters that must be tuned to improve the results even more (i.e., [102]).

Fortunately, the tuning of parameters can be automatically carried out. There are several useful and consolidated tools of this type with different features [50]. In the following paragraphs we briefly describe some of the state-of-the-art methods:

- Sequential Parameter Optimization, SPO [103], is a heuristic that uses Latin Hypercube Sampling to determine the parameter values keeping the computation cost low.
- Iterated Racing for Automatic Algorithm Configuration, IRACE [52], is an optimization algorithm that uses different runs (races) and statistical testing to identify combinations of parameters that are worse than others, providing an automatic process to optimize a variety of parameters.
- Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters, REVAC [53], is an evolutionary algorithm that uses multi-parent crossover and entropy measures to estimate the relevance of parameters.
- ParamILS [54], is a versatile stochastic local search approach for automated algorithm configuration.

Although I-RACE has yielded better results for us in the past, all the aforementioned alternatives are robust and consolidated tools, so the selection of one tool over the others will depend on the problem features and personal preferences.

To conclude with this issue, we would like to make an additional remark. The tuning of the parameters of a new proposal can have an important impact in the objectivity of the comparison. This situation can occur when the new proposal is the only algorithm which parameters are tuned for the particular benchmark used in the experimentation, whereas reference algorithms use parameters values proposed by their respective authors under different experimental conditions. This will probably mean that those algorithms are not expected to report very good results for the new experimental scenario. In this case, having the parameters of the new proposal tuned for the benchmark considered in the work could give our proposal an unfair advantage over the other ones, generating a bias in favor of the proposed algorithm in the comparison. Ideally, the solution should be to compare tuned versions of all the algorithms [96], but the cost of doing that could become too expensive and computationally unaffordable. However, when the algorithms are compared against standard benchmarks (not defined *ad-hoc* for each paper), this risk is minimized, because all the algorithms were tested under the same experimental conditions.

6. Guideline #4: Why is my algorithm useful?

The final step of a successful proposal is a thorough discussion of the results. This discussion must answer a crucial question: why is my algorithm useful?

The most obvious answer to this question is *"because it outperforms current state-of-the-art methods"*. If the algorithm falls within this first category of proposals, and if this outperforming behavior is validated by principled means (as those shown in this manuscript), the contribution has a clear scientific value and can be contributed to the community in the form of a publication. However, this is not always the case, but it does not mean that the contribution is not significant.

There are a number of reasons to accept a new proposal even if it is unable to outperform the best-so-far algorithms. Nonetheless, under these circumstances it is even more important the discussion of the results. The benefits of adopting the method proposed in such a contribution should be clearly stated and highlighted accordingly. We next discuss on some of the reasons that can be considered enough for a new proposal to be accepted:

- The first of these reasons is the quality of the results. If, as mentioned before, the results clearly outperform current state-of-the-art methods, the authors have a solid argument for their paper to be accepted. Sometimes, it could be enough that the results are particularly good for a subset of the problems, given that this behavior can be identified and characterized. This does not imply that the rest of the guidelines provided in our paper can be neglected. The discussion of the results should be rigorous and the conclusions should be clearly presented, without any ambiguity nor vagueness.
- The second of these reasons is novelty: if a newly proposed algorithm has the potential to evolve and become competitive with current state-of-the-art methods, it should be presented to the community. Nonetheless, special attention should be paid at this point to avoid the problems described by [1,104]: it is absolutely mandatory that, besides the bio-inspired metaphor, the new algorithmic proposal is competitive enough for a set of problems. Furthermore, we firmly advocate for the development of a unified description language for meta-heuristic algorithms, capable of unambiguously describing each of the algorithmic steps of new proposals, leaving aside any metaphorical language. We utterly believe that efforts in this direction should be intensified, building upon the initial postulations established in some recent works [105,106]. Specifically, metaheuristics components (including search operators and algorithmic behavioral patterns) and interfaces between them should be standardized towards objectively assessing similarities and differences between metaphor-based solvers [107]. A novel metaphor is by no means a sufficient guarantee for a significant scientific contribution.
- The third of these reasons is methodological, i.e., the relevance of some of the building blocks of the overall algorithm. A particular algorithm can include a given component (for example, a local optimizer) that can be of relevance even if the algorithm as a whole is not completely competitive with respect to the prevailing literature. A good example supporting this claim can be observed in co-evolutionary frameworks, which usually include a procedure to identify the subcomponents that will be individually co-evolved. In those cases, even if the subcomponent optimizer is not very sophisticated, the co-evolutionary framework can be relevant by itself. In this sense, it is important to select the appropriate framework to highlight the desired characteristic of the proposed algorithm, as discussed in Section 3.1. Following the same example of subcomponents identification, a researcher focused on large-scale global optimization should consider the CEC'2013 benchmark that explicitly studies this issue [23]. Nevertheless, this is a quite subjective consideration, so authors should clearly highlight these benefits to avoid debatable claims.

7. Case studies

In order to exemplify the application of the previous guidelines, this section elaborates on several case studies that follow our proposed methodology. In particular, we consider the following two scenarios:

- The study of different existing algorithms, wherein the goal is to analyze the advantages and drawbacks of each one of them. In this first scenario, the priority is to compare all the methods following different criteria, towards determining under which circumstances each of them can be recommended.
- The proposal of a new algorithm, in which the priority is to provide informed evidence of the competitiveness of the new proposal, as well as its advantages over previous algorithms.

The structure of this section conforms to the above scenarios. First, in Subsection 7.1 we present several recent bio-inspired algorithms for real-parameter optimization, and subsequently we resort to our proposed guidelines so as to compare them fairly. Then, in Subsection 7.2 we discuss on the second case of study, where a new algorithm is proposed for a specific type of optimization problem: large-scale global optimization.

7.1. Several modern bio-inspired algorithms for real-parameter optimization

In this first case study we simulate the scenario in which a comparison of several bio-inspired algorithms is designed for real-parameter optimization. There are two possible reasons for which this comparison can be tackled. On the one hand, we might be willing to propose a new solver, for which we must assess the performance of existing algorithms to use them as a reference for our proposal. On the other hand, we might be interested in solving a particular problem similar to the ones considered in the comparison, for which we analyze several algorithmic options in order to ascertain which one to use.

The algorithms considered in this first case study are recent methods presented in top-tier journals:

Squirrel Search algorithm (SSA): This is a bio-inspired algorithm that imitates the foraging behavior of squirrels. It divides the solutions into three groups based on their fitness (the best one, the three next best ones, and the remaining ones), and adopts different criteria to combine them considering this grouping arrangement, combining each solution with a solution in a superior category by a random lineal combination [108].

Gaining-sharing knowledge based algorithm (GSKA): This is an algorithm inspired by the human behavior when sharing knowledge. It has been observed that there is more acceptation in the early phases, and that people get more questioning in later stages. For an algorithmic point of view, it uses two different criterion to optimize each dimension: in the first one, referred to as *junior gaining-sharing*, the variable at hand is updated considering the variables of its immediate better and worse individuals in the population. In the second criterion, *senior gaining-sharing*, the variable is updated considering the best and worst individuals in the population. Initially all variables are updated by the junior gaining-sharing mechanism, and during the search, an increasing number of variables are updated by the senior gaining-sharing criterion [109].

Artificial Ecosystem-Based Optimization (AEO): This is a nature-inspired meta-heuristic that finds its motivation in the flow of energy through an ecosystem. In this proposal, the population is updated by means of an iterative process composed by different phases. The first phase is *production*, in which one solution undergoes a small and decreasing random update. The second one is *consumption*, in which every individual is randomly classified as *herbivore, omnivore*, or *carnivore*. Depending on its category, the individual is updated as per a different rule. Newly produced solutions are inserted into the population if they improve the previous ones (replacing them). In the final phase (*decomposition*), each individual is mutated by a dispersion equation [110].

Enhanced LSHADE-SPACMA Algorithm (ELSHADE-SPACMA): This is a new hybrid algorithm that alternates, at each iteration, i) the application of a DE strategy that considers, in its mutation operator, not only the best individuals but also the worst ones [111], ii) with an improved LSHADE-SPACMA [112] that adapts its p parameter to enforce exploitation in its final stages [113]. LSHADE-SPACMA is a hybrid algorithm that applies, with a certain probability, one of the well-known algorithms L-SHADE [90] and CMA-ES [14]. The probability of applying each algorithm is adapted by considering the improvements obtained by each of them over the search. In its paper, ELSHADE-SPACMA is stated to outperform previous winners of real-parameter optimization competitions.

Following our suggestions, we have used existing implementations of the algorithms, in particular, for AEO and GSKA we have used the Mealpy software [9]. For SSA we have used the implementation of the Indago project [10]. Finally, for ELSHADE-SPACMA we have used the source code provided by the authors of the algorithm [11].

In the comparison study discussed throughout this section, we follow the guidelines proposed in this work so as to properly conduct the experiments with these algorithms and other reference methods. During this study, we also discuss the advantages of our methodological proposal. We next start by the first guideline.

7.1.1. Selecting the benchmark as per Guideline #1

First, we must choose an adequate benchmark for measuring the performance of the considered algorithms. Following the recommendations of Guideline #1 described in Section 3:

- We must properly select the benchmark: without any unexpected bias, with the right level of complexity, and for the type of problem addressed by the algorithm.
- We must enforce the usage of a standard benchmark that fulfills the previous requirements, considering the characteristics of the problems for which the algorithms were originally designed.

In this case, all these algorithms have been especially designed for real-parameter optimization. Fortunately, there are several benchmarks well-designed for this type of problems, known to avoid unexpected bias, and endowed with different levels of complexity. In particular, several benchmarks have been proposed within the context of real-parameter optimization, both in the IEEE CEC and GECCO competitions. For our case study, we embrace two of these benchmarks: the CEC'2017 [19] and the COCO benchmarks [55], for the following reasons:

- There are several benchmarks, and all of them are equally good choices. However, some of the benchmarks are rather complementary as they follow different approaches. The real-parameter CEC benchmarks are more focused on proposing difficult objective functions and measuring the final error obtained by each algorithm. On the other hand, the COCO benchmark is made up of simpler functions to allow measuring the performance of an algorithm from different perspectives: how many problems each algorithm is able to solve, how many objective function evaluations every compared solver requires for solving each problem, and other aspects alike.
- When the objective is to compare several algorithms and to analyze them fairly, it is suitable to have several sources of information that allow us to confirm our insights and compare the algorithms from different points of view.

7.1.2. Selecting the performance measure as per Guideline #1

Another important decision to make is the choice of an adequate performance measure. As discussed in Guideline #1, although the final

[9] https://github.com/thieu1995/mealpy

[10] https://pypi.org/project/Indago/

[11] https://sites.google.com/view/optimization-project/files

value of the fitness error is a popular choice, it is not the only alternative to compare the performance of several algorithms. Another interesting option is, for example, the analysis of the evolution of the fitness value as the algorithms in the benchmark iterate to solve each problem.

In our case, the two selected benchmarks are focused on different yet complementary measures:

- In the CEC'2017 benchmark, the error in reported for different milestones, with different ratios of evaluations: 1%, 2%, 3%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100%. In our study we aim at showing the evolution of the algorithms as the number of function evaluations increases. Consequently, we select several of these milestones when reporting the results attained by each algorithm in the comparison.
- Additionally, we pay special attention to the 100% milestone towards providing a final comparison score of the results, without downplaying the importance of reaching competitive fitness values with fewer evaluations.
- In the COCO benchmark, the performance is measured in a different fashion. In particular, the ratio of functions for which the optimum has been reached is computed as the number of objective function evaluations increases.

We consider that with these three measures we will obtain a relatively complete view of the performance of the different algorithms included in our study.

Finally, the results are gathered for every function, independently. However, for a more general analysis, we are also going to use aggregation, considering the *average ranking*, which is calculated by sorting the algorithms for each function based on their error (assigning lower rankings to better algorithms). Then, the average ranking is computed so that an algorithm with a lower average ranking value is declared to perform better, on average, than another one with a higher ranking value.

7.1.3. Selecting the reference algorithms as per Guideline #1

In order to conduct a fair comparison, a clear criterion is needed to select other algorithms to be included in the benchmark, grounded on the need for assessing the convenience of the considered solvers with regards to its competitive performance when compared with prevailing methods. Following our guidelines, we should:

- *Compare against reference algorithms*: The idea is to select a well-known method to compare whether the algorithms perform competitively against it. For this purpose we have considered a classic algorithm: Particle Swarm Optimization (PSO).
- *Compare against similar algorithms*: this decision is especially relevant when an new algorithm is proposed based on particular methods or with well-defined characteristics that resemble those of other existing algorithms. However, in this use case the algorithms to be evaluated have very different sources of inspiration and different algorithmic behavioral patterns. Thus, we consider that there is no need for selecting other similar algorithms in addition to PSO.
- *Compare against competitive algorithms*: this is a hard decision to make, since it is often difficult to scrutinize the entire state-of-the-art related to the optimization problem/algorithm/benchmark under consideration. However, since the considered benchmarks were used in several competitions, we can easily find out competitive algorithms for each of them. In particular, the COCO benchmark tools[12] always compare against a previous winner of the competition. In the case of the CEC'2017 benchmark, we compare against the two best algorithms of the competition: jSO [114] and EBOwithCMAR [115].

To summarize, in this case study we compare each algorithm: (1) to each other; (2) to a well-know reference algorithm (PSO); and (3) to three competitive algorithms (jSO and EBOwithCMAR for the CEC'2017 benchmark, and the winner of 2009 for the COCO benchmark).

Table 3
Average ranking of the algorithms for each dimension.

Algorithm	D10	D30	D50	D100	Mean
ELSHADE-SPACMA	2.667	2.133	**1.783**	**1.383**	**1.9915**
EBOwithCMAR	**1.933**	**1.917**	2.167	2.367	2.0960
jSO	2.317	2.050	2.117	2.583	2.2667
GSKA	3.883	4.200	4.433	4.533	4.2623
PSO	5.367	5.867	6.000	6.400	5.9085
SSA	5.733	5.667	5.333	4.733	5.3665
AEO	6.100	6.167	6.167	6.000	6.1085

7.1.4. Experimenting and validating the results as per Guideline #2

Once the design of the experimentation is set up, the actual experiments are carried out, and results are validated. Following the recommendations on the use of statistical validation provided in Guideline #2 (see Section 4.1), normality and homocedasticity should be checked before the appropriate statistical test can be selected. However, in [41] the results on a previous similar benchmark were analyzed for this two hypotheses. Such results clearly indicated than none of the assumptions were satisfied. For this reason, we have decided to use non-parametric tests. Furthermore, as also suggested by Guideline #2, we have also applied Bayesian tests to compare the best performing approaches to each other.

CEC'2017 benchmark

The first step suggested in Guideline #2 is to calculate the average ranking of the algorithms, followed by non-parametric hypothesis testing. In order to realize this comparison, we resort to Tacolab [116][13], a web tool that eases the comparison of algorithms with different criteria.

Table 3 shows the average ranking for the CEC'2017 benchmark. Several observations can be made over the results in this table:

- Only ELSHADE-SPACMA performs best than previous winners: jSO and EBOwithCMAR. This is particularly the case for higher dimension problems (D50 and D100). None of the other recently proposed algorithms are competitive against these older reference algorithms. This is an interesting result, since the papers in which these new algorithms were first proposed do not compare against state-of-the-art methods. Instead, classic algorithms are just considered. This is even the case of GSKA, which was tested over the CEC'2017 benchmark, but did not include jSO nor EBOwithCMAR in the experiments.
- Among the recent proposals, ELSHADE-SPACMA clearly performs best, even better than previous winners: jSO and EBOwithCMAR.
- Apart from the aforementioned ones (ELSHADE-SPACMA, EBOwithCMAR and jSO), the proposal with the best average performance in this benchmark is GSKA, followed by SSA. We must highlight that these algorithms do not result from the hybridization of previous ones.
- The reference algorithm PSO obtains worse results than most of the algorithms, except for AEO.

A full picture of these results can be displayed if we also measure how the performance of the different algorithms evolves during the search. Fortunately, the experimental conditions of the benchmark require that the error must be measured at different milestones: 1%, 2%, 3%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100% of the fitness evaluations.

Tables 4, 5, 6 and 7 show the average ranking of each method at these different milestones for dimensions 10, 30, 50, and 100, respectively. From these tables the following conclusions can be drawn:

- For dimensions 30 and 50, GSKA is a much faster algorithm and it is clearly better than all the other proposals up to 10% of the

[12] Available at: https://github.com/numbbo/coco

[13] Tacolab website: https://tacolab.org/

Table 4
Evolution of the average ranking with regards to fitness evaluations in the CEC'2017 benchmark (dimension 10).

Algorithm	1	2	3	5	10	20	30
AEO	**1.70**	**1.97**	2.60	3.80	4.97	5.47	5.80
EBOwithCMAR	4.50	4.50	4.03	3.13	**2.30**	**1.90**	2.05
ELSHADE-SPACMA	5.87	5.60	5.20	4.13	3.23	2.27	**1.75**
GSKA	2.83	2.20	**2.20**	**2.37**	2.53	3.27	3.70
PSO	6.20	6.40	6.47	6.47	6.27	6.17	5.93
SSA	1.87	3.17	3.97	4.67	5.20	5.43	5.47
jSO	5.03	4.17	3.53	3.43	3.50	3.50	3.30

(a) 1–30% evaluations

Algorithm	40	50	60	70	80	90	100
AEO	5.93	6.03	6.03	6.07	6.13	6.10	6.10
EBOwithCMAR	**1.88**	**1.80**	**1.85**	**1.85**	**1.73**	**1.88**	**1.93**
ELSHADE-SPACMA	1.88	2.17	2.25	2.37	2.58	2.65	2.67
GSKA	4.00	3.97	4.07	3.98	3.93	3.93	3.88
PSO	5.77	5.70	5.57	5.47	5.47	5.43	5.37
SSA	5.60	5.63	5.63	5.70	5.67	5.70	5.73
jSO	2.93	2.70	2.60	2.57	2.48	2.30	2.32

(b) 40–100% evaluations

Table 5
Evolution of the average ranking with regards to fitness evaluations in the CEC'2017 benchmark (dimension 30).

Algorithm	1	2	3	5	10	20	
AEO	2.33	3.70	4.53	5.10	5.57	6.00	6.10
EBOwithCMAR	5.27	4.67	4.03	3.47	2.77	**1.90**	2.07
ELSHADE-SPACMA	6.53	5.63	5.27	4.17	2.83	1.93	**1.70**
GSKA	**1.63**	**1.40**	**1.40**	**1.43**	**2.10**	3.20	3.97
PSO	5.33	6.07	6.37	6.50	6.30	6.17	6.07
SSA	2.30	2.87	3.20	3.57	4.57	5.07	5.20
jSO	4.60	3.67	3.20	3.77	3.87	3.73	2.90

(a) 1–30% evaluations

Algorithm	40	40	60	70	80	90	100
AEO	6.20	6.27	6.27	6.27	6.20	6.17	6.17
EBOwithCMAR	2.18	2.37	2.43	2.32	1.92	1.92	**1.92**
ELSHADE-SPACMA	**1.52**	**1.40**	**1.38**	**1.45**	**1.72**	**1.83**	2.13
GSKA	4.10	4.20	4.17	4.17	4.17	4.17	4.20
PSO	6.03	5.97	5.97	5.97	5.93	5.93	5.87
SSA	5.27	5.40	5.43	5.47	5.60	5.63	5.67
jSO	2.70	2.40	2.35	2.37	2.47	2.35	2.05

(b) 40–100% Evaluations

fitness evaluations, including the previous winners of CEC competitions (EBOwithCMAR and jSO) and the most competitive recent algorithm (ELSHADE-SPACMA).

- EBOwithCMAR is the algorithm with the best results for dimensions 10 and 30 (very close to ELSHADE-SPACMA in those dimensions), whereas ELSHADE-SPACMA is the best performing algorithm for dimensions 50 and 100, closely followed by EBOwithCMAR and jSO.
- For dimensions 30, 50 and 100, ELSHADE-SPACMA is the best one since the 40% of the budget of evaluations. Although Table 3 indicates that EBOwithCMAR obtains the best final results, ELSHADE-SPACMA achieves better results during most of the search, being only improved by the former at the end.
- Deciding which algorithm should be applied to a specific problem strongly depends on the effort that can be devoted to the search. In this benchmark, GSKA is better when less evaluations are allowed, whereas ELSHADE-SPACMA is preferred when a higher number of evaluations can be afforded.

Additionally, following Guideline #2 (Section 4) we have conducted a statistical validation of the results to reject the hypothesis that the differences observed in the performance of the algorithms is due to their stochastic nature and not to actual differences in their performance. First, we use the Friedman rank-sum test to find out if significant differences can be found among all the algorithms. The p-values reported by this test are 3.61e-08, 1.51e-07, 7.58e-7, and 9.62e-10 for dimensions 10, 30, 50, and 100, respectively. Since all the p-values values are clearly lower than the $\alpha = 0.05$ confidence level, we can state that differences among the algorithms exist and that are significant.

Table 6
Evolution of the average ranking with regards to fitness evaluations in the CEC'2017 benchmark (dimension 50).

Algorithm	1	2	3	5	10	20	
AEO	2.97	4.20	4.90	5.47	5.77	5.97	6.03
EBOwithCMAR	5.77	4.57	4.17	3.67	2.77	2.27	2.40
ELSHADE-SPACMA	6.57	5.90	5.43	4.37	3.10	**2.07**	**1.93**
GSKA	**1.67**	**1.83**	**1.83**	**1.80**	**2.60**	3.70	4.17
PSO	4.70	5.90	6.17	6.40	6.40	6.23	6.23
SSA	1.90	2.17	2.37	2.73	3.63	4.40	4.73
jSO	4.43	3.43	3.13	3.57	3.73	3.37	2.50

(a) 1–30% evaluations

Algorithm	40	50	60	70	80	90	100
AEO	6.07	6.10	6.10	6.17	6.23	6.20	6.17
EBOwithCMAR	2.47	2.50	2.57	2.50	2.23	2.27	2.17
ELSHADE-SPACMA	**1.77**	**1.37**	**1.23**	**1.20**	**1.30**	**1.38**	**1.78**
GSKA	4.30	4.33	4.33	4.40	4.40	4.40	4.43
PSO	6.13	6.10	6.07	6.03	6.03	6.00	6.00
SSA	4.93	5.03	5.07	5.13	5.27	5.33	5.33
jSO	2.33	2.57	2.63	2.57	2.53	2.42	2.12

(b) 40–100% evaluations

Table 7
Evolution of the average ranking with regards to fitness evaluations in the CEC'2017 benchmark (dimension 100).

Algorithm	1	2	3	5	10	20	30
AEO	3.40	4.57	5.13	5.50	5.73	5.83	5.80
EBOwithCMAR	5.83	4.40	4.10	3.90	3.10	2.53	2.70
ELSHADE-SPACMA	6.40	6.10	5.33	4.47	3.70	2.93	2.43
GSKA	1.83	**1.93**	**1.97**	**2.17**	**2.77**	4.07	4.13
PSO	4.43	5.77	6.30	6.40	6.40	6.40	6.43
SSA	**1.63**	1.97	2.07	2.50	2.97	3.77	4.17
jSO	4.47	3.27	3.10	3.07	3.33	**2.47**	**2.33**

(a) 1–30% evaluations

Algorithm	40	50	60	70	80	90	100
AEO	5.90	5.97	6.00	6.03	6.07	6.07	6.00
EBOwithCMAR	2.67	2.50	2.53	2.43	2.13	2.23	2.37
ELSHADE-SPACMA	**2.00**	**1.43**	**1.33**	**1.40**	**1.47**	**1.43**	**1.38**
GSKA	4.33	4.43	4.43	4.47	4.50	4.50	4.53
PSO	6.43	6.40	6.40	6.40	6.37	6.37	6.40
SSA	4.33	4.53	4.63	4.63	4.67	4.73	4.73
jSO	2.33	2.73	2.67	2.63	2.80	2.67	2.58

(b) 40–100% evaluations

Table 8
Statistical validation for the CEC'2017 benchmark and dimension 10 (EBOwithCMAR is the control algorithm).

EBOwithCMAR versus	Wilcoxon p-value	Wilcoxon p-value*
AEO	2.702e-06	1.621e-05 $\surd$
SSA	3.703e-06	1.852e-05 $\surd$
PSO	4.110e-06	1.852e-05 $\surd$
GSKA	4.374e-05	1.312e-04 $\surd$
ELSHADE-SPACMA	0.027	0.055
jSO	0.225	0.225

$\surd$: statistical differences exist with significance level $\alpha = 0.05$.
*: p-value corrected with the Holm procedure.

Once that significant differences are detected, we proceed with a multiple comparison, as the use of the Holm procedure keeps the family-wise error rate under control. Tables 8, 9, 10, 11 summarize the results for dimensions 10, 30, 50 and 100, respectively. Inspecting these results we arrive at the following insights:

- EBOwithCMAR is the best algorithm for dimensions 10 and 50, whereas ELSHADE-SPACMA is the best algorithm for dimensions 50 and 100.

Table 9
Statistical validation for the CEC'2017 benchmark and dimension 30 (EBOwithCMAR is the control algorithm).

EBOwithCMAR versus	Wilcoxon p-value	Wilcoxon p-value*
AEO	2.702e-06	1.621e-06 √
PSO	2.702e-06	1.621e-06 √
SSA	2.702e-06	1.621e-06 √
GSKA	4.110e-06	1.621e-06 √
ELSHADE-SPACMA	0.226	0.452
jSO	0.431	0.452

√: statistical differences exist with significance level $\alpha = 0.05$.
*: p-value corrected with the Holm procedure.

Table 10
Statistical validation for the CEC'2017 benchmark and dimension 50 (ELSHADE-SPACMA is the control algorithm).

ELSHADE-SPACMA versus	Wilcoxon p-value	Wilcoxon p-value*
AEO	1.863e-09	1.118e-09 √
GSKA	1.863e-09	1.118e-09 √
PSO	1.863e-09	1.118e-09 √
SSA	1.863e-09	1.118e-09 √
jSO	0.054	0.107
EBOwithCMAR	0.509	0.509

√: statistical differences exist with significance level $\alpha = 0.05$. *: p-value corrected with the Holm procedure.

Table 11
Statistical validation for the CEC'2017 benchmark and dimension 100 (ELSHADE-SPACMA is the control algorithm).

ELSHADE-SPACMA versus	Wilcoxon p-value	Wilcoxon p-value*
AEO	3.725e-09	2.235e-08 √
GSKA	3.725e-09	2.235e-08 √
PSO	3.725e-09	2.235e-08 √
SSA	5.588e-09	2.235e-08 √
jSO	2.254e-05	4.507e-05 √
EBOwithCMAR	3.128e-04	3.128e-04 √

√: statistical differences exist with significance level $\alpha = 0.05$. *: p-value corrected with the Holm procedure.

- Most of the new proposals (AEO, GSKA, and SSA) are statistically worse than competitive algorithms proposed back in 2017 (EBOwithCMAR) and the newest proposal ELSHADE-SPACMA.
- For dimensions 10, 30, and 50, there are no significant differences among EBOwithCMAR, ELSHADE-SPACMA and jSO. However, for dimension 100 the recent algorithm ELSHADE-SPACMA is statistically better than them.

As stated in Guideline #2, more statistical tests beyond those used so far can be applied to reinforce our conclusions on the case study. We next discuss on the results of two new tests utilized for this end: the critical distance, and Bayesian tests.

Fig. 4 shows the statistical relevance of the differences between the average rankings over the CEC'2017 benchmark. The Critical Distance (CD) value (given by a Nemenyi post-hoc test at a significance $\alpha = 0.05$) indicates the minimal absolute distance between two average rankings to be declared as statistically different to each other. It can be observed that among the new proposals under comparison, only ELSHADE-SPACMA is competitive with respect to the best performing approaches, though GSKA is also competitive for small problems. When the dimensionality increases, the distance between GSKA and the best algorithms (ELSHADE-SPACMA, EBOwithCMAR and jSO) increases, and the ranking difference with SSA is reduced.

In dimension 10 the gaps between the ranks of the different algorithms become narrower. In order to arrive at more insightful conclusions, we apply several Bayesian tests over the results obtained for this dimensionality value, visualizing the adjusted Bayesian probability in barycentric coordinates. Bayesian analysis performed over the results of selected pairs of algorithms yields the probability that one solver outperforms another, based on the objective function values obtained by each of them over all runs and problems of the benchmark. The computed probability distribution displayed in barycentric coordinates after Monte Carlo sampling, depicting three regions: one where the first algorithm outperforms the second; a second one with the converse case (the second outperforms the first); and a region of practical equivalence where the results attained by each algorithm can be considered to be statistically equivalent to each other. To decide on this equivalence, a parameter called *rope* indicates the minimum difference between the scores of both methods for them to be considered significantly different to each other.

Table 12
Average ranking of the most competitive algorithms for each dimension.

Algorithm	D10	D30	D50	D100	Mean
EBOwithCMAR	**1.683**	**1.817**	2.100	2.300	1.9750
ELSHADE-SPACMA	2.267	2.133	**1.783**	**1.250**	**1.858**
jSO	2.050	2.050	2.117	2.450	2.1667

At this point it is important to highlight the fact that *rope* denotes a threshold imposed on the absolute difference of fitness values between the two compared algorithms. Consequently, *rope* is interpretable and overcomes acknowledged issues identified around the use of p-values and significance levels in studies resorting to NHST for statistical assessment [74].

Turning back the focus on our case study, we consider several specific pairs of algorithms: (EBOwithCMAR, jSO), (jSO, GSKA) and (SSA, GKSA). We have considered only the first 20 runs for each function to make the processing time of the test computationally affordable. The *rope* value is set to 20. Fig. 5 depicts the Bayesian plots for each of the considered pairs, from which we confirm the following facts:

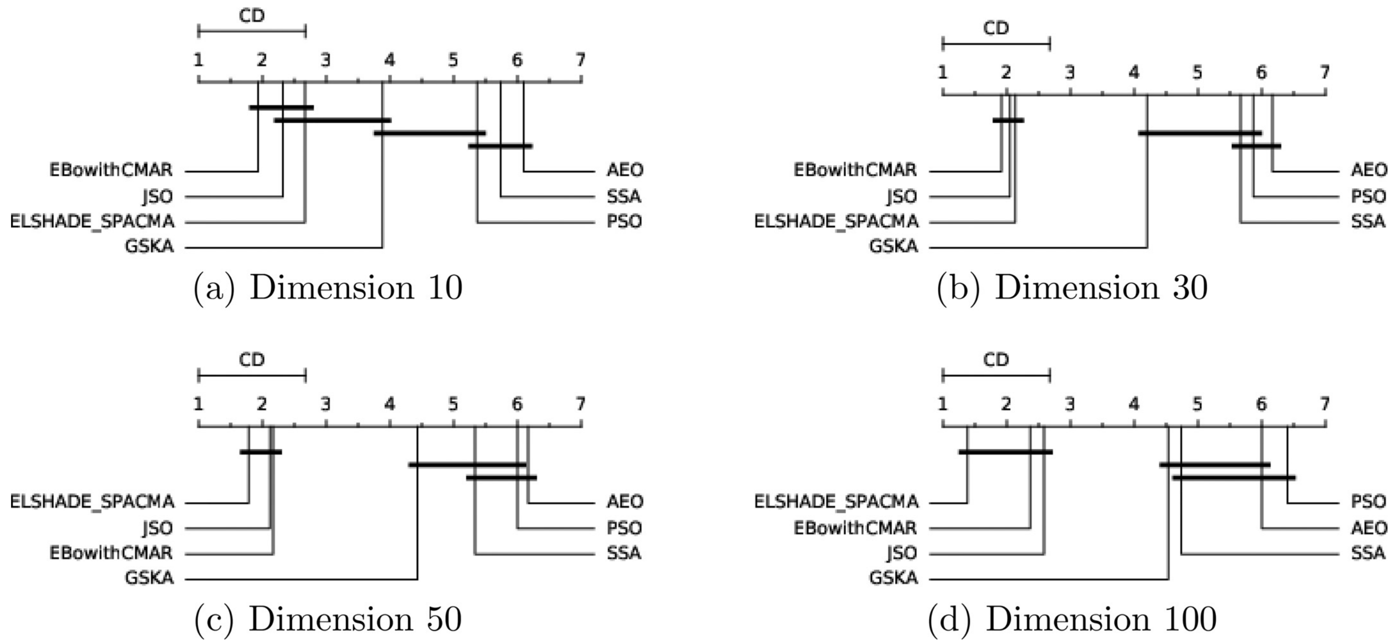
- There are almost no relevant differences among the results attained by ELSHADE-SPACMA, EBOwithCMAR and jSO.
- For dimensions 30, 50, and 100 there are clearly two groups: one made up by EBOwithCMAR, ELSHADE-SPACMA, and jSO; and another one, made up by GSKA, SSA, PSO, and AEO.
- The lack of sampled point in the plots comparing jSO and GSKA reveals that the probability that GSKA achieves better results than jSO is exactly zero. This is specially relevant because the threshold value used is relatively high given the range of the objective functions characterizing the benchmark.
- Differences between SSA and GKSA can be claimed to be significant for dimension 10, with GKSA emerging as the best performing approach.

Results show that ELSHADE-SPACMA outperforms the previous winning algorithms EBOwithCMAR and jSO. Thus it could be considered the new state-of-the-art method for that benchmark. In order to conduct a deeper analysis, our study include direct comparisons of ELSHADE-SPACMA against EBOwithCMAR and jSO. The results of this comparison are shown in Table 12, where it can be observed that among the more competitive algorithms, ELSHADE-SPACMA continues obtaining the best overall results.

Table 13 reports the results of pairwise comparisons among ELSHADE_SPACMA, EBOwithCMAR and jSO with the Wilcoxon test. These results reveal that there are statistical differences only in dimension 100. This finding is confirmed in Table 14, in which we provide the results corrected with the Holm method to account for the family-wise error in dimension 100.

COCO benchmark

We continue our discussions on this first case study with the results over the COCO benchmark. To this end, we first use the source code and tools available at https://github.com/numbbo/coco to run the experi-



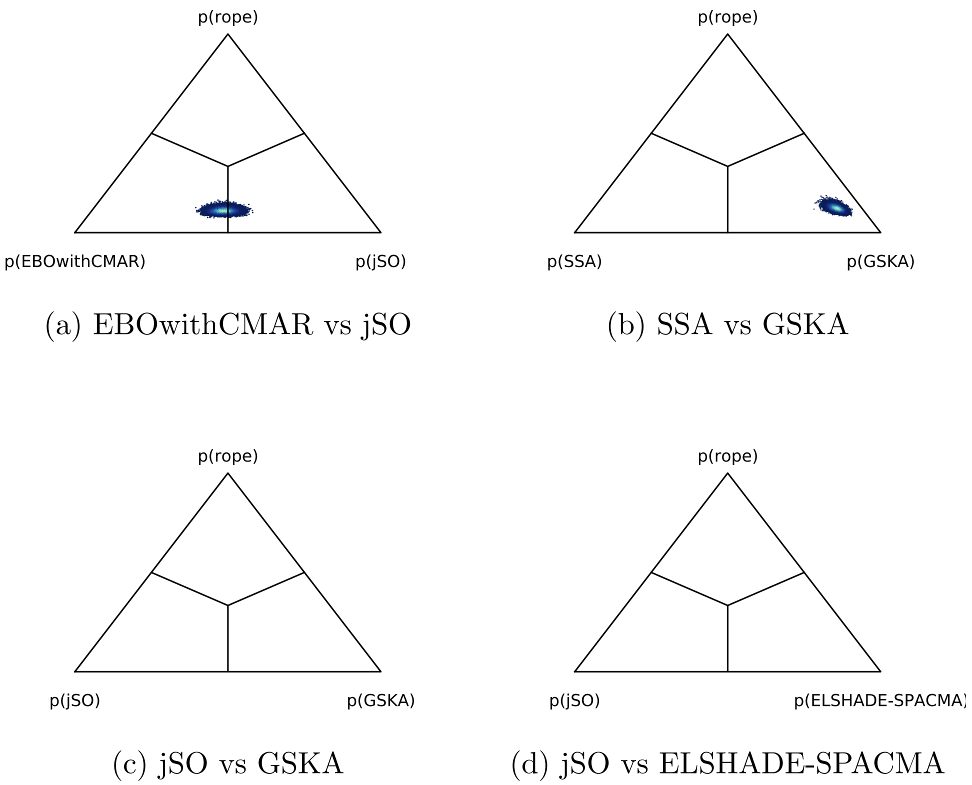
(a) Dimension 10

(b) Dimension 30

(c) Dimension 50

(d) Dimension 100

Fig. 4. Critical distance plots for the CEC'2017 benchmark.



(a) EBOwithCMAR vs jSO

(b) SSA vs GSKA

(c) jSO vs GSKA

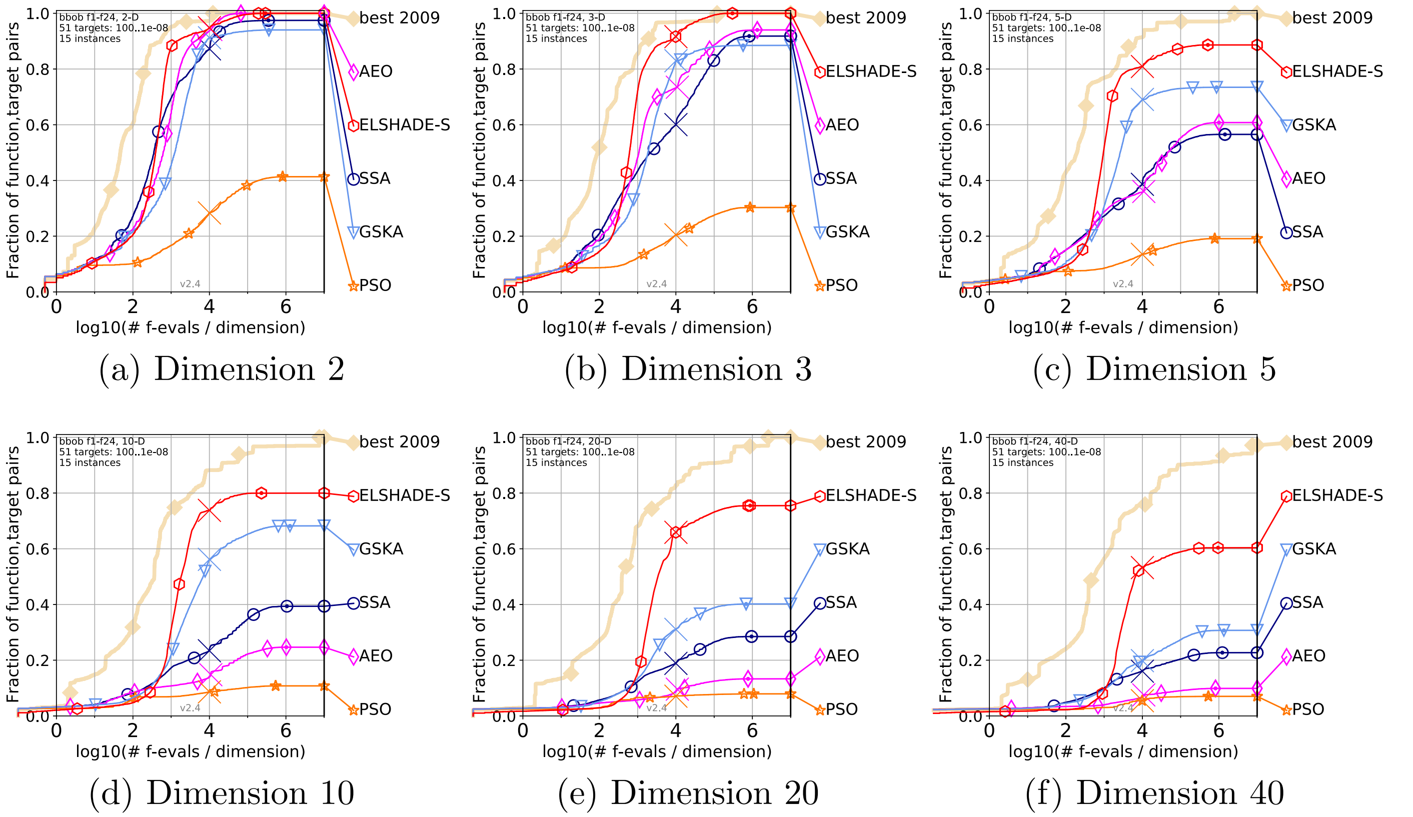
(d) jSO vs ELSHADE-SPACMA

Fig. 5. Bayesian plots for the CEC'2017 benchmark and dimension 10..

ments and obtain results of the performance of every algorithm. Furthermore, these tools allow including in the benchmark the competitive algorithm proposed in 2009 for this benchmark.

Fig. 6 visualizes, for each algorithm, the ratio of solved problems (problems for which the error obtained is lower than a threshold) when the number of evaluations increases. The benchmark is designed for different dimension values, so Fig. 6 includes a subplot for each one of them. It can be observed that:

- For low dimensionality values (e.g. 2 and 5), the results obtained for the other proposals are very similar and competitive, except for the reference algorithm (PSO). However, this behavior changes when the dimensionality increases.
- For dimensionality equal to 5, ELSHADE-SPACMA (the acronym is shortered in Fig. 6) and GSKA are notably better than SSA and AEO. For dimension 10, this noted improvement increases further.



(a) Dimension 2 (b) Dimension 3 (c) Dimension 5

(d) Dimension 10 (e) Dimension 20 (f) Dimension 40

Fig. 6. Ratio of solved problems versus fitness evaluation for the COCO benchmark.

Table 13
Results of the wilcoxon's test for the most competitive algorithms in the CEC'2017 benchmark for different dimension values.

Algorithms	D10	D30	D50	D100
ELSHADE-SPACMA vs EBOwithCMAR	0.027	0.226	0.509	3.128e-04
EBOwithCMAR vs jSO	0.225	0.431	0.054	0.477
ELSHADE-SPACMA vs jSO	0.796	0.706	0.556	2.254e-05

Table 14
Statistical validation for more competitive algorithms for the CEC'2017 benchmark and dimension 100 (ELSHADE-SPACMA is the control algorithm).

ELSHADE-SPACMA versus	Wilcoxon p-value	Wilcoxon p-value*
jSO	2.254e-05	4.507e-05 √
EBOwithCMAR	3.128e-04	3.128e-04 √

√: statistical differences exist with significance level $\alpha = 0.05$. *: p-value corrected with the Holm procedure.

- For dimensionality values 20 and 40, GSKA still performs better than AEO, SSA and GSKA, but the difference reduces dramatically. GSKA exhibits a better behavior for lower dimensionality values.
- For dimensionality values 20 and 40, ELSHADE-SPACMA is the algorithm that scores best when compared to the other proposals by a great gap. However, it does not approach the results of the best algorithm reported in 2009.

Following the same procedure as in the CEC'2017 benchmark, we have also considered whether to apply statistical tests. Unfortunately, the reduced number of runs for each function (15) is too low to allow for statistical tests with minimum significance guarantees.

Conclusions considering both benchmarks

After analyzing together all experimental results discussed previously, our conclusions can be summarized as follows:

- GSKA obtains better results than AEO and SSA.
- For a lower budget in terms of number of objective function evaluations, GSKA is better than the other algorithms. On the contrary, when the number of evaluations is increased, EBOwithCMAR and jSO obtain better results.
- The new proposals (AEO, GSKA and SSA) perform competitively when compared to classic optimization algorithms (e.g. PSO), but none of them can rival modern solvers like the standing winners of renowned competitions.
- GSKA, for small dimensionality values, is better than other compared algorithms, but its advantage is reduced when the dimension increases.
- ELSHADE-SPACMA has shown a rather opposite behavior: it improves as the dimension value increases. It is actually the only algorithm that is able to outperform existing competitive algorithms, specially in higher dimensionality values. For example, for dimension 100 it is even statistically better than the previous winning algorithms. In the COCO benchmark, on the other hand, although it is very competitive, it is not better than the previous winner of the 2009 competition. This could be due to the lower dimensionality values used in this competition.

All in all, the performance of these recent meta-heuristic algorithms can be of interest when compared to that offered by classic approaches, but does not reach the levels of performance that are achieved by modern competitive optimization methods.

7.1.5. Components analysis and Tuning as per Guideline #3

Comparing with just reference and/or state-of-the-art algorithms is not enough. Following Guideline #3 (Section 5), it is important for new

Table 15
Average ranking for different attraction and swarm size values for SSA algorithm and dimension.

		swarm size				
		10	25	50	75	100
attraction	**0.1**	20.40	18.05	17.82	17.35	17.52
	0.3	19.87	15.58	17.02	14.05	13.58
	0.5	19.30	18.38	13.88	12.65	10.25
	0.7	17.80	17.25	13.08	12.55	**8.22**
	0.9	18.87	18.15	14.75	10.82	10.35
	1.0	18.83	18.78	14.82	13.48	11.55

(b) Dimension 10

		swarm size				
		10	25	50	75	100
attraction	**0.1**	19.75	19.68	14.72	13.72	12.55
	0.3	18.75	17.32	15.92	11.95	12.82
	0.5	18.15	19.28	16.48	**9.25**	10.68
	0.7	19.88	15.88	14.32	12.65	**9.35**
	0.9	18.78	18.48	15.22	11.85	12.38
	1.0	20.35	18.68	16.45	15.28	14.42

(b) Dimension 30

		swarm size				
		10	25	50	75	100
attraction	**0.1**	20.12	17.42	16.68	15.08	16.75
	0.3	18.58	15.82	12.02	10.75	13.12
	0.5	19.38	17.65	15.05	13.82	11.25
	0.7	17.55	16.48	12.62	13.25	**10.28**
	0.9	21.25	18.02	15.25	14.08	11.65
	1.0	18.05	19.92	16.68	12.88	13.55

(c) Dimension 50

proposals to analyze the behavior of the algorithm to gauge the contribution of each component. In this use case, since we have several new algorithms and the goal is to compare them fairly, analyzing the different components of each one of them because could entail an extensive study that goes beyond the scope of this work. For a example of component analysis, we refer to Section 7.2.5.

Following Guideline #3, the use of an automatic tuning mechanism is also recommended for the different algorithms in the comparison. In particular, we focus on the new proposal algorithms (AEO, GSKA, and SSA), because winners of the competitions, EBOwithCMAR and jSO, have been previously optimized by their authors for the competition. However, in our case the original publication where the most promising algorithm (GSKA and ELSHADE-SPACMA were proposed already considered the same benchmark (CEC'2017), so the parameter value used in this seminal reference are assumed to be the most adequate ones for this benchmark used in the case study. On the other hand, despite originally evaluated over a different benchmark, AEO has no parameters to be optimized. Its only free parameter (the population size) was analyzed in several works of its authors, concluding that *varying the population size does not result in a significant change in precision* [110]. Therefore, we center our discussion on the second-best performing algorithm in the results (SSA):

Tuning SSA

To illustrate this process, we tune the two parameters driving the search behavior of the SSA approach, namely, the so-called *swarm size* and *attraction factor*, the lower value of the gliding distance d_g. Since there are only two parameters, instead of using a tool such as the ones commented in Subsection 5.4, we have applied a grid search for dimensions 10, 30, and 50 (dimension 100 was omitted for the sake of computational affordability). In this grid search, different values are explored for each parameter:

- For the swarm size, we have tested values **10, 25, 50** (recommended by authors), **75**, and **100**.
- For the attraction factor, we have considered values equal to **0.1, 0.3, 0.5** (recommended by authors), **0.7**, **0.9**, and **1.0**.

As a result of this grid search, the $5 \times 6 = 30$ parameter combinations have given rise to 30 configurations of SSA that we have compared to each other in terms of average rank. Such results are given in Table 15, where we can notice that:

Table 16
Average ranking of the tuned algorithms for each dimension.

Algorithm	D10	D30	D50
ELSHADE-SPACMA	2.667	2.133	**1.783**
EBOwithCMAR	**1.933**	**1.917**	2.167
jSO	2.317	2.050	2.117
GSKA	3.883	4.200	4.433
PSO	5.400	5.933	6.067
SSA (Tuned)	5.533 (–0.40)	5.433 (–0.23)	5.200 (+0.13)
AEO	6.267	6.333	6.233

- The best results are obtained with an attraction factor of 0.7 and swarm size equal to 100 for dimensionality values of 10 and 50. Furthermore, this configuration is the second best for dimension 30. Thus, these values can be declared to be the recommended setting for the algorithm.
- In general, results are better with a larger swarm size, and with a medium-range value of the attraction factor.
- The value recommended by the authors for the attraction value (0.5) is close to the best obtained during the tuning process (0.7). Moreover, it is the best one for dimension 30.
- The recommended value for its authors in swarm size, 50, it is not adequate for this benchmark.

We have concluded that the net SSA results improve via parameter tuning. However, we must assess whether this performance improvement has any impact on the comparison previously discussed in Section 7.1.4. As a global recommendation, parameters of all compared algorithms should be tuned. However, in this case the most competitive algorithms (ELSHADE-SPACMA, EBOwithCMAR, jSO and GSKA) were originally tuned over this benchmark. As a result, the only competitive approach that requires tuning is SSA. The rest of algorithms were not tuned considering its non-competitive results and the limited computational resources.

In Table 16 are shown the results of tuned algorithms, showing for the tuned algorithm (SSA) the difference in the average ranking in comparison with the non-tuned version. Dimension 100 was not included because there was no tuning performed in this dimension. It can be observed that the tuned version of the algorithm achieves an improvement over its non-tuned counterpart for dimensions 10 and 30, and it is worse for dimension 50. However, the improvement is not enough to change the relative position of the algorithm in the ranking.

7.1.6. Justifying the usefulness of the algorithm as per Guideline #4

Following Guideline #4 (Section 6), there are several perspectives from which one may claim the usefulness of an algorithm:

- *Quality of the results*: in this case, the majority of the evaluated algorithms have been found to be not competitive enough in comparison with the *state-of-the-art methods*. Only ELSHADE-SPACMA has been able to improve them, thus becoming the current *state-of-the-art* algorithm in the benchmark considering these results. Also, the comparisons between the different algorithms expose significant differences between them. Therefore, the results can be considered to be interesting and informative for researchers aiming to elaborate further on their design. Furthermore, while GSKA is not competitive against jSO and EBOwithCMAR, it is able to perform best when the number of function evaluations is low. In many real-world problems, the evaluation of a solution can be very costly in terms of computational resources (e.g. when the fitness value is produced by long computer simulations). Under these circumstances, it is essential to rely on algorithms capable of obtaining good results within a small number of evaluations.
- *Technical novelty*: the compared proposals not only have very different biological inspiration, but they differ notably regarding their algorithmic behavior. In particular, ELSHADE-SPACMA, SSA and GSKA propose interesting ideas that could be considered for new algorithms. ELSHADE-SPACMA combines different previous algorithms (SHADE-ILS and CMA-ES) with a novel mechanism to improve the diversity into the population, a mutation operator that not only considers the best solutions but also the worst ones. Another contribution is the adaptation of one parameter used to increase diversity in the early stages of the search. With these two main changes that make it different than the previous LSHADE-SPACMA algorithm, it is able to improve the results of the competitive EBOwithCMAR and jSO. LSHADE-SPACMA failed to do so. On the other hand, SSA exploits the idea of ranking the different solutions to create several categories (the current best, the best ones, and the normal, which is the largest group), and use a different mutation method considering the category of each individual. In addition, solutions which cannot get improved are periodically restarted. These two strategies yield a very straightforward yet effective optimization algorithm. GSKA also suggests other interesting concepts from the technical point of view: it updates the variables of each individual under two possible criteria, among which the selected one is updated during the search to increase the exploitation during the run of the algorithm. Furthermore, in the exploration GSKA uses for each individual the most similar ones in fitness, the immediate better one and the immediate worse one.
- *Methodological contribution*: There is no methodological contribution.
- A special attention should be paid to the *simplicity*, in which SSA outstand in the benchmark. However, modifications should be applied to improve its results and to avoid its apparently premature convergence.

7.1.7. Summary of the use case

To conclude, this first use case follows most of the guidelines of our proposed methodology. The main procedures followed in the use case are highlighted in Fig. 7. In addition, we briefly describe now the main actions taken for each of the proposed guidelines:

- Guideline #1: We have selected two different real-parameter benchmarks, CEC'2017 and COCO. Both are widely accepted by the community working in real-parameter optimization. We have also compared our algorithms against competitive solvers that have won competitions using the same benchmarks, as well as a reference baseline for swarm intelligence (i.e. PSO).
- Guideline #2: following this guideline, we have shown and validated the results according to the good practices described in Section 4, including statistical non-parametric tests and Bayesian analysis. Furthermore, we have provide evidence on how the identification of the best algorithm can strongly be biased by the stopping criterion in use (e.g. the maximum number of objective function evaluations).
- Guideline #3: following the suggestions in this guideline, we have applied a simple tuning process and unveiled that the right parameter values can influence the results and conclusions held from the previous comparison.
- Guideline #4: in this use case, the comparisons between the different algorithms could be justified as per the relevant differences found among them. Only one of the compared algorithms (ELSHADE-SPACMA) has been able to outperform other existing competitive algorithms. Moreover, one of them (GSKA) has shown a superior behavior when the number of function evaluations is low. Nevertheless, the fact that the majority of these modern algorithms could not improve previous competitive solvers is a relevant fact that should stimulate fairer comparison studies in prospective works with new and/or improved versions of bio-inspired optimization algorithms. Finally, some of these algorithms pose innovative algorithmic ideas that should be investigated further towards their use in the design of new search methods.

7.2. SHADE-ILS for large-scale global optimization

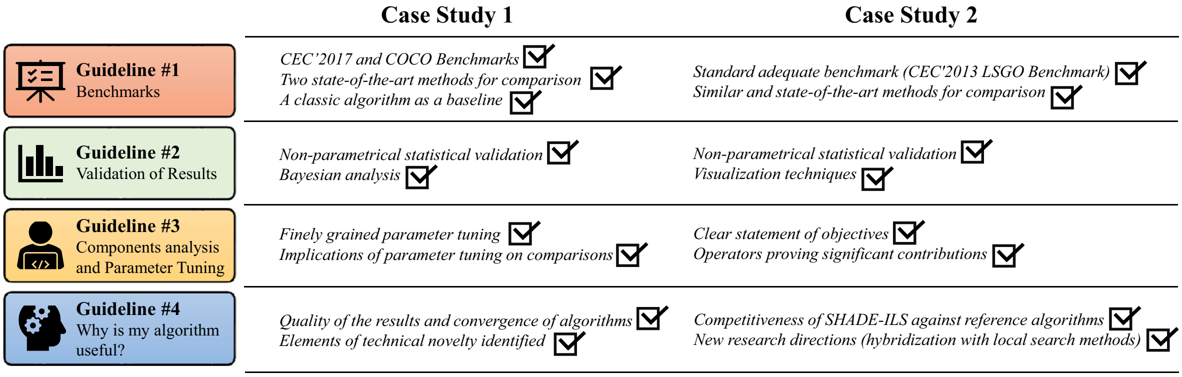
In this second case study we simulate the situation in which we design a new algorithm, SHADE-ILS, specially designed for large-scale global optimization. In this section we follows the guidelines for properly conducting the experiments, comparisons with other reference algorithms, and the analysis to put in value the advantages of our methodological proposal.

7.2.1. Selecting the benchmark as per Guideline #1

First, we have to choose the right benchmark for the experimental assessment of the performance of our newly proposed algorithm. Following the recommendations of Guideline #1 (described in Section 3), we have to:

- Properly select the benchmark: without any unexpected bias, with the right level of complexity, and for the type of problem addressed by the algorithm.
- Enforce the usage of a standard benchmark that fulfills the previous requirements.

The selection of the benchmark cannot be done without considering the proposed algorithm, since it depends on the characteristics of the problem for which the algorithm was implemented (or the type of problems for which we want to test it). In our example, we have designed SHADE-ILS [117], an algorithm specially devised for real-parameter optimization problems that comprise a high number of variables. This family of optimization problems is collectively referred to as large-scale global optimization, for which several benchmarks have been proposed [23,27,89]. If any of them allows for an unbiased comparison, we should use it, avoiding in this way the design of our own benchmark. In particular, our first option is the CEC'2013 benchmark [23], since it is both the most recent and the most popular competition to date. Furthermore, its popularity yields many previous results that we can use for comparison purposes. Nevertheless, before proceeding further we have to verify whether the selected benchmark allows for good comparisons. For this purpose, information and data available about the benchmark should comply with several requirements:



	Case Study 1	Case Study 2
Guideline #1 Benchmarks	*CEC'2017 and COCO Benchmarks* ☑ *Two state-of-the-art methods for comparison* ☑ *A classic algorithm as a baseline* ☑	*Standard adequate benchmark (CEC'2013 LSGO Benchmark)* ☑ *Similar and state-of-the-art methods for comparison* ☑
Guideline #2 Validation of Results	*Non-parametrical statistical validation* ☑ *Bayesian analysis* ☑	*Non-parametrical statistical validation* ☑ *Visualization techniques* ☑
Guideline #3 Components analysis and Parameter Tuning	*Finely grained parameter tuning* ☑ *Implications of parameter tuning on comparisons* ☑	*Clear statement of objectives* ☑ *Operators proving significant contributions* ☑
Guideline #4 Why is my algorithm useful?	*Quality of the results and convergence of algorithms* ☑ *Elements of technical novelty identified* ☑	*Competitiveness of SHADE-ILS against reference algorithms* ☑ *New research directions (hybridization with local search methods)* ☑

Fig. 7. Checklist of the guidelines' recommendations followed by the use cases.

done *Clear experimental conditions*: the experimental setup is well defined, and conditions are set the same for all algorithms.

done *The implementation of the benchmark is openly available*: the CEC'2013 benchmark is specially appropriate in this regard, as implementations of the problems comprising the benchmark are not only made publicly available, but also in several programming languages: C/C++, Matlab, Java, and Python[1]

done *The optima is not at the center of the domain search*: all functions in the chosen benchmark are shifted to guarantee this feature.

wontfix *Functions are rotated*: Although this feature is not present in the chosen benchmark, the importance of this requirement is not as critical as the aforementioned shifting.

done *Presence of local optima*: In the benchmark there are several functions with different local optima. Actually, it is not the only criterion to provide functions with varying levels of difficulty. In particular, in this benchmark there are different degrees of interrelation between variables, which makes sense given the large dimensionality of the problems.

Summarizing, the above analysis concludes that the CEC'2013 benchmark for large-scale global optimization follows most of the requirements imposed by our methodology. This is the reason why we select it as benchmark for the experimentation.

7.2.2. Selecting the performance measure as per Guideline #1

Another important decision to make is the choice of an adequate performance measure. In this regard, we can measure not only the final fitness error (deviation with respect to the global optimum that is known *a priori*), but also the error for different number of fitness evaluations (called the *accuracy level*). This way, we can fairly measure the efficiency of the algorithms. There are two possibilities in this matter: i) to report the performance for each accuracy level; and ii) to provide the performance for the maximum number of fitness evaluations considered in the experiments. To show the results concisely, we will only discuss on the latter of these alternatives (i.e. the results for the maximum number of fitness evaluations). However, the study should be done in a similar fashion for each level of accuracy.

About the performance, there are also several possibilities: we can report the fitness error function by function, or we can compute an aggregate measure of performance (such as an average). Initially, we opt for an aggregate measure, considering two options:

- *Average ranking*, which is calculated by sorting the algorithms for each function based on its error (lower position to best ones). Then the average ranking is calculated so that an algorithm with a lower average ranking value is declared to perform better, on average, than other with a higher ranking value.
- *A particular measure proposed in the considered benchmark*, which assigns for each function a different score to each algorithm, based on its ranking position.

The performance measure recommended in competitions with the CEC'2013 benchmark is the second one of these options. However, we will first depict the average ranking, since it evaluates the performance of algorithms in a more general and understandable way.

7.2.3. Selecting the reference algorithms as per Guideline #1

In order to do a right comparison, a clear criterion is needed to select the algorithms included in the comparison, aiming at fairly proving the convenience of the algorithm in regards to its competitive performance with other methods. Following the guidelines, we should:

- *Compare against reference algorithms*: in this benchmark DECCG [23] will take this role.
- *Compare against similar algorithms*: this aspect is specially relevant when the proposed algorithm is a modified version of a previously published approach. In our case, SHADE-ILS can be deemed a new algorithm. However, other proposals featuring similar concepts were previously proposed in the literature, such as IHDELS [118]. Following our guidelines, we have included these previous methods for their comparison to our proposal.
- *Compare against competitive algorithms*: this is often a hard decision to make, since it is difficult to scrutinize the entire state-of-the-art related to the optimization problem/algorithm/benchmark under consideration. However, since the benchmark is widely used in international competitions, we can use the winning approaches in these competitions as competitive algorithms to which to compare our proposed approach. As such, one of the solvers in this field is MOS-CEC2013 [119], which has been the best algorithm in these competitions for years. Additionally, we are going also to include MLSHADE-SPA [120] in our comparison, as it was reported to outperform MOS-CEC2013 results in the 2018 competition. Nowadays, there are other competitive algorithms, but we focus on the algorithms proposed until 2018, the year in which the algorithm was presented [117].

On balance, we compare our method against a considered previous version (IHDELS), competitive algorithms (MOS-CEC2013, MLSHADE-SPA), and a reference algorithm (DECCG).

7.2.4. Testing and validating the results as per Guideline #2

After the design of the experimentation, experiments are carried out, and results are validated. Following the recommendations about statistical validation in Guidelines 3.2 and 4.1, normality or homocedasticity tests should be performed. However, it has been proven that such tests

[1] Code for the CEC'2013 Large Scale Global Optimization benchmark: https://www.tflsgo.org/special$_s$essions/wcci2020.html}new-code (accessed on April 16th, 2020).

Table 17
Average ranking of the algorithms considered for the statistical comparison.

Algorithm	Ranking
SHADE-ILS	1.967
MLSHADE-SPA	2.433
MOS-CEC2013	2.700
IHDELS	3.633
DECCG	4.267

Table 18
Statistical validation (SHADE-ILS is the control algorithm).

SHADE-ILS versus	Wilcoxon p-value	Wilcoxon p-value*
MLSHADE-SPA	1.51e-01	1.51e-01 ≈
MOS-CEC2013	4.79e-02	9.58e-02 ≈
IHDELS	1.53e-03	2.50e-02 √
DECCG	8.36e-03	6.10e-03 √

√: statistical differences exist with significance level $\alpha = 0.05$. *: p-value corrected with the Holm procedure.

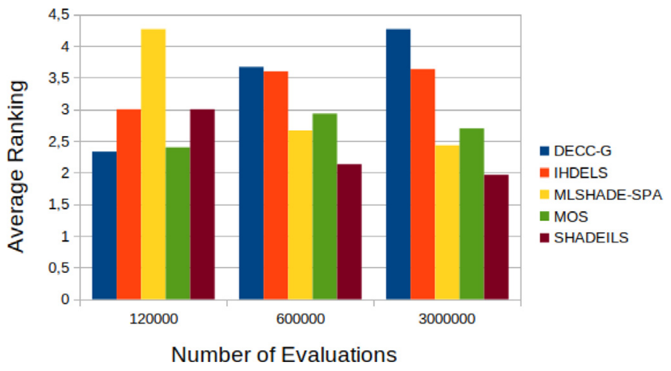


Fig. 8. Average ranking of the algorithms for different numbers of fitness evaluations..

do not usually pass for a benchmark as the chosen one [41]. Therefore, we have opted for non-parametric tests, given that it is unlikely that normality and homocedasticity hold for the CEC'2013 benchmark.

Thus, the first step to take is to calculate the average ranking, followed by the non-parametric hypothesis test. In order to compare the algorithms, we resort to Tacolab [116][14], a web tool that eases the application of different comparison methods among algorithms.

Table 17 shows the average ranking of the four algorithms under comparison over the CEC'2013 LSGO benchmark. We recall that SHADE-ILS is the new algorithmic proposal, whereas MLSHADE-SPA is another proposal presented in the same competition than IHDELS, and MOS-CEC2013 and DECCG are the two state-of-the-art methods considered as reference algorithms. The table depicts the average ranking computed from the relative position of the four methods when ranked for each of the functions in the benchmark. As can be observed in this table, SHADE-ILS exhibits a slightly better performance than MLSHADE-SPA and MOS-CEC2013, and a much better rank than IHDELS and DECCG. In particular, although the preceding approach (IHDELS) performed worse than MOS-CEC2013, SHADE-ILS renders a significantly better performance. This aspect is quite important, because it is not common to directly design a competitive algorithm from scratch.

As stated in Guideline #2 (Section 4), performance measures like the average ranking are not conclusive, since performance gaps may occur due to the stochastic nature of the algorithms under comparison. This is the reason why these results should be further analyzed for elucidating whether the differences are significant. For this purpose, we use the Friedman rank-sum test. The p-value reported by this test is 4.87e-03, which is clearly significant at the $\alpha = 0.05$ confidence level. Now that the aforementioned differences have been assessed, we can proceed towards the multiple comparison, including a familywise error rate correction tackled with the Holm procedure.

Table 18 presents the results of this analysis. As can be observed, differences are significant between SHADE-ILS, DECCG and IHDELS. Regarding MLSHADE-SPA, there are not statistical differences with respect to SHADE-ILS. Finally, in the case of MOS-CEC2013, there are also no statistical differences after applying the Holm correction procedure and using a confidence level of 5%, yet increasing the confidence level up to 10% would make it possible to conclude that differences are statistically significant. This comparison should also consider different checkpoints, e.g., 1%, 10% and 100%, or every 10% of the maximum number of fitness evaluations available. This complementary analysis would reflect not only the final result of the algorithms, but also their convergence speed.

Besides that, we indicated in Section 4.2 that a graphical visualization is useful for the analysis. In this case, we complement the study summarized in Table 17 with Fig. 8. In this figure, it is more evident that the differences between algorithms increase with the number of fitness evaluations: whereas MOS is better than DECCG and IHDELS, both MLSHADE-SPA and SHADE-ILS improve their results since 600,000 evaluations, showing that SHADE-ILS achieves the best average ranking with 600,000 and 3,000,000 evaluations. Ideally, a convergence plot could be more informative, but in the CEC'2013 benchmark the milestones posed by the competition are very reduced, so a bar plot like the depicted one results to be more helpful for the purpose of this analysis.

7.2.5. *Components analysis and tuning as per Guideline #3*

As mentioned before, a comparison of a proposed method with just reference and/or state-of-the-art algorithms is usually not enough. Following Guideline #3 (Section 5), when analyzing the algorithm it is also important to clarify the objectives for the proposed design, and then show quantitative evidence of the claims about the behavior of the algorithm. This way, the study can shed light on the influence of the different components over the reported final results. In our use case, we do not explain the objectives and main ideas of the algorithm. Instead, we remark that the main changes featured by SHADE-ILS with respect to IHDELS is i) a modification of the Differential Evolution component (from SaDE to SHADE); and ii) the restart mechanism. We refer interested readers to [20] for further details.

Table 19 shows the results obtained by the different components of the algorithm. This table clearly exposes that the outperforming behavior of the proposed method is due to all its novel contributions, rather than a subset of them. Furthermore, these changes do not add complexity to the overall search process.

Following Guideline #3, the use of an automatic tuning mechanism is also recommended for the different algorithms in the comparison. However, in our case we use the results reported by their authors in the contributions where the algorithms were first presented, so it is expected that these results were obtained by using the best parameter values. Regarding the parameter values of the SHADE-ILS proposal, they should be obtained by a tuning process, ideally conducted by an automatic tool. In our case, a manual tuning has been conducted due to computational constraints (in particular, processing time). If more resources for computation were available, a complete tuning process could be conducted by resorting to available tools such as the ones commented in Subsection 5.4.

7.2.6. *Justifying the usefulness of the algorithm as per Guideline #4*

Following Guideline #4 (Section 6), there are several ways to show the usefulness of an algorithm:

[14] Tacolab website: https://tacolab.org/

Table 19
Comparisons between different components of the proposal.

Func.	Using SHADE +New Restart	Using SaDE +New Restart	Using SHADE +Old Restart	IHDELS
F_1	2.69e-24	1.21e-24	1.76e-28	**4.80e-29**
F_2	**1.00e+03**	1.26e+03	1.40e+03	1.27e+03
F_3	2.01e+01	2.01e+01	2.01e+01	**2.00e+01**
F_4	**1.48e+08**	1.58e+08	2.99e+08	3.09e+08
F_5	**1.39e+06**	3.07e+06	1.76e+06	9.68e+06
F_6	**1.02e+06**	1.03e+06	1.03e+06	1.03e+06
F_7	**7.41e+01**	8.35e+01	2.44e+02	3.18e+04
F_8	**3.17e+11**	3.59e+11	8.55e+11	1.36e+12
F_9	**1.64e+08**	2.48e+08	2.09e+08	7.12e+08
F_{10}	**9.18e+07**	9.19e+07	9.25e+07	9.19e+07
F_{11}	5.11e+05	**4.76e+05**	5.20e+05	9.87e+06
F_{12}	**6.18e+01**	1.10e+02	3.42e+02	5.16e+02
F_{13}	**1.00e+05**	1.34e+05	9.61e+05	4.02e+06
F_{14}	**5.76e+06**	6.14e+06	7.40e+06	1.48e+07
F_{15}	**6.25e+05**	8.69e+05	1.01e+06	3.13e+06
Better	12	1	0	2

- *Quality of the results*: in this case, given the good results in the comparisons against *state-of-the-art* and reference methods, the scientific value of the contribution is clear.
- *Technical novelty*: the combination of local search methods and the hybridization of SHADE are novel. However, for the sake of conciseness we will not elaborate here on the originality of these ingredients, as it would require an exhaustive review of the recent history of DE approaches for large-scale global optimization. We defer the reader to the analysis made in [117] in this regard.
- *Methodological contribution*: SHADE-ILS improves a previous hybridization of DE with local search [118] by embracing, among other algorithmic additions, Success-History based Adaptive Differential Evolution (SHADE) at its core, which culminates a historical series of adaptive DE solvers. This poses no doubt on the scientific contribution of this study, as can stimulate new research directions towards considering new local search methods hybridized with SHADE.
- A special attention should be given to the *simplicity* of SHADE-ILS. In this algorithm the model is not very complex, and the number of parameters is simpler than other proposals (due that its components require few parameters). Besides, changes made with respect to IHDELS do not increase its number of parameters.

7.2.7. Summary of the use case

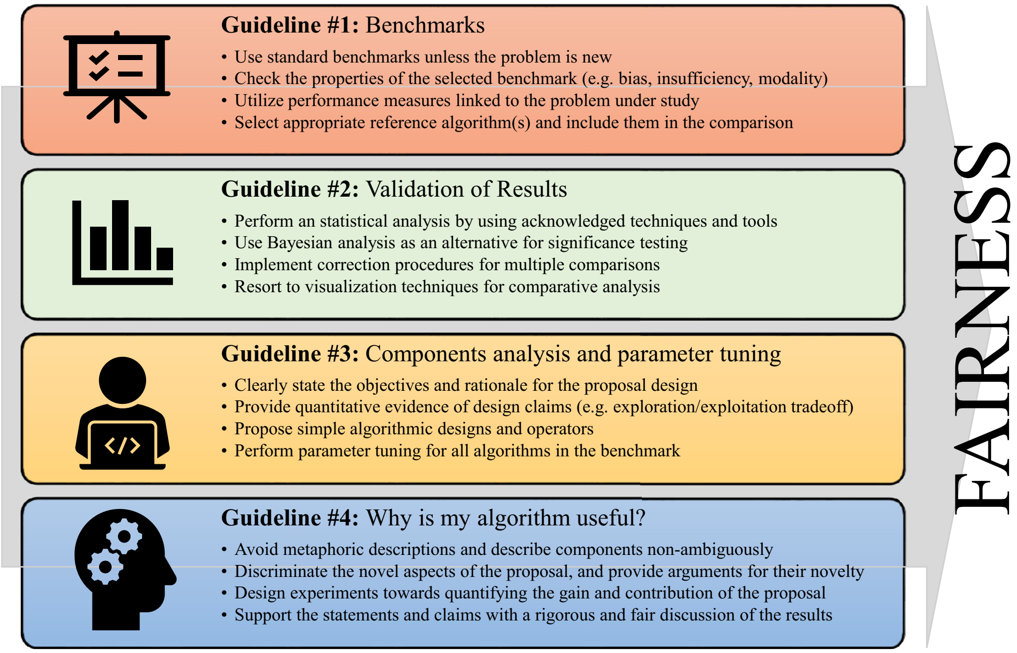
On a closing note, the use case depicted in this section follows most of the guidelines of our proposed methodology. As in the previous use case, the main procedures followed are highlighted in Fig. 7. In addition, we briefly describe now the main actions taken for each of the proposed guidelines:

- Guideline #1: we have resorted to the standard CEC'2013 benchmark, which is widely accepted by the community working on large-scale global optimization. Also, we have checked that the benchmark follows several of the requirements imposed by the guidelines. Finally, we have compared our proposed method against similar state-of-the-art techniques and a reference baseline from the field.
- Guideline #2: as has been shown throughout the discussion, the validation of the results has been done according to the good practices prescribed in Section 4, including non-parametric hypothesis tests. Also, we have also shown how several results can be properly visualized to make the outcome of the comparisons more understandable to the audience.
- Guideline #3: we have clearly highlighted the objectives of the algorithms, and we have compared the influence of the different novel elements of the proposed algorithm. Thus, we have shown that the good results are not influenced by just one component, but to the synergy between the different elements. We have also shown that the proposal is not unnecessarily complex. Finally, a tuning process has been also applied.
- Guideline #4: in this use case, the proposal of our method is easy to justify. SHADE-ILS not only improves a previous hybridization of DE with local search [118], but also surpasses MOS-CEC2013 and MLSHADE-SPA (although without statistical significance), which have dominated the competition over the last few years. This poses no doubt on the scientific contribution of this study, as can stimulate new research directions towards considering new local search methods hybridized with SHADE. In addition, the previous state-of-the-art algorithm, MOS, has been clearly surprised by SHADE-ILS and MLSHADE-SPA, hence becoming the most competitive algorithms (with a preference by SHADE-ILS, by its better performance and simplicity).

8. Conclusions and outlook

In this work we have stressed on the need for circumventing common mistakes and flaws observed in the field of bio-inspired optimization, particularly when new meta-heuristic algorithms are proposed and experimentally validated over benchmarks designed to this end. Specifically, we have reviewed and critically analyzed contributions dealing with experimental recommendations and practices related to meta-heuristics. Following our literature study, we have prescribed a set of methodological recommendations for preparing a successful proposal of bio-inspired meta-heuristic algorithms, from the definition of the experimentation to the presentation of the results. A number of useful techniques (graphically summarized in Fig. 9) have been suggested for prospective studies to implement our proposed methodological framework, in an attempt at ensuring fairness, coherence and soundness in future studies on the topic. Two different case studies have been designed to exemplify the application of our prescribed methodology, discussing on the results of the application of each guideline. Although both case studies deal with well-known benchmarks, we envision that our methodology can be a core part of the design process of meta-heuristic algorithms for real-world optimization problems, following the guidelines of our recently published tutorial in this matter [121]. In those cases, the statistical validation of the results should not be considered the final step of the analysis: the significance of the results should be analyzed from different perspectives, and taking into consideration other measures of practical relevance (e.g. memory consumption).

In such a vibrant field, with new algorithmic proposals flourishing vigorously, common methodological grounds are urgently needed. Scientific advancements in years to come will only be achieved if the community reaches an agreement on how algorithms should be tested and compared to each other. This is indeed the aim of our work: to gather and group recommended practices around an unified set of systematic



Guideline #1: Benchmarks

- Use standard benchmarks unless the problem is new
- Check the properties of the selected benchmark (e.g. bias, insufficiency, modality)
- Utilize performance measures linked to the problem under study
- Select appropriate reference algorithm(s) and include them in the comparison

Guideline #2: Validation of Results

- Perform an statistical analysis by using acknowledged techniques and tools
- Use Bayesian analysis as an alternative for significance testing
- Implement correction procedures for multiple comparisons
- Resort to visualization techniques for comparative analysis

Guideline #3: Components analysis and parameter tuning

- Clearly state the objectives and rationale for the proposal design
- Provide quantitative evidence of design claims (e.g. exploration/exploitation tradeoff)
- Propose simple algorithmic designs and operators
- Perform parameter tuning for all algorithms in the benchmark

Guideline #4: Why is my algorithm useful?

- Avoid metaphoric descriptions and describe components non-ambiguously
- Discriminate the novel aspects of the proposal, and provide arguments for their novelty
- Design experiments towards quantifying the gain and contribution of the proposal
- Support the statements and claims with a rigorous and fair discussion of the results

FAIRNESS

Fig. 9. Guidelines composing the methodological framework for comparing meta-heuristics proposed in this work.

methodological guidelines. We sincerely hope that the material and prescriptions given herein will guide newcomers in their arrival to this exciting research avenue.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Antonio LaTorre: Conceptualization, Methodology, Investigation, Software, Writing – original draft, Writing – review & editing. **Daniel Molina:** Conceptualization, Methodology, Investigation, Software, Writing – original draft, Writing – review & editing. **Eneko Osaba:** Methodology, Investigation, Writing – original draft. **Javier Poyatos:** Methodology, Investigation, Software, Validation. **Javier Del Ser:** Conceptualization, Methodology, Investigation, Validation, Supervision, Writing – original draft, Writing – review & editing, Funding acquisition. **Francisco Herrera:** Conceptualization, Supervision, Writing – review & editing, Funding acquisition.

Acknowledgments

This work was supported by grants from the Spanish Ministry of Science (TIN2016-8113-R, TIN2017-89517-P and TIN2017-83132-C2-2-R) and Universidad Politécnica de Madrid (PINV-18-XEOGHQ-19-4QTEBP). Eneko Osaba and Javier Del Ser-would also like to thank the Basque Government for its funding support through the ELKARTEK and EMAITEK programs. Javier Del Ser-receives funding support from the Consolidated Research Group MATHMODE (IT1294-19) granted by the Department of Education of the Basque Government.

References

[1] J. Del Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P.N. Suganthan, C.A.C. Coello, F. Herrera, Bio-inspired computation: where we stand and what's next, Swarm Evol Comput 48 (2019) 220–250, doi:10.1016/j.swevo.2019.04.008.

[2] D. Molina, J. Poyatos, J. Del Ser, S. García, F. Herrera, Comprehensive taxonomies of nature- and bio-inspired optimization: inspiration versus algorithmic behavior, critical analysis and recommendations, Cognit Comput 12 (2020) 897–939, doi:10.1007/s12559-020-09730-8.

[3] J.C. Stanley, The influence of fisher's "the design of experiments" on educational research thirty years later, Am Educ Res J 3 (3) (1966) 223–229, doi:10.3102/00028312003003223.

[4] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol Comput 1 (1) (2011) 3–18, doi:10.1016/j.swevo.2011.02.002.

[5] D.S. Johnson, A theoretician's guide to the experimental analysis of algorithms, in: M.H. Goldwasser, D.S. Johnson, C.C. McGeoch (Eds.), Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges, American Mathematical Society, 2002, pp. 215–250.

[6] M. Hellwig, H.-G. Beyer, Benchmarking evolutionary algorithms for single objective real-valued constrained optimization – a critical review, Swarm Evol Comput 44 (2019) 927–944, doi:10.1016/j.swevo.2018.10.002.

[7] T. Weise, R. Chiong, K. Tang, Evolutionary optimization: pitfalls and booby traps, J Comput Sci Technol 27 (5) (2012) 907–936, doi:10.1007/s11390-012-1274-4.

[8] A.V. Kononova, D.W. Corne, P.D. Wilde, V. Shneer, F. Caraffini, Structural bias in population-based algorithms, Inf Sci (Ny) 298 (2015) 468–490, doi:10.1016/j.ins.2014.11.035.

[9] R. Storn, K.V. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359, doi:10.1023/A:1008202821328.

[10] Z. Hu, Q. Su, X. Yang, Z. Xiong, Not guaranteeing convergence of differential evolution on a class of multimodal functions, Appl Soft Comput 41 (2016) 479–487, doi:10.1016/j.asoc.2016.01.001.

[11] A.P. Piotrowski, J.J. Napiorkowski, Searching for structural bias in particle swarm optimization and differential evolution algorithms, Swarm Intell. 10 (4) (2016) 307–353, doi:10.1007/s11721-016-0129-y.

[12] F. Caraffini, A.V. Kononova, D. Corne, Infeasibility and structural bias in differential evolution, Inf Sci (Ny) 496 (2019) 161–179, doi:10.1016/j.ins.2019.05.019.

[13] K.V. Price, How symmetry constrains evolutionary optimizers, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1712–1719. DOI: 10.1109/CEC.2017.7969508

[14] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol Comput 9 (2) (2001) 159–195, doi:10.1162/106365601750190398.

[15] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Technical Report, Nanyang Technological University, Singapore, 2010. http://www.ntu.edu.sg/home/EPNSugan/

[16] S. Das, P.N. Suganthan, Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems, Technical Report, Jadavpur University, India and Nanyang Technological University, Singapore, 2010.

[17] J.J. Liang, B.-Y. Qu, P.N. Suganthan, A. Hernández-Díaz, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization, Technical Report, Computational Intelligence Laboratory,

Zhengzhou University, China and Nanyang Technological University, Singapore, 2013.

[18] J.J. Liang, B.-Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real–Parameter Numerical Optimization, Technical Report, Computational Intelligence Laboratory, Zhengzhou University, China and Nanyang Technological University, Singapore, 2013.

[19] N.H. Award, M.Z. Ali, P.N. Suganthan, J.J. Liang, B.Y. Qu, Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-parameter Numerical Optimization, Technical Report, Nanyang Technological University, Singapore, 2016.

[20] D. Molina, A. LaTorre, F. Herrera, An insight into bio-inspired and evolutionary algorithms for global optimization: review, analysis, and lessons learnt over a decade of competitions, Cognit Comput 10 (2018) 517–544, doi:10.1007/s12559-018-9554-0.

[21] G. Wu, W. Pedrycz, P.N. Suganthan, H. Li, Using variable reduction strategy to accelerate evolutionary optimization, Appl Soft Comput 61 (2017) 283–293.

[22] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen, Z. Yang, Benchmark Functions for the CEC 2008 Special Session and Competition on Large Scale Global Optimization, Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.

[23] X. Li, K. Tang, M.N. Omidvar, Z. Yang, K. Quin, Benchmark Functions for the CEC 2013 Special Session and Competition on Large Scale Global Optimization, Technical Report, Evolutionary Computation and Machine Learning Group, 2013.

[24] R. Bellman, Dynamic programming, Science 153 (3731) (1966) 34–37.

[25] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225–239, doi:10.1109/tevc.2004.826069.

[26] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif Intell Rev 33 (1–2) (2009) 61–106, doi:10.1007/s10462-009-9137-2.

[27] K. Tang, X. Li, P.N. Suganthan, Z. Yang, T. Weise, Benchmark Functions for the CEC 2010 Special Session and Competition on Large Scale Global Optimization, Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, 2010.

[28] P. Oliveto, T. Paixão, J. Pérez Heredia, D. Sudholt, B. Trubenová, How to escape local optima in black box optimisation: when non-elitism outperforms elitism, Algorithmica 80 (5) (2018) 1604–1633, doi:10.1007/s00453-017-0369-2.

[29] L. Hernando, A. Mendiburu, J. Lozano, An evaluation of methods for estimating the number of local optima in combinatorial optimization problems, Evol Comput 21 (4) (2013) 625–658, doi:10.1162/EVCO_a_00100.

[30] A.M. Sutton, M. Lunacek, L.D. Whitley, Differential evolution and non-separability: Using selective pressure to focus search, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, in: GECCO '07, Association for Computing Machinery, New York, NY, USA, 2007, pp. 1428–1435. DOI: 10.1145/1276958.1277221

[31] S. Bagheri, W. Konen, T. Bäck, Solving optimization problems with high conditioning by means of online whitening, in: GECCO 2019 Companion - Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion, 2019, pp. 243–244. DOI: 10.1145/3319619.3322008

[32] S. Finck, N. Hansen, R. Ros, A. Auger, Real-Parameter Black-Box Optimization Benchmarking 2010: Presentation of the Noisy Functions, Technical Report, Research Center PPE, 2010.

[33] J. Rapin, O. Teytaud, Nevergrad - A gradient-free optimization platform, 2018, (https://GitHub.com/FacebookResearch/Nevergrad).

[34] H.-G. Beyer, Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice, Comput Methods Appl Mech Eng 186 (2–4) (2000) 239–267, doi:10.1016/s0045-7825(99)00386-2.

[35] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments-a survey, IEEE Trans. Evol. Comput. 9 (3) (2005) 303–317, doi:10.1109/TEVC.2005.846356.

[36] C. García-Martínez, P.D. Gutiérrez, D. Molina, M. Lozano, F. Herrera, Since CEC 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness, Soft comput 21 (19) (2017) 5573–5583, doi:10.1007/s00500-016-2471-9.

[37] P. Civicioglu, E. Besdok, A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms, Artif Intell Rev 39 (4) (2011) 315–346, doi:10.1007/s10462-011-9276-0.

[38] V. Osuna-Enciso, E. Cuevas, H. Sossa, A comparison of nature inspired algorithms for multi-threshold image segmentation, Expert Syst Appl 40 (4) (2013) 1213–1219, doi:10.1016/j.eswa.2012.08.017.

[39] J. Demšar, Statistical comparisons of classifiers over multiple datasets, Journal of Machine Learning Research 7 (2006) 1–30.

[40] J.M. Whitacre, Use of statistical outlier detection method in adaptive evolutionary algorithms, in: 2006 Conference on Genetic and Evolutionary Computation (GECCO '06), ACM Press, 2006, pp. 1345–1352. DOI: 10.1145/1143997.1144205

[41] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, Journal of Heuristics 15 (6) (2008) 617–644, doi:10.1007/s10732-008-9080-4.

[42] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf Sci (Ny) 180 (10) (2010) 2044–2064, doi:10.1016/j.ins.2009.12.010.

[43] C.C. McGeoch, Experimental Analysis of Algorithms, in: P.M. Pardalos, H.E. Romeijn (Eds.), Handbook of Global Optimization: Volume 2, Springer US, Boston, MA, 2002, pp. 489–513. DOI: 10.1007/978-1-4757-5362-2_14

[44] A.E. Eiben, M. Jelasity, A critical note on experimental research methodology in EC, in: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), volume 1, 2002, pp. 582–587 vol.1, doi:10.1109/CEC.2002.1006991.

[45] A.H. Halim, I. Ismail, S. Das, Performance assessment of the metaheuristic optimization algorithms: an exhaustive review, Artif Intell Rev (2020), doi:10.1007/s10462-020-09906-6.

[46] A. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Trans. Evol. Comput. 3 (2) (1999) 124–141, doi:10.1109/4235.771166.

[47] J. Grefenstette, Optimization of control parameters for genetic algorithms, IEEE Trans Syst Man Cybern 16 (1) (1986) 122–128, doi:10.1109/tsmc.1986.289288.

[48] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Inf Process Lett 85 (6) (2003) 317–325, doi:10.1016/s0020-0190(02)00447-7.

[49] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417, doi:10.1109/tevc.2008.927706.

[50] E. Montero, M.-C. Riff, B. Neveu, A beginner's guide to tuning methods, Appl Soft Comput 17 (2014) 39–51, doi:10.1016/j.asoc.2013.12.017.

[51] P. Balaprakash, M. Birattari, T. Stützle, Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement, vol. 4771 of Hybrid Metaheuristics, Springer Berlin Heidelberg, 2007, pp. 108–122. DOI: 10.1007/978-3-540-75514-2_9

[52] M. López-Ibáñez, J. Dubois-Lacoste, L.P. Cáceres, M. Birattari, T. Stützle, The irace package: iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (2016) 43–58, doi:10.1016/j.orp.2016.09.002.

[53] V. Nannen, A.E. Eiben, Relevance estimation and value calibration of evolutionary algorithm parameters, in: M.M. Veloso (Ed.), 20th International Joint Conference on Artificial Intelligence (IJCAI), 2007, pp. 975–980.

[54] F. Hutter, H.H. Hoos, K. Leyton-Brown, T. Stützle, Paramils: an automatic algorithm configuration framework, Journal of Artificial Intelligence Research 36 (2009) 267–306, doi:10.1613/jair.2861.

[55] N. Hansen, D. Brockhoff, O. Mersmann, T. Tusar, D. Tusar, O.A. ElHara, P.R. Sampaio, A. Atamna, K. Varelas, U. Batu, D.M. Nguyen, F. Matzner, A. Auger, COmparing Continuous Optimizers: numbbo/COCO on Github, 2019, DOI: 10.5281/zenodo.2594848.

[56] M. Helbig, A.P. Engelbrecht, Performance measures for dynamic multi-objective optimisation algorithms, Inf Sci (Ny) 250 (2013) 61–81, doi:10.1016/j.ins.2013.06.051.

[57] S. Mirjalili, A. Lewis, Novel performance metrics for robust multi-objective optimization algorithms, Swarm Evol Comput 21 (2015) 1–23, doi:10.1016/j.swevo.2014.10.005.

[58] X.-S. Yang, Firefly algorithms for multimodal optimization, in: Stochastic Algorithms: Foundations and Applications, 2009, pp. 169–178.

[59] S. Saremi, S. Mirjalili, A. Lewis, Grasshopper optimisation algorithm: theory and application, Adv. Eng. Software 105 (2017) 30–47, doi:10.1016/j.advengsoft.2017.01.004.

[60] W. Gong, Z. Cai, Differential evolution with ranking-based mutation operators, IEEE Trans Cybern 43 (6) (2013) 2066–2081, doi:10.1109/tcyb.2013.2239988.

[61] S.S. Shapiro, M.B. Wilk, An analysis of variance test for normality (complete samples), Biometrika 52 (3–4) (1965) 591–611, doi:10.1093/biomet/52.3-4.591.

[62] N.M. Razali, Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests, Journal of Statistical Modeling and Analytics 2 (1) (2011) 21–33.

[63] H. Levene, Robust Tests for Equality of Variances, in: I. Olkin, S.G. Ghurye, W. Hoeffding, W.G. Madow, H.B. Mann (Eds.), Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling, Stanford University Press, Stanford, Calif, 1960, pp. 278–292.

[64] D.J. Sheskin, Handbook of parametric and nonparametric statistical procedures, 4th, Chapman & Hall/CRC, 2007.

[65] B.L. Welch, The generalization of 'student's' problem when several different population variances are involved, Biometrika 34 (1–2) (1947) 28–35, doi:10.1093/biomet/34.1-2.28.

[66] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (6) (1945) 80–83, doi:10.2307/3001968.

[67] W.W. Daniel, Applied nonparametric statistics, The Duxbury advanced series in statistics and decision sciences, Duxbury Thomson Learning, 1990.

[68] M. Aickin, H. Gensler, Adjusting for multiple testing when reporting research results: the bonferroni vs holm methods., Am J Public Health 86 (5) (1996) 726–728, doi:10.2105/AJPH.86.5.726.

[69] O.J. Dunn, Multiple comparisons among means, J Am Stat Assoc 56 (293) (1961) 52–64, doi:10.1080/01621459.1961.10482090.

[70] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (1979) 65–70.

[71] Y. Benjamini, Y. Hochberg, Controlling the false discovery rate: a practical and powerful approach to multiple testing, Journal of the Royal Statistical Society. Series B (Methodological) 57 (1) (1995) 289–300.

[72] G. Hommel, A stagewise rejective multiple test procedure based on a modified bonferroni test, Biometrika 75 (2) (1988) 383–386, doi:10.1093/biomet/75.2.383.

[73] M. Lin, H.C. Lucas, G. Shmueli, Research commentary - too big to fail: large samples and the p-value problem, Information Systems Research 24 (4) (2013) 906–917, doi:10.1287/isre.2013.0480.

[74] A. Benavoli, G. Corani, J. Demšar, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis, Journal of Machine Learning Research 18 (1) (2017) 2653–2688.

[75] J. Carrasco, S. García, M. del Mar Rueda, F. Herrera, rNPBST: An R package covering non-parametric and bayesian statistical tests, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, 2017, pp. 281–292.

[76] A. Benítez-Hidalgo, A.J. Nebro, J. García-Nieto, I. Oregi, J. Del Ser, Jmetalpy: A Python framework for multi-objective optimization with metaheuristics, Swarm Evol Comput 51 (2019) 100598, doi:10.1016/j.swevo.2019.100598.

[77] J. Carrasco, S. García, M. del Mar Rueda, S. Das, F. Herrera, Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: practical guidelines and a critical review, Swarm Evol Comput 54 (2020) 100665, doi:10.1016/j.swevo.2020.100665.

[78] A. Herrera-Poyatos, F. Herrera, Genetic and memetic algorithm with diversity equilibrium based on greedy diversification, CoRR abs/1702.03594 (2017).

[79] A. LaTorre, S. Muelas, J.M. Peña, A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test, Soft Computing - A Fusion of Foundations, Methodologies and Applications 15 (11) (2010) 2187–2199, doi:10.1007/s00500-010-0646-3.

[80] M. Črepinšek, S.H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Comput Surv 45 (3) (2013) 1–33, doi:10.1145/2480741.2480752.

[81] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Balancing the exploration and exploitation capabilities of the Differential Evolution Algorithm, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), 2008, pp. 2686–2693. ISSN: 1941-0026, DOI: 10.1109/CEC.2008.4631159

[82] E.-u. Haq, I. Ahmad, A. Hussain, I.M. Almanjahie, A novel selection approach for genetic algorithms for global optimization of multimodal continuous functions, Comput Intell Neurosci 2019 (2019), doi:10.1155/2019/8640218. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6915132/

[83] A. Hussain, Y.S. Muhammad, Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator, Complex & Intelligent Systems 6 (1) (2020) 1–14, doi:10.1007/s40747-019-0102-7.

[84] F. Vafaee, G. Turán, P.C. Nelson, T.Y. Berger-Wolf, Balancing the exploration and exploitation in an adaptive diversity guided genetic algorithm, in: 2014 IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 2570–2577. ISSN: 1941-0026, DOI: 10.1109/CEC.2014.6900257

[85] A. LaTorre, S. Muelas, J.M. Peña, A comprehensive comparison of large scale global optimizers, Inf Sci (Ny) 316 (2014) 517–549, doi:10.1016/j.ins.2014.09.031.

[86] A.P. Piotrowski, J.J. Napiorkowski, Some metaheuristics should be simplified, Inf Sci (Ny) 427 (2018) 32–62, doi:10.1016/j.ins.2017.10.039.

[87] N.H. Awad, M.Z. Ali, P.N. Suganthan, R.G. Reynolds, An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC 2014 benchmark problems, in: 2016 IEEE Congress on Evolutionary Computation (CEC), 2016, pp. 2958–2965. DOI: 10.1109/CEC.2016.7744163

[88] L.Y. Tseng, C. Chen, Multiple trajectory search for large scale global optimization, in: 2008 IEEE Congress on Evolutionary Computation (CEC), IEEE Press, 2008, pp. 3052–3059. DOI: 10.1109/CEC.2008.4631210

[89] M. Lozano, D. Molina, F. Herrera, Editorial: scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems, Soft Computing - A Fusion of Foundations, Methodologies and Applications 15 (2011) 2085–2087, doi:10.1007/s00500-010-0639-2.

[90] R. Tanabe, A.S. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: 2014 IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 1658–1665. DOI: 10.1109/CEC.2014.6900380

[91] K. De Jong, Parameter Setting in EAs: A 30 Year Perspective, in: F.G. Lobo, C.F. Lima, Z. Michalewicz (Eds.), Parameter Setting in Evolutionary Algorithms, Springer, Berlin, Heidelberg, 2007, pp. 1–18. DOI: 10.1007/978-3-540-69432-8_1

[92] G. Eiben, M.C. Schut, New Ways to Calibrate Evolutionary Algorithms, in: P. Siarry, Z. Michalewicz (Eds.), Advances in Metaheuristics for Hard Optimization, Springer, Berlin, Heidelberg, 2008, pp. 153–177. DOI: 10.1007/978-3-540-72960-0_8

[93] T. Lundstedt, E. Seifert, L. Abramo, B. Thelin, Å. Nyström, J. Pettersen, R. Bergman, Experimental design and optimization, Chemometrics and Intelligent Laboratory Systems 42 (1–2) (1998) 3–40, doi:10.1016/s0169-7439(98)00065-3.

[94] G. Taguchi, Introduction to quality engineering: Designing quality into products and processes, Asian Productivity Organization, 1986.

[95] S. Wessing, N. Beume, G. Rudolph, B. Naujoks, Parameter Tuning Boosts Performance of Variation Operators in Multiobjective Optimization, in: R. Schaefer, C. Cotta, J. Kołodziej, G. Rudolph (Eds.), Parallel Problem Solving from Nature, PPSN XI, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 728–737. DOI: 10.1007/978-3-642-15844-5_73

[96] T. Liao, D. Molina, T. Stützle, Performance evaluation of automatically tuned continuous optimizers on different benchmark sets, Appl Soft Comput 27 (2015) 490–503, doi:10.1016/j.asoc.2014.11.006.

[97] S. García, J. Derrac, S. Ramírez-Gallego, F. Herrera, On the statistical analysis of the parameters' trend in a machine learning algorithm, Progress in Artificial Intelligence 3 (1) (2014) 51–53, doi:10.1007/s13748-014-0043-8.

[98] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (6) (2006) 646–657, doi:10.1109/tevc.2006.872133.

[99] J. Teo, Scalability analysis of fixed versus self-adaptive differential evolution for unconstrained global optimization, Adv Sci Lett 23 (6) (2017) 5144–5146, doi:10.1166/asl.2017.7328.

[100] J. Zhang, A.C. Sanderson, JADE: Adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958, doi:10.1109/TEVC.2009.2014613.

[101] R. Tanabe, A.S. Fukunaga, Evaluating the performance of shade on CEC 2013 benchmark problems, in: 2013 IEEE Congress on Evolutionary Computation (CEC), 2013, pp. 1952–1959. DOI: 10.1109/CEC.2013.6557798

[102] J. Brest, M.S. Maucec, B. Boskovic, Single objective real-parameter optimization: Algorithm jso, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1311–1318. DOI: 10.1109/cec.2017.7969456

[103] T. Bartz-Beielstein, C.W.G. Lasarczyk, M. Preuss, Sequential parameter optimization, in: 2005 IEEE Congress on Evolutionary Computation, volume 1, 2005, pp. 773–780Vol.1. DOI: 10.1109/CEC.2005.1554761

[104] K. Sörensen, Metaheuristics — the metaphor exposed, International Transactions in Operational Research 22 (1) (2015) 3–18, doi:10.1111/itor.12001.

[105] J. Swan, S. Adriaensen, M. Bishr, E.K. Burke, J.A. Clark, P. De Causmaecker, J. Durillo, K. Hammond, E. Hart, C.G. Johnson, et al., A research agenda for metaheuristic standardization, in: Proceedings of the XI metaheuristics international conference, 2015, pp. 1–3.

[106] J. Swan, S. Adraensen, A.E. Brownlee, C.G. Johnson, A. Kheiri, F. Krawiec, J. Merelo, L.L. Minku, E. Özcan, G.L. Pappa, et al., Towards metaheuristics” in the large”, arXiv preprint arXiv:2011.09821 (2020).

[107] M.A. Lones, Mitigating metaphors: a comprehensible guide to recent nature-inspired algorithms, SN Computer Science 1 (1) (2020) 49.

[108] M. Jain, V. Singh, A. Rani, A novel nature-inspired algorithm for optimization: squirrel search algorithm, Swarm Evol Comput 44 (2019) 148–175, doi:10.1016/j.swevo.2018.02.013.

[109] A.W. Mohamed, A.A. Hadi, A.K. Mohamed, Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm, Int. J. Mach. Learn. Cybern. 11 (7) (2020) 1501–1529, doi:10.1007/s13042-019-01053-x.

[110] W. Zhao, L. Wang, Z. Zhang, Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm, Neural Computing and Applications 32 (13) (2020) 9383–9425, doi:10.1007/s00521-019-04452-x.

[111] A.W. Mohamed, A.K. Mohamed, Adaptive guided differential evolution algorithm with novel mutation for numerical optimization, Int. J. Mach. Learn. Cybern. 10 (2) (2019) 253–277, doi:10.1007/s13042-017-0711-7.

[112] A.W. Mohamed, A.A. Hadi, A.M. Fattouh, K.M. Jambi, LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 145–152. DOI: 10.1109/CEC.2017.7969307

[113] A.A. Hadi, A.W. Mohamed, K.M. Jambi, Single-Objective Real-Parameter Optimization: Enhanced LSHADE-SPACMA Algorithm, in: F. Yalaoui, L. Amodeo, E.-G. Talbi (Eds.), Heuristics for Optimization and Learning, volume 906, Springer International Publishing, Cham, 2021, pp. 103–121. DOI: 10.1007/978-3-030-58930-1_7

[114] J. Brest, M.S. Maučec, B. Bošković, Single objective real-parameter optimization: Algorithm jso, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1311–1318. DOI: 10.1109/CEC.2017.7969456

[115] A. Kumar, R.K. Misra, D. Singh, Improving the local search capability of effective butterfly optimizer using covariance matrix adapted retreat phase, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1835–1842. DOI: 10.1109/CEC.2017.7969524

[116] D. Molina, A. LaTorre, Toolkit for the automatic comparison of optimizers: Comparing large-scale global optimizers made easy, in: 2018 IEEE Congress on Evolutionary Computation (CEC), 2018, pp. 1–8. DOI: 10.1109/CEC.2018.8477924

[117] D. Molina, A. LaTorre, F. Herrera, SHADE with iterative local search for large-scale global optimization, in: 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 2018, pp. 1–8. DOI: 10.1109/CEC.2018.8477755

[118] D. Molina, F. Herrera, Iterative hybridization of DE with local search for the CEC'2015 special session on large scale global optimization, in: 2015 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2015, pp. 1974–1978. DOI: 10.1109/CEC.2015.7257127

[119] A. LaTorre, S. Muelas, J.M. Peña, Large scale global optimization: Experimental results with MOS-based hybrid algorithms, in: 2013 IEEE Congress on Evolutionary Computation (CEC), IEEE Press, 2013, pp. 2742–2749. DOI: 10.1109/CEC.2013.6557901

[120] A.A. Hadi, A.W. Mohamed, K.M. Jambi, LSHADE-SPA memetic framework for solving large-scale optimization problems, Complex & Intelligent Systems 5 (1) (2019) 25–40, doi:10.1007/s40747-018-0086-8.

[121] E. Osaba, E. Villar-Rodriguez, J. Del Ser, A. J. Nebro, D. Molina, A. LaTorre, P.N. Suganthan, C. Coello, F. Herrera, A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems, Swarm Evol Comput (2021) 100888, doi:10.1016/j.swevo.2021.100888.