Bayesian performance analysis for black-box optimization benchmarking

Borja Calvo, Ofer Shir, Josu Ceberio, Carola Doerr, Hao Wang, Thomas Back, Jose Lozano

▶ **To cite this version:**

Borja Calvo, Ofer Shir, Josu Ceberio, Carola Doerr, Hao Wang, et al.. Bayesian performance analysis for black-box optimization benchmarking. Genetic and Evolutionary Computation Conference GECCO 2019, Jul 2019, Prague, Czech Republic. pp.1789-1797, 10.1145/3319619.3326888 . hal-02179609

HAL Id: hal-02179609

https://hal.sorbonne-universite.fr/hal-02179609v1

Submitted on 14 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Bayesian Performance Analysis for Black-Box Optimization Benchmarking

Borja Calvo
University of the Basque Country
UPV/EHU
Donostia-San Sebastian, Spain
borja.calvo@ehu.es

Ofer M. Shir
Tel-Hai College and Migal Institute
Upper Galilee, Israel
ofersh@telhai.ac.il

Josu Ceberio
University of the Basque Country
UPV/EHU
Donostia-San Sebastian, Spain
josu.ceberio@ehu.es

Carola Doerr
Sorbonne University, CNRS
Paris, France
Carola.Doerr@mpi-inf.mpg.de

Hao Wang
LIACS, Leiden University
Leiden, The Netherlands
h.wang@liacs.leidenuniv.nl

Thomas Bäck
LIACS, Leiden University
Leiden, The Netherlands
t.h.w.baeck@liacs.leidenuniv.nl

Jose A. Lozano
University of the Basque Country
UPV/EHU
Donostia-San Sebastian, Spain
ja.lozano@ehu.eus

ABSTRACT

The most commonly used statistics in Evolutionary Computation (EC) are of the Wilcoxon-Mann-Whitney-test type, in its either paired or non-paired version. However, using such statistics for drawing performance comparisons has several known drawbacks. At the same time, Bayesian inference for performance analysis is an emerging statistical tool, which has the potential to become a promising complement to the statistical perspectives offered by the aforementioned p-value type test. This work exhibits the practical use of Bayesian inference in a typical EC setting, where several algorithms are to be compared with respect to various performance indicators. Explicitly, we examine performance data of 11 evolutionary algorithms (EAs) over a set of 23 discrete optimization problems in several dimensions. Using this data, and following a brief introduction to the relevant Bayesian inference practice, we demonstrate how to draw the algorithms' probabilities of winning. Apart from fixed-target and fixed-budget results for the individual problems, we also provide an illustrative example per groups of problems. We elaborate on the computational steps, explain the associated uncertainties, and articulate considerations such as the prior distribution and the sample sizing. We also present as a reference the classical p-value tests.

CCS CONCEPTS

• **Computing methodologies** → **Randomized search**; • **Mathematics of computing** → **Statistical paradigms**;

KEYWORDS

performance measures, Bayesian inference, Plackett-Luce model, black-box optimization, benchmarking, evolutionary algorithms

1 INTRODUCTION

Benchmarking environments and competitions are designed to compare algorithms' performance over a set of predefined representative problems. In Evolutionary Computation (EC), benchmarking black-box optimization problems evolved differently in continuous versus discrete domains. While the Black-box Optimization Benchmarking suite (BBOB, [17]) constitutes an established testing framework for evaluating performance of continuous optimizers, the discrete domain, on the other hand, has not had the benefit of an equivalent suite. Attempts to establish such an environment have lately become prominent [12, 23–25]. Wherever performance comparisons are sought, based on empirical data, they call for statistical assessments, to evaluate whether the observed performance gaps can be supported by an appropriate estimator for the true, underlying performance distribution, i.e., a distribution which assigns a probability to each possible result of the algorithms. Within the typical EC context, arrays of recorded optimization runs merely constitute *samples* of the corresponding performance distribution.

Statistical tests fall into the category of statistical inference methods, which in this case seek to deduce measures about the unknown population[1] (the set of all possible results), when given only a sample drawn from that population. The most commonly used statistics in EC are of the Wilcoxon-Mann-Whitney-test type [26], which facilitates non-parametric pairwise performance comparisons, in either *paired* or *non-paired* scenarios [19]. In its non-paired version, the test concentrates on a null hypothesis, H_0, stating that the compared two populations share the same mean. Furthermore, this test assumes that both populations share the same shape and spread of their underlying continuous frequency distributions [19]. To test this hypothesis, a sample from the populations, denoted as $\mathbf{s}$ and referring to two arrays of recorded runs of the algorithms, is used to get a certain statistic $R(\mathbf{s})$, whose probability distribution under H_0 ($\mathbf{Pr}_{H_0}(R)$) is known, given the set of assumptions. Then, the so-called p-value could be calculated, representing the probability of observing such an extreme value for the statistic, $p = \mathbf{Pr}_{H_0}(R \geq R(\mathbf{s}))$, subject to all the assumptions (including H_0). Small p-values, e.g., $p \leq 0.05$, are taken as evidence of H_0 being false, that is, providing confidence that the averages of the populations significantly differ. In other words, such p-value are used as a statistical support to claims concerning, for example, outperformance. However, often times this probability measure is misinterpreted as the probability of H_0 being true (or, alternatively, $1 - p$ misinterpreted as the probability of the *counterpart hypothesis* to hold, i.e., the averages being different) [16].

Recently, Bayesian inference [20] has been proposed as an alternative to analyze algorithms' performance within the Machine Learning community [3, 4], and has since been gradually adopted therein [2]. In this alternative, rather than testing a certain hypothesis, the methods focus on estimating relevant information about the underlying performance distribution, represented by a set of parameters $\boldsymbol{\theta}$ (e.g., subscribing to a performance difference between algorithms, or the probability of this difference being in a certain interval). They directly assess the distribution of $\boldsymbol{\theta}$ conditioned on a sample $\mathbf{s}$ drawn from the performance distribution, which formally reads $\mathbf{Pr}(\boldsymbol{\theta}|\mathbf{s})$.

The current study aims to demonstrate the effectiveness of Bayesian inference analysis for algorithms' performance ranking. To this end, following a short introduction to this analysis, we report on its application to a specific benchmarking campaign, which comprises a set of 23 discrete optimization problems. Technically, the actual benchmarking execution relies on the recently announced IOHprofiler [12], and more concretely on the data of the experiments described in [13].

2 BAYESIAN INFERENCE FOR ALGORITHM RANKING ANALYSIS

Comparing heuristic optimization algorithms implies handling the natural uncertainty associated with empirical results. Usually, this uncertainty stems from two main sources: the stochastic nature of the algorithms, and the extremely large number of possible instances. Therefore, in order to draw sound conclusions, it is not only essential to craft the *design of the experimentation* very carefully. It is equally important to apply statistical tools to the analysis of the results. Statistical tools are applied to do *inference*, i.e., to extract information about a certain underlying probability distribution using a sample drawn from that distribution. If this distribution represents, for example, the probability associated with the possible results obtained by two stochastic algorithms when run on a certain problem instance, then a sample of such a distribution would be a certain number of random runs of the algorithms on that instance. Based on that sample of results, statistical inference methods are used to obtain information about the performance of the algorithms on that particular instance.

The most commonly used statistical tools for this purpose are statistical tests. These methods require the formulation of a *null hypothesis* and provide a measure on whether or not this hypothesis can be refuted with a certain probability. A typical example for a null hypothesis is the statement that the average performance of two or more algorithms is the same for a given set of problems. The statistical test uses the empirical evidence, i.e., the results obtained by the algorithms on one or more instances, to decide whether or not the null hypothesis can be rejected with an appropriate degree of confidence. Therefore, this type of statistical tool provides us with two possible answers. If the null hypothesis is rejected, then we state that "there are statistically significant differences between the algorithms". Otherwise, we cannot make a statement, since there is not enough statistical evidence to ensure that the average performance is different – which is not the same as stating that there are no differences.

Therefore, statistical tests can either confirm that there are differences or simply tell us that we have not enough data to confirm the differences. At this point it is important to note that, in almost all realistic use cases, we already know that *there are* differences between the algorithms (no matter how small), as the algorithms are of different design. In this sense, some authors argue that using statistical tests when comparing algorithms is not informative enough to draw sound conclusions from the experimental data [2]. One of the weak points of statistical tests is that, when the null hypothesis is not rejected, it is not a trivial task to explain it – e.g., either by the small sample size or by the similar performance of the algorithms. As an alternative to statistical tests, Bayesian methods provide us with interpretable information, distinguishing between the estimated measures and the uncertainty about the estimations [2].

Rather than having a single probability distribution to model the data, Bayesian statistics simultaneously considers all possible distributions of a certain family and assigns a probability to each distribution. Therefore, we represent the probability of the data with a certain probability model whose parameters themselves have an associated probability distribution. Before observing any data, the probability of the parameters is represented by the *prior distribution*. This distribution represents our prior belief about which are the true parameters of the distribution from where the data comes. Obviously, when we have access to actual data, we have to update this belief accordingly. This is done using Bayes' rule. If we represent the data (our sample) as $\mathbf{x}$ and its probability (also known as *likelihood function*) as $p(\mathbf{x}|\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ represents the set

[1] The careful reader should note that the term 'population' is exclusively used herein to describe the underlying population of the analysis, rather than the common use in EC referring to a set of candidate solutions.

of parameters of the distribution, then we have:

$$\mathbf{Pr}(\boldsymbol{\theta}|\mathbf{x}) = \frac{\mathbf{Pr}(\mathbf{x}|\boldsymbol{\theta})\mathbf{Pr}(\boldsymbol{\theta})}{\mathbf{Pr}(\mathbf{x})}, \tag{1}$$

where $p(\boldsymbol{\theta})$ denotes the prior distribution of the parameters, and $p(\boldsymbol{\theta}|\mathbf{x})$ stands for the *posterior probability* of the parameters given the data. We can use this posterior probability to obtain relevant information about the distribution (population) for where the data comes, such as the (posterior) expected value for X. Moreover, we can also derive information about the uncertainty of the estimations.

Several Bayesian methods have been proposed to compare algorithms' performance by analyzing the underlying empirical data [3, 4, 6]. However, to the best of our knowledge, all of them focus on the pairwise comparisons of algorithms. Despite the high relevance of these methods, they come short when attempting to address the following, very commonly asked, broad comparison question:

> Given a set of algorithms, and given their associated empirical data when run on a predefined set of problems: Which of those algorithms performs best for those problems?

One approach to address this question is to explicitly model the ranking of the algorithms. In [5] a first attempt to answer that question was made by proposing a Bayesian Plackett-Luce model to perform the comparative analysis of experimental results.

2.1 Bayesian Plackett-Luce model

The Plackett-Luce model is a probabilistic model defined over the space of permutations [21, 22], which in our context correspond to rankings of algorithms. Among the probabilistic models proposed for permutations (i.e., functions that assign a probability to each permutation) [18], the Plackett-Luce model collects some features that make it suitable to model rankings of algorithms. The first one is that it fulfills Luce's axiom, which states that the relative ordering of two algorithms is independent of the non-considered ones.[2]

The second interesting feature has to do with the interpretation of the set of parameters of the probability model. When modeling rankings of size k, the Plackett-Luce model has k parameters, one per algorithm. We will refer to these parameters as *weights*, w_i.

If we represent rankings as $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_k)$ where $\sigma_i = j$ means that the j-th algorithm is ranked in the i-th position, the probability of a given ranking σ under the Plackett-Luce model is:

$$\mathbf{Pr}_{PL}(\boldsymbol{\sigma}|\mathbf{w}) = \prod_{i=1}^{n} \frac{w_{\sigma_i}}{\sum_{j=i}^{n} w_{\sigma_j}}. \tag{2}$$

In general, the weights can take any strictly positive value. However, if we restrict them such that $\sum w_i = 1$,[3] then they can be interpreted as the probability of each algorithm being the best (the top-ranked one), i.e., $w_i = \mathbf{Pr}(\sigma_1 = i)$.

[2]To ease the reading, instead of using terms such as *item* or *element* to name the components of the permutations, from now on, we will refer to them as *algorithms*.
[3]Note that, due to the way the probabilities are computed, we can force the sum to be any positive value without loss of generality, as any common factor for the parameters will not affect the probabilities.

In what follows, we will briefly review the Bayesian Plackett-Luce model, for more details, the interested reader is referred to [5]. As in any Bayesian model, we identify the three distributions mentioned above: the likelihood function, the prior distribution, and the posterior distribution. In this model, the likelihood function is calculated as the product of the probabilities of the observed rankings (of algorithms) under the Plackett-Luce model. The prior distribution of the weights is modeled with a Dirichlet distribution, which is the generalization of the Beta distribution. Briefly, the Dirichlet distribution models probability distributions over real-valued vectors that sum 1 and, thus, it is particularly suitable to model the prior distribution of the weight vector. By representing the vector of weights as $\mathbf{w}$ and the data (the rankings of the algorithms) as $R = \{\boldsymbol{\sigma}^{(1)}, \ldots, \boldsymbol{\sigma}^{(N)}\}$, the posterior distribution of the weights can be computed as:

$$\mathbf{Pr}(\mathbf{w}|R) = \frac{\mathbf{Pr}(R|\mathbf{w})\mathbf{Pr}(\mathbf{w})}{\mathbf{Pr}(R)}. \tag{3}$$

The above equation has no closed-form, but we can easily sample the posterior distribution of weights using Markov Chain Monte Carlo methods.

The inference process is as follows. First, the performance data of the algorithms is collected, considering the scenario (the population) from which we want to draw our conclusions. The results are then transformed into rankings of algorithms (one per run) and these rankings are used to produce a sample from the posterior distribution of weights.

Once the posterior distribution of the weights has been defined, there are several aspects of the rankings' distribution represented by the model. We will focus on the probability of each algorithm being the top-ranked. There are other aspects related with the ranking of the algorithms that can be analyzed using the Bayesian Plackett-Luce model, such as the probability of an algorithm being among the two best algorithms or the expected rank for an algorithm. However, for space reasons we will limit the analyses to the two points mentioned above.

3 BENCHMARKING AND ANALYSIS PREPARATION

In this section we present the preliminaries for the Bayesian inference analysis – the details behind the actual benchmarking campaign (that is, the source of the generated datasets), as well as the analysis' planning and its associated practical considerations.

3.1 Black-Box Discrete Optimization Benchmarking

We build our statistical analyses upon performance data that has been obtained from an independent research thread [13], which is aimed at identifying suitable benchmark problems for discrete optimization heuristics and at showing that the `IOHprofiler` environment announced in [12] is capable of handling mid-sized benchmarking requirements. Next, we list the 23 optimization problems that form the current benchmarking suite:

- OneMax (F1) and W-model extensions (F4-F10)
- LeadingOnes (F2) and W-model extensions(F11-F17)
- Harmonic (F3): linear harmonic function

- LABS: Low Autocorrelation Binary Sequences (F18)
- Ising-Ring (F19)
- Ising-Torus (F20)
- Ising-Triangular (F21)
- MIVS: Maximum Independent Vertex Set (F22)
- NQP: N-Queens problem (F23)

The specified functions have been implemented altogether within the `IOHexperimenter`. Each function F is assessed over four problem dimensions, $n \in \{16, 64, 100, 625\}$, yielding altogether 92 (F, n) pairs. Each algorithm is run on 11 different instances of each of these 92 pairs, yielding 1,012 different runs per each algorithm. Each run is granted a budget of $100n^2$ function evaluations for dimensions $n \in \{16, 64, 100\}$ *versus* $5n^2$ evaluations for $n = 625$.

The following 11 EAs were benchmarked (see [13] for details):

(1) **gHC:** a greedy (1+1)-type hill climber which flips one bit per iteration, following a deterministic order.
(2) **RLS:** Randomized Local Search. Like gHC but flips one randomly chosen bit per iteration.
(3) (1 + 1) **EA:** The (1+1) EA with static mutation rate $p = 1/n$. Fips one random bit on average, but can escape local optima by flipping more than one bit.
(4) **fGA:** The "fast GA" proposed in [11]. The number of bits flipped follow a heavy-tail power-law distribution with exponent $\beta = 1.5$.
(5) (1+10) **EA:** The (1+10) EA with static $p = 1/n$. Like (1+1) EA but keeps best of ten offspring.
(6) (1+10) $\mathbf{EA}_{r/2,2r}$**:** The two-rate (1+10) EA with self-adjusting mutation rates. Taken from [10].
(7) (1 + 10) $\mathbf{EA}_{\text{norm.}}$**:** (1 + 10) EA sampling the number of bits to flip from an adaptive normal distribution. Taken from [27].
(8) (1 + 10) $\mathbf{EA}_{\text{var.}}$**:** The (1 + 10) $\text{EA}_{\text{norm.}}$ with variance control, also from [27].
(9) (1 + 10) $\mathbf{EA}_{\text{log-n.}}$ The (1+10) EA with log-normal self-adaptation of the mutation rate proposed in [1].
(10) (1 + (λ, λ)) **GA:** single-trajectory genetic algorithm using cross-over [9]. We use the variant with self-adjusting parameters analyzed in [8].
(11) **vGA:** A (30, 30) "vanilla" GA (see, e.g., [15]).

3.2 Analysis' Considerations and Planning

We illustrate the application of the Bayesian Plackett-Luce model by analyzing some of the performance data of the aforementioned benchmarking campaign. As described above, the numerical data encompass optimization results for 23 objective functions in 4 dimensions, each benchmarked over 11 instances by 11 competing algorithms. For each of these $23 \times 4 \times 11 \times 11 = 11,132$ runs, the following information is recorded: whenever the algorithm has identified a solution of strictly better quality than the previous-best solution, a new data entry is created which stores the new incumbent fitness value and the number of the fitness evaluations elapsed before sampling it.[4]

The analyses reported in Section 4 involve performance data of all eleven algorithms. Regarding the functions, we will show how to treat them either individually or collectively (in groups). When considered in groups, we will take two approaches – either considering the results altogether (i.e., aggregating the performances across the eleven instances of each function), or taking the median of these runs. As previously noted, we use the Bayesian Plackett-Luce model to estimate the probability of each algorithm being top-ranked.

In practice, we will estimate the probability of an algorithm being the best among its competitors as its expected weight in the posterior distribution of weights. Additionally, we will analyze the uncertainty about these probabilities by estimating the 90% credible intervals for which we will use the corresponding quantiles (5% and 95%) of the posterior distribution of weights.

Finally, we address one of the main concerns with Bayesian methods, which lies in the fact that they make use of a prior distribution for the parameters. Usually, this prior belief is set as a non-informative one, to which we assign a relatively low strength. During the Bayesian inference analysis the prior belief and the information within the data are merged. The impact of the prior belief decreases as the amount of available data increases. We will illustrate and discuss the effects of the prior distribution on the final estimations in Section 4.3.

4 INFERENCE ANALYSES AND RESULTS

We begin our report by providing a *qualitative* summary of the observations obtained from a manual examination of the benchmark data. This report is followed by a rigorous, *quantitative* Bayesian analysis that is meant to address some of the qualitative statements with the goal to corroborate or to refute them.

As an illustration of the data visualization used to support the qualitative analysis, we depict in Fig. 1 the Expected Running Time (ERT) values for all 11 algorithms and all 23 functions in dimension $n = 625$. The ERT values are with respect to the best solution found by any of the (algorithm, instance) pairs. Importantly, the nature of the ERT map could substantially change upon setting alternative target values. Also note that the different instances of each problem adhere altogether to the same fitness landscape, which just undergoes isomorphic transformations (rotation, scaling, etc., see [13] for details). For the purpose of our analysis, we can thus regard the data for the eleven instances as eleven independent runs per each algorithm on a given problem. The following list summarizes some of the conclusions drawn in [13].

- The following class of algorithms exhibits similar performance profiles over the majority of problems: the (1 + (λ, λ))-GA, (1+1)-EA, (1+10)-$\text{EA}_{\text{var.}}$, (1 + 10) $\text{EA}_{\text{log-n.}}$, (1+10)-EA, (1+10)-$\text{EA}_{\text{norm.}}$, (1+10)-$\text{EA}_{r/2,2r}$, and (1+1)-fGA. These algorithms behave quite consistently, typically exhibiting fine performance.
- The gHC and the vGA usually exhibit extreme performance with respect to the other algorithms. The vGA consistently suffers from poor performance over all functions, while the gHC either leads the performance on certain functions or shows deteriorated performance on others.
- When counting the number of runs in which a solution of overall best target value has been identified per algorithm, the (1+10)-$\text{EA}_{r/2,2r}$ leads the hitting scores on the "low-dimensional" functions ($n = 16,\ 64$), with the (1+1)-fGA and

[4] More precisely, `IOHexperimenter` tracks much more data, but all computations in this work are solely based upon the here-described records.

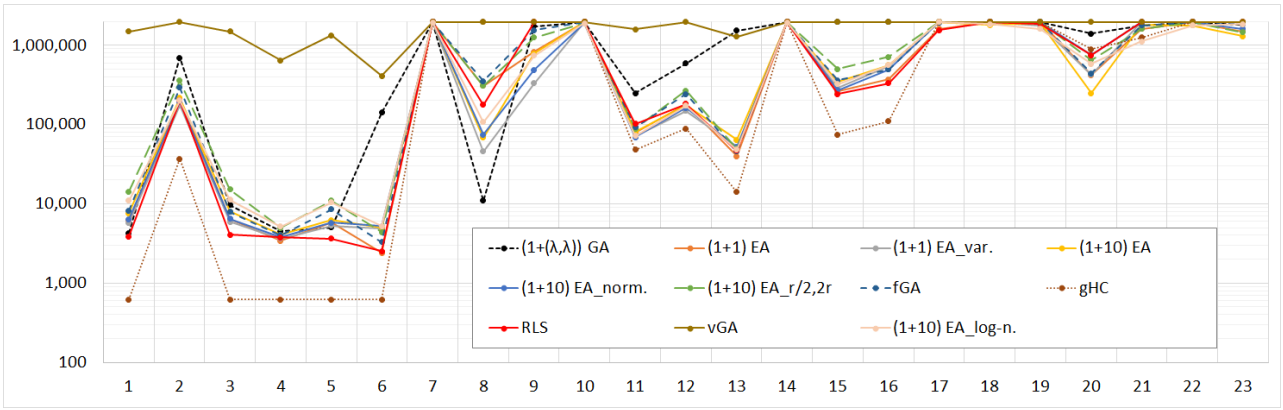


Figure 1: ERT values of all 11 algorithms for the 625-dimensional test suite, with respect to the best solution quality found by any of the algorithms in any of the eleven runs.

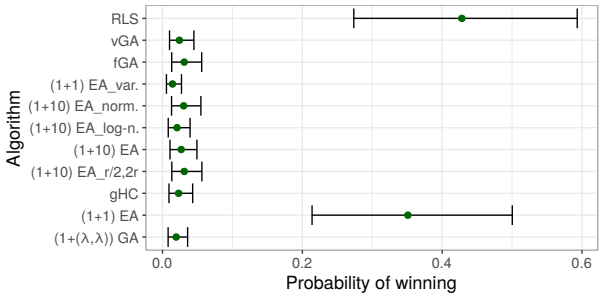
the (1 + 10) $EA_{log\text{-}n.}$ being the first runner-up on $n = 16$ and the (1+1)-fGA and the (1+10)-$EA_{norm.}$ being second for $n = 64$. For $n = 100$ the (1 + 10) $EA_{log\text{-}n.}$ has the best success rate, with (1+10)-$EA_{r/2,2r}$ and (1+1)-fGA being the runner-up. The (1+10)-EA also exhibits fine ranking across all dimensions. On the other hand, the (1+1)-EA leads the hitting scores on the "high-dimensional" functions at $n = 625$, with the (1+10)-EA being the runner-up. When summing over all dimensions, gHC, vGA, and RLS are with the lowest scores.

- Function-wise, it is evident that the low-dimensional F1-F6, F8, F11-F13 and F15-F16 are easily treated by the majority of the algorithms.

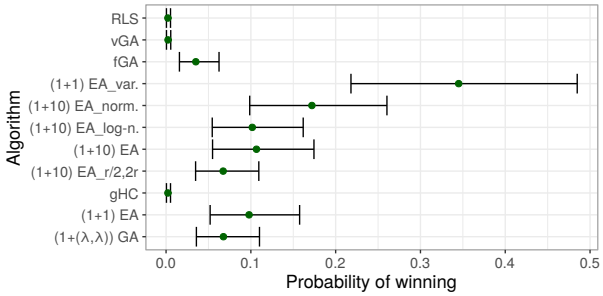
4.1 Fixed-Targets Perspective

As a first analysis, we will demonstrate algorithms' ranking by considering the results of a particular function in a certain dimension and upon setting a fixed target value φ, denoted as a triplet (F, n, φ). This subset of the data defines running results for the 11 algorithms, and in particular, it contains the recorded running time (i.e., number of evaluations) consumed by each algorithm to hit the prescribed target. Then, for each run, we rank the algorithms according to this running time – where the top-ranked algorithm is defined as the first algorithm to hit the target. Importantly, the distribution that undergoes inference is the rankings we obtain in random runs of the 11 algorithms for the selected (F, n, φ)-triplet. Fig. 2 depicts two examples of the credible intervals and expected values obtained for the probability of each algorithm being the best.

These plots illustrate how the Bayesian method does not only provide us with the expected probability, but also with information about its associated uncertainty. Taking into account that we have only 11 samples to do inference, we should expect relatively high uncertainty about the probability of winning. Clearly, this is the case for certain algorithms, such as the two best algorithms in the first example (Fig. 2 (a)). However, for those algorithms that have the smallest probability of winning, 11 samples can be enough to get a quite tight interval for the probability. In general, it is hard to say how many runs are needed to get confident results. However, in



(a) $F17$, $n = 625$, $\varphi = 625$



(b) $F9$, $n = 100$, $\varphi = 100$

Figure 2: Credible intervals (5% and 95% quantiles) and expected probability of winning (green dots) for two combinations of function, dimension and target.

Bayesian statistics we can empirically observe how the uncertainty decreases as the number or runs is increased – one of the strong points where the Bayesian inference approach shows its potential.

Another key argument in favor of using Bayesian methods is that they provide us with *interpretable* information. As an example, attending to the results produced by the analysis above we can state that if we run all the 11 algorithms on F17, dimension $n = 625$ and fix the target to $\varphi = 625$, then the estimated expected

probability of RLS being the fastest algorithm hitting the target is 0.426, which is almost 15 times higher than the expected probability for gHC (that reads 0.03). Moreover, the results above estimate that, with probability 0.9, the true probability of RLS being the fastest algorithm lies in the interval [0.272, 0.585]. Statistical tests can also be used to estimate and then present uncertainty in the form of confidence intervals. Although this is not the common practice when comparing algorithms, a binomial test could be used to get intervals for the probability of being the best, but comparing this approach with the proposed Bayesian method is out of the scope of this work. Importantly, mind should be given to the different interpretation of these two approaches (see [16, p.343]).

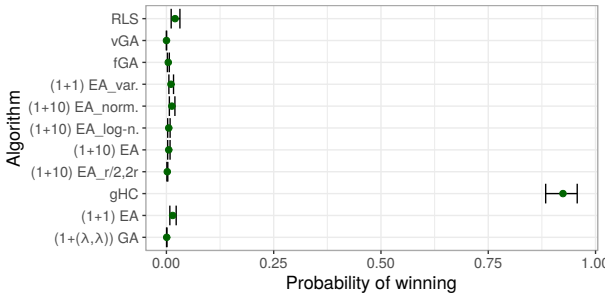
The plots in Fig. 2 show that all the algorithms have non-zero probabilities of being the best, even if some of them never arise as best in the data. One could think that this is due to the prior distribution, which assigns equal probabilities to all the algorithms before the data is incorporated into the model. However, the reason is not rooted in the prior, which has a negligible impact, as will be demonstrated in Section 4.3. Rather, the Plackett-Luce model itself is responsible for this behavior, since its parameters (i.e., the weights associated with each algorithm), reflect the relative ordering of all the algorithms, and not only the probability of being the best.

So far we have focused on individual functions and, therefore, our conclusions are limited to the analyzed functions. In some situations, we may have a set of instances about which we want to do inference. In such cases, our sample will be the results obtained by the algorithms in a sample of instances. For each instance we can have different runs of the algorithms and, thus, there are two possible paths to follow. The common practice is to average the results of the different runs – yielding a single value for each instance and algorithm. The simplest way of averaging the runs is by taking the mean value, but due to the high risk of not having unimodal, symmetric distributions for the data, a safer approach is to take the median. An alternative way consists in taking all the runs at the same time.

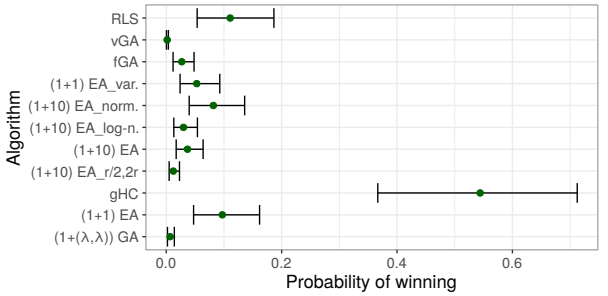
Both strategies are sensible, but they are not equivalent, as the interpretation of the results changes. The difference in the interpretation is rooted in the underlying distribution behind the analyzed samples. Let us assume that the instances at hand constitute a random sample from that statistical population.[5] Then, upon taking the average value, the underlying statistical population becomes the *average* results obtained by the algorithms in a random instance from the population of instances. Conversely, if all values are accounted for, the underlying population becomes the result of the algorithms in random runs over random instances (again, having exactly the same number of repetitions for each instance is not a proper sample, but the results may be close enough to those obtained with a proper sample).

We will illustrate the differences between these two approaches upon comparing the algorithms' performance over two subsets of the functions at $n = 625$, corresponding to an "easy" subset of functions ($F \in \{1-6, 11-13, 15, 16\}$) versus the complementary subset of "non-easy" functions.

[5]In practice, the instances are often not randomly sampled but rather selected based on some criteria. Formally, this is not a correct way of proceeding if the goal is doing inference. Nevertheless, in many practical situations defining a statistical population of instances is not feasible and, thus, we can take the selected instances as a sample from a certain population.



(a) Easy functions, all runs



(b) Easy functions, medians

Figure 3: Credible intervals (5% and 95% quantiles) and expected probability of winning (green dots) for the "easy" subset of functions at $n = 625$. Plot (a) represents the results using all the runs, whereas plot (b) has been obtained with the median of the runs.

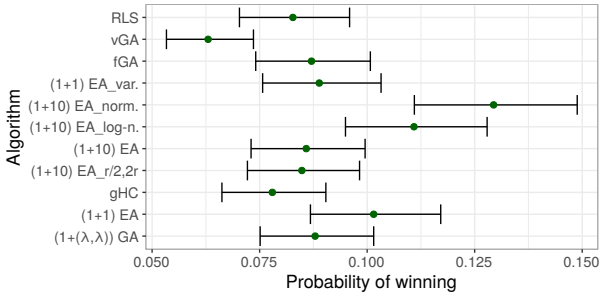
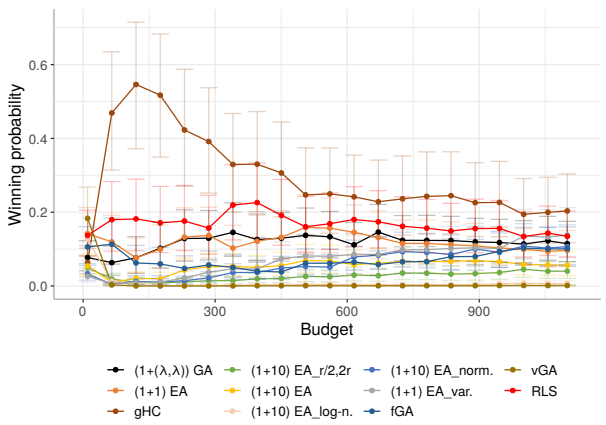
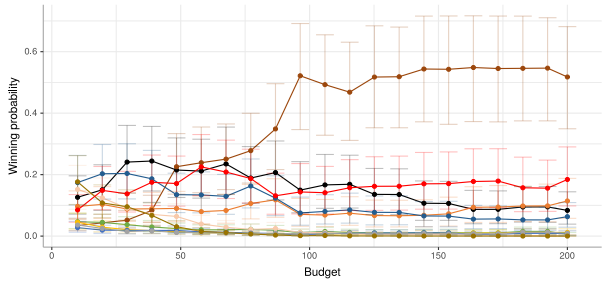


Figure 4: Credible intervals (5% and 95% quantiles) and expected probability of winning (green dots) for the "non-easy" subset of functions at $n = 625$, based on all the runs. Note the relatively small probability scale and the tight credible intervals.

Fig. 3 shows the results for the "easy" functions. Clearly, the best performing algorithm in both cases is gHC. When we analyze random runs in random instances (Fig. 3 (a)), we can see that there is a big difference between gHC and the rest. Evidently, the credible intervals are quite tight, suggesting a considerable reduction of the



(a) $F21$, $n = 100$



(b) $F22$, $n = 100$

Figure 5: Evolution of the probability of winning for different budgets. The lines represent the evolution of the expected probability of winning while the error bars represent the 90% credible intervals (5% and 95% quantiles).

uncertainty. This is, mostly, because we have a larger sample than in the second case (11 samples versus 1 per function). Fig. 4 shows the results for all the runs in the "non-easy" subset of functions. In this case we get a somewhat different picture: the estimation of the probabilities is fairly good (the credible intervals are smaller than 0.05, whereas mind should be given to the plot's scaling), but the probability values are fairly similar (simply due to overlapping). Overall, there is an uncertainty about which algorithm is best performing over this subset of functions, but not due to limitation of data, rather due to equivalence in the algorithms' performance.

4.2 Fixed-Budget Perspective

A different perspective to empirically analyze the algorithms' runs considers comparisons that are drawn per a fixed budget. The proposed Bayesian method can be used to compare the algorithms also in this scenario. In order to show a different approach to the analysis, we will demonstrate the method by tracking the *evolution of the probability* of each algorithm being the best as the budget increases, which is a novel perspective to the best of our knowledge. Fig. 5 presents two examples of such probabilities' evolution plots, for different cases.

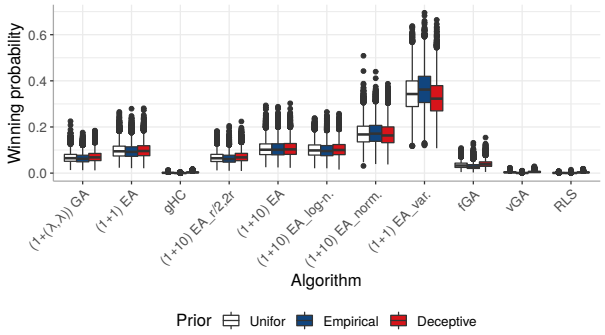


Figure 6: Comparing the posterior distribution of weights, when obtained with different prior distributions, for the data analysis corresponding to $F9$, $n = 100$, $\varphi = 100$.

In Fig. 5(a), showing results for F21, it is evident that the behavior of one of the algorithms, gHC, differs from the rest. Its probability of winning quickly increases, showing that for low budgets it is clearly the best algorithm. However, although fine objective function values are quickly acquired by gHC, other algorithms obtain better values as the budget increases, and accordingly, their probability of winning increases, thus reducing the probability of gHC being the best. The picture is quite different when analyzing F22, as depicted in Fig. 5(b): the gHC exhibits steady increase in its probability of being the best, up to a probability of about 50% (a high value, taking into account that we consider altogether 11 algorithms).

4.3 Impact of the Prior Distribution

One of the main concerns about Bayesian statistics is the need to define a prior distribution for the parameters of the model. For this reason, an important aspect to investigate is the impact of the prior distribution over the final estimations. In this section we will illustrate this type of analysis using one of the previous examples. In particular, we will compare the posterior distribution of the weights obtained in the analysis of the results for function $F9$ at $n = 100$ with a target value $\varphi = 100$ (see Fig. 2).

To this end, we have used three different types of prior distributions: (i) equal treatment of all the algorithms (uniform prior), (ii) preferred probability to the best algorithms (assigning the best algorithm higher prior probability of winning), and (iii) preferred probability to the worst-performing algorithms (deceptive prior). Distributions (ii) and (iii) make use of the empirical results to determine which algorithms have a higher prior probability of being the best.

Fig. 6 shows the results of the comparison as boxplots of the posterior sample of weights. Evidently, when the empirical data is used to bias the prior distribution toward the best algorithms, the two best algorithms have slightly higher values and the opposite happens when the deceptive prior is used, as they have slightly smaller values. In any case, the actual differences are practically negligible, even if we have a relatively small sample (11 runs). These small differences are translated into small differences in our estimations, but they definitely will not alter the general comparison overview.

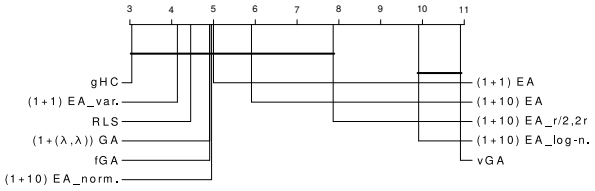
Although the common practice is to use non-informative priors, one nice property of Bayesian statistics is that it provides a natural way to do "incremental" analyses. That is, rather than using a non-informative prior to start the comparison of the algorithms from scratch, we can use previous analyses to set an "informed" prior, thereby testing the algorithms in a loop fashion.

4.4 Comparison with Classical Statistical Tests

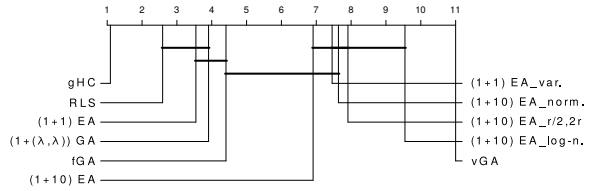
Finally, as an illustration of the differences between Bayesian inference and the classical statistical test approach, we will show the results obtained with a standard statistical test workflow for the two largest budgets in Fig. 5. All the pairwise comparisons are tested using a Wilcoxon test and the p-values are corrected for multiple testing using Shaffer's method [14]. The results are displayed in Fig. 7. The plots are similar to the Critical Differences plot presented in [7]. Each algorithm is displayed according to its average rank in the data. Then, the algorithms which, according to the test, do not significantly differ (corrected p-value above 0.05) are linked by a horizontal line.

The results shown in these plots, although reflecting a different perspective on the data, match those observed in Fig. 5. For the two functions, the ordering of the algorithms is roughly the same in both approaches. Regarding the uncertainty, it is too high for function F21 to conclude that any of the algorithms is clearly the best. Conversely, for F22 both analyses show that gHC is clearly better than the rest of the algorithms (it has significant differences with the second one and the credible intervals in Fig. 5 do not overlap).

In the case of the Bayesian analysis we have the estimated probability of each algorithm being the best, together with a notion (in the form of credible interval) of the uncertainty about this estimation. Moreover, the Bayesian analysis can also be used to estimate the expected rank (similar, in this example, to the ones directly derived form the data; these calculations are not included herein for reasons of space).



(a) $F21$, $n = 100$, Budget 1100



(b) $F22$, $n = 100$, Budget 200

Figure 7: Demonstration of the statistical test workflow applied to the results for two problem instances for a fixed budget. The plots present the average rank of each algorithm, and also display which algorithms do not exhibit statistically significant differences (linked by a horizontal line).

5 DISCUSSION AND OUTLOOK

An important task in Computer Science in general, and in EC in particular concerns the evaluation and comparison of algorithms. Two critical steps in such an assessment comprise an appropriate design of experimentation as well as a proper statistical analysis of the performance data. Different statistical procedures and methods have been suggested in the literature to gain insight into particular aspects of algorithms' performance. In this work we have demonstrated that Bayesian inference using the Plackett-Luce model facilitates an effective analysis of algorithms' performance ranking. Since these results nicely complement classical statistical tests, we suggest to include Bayesian inference with the Plackett-Luce model as a standard statistical tool in the practical EC performance comparisons' repertoire. In this vein, we also suggest to include a Bayesian inference analysis module in IOHanalyzer, the data-analysis component of IOHprofiler.

The Bayesian Plackett-Luce model described in this work has some strong points (e.g., the ability to handle multiple algorithms), but – as with any statistical tool – also some weaknesses. It is therefore important to understand the limits of the model. An important point to note is that by aggregating performances into rankings we lose information about the magnitude of the differences. Put differently, the model is capable of estimating the probability of an algorithm being the best, but it does not quantify the performance gap with respect to the first runner-up. Developing new statistical procedures to inspect the results from other, alternative points of view, forms an important line for future research.

Another important line of further research concerns a deeper analysis of the Bayesian Plackett-Luce model – not only regarding aspects of convergence of the estimators, but also in terms of identifying other useful metrics. An example for such metrics is the probability of an algorithm being among the top k algorithms, which has the potential to enrich the algorithms' comparison.

Finally, an ambitious but a necessary goal involves the careful definition of the most appropriate workflow for the comparisons. There is no single method that is best suited to answer all the questions we may ask, and not all the methods require the same type of data. Thus, clearly articulating the questions, devising the appropriate methods to address them, and formulating the data's requirements for their application – are tasks from which the community can greatly benefit.

ACKNOWLEDGMENTS

This work has been partially supported by the BERC 2018-2021 and ELKARTEK programs (Basque Government), and the project TIN2016-78365-R and Severo Ochoa Program SEV-2017-0718 (Spanish Ministry of Economy, Industry and Competitiveness). Research was also supported by COST Action CA15140 "Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO)".

REFERENCES

[1] Thomas Bäck and Martin Schütz. 1996. Intelligent Mutation Rate Control in Canonical Genetic Algorithms. In *International Symposium on Foundations of Intelligent Systems (ISMIS'96) (Lecture Notes in Computer Science)*, Vol. 1079. Springer, 158–167.

[2] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. 2017. Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis. *Journal of Machine Learning Research* 18, 77 (2017), 1–36.

[3] Alessio Benavoli, Giorgio Corani, Francesca Mangili, Marco Zaffalon, and Fabrizio Ruggeri. 2014. A Bayesian Wilcoxon signed-rank test based on the Dirichlet process. *Proceedings of the 31 International Conference on Machine Learning* 32, 1026–1034.

[4] Alessio Benavoli, Francesca Mangili, Fabrizio Ruggeri, and Marco Zaffalon. 2015. Imprecise Dirichlet process with application to the hypothesis test on the probability that $X \leq Y$. *Journal of Statistical Theory and Practice* 9 (2015), 658–684.

[5] Borja Calvo, Josu Ceberio, and Jose A. Lozano. 2018. Bayesian Inference for Algorithm Ranking Analysis. In *Proc. of Genetic and Evolutionary Computation Conference, Companion (GECCO'18)*. ACM, 324–325. https://doi.org/10.1145/3205651.3205658 Extended version available at http://www.sc.ehu.es/ccwbayes/members/borxa/bPL/.

[6] Cassio P. de Campos and Alessio Benavoli. 2016. Joint Analysis of Multiple Algorithms and Performance Measures. *New Generation Computing* 35, 1 (2016), 69–86.

[7] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 7 (Dec. 2006), 1–30.

[8] Benjamin Doerr and Carola Doerr. 2018. Optimal Static and Self-Adjusting Parameter Choices for the $(1+(\lambda, \lambda))$ Genetic Algorithm. *Algorithmica* 80 (2018), 1658–1709.

[9] Benjamin Doerr, Carola Doerr, and Franziska Ebel. 2015. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science* 567 (2015), 87–104.

[10] Benjamin Doerr, Christian Gießen, Carsten Witt, and Jing Yang. 2017. The $(1 + \lambda)$ Evolutionary Algorithm with Self-Adjusting Mutation Rate. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'17)*. ACM, 1351–1358.

[11] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. 2017. Fast genetic algorithms. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'17)*. ACM, 777–784. https://doi.org/10.1145/3071178.3071301

[12] Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, and Thomas Bäck. 2018. IOHprofiler: A Benchmarking and Profiling Tool for Iterative Optimization Heuristics. *arXiv e-prints:1810.05281* (Oct. 2018). arXiv:1810.05281 https://arxiv.org/abs/1810.05281

[13] Anonymous for double-blind reasons. 2019. Benchmarking Discrete Optimization Heuristics with IOHprofiler. under submission.

[14] Salvador García and Francisco Herrera. 2008. An extension on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons. *Journal of Machine Learning Research* 9 (2008), 2677–2694.

[15] David Goldberg. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, Reading, MA.

[16] Sander Greenland, Stephen J. Senn, Kenneth J. Rothman, John B. Carlin, Charles Poole, Steven N. Goodman, and Douglas G. Altman. 2016. Statistical tests, P values, confidence intervals, and power: a guide to misinterpretations. *European Journal of Epidemiology* 31, 4 (2016), 337–350.

[17] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík. 2010. Comparing Results of 31 Algorithms from the Black-box Optimization Benchmarking BBOB-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '10)*. ACM, New York, NY, USA, 1689–1696. https://doi.org/10.1145/1830761.1830790

[18] Ekhine Irurozki. 2014. *Sampling and Learning Distance-based Probability Models for Permutation Spaces*. Ph.D. Dissertation. Faculty of Computer Science, University of the Basque Country.

[19] Gopal K. Kanji. 2006. *100 Statistical Tests*. SAGE Publications.

[20] Dennis V. Lindley. 1965. *Introduction to Probability and Statistics from a Bayesian Viewpoint: Inference*. Vol. 2. Cambridge University Press, London, UK.

[21] Duncan Luce R. 1959. *Individual Choice Behavior*. Wiley, New York.

[22] Robin L. Plackett. 1975. The Analysis of Permutations. *Journal of the Royal Statistical Society* 24, 10 (1975), 193–202.

[23] Jeremy Rapin and Olivier Teytaud. 2018. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad.

[24] Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, and Tomasz Sternal. 2016. Optil.io: Cloud Based Platform For Solving Optimization Problems Using Crowdsourcing Approach. In *Proc. of ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW'16), Companion Volume*. ACM, 433–436. https://doi.org/10.1145/2818052.2869098

[25] Thomas Weise. 2016. Optimization Benchmarking. http://optimizationbenchmarking.github.io/

[26] Mark Wineberg. 2018. Introductory Statistics for EC: A Visual Approach. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '18)*. ACM, New York, NY, USA, 649–684. https://doi.org/10.1145/3205651.3207872

[27] Furong Ye, Carola Doerr, and Thomas Bäck. 2019. Interpolating Local and Global Search by Controlling the Variance of Standard Bit Mutation. In *Proc. Conference on Evolutionary Computation (CEC'19)*. To appear. Available online at http://arxiv.org/abs/1901.05573.