On the fair comparison of optimization algorithms in different machines

Etor Arza[a], Aritz Pérez[a], Josu Ceberio[b], Ekhiñe Irurozki[c]

[a] *BCAM - Basque Center for Applied Mathematics, Spain*

[b] *University of the Basque Country UPV/EHU, Spain*

[c] *Télécom Paris, France*

Abstract

An experimental comparison of two or more optimization algorithms requires the same computational resources to be assigned to each algorithm. When a maximum runtime is set as the stopping criterion, all algorithms need to be executed in the same machine if they are to use the same resources. Unfortunately, the implementation code of the algorithms is not always available, which means that running the algorithms to be compared in the same machine is not always possible. And even if they are available, some optimization algorithms might be costly to run, such as training large neural-networks in the cloud.

In this paper, we consider the following problem: how do we compare the performance of a new optimization algorithm B with a known algorithm A in the literature if we only have the results (the objective values) and the runtime in each instance of algorithm A? Particularly, we present a methodology that enables a statistical analysis of the performance of algorithms executed in different machines. The proposed methodology has two parts. Firstly, we propose a model that, given the runtime of an algorithm in a machine, estimates the runtime of the same algorithm in another machine. This model can be adjusted so that the probability of estimating a runtime longer than what it should be is arbitrarily low. Secondly, we introduce an adaptation of the one-sided sign test that uses a modified p-value and takes into account that probability. Such adaptation avoids increasing the probability of type I error associated with executing algorithms A and B in different machines.

Keywords: Algorithms, Optimization, Benchmarking, Statistical Tests

Email addresses: earza@bcamath.org (Etor Arza), aperez@bcamath.org (Aritz Pérez), josu.ceberio@ehu.eus (Josu Ceberio), irurozki@telecom-paris.fr (Ekhiñe Irurozki)

1. Introduction

Finding an appropriate reference point for evaluating the performance of an optimization algorithm is not trivial. The key question is: when can we say that the performance of an optimization algorithm is good? The answer depends on how we define good performance. A possible solution is to compare the performance of several algorithms in the same problem. This comparison can show that some algorithms perform better than others on average. Precisely, this is what a comparative study among different algorithms tries to accomplish: analyze the relative performance of a set of algorithms.

It is crucial, for the sake of fairness, that the optimization algorithms being compared are given the same computational resources (see AppendixA for an illustrative example). We say that two algorithms use the same amount of resources when they both take the same time to complete in the same machine. Usually, this is achieved by executing both algorithms in the same machine and by setting a common maximum runtime as the stopping criterion. Unfortunately, it is not always possible to execute all the algorithms being compared in the same machine, as, in many cases, the code for some of the algorithms is not available. Even when the code is available, executing all the algorithms might involve wasting a lot of computational resources.

Overcoming this limitation, a comparative study can still be carried out when the results of the experimentation of an algorithm, the CPU model, and the maximum runtime used to obtain these results are known. Dominguez et al. [7] proposed adjusting the runtimes of the algorithms by assigning a shorter runtime to the algorithms executed in faster machines, thus making the execution of algorithms in different machines comparable.

Let us now look at a practical example. Let us imagine that a researcher reads a paper in which algorithm A is executed in machine M_1, taking time t_1. Now, the researcher wants to compare a new algorithm, B, with A, but has no access to algorithm A nor to machine M_1. Instead, the researcher only has access to machine M_2 and algorithm B. In this case, he/she can execute algorithm B in machine M_2 for time t_2. The runtime t_2 needs to be adjusted in such a way that executing B in M_1 is equivalent to executing it in M_2 for time t_2. From now on, we will refer to the adjusted runtime as *equivalent runtime*.

The exact equivalent runtime t_2 can be obtained if the exact optimization process that was carried out in machine M_1 is replicated in machine M_2. This implies executing algorithm A in machine M_2, which defeats the purpose of using an equivalent runtime. Fortunately, an estimation of the equivalent runtime $\hat{t}_2$ can be used instead. The estimation is carried out taking into account the computational capabilities of machines M_1 and M_2, denoted as s_1 and s_2 in the rest of the paper. Table 1 offers a brief overview of the terms used in the paper.

Related work: Domínguez et al. [7] proposed the following regression model called the Reciprocal Mixed Inhibition function to estimate the equivalent run-

A summary of the notation of the paper

Name	Notation	Explaination
Optimization algorithm A	A	The optimization algorithm that is not executed in the comparison. Instead, already published results of this algorithm are used in the comparison.
Optimization algorithm B	B	The optimization algorithm that is executed to obtain the results to be compared.
Machine M_1	M_1	The machine in which algorithm A was executed. We have no access to this machine.
Machine M_2	M_2	The machine in which algorithm B is executed to obtain the results used in the comparison.
The score of machine M_1	s_1	A measure proportional to the computational capability of machine M_1.
The score of machine M_2	s_2	A measure proportional to the computational capability of machine M_2.
The runtime of A in machine M_1	t_1	The stopping criterion (in terms of maximum runtime) that was used in the execution of algorithm A in machine M_1.
The equivalent runtime of A in machine M_2	t_2	The time it takes to replicate in machine M_2 the exact optimization process (with algorithm A) that took time t_1 in machine M_1.
The estimated equivalent runtime of algorithm A in machine M_2	$\hat{t_2}$	An estimation of the equivalent runtime t_2. This value is used as the stopping criterion of algorithm B, which is executed in machine M_2.

Table 1: A summary of the notation and terms considered in this paper.

time of an algorithm:

$$\hat{t_1}(s_1, t_0) = \left(\frac{as_1}{b(1+\frac{t_0}{c})} + s_1 + \frac{s_1 t_0}{d} \right)^{-1}, \hat{t_2}(s_2, t_0) = \left(\frac{as_2}{b(1+\frac{t_0}{c})} + s_2 + \frac{s_2 t_0}{d} \right)^{-1} \tag{1}$$

where s_1, s_2 are the scores of machines M_1, M_2 (measured as the *dhrystone2* score [13] of the CPU) in which the algorithm is executed, t_0 represents the base or hardware-independent complexity of the optimization process (the complexity of the exact sequence of computational operations carried out throughout the optimization process) and a, b, c, d are the four parameters to be fitted. Given the runtime of algorithm A in machine M_1 (by replacing $\hat{t_1}$ with t_1), the base runtime of the algorithm t_0 can be isolated from Equation (1). Finally, the equivalent runtime of the algorithm in another machine, $\hat{t_2}$, can be estimated with this baseline runtime t_0 and the speed of the other machine s_2. It is noteworthy that, unfortunately, the methodology introduced by Domínguez et al.(2012) does not take into account that the estimated equivalent runtime can be longer or shorter than the real equivalent runtime, and thus, may introduce undesired biases to the comparison of the performance of algorithms.

This can be a major problem because this bias can increase the probability of type I error when deciding if algorithm B is better than A. In the context

of performance comparison of optimization algorithms (with null-hypothesis statistical tests), a type I error is defined as finding a statistically significant difference in the performance of the algorithms, when in reality, there is none. Making a type II error, on the other hand, means not finding a statistically significant difference in the performance of the algorithms, when in reality, the performances are different. In the context of the comparison of optimization algorithms, making a type II error is preferred to a type I error: falsely concluding a nonexistent difference in the performance of the algorithms is worse than not finding a statistically significant difference in the performance of the algorithms[1].

Proposed methodology: In this paper, inspired by the work of Dominguez et al. [7], we propose a methodology to statistically assess the difference in the performance of optimization algorithms executed in different machines. Specifically, the proposed methodology can be used to show that an algorithm B performs statistically significantly better than another algorithm A, without executing A and, instead, using the available results of A in terms of the objective function value and the runtime in each instance. To that end, we propose a conservative methodology in which the probability of giving algorithm B an unfairly longer time is kept in check by i) proposing a two-parameter estimation of the equivalent runtime with an arbitrarily low probability of estimating an unfairly longer runtime and ii) by modifying the one-sided sign test [6] so that it takes this probability into account.

Alongside this paper, we present a tutorial on how to apply the proposed methodology. This tutorial and the code of all the experimentation is available in our GitHub repository[2]. Besides, we also give two examples of how the methodology is applied in this paper. It is noteworthy that applying the proposed methodology to compare algorithms in different machines does not involve executing any additional code.

The rest of the paper is organized as follows: The next section describes and motivates a two-parameter model proposed to estimate the equivalent time. Section 3 presents the modifications made to the sign test to overcome the limitations introduced by the execution of the algorithms in different machines. Afterward, in Section 4 we introduce two examples in which we apply the proposed methodology. Finally, Section 5 concludes the paper and proposes some research lines for future investigation.

[1] Failing to reject the null hypothesis does not imply evidence in favor of the null hypothesis. Instead, it only shows a lack of evidence against it. In our context, failing to reject the null hypothesis does not mean that the performance of the algorithms is the same. The correct conclusion is, in this case, that there is not enough evidence to show a statistically significant difference between the performance of the algorithms. Therefore, a type II error just means that additional experimentation is needed to verify an existing difference in the performance of the algorithms, which is not an erroneous conclusion in itself.

[2] Repository available in https://github.com/EtorArza/RTDHW.

2. The estimation model of the equivalent runtime

Given i) an optimization algorithm, ii) a machine, iii) a problem instance, iv) a stopping criterion and v) a random seed number, executing the optimization algorithm will produce a specific sequence of computational instructions. This sequence is completely determined by these five parameters. We call this sequence of instructions that is reproducible in any machine *optimization process*. By recording the optimization process carried out with these parameters, we can later reproduce the exact optimization process in another machine. Notice that reproducing the optimization process will take a different time in each machine, even though the final result is the same (because the executed sequence of instructions is the same). We say that the times required to replicate the same optimization process in different machines are equivalent.

Definition 1. *(Optimization process)*
Let M be a machine, A an optimization algorithm, i a problem instance, t_1 a stopping criterion, and r a positive integer (the seed for the random number generator). We define the optimization process $\rho(M, A, i, t_1, r)$ as the sequence of computational instructions carried out when optimizing instance i with algorithm A and seed r in machine M_1 with stopping criterion t_1.

The aim is to compare algorithm A executed in machine M_1, with algorithm B, executed in machine M_2. A fair comparison can be carried out by estimating the time it takes to replicate $\rho(M_1, A, i, t_1, r)$ in machine M_2 and using the estimated value as the stopping criterion for algorithm B in machine M_2. We will sometimes denote the optimization process $\rho(M_1, A, i, t_1, r)$ as ρ, for the sake of brevity.

Definition 2. *(Runtime of an optimization process)*
Let M be a machine and ρ an optimization process. We define the runtime of ρ in M, denoted as $t(M, \rho)$, as the time it takes to carry out the optimization process ρ in machine M.

Considering the above definitions, it follows that, $t(M_1, \rho) = t_1$.

Definition 3. *(Equivalent runtime)*
Let M_1, M_2 be two machines, ρ an optimization process and $t(M_1, \rho)$ and $t(M_2, \rho)$ the times required to run ρ in M_1 and M_2 respectively. Then, we say that $t(M_2, \rho)$ is the equivalent runtime of $t(M_1, \rho)$ for machine M_2.

From here on, we will denote $t(M_2, \rho)$, the equivalent runtime of $t_1 = t(M_1, \rho)$ in machine M_2, as t_2.

Given t_1 (the runtime of optimization process ρ in a machine M_1), in the following, we will propose a model to estimate t_2 (the equivalent runtime in another machine M_2). The proposed estimation model is based on an assumption that

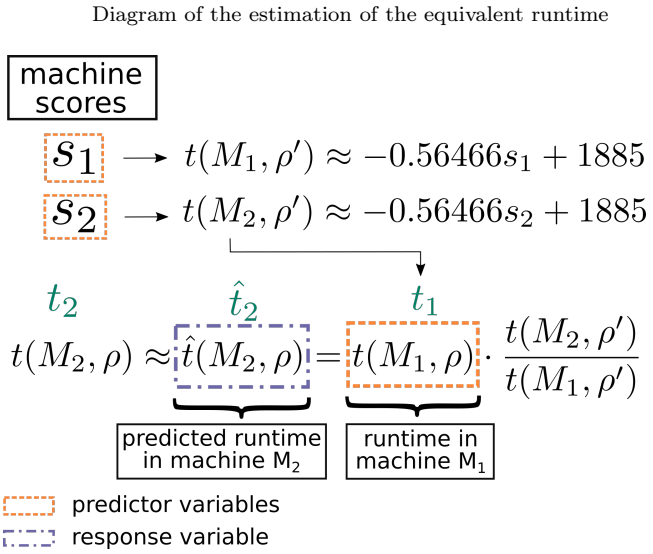

Figure 1: Estimated equivalent runtime of ρ in machine M_2 (the response variable $\hat{t}(M_2, \rho)$). The estimation is carried out with three predictor variables: the machine scores s_1 and s_2 and $t(M_1, \rho)$.

the ratio of the runtime of two different optimization processes is constant with respect to the machine in which it is measured (in AppendixB, we empirically show why this assumption is reasonable).

Assumption 1. *(Constant ratio of the runtime of two optimization processes) Let ρ, ρ' be two optimization processes. Then, we assume that:*

$$\frac{t(M_2, \rho)}{t(M_2, \rho')} \approx \frac{t(M_1, \rho)}{t(M_1, \rho')}$$

for any two machines M_1 and M_2.

Based on this assumption, we propose a model to estimate the equivalent runtime of an optimization process in a machine, given its runtime in another machine, as well as the scores (relative to the computational capabilities) of both machines. Notice that in Assumption 1, we use a reference optimization process ρ' to estimate the equivalent runtime of the optimization process ρ. Any optimization process ρ' can be used as a reference. In the following, we will define an optimization process ρ' whose runtime we will be able to estimate with the scores s_1 and s_2 of the machines. This will allow the estimation of the equivalent runtime t_2 without executing any reference optimization processes, as shown in Figure 1.

Let us now define the optimization process ρ', whose runtime can be estimated. Recall that an optimization process is just a sequence of computational instruc-

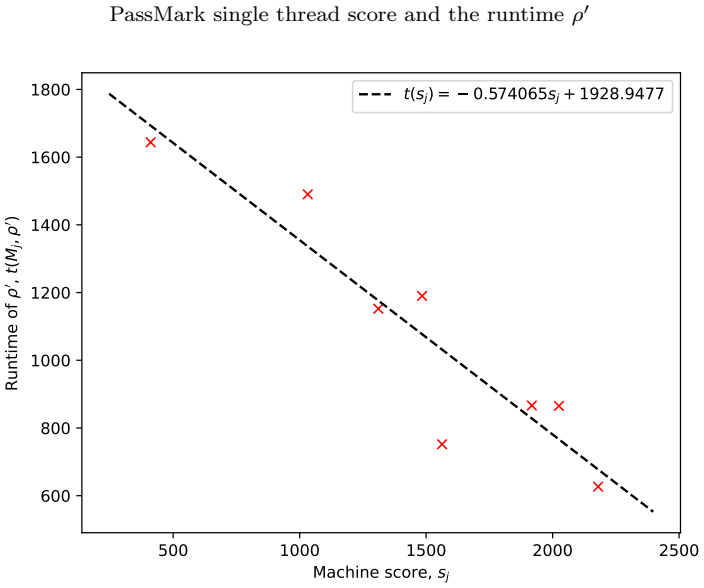

Figure 2: The runtime of ρ', the optimization process used as a reference to define the regression model. Every point represents a different CPU, each with a different machine score and runtime of ρ' in this machine.

tions that can be reproduced in any machine. Aiming to obtain a more diverse sequence of computational instructions, we define the optimization process ρ' as the computational instructions generated by consecutively executing 4 different optimization algorithms in 16 problem instances. Each of the 64 executions involves solving a permutation problem with an optimization algorithm, with a stopping criterion of a maximum of $2 \cdot 10^6$ evaluations (see AppendixC for details on the optimization problems and algorithms used).

The runtime of the optimization process ρ' in a machine can be estimated with its machine score. In this paper, we measure the score of a machine (its computational capability) in terms of its PassMark single thread CPU score [3], although adapting the proposed methodology to other benchmarks is also possible. The advantage of using the PassMark score is that it can be looked up, as the score of most consumer CPUs is listed on their website. From the experimental results depicted in Figure 2, we see that the runtime of ρ' decreases linearly with respect to the machine score. Every point in Figure 2 represents a different

[3] The PassMark CPU score is one of the most popular CPU benchmark scores, with over 1000 CPUs listed on their website. In this paper, we use the version 10 single thread score of this benchmark. The highest and lowest scores at the time of writing this paper are 3174 and 147, respectively, with a higher value of the score being associated with a better relative performance of the CPU.

machine, each with its own machine score and runtime of ρ'.

Based on the figure shown, we infer that i) linear regression is suitable to model the runtime of ρ' with respect to the machine score, and ii) the machine score has a good correlation with the runtime of ρ'. With this intuition in mind, we define the estimation of the runtime in a machine:

Definition 4. *(Prediction of the runtime of ρ' in a machine)*
Let M_j be a machine and s_j its machine score. Then, the runtime of ρ' in M_j is modeled as

$$t(M_j, \rho') \approx -0.57406 s_j + 1929$$

where s_j is the score of machine M_j.

Considering together Assumption 1 and Definition 4, the equivalent runtime can be estimated as:

$$t_2 \approx \hat{t}_2 = t_1 \cdot \frac{-0.57406 s_2 + 1929}{-0.57406 s_1 + 1929} = t_1 \cdot \frac{3360.16 - s_2}{3360.16 - s_1} \tag{2}$$

where s_1 and s_2 are the PassMark single thread scores of the CPUs on machines M_1, M_2, respectively. Due to the approximation errors in Assumption 1 and Definition 4, the estimated equivalent runtime $\hat{t}_2 = \hat{t}(M_2, \rho)$ will often differ from the real equivalent runtime value $t_2 = t(M_2, \rho)$. This means that when using the estimated equivalent runtime as the stopping criterion for algorithm B, sometimes, the runtime will be longer or shorter than the runtime used by algorithm A.

To statistically assess the uncertainty associated with the comparison of the performance of algorithms A and B, in this methodology, we propose using a one-sided statistical test. Under this test, the alternative hypothesis states that the performance of algorithm B is better than the performance of algorithm A. As a result, a type I error (erroneously finding a statistically significant difference in the performance of A and B) can only be made when algorithm B performs better than A.

When a shorter runtime is estimated, algorithm B has an "unfairly" shorter stopping criterion for the optimization. This implies that the measured performance of B will be worse than or equal to the performance that would have been measured, if the actual equivalent runtime were used. Consequently, taking into account the one-sided nature of the test, estimating a lower than actual runtime will not increase the probability of type I error (estimating a lower than actual runtime can never help algorithm B perform better than algorithm A). It might, however, increase the probability of type II error.

As mentioned in the introduction, in the context of algorithm performance comparison, making a type II error is better than making a type I error. When using null hypothesis statistical tests, a type I error involves falsely rejecting H_0. A type II error, on the other hand, means failing to reject H_0, when H_0 is true.

Type I and II errors

		reality	
		H_0 is true	H_1 is true
conclusion	reject H_0, accept H_1	type I error, prob. α	✓
	not reject H_0, not accept H_1	✓	type II error, prob. β

Table 2: A table summarizing the type I and II errors. In our context of one-sided algorithm performance comparison, the hypothesis H_0 states that the performance of algorithms A and B is the same, or that the performance of algorithm A is better than the performance of algorithm B. The alternative hypothesis H_1 states that the performance of algorithm B is better than the performance of algorithm A.

It is noteworthy that failing to reject H_0 does not imply accepting H_0 (see Table 2). When a type II error is made, the conclusion is that there is not enough evidence to state that the performance of B is better than the performance of A, which is not an erroneous conclusion in itself. On the other hand, when a type I error is made, the conclusion is that the performance of B is better than the performance of A, when in reality it is not. Unlike the previous case, this is an erroneous conclusion.

To avoid drawing erroneous conclusions, we present a modification to Equation (2) so that the probability of estimating a longer time than the actual equivalent runtime stays under 1%. We reformulate the unbiased estimator shown in Equation (2) to reduce the probability of estimating a longer than real equivalent runtime. The new biased estimator is defined by multiplying the unbiased estimator with a correction parameter $\gamma \in (0, 1]$:

Definition 5. *(estimation of the equivalent runtime in machine M_2)*
Let ρ be an optimization process, M_1, M_2 two machines and t_1 the runtime of ρ in machine M_1. We estimate the equivalent runtime of ρ for machine M_2 as:

$$t(M_2, \rho) \approx \hat{t}(M_2, \rho) = t(M_1, \rho) \cdot \frac{3360.16 - s_2}{3360.16 - s_1} \cdot \gamma$$

By adjusting γ, the probability of estimating a longer runtime than the equivalent runtime, $\mathcal{P}[\hat{t}_2 > t_2]$, can be reduced; at the cost of estimating, on average, a shorter runtime. With $\gamma = 1.0$, the original, unbiased estimator is obtained. A lower value of γ has associated a lower probability of estimating a longer than real runtime. Specifically, the parameter γ is equal to $\mathbb{E}[\frac{\hat{t}_2}{t_2}]$: how much shorter the estimated equivalent runtime is than the real equivalent runtime on average. Figure 3 shows the probability of estimating a longer than the real equivalent runtime $\mathcal{P}[\hat{t}_2 > t_2]$ with respect to γ. As can be seen in the figure, with a correction coefficient lower than 0.5, the probability of estimating a longer than the real equivalent runtime is close to zero. In this paper, we propose using

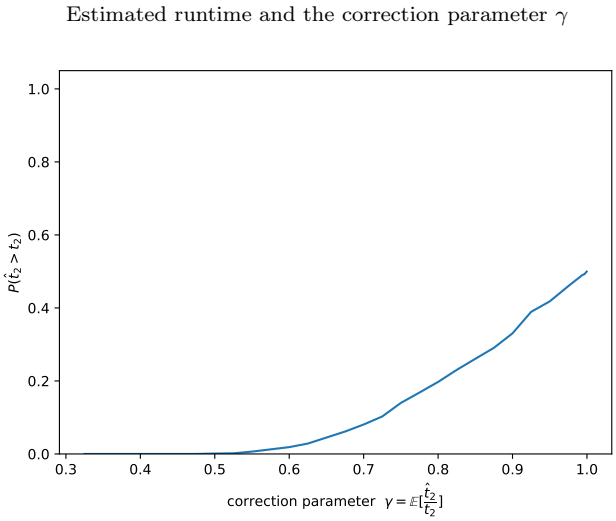

Figure 3: The probability of estimating a longer than real runtime with respect to γ.

$\gamma = 0.55$, which has an associated $\mathcal{P}[\hat{t}_2 > t_2]$ lower than 1%[4].

Even though the proposed model has a low probability of estimating a longer than actual equivalent runtime, this probability is not zero. In the following section, we propose a modification of the sign test that takes into account this probability and avoids an increase in the probability type I error.

3. Modifying the one-sided sign test

In the previous sections, we proposed an estimator of the equivalent runtime of an algorithm in a machine. Specifically, we proposed a biased estimator with a probability of less than 1% of estimating a longer than real equivalent runtime. By using this estimator, over 99% of the samples of the performance of algorithm B will not be computed with a longer than real equivalent runtime. However,

[4] The choice of 1% was arbitrary, but if other values are chosen, then the corrections that we later propose for the sign test also need to be modified. This percentage was computed with the optimization processes defined in AppendixC. The probability of estimating a higher than real equivalent time was computed by measuring the probability of estimating a runtime of an optimization process in a machine that was higher than the time it took to complete this optimization process in that machine. A cross-validation setting was used, to make sure that the optimization processes used to fit the parameters of Definition (4) never involved the same optimization problems or CPUs used to compute the probability of predicting a higher than real runtime. This cross-validation framework ensures that the computed percentage is as realistic as possible regarding the use of the proposed model to estimate runtimes in new CPUs and optimization problems.

there is still a chance that a longer than real equivalent time is estimated. And when it is, the probability of making a type I error is higher than if the comparison were carried out in the same machine. Therefore, in this section, we propose a correction of the one-sided statistical test that takes into account the probability of estimating a longer than the real equivalent time and its subsequent increase in the probability type I error.

Given algorithms A, B, a one-sided hypothesis test in algorithm performance comparison is as follows[5]:

H_0: The performance of algorithm B is worse than or equal to A.
H_1: The performance of algorithm B is better than A.

Going back to the example in the introduction, let us assume a researcher reads a paper in which algorithm A is executed in machine M_1, taking time t_1. Now, the researcher wants to compare algorithm B with A, but has no access to algorithm A nor to machine M_1. Instead, he/she only has access to machine M_2 and algorithm B. Executing algorithm B in machine M_2 for the equivalent runtime t_2 would make a fair comparison (due to Definition 3). However, determining t_2 defeats the purpose of using this methodology, as it involves executing A in machine M_1. Therefore, the real equivalent runtime is estimated with the formula in Definition 5.

We believe that the sign test [6] is a suitable hypothesis in the context of this paper and, in general, for comparing the performance of optimization algorithms (see AppendixD for details). We limit the statistical test to the one-sided sign test, with the alternative hypothesis being that the algorithm whose equivalent runtime was estimated has a higher performance. In the following, we propose a correction for the sign test that does not increase the probability of type I error.

3.1. One-sided sign test

The sign test [6] is a special case of the binomial test, for $p = 0.5$. In the context of algorithm performance comparison, the sign test statistically assesses if the paired performance of two algorithms is the same or not. Performing this statistical test involves first executing the optimization algorithms A and B in the same machine, with the same stopping criterion, in a set of n problem instances ($i \in \{1, ..., n\}$), obtaining the scores a_i and b_i for each algorithm-instance pair.

[5]It is noteworthy that failing to reject H_0 does not imply statistical evidence that H_0 is true, instead it suggests a lack of evidence against H_0. Therefore, in this case, it would not be correct to conclude that "the algorithms compared perform the same with a statistical significance of $1 - \alpha$".

These scores depend on which random seed was chosen (this seed represents all the non-deterministic parts of the algorithms, such as solution initialization). Thus, the performance of an algorithm in an instance can also be seen as a random variable that is completely determined, given a certain seed. We denote the random variables that represent the performance of algorithms A and B in an instance i as A_i and B_i, respectively.

The statistical test allows us to draw conclusions about the algorithms on a larger set of problem instances based on the observed results in the set of n instances. The sing test is based on these three assumptions [6]:

- Each of the sample pairs A_i, B_i are mutually independent of the rest of the pairs.
- Any observable pair a_i, b_i can be compared, that is, we can say that $a_i < b_i$, $b_i < a_i$ or $a_i = b_i$.
- The pairs are internally consistent, or if $\mathcal{P}[A_i > B_i] > \mathcal{P}[A_i < B_i]$ for one pair, then the same is true for all pairs.

In the context of algorithm comparison, the most problematic assumption is the first one. The reason is that in real-life benchmarks, some problem instances may share similarities, which means that there is no complete independence among all sample pairs A_i, B_i. The Mann-Whitney and the Wilcoxon signed rank test also contain this first assumption [6]. However, in practice, this limitation is usually ignored. This is why, in general, it is a good idea to use a set of benchmark problems with many kinds of different instances.

From now on, without loss of generality[6], we assume that the algorithms deal with a minimization problem, (i.e., a_i is better than $b_i \iff a_i < b_i$). We define $\#\{A_i < B_i\}$ as a random variable that counts the number of cases[7] that $A_i < B_i$ in n instances. Then, the following hypothesis test corresponds to the one-sided sign test [6]:

$$\boxed{\begin{array}{l} H_0 : \mathcal{P}[A_i < B_i] \geq \mathcal{P}[A_i > B_i] \\ H_1 : \mathcal{P}[A_i < B_i] < \mathcal{P}[A_i > B_i] \end{array}}$$

Under H_0, the null distribution of $\#\{A_i < B_i\}$ is $Bin(n, 0.5)$, where $Bin(n, 0.5)$ is the binomial distribution of size n and rate of success 0.5 [6]. The p-value for this hypothesis test is

$$p(k) = \mathcal{P}[\#\{A_i < B_i\} \leq k \ \mid \ H_0] = \mathcal{P}[Bin(n, 0.5) \leq k] \tag{3}$$

[6] A maximization problem can be converted into a minimization problem by multiplying the objective function with -1.

[7] Without loss of generality, we can assume that $a_i \neq b_i$, because samples in which $a_i = b_i$ are removed when performing the sign test [6].

where k is substituted by the statistic of the sign test: the number of cases that $a_i < b_i$ in all $i \in \{1, ..., n\}$ samples, denoted as $\#\{a_i < b_i\}$. By definition [12], the p-value can be interpreted as the probability of obtaining a more extreme (lower) statistic than the observed, assuming H_0 is true. If we reject the null hypothesis when $p(\#\{a_i < b_i\}) \leq \alpha$, then the probability of type I error is less than or equal to α.

The critical value can also be used instead of the p-value to decide when to accept or reject the null hypothesis. The value of the statistic associated with a certain p-value is called critical value. For a given target α value, the critical value $\text{Crit}_{\#\{a_i<b_i\}}(\alpha)$ is the largest value of the statistic that produces a p-value lower than or equal to α [6]. The advantage of the critical value over the p-value is that it is easier to compute. It follows that in the sign test [12], rejecting the null hypothesis when the value of the statistic $\#\{a_i < b_i\}$ is equal or lower than $\text{Crit}_{\#\{a_i<b_i\}}(\alpha)$ has an associated probability of type I error lower than α.

3.2. The corrected critical value

In practice, the statistic $\#\{a_i < b_i\}$ cannot be computed because the real equivalent runtime t_2 is unobservable. The equivalent runtime is approximated with $\hat{t}_2$ (see Definition 5). As a result, each b_i is substituted with its corresponding $\hat{b}_i$, which is computed by using $\hat{t}_2$ instead of t_2 as the stopping criterion. This means that the statistic $\#\{a_i < b_i\}$ is replaced by $\#\{a_i < \hat{b}_i\}$, which counts the number of times that $a_i < \hat{b}_i$ (without loss of generality, minimization is assumed) is observed. Therefore, we need to define the function to obtain the p-value associated with the statistic $\#\{a_i < \hat{b}_i\}$:

$$\hat{p}(k) = \mathcal{P}[\#\{A_i < \hat{B}_i\} \leq k \ \mid \ H_0] \tag{4}$$

where the p-value is obtained by substituting k with the observed statistic $\#\{a_i < \hat{b}_i\}$. The p-value is the probability of obtaining a statistic that is lower than or equal to the observed, when H_0 is true. Notice that if we reject the null hypothesis when $\hat{p}(\#\{a_i < \hat{b}_i\}) \leq \alpha$, then the probability of type I error is less than or equal to α.

As seen in Section 3, for each instance i, the probability of $\hat{t}_2 > t_2$ is lower than 0.01. Consequently, in more than 99% of cases, b_i is obtained with a longer runtime than $\hat{b}_i$ and therefore, the probability of $\hat{b}_i \geq b_i$ is greater than 0.99. This means that $\#\{a_i < \hat{b}_i\}$ is expected to be higher than or equal to $\#\{a_i < b_i\}$, but it will not always be so. To overcome this limitation, we need to define a corrected p-value $\hat{p}_c$, an upper bound of the actual p-value associated with statistic $\#\{a_i < \hat{b}_i\}$, that takes into account the small but nonzero probability (less than 1%) of $\hat{t}_2 > t_2$. Specifically, we define this upper

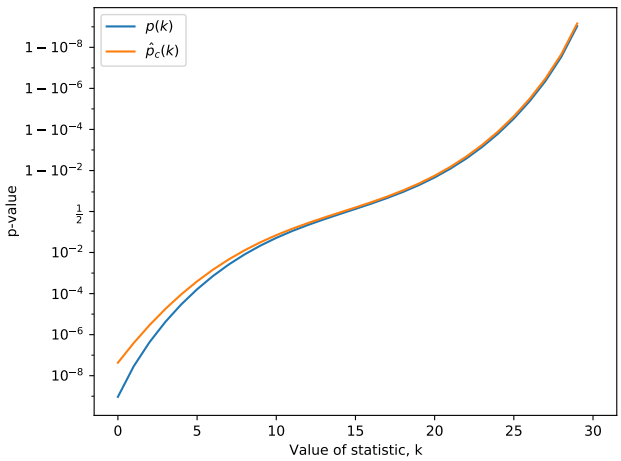

Figure 4: This figure shows the p-value p and the corrected p-value $\hat{p}_c$ for the sign test with a sample size of $n = 30$. Under the null hypothesis H_0, the p-value represents the probability of $\#\{a_i < b_i\} \leq k$, while the corrected p-value represents an upper bound of the probability of $\#\{a_i < \hat{b}_i\} \leq k$. The null hypothesis H_0 is that $\#\{a_i < b_i\}$ follows a binomial distribution of size n and probability of success 0.5.

bound as

$$\hat{p}_c(k) = \sum_{v=0}^{n} (1 - \mathcal{P}[Bin(n, 0.01) < \max(0, v - k)])\mathcal{P}[Bin(n, 0.5) = v] \tag{5}$$

such that

$$\hat{p}_c(k) > \mathcal{P}[\#\{A_i < \hat{B}_i\} \leq k \;\; | \;\; H_0] = \hat{p}(k) \tag{6}$$

is satisfied (we prove this inequality in AppendixE), where H_0 implies that statistic $\#\{A_i < B_i\}$ follows the null distribution $Bin(n, 0.5)$ [6]. Figure 4 shows p and $\hat{p}_c$ side by side. Notice that $\hat{p}_c$ is slightly higher, because it needs to account for the probability that $\hat{t}_2 > t_2$. The corrected p-value $\hat{p}_c$ is interesting because rejecting H_0 when $\hat{p}_c(\#\{a_i < \hat{b}_i\}) < \alpha$ has also an associated probability of type I error lower than α. The reason is that $\hat{p}_c(\#\{a_i < \hat{b}_i\}) > \hat{p}(\#\{a_i < \hat{b}_i\})$, and therefore, $\hat{p}_c(\#\{a_i < \hat{b}_i\}) < \alpha \Rightarrow \hat{p}(\#\{a_i < \hat{b}_i\}) < \alpha$.

Finally, given a target probability of type I error α, we define the corrected critical value $\text{Crit}_c(\alpha)$ as the largest $\#\{a_i < \hat{b}_i\}$ value that produces a corrected p-value lower than α. Observe that rejecting H_0 when the observed $\#\{a_i < \hat{b}_i\}$ is lower than or equal to $\text{Crit}_c(\alpha)$ has an associated probability of type I error

lower than α. Again, the critical value is interesting because it is easier to compute than the p-value. We computed these corrected critical values for the target α values of $0.05, 0.01$ and 10^{-3}, shown in Table F.7 in AppendixF.

3.3. Assumptions and limitations

The proposed methodology is based on certain assumptions and should only be used taking into account certain limitations that will be addressed in this section. First, we will address the assumption related to the probability of predicting a longer than real equivalent runtime. Let $(\hat{t}_2 > t_2)_i$ be a random variable that represents if the estimated equivalent runtime for algorithm A, instance i in machine M_2, is longer than the real equivalent runtime or not. The required assumption is similar to assuming that $\hat{t}_2 > t_2$ is mutually independent for each instance i. Specifically, it is required[8] that $\mathcal{P}[(\hat{t}_2 > t_2)_i | \cap_{i \neq j} (\hat{t}_2 > t_2)_j] < 0.01$.

One could argue that this assumption is false because $(\hat{t}_2 > t_2)$ depends on many factors, such as the machines used in the experimentation. The same two machines are used to compute all samples a_i, b_i suggesting that all $(\hat{t}_2 > t_2)_i$ can never be truly independent among each other. However, even though we can not ensure that $\mathcal{P}[(\hat{t}_2 > t_2)_i | \cap_{i \neq j} (\hat{t}_2 > t_2)_j] < 0.01$, by choosing a suitable correction coefficient γ, in Section 2, we estimated that $\mathcal{P}[(\hat{t}_2 > t_2)_i] < 0.01$.

In addition to the previous assumption, the proposed methodology only considers one side hypothesis testing. In this regard, it should only be applied to show a statistically significantly superior performance of the algorithm whose equivalent runtime was estimated (denoted as algorithm B in this paper). The reason is that algorithm B has a high probability of having a lower runtime, thus, it is easy that B performs worse than A, while the opposite is difficult. Failing to reject H_0 only indicates a lack of evidence against H_0, and in our context, it only indicates that there is not enough evidence to say that B performs better than A (it tells us nothing about A performing better than B).

Finally, there is a limitation regarding the chosen machine score: the PassMark single thread score. In Section 2, we saw that a linear function is a suitable function to model the relationship between the machine score and the runtime of the reference optimization process ρ'. The formula of the fitted linear regression is $t(M_j, \rho') \approx -0.57406 s_j + 1929$ where $t(M_j, \rho')$ is the equivalent runtime of ρ' in machine M_j, and s_j is the score of machine M_j. With this formula, a PassMark single thread score higher than 3360 produces a negative estimated equivalent runtime, which does not make sense. However, for the 8 machines used to fit the data, as Figure 2 shows, the linear model seems to be suitable. Therefore, we recommend applying the proposed methodology only in machines with PassMark single thread scores in the interval $(411, 2176)$. These values correspond to the highest and lowest values used in the fitting of the linear

[8]This assumption is required in the proof of Equation (6) in AppendixE. Specifically, it is used in Lemma 3.

regression. As future work, and especially when more powerful processors are available, the methodology can be updated to incorporate these new processors or even change the machine score to other benchmark scores beyond the PassMark single thread score.

4. Applying the methodology

In this section, we illustrate how to apply the proposed methodology with two examples. The proposed methodology is very simple to use and does not require any additional software. Further details and material are available in our GitHub repository.

4.1. Example I

In this example, we will compare a simple random initialization local search procedure with a memetic search algorithm by Benlic et al. [2] for the QAP. Using the proposed methodology, we will statistically assess the difference in the performance between these two algorithms, without having to execute the code of the memetic algorithm. In this case, the memetic search algorithm is algorithm A, because this is the algorithm of which we already have the results, and the local search algorithm is algorithm B, because this is the algorithm whose runtime is going to be estimated.

Step 1: Obtaining the data

To apply the proposed methodology, we need to find certain information about the execution of the memetic algorithm. The required data includes the list of instances to be used in the experimental comparison, the average objective value obtained by the memetic search algorithm, and the runtime of the memetic search algorithm in each of the instances. The information extracted from the article by Benlic et al. [2] is listed in the first three columns of Table 3. Also, we need to find the CPU model of the machine in which the memetic search was run (machine M_1), which is "Intel Xeon E5440 2.83GHz" as specified in their article. Finally, the machine score of this CPU, measured as PassMark version 10 single thread score, is $s_1 = 1230$ (as seen on the PassMark website).

Step 2: Estimating the equivalent runtime

With the data already gathered, the next step is to estimate the equivalent runtime of each instance for the machine in which the local search algorithm will be executed (machine M_2). Estimating the runtime requires the score s_2 of this machine to be known. The CPU model of M_2 is "Intel Celeron N4100", with a PassMark single thread score of $s_2 = 1032$. With this information, we are ready to estimate the equivalent runtime in machine M_2 (Definition 5):

$$\hat{t}_2 = t_1 \cdot \frac{3360.16 - s_2}{3360.16 - s_1} \cdot \gamma = t_1 \cdot \frac{3360.16 - 1032}{3360.16 - 1230} \cdot 0.55 = 0.601 \cdot t_1$$

instance	**Data obtained from the paper by Benlic et al. [2]**		**Data corresponding to the execution of B in machine M_2**	
	runtime in seconds, t_1	objective value, a_i	estimated equivalent runtime, $\hat{t}_2 = 0.601 \cdot t_1$	objective value, $\hat{b}_i$
tai40a	486	**3141222**	292.09	3207604
tai50a	2520	**4945266**	1514.52	5042830
tai60a	4050	**7216339**	2434.05	7393900
tai80a	3948	**13556691**	2372.75	13840668
tai100a	2646	**21137728**	1590.25	21611122
tai50b	72	**458821517**	43.27	459986202
tai60b	312	**608215054**	187.51	609946393
tai80b	1878	**818415043**	1128.68	824799510
tai100b	816	**1185996137**	490.42	1195646366
tai150b	4686	**499195981**	2816.29	505187740
sko100a	1338	**115534**	804.14	153082
sko100b	390	**152002**	234.39	155218
sko100c	720	**147862**	432.72	149076
sko100d	1254	**149584**	753.65	150568
sko100e	714	**149150**	429.11	150638
sko100f	1380	**149036**	829.38	150006

Table 3: This table shows all the data in the first example. The first three columns correspond to the QAP instances in which the memetic search algorithm by Benlic et al. [2] was tested, the runtime of the memetic search algorithm in each instance, and the best objective value they obtained in each execution, averaged in 10 executions per instance. The information in these three columns was directly obtained from the paper by Benlic et al., without any additional executions. The next two columns show the estimated equivalent runtimes and the average objective value scores that the local search algorithm obtained with this runtime as the stopping criterion. The local search algorithm was executed in machine M_2.

where t_1 is substituted with the runtime of the memetic search algorithm in each instance, listed in Table 3.

Step 3: Running the experiments

Now, we execute the local search algorithm in the instances listed in Table 3, using the estimated runtimes $\hat{t}_2$ as the stopping criterion. This execution is carried out in machine M_2, and the best objective function values $\hat{b}_i$ are listed in Table 3. Following the procedure by Benlic et al., these best objective values are averaged over 10 executions.

Step 4: Obtaining the statistic and comparing it with the corrected critical value

Once all the results have been computed, the next step is to compute the statistic $\#\{a_i < \hat{b}_i\}$, which counts the number of times that $a_i < \hat{b_i}$. In this case, $a_i < \hat{b_i}$ happens 15 times, and therefore, $\#\{a_i < \hat{b}_i\} = 15$. Now we compare the value of the statistic with the corrected critical value for a sample size (the number of instances) $n = 15$ and a target probability of type I error of $\alpha = 0.05$. The corrected critical value for this sample size and α is $\text{Crit}_c(\alpha) = 3$, as shown in shown in AppendixF. The observed statistic $\#\{a_i < \hat{b}_i\} = 15$ is not lower than or equal to the corrected critical value $\text{Crit}_c(\alpha) = 3$.

Step 5: Conclusion

Since the observed statistic is not lower than or equal to the critical value, we cannot reject H_0. In this case, the conclusion is that with the amount of data that we have and the chosen target probability of type I error of $\alpha = 0.05$, we can not say that the local search algorithm has a statistically significantly better performance than the memetic search algorithm[9].

It is important to note that, if we had considered the original runtimes t_1 as the stopping criterion for algorithm B in machine M_2 (longer than the estimated equivalent runtime $\hat{t}_2$), the local search would have had an unfairly longer runtime. In other words, the comparison would have been biased towards the local search.

4.2. Example II

In this second example, we will compare the same simple random initialization local search procedure with an estimation of distribution algorithm (EDA) for the QAP [1]. In this case, the EDA is algorithm A, because this is the algorithm of which we already have the results, and the local search algorithm is algorithm B, because this is the algorithm whose runtime is estimated.

[9]It would not be correct to conclude that the two algorithms perform (statistically significantly) the same, or that the memetic search performs statistically significantly better than the local search.

instance	Data obtained from the paper by Arza et al. [1]		Data corresponding to the execution of B in machine M_2	
	runtime in seconds, t_1	objective value, a_i	estimated equivalent runtime, $\hat{t}_2 = 1.085 \cdot t_1$	objective value, $\hat{b}_i$
bur26a	1.45	5432374	1.57	**5426670**
bur26b	1.45	3824798	1.57	**3817852**
bur26c	1.43	5427185	1.55	**5426795**
bur26d	1.44	3821474	1.56	**3821239**
nug17	0.44	1735	0.48	**1734**
nug18	0.51	1936	0.55	1936
nug20	0.68	2573	0.74	**2570**
nug21	0.77	2444	0.84	2444
tai10a	0.12	135028	0.13	135028
tai10b	0.12	1183760	0.13	1183760
tai12a	0.18	224730	0.20	**224416**
tai12b	0.19	39464925	0.21	39464925
tai15a	0.31	388910	0.34	**388214**
tai15b	0.31	51768918	0.34	**51765268**
tai20a	0.69	709409	0.75	**703482**
tai20b	0.68	122538448	0.74	**122455319**
tai40a	5.41	**3194672**	5.87	3227894
tai40b	5.41	644054927	5.87	**637470334**
tai60a	19.23	**7367162**	20.86	7461354
tai60b	19.21	**611215466**	20.84	611833935
tai80a	50.09	**13792379**	54.35	13942804
tai80b	50.1	836702973	54.36	**830729983**

Table 4: This table shows all the data in the second example. The first three columns correspond to the QAP instances in which the EDA algorithm by Arza et al. [1] was tested, the runtime of the memetic search algorithm in each instance, and the best objective value they obtained in each execution, averaged in 10 executions per instance. The information in these three columns was directly obtained from this paper, without any additional executions. The next two columns show the estimated equivalent runtimes and the average objective value scores that the local search algorithm obtained with this runtime as the stopping criterion. The local search algorithm was executed in machine M_2.

Step 1: Obtaining the data

To apply the proposed methodology, we need to find certain information about the execution of the memetic algorithm. The required data includes the list of instances to be used in the comparison, the average objective value obtained by the EDA, and the runtime used in each instance. The information extracted from the paper [1] is listed in Table 4. In addition, we need to find the CPU model of the machine in which the memetic search was run (machine M_1), which is "AMD Ryzen 7 1800X", as specified in the paper. Finally, the machine score of this CPU, measured as PassMark single thread score is $s_1 = 2182$, as seen on the PassMark website.

Step 2: Estimating the equivalent runtime

With the data already gathered, the next step is to estimate the equivalent runtime of each instance for the machine in which the local search algorithm will be executed (machine M_2). This estimation requires the machine score s_2 of this machine, which is the same as in the previous example, because the same PC was used, $s_2 = 1032$. The formula to estimate the runtime for each instance is:

$$\hat{t}_2 = t_1 \cdot \frac{3360.16 - s_2}{3360.16 - s_1} \cdot \gamma = t_1 \cdot \frac{3360.16 - 1032}{3360.16 - 2182} \cdot 0.55 = 1.085 \cdot t_1$$

where t_1 is substituted with the runtime of the EDA algorithm in each instance, listed in Table 4.

Step 3: Running the experiments

Now, we execute the local search algorithm in the instances listed in Table 4, using the estimated runtimes $\hat{t}_2$ as the stopping criterion. This execution is carried out on machine M_2, and the best objective function values $\hat{b}_i$ are listed in Table 4. Following the procedure by Arza et al., these best objective values are averaged over 20 executions.

Step 4: Obtaining the statistic and comparing it with the corrected critical value

After the executions, the statistic $\#\{a_i < \hat{b}_i\}$ is computed, which counts the number of times that $a_i < \hat{b}_i$. In this case, $a_i < \hat{b}_i$ happens 4 times, and therefore, $\#\{a_i < \hat{b}_i\} = 4$. Now we compare the value of the statistic with the corrected critical value for a sample size (the number of instances in which $a_i \neq \hat{b}_i$) $n = 17$ and a target probability of type I error of $\alpha = 0.05$. The corrected critical value for this sample size and α is $\text{Crit}_c(\alpha) = 4$, as shown in shown in AppendixF. The observed statistic $\#\{a_i < \hat{b}_i\} = 4$ is lower than or equal to the corrected critical value $\text{Crit}_c(\alpha) = 4$.

Step 5: Conclusion

Since the observed statistic is lower than or equal to the critical value, we reject H_0. In this case, the conclusion is that with a probability of type I error of $\alpha = 0.05$, the performance of the local search procedure is statistically significantly better than the performance of the EDA.

In this case, machine M_1 is more powerful (in terms of computational capabilities) than machine M_2. If we had considered the original runtimes t_1 as the stopping criterion for algorithm B in machine M_2 (shorter than the estimated equivalent runtime $\hat{t}_2$), it would have been more difficult for the local search to perform better than the EDA. In that case, H_0 might not have been rejected.

5. Conclusion and future work

Usually, comparing optimization algorithms with a maximum runtime as a common stopping criterion requires the algorithms to be executed in the same machine. Unfortunately, the code of all the algorithms is not always available. An alternative is to adjust the runtime of the algorithms relative to the speed of the machine in which they are executed. In this paper, we proposed a methodology to statistically compare the performance of two optimization algorithms in two different machines, when the results of one of the algorithms are already known and without having to execute this algorithm again. The methodology ensures that the probability of type I error does not increase due to the algorithms being executed in different machines. To achieve this, first, the runtime of the executed algorithm is adjusted based on the speed of the CPUs of both machines. Then, a modified one-sided sign test is used so that the probability of using an unfairly longer runtime is taken into account. We illustrate the application of the proposed methodology with two examples.

Alongside this paper, a tutorial with examples is presented in our GitHub repository. This will hopefully make it simple for people to apply the proposed methodology.

As future work, we expect to address the assumptions mentioned in Section 3.3. Besides, using more machines to fit the data may reduce the error of the estimation. This reduction in error might allow a higher correction coefficient to be chosen, which in turn might make the corrected sign test more powerful (a reduction of type II error). Finally, it might be worth investigating the possibility of adopting the presented methodology to the Wilcoxon sign-rank and Mann-Whitney tests, or even other non-hypothesis testing statistical methods, such as Bayesian analysis.

Acknowledgments

This work was funded in part by the Spanish Ministry of Science, Innovation and Universities through PID2019-1064536A-I00 and the BCAM Severo Ochoa excellence accreditation SEV-2017-0718; by Basque Government through consolidated groups 2019-2021 IT1244-19, ELKARTEK program and BERC 2018-2021 program; and by the Spanish Ministry of Economy and Competitiveness trough the project TIN2017-82626-R.

6. Bibliography

[1] Arza, E., Pérez, A., Irurozki, E., and Ceberio, J. Kernels of Mallows Models under the Hamming Distance for solving the Quadratic Assignment Problem. *Swarm and Evolutionary Computation* (July 2020), 100740.

[2] Benlic, U., and Hao, J.-K. Memetic search for the quadratic assignment problem. *Expert Systems with Applications 42*, 1 (Jan. 2015), 584–595.

[3] Beyer, W. H. *Standard Probability and Statistics: Tables and Formulae*. CRC Press, 1991.

[4] Ceberio, J., Irurozki, E., Mendiburu, A., and Lozano, J. A. A review of distances for the Mallows and Generalized Mallows estimation of distribution algorithms. *Computational Optimization and Applications 62*, 2 (Nov. 2015), 545–564.

[5] Ceberio, J., Mendiburu, A., and Lozano, J. A. The linear ordering problem revisited. *European Journal of Operational Research 241*, 3 (Mar. 2015), 686–696.

[6] Conover, W. J., and Conover, W. J. Practical nonparametric statistics.

[7] Domínguez, J., and Alba, E. A Methodology for Comparing the Execution Time of Metaheuristics Running on Different Hardware. In *Evolutionary Computation in Combinatorial Optimization*, vol. 7245. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 1–12.

[8] Goldberg, D. E., and Lingle, Jr., R. Alleles, Loci and the Traveling Salesman Problem. In *Proceedings of the 1st International Conference on Genetic Algorithms* (1985), L. Erlbaum Associates Inc., pp. 154–159.

[9] Gupta, J. N., and Stafford, E. F. Flowshop scheduling research after five decades. *European Journal of Operational Research 169*, 3 (Mar. 2006), 699–711.

[10] Koopmans, T. C., and Beckmann, M. Assignment Problems and the Location of Economic Activities. *Econometrica 25*, 1 (Jan. 1957), 53.

[11] Schiavinotto, T., and Stützle, T. A review of metrics on permutations for search landscape analysis. *Computers & operations research 34*, 10 (2007), 3143–3153.

[12] Wasserstein, R. L., and Lazar, N. A. The ASA Statement on p-Values: Context, Process, and Purpose. *The American Statistician 70*, 2 (Apr. 2016), 129–133.

[13] Weicker, R. P. Dhrystone benchmark: Rationale for version 2 and measurement rules. *ACM SIGPLAN Notices 23*, 8 (Aug. 1988), 49–62.

[14] Zou, K. H., Tuncali, K., and Silverman, S. G. Correlation and simple linear regression. *Radiology 227*, 3 (2003), 617–628.

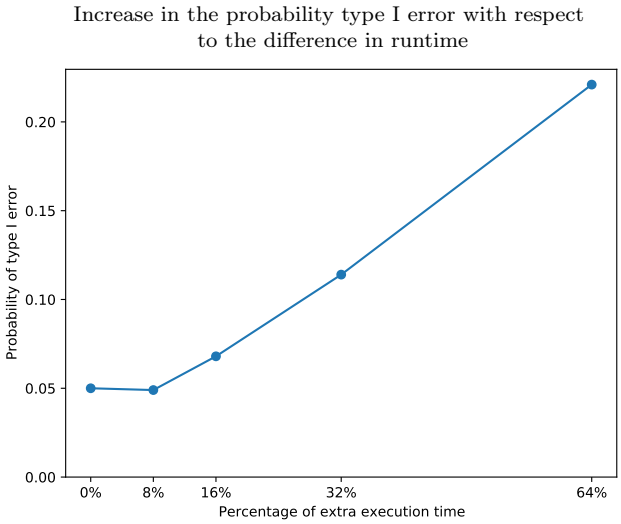

Figure A.5: Probability of type I error in the one-sided sign test when comparing two identical random search algorithms. One of the algorithms is given extra runtime, according to the X-axis. The test is applied to a set of 16 problem instances.

AppendixA. The importance of using the same resources in algorithm comparison

As claimed in the introduction, it is essential to run the algorithms with the same computational resources to carry out a fair comparison. To better illustrate this point, in the following lines, a small experiment is presented. This experiment illustrates the increase in the probability of type-I error (the probability of erroneously concluding a difference in performance, when in reality, there is none) with respect to the difference in execution time. Specifically, we run a random search algorithm twice in each problem instance[10] and perform the one-sided sign test [6][11] (see Section 3.1 for an explanation of the sign test), on the set of results obtained. The significance level is set to $\alpha = 0.05$. Even though the random search algorithm is being compared with itself, we increase the runtime of one of the executions by 8, 16, 32, or 64 percent. We repeat the steps above 1000 times to estimate the probability of type I error (estimated as the probability of rejecting H_0).

Figure A.5 shows the estimated probability of type I error. Notice that the

[10] A set of 16 permutation problem instances is considered, 4 instances of 4 problems. The four permutation problems considered are the traveling salesman problem, the permutation flowshop scheduling problem, the linear ordering problem, and the quadratic assignment problem.

[11] In AppendixD, we explain why we limit the statistical analysis to the sign test in this paper.

type I error starts at 0.05, which is the expected result for a significance level of $\alpha = 0.05$. However, the error shoots up dramatically when the difference in runtime increases, more than doubling when the percentage of extra runtime reaches 32%. Therefore, a discrepancy in the runtime of the algorithms being compared, if high enough, can lead to falsely concluding that the performance of the algorithms is not the same. A fair comparison requires the same computational resources to be assigned in the execution of each algorithm.

AppendixB. Proportional runtime of optimization processes in different machines

The runtime of an optimization process (a sequence of computational instructions) is different in each machine. However, even though it is different, there might be a proportional relationship between the runtime of the same optimization process in two different machines. To study this, we compute the correlation that several optimization processes have on two machines. Specifically, we computed the correlation of 64 different optimization processes (see AppendixC for additional details about the experiment) for every possible pair of machines from the 8 different machines used in the experimentation. The average Pearson's correlation coefficient of the runtimes is 0.989987, which shows a strong linear [14] relationship between the runtime of the same optimization process in two different machines.

Given two machines M_1 and M_2, the runtime of an optimization process can be considered as a two-dimensional vector, where each of the dimensions represents the runtime of the optimization process in each of the machines. Thus, knowing the runtime $t(s, M_1)$ of an optimization process ρ in a machine M_1, it is reasonable to estimate the equivalent runtime of ρ in another machine M_2, when the runtime of two other optimization processes ρ' and ρ'' is known for both machines. In fact, with such a high Pearson's correlation coefficient, the runtimes of these optimization processes (red crosses in Figure B.6) will almost be aligned in a line [14]. Therefore, the estimated runtime of ρ in machine M_2 is defined as the value that makes the runtime of the three optimization processes aligned. This is shown by the orange line in Figure B.6.

Observe that this procedure requires the runtime of two optimization processes ρ' and ρ'' to be known in both machines M_1, M_2. However, by considering an additional hypothesis, we can reduce the requirement to only one optimization process ρ'. This additional hypothesis is that the regression line has to cross the origin. Intuitively, if an optimization process (sequence of computational instructions) takes no time in a machine, it makes no sense that it takes a positive amount of time in another machine. In addition, without this condition, it could be possible to estimate a negative runtime, which is not properly defined.

In this setting, the estimated runtime for the optimization process ρ in machine M_2 is set so that the runtime of the optimization processes ρ and ρ' and the origin are in the same line. This is represented by the blue line in Figure B.6.

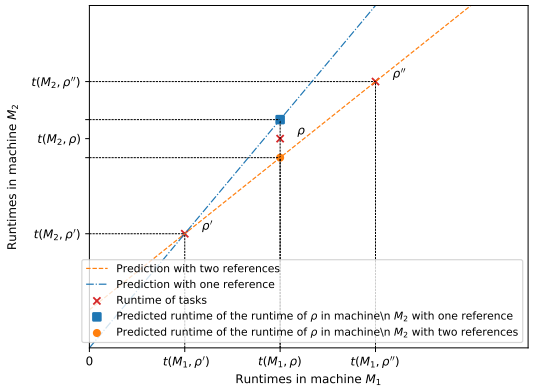

Figure B.6: Estimation of the runtime of an optimization process ρ with one ρ' or two ρ', ρ'' reference optimization processes. The x-axis represents the runtime in machine M_1, while the y-axis is the runtime in machine M_2. The runtime of the optimization process ρ is estimated for machine M_2.

The estimation of the runtime of the optimization process ρ in machine M_2, shown in the figure as a blue square, is given by the slope-intercept formula for the points $(0, 0)$ and $(t(M_1, \rho'), t(M_2, \rho'))$:

$$t(M_2, \rho) \approx \frac{t(M_2, \rho')}{t(M_1, \rho')} t(M_1, \rho) \tag{B.1}$$

By rewriting Equation (B.1), we obtain that the ratio of two optimization processes is (approximately) constant in different machines:

$$\frac{t(M_2, \rho)}{t(M_2, \rho')} \approx \frac{t(M_1, \rho)}{t(M_1, \rho')} \tag{B.2}$$

AppendixC. Details on the executed optimization processes

In the following, we detail the optimization processes executed in the experimentation. As defined in this paper, an optimization process is just a sequence of computational instructions that can be executed in any machine. Specifically, each of the optimization processes described in this section consists of executing an optimization algorithm in a problem instance for a maximum of $2 \cdot 10^6$ objective function evaluations.

Problem instances: We solved four types of optimization problems (all of them are permutation problems): the traveling salesman problem [8], the quadratic

Problem instances

instance name	problem	size
tai75e02	qap	75
sko100a	qap	100
tai100a	qap	100
tai100b	qap	100
eil101	tsp	101
pr136	tsp	136
kroA200	tsp	200
kroB200	tsp	200
tai100_20_0	pfsp	(100,20)
tai100_20_1	pfsp	(100,20)
tai200_20_1	pfsp	(200,20)
tai200_20_1	pfsp	(200,20)
N-be75np_150	lop	150
N-stabu3_150	lop	150
N-t65d11xx_150	lop	150
N-t70f11xx_150	lop	150

Table C.5: The list of 16 problem instances and their size.

Machines

CPU model name	speed score
Intel i5 470U	411
Intel Celeron N4100	1032
AMD A9 9420 with Radeon R5	1311
AMD FX 6300	1484
Intel i7 2760QM	1550
Intel i7 6700HQ (2.60GHz)	1918
Intel i7 7500U	2025
AMD Ryzen7 1800X	2180

Table C.6: The list of 8 machines used in the experimentation and their speed score, measured in terms of PassMark single thread score.

assignment problem [10], the linear ordering problem [5] and the permutation flowshop scheduling problem [9]. For each of these four problem types, we chose 4 problem instances, as listed in Table C.5.

Optimization algorithms: Each of the 16 problem instances was optimized with four optimization algorithms. These optimization algorithms are random search and local search with three different neighborhoods: swap, interchange and insert [11, 4]. The local search is a best-first or greedy approach that is randomly reinitialized when a local optimum is found.

We define each of the 64 different optimization processes as running each of these four optimization algorithms in each of the 16 problem instances.

Machines: The experimentation was carried out in a set of 8 different machines. Table C.6 lists the CPU models of these machines, as well as their single thread PassMark CPU scores.

AppendixD. The sign test for algorithm performance comparison

When statistically assessing the comparison of the performance of optimization algorithms, a classical way is to use non-parametric tests as the distribution of the performance is usually unknown. In the literature, the Wilcoxon signed-rank test, the Mann-Whitney test and the sign test [6] are often used to assess a statistically significant difference in the performance of two algorithms. We argue that, in the context of optimization algorithm performance comparison, it may be more suitable to use the sign test than the Wilcoxon signed-rank or the Mann-Whitney test.

It turns out that the result of the Wilcoxon and the Mann-Whitney tests might change when the objective function value of some of the problems is scaled (multiplied or divided by a positive constant). The reason is that they both take into account the magnitude of the differences between the observations, and these differences change with scaling. A usual solution is to consider the average relative deviation percentage with respect to the optimum (or any other reference solution) instead of the objective value, but this only changes the problem: now the results of these tests change when the objective function value of some of the problems is shifted (add or subtract a constant). In our opinion, the performance comparison of two optimization algorithms should be invariant to these two alterations, otherwise, problems that are on a higher scale (for example, when the dimension of the problem is high), will have a larger impact on the result of the statistical test. In addition, we believe that it is reasonable that all problem instances have the same weight in the conclusion of the statistical test, which both the Wilcoxon signed-rank and the Mann-Whitney test are unable to accomplish due to their dependence on the magnitude of the differences.

An alternative is the sign test [6], which is invariant to the shifting and scaling of the problems. In fact, the result of the sign test does not change even if some of the problems are modified by composing the objective function with any strictly increasing function. For this reason, and even though the sign test is a less powerful alternative (higher probability of type II error), we believe it is the most suitable hypothesis test for algorithm performance comparison when the objective functions of all the problems are not directly comparable.

AppendixE. Proof of Equation (6).

When performing the statistical analysis, a set of n problem instances is used to compute the statistic and the p-value. The goal of the analysis is to draw conclusions on a larger set of problem instances based on the observed sample of size n. Given a problem instance, we can define the performance of an algorithm in this instance.

Definition 6. *(The performance of an algorithm in an instance)*
Let M be a machine, t a stopping criterion in terms of maximum runtime, A an optimization algorithm and i a problem instance. The performance of algorithm A in an instance i, denoted $A(M,t,i)$, is defined as a random variable whose outcome is obtained by first sampling a random seed r and then optimizing instance i with optimization algorithm A in machine M for time t. Given this random seed r, the performance of an algorithm in an instance is deterministic.

In Section 2, we defined t_1 as the stopping criterion for algorithm A in machine M_1, which is obviously the time it takes to carry out this optimization process in machine M_1. We also defined the equivalent runtime t_2 as the time it takes to replicate the exact same optimization process in machine M_2 in definition 3. Because of this definition, $A(M_1,t_1,i)$ and $A(M_2,t_2,i)$ are the same random variables. Therefore, it makes sense to denote $A(M_1,t_1,i)$ and $A(M_2,t_2,i)$ or $B(M_1,t_1,i)$ and $B(M_2,t_2,i)$ as A_i or B_i, respectively. To ease the notation, we will also denote $B(M_2,\hat{t}_2,i)$ as $\hat{B}_i$.

Finally, as discussed in Section 3.3, we assume that whether $\hat{t}_2 < t_2$ is true or not is independent for each instance i, and that $\mathcal{P}(\hat{t}_2 < t_2) < 0.01$. Let us now prove Equation (6).

Lemma 1. *Let n be an integer, X and Y two random variables. Let $X_1,...,X_n$ be n independent random variables distributed as X. Let $Y_1,...,Y_n$ be n independent random variables distributed as Y. Let v_x and v_y be two possible outcomes of the random variables X and Y respectively, $l \in \{0,...,n\}$ be an integer and $p \in (0,1)$ be a real number.*

I) If $\mathcal{P}[Y = v_y \mid X = v_x] = 1$, then

$$\mathcal{P}[X = v_x] \leq \mathcal{P}[Y = v_y]$$

and

$$\#\{X_i = v_x\} \leq \#\{Y_i = v_y\}$$

II) If $\mathcal{P}[Y = v_y \mid X = v_x] = 1$ and $\mathcal{P}[X = v_x \mid Y = v_y] = 1$ then

$$\mathcal{P}[X = v_x] = \mathcal{P}[Y = v_y]$$

III) If $\mathcal{P}[X = v_x] < p$ *then*

$$\mathcal{P}[\#\{X_i = v_x\} \geq l] < \mathcal{P}[Bin(n, p) \geq l]$$

Lemma 2. *Let* $i \in \{1, .., n\}$ *be* n *problem instances and let* A *and* B *be two optimization algorithms. Let* a_i, b_i *and* $\hat{b}_i$ *be the observed values of* A_i, B_i *and* $\hat{B}_i$ *respectively,* $\forall i \in \{1, ..., n\}$. *Let* k *and* $v \in \{0, ..., n\}$ *be two integers. Suppose that* $A_i \neq B_i$ *and* $A_i \neq \hat{B}_i$.

Then,

$$\mathcal{P}[\#\{A_i < min(\hat{B}_i, B_i)\} \leq \min(k, v) \ \mid \ \#\{A_i < B_i\} = v] \leq$$
$$\mathcal{P}[\#\{A_i > \hat{B}_i \wedge A_i < B_i\} \geq \max(0, v - k) \ \mid \ \#\{A_i < B_i\} = v]$$

Proof.

$$\#\{A_i < min(\hat{B}_i, B_i)\} \leq \min(k, v) \implies$$
$$\#\{A_i < \hat{B}_i \wedge A_i < B_i\} \leq \min(k, v) \implies$$
$$\#\{A_i < B_i\} - \#\{A_i > \hat{B}_i \wedge A_i < B_i\} \leq \min(k, v) \implies$$

Substituting $\#\{A_i < B_i\} = v$,

$$v - \min(k, v) \leq \#\{A_i > \hat{B}_i \wedge A_i < B_i\} \implies$$

Considering $v - \min(k, v) = \max(0, v - k)$,

$$\#\{A_i > \hat{B}_i \wedge A_i < B_i\} \geq \max(0, v - k)$$

We have just shown that

$$\#\{A_i < min(\hat{B}_i, B_i)\} \leq \min(k, v) \implies \#\{A_i > \hat{B}_i \wedge A_i < B_i\} \geq \max(0, v - k)$$

Which means that,

$$\mathcal{P}[\#\{A_i > \hat{B}_i \wedge A_i < B_i\} \geq \max(0, v - k) \mid \#\{A_i < min(\hat{B}_i, B_i)\} \leq \min(k, v)] = 1$$

Finally, we apply Lemma 1 I), obtaining

$$\mathcal{P}[\#\{A_i < min(\hat{B}_i, B_i)\} \leq \min(k, v) \ \mid \ \#\{A_i < B_i\} = v] \leq$$
$$\mathcal{P}[\#\{A_i > \hat{B}_i \wedge A_i < B_i\} \geq \max(0, v - k) \ \mid \ \#\{A_i < B_i\} = v]$$

□

Lemma 3. *Let $i \in \{1, .., n\}$ be n problem instances and let A and B be two optimization algorithms. Let a_i, b_i and $\hat{b}_i$ be the observed values of A_i, B_i and $\hat{B}_i$ respectively, $\forall i \in \{1, ..., n\}$. Let k and $v \in \{0, ..., n\}$ be two integers. Suppose that $A_i \neq B_i$ and $A_i \neq \hat{B}_i$.*

Then,

$$\mathcal{P}[\#\{A_i < \hat{B}_i\} \leq k \ \mid \ \#\{A_i < B_i\} = v] < \mathcal{P}[Bin(n, 0.01) \geq \max(0, v - k)]$$

Proof.

$$\mathcal{P}[\#\{A_i < \hat{B}_i\} \leq k \ \mid \ \#\{A_i < B_i\} = v] \leq$$
$$\mathcal{P}[\#\{A_i < min(\hat{B}_i, B_i)\} \leq k \ \mid \ \#\{A_i < B_i\} = v]$$

Now, observe that $\#\{A_i < \min(B_i, \hat{B}_i)\} \leq \#\{A_i < B_i\} = v$, which implies that

$$\#\{A_i < min(\hat{B}_i, B_i)\} \leq k \iff \#\{A_i < min(\hat{B}_i, B_i)\} \leq min(k, v)$$

This means that

$$\mathcal{P}[\#\{A_i < min(\hat{B}_i, B_i)\} \leq k \mid \#\{A_i < min(\hat{B}_i, B_i)\} \leq min(k, v) \wedge \#\{A_i < B_i\} = v] = 1$$

and

$$\mathcal{P}[\#\{A_i < min(\hat{B}_i, B_i)\} \leq min(k, v) \mid \#\{A_i < min(\hat{B}_i, B_i)\} \leq k \wedge \#\{A_i < B_i\} = v] = 1$$

We apply Lemma 1 II), obtaining

$$\mathcal{P}[\#\{A_i < min(\hat{B}_i, B_i)\} \leq k \ \mid \ \#\{A_i < B_i\} = v] =$$
$$\mathcal{P}[\#\{A_i < min(\hat{B}_i, B_i)\} \leq \min(k, v) \ \mid \ \#\{A_i < B_i\} = v]$$

Applying Lemma 2, we obtain

$$\mathcal{P}[\#\{A_i < min(\hat{B}_i, B_i)\} \leq \min(k, v) \ \mid \ \#\{A_i < B_i\} = v] \leq$$
$$\mathcal{P}[\#\{A_i > \hat{B}_i \wedge A_i < B_i\} \geq \max(0, v - k) \ \mid \ \#\{A_i < B_i\} = v]$$

Note that b_i is the score obtained with the real equivalent runtime t_2 as the stopping criterion, while in the case of $\hat{b}_i$, the stopping criterion is the estimated equivalent runtime $\hat{t}_2$. In a minimization context, $\hat{b}_i < b_i \implies \hat{t_2} > t_2$, because a better score can only be obtained with a longer runtime (a shorter runtime implies an equal or worse performance). Let us consider the following implications:

$$a_i > \hat{b}_i \wedge a_i < b_i \implies \hat{b}_i < b_i \implies \hat{t_2} > t_2$$

We infer that

$$\mathcal{P}[\hat{t_2} > t_2 \ \mid \ A_i > \hat{B}_i \wedge A_i < B_i] = 1$$

Applying Lemma 1 I), we obtain

$$\mathcal{P}[\#\{A_i > \hat{B}_i \wedge A_i < B_i \ \mid \ \#\{A_i < B_i\} = v\} \geq \max(0, v-k)] \leq$$
$$\mathcal{P}[\#\{\hat{t_2} > t_2 \ \mid \ \#\{A_i < B_i\} = v\} \geq \max(0, v-k)] =$$
$$\mathcal{P}[\#\{\hat{t_2} > t_2\} \geq \max(0, v-k)]$$

The estimated runtime $\hat{t}_2$ was computed with the equation in Definition 5 in Section 2, with an estimated probability that $\hat{t}_2 < t_2$ lower than 0.01. With this information, we apply Lemma 1 III) taking into account that $\mathcal{P}[\hat{t}_2 > t_2] < 0.01$:

$$\mathcal{P}[\#\{\hat{t_2} > t_2\} \geq \max(0, v-k)] <$$
$$\mathcal{P}[Bin(n, 0.01) \geq \max(0, v-k)]$$

□

Theorem 1. *Let $i \in \{1, .., n\}$ be n problem instances and let A and B be two optimization algorithms. Let a_i, b_i and $\hat{b}_i$ be the observed values of A_i, B_i and $\hat{B}_i$ respectively, $\forall i \in \{1, ..., n\}$. Let H_0 be the null hypothesis that implies the null distribution $Bin(n, 0.5)$ for the statistic $\#\{A_i < B_i\}$. Suppose that $A_i \neq B_i$ and $A_i \neq \hat{B}_i$. Then,*

$$\mathcal{P}[\#\{A_i < \hat{B}_i\} \leq k \ \mid \ H_0] \leq$$
$$\sum_{v=0}^{n} (1 - \mathcal{P}[Bin(n, 0.01) < \max(0, v-k)]) \cdot \mathcal{P}[Bin(n, 0.5) = v]$$

Proof. Let X, C be a two random variables, where S_C and S_X are the sets of all possible outcomes of C and X respectively. Consider the law of total probability [3]:

$$\forall x \in S_X \ , \ \mathcal{P}[X = x] = \sum_{c \in S_C} \mathcal{P}[C = c] \cdot \mathcal{P}[X = x \ \mid \ C = c]$$

Applying this formula, we obtain

$$\mathcal{P}[\#\{A_i < \hat{B}_i\} \leq k \ \mid \ H_0] =$$

$$\sum_{v=0}^{n} \mathcal{P}[\#\{A_i < \hat{B}_i\} \leq k \ \mid \ H_0 \wedge \#\{A_i < B_i\} = v] \cdot \mathcal{P}[\#\{A_i < B_i\} = v \ \mid \ H_0] =$$

Given that $\#\{A_i < B_i\} = v$, we can say that $\#\{A_i < \hat{B}_i\} \leq k$ is independent of H_0, because $\#\{A_i < \hat{B}_i\}$ is determined by how many times $\hat{t}_2 > t_2$ resulted in $A_i < B_i \wedge A_i > \hat{B}_i$ and $\hat{t}_2 < t_2$ resulted in $A_i > B_i \wedge A_i < \hat{B}_i$. Specifically, H_0 gives the prior probabilities of $A_i > B_i$, which are not relevant when we know that $\#\{A_i > B_i\} = v$. That gives us

$$\sum_{v=0}^{n} \mathcal{P}[\#\{A_i < \hat{B}_i\} \leq k \ \mid \ \#\{A_i < B_i\} = v] \cdot \mathcal{P}[\#\{A_i < B_i\} = v \ \mid \ H_0] <$$

Applying Lemma 3 and considering that H_0 implies the null distribution $Bin(n, 0.5)$ for the statistic $\#\{A_i < B_i\}$,

$$\sum_{v=0}^{n} \mathcal{P}[Bin(n, 0.01) \geq \max(0, v-k)] \cdot \mathcal{P}[\#\{A_i < B_i\} = v \ \mid \ H_0] =$$

$$\sum_{v=0}^{n} \mathcal{P}[Bin(n, 0.01) \geq \max(0, v-k)] \cdot \mathcal{P}[Bin(n, 0.5) = v] =$$

$$\sum_{v=0}^{n} (1 - \mathcal{P}[Bin(n, 0.01) < \max(0, v-k)]) \cdot \mathcal{P}[Bin(n, 0.5) = v]$$

□

AppendixF. Corrected critical values for the sign test

	α				α				α		
	0.05	0.01	10^{-3}		0.05	0.01	10^{-3}		0.05	0.01	10^{-3}
n	$\text{Crit}_c(\alpha)$			n	$\text{Crit}_c(\alpha)$			n	$\text{Crit}_c(\alpha)$		
2	-	-	-	33	10	8	6	64	24	21	18
3	-	-	-	34	11	9	7	65	24	21	18
4	-	-	-	35	11	9	7	66	25	22	19
5	0	-	-	36	12	10	7	67	25	22	19
6	0	-	-	37	12	10	8	68	25	23	19
7	0	-	-	38	12	10	8	69	26	23	20
8	1	0	-	39	13	11	8	70	26	23	20
9	1	0	-	40	13	11	9	71	27	24	21
10	1	0	-	41	14	12	9	72	27	24	21
11	2	1	-	42	14	12	9	73	28	25	21
12	2	1	0	43	15	12	10	74	28	25	22
13	2	1	0	44	15	13	10	75	28	25	22
14	3	2	0	45	15	13	11	76	29	26	23
15	3	2	0	46	16	14	11	77	29	26	23
16	4	2	1	47	16	14	11	78	30	27	23
17	4	3	1	48	17	14	12	79	30	27	24
18	4	3	1	49	17	15	12	80	31	28	24
19	5	3	2	50	18	15	12	81	31	28	25
20	5	4	2	51	18	16	13	82	32	28	25
21	5	4	2	52	18	16	13	83	32	29	25
22	6	4	3	53	19	16	14	84	32	29	26
23	6	5	3	54	19	17	14	85	33	30	26
24	7	5	3	55	20	17	14	86	33	30	27
25	7	5	4	56	20	18	15	87	34	31	27
26	7	6	4	57	21	18	15	88	34	31	27
27	8	6	4	58	21	18	15	89	35	31	28
28	8	6	5	59	21	19	16	90	35	32	28
29	9	7	5	60	22	19	16	91	36	32	29
30	9	7	5	61	22	20	17	92	36	33	29
31	10	8	6	62	23	20	17	93	36	33	29
32	10	8	6	63	23	20	17	94	37	34	30

Table F.7: The corrected critical values for the sign test. Rejecting H_0 when the observed number of cases that $a_i < \hat{b_i}$ is lower than or equal to $\text{Crit}_c(\alpha)$ has an associated probability of type I error lower than α.

	α				α				α		
	0.05	0.01	10^{-3}		0.05	0.01	10^{-3}		0.05	0.01	10^{-3}
n	$\text{Crit}_c(\alpha)$			n	$\text{Crit}_c(\alpha)$			n	$\text{Crit}_c(\alpha)$		
95	37	34	30	135	55	51	47	175	74	69	64
96	38	34	31	136	56	52	47	176	74	70	64
97	38	35	31	137	56	52	48	177	75	70	65
98	39	35	31	138	57	53	48	178	75	70	65
99	39	36	32	139	57	53	49	179	75	71	66
100	40	36	32	140	58	54	49	180	76	71	66
101	40	37	33	141	58	54	49	181	76	72	67
102	41	37	33	142	59	54	50	182	77	72	67
103	41	37	34	143	59	55	50	183	77	73	67
104	41	38	34	144	59	55	51	184	78	73	68
105	42	38	34	145	60	56	51	185	78	74	68
106	42	39	35	146	60	56	52	186	79	74	69
107	43	39	35	147	61	57	52	187	79	74	69
108	43	40	36	148	61	57	52	188	80	75	70
109	44	40	36	149	62	58	53	189	80	75	70
110	44	40	36	150	62	58	53	190	81	76	70
111	45	41	37	151	63	58	54	191	81	76	71
112	45	41	37	152	63	59	54	192	81	77	71
113	45	42	38	153	64	59	55	193	82	77	72
114	46	42	38	154	64	60	55	194	82	78	72
115	46	43	38	155	65	60	55	195	83	78	73
116	47	43	39	156	65	61	56	196	83	78	73
117	47	44	39	157	65	61	56	197	84	79	73
118	48	44	40	158	66	62	57	198	84	79	74
119	48	44	40	159	66	62	57	199	85	80	74
120	49	45	41	160	67	62	57	200	85	80	75
121	49	45	41	161	67	63	58	201	86	81	75
122	50	46	41	162	68	63	58	202	86	81	76
123	50	46	42	163	68	64	59	203	87	82	76
124	50	47	42	164	69	64	59	204	87	82	76
125	51	47	43	165	69	65	60	205	87	82	77
126	51	47	43	166	70	65	60	206	88	83	77
127	52	48	44	167	70	66	60	207	88	83	78
128	52	48	44	168	70	66	61	208	89	84	78
129	53	49	44	169	71	66	61	209	89	84	79
130	53	49	45	170	71	67	62	210	90	85	79
131	54	50	45	171	72	67	62	211	90	85	80
132	54	50	46	172	72	68	63	212	91	86	80
133	54	50	46	173	73	68	63	213	91	86	80
134	55	51	46	174	73	69	64	214	92	87	81

	α				α				α		
	0.05	0.01	10^{-3}		0.05	0.01	10^{-3}		0.05	0.01	10^{-3}
n	$\text{Crit}_c(\alpha)$			n	$\text{Crit}_c(\alpha)$			n	$\text{Crit}_c(\alpha)$		
215	92	87	81	255	111	105	99	295	129	123	116
216	93	87	82	256	111	105	99	296	130	124	117
217	93	88	82	257	111	106	100	297	130	124	117
218	93	88	83	258	112	106	100	298	131	125	118
219	94	89	83	259	112	107	101	299	131	125	118
220	94	89	83	260	113	107	101	300	131	125	119
221	95	90	84	261	113	108	101	301	132	126	119
222	95	90	84	262	114	108	102	302	132	126	120
223	96	91	85	263	114	109	102	303	133	127	120
224	96	91	85	264	115	109	103	304	133	127	120
225	97	91	86	265	115	110	103	305	134	128	121
226	97	92	86	266	116	110	104	306	134	128	121
227	98	92	87	267	116	110	104	307	135	129	122
228	98	93	87	268	117	111	105	308	135	129	122
229	99	93	87	269	117	111	105	309	136	130	123
230	99	94	88	270	118	112	105	310	136	130	123
231	99	94	88	271	118	112	106	311	137	130	124
232	100	95	89	272	118	113	106	312	137	131	124
233	100	95	89	273	119	113	107	313	138	131	125
234	101	96	90	274	119	114	107	314	138	132	125
235	101	96	90	275	120	114	108	315	138	132	125
236	102	96	90	276	120	115	108	316	139	133	126
237	102	97	91	277	121	115	109	317	139	133	126
238	103	97	91	278	121	115	109	318	140	134	127
239	103	98	92	279	122	116	109	319	140	134	127
240	104	98	92	280	122	116	110	320	141	135	128
241	104	99	93	281	123	117	110	321	141	135	128
242	105	99	93	282	123	117	111	322	142	135	129
243	105	100	94	283	124	118	111	323	142	136	129
244	105	100	94	284	124	118	112	324	143	136	129
245	106	100	94	285	124	119	112	325	143	137	130
246	106	101	95	286	125	119	113	326	144	137	130
247	107	101	95	287	125	120	113	327	144	138	131
248	107	102	96	288	126	120	113	328	145	138	131
249	108	102	96	289	126	120	114	329	145	139	132
250	108	103	97	290	127	121	114	330	145	139	132
251	109	103	97	291	127	121	115	331	146	140	133
252	109	104	97	292	128	122	115	332	146	140	133
253	110	104	98	293	128	122	116	333	147	141	133
254	110	105	98	294	129	123	116	334	147	141	134

	α				α				α		
	0.05	0.01	10^{-3}		0.05	0.01	10^{-3}		0.05	0.01	10^{-3}
n	$\text{Crit}_c(\alpha)$			n	$\text{Crit}_c(\alpha)$			n	$\text{Crit}_c(\alpha)$		
335	148	141	134	375	167	160	152	415	185	178	170
336	148	142	135	376	167	160	153	416	186	179	171
337	149	142	135	377	167	161	153	417	186	179	171
338	149	143	136	378	168	161	154	418	187	180	172
339	150	143	136	379	168	162	154	419	187	180	172
340	150	144	137	380	169	162	154	420	188	180	173
341	151	144	137	381	169	163	155	421	188	181	173
342	151	145	137	382	170	163	155	422	189	181	173
343	152	145	138	383	170	163	156	423	189	182	174
344	152	146	138	384	171	164	156	424	189	182	174
345	152	146	139	385	171	164	157	425	190	183	175
346	153	146	139	386	172	165	157	426	190	183	175
347	153	147	140	387	172	165	158	427	191	184	176
348	154	147	140	388	173	166	158	428	191	184	176
349	154	148	141	389	173	166	159	429	192	185	177
350	155	148	141	390	174	167	159	430	192	185	177
351	155	149	141	391	174	167	159	431	193	186	177
352	156	149	142	392	174	168	160	432	193	186	178
353	156	150	142	393	175	168	160	433	194	186	178
354	157	150	143	394	175	169	161	434	194	187	179
355	157	151	143	395	176	169	161	435	195	187	179
356	158	151	144	396	176	169	162	436	195	188	180
357	158	152	144	397	177	170	162	437	196	188	180
358	159	152	145	398	177	170	163	438	196	189	181
359	159	152	145	399	178	171	163	439	197	189	181
360	159	153	146	400	178	171	163	440	197	190	182
361	160	153	146	401	179	172	164	441	197	190	182
362	160	154	146	402	179	172	164	442	198	191	182
363	161	154	147	403	180	173	165	443	198	191	183
364	161	155	147	404	180	173	165	444	199	192	183
365	162	155	148	405	181	174	166	445	199	192	184
366	162	156	148	406	181	174	166	446	200	192	184
367	163	156	149	407	182	174	167	447	200	193	185
368	163	157	149	408	182	175	167	448	201	193	185
369	164	157	150	409	182	175	168	449	201	194	186
370	164	157	150	410	183	176	168	450	202	194	186
371	165	158	150	411	183	176	168	451	202	195	187
372	165	158	151	412	184	177	169	452	203	195	187
373	166	159	151	413	184	177	169	453	203	196	187
374	166	159	152	414	185	178	170	454	204	196	188

	α				α				α		
	0.05	0.01	10^{-3}		0.05	0.01	10^{-3}		0.05	0.01	10^{-3}
n	$\text{Crit}_c(\alpha)$			n	$\text{Crit}_c(\alpha)$			n	$\text{Crit}_c(\alpha)$		
455	204	197	188	471	212	204	196	487	219	211	203
456	205	197	189	472	212	205	196	488	220	212	203
457	205	198	189	473	213	205	196	489	220	212	204
458	205	198	190	474	213	205	197	490	221	213	204
459	206	198	190	475	213	206	197	491	221	213	205
460	206	199	191	476	214	206	198	492	221	214	205
461	207	199	191	477	214	207	198	493	222	214	206
462	207	200	192	478	215	207	199	494	222	215	206
463	208	200	192	479	215	208	199	495	223	215	206
464	208	201	192	480	216	208	200	496	223	216	207
465	209	201	193	481	216	209	200	497	224	216	207
466	209	202	193	482	217	209	201	498	224	217	208
467	210	202	194	483	217	210	201	499	225	217	208
468	210	203	194	484	218	210	201	500	225	217	209
469	211	203	195	485	218	211	202				
470	211	204	195	486	219	211	202				