Submitted to the Annals of Applied Statistics

APPENDICES OF "ON THE FAIR COMPARISON OF OPTIMIZATION ALGORITHMS IN DIFFERENT MACHINES"

BY ETOR ARZA[1], JOSU CEBERIO[2], EKHIÑE IRUROZKI[3] AND ARITZ PÉREZ[1]

[1]*BCAM - Basque Center for Applied Mathematics, Spain, earza@bcamath.org; aperez@bcamath.org*

[2]*University of the Basque Country UPV/EHU, Spain, josu.ceberio@ehu.eus*

[3]*Télécom Paris, France, irurozki@telecom-paris.fr*

APPENDIX A: 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[1] and perform the one-sided

Keywords and phrases: Algorithms, Optimization, Benchmarking, Statistical Tests.

[1]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.

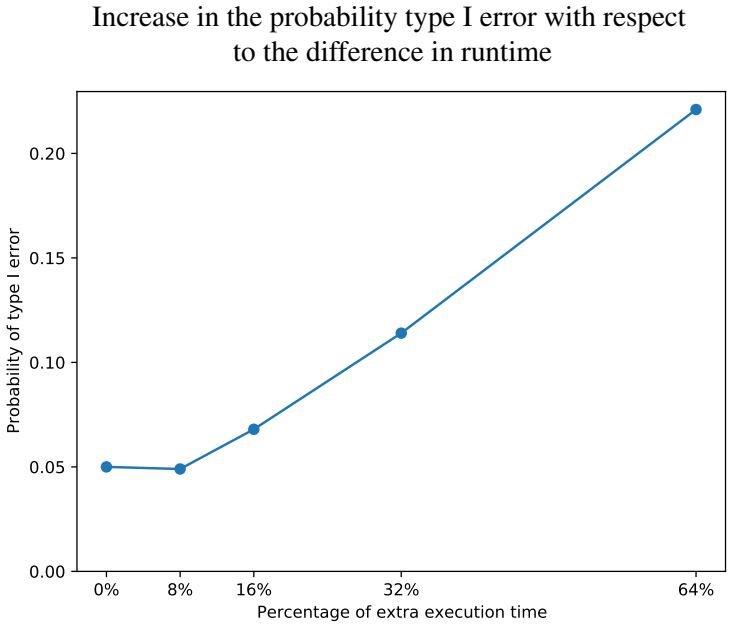

Figure 1: Probability of type I error in the one-sided sign test when comparing two identical random search algorithms. One of the algorithms is given extra runtime, according to the X-axis. The test is applied to a set of 16 problem instances.

sign test Conover (1980)[2] (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 1 shows the estimated probability of type I error. Notice that the 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.

APPENDIX B: THEORETICAL AND EXPERIMENTAL JUSTIFICATION OF ASSUMPTION 1

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. This hypothesis is the basis of Assumption 1.
To experimentally study this assumption, we compute the correlation of the runtime that several optimization processes have on two machines. Specifically, we computed the correlation of 64 different optimization processes (see Appendix C for additional details on the optimization processes) 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 Zou, Tuncali and Silverman (2003) (not necessarily proportional) 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 2) will almost be aligned in a line Zou, Tuncali and Silverman (2003). 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 2.
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 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 $(t(M_1, \rho'), t(M_2, \rho'))$:

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

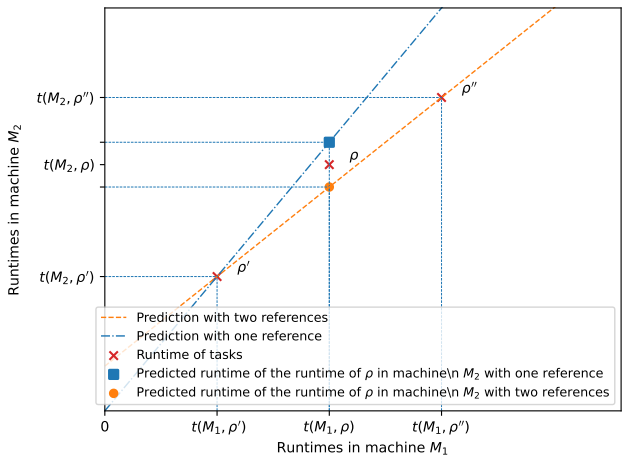

Figure 2: Estimation of the runtime of an optimization process ρ with one ρ' or two ρ', ρ'' reference optimization processes. The x-axis represents the runtime in machine M_1, while the y-axis is the runtime in machine M_2. The runtime of the optimization process ρ is estimated for machine M_2.

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

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

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

Which is exactly Assumption 1.

APPENDIX C: OPTIMIZATION PROCESSES

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. In total, we considered 64 optimization processes, executing 4 algorithms in 16 problem instances. The optimization process ρ' is the sequential execution of these 64 optimization processes.

Problem instances: We solved four types of optimization problems (all of them are permutation problems): the traveling salesman problem Goldberg and Lingle (1985), the quadratic assignment problem Koopmans and Beckmann (1957), the linear ordering problem Ceberio, Mendiburu and Lozano (2015) and the permutation flowshop scheduling problem Gupta and Stafford (2006). For each of these four problem types, we chose 4 problem instances, as listed in Table 1.

Optimization algorithms: Each of the 16 problem instances was optimized with four optimization algorithms. These optimization algorithms are random search and local search with

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 1
The list of 16 problem instances and their size.

Machines

CPU model name	PassMark score
Intel i5 470U	539
Intel Celeron N4100	1012
AMD A9 9420 with Radeon R5	1344
AMD FX 6300	1486
Intel i7 2760QM	1559
Intel i7 6700HQ (2.60GHz)	1921
Intel i7 7500U	1955
AMD Ryzen7 1800X	2185

TABLE 2
The list of 8 machines used in the experimentation and their speed score, measured in terms of PassMark single thread score.

three different neighborhoods: swap, interchange and insert Schiavinotto and Stützle (2007); Ceberio et al. (2015). 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 2 lists the CPU models of these machines, as well as their single thread PassMark CPU scores.

APPENDIX D: 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 Conover (1980) 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 Conover (1980), 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.

APPENDIX E: PROOF OF EQUATION 5.

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 1. (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 $\mathcal{P}(\hat{t}_2 < t_2) < p_\gamma$. Let us now prove Equation 5.

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,*

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

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 true 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 under which the statistic $\#\{A_i < B_i\}$ follows the null distribution $Bin(n, 0.5)$. 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 Beyer (1991):

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

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

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

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

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

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

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

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

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

□

REFERENCES

BEYER, W. H. (1991). *Standard Probability and Statistics: Tables and Formulae*. CRC Press.

CEBERIO, J., MENDIBURU, A. and LOZANO, J. A. (2015). The Linear Ordering Problem Revisited. *European Journal of Operational Research* **241** 686–696.

CEBERIO, J., IRUROZKI, E., MENDIBURU, A. and LOZANO, J. A. (2015). A Review of Distances for the Mallows and Generalized Mallows Estimation of Distribution Algorithms. *Computational Optimization and Applications* **62** 545–564.

CONOVER, W. J. (1980). Practical Nonparametric Statistics.

GOLDBERG, D. E. and LINGLE, R. JR. (1985). Alleles, Loci and the Traveling Salesman Problem. In *Proceedings of the 1st International Conference on Genetic Algorithms* 154–159. L. Erlbaum Associates Inc.

GUPTA, J. N. D. and STAFFORD, E. F. (2006). Flowshop Scheduling Research after Five Decades. *European Journal of Operational Research* **169** 699–711.

KOOPMANS, T. C. and BECKMANN, M. (1957). Assignment Problems and the Location of Economic Activities. *Econometrica* **25** 53.

SCHIAVINOTTO, T. and STÜTZLE, T. (2007). A Review of Metrics on Permutations for Search Landscape Analysis. *Computers & operations research* **34** 3143–3153.

ZOU, K. H., TUNCALI, K. and SILVERMAN, S. G. (2003). Correlation and Simple Linear Regression. *Radiology* **227** 617–628.