Graduate Program in Electrical Engineering

SAMPLE SIZE ESTIMATION FOR POWER AND ACCURACY IN THE EXPERIMENTAL COMPARISON OF METAHEURISTICS

FERNANDA C TAKAHASHI

2018

Universidade Federal de Minas Gerais
Graduate Program in Electrical Engineering

Thesis

Sample size estimation for power and accuracy in the experimental comparison of metaheuristics

Fernanda C Takahashi
Supervisor: Prof. Dr. Felipe Campelo

Belo Horizonte, Brazil
2018

"Sample Size Estimation for Power and Accuracy in the Experimental Comparison of Metaheuristics"

Fernanda Caldeira Takahashi

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 03 de dezembro de 2018.

Por:

[illegible]

Prof. Dr. Felipe Campelo França Pinto
DEE (UFMG) - Orientador

[illegible]

Prof. Dr. Claus de Castro Aranha
Dep. Computer Science (Tsukuba University, Japão)

[illegible]

Prof. Dr. Thiago Ferreira de Noronha
DCC (UFMG)

[illegible]

Prof. Dr. Luiz Henrique Duczmal
Dep. Estatística (UFMG)

[illegible]

Prof. Dr. Helio José Corrêa Barbosa
(LNCC)

That's the great secret of creativity. You treat ideas like cats: you make them follow you.

Ray Bradbury, *Zen in the Art of Writing*

ABSTRACT

Experimental algorithmics encompasses the study of guidelines and methods for computational evaluation of algorithms. In the optimization field, it is useful for testing the performance of algorithms when solving a certain type of problem. In this work we develop a methodology for generating adequate experimental designs for comparing the performance of optimization metaheuristics, with a focus on statistical power and accuracy in parameter estimation. In particular, we deal with sample size estimation for experiments involving optimization algorithms, both in terms of within-instance repeated executions and the number of instances required. A statistically sound methodology is presented for sample size calculation, allowing relevant comparisons between the performances of two algorithms for a given class of problems. The methodology's effectiveness is validated using simulated models and exemplified with two case studies. The proposed methodology was implemented in the form of an open source R package, published in the CRAN repository.

RESUMO

Experimentação algoritmica contempla o estudo de diretrizes e métodos para avaliação computacional de algoritmos. No campo da otimização, ela é útil para testar o desempenho de algoritmos ao resolver classes específicas de problemas. Nesse trabalho estamos desenvolvendo uma metodologia para geração planejamentos experimentais adequados para comparação de desempenho de meta-heurísticas, com um foco em potência estatística e precisão na estimação de parâmetros. Em particular, lidamos com estimação do tamanho amostral para experimentos que envolvem algoritmos de otimização, tanto em termos do número de execuções em uma mesma instância quanto do número de instâncias necessárias. Uma metodologia estatisticamente válida é apresentada para o calculo de tamanho amostral, permitindo comparações relevantes entre as performances de dois algoritmos para uma dada classe de problemas. A eficácia da metodologia é validada usando modelos simulados e exemplificada com dois estudos de caso. A metodologia proposta foi implementada na forma de pacote em R código aberto, publicado no repositório CRAN.

CONTENTS

1 INTRODUCTION

Science and everyday life cannot and should not be separated.

Rosalind Franklin, *Rosalind Franklin: the Dark Lady of DNA*

In the optimization field, some problems are deemed to difficult for a solution to be found in practical time using exact methods, such problems are commonly solved using heuristics. Which are specialized sets of procedures to solve a single given problem, such as vehicle routing problems (VRP) or travelling salesman problem (TSP) [42]. When a strategy is not problem-specific, but rather a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms it is called a metaheuristic [71]

When dealing with metaheuristics for optimization, where repeated executions of the same algorithm may not produce the same results, statistical methods are important to understand the intrinsic variability in such algorithms and their behaviour [58]. However, the planning and analysis of experimental comparison of different metaheuristics are frequently made in an incomplete, or even wrong, manner. It is possible to conjecture that several works in which the algorithms are poorly described and the experiments lack reproducibility may have led to wrongful or invalid conclusions[76].

Although not widely used, guidelines for such experimental design have existed for a while. Barr *et al.* [3] clearly define the necessary steps for a proper statistical analysis of experiments: (i) objective definition; (ii) choice of the performance metrics and factors to be explored; (iii) design and execution of the experiment; (iv) data analysis and conclusions; (v) publication of experimental results. A proper disclosure of the data also has its own rules to be followed: (a) reproducibility, (b) detailed specification of the important factors; (c) precision on reported run times; (d) disclosure of the configuration parameters; (e) use of statistical techniques of experimental design and analysis; (f) comparison with other methods; (g) reduction of variability; and (h) generation of a result report.

Some other works propose different ways of doing more rigorous designs and analysis of computational experiments[2, 58, 41]. However, in many cases, characteristics of experimentation already known in other fields are completely ignored, such as the importance of defining the magnitude of effects of practical interest, or sample size calculation.

With a careful examination of the literature, it is noticeable that the experimental analysis in the field of metaheuristics tends to be lacking when considering the tools available in other applied fields, such as clinical trials [72]. Currently, there is no standard way of comparing these algorithms, leaving to the researchers the responsibility of designing their own methodology for comparing the algorithms.

Nevertheless, there is no controversy regarding the need of clear designs and proper data analysis, as they are fundamental for generating high-quality research in the field. A helpful first step, for instance, would be for researchers to provide enough information so that their work could be properly evaluated and replicated.

In this work, another step is proposed to allow reliable and standardized statistical comparisons of the performance of different metaheuristics. The comparison of stochastic optimization algorithms is usually done considering the value of the solution obtained for the optimization problem [1], which is commonly described as a minimization problem as defined in (1) and (2).

$$\min f(\mathbf{x}) \tag{1}$$

$$\text{subject to} \begin{cases} g_i(\mathbf{x}) \leq 0 & i = 1, ..., p \\ h_j(\mathbf{x}) = 0 & j = 1, ..., q \\ x \in \mathcal{X} \end{cases} \tag{2}$$

where $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ is the objective function to be minimized, $g_i(\cdot)$ and $h_j(\cdot)$ are the inequality and equality constraints, respectively, and $\mathcal{X}$ represents the space where the problem is defined.

Many metaheuristics used to solve this type of problem are stochastic in nature, and as such can return different solutions when run multiple times on a given instance. Each independent execution of an algorithm in a given instance will be called a repetition throughout this work. A summary statistic of a given set of repetitions, which will be used as a single observation for the statistical comparisons treated in this work, will be called a replication or replicate.

1 The comparison of heuristic algorithms can be done in terms of the convergence time of the algorithm, or any other quality indicator.

In this work, we consider the algorithm comparison problem as the question of determining whether two (or more) algorithms present differences in terms of a given performance parameter (e.g., mean or median performance) for a given problem class. We approach this problem from a statistical inference perspective, performing significance testing on samples composed of observed values of algorithmic performance for sets of problem instances, to infer their differences for the problem class represented by those instances. Let the set of algorithms to be compared, denoted as $\theta_1, ..., \theta_m$, to be evaluated using a set of instances $\gamma_1, ..., \gamma_n$, which are assumed as being a representative and independent sample of the problem class Γ. For each execution of algorithm θ_i on the instance γ_j, there is a value of the considered performance metric for the repetition y_{ijk}. Using the repetitions, an estimate can be calculated for the average performance value of θ_i on γ_j, $\mu_{\theta_i;\gamma_j}$. [2]. By estimating these values for several instances γ_j, it is possible to generate an estimate for the overall average performance of algorithm θ_i on the whole problem class Γ, which we will call here $\mu_{\theta_i;\Gamma}$. Given these definitions, the algorithm comparison problem can be defined here as the test for differences among the values of $\mu_{\theta_i;\Gamma}$, $i = 1, \ldots, m$ and, if differences are detected, the determination of their magnitudes and practical relevance.

There is a common misconception that this work tries to address, which is related to the sample size when comparing algorithms. In fact, there are methodologically sound ways to calculate the required number of repetitions of an algorithm solving an instance, as well as the minimal number of problem instances (number of replications) needed to perform statistical inference on the differences of algorithmic performance on a given problem class, so that the statistical parameters of the test (confidence level and statistical power for a given effect size) can be controlled at desired levels. Regardless, the standard approach tends to be one of maximizing the number of instances limited only by the computational budget available; and of using "standard" values for the number of repeated runs (30, 50, occasionally 100 repetitions or more). With an increased sample size, the probability of detecting a difference in performance between algorithms does increases, but it does not imply that sample sizes need to be arbitrarily large to yield high quality solutions.

The main objective of this work is to develop a methodology for sample size and power estimation in the experimental comparison of algorithms, so that the number of repeated runs

2 This average can be the mean, median, trimmed mean, or any other location parameter. Throughout this text we focus on comparisons of average performance, but in principle the ideas presented here could be applied to any other parameter of interest - e.g., best or worst case performance, variance, etc..

and the required amount of test instances can be determined based on the desired statistical properties of the comparison. This methodology was implemented in the form of an open source package in R, so that researchers and practitioners in the field of metaheuristics can immediately employ its concepts in their work. The proposed methodology iteratively samples the algorithms on each instance, so that, the accuracy of estimates can be improved while attempting to use the smallest possible total number of runs. The proposed approach to the design of algorithmic experiments calculates the sample sizes for a given set of algorithms and problems while considering other specifications such as standard error, test power, confidence level and magnitude of differences of practical significance (i.e., minimally relevant *effect size*). In this way, the comparison of algorithms can be made considering desired statistical parameters which leads to stronger conclusions and more accurate information on the behaviour of the algorithms. Although this work focuses in heuristics and metaheuristics, the methodology proposed can be used in any algorithmic comparison involving experimental aspects. That being said, the term metaheuristic will be the one used along this text to emphasize the context in which the method was conceived.

The second chapter of this work presents a brief review of the literature on experimental algorithmics, as well as an analysis on how the experiments have been done lately. Also, a short theoretical background on the statistics used for this work is provided at the end of this chapter. In the third chapter the proposed method is explained, the equations used to estimate the sample sizes and the pseudo-codes are presented. The fourth chapter contains the experimental evaluations of the method, presenting the results of tests that use a synthetic statistical model to simulate algorithms. Finally, the fifth chapter presents a discussion of the current state of this work and the conclusions that can be drawn from its current state. The main ideas for continuity and a proposed schedule for the completion of this research are also presented. At the end of the thesis two appendices can be found. The first one, Appendix A, gives an brief introduction of statistical concepts useful to the comprehension of this work. And the second one presents the usage of the caiseR package, which was developed thorough this work.

2 LITERATURE REVIEW

Research is formalized curiosity. It is poking and prying with a purpose.

Zora Neale Hurston, *Dust Tracks on a Road*

2.1 OPTIMISATION AND HEURISTICS

Amongst optimisation problems, there are some which cannot be solved to optimality by exact methods in reasonable time, which are known as hard optimisation problems. Thus, a category of algorithms, named heuristics, was introduced as an attempt to provide good solutions in reasonable time for these problems. One important difference between them lies in the fact that exact methods are frequently based on optimization methods that calculate a search direction based on the function gradient, therefore requiring a differentiable problem. On the other hand heuristics can solve non differentiable problems, using intuitive strategies that are problem specific, which start from an arbitrary initial solution and then use the knowledge obtained from the previous solutions to guide the search[50]. Furthermore, heuristics do not guarantee that an optimal solution is found, but they can often return some solution in viable time [70].

metaheuristics are framework for building heuristics, which are not problem specific, hence possessing parameters to be fitted to each problem. Therefore, a metaheuristic is an algorithm that can solve a wide range of problems, but needs to be tuned to work properly [10]. Since the 1950s[60] they have been increasingly used as an interesting alternative to solve complicated optimisation problems that would be impossible or impractical to solve to optimality [56, 64, 26, 46, 68, 69, 24, 22].

A common trait of these algorithms is their stochastic behaviour, mainly arising from their use of a random component whose variability helps exploring the search space, but which also results in distinct solutions for different runs of these algorithms.

Since metaheuristics are complex tools that are often difficult to describe in purely analytical terms, and that can yield results that vary stochastically between runs, it is clear the necessity of an appropriate experimental analysis. Yet, it is still common to find works in the literature using incomplete or incorrect experimental setups and analysis techniques for the evaluation and comparisons of performance. Furthermore, it is also possible to argue that the low methodological standards may have contributed to the emergence of a great quantity of "novel" algorithms that claim superiority to existing methods, even though they represent essentially the same ideas [70, 34].

2.2 ALGORITHM EXPERIMENTATION

The use of statistical methods for experimentation with algorithms is not recent. One early example is the book *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modelling* [38] released back in 1991. This book presents methods for evaluating algorithms, as well as some experimental designs and analysis techniques that can be used for testing them. A proper statistical analysis is essential for the quantification of the conclusions drawn from the data. Done correctly, this analysis involves experimental design, choosing a model consistent with the experiment, testing the presence of interesting effects, and the calculation of confidence intervals for the size of the effects found. Other important aspects of experimental research on algorithms, such as the reporting of p-values and the validation of assumptions of the statistical models and tests used, were also highlighted by Coffin and Saltzman [16] as early as 2000.

When experimenting with heuristics, the proper steps of computational experimentation were described by Barr *et al.* in 1995 [3]. It is stated that statistical methodology is the proper approach to conduct the algorithm analysis because it is an objective way to design experiments and analyse the data gathered. Besides, with the correct use of a statistical methodology, it is possible to obtain more information from the experiment, without increasing the test effort. Around the same time, a work by Catherine McGeoch [53] exposed some issues still common in algorithm experimentation and comparison, such as implementation-specific behaviour and generalization of results. Also, as mentioned by McGeoch, Hooker[35] noticed the difficulty in extrapolating the results obtained with one class of instances to others.

In 2006, Demšar showed how to perform statistical comparison of multiple classifiers on multiple datasets [19], a problem analogous to comparing optimisation algorithms on multiple problem instances. In that work, he concluded that non-parametric tests offer more suitable results, due to weaker assumptions on the distributional properties of the data, and should be the ones used. Demšar's work mentions that the assumptions of parametric tests are frequently violated in algorithmic experimentation, and that in these cases non-parametric alternatives frequently present better statistical properties. The work fails to mention, however, that the non-parametric methods suggested are also based on certain distributional assumptions [66] which, albeit less restrictive, are also usually violated and widely ignored in the field. To compare two classifier he applies the Wilcoxon signed rank test, and the Friedman test for multiple classifiers. His proposals were used not only in machine learning, to test classifiers, but in a substantial part of algorithm testing and comparison, and represents possibly the most influential reference on the statistical testing of algorithms to date.

Still in the field of algorithm experimentation, several papers on the use of statistical analysis for metaheuristics were published over the past decade by a group based primarily in the University of Granada in Spain. Under the same motivation as Demšar's, namely the usual non-normality of algorithmic data, this group advocates the use of non-parametric tests [29, 28, 30, 21, 20], which were not commonly used at that time. In these works it was proposed the use of non-parametric tests in different heuristics, such as evolutionary, genetic based machine learning or swarm intelligence algorithms.

Another non-parametric approach that is widely used in other fields, but has so far attracted little attention in experimental algorithmics, is bootstrapping. It consists in estimating a distribution according to random re-sampling of data generated by executions of the algorithm. This way, the null distributions of the test statistics used to compare the performance of algorithms can be built independently of the actual distribution of the data. Carrano *et al.* [15] use bootstrapping to empirically estimate the probability density function (PDF) of the mean of the comparison criteria for each algorithm, with which the statistical tests are performed. The test to rank the algorithms is a non-parametric one, and another (parametric) test is used to verify if the ranking is stable or if more executions are needed.

Still considering ranking algorithms, Krohling *et al.* [41] proposed a method for comparing multiple algorithms in multiple benchmarks which uses closeness between the obtained solutions to rank the algorithms. The algorithms tested are, usually, stochastic, therefore an approximation of a Gaussian distribution estimated from the results is used as the solution. The Hellinger distance is then used to measure the gap between the distributions. With these measurements, a coefficient of closeness can be computed using the distance from each distribution to the two most extreme ones, on every benchmark.

Some other works have shown more ways for statistically comparing algorithms. del Amo *et al.* [2] use a full factorial design to generate the data for comparing multiple algorithms on multiple problem instances, and then apply rank-based methods for the inferential procedures. Pais *et al.* [58], on the other hand, employs a 2^k full factorial design and parametric tests to determine which and how some implementation factors interfere with the results. The data is adjusted with a logarithmic transformation trying to obtain a distribution that more closely resembles a Gaussian when testing the assumptions; and by removing the residuals with a large Cook's distance, if the removed observations do not influence the results.

2.2.1 What to compare

Depending on what is being studied, a wide range of characteristics can be used to compare the algorithms. Part of them are particular for some types of algorithms such as the convergence performance through evolutionary search[20]; or speed-ups in parallel algorithms [58]. Most works, however, tend to compare some traits already considered important in optimization algorithms. Amongst them, two criteria commonly used are average solution quality and convergence speed [16].

Comparing speed is complex because of its machine/implementation dependence [16, 58]. To deal with this complication, some strategies have been proposed for implementation-independent evaluations of speed, such as counting the number of function evaluations. When comparing algorithms using their running time, it is important to be aware that the distribution of the population of running times will hardly follow a normal distribution. Instead their sample distribution of means will tend to normality as the sample size increases [58], enabling analysis assuming normality as a premise [55]. It is a common practice to omit results that do not converge to an optimal or sub-optimal solution. This however, may lead to inaccurate conclusions about the heuristic behaviour and its precision is not guaranteed, since the non-converged runs imply in an error even if it cannot be measured: running an algorithm 100 times, in which it converges in 10, and reach optimal solution in 8 of them, does not mean it has an 80% efficiency rate. Analogously, if only the convergence rates of runs that achieved optimal solution is used, it does not characterize properly the algorithms convergence speed behaviour.

Solution quality comparison is relevant only when dealing with algorithms that do not guarantee an optimal solution, which is a common trait in heuristics and metaheuristics. There are many different methods to measure the performance of heuristics, a usual one is to offer a percent deviation of the optimal solution (or a upper/lower boundary); the success rate can also be used, or even an *offline error*, where the error of the best solution achieved is considered for each pair function/algorithm. It is noteworthy the importance of choosing a quality measure that represents the *quantity of interest*[1] for the comparison. While comparisons of expected performance (mean or median) are almost universal in the metaheuristics literature, there are many cases in which the quantity of interest may be different - best or worst-case performances, consistency of performance, or expected time to convergence are some examples of comparisons that are mostly absent from works in experimental comparative algorithmics [23].

Either way, for these measures to have any practical significance an appropriate sample size is needed. If the experiment is made with a small sample, the desired power for a given effect size of practical importance might not be reached, thus the conclusions drawn will not be as strong as they could be. On the other hand, studies performed with overestimated sample sizes may detect a strong statistical significance even for effects that are completely irrelevant from a practical point of view [51, 57, 8, 32]. Closely associated with the concept of an adequate sample size is the idea of a minimally interesting effect size, that is, the minimal difference on average (or best, worst, etc.) performance that represents any practical consequence in terms of algorithm performance.

2.3 SAMPLE SIZE

The chances of detecting an effect, i.e., a difference among the algorithms in terms of a given parameter of interest, increase as the sample size grows [5]. Hence the analysis obtained with a large sample size is commonly considered stronger than one with a small one. However, statistical analysis made with moderate sized data sets, or even small ones, can be as useful

1 quantitative metric of the evaluated characteristic

and convincing as the ones made with very large data sets [16]. Seeing that it is important to detect effect sizes with practical consequences, and not just any effect, it is unnecessary to indefinitely grow the sample size. It is enough to determine the amount of observations needed to detect a minimally interesting effect size with a predefined test power. This knowledge allows for a more efficient use of resources as well as preventing p-hacking [2] by specifying the desired parameters to the experimental analysis. Running the algorithms only the number of times necessary to draw correct conclusions saves both time and computational resources, and simplifies the analysis of the results by providing a threshold of relevance in the differences of performance among algorithms.

Despite its importance for algorithm comparison, sample size estimation has been generally poorly addressed in the field of metaheuristics. As a rule, it is a common recommendation to use the largest possible sample size, to increase the power of the statistical tests [29, 28, 21]. However, such an approach may not be ideal since it is possible to reach an appropriate power and effect size with smaller sample sizes. Moreover, the question of "how large is large enough" in terms of sample size for providing statistical power is heavily dependent on the magnitude of the differences one is interested in detecting - even "large" samples may be insufficient, resulting in hopelessly underpowered studies [52, 51, 74]. Even with the existence of works focused on small-samples[78], the use of very large data sets is constantly reinforced despite the known computational difficulty. In the article *An algorithm comparison for dynamic optimization problems* [2], for example, it is alleged that the researcher should gather as much data as possible in order to guarantee safe practical conclusions and the only limitation to data gathering would be practical such as time constraints, or computational resources. However, too large of sample sizes can be used to declare a significance for differences without any practical effect, which, in many cases, could be easily attributed to subtle tuning differences instead of differences on the algorithms themselves. Moreover, we argue that the practice of defining a threshold of practical interest *a priori* may benefit the development of the field as a whole, by discouraging the reporting of inconsequential improvements as evidence of superiority, a practice that is called (in a somewhat tongue-in-cheek manner) "up the wall gaming" [70].

In this work, the number of replications refers to the number of instances needed to achieve a certain effect size and power while comparing algorithms. It should not be confused with

2 p-hacking is a type of bias that occurs when the data or statistical analysis is carefully selected until non-significant results become significant[33]

the number of repetitions, which are independent runs of one algorithm on each instance. These repeated measures cannot be used to define the algorithm behaviour, instead they are useful to obtain an estimate of the experimental estimation error on the performance of each algorithm on each problem instance [58], as well as to provide additional information, e.g., on the performance variability of each algorithms on each instance.

On calculating the appropriate number of repetitions on each instance, Jain [38] shows how to determinate this number for a desired accuracy and confidence, given the distribution mean and standard deviation. This method requires not only some knowledge of the tested distribution, but also normality of the sampling distribution of means. Therefore, an alternative is to generate many solution sets and use their average to estimate the distribution mean and deviation, which guarantees normality under the relatively mild assumptions of the Central Limit Theorem (CLT), when the number of samples is large enough. However, there is no established rule on how to estimate the necessary number of repetitions for testing heuristics and metaheuristics. Usually, for such non-deterministic algorithms, a large amount of runs per instance is believed to be needed [15, 58]. The work by Birattari [8], on the contrary, advocates a greater importance to number of instances than on many repetitions in a single one, and demonstrates that the optimal trade-off between number of instances and number of runs is to maximize the quantity of test instances, running each algorithm a single time on each one.

On the other hand, it is the sample size, or number of instances, that will interfere with the variance of the expected performance estimate [8]. While it is usually considered that *the more test problems, the more informative the study*[3], the sample size calculation often gives place to benchmarks with predefined sizes or arbitrary numbers of generated instances. In these cases, when the sample size is fixed, calculating the test power is important to give an idea of the sensitivity of the experiment, which may provide an indicator of the reliability of the experimental conclusions [51].

In this scenario, the performance estimation obtained with a particular benchmark provides information on how the algorithm performs on such instance set (or, at most, on the problem class for which the benchmark set is representative), and not on how they perform in general. Although providing a generalizable conclusion is known to be a complex problem [53, 6], some strategies have been created to enable it. For example, Bartz-Beielstein [6] proposed a method to generate instances estimating a model from real world data or artificial problem generators. With this model it would be possible to generate as many random test instances as needed and, also, the results are general in the real-world setting where the studied algorithms will be applied. Therefore, this method is useful, not only for generalization, but also to provide the sample size necessary for testing with controlled errors of type I and type II (see section A.3.1).

2.4 RECENT WORKS

Most works published recently on the proposal and comparison of algorithms still appear to fall into pitfalls of experimental analysis that are widely disseminated in the field. To provide a short overview of the state of experimental design in the research on metaheuristics, particularly with regards to statistical power and sample size, a survey of recent papers on some of the most prestigious journals in the field was performed. This review included all papers dealing with the experimental comparison of algorithms published in the following venues: IEEE Transaction on Evolutionary Computation (vol.20) from April 2016, Journal of Heuristics(2016) and Information Sciences(2016).

Many of these works use an arbitrary number of both instances and repetitions [45, 1, 81, 48, 59, 79, 67, 40]. The chosen values for the former, usually, are not explained: some use a high number of executions [40], others obey the common 30-run rule-of-thumb [81, 48, 59, 79]. The said rule is a misinterpretation of a CLT statement which suggests that the sampling distribution of the means of even markedly asymmetric distributions, such as the exponential, will converge to an approximately normal shape for sample sizes greater than approximately 30, provided that no outliers are present. This rule, however, rests on assumptions on the behaviour of the data that can be easily violated, so caution is required when using the rule of 30 [51].

The number of instances, on the other hand, is typically arbitrary, mostly because benchmark sets are used and they commonly have a finite size. However, these benchmarks are often merged in order to increase the instance number and, with such heterogeneous sets [79], it is difficult to realize the purpose of the experiment or, more importantly, the features of the class of problems for which the conclusions of a given study could be generalized to. The common approach is to verify the difference of means for each problem in order the demonstrate the advantage of a given method, which does not necessarily provide insights into the general behaviour of the algorithm for any specific problem class.

The comparison of the algorithms is usually performed with a test of significance, such as Wilcoxon-Mann-Whitney's test on paired samples[77, 45, 48, 59, 79]. However, testing the results obtained with an arbitrary sample might lead to the detection of effects particular to the benchmark or, depending on the sample size, to the detection of irrelevant effects or the failure to detect important ones due to lack of appropriate power. Even though the significance level of the test is often given as an experimental parameter, the power of the test is utterly ignored, which means that the experiment can be considered valid, but the type II error is not controlled. Ideally, the desired test power, together with the difference of minimal practical relevance should determine the number of instances needed, or would be calculated, for a particular sample size, as a sensitivity measure of the test.

Even with some aspects to be improved, experimental designs used in algorithm comparisons and evaluations have developed over time. Recent works have been using statistical methods more appropriate to compare algorithms and to validate their claims. Such advances make not only these studies more reliable, but contribute to making the whole field more scientifically accurate.

2.5 THEORETICAL BACKGROUND

2.5.1 The Algorithm Comparison Problem

Let $\Gamma = \{\gamma_1, \gamma_2, \ldots\}$ represent a *problem class* consisting of a set of (possibly infinitely many) problem instances γ_j which are of interest as a group (e.g., the set of all possible TSP instances within a given size range); and let $\mathcal{A} = \{a_1, a_2, \ldots\}$ denote a set of algorithms capable of returning tentative solutions to each instance $\gamma_j \in \Gamma$.[3] In this work, we are interested in comparing the performance of two algorithms $a_1, a_2 \in \mathcal{A}$ as solvers for a given problem class Γ.[4] We assume that both algorithms of interest can be run on the same subset of instances, and that any run of the algorithm returns some tentative solution, which can be used to assess the quality of that result.

3 Throughout this work we refer to an algorithm as the full structure with specific parameter values, i.e., to a completely defined instantiation of a given algorithmic framework.

4 Γ can either be explicitly known or implicitly defined as a hypothetical set for which some available test instances can be considered a representative sample.

Let $\phi_j = f\left(a_1, a_2, \gamma_j\right) : \mathcal{A}^2 \times \Gamma \mapsto \mathbb{R}$ denote the difference in performance between algorithms a_1, a_2 on instance γ_j, measured according to some indicator of choice; and let $\Phi = \left\{\phi_j : a_1, a_2 \in \mathcal{A}, \gamma_j \in \Gamma\right\}$ denote the set of these *paired differences in performance* between a_1 and a_2 for all instances $\gamma_j \in \Gamma$, with $P\left(\Phi\right)$ denoting the probability density function describing the distribution of values $\phi_j \in \Phi$.

Given these definitions, the *algorithm comparison problem* discussed in this work can be generally defined, given two algorithms $a_1, a_2 \in \mathcal{A}$ and a problem class Γ, as the problem of performing inference about a given parameter θ of the underlying distribution $P\left(\Phi\right)$, based on information obtained by running a_1 and a_2 a certain number of times on a finite sample of instances $\Gamma_S \subset \Gamma$. The parameter of interest, θ, should represent a relevant quantity on which algorithms are to be compared. Common examples of parameters of interest are the mean of $P\left(\Phi\right)$, in which case the comparison problem presented here would result in the test of hypotheses on the paired difference of means (performed using, e.g., a *paired t-test* [54]); or the median, in which case we could use the *Wilcoxon signed-rank test* or the *binomial sign test* [54].

Finally, assume that the result of a given run of algorithm a_i on instance γ_j, denoted x_{ij}, is subject to random variations – e.g., due to a_i being a randomized algorithm, or to randomly defined initial states in a deterministic method – such that $x_{ij} \sim X_{ij}$, where X_{ij} is the underlying random variable associated with the distribution of performance values for the pair $\left(a_i, \gamma_j\right)$.

Notice that these assumptions, which represent the usual case for the majority of experimental comparisons of algorithms, mean that there are two sources of uncertainty that must be considered when trying to address the algorithm comparison problem. First, there is the uncertainty arising from the fact that we are trying to answer questions about a population parameter θ based on a limited sample, which is the classical problem of statistical inference. The second source of variability is the uncertainty associated with the estimation of ϕ_j from a finite number of runs.

These two components of the total variability of the results to be used for comparing two algorithms influence the statistical power of any inferential task to be performed on the value of θ. To control these influences there are two types of sample sizes that need to be considered:

- The number of repeated runs (*repetitions*), i.e., how many times each algorithm a_i needs to be run on each instance γ_j. These sample sizes, which will be denoted n_{ij}, can be used to control the accuracy of estimation of ϕ_j and, to a lesser extent, contribute to the statistical power of the comparison;

- The number of problem instances used in the experiment (*replicates*), also called here the *effective sample size*. This value, which will be denoted $N = |\Gamma_S|$, can be used to more directly set the statistical power of the comparison at a desired level.

In this work we focus on comparisons of mean performance, with simple extensions to the testing of medians. The specifics of this particular case are discussed next.

Comparison of mean performance

When comparing two algorithms in terms of their mean performance over a given problem class of interest, we are generally interested in performing inference on the value of $\theta = \mu_D = E\left[P\left(\Phi\right)\right]$. In this case, the statistical hypotheses to be tested, if we are interested in simply investigating the existence of differences in mean performance between the two algorithms, regardless of their direction, are:

$$\begin{aligned} H_0 &: \quad \mu_D = \mu_0 \\ H_1 &: \quad \mu_D \neq \mu_0; \end{aligned} \tag{3}$$

or, if we are interested in specifically determining whether algorithm a_2 (e.g., a proposed approach) is superior to a_1 (e.g., a state-of-the-art approach) in terms of mean performance over the problem class of interest[5],

$$\begin{aligned} H_0 &: \quad \mu_D \geq \mu_0 \\ H_1 &: \quad \mu_D < \mu_0. \end{aligned} \tag{4}$$

The value of μ_0 in (3) and (4), i.e., the mean of the paired differences of performance under the null hypothesis H_0, is commonly set as $\mu_0 = 0$ when comparing algorithms, reflecting the absence of prior knowledge of differences in performance between the two algorithms compared.

As mentioned earlier in this section, there are two types of sample sizes that need to be considered for comparing algorithms: the number of within-sample repetitions for each algorithm, and the number of instances to be employed. In Chapter 3 we present a methodology for calculating these two sample sizes for the algorithm comparison problem defined in this section. Prior to describing the method, however, it is important to review some relevant statistical concepts that provide the basis for the proposed approach provided in chapter A .

5 The direction of the inequalities in (4) will depend on the type of performance measure used, i.e., on whether *larger = better* or vice versa.

3 PROPOSED METHOD

Humans are allergic to change. They love to say, 'We have always done it this way' I try to fight that. That is why I have a clock on my wall that runs counter clockwise

— Grace Hopper, *The Wit and Wisdom of Grace Hopper*

Given the definitions provided in the preceding sections, we present a methodology for estimating the relevant sample sizes for the algorithm comparison problem described in Section 2.5.1, that is, the comparison of two algorithms in terms of their mean paired differences of performance over instances belonging to a given problem class. More specifically, we describe (i) an algorithmic approach to iteratively sample each algorithm on each problem instance (i.e., *repetitions*) with sample size ratios close to theoretical optimal values, so that a predefined accuracy in the estimation of each ϕ_j is obtained; and (ii) specific formulas for determining the required number of instances (i.e., *replicates*), so that a desired power level can be achieved for a predefined MRES - minimally relevant effect size (see section A.3.1).

As mentioned in the Introduction, it is important to highlight here that the two main results of the proposed methodology, namely the estimation of the number of instances and number of within-instance replicates, are independent: researchers can employ the two calculations separately if desired or required by the specifics of a particular experiment. For instance, it is common for certain application domains to have predefined test sets composed of heterogeneous instances, aimed at testing the behaviour of algorithms on a variety of possible situations. In these cases the researcher may wish to employ the full set of available test instances (assuming computational time is not an issue), but he or she can still employ the proposed methodology to: (i) determine the number of runs for each algorithm on each instance (see Section 3.1 below), and (ii) determine the statistical properties of the experiment in terms of the power to detect differences of a given magnitude (see Section 3.5). In any case, the application of the principles discussed in this work can aid the research to design and perform comparative experiments with increased statistical soundness.

3.1 ESTIMATING THE NUMBER OF REPETITIONS

The proposed strategy to calculate the number of runs of each algorithm a_i on a given instance γ_j, (i.e., the number of repetitions, n_{ij}) consists in iteratively increasing the number of observations of each algorithm until the standard error of $\widehat{\phi}_j$ (the estimate of the difference in performance between the two algorithms for instance γ_j) falls below a given threshold. While the specifics of standard error estimation depend on which statistic is being used to quantify the difference in performance, the accuracy of estimation improves as the sample sizes n_{1j} and n_{2j} are increased. This allows us to define the problem of estimating the number of runs of algorithms a_1, a_2 on a given instance γ_j as that of *finding the smallest total sample size, $n_{1j} + n_{2j}$, such that the standard error of $\widehat{\phi}_j$ falls below a desired accuracy threshold se^*.*

Notice that unlike the usual practice in the experimental comparison of algorithms, the solution for this problem will almost always result in different numbers of runs of a_1 and a_2 on any given instance. This is a consequence of the fact that distinct algorithms will present different variances of performance within any instance, which means that their contributions to the standard error of any estimator used to quantify the paired differences in performance will be unequal. In general, the larger-variance algorithm will need a larger sample size, as will be made clear in this section. Notice, however, that the method presented in this section will work perfectly well if the experimenter forces equal sample sizes (which can be done by a small, trivial modification of Algorithm 1), although the total number of runs in this case may be larger than necessary.

In what follows we provide the derivation of the optimal sample sizes for two specific cases of ϕ_j, namely the *simple* and the *percent* difference between two means. The derivations are performed assuming that the conditions for the Central Limit Theorem (CLT) are met [54], which means that the sampling distributions of the means are approximately normal. An alternative approach, which does not need to comply with this particular set of assumptions (at the cost of increased computational costs) involves the use of resampling strategies such as the Bootstrap [11], which is discussed in Section 3.4.

3.1.1 Using the Simple Difference of Two Means

Assume that we are interested in using the simple difference of mean performance between algorithms a_1, a_2 on each instance as our values of ϕ_j. In this case we define $\phi_j = \mu_{2j} - \mu_{1j}$, for which the sample estimator is given by

$$\widehat{\phi}_j^{(1)} = \widehat{\Delta\mu} = \widehat{\mu}_{2j} - \widehat{\mu}_{1j} = \bar{X}_{2j} - \bar{X}_{1j}, \tag{5}$$

where $\bar{X}_{ij}$ is the sample mean of algorithm a_i on instance γ_j. Let the distribution of performance values of algorithm a_i on instance γ_j be expressed as an (unknown) probability density function with expected value μ_{ij} and variance σ_{ij}^2, i.e.,

$$x_{ij} \sim X_{ij} = \mathcal{P}\left(\mu_{ij}, \sigma_{ij}^2\right).$$

Assuming that the conditions of the Central Limit Theorem hold, we expect $\bar{X}_{ij} \sim \mathcal{N}\left(\mu_{ij}, \sigma_{ij}^2 / n_{ij}\right)$ and, consequently,

$$\widehat{\phi}_j^{(1)} \sim \mathcal{N}\left(\mu_{2j} - \mu_{1j}, \frac{\sigma_{1j}^2}{n_{1j}} + \frac{\sigma_{2j}^2}{n_{2j}}\right). \tag{6}$$

By definition, the standard error of $\widehat{\phi}_j^{(1)}$ is the standard deviation of this sampling distribution of the estimator,

$$se_{\widehat{\phi}_j^{(1)}} = \sqrt{\sigma_{1j}^2 n_{1j}^{-1} + \sigma_{2j}^2 n_{2j}^{-1}}.$$

Given a desired upper limit for the standard error, se^*, the optimal sample sizes for the two algorithms $a1, a2$ on instance γ_j can be obtained by solving the optimization problem defined as

$$\begin{aligned} &\text{Minimize: } f(\mathbf{n}_j) = n_{1j} + n_{2j}, \\ &\text{Subject to: } g(\mathbf{n}_j) = se_{\widehat{\phi}_j^{(1)}} - se^* \leq 0. \end{aligned} \tag{7}$$

This problem can be solved analytically using the Karush-Kuhn-Tucker (KKT) optimality conditions,

$$\begin{aligned} \nabla f(\mathbf{n}_j) + \beta \nabla g(\mathbf{n}_j) &= 0, \\ \beta g(\mathbf{n}_j) &= 0, \\ \beta &\geq 0. \end{aligned} \tag{8}$$

The solution of (8) for the objective and constraint functions in (7) yields the optimal ratio of sample sizes,

$$r_{opt} = \frac{n_{1j}}{n_{2j}} = \frac{\sigma_{1j}}{\sigma_{2j}}, \tag{9}$$

which means that algorithms a_1 and a_2 must be sampled on instance γ_j in direct proportion to the standard deviations of their performances on that instance. The result in (9) is known in the statistical literature [51] as the *optimal allocation of resources* for the estimation of confidence intervals on the simple difference of two means.

Since the populational variances $\sigma^2_{1j}, \sigma^2_{2j}$ are usually unknown, their values need to be estimated from the data. This results in the sample standard error,

$$\widehat{se}_{\widehat{\phi}^{(1)}_j} = \sqrt{s^2_{1j}n^{-1}_{1j} + s^2_{2j}n^{-1}_{2j}}. \tag{10}$$

A good approximation of the optimal ratio of sample sizes can be similarly obtained by replacing σ_{ij} by s_{ij} in (9). This requires that an initial sample size n_0 be obtained for each algorithm, to calculate initial estimates of s_{1j}, s_{2j}[1], suggesting the iterative procedure described in Algorithm 1, where $\texttt{Sample}(a_i, \gamma_j, n \text{ times})$ means to obtain n observations of algorithm a_i on instance γ_j. To prevent an explosion of the number of repetitions in the case of poorly specified threshold values se^* or particularly high-variance algorithms, a maximum budget n_{max} can also be defined for the sampling on a given problem instance.

Algorithm 1: Sample algorithms on one instance.

```
Data: Instance $\gamma_j$; Algorithms $a_1, a_2$; accuracy threshold $se^*$; initial sample size $n_0$;
      maximum sample size $n_{max}$
Result: $\mathbf{x}_{1j}$, $\mathbf{x}_{2j}$
$\mathbf{x}_{1j} \leftarrow$ Sample$(a_1, \gamma_j, n_0$ times$)$ $\mathbf{x}_{2j} \leftarrow$ Sample$(a_2, \gamma_j, n_0$ times$)$ $n_{1j} \leftarrow n_0$; $n_{2j} \leftarrow n_0$;
 Calculate $\widehat{se}$ using (10) or (13) or Algorithm 4; while $(\widehat{se} > se^*)$ & $(n_{1j} + n_{2j} < n_{max})$ do
    Calculate $r_{opt}$ using (9) or (15); if $(n_{1j}/n_{2j} < r_{opt})$ then
       $x \leftarrow$ Sample$(a_1, \gamma_j, 1$ time$)$; $\mathbf{x}_{1j} \leftarrow [\mathbf{x}_{1j}, x]$;
    else
       $x \leftarrow$ Sample$(a_2, \gamma_j, 1$ time$)$; $\mathbf{x}_{2j} \leftarrow [\mathbf{x}_{2j}, x]$;
    Calculate $\widehat{se}$ using (10) or (13) or Algorithm 4;
```

After performing the procedure shown in Algorithm 1, the estimate $\widehat{\phi}_j$ can be calculated using the vectors of observation $\mathbf{x}_{1j}$ and $\mathbf{x}_{2j}$ into (5) or (11), depending on the type of difference used.

1 The definition of an initial value of n_0 also helps increasing the probability that the sampling distributions of the means will be approximately normal.

3.1.2 Using the Percent Difference of Two Means

While the approach of defining ϕ_j as the simple difference between the means of algorithms a_1, a_2 on a given instance γ_j is certainly useful, it may be subject to some difficulties. In particular, defining a single precision threshold se^* for problem classes containing instances with vastly different scales can be problematic and lead to wasteful sampling. In these cases, it is generally more practical and more intuitive to define the differences in performance ϕ_j as the *percent mean gains* of algorithm a_2 over a_1. In this case we define[2] $\phi_j = \left(\mu_{2j} - \mu_{1j}\right) / \mu_{1j}$, for which the sample estimator is

$$\widehat{\phi}_j^{(2)} = \widehat{\Delta\mu}_{(\%)} = \frac{\bar{X}_{2j} - \bar{X}_{1j}}{\bar{X}_{1j}} = \frac{\widehat{\phi}_j^{(1)}}{\bar{X}_{1j}} \tag{11}$$

For this definition to be used we need to consider an additional assumption, namely that $P(\bar{X}_{1j} \leq 0) \to 0$ (which is guaranteed, for instance, when objective function values are always strictly positive, which is common in several problem classes).[3] The distribution of $\phi_j^{(1)}$ is given in (6), which means that under our working assumptions $\widehat{\phi}_j^{(2)}$ is distributed as the ratio of two independent normal variables.[4]

A commonly used estimator of the standard error of $\widehat{\phi}_j^{(2)}$ is based on confidence interval derivations by Fieller [25]. Considering the assumption that $P(\bar{X}_{1j} \leq 0) \to 0$, a simplified form of Fieller's estimator can be used [27], which provides good coverage properties. Under balanced sampling, i.e., $n_{1j} = n_{2j} = n_j$, the standard error is given as

$$\widehat{se}_{\widehat{\phi}_j^{(2)}} = \left|\widehat{\phi}_j^{(2)}\right| \left[\frac{s_{1j}^2/n_j}{\bar{x}_{1j}^2} + \frac{\left(s_{1j}^2/n_j + s_{2j}^2/n_j\right)}{\left(\widehat{\phi}_j^{(1)}\right)^2} + \frac{2}{n_j}\frac{cov\left(\mathbf{x}_{1j}, \left(\mathbf{x}_{2j} - \mathbf{x}_{1j}\right)\right)}{\widehat{\phi}_j^{(1)}\bar{x}_{1j}}\right]^{1/2}, \tag{12}$$

where $\mathbf{x}_{ij} \in \mathbb{R}^{n_j}$ represents the vector of observations of algorithm a_i on instance γ_j; and $cov\left(\cdot,\cdot\right)$ is the sample covariance of two vectors.

Under the assumption of within-instance independence, i.e., that X_{1j} and X_{2j} are independent, the expected value of covariance will be close to zero, allowing us to disregard the covariance term in (12). This offers two advantages: first, it simplifies calculations of the standard error, particularly for larger sample sizes. Second, and more importantly, it allows us to consider unbalanced sampling, as we did for the case of simple differences, which can lead

2 Considering a comparison where larger is better.

3 If this assumption cannot be guaranteed, the use of percent differences is not advisable, and the researcher should instead perform comparisons using the simple differences.

4 The independence between $\phi_j^{(1)}$ and $\bar{X}_{1j}$ is guaranteed as long as X_{1j} and X_{2j} are independent.

to gains in efficiency. Removing the covariance term, replacing the n_j dividing each sample standard deviation by the corresponding n_{ij} and simplifying (12) results in

$$\widehat{se}_{\widehat{\phi}_j^{(2)}} = \left|\widehat{\phi}_j^{(2)}\right| \sqrt{c_1 n_{1j}^{-1} + c_2 n_{2j}^{-1}}, \tag{13}$$

with

$$\begin{aligned} c_1 &= s_{1j}^2 \left[\left(\widehat{\phi}_j^{(1)}\right)^{-2} + \left(\bar{x}_{1j}\right)^{-2}\right]; \\ c_2 &= s_{2j}^2 \left(\widehat{\phi}_j^{(1)}\right)^{-2}. \end{aligned} \tag{14}$$

The problem of calculating the smallest total sample size required to achieve a desired accuracy is equivalent to the one stated in (7) (substituting $\widehat{se}_{\widehat{\phi}_j^{(1)}}$ by $\widehat{se}_{\widehat{\phi}_j^{(2)}}$ in the constraint function) and can be solved in a similar manner to yield the optimal ratio of sample sizes in the case of percent differences,

$$r_{opt} = \frac{n_{1j}}{n_{2j}} = \sqrt{\frac{c_1}{c_2}} = \frac{s_{1j}}{s_{2j}} \sqrt{1 + \left(\widehat{\phi}_j^{(2)}\right)^2} \tag{15}$$

The expressions in (13) and (15) can be used directly into Algorithm 1, so that the adequate sample sizes for obtaining an estimate $\widehat{\phi}_j^{(2)}$ with a standard error controlled at a given threshold se^* can be iteratively generated.

3.2 ESTIMATING THE NUMBER OF INSTANCES

As described in Section 2.5.1, the algorithm comparison problem treated in this work naturally induces a paired design [54], which allows instance effects to be modeled out. Here we discuss the definition of the number of instances required for the experiment to obtain the desired statistical properties, namely a power of at least $\pi^* = 1 - \beta^*$ to detect differences equal to or greater than a minimally relevant effect size d^* at a predefined significance level α.

Before we proceed it is important to highlight that, since the hypotheses of interest concern the expected value of a distribution defined over the set of paired differences in performance, Φ, the *independent observations* to be used in the test of this hypotheses are the individual values ϕ_j (or, more accurately, their estimates $\widehat{\phi}_j$), and not the individual runs of the algorithms on each instance. Failure to realize this point leads to pseudoreplication [37, 43], i.e., the calculation of test statistics under falsely inflated degrees-of-freedom, with a consequent loss of control over the statistical error rates of the tests.

Under the assumption that the sampling distributions of means for the paired differences are approximately normal, i.e., that

$$\frac{1}{N}\sum_{\gamma_j\in\Gamma_S}\widehat{\phi}_j=\widehat{\mu}_D\sim\mathcal{N}\left(\mu_D,\sigma_\Phi^2/N\right), \tag{16}$$

where $N=|\Gamma_S|$ is the number of instances used in the experiment, the uniformly most powerful unbiased test for hypotheses of the forms (3)–(4) is the *paired t-test* [54]. The test statistic for this procedure is calculated as:

$$t_0=\frac{\widehat{\mu}_D-\mu_0}{\widehat{\sigma}_\Phi/\sqrt{N}}=\frac{\widehat{\delta}}{\widehat{\sigma}_\Phi}\sqrt{N}=\widehat{d}\sqrt{N}, \tag{17}$$

where $\widehat{\sigma}_\Phi$ is the sample estimate of the total standard deviation σ_Φ, and $\widehat{d}$ is the sample estimate of Cohen's d coefficient (36). Under H_0 this test statistic is distributed according to Student's t distribution with $N-1$ degrees of freedom [54], leading to the criterion for rejecting the null hypothesis at the $1-\alpha$ confidence level being, for hypotheses of form (3)

$$|t_0|\geq t_{1-\alpha/2}^{(N-1)}; \tag{18}$$

or, for (4),

$$t_0\geq t_{1-\alpha}^{(N-1)}, \tag{19}$$

where $t_q^{(df)}$ denotes the q-th quantile of Student's t distribution with df degrees-of-freedom [54].

Under the alternative hypothesis H_1, t_0 is distributed according to a *noncentral t distribution* [51] with noncentrality parameter

$$ncp=(\mu_D-\mu_0)\sqrt{N}/\widehat{\sigma}_\Phi=\delta\sqrt{N}/\widehat{\sigma}_\Phi=d\sqrt{N}$$

Assuming a MRES $d^*=|\delta^*|/\sigma_\Phi$, the power of the test is given by the integral of the noncentral t distribution with $ncp^*=d^*\sqrt{N}$ over the values of t_0 for which H_0 is rejected. For instance, for the case (3) the rejection region is given in (18), and the power can be calculated as

$$\pi^*=1-\beta^*=1-\int_{t=t_{\alpha/2}^{(N-1)}}^{t_{1-\alpha/2}^{(N-1)}}\left[t_{|ncp^*|}^{(N-1)}\right]dt. \tag{20}$$

The sample size for this test can then be calculated as the smallest integer such that π^* is equal to or larger than a desired power. This leads to the formulas for the required sample size for the case of the paired t-test [51] for the two-sided alternative hypothesis (3),

$$N^*=\min\ N\left|t_{1-\alpha/2}^{(N-1)}\leq t_{\beta^*;|ncp^*|}^{(N-1)}\right., \tag{21}$$

or, for the directional alternative (4),

$$N^* = \min\ N \left| t_{1-\alpha}^{(N-1)} \leq t_{\beta^*;|ncp^*|}^{(N-1)} \right. . \tag{22}$$

While there are no analytical solutions for (21)–(22), the calculation of these sample sizes can be easily done iteratively [51], and is available in most statistical packages, e.g., R. Algorithm 2 summarizes the full procedure for calculating the relevant sample sizes and running the experimental comparison of mean performance of two algorithms for a given problem class. As mentioned earlier, the researcher can either adopt the full procedure or, if desired, opt for using only part of the methodology (e.g., employ a predefined number of instances, use a predefined number of repetitions, force balanced runs on each instance, etc.).

Algorithm 2: Full procedure for the comparison of algorithms

```
Data: Set of available instances $\Gamma_S$; algorithms $a_1, a_2$; accuracy threshold $se^*$; initial
      sample size $n_0$; maximum sample size $n_{max}$; desired significance level $\alpha$, type-II
      error rate $\beta^*$, and MRES $d^*$.
Result: Test results; power profile (if needed)
Calculate $N^*$ /* Using (21) or (22) */
if $N^* > |\Gamma_S|$ then
    Investigate power, change parameters if needed. /* Sec. 3.5 */
$\mathbf{x} \leftarrow [\,]$ for $\min(N^*, |\Gamma_S|)$ times do
    Sample (without replacement) instance $\gamma_j \in \Gamma_S$ Sample $a_1, a_2$ on $\gamma_j$ /* Algorithm 1
      */
    Calculate $\widehat{\phi}_j$ /* Using (5) or (11) */
    $\mathbf{x} \leftarrow [\mathbf{x}, \widehat{\phi}_j]$
Test of hypotheses using $\mathbf{x}$ as the test sample Verify test assumptions /* Secs.
    3.3-3.4 */
```

Finally, as mentioned in Section 2.5.1, there are two sources of variability that affect the total variance σ_Φ^2, namely the across-instances variance σ_ϕ^2, which represents the variance of the values of paired differences in performance if all ϕ_j were precisely known; and the within-instances variance $\sigma_\epsilon^2 = se_{\widehat{\phi}_j}^2$, which quantifies the "measurement error" on the values of ϕ_j. Considering this, the standardized effect size used in the preceding discussion can be expressed as

$$d = \frac{\delta}{\sigma_\Phi} = \frac{\delta}{\sqrt{\sigma_\phi^2 + \sigma_\epsilon^2}}. \tag{23}$$

While the experimenter can do little to change the across-instances variance, it is possible to reduce the standard error of estimation, as presented in Section 3.1. This composition of the total variance can be helpful in defining se^* when calculating of the number of repetitions. Some guidelines are provided in Section 3.6.

3.3 INDEPENDENCE AND NORMALITY

The techniques presented so far have been based on two explicit assumptions: independence, i.e., the assumption that observations used for calculating the statistics of interest do not present any unmodeled dependencies, or that one observation does not influence another [54, 66]; and normality of the sampling distribution of the means.

In the case of this work, the assumption of independence can be guaranteed by design. In Algorithm 1, the samples generated for the two algorithms on any given instance are produced without one observation influencing the value of any other – e.g., by the usual (and rather obvious) practice of using different random seeds for different runs of randomized algorithms; or of using distinct, preferably randomly distributed initial points for deterministic methods. As for the paired test and sample size calculations, the assumption of independence can also be guaranteed by design. By using the values of $\widehat{\phi}_j$ as the individual observations we avoid the most common error in this kind of experiment, namely that of pseudoreplication [37, 43], i.e., the use of repeated measurements x_{ij} as if they were independent replicates. Ensuring independent algorithmic runs, as mentioned previously, also helps guarantee this assumption.

The other assumption, i.e., that of normality of the sampling distribution of the means, cannot be so easily guaranteed by design. It can, however, be verified *a posteriori* without much effort. A first test of this assumption relies on the fact that, if the distribution of the data is normal, then the sampling distribution of the means will also be normal, regardless of the sample size. This suggests that a first test of normality can be performed on the data itself – i.e., on the sets of observations x_{1jk} and x_{2jk} used to estimate ϕ_j; and on the set of estimates $\widehat{\phi}_j$ used for testing the hypotheses of interest. Common statistical tests of normality include the Kolmogorov-Smirnov or Shapiro-Wilk tests [66], although in most cases visual inspection using a normal Q-Q plot is considered sufficient [54]. If the data is found not to be significantly deviant from normality then the methods presented in this section can be considered accurate.

If the data itself deviates significantly from normality, an approximate test can be performed on the estimated sampling distribution of the means instead, e.g., using bootstrap [18]. A quick (albeit computationally intensive) procedure for assessing normality of the sampling distribution of the means is to generate a vector $\mathbf{y}_B$ of resampling estimates of the mean using a bootstrap procedure and then visually inspecting this vector using a normal Q-Q plot.[5] This assessment strategy is summarized in Algorithm 3, where `SampleWithReplacement`$(\mathbf{y}, n$ times$)$ means to build a vector of n observations sampled (with replacement) from $\mathbf{y}$. If this estimated sampling distribution of the mean does not deviate from normality, then the assumption can be considered satisfied for the methods presented in this section.

Algorithm 3: Bootstrapping the sampling distribution of the mean

```
Data: Sample vector $\mathbf{y}$; number of bootstrap resamplings $R$.
Result: $\bar{\mathbf{y}}_B$
$\bar{\mathbf{y}}_B \leftarrow [\,]$ /* Initialize empty vector                                   */
; $n \leftarrow dim(\mathbf{y})$ /* Vector length                                          */
; for (R times) do
    $\mathbf{y}_T \leftarrow$ SampleWithReplacement($\mathbf{y}, n$ times); $\bar{y}_T \leftarrow mean(\mathbf{y}_T)$; $\bar{\mathbf{y}}_B \leftarrow [\bar{\mathbf{y}}_B, \bar{y}_T]$
        /* Append $\bar{y}_T$ to $\bar{\mathbf{y}}_B$                                       */
```

Finally, if the assumption of normality is violated (or expected to be, in the design phase of the experiment), one must employ nonparametric methods instead. A brief discussion of these techniques is provided next.

3.4 NONPARAMETRIC ALTERNATIVES

If the assumption of normality of the sampling distribution of the means cannot be guaranteed,[6] different procedures should be employed. Some possibilities for estimating ϕ_j and for testing hypotheses regarding the expected performance difference between two algorithms are presented in this section.

5 Using inferential tests on $\mathbf{y}_B$ is not good practice, as the number of resamples can be made arbitrarily large, which would artificially inflate the degrees-of-freedom of any such test.

6 Notice that it is relatively common for the normality assumption to be violated in the original data, but valid under transformations such as *log* or *square root*. The topic of data transformations is, however, outside the scope of this thesis.

3.4.1 Nonparametric estimation of se_{ϕ_j} and of the number of repetitions

When the assumptions regarding the sampling distribution of $\widehat{\phi}_j$ are not true, the estimates of the standard error calculated in Section 3.1 may be incorrect (particularly for the case of percent differences). If that is the case, a bootstrap approach can be used to estimate $se_{\widehat{\phi}_j}$ and, consequently, the required number of repetitions.

To obtain a bootstrap estimation of $se_{\widehat{\phi}_j}$, recall the definition of standard error as the standard deviation of the sampling distribution of a given estimator. A bootstrap estimator of $se_{\widehat{\phi}_j}$ can be calculated using the routine shown in Algorithm 4, and the value returned can then be used directly into Algorithm 1.

Algorithm 4: Bootstrap estimation of $se_{\widehat{\phi}_j}$

```
Data: Sample vectors $\mathbf{x}_{1j}, \mathbf{x}_{2j}$; number of bootstrap runs $R$.
Result: sample standard deviation of $\widehat{\boldsymbol{\phi}}_j$
$\widehat{\boldsymbol{\phi}}_j \leftarrow [\,]$;
$n_{1j} \leftarrow dim(\mathbf{x}_{1j})$;
$n_{2j} \leftarrow dim(\mathbf{x}_{2j})$;
for ($R$ times) do
    $\mathbf{x}_1^b \leftarrow$ SampleWithReplacement($\mathbf{x}_{1j}, n_{1j}$ times);
    $\mathbf{x}_2^b \leftarrow$ SampleWithReplacement($\mathbf{x}_{2j}, n_{2j}$ times);
    Calculate $\widehat{\phi}_{j,r}$ using (5) or (11);
    $\widehat{\boldsymbol{\phi}}_j \leftarrow [\widehat{\boldsymbol{\phi}}_j, \widehat{\phi}_{j,r}]$;
```

Notice that, unlike in the parametric approach, the optimal ratio r_{opt} used in Algorithm 1 to determine which algorithm should be sampled may not be optimal in the theoretical sense when using a bootstrap estimate of $se_{\widehat{\phi}_j}$. Nonetheless, there are two arguments that can be advanced for using it in this case: first, it will always result in more intensive sampling of the algorithm presenting the greatest variance, which makes sense from the perspective of reducing the standard error of the estimates of ϕ_j. Second, since the sampling distribution of the means will become progressively closer to a normal distribution as the sample sizes are increased, the estimation of r_{opt} will become increasingly better as more observations are collected, and thus the sample sizes yielded by Algorithm 1 should approach optimality as the sampling progresses.

Finally, it is important to highlight that the bootstrap procedure tends to be considerably more computationally intensive than the parametric one, due to the resampling procedures involved in its calculation. This difference, however, becomes less important when the run times of a_1, a_2 are longer, needing, e.g., seconds or minutes to complete.

3.4.2 Nonparametric tests of hypotheses

Common alternatives for the paired t-test include *Wilcoxon's signed-ranks test*, which assumes independence and symmetry about the median[7] of $P(\Phi)$ [66]; and the *binomial sign test*, which requires only the assumption of independence [66], at the cost of reduced power. Both can be used to test hypotheses regarding the *median* of $P(\Phi)$ instead of the mean, which is another way to quantify the expected differences between two algorithms.

The determination of the number of instances for these cases can be done using an argument based on the *asymptotic relative efficiency* (ARE) of these tests relative to the paired t-test. The ARE can be defined [54] as "*the limiting ratio of the sample sizes necessary to obtain identical error probabilities for the two procedures.*". In the specific case of the Wilcoxon test, we have that [54] "*For normal populations, the ARE of the Wilcoxon signed-rank test relative to the t-test is approximately 0.95; For non-normal populations, the ARE is at least 0.86, and in many cases it will exceed unity.*". As for the binomial sign test, the ARE is 0.637 [66], showing its more conservative characteristic.

Under these considerations, a reasonable rule-of-thumb is to calculate the required number of instances using the formulas for the paired t-test, and then dividing the value of N^* by the ARE of the test under normality:

$$\begin{aligned} N^*_{wilc} &= N^*/0.86 \approxeq 1.16N^* \\ N^*_{sign} &= N^*/0.637 \approxeq 1.57N^* \end{aligned} \tag{24}$$

Notice that i) these are conservative estimates, reflecting (particularly in the case of Wilcoxon's test) an expected worst-case scenario, which means that the actual power can be larger than the one used for the calculations; and ii) the binomial sign test requires over 50% more instances to achieve the same power under this supposed worst-case scenario, which may be unreasonable in many situations. However, if $P(\Phi)$ is severely skewed, this may be the only

[7] Although it is very common in the literature on the experimental comparison of algorithms to ignore the fact that Wilcoxon's signed-ranks test works under the assumption of symmetry.

test for which the assumptions can be maintained (i.e., for which nominal error rates can be reasonably expected to hold), and as such it remains as an interesting last resource.[8]

3.5 THE CASE OF PREDEFINED N

The second part of the proposed methodology, described in Sections 3.2 and 3.4.2, is concerned with estimating the smallest number of instances required for achieving predefined statistical properties for a given experiment. This estimation can be very useful in several distinct situations, e.g., when designing test sets for specific problem classes, or when performing experiments on algorithms for computationally expensive optimization problems [39]. As mentioned earlier, however, there are cases in which it may not be possible to arbitrarily choose the sample size for a given experiment. Common examples include situations when only a limited number of instances is available, or when a predefined test set needs to be employed, as is often the case in standardized comparison experiments [31].

Even if that is the case, however, it is still possible to employ the principles discussed in the preceding sections to obtain a better perspective of the statistical properties of the experiment. For instance, even for predefined N, the proposed methodology can still be used to determine the number of runs of each algorithm on each instance, so as to guarantee a desired standard error for the paired differences in each instance. Moreover, the sample size calculations provided in Section 3.2 can be easily adapted to maintain a fixed N and estimate instead other relevant properties, e.g., on the expected statistical power for a given MRES. For instance, by keeping α and N fixed in (20), one can iterate over different values of $ncp = d/\sqrt{N}$ and obtain a power curve for a fixed-sample experiment, prior to actually collecting the data. Figure 3.1 provides an example of this kind of power curve, which can be quite useful for researchers interested in evaluating which differences between algorithms could the experiment be reasonably expected to detect. Similar curves can be constructed for other pairs of power-related variables, e.g., maintaining a fixed power and iterating over N to obtain a curve of effect sizes d for which that power is expected as a function of sample size. Power curves can also be useful for evaluating existing test sets, for example, to answer questions regarding the sensitivity of experiments performed using existing test sets.

8 There are other ways to calculate the sample size for the binomial sign test that are less conservative, but for the sake of brevity this will not be discussed here.

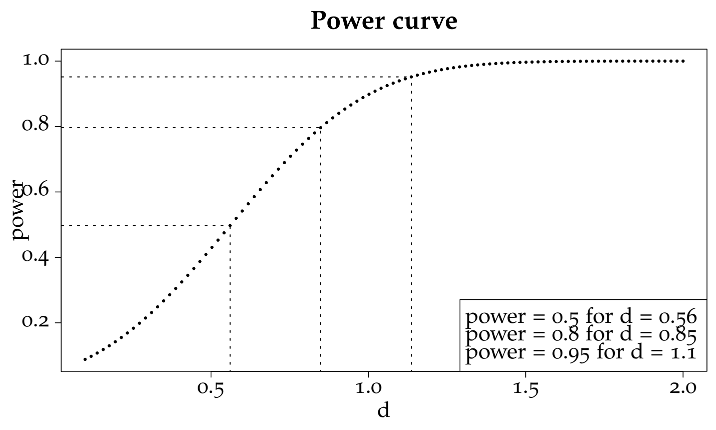

Figure 3.1: Example of power curve that can be derived in cases with a predefined number of instances.

3.6 DEFINING REASONABLE EXPERIMENTAL PARAMETERS

Finally, it is important to discuss the choice of reasonable values for the experimental parameters. In terms of the proposed methodology, the required parameters are shown at the beginning of Algorithm 2. The set of available instances Γ_S and the algorithms to be compared, a_1, a_2, are relatively straightforward - Γ_S is usually a list of available instances, which may or may not be exhausted in the experiment, and a_1, a_2 are the algorithms to be compared.

The definition of the remaining user-defined parameters for the experimental protocol – namely se^*, n_0 and n_{max} for calculating the number of repetitions, and α, β^* and d^* for the number of replicates – is a little more subtle. Starting with the statistical error rates, α and β^* should ideally be defined based on the consequences of the errors they control - i.e., the consequences of falsely detecting a nonexistent difference, or of failing to detect an existing one. However, defining these consequences can be very challenging even in experiments with more easily quantifiable consequences, and in practice "standard" values are often used – 0.05 or 0.01 for α, and 0.2 or 0.15 for β^*. It is important to recall that (i) there is nothing inherently special about these values, they are simply conventions that can, and often should, be challenged; and (ii) there is a tradeoff between the error rates and the sample size, so that the lower these values, the larger the number of instances needed to control both errors at their nominal values for a given MRES.

Determining a good value for the MRES is also heavily experiment-dependent, since a small difference in one context could be considered substantial in another. In our discussions we have been using the standardized effect size d for the power calculations, in which case the MRES, d^*, should be selected based on units of standard deviations - e.g., a $d^* = 0.5$

would mean that we are interested in detecting differences equal to or larger than one half standard deviation. While some fields possess somewhat standard target values for "small", "medium" and "large" effects (see, e.g., the discussion by Sawilowsky [63]), researchers should be aware that specific features of different application areas can and should take precedence over application-agnostic predefined values.

When ϕ_j is defined as the percent differences (Sec. 3.1.2), it may be more intuitive to use the simple effect size, δ, instead of the standardized one. This would allow statements such as "we are interested in detecting mean performance gains of more than 5%", which tend to be more straightforward. In this case, however, a reasonable upper bound for the total standard deviation – the denominator of the r.h.s. of (23) – must be provided by the user, which may be obtained using either a pilot study, estimated from published results, or defined using previous knowledge about the algorithms tested.

Regarding the experimental parameters necessary for estimating the number of runs on each instance, n_0 should ideally be set based on the expected shape of the distribution of observations of algorithm performance – bell-shaped distributions can use lower n_0 (values as low as 3 or 4 are sufficient for the sampling distribution of means to converge to a Gaussian shape in these cases), other symmetric distributions can use intermediate values (e.g., 10), and more strongly skewed distributions should use larger values ($n_0 = 20$ or 30). If the distribution is severely skewed, it is often more practical to work on log-transformed data, which tends to bring the distribution to a more well-behaved shape [17]. The value of n_{max} should be selected based on the available computational budget for the experiment, but knowing that lower values will result in sample sizes that may fail to control the within-instances error $se_{\widehat{\phi}_j}$ at the predefined level se^*, which can result in reduced overall power for the experiment.

The definition of the measurement error threshold, se^*, should be performed in such a way that this component of the total standard deviation does not dominate the power calculations – in other words, the value of se^* should be much smaller than the expected across-instances variance – e.g., $(se^*)^2 \leq 0.1\sigma_\Phi^2$.

Finally, it is important to remember that even if the number of available instances is much larger than the calculated N^* and the researchers desire (or are required) to employ all in the comparison, the methodology presented in this section can still provide precious information - both for the determination of the number of runs on each instance and, critically, for defining a MRES prior to the experiment, so that the results obtained are interpreted under the light of practical relevance, and not only statistical significance.

4 METHOD EVALUATION

The way to do research is to attack the facts at the point of greatest astonishment.

Celia Green

When testing a methodology to compare algorithms, it is important to use functions whose behaviour is known. This allows the researcher to compare the results obtained with the expected behaviour, which enables the validation of the proposed method. In this chapter we present results using a simulation model in which we can control several aspects of the algorithm comparison problem: true effect sizes, residual variances, distribution of errors, etc. This model is used to assess the nominal properties of the proposed method, namely the estimate of standard error, location parameter and effect size, which are part of the sample size calculation.

4.1 SIMULATION MODEL

The model represented by equation (25) simulates executions of a set of algorithms θ_i, $i = 1, \ldots, m$, on a sample of instances $\gamma_j \in \Gamma$, $j = 1, \ldots, N$, where Γ represents a (supposed) problem class of interest:

$$y_{ijk} = \mu_{i;\Gamma} + \tau_{ij} + \epsilon_{ijk} \tag{25}$$

where y_{ijk} is a (simulated) performance value obtained by a hypothetical algorithm θ_i on an equally hypothetical instance γ_j at a given execution k; $\mu_{i;\Gamma}$ represents the expected value of the performance of θ_i in the family of instances Γ, i.e. the grand mean. The coefficient $\tau_{ij} = \mu_{ij} - \mu_{i,\Gamma}$ is the deviation between the reference performance of θ_i on the γ_j instance (μ_{ij}) and the grand mean. And $\epsilon_{ijk} = y_{ijk} - \mu_{ij}$ represents the variation of the k^{th} execution from the reference value of the algorithm on the instance.

The value of $\mu_{i;\Gamma}$ can be arbitrarily set, and differences in $\mu_{i;\Gamma}$ can be seen as simulating effect sizes in an experiment, i.e., differences in the mean performance of algorithms on the problem class. To simulate the sampling of problem instances from a given problem class, the values of the coefficients τ_{ij} and ϵ_{ijk} are sampled from a probability distribution, e.g., a Gaussian or Exponential distribution. In the initial experiments of this work, $\tau_i j$ are sampled using:

$$\tau_{ij} \sim \mathcal{N}(0, \sigma^2_{AI}) \tag{26}$$

where σ^2_{AI} represents the *across instance variance*, which models the variability across instances of the problem class - indirectly, it can be seen as a way to measure the heterogeneity of instance "difficulties" in the problem class.

The coefficient ϵ_{ijk} is obtained using:

$$\epsilon_{ijk} \sim \mathcal{N}(0, \sigma^2_{WI}) \tag{27}$$

where σ^2_{WI} is the *within instance variance*, which models the variability of the algorithm performance across repeated runs on the same instance.

This model simulates outputs of algorithms whose expected performances and variations are known because they are defined by the user. For this reason, it is suitable to verify the consistency of the proposed method, which must present an average performance of the algorithm, or algorithm difference, for each instance within an error. With such average and variations predefined, it is possible to check if the values obtained by the proposed method are in agreement with those defined previously.

4.2 TEST SETTINGS

While the proposed method is not stochastic, its entries - namely the output of an algorithm on an instance drawn from a problem class - are. Thus, the first test of the proposed method consists in validating if the output is compatible with the input, given known parameters used to generate the random inputs: τ_{ij}, which is a fixed value for each instance, is sampled from $\tau_{ij} \sim \mathcal{N}(0, 9)$; and $\sigma_{WI,j}$ is also predefined for each instance and is sampled from a normal variable with zero mean and standard deviation sampled as $\sigma_{WI,j} \sim \mathcal{U}(1, 5)$.

The value for parameter $\mu_{i,\Gamma}$, which can be arbitrarily chosen, was set as 10, 9.9 and 15 for simulated algorithms 1, 2 and 3 respectively. Notice that the values chosen for these simulation parameters make it possible to have simulated instances for which the paired difference in mean performance is negative or very close to zero, which would violate the stated assumptions of the method used to derive standard errors for the percent differences (Section 3.1.2). This is intentional, so that we can observe what effects can be expected when such violations are present, and alert future users of this methodology accordingly.

Once the inputs are properly defined, it is possible to execute the method and verify if its outputs correspond to the values chosen for the inputs. First, it was tested if the mean performance of the algorithms on each instance, obtained from the estimated quantity of sampled observations, was appropriate. This was done by running the proposed method with 100 (simulated) instances, and checking whether the estimate obtained by the proposed method for the performance of each "algorithm" on each "instance" did match the nominal value $\mu_{i,\Gamma} + \tau_{ij}$. In the case of estimating the differences, it was evaluated if the obtained differences were consistent with $(\mu_{2j} - \mu_{1j})/\mu_{1j}$ for the percent differences, or $\mu_{2j} - \mu_{1j}$ for the absolute differences, the first considering a threshold for the standard error of 0.1, which means a difference of 10% between the algorithms, and the second of 0.35, a difference of 35%.

To test if the number of replications estimated by the proposed method is appropriate, the method was executed 1000 times, every time randomly sampling N simulated instances. The differences in algorithm performances discovered by the proposed method should be consistent with the expected value, with an error rate of approximately α.

4.2.1 Validation of the estimated number of repetitions

The test to verify whether the estimate of the number of repetitions is appropriate considered two situations: one where the pair of algorithms had very similar performances, being *algorithm 1* with $\mu_{\theta_1;\Gamma} = 10$ and *algorithm 2* with $\mu_{\theta_2;\Gamma} = 9.9$; and one with a pair of algorithms with more clearly distinct performances, being *algorithm 1* with $\mu_{\theta_1;\Gamma} = 10$ and *algorithm 3* with $\mu_{\theta_3;\Gamma} = 15$;

Figure 4.1 shows the confidence intervals for the percent differences between algorithms 1 and 2, generated using the proposed sampling procedure, for 100 simulated instances. The intervals were centered, so that those that intercept the zero line represent the cases in which the interval was able to capture the correct parameter value. In the case of algorithms 1 and 2, the real percent difference in performance for each instance was calculated as $(9.9 + \tau_{2j} - 10 - \tau_{1j})/(10 + \tau_{1j})$, with the values of τ_{ij} sampled according to the description in Section 4.2.

Figure 4.2 shows the result of the same experiment for algorithms 1 and 3, whose real percent differences were calculated by $(15 + \tau_{3j} - 10 - \tau_{1j})/(10 + \tau_{1j})$. When testing this experiment 500 times the percentage of errors was in average 4.9% for the algorithms 1 and 2 and 10.7% for algorithms 1 and 3. This decrease in the effective confidence level of the intervals for larger percent differences reflects a known problem with intervals derived from the approximation based on Fieller's theorem [47], and alternative strategies for a more precise calculation of standard errors for percent differences [47] are under investigation.

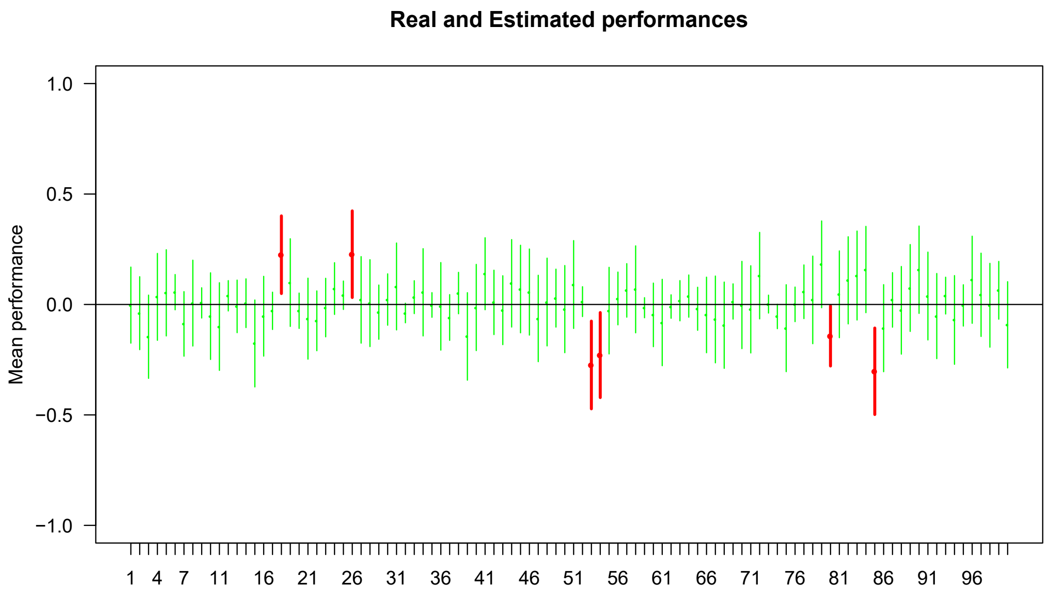

Figure 4.1: Mean performances of 100 instances on percent difference between *algorithm 1* and *algorithm 2*. The instances for which the real value of μ_{ij} falls outside the confidence interval are marked in red and counted as errors

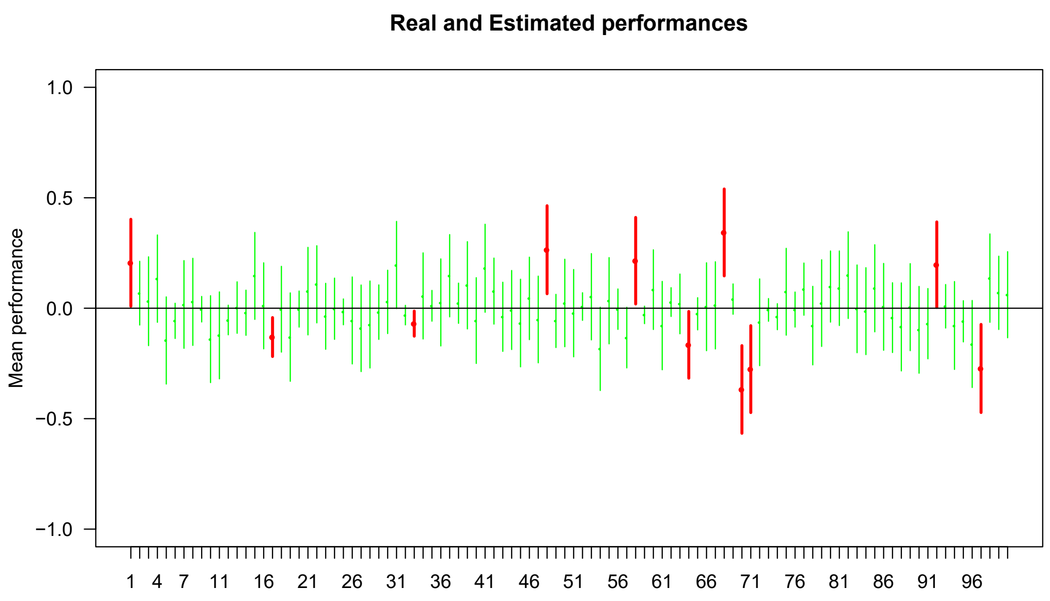

Figure 4.2: Mean performances of 100 instances on percent difference between *algorithm 1* and *algorithm 3*. The instances for which the real value of μ_{ij} falls outside the confidence interval are marked in red and counted as errors

Figure 4.3 shows that there are some instances that require a large number of repetitions. These cases represent situations in which the value of μ_{1j} approached zero, which, as stated earlier, violates the assumptions of the method used to derive intervals for the percent differences. In these cases, the most advisable course of action for the user would be to work with absolute differences instead of percent ones.

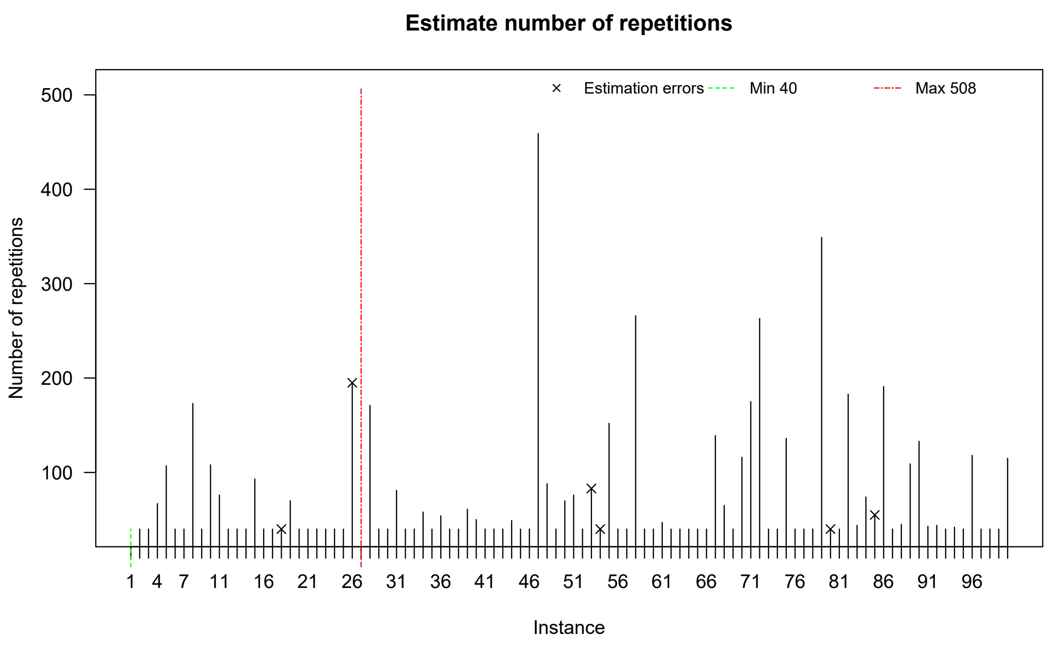

Figure 4.3: Number of repetitions of 100 instances for percent difference between *algorithm 1* and *algorithm 2*. The x markers represent the instances with wrongly estimated μ_{ij} (see Figure 4.1).

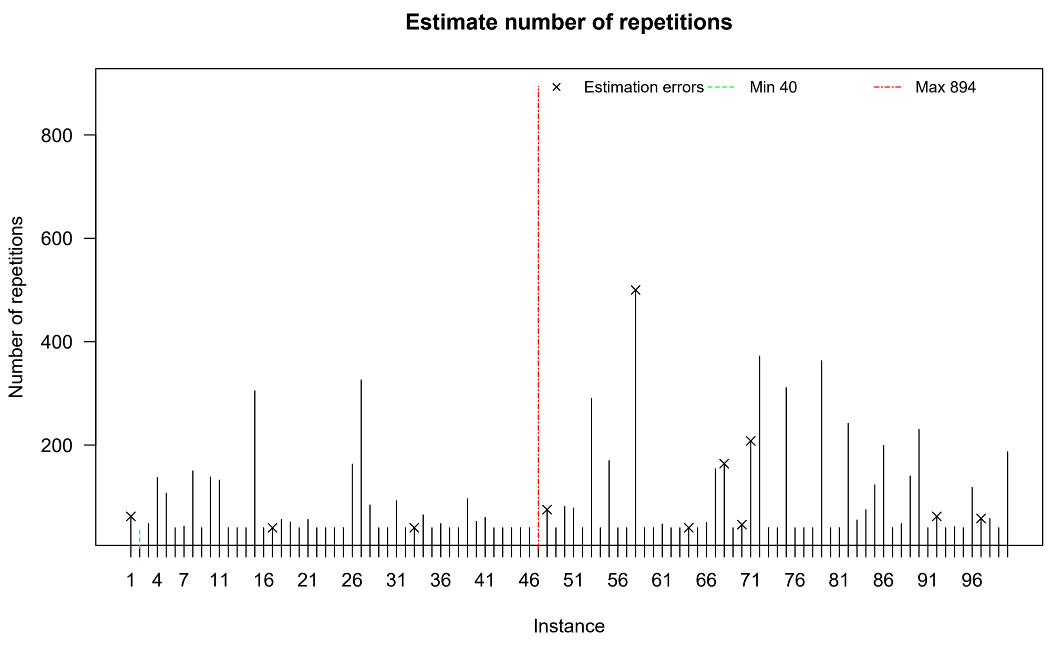

Figure 4.4: Number of repetitions of 100 instances for percent difference between *algorithm 1* and *algorithm 3*. The x markers represent the instances with wrongly estimated μ_{ij} (see Figure 4.2).

The same tests were performed with the absolute differences approach, and the results are shown in Figures 4.5-4.8.

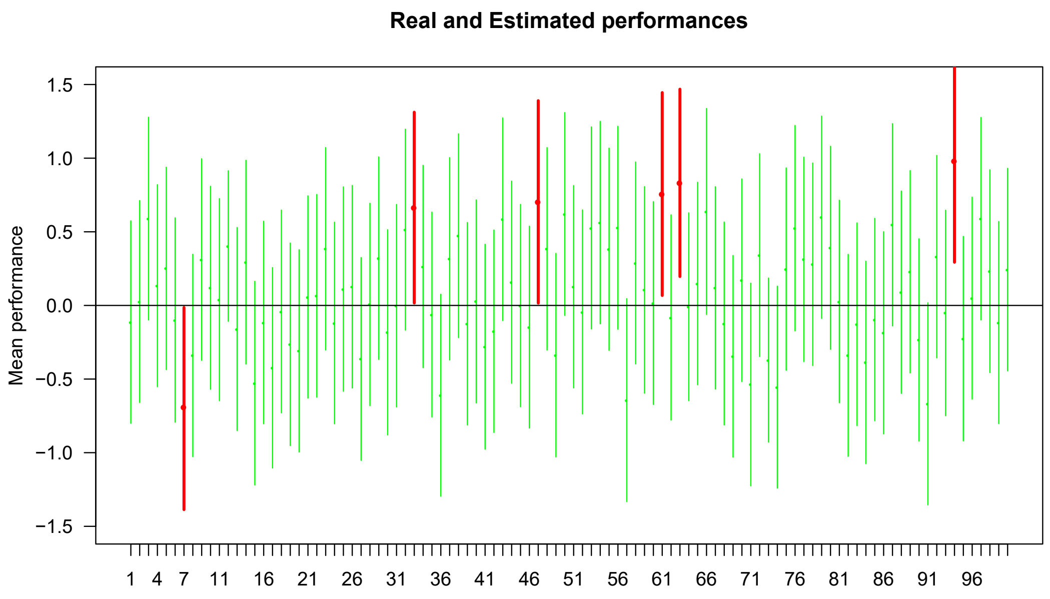

Figure 4.5: Mean performances of 100 instances on absolute difference between *algorithm 1* and *algorithm 2*. The instances for which the real value of μ_{ij} falls outside the confidence interval are marked in red and counted as errors

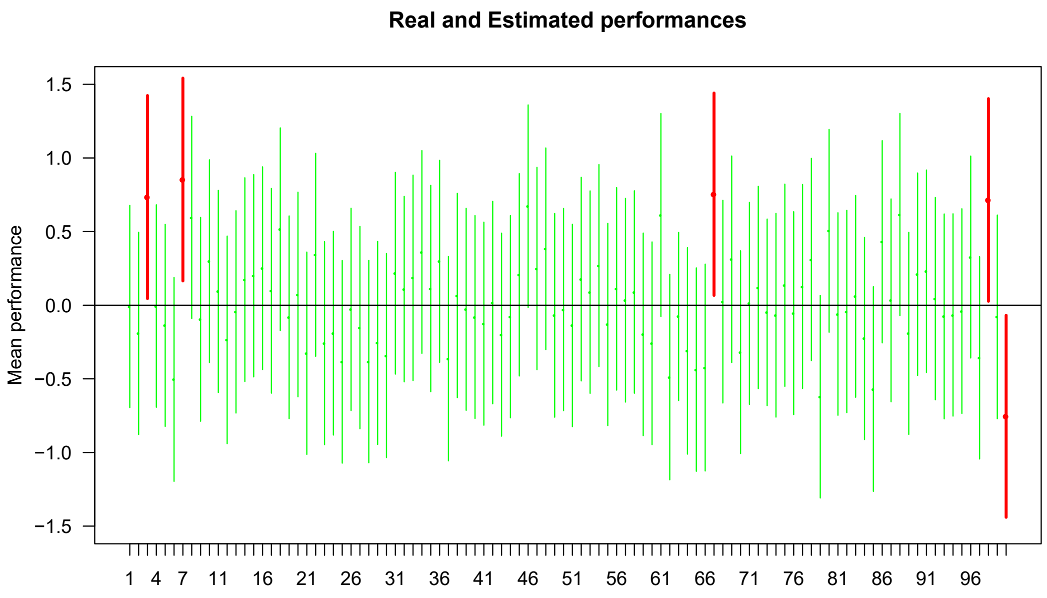

Figure 4.6: Mean performances of 100 instances on absolute difference between *algorithm 1* and *algorithm 3*. The instances for which the real value of μ_{ij} falls outside the confidence interval are marked in red and counted as errors

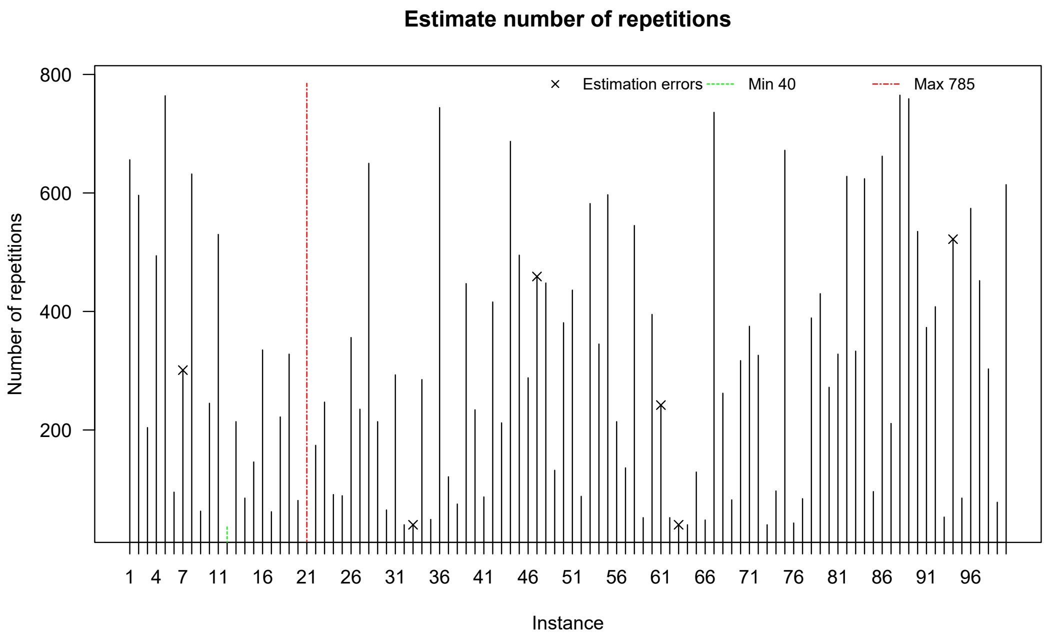

Figure 4.7: Number of repetitions of 100 instances for absolute difference between *algorithm 1* and *algorithm 2*. The x markers represent the instances with wrongly estimated μ_{ij} (see Figure 4.5).

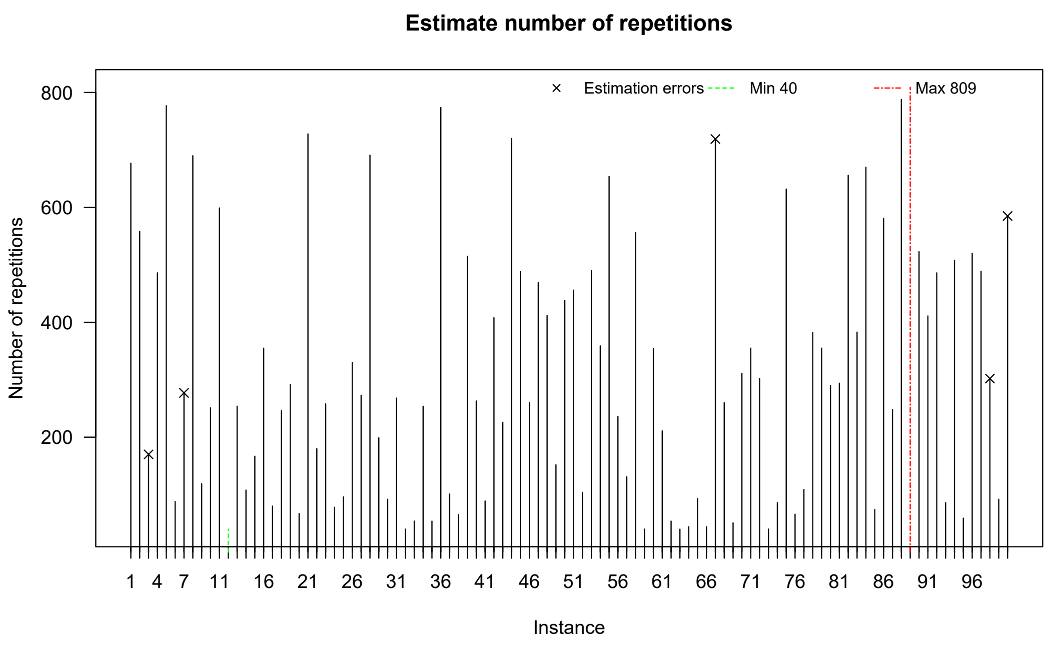

Figure 4.8: Number of repetitions of 100 instances for absolute difference between *algorithm 1* and *algorithm 3*. The *x* markers represent the instances with wrongly estimated μ_{ij} (see Figure 4.6).

With the absolute difference approach, the estimate of the differences of means of algorithms with a large difference in performance did not present more errors than the comparison of algorithms similar in performance, the error percentage in the first case was of 4.9% and for the second one it was only 5.04%. This was also expected, as all assumptions of the confidence interval used in this case are observed.

The test considering shifted Exponential distributions with zero mean, instead of the Gaussian one, samples τ_{ij} and ϵ_{ijk} according to equations (28) and (29), respectively.

$$\tau_{ij} \sim \text{Exp}(\lambda = \sigma_{AI}^{-1}) - \sigma_{AI} \tag{28}$$

$$\epsilon_{ijk} \sim \text{Exp}(\lambda = \sigma_{WI}^{-1}) - \sigma_{WI} \tag{29}$$

The results obtained from these experiments demonstrated similar behaviour to those with the normal distribution. The table 4.1 shows that the same characteristic, with a wider difference between the algorithms performance leading to a wider error for the percent differences case, also happens in this case, as expected.

	Percent diff.	Absolute diff.
algorithm 1 - algorithm 2	5.29	5.28
algorithm 1 - algorithm 3	11.34	5.31

Table 4.1: Error rates when testing with 100 different instances with simulation parameters distributed according to Exponential distributions. An error is considered to happen when the real difference of the means falls outside the confidence interval found for the instance

4.2.2 Validation of the estimated number of replications with fixed sample size

The second objective of the proposed method is to estimate the number of instances required for the comparison of algorithms, so that predefined statistical properties can be maintained. To test the estimation of the required number of instances, we set the effect size of minimal practical significance as $d = 1$ (i.e., it assumes that the user is interested in detecting differences in paired mean performance greater than 1 standard deviation, which is generally considered a large effect size). In this case, sample sizes calculated for power levels of 70%, 80% and 90% yielded 7, 8 and 11 instances, respectively.

To test if these estimated sample sizes can provide the desired power level, the method was executed 1000 times.After each run a paired-samples t-test was performed, and its ability to detect the true difference between the algorithms was evaluated, and the rate of false negatives was recorded. With that, the observed power was estimated, as shown in Table 4.2.e Confidence intervals on the differences between the two algorithms were also derived, and the observed significance level (i.e., the proportion of cases in which the confidence interval did not capture the true value of the parameter of interest) was also recorded. These values are shown in Table 4.3.

In the first case exposed on Table 4.2, the actual effect size was much smaller than the one used for the calculation of the required sample size ($d^* = 1$), resulting in a much lower observed power - the experiment was simply not designed to detect differences that small, and the tests generally fail to report significant results. This was exactly as expected - effect sizes smaller than the *difference of minimal practical relevance*, which is used for planning the experiment, will result in lower power, but this is harmless since they are, by definition, smaller than what is considered to have practical relevance.

		Power for $d^* = 1$		
		P70	P80	P90
algorithm 1 × 2	Percent differences	3.7	3.2	4.9
	Absolute differences	4.0	5.3	3.7
algorithm 1 × 3	Percent differences	73.8	83.6	97.5
	Absolute differences	85.7	93.6	99.4

Table 4.2: Observed power when using Gaussian sampling.

The design was sufficient, however, to detect the differences between algorithms 1 and 3, which was also according to what was expected: if the actual effect size is larger than the minimally relevant one used in the design of the experiment, a larger power will result.

		Power for $d^* = 1$		
		P70	P80	P90
algorithm 1 × 2	Percent differences	4.1	5.2	4.3
	Absolute differences	4.5	4.9	3.8
algorithm 1 × 3	Percent differences	4.8	4.9	6.8
	Absolute differences	5.2	4.7	4.7

Table 4.3: Observed significance levels when using Gaussian sampling.

Table 4.3 shows that the desired significance level was not particularly affected, which is in accordance with the theory - since all parametric assumptions are guaranteed in this first experiment, the (strong) control over the confidence level is ensured.

The behaviour of the method using Exponential sampling for (28) and (29) was similar to the behaviour with the Gaussian sampling. Tables 4.4 and 4.5 show the observed power and significance levels for this experiment.

The conclusions drawn from Table 4.4 are the same as in the Gaussian sampling case, with the tests failing to detect the very small differences (below the stated relevance threshold) in most cases. It is important to notice that the value of the error threshold can also interfere with these conclusions, since the variation of the estimates calculated for each instance propagate down to the total variance, which has a strong impact in the power of the test. For instance, while Table 4.4 reports results calculated with $d_{max} = 0.35$, the same experiment for algorithm

1 and 2 performed with $d_{max} = 0.1$ would yield observed powers of 58.1%, 68.7% and 81.9%, for nominal powers levels of 70%, 80% and 90%, respectively.

		Power for $d^* = 1$		
		P70	P80	P90
algorithm 1 $\times$ *2*	Percent differences	3.4	5.5	11.2
	Absolute differences	8.4	8.9	12.4
algorithm 1 $\times$ *3*	Percent differences	100	100	100
	Absolute differences	100	100	100

Table 4.4: Observed power when using Exponential sampling.

When comparing *algorithms 1* and *3* with Exponential sampling, the difference between the algorithms was also much larger than the effect size for which the tests were designed, leading to extremely high observed power, according to table 4.4.

		Power for $d^* = 1$		
		P70	P80	P90
algorithm 1 $\times$ *2*	Percent differences	2.5	3.2	6.3
	Absolute differences	4.9	4.1	4.7
algorithm 1 $\times$ *3*	Percent differences	6.9	6.1	8.2
	Absolute differences	4.7	5.4	5.3

Table 4.5: Observed significance levels when using Exponential sampling.

It is also interesting to notice that despite using Exponential distributions instead of Gaussian, there were no observable degradations in terms of the statistical properties of the methods. This can be explained in part by the relative robustness of the estimators used to deviations from normality (as long as the sampling distribution of means remains approximately normal). This is of course not always guaranteed when testing heuristics - very heavy tailed and asymmetric distributions can lead to degraded performances, and the use of robust statistics may be necessary to avoid these problems in the general use of the methodology. The investigation of these methods represents a natural next step in this work.

4.3 SUMMARY

The results reported in this chapter validate the proposed methodology on a simulation model. This validation is performed by using the sample size estimation approach on a controlled environment, in which the real variances and effect sizes can be controlled at predefined levels, which allows the comparison between the observed results and the "ground truth" of these simulated experiments. By using the simulation model, we were able to assess that the proposed approach is indeed capable of generating designs that maintain the desired statistical properties stated by the experimenter. An exception that was detected in our simulation experiments was the decrease in the effective confidence level of intervals on the percent differences as the differences increase, something that can be corrected by employing a more consistent interval equation for this case. Another point of attention, also in the case of percent differences, is the need to observe the assumptions under which this approach can be used, particularly with regards to the requirement of strictly positive values for the observations, something that can be easily guaranteed in some problem classes (e.g., routing or scheduling problems) but not so much in others.

5 EXPERIMENTAL EXAMPLES

One essential object is to choose that arrangement which shall tend to reduce to a minimum the time necessary for completing the calculation.

Ada Lovelace, *Notes upon L. F. Menabrea's "Sketch of The Analytical Engine Invented by Charles Babbage". 1842.*

5.1 COMPARISON OF TWO MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS

A case study was to use the proposed method for comparing two variations of the MOEA/D, a decomposition-based multi-objective optimization algorithm, namely the original MOEA/D [80] and the MOEA/D-DE [44]. In this experiment, the performance of the algorithms was measured by means of the IGD indicator [82]. The configuration used for both algorithms is described in the example presented in Appendix B, the implementations used were those available from R package MOEADr [13].

The set of available instances for this experiment was determined considering the protocol suggested by Bezerra *et al* [7]. The instance class used consists of the functions UF1 - UF7 from the the CEC 2009 benchmark with the number of variables ranging from 20 to 60, and excluding 30, 40 and 50, which are reserved for posterior validation. In this way, the maximum number of instances available for the experiment is 266. While in this case one would be justifiably interested in using all available test instances (a total of 217) to obtain a more complete understanding of the behaviour of these algorithms on the problem class of interest, the resulting computational cost of such an exhaustive experiment may be quite large.[1]

1 While in this particular example the required computational budget for exhausting all available instances would not be unattainable, limitations to the number of instances that can be reasonably employed in an experiment can

Consequently, a first step in comparing these two methods may be to investigate whether they present differences in mean IGD performance that exceed some minimal threshold of practical relevance, which can be achieved using a subset of the available instances, at a computational cost much smaller than what would be required for the full investigation.

Initially, the test had the following configurations: Statistic of interest: mean of percent differences; significance level $\alpha = 0.05$; minimally relevant effect size, $d^* = 1.0$; Desired power, $\pi = 0.8$; standard error threshold $d_{max} = 0.1$. The results of running the proposed method for this experimental setup are summarized in table 5.1.

	Instance	est	se	CIl	CIu	n.1	n.2
1	UF_7_37	-0.39	0.10	-0.58	-0.19	29	10
2	UF_1_22	-0.41	0.10	-0.61	-0.22	28	18
3	UF_3_60	-0.50	0.10	-0.70	-0.30	27	10
4	UF_4_48	-0.29	0.10	-0.49	-0.09	12	10
5	UF_3_21	-0.48	0.10	-0.68	-0.28	34	10
6	UF_6_33	-0.37	0.10	-0.57	-0.16	13	10
7	UF_5_24	-0.37	0.10	-0.58	-0.16	11	10
8	UF_1_32	-0.39	0.10	-0.60	-0.19	14	10

Table 5.1: Results returned by the proposed method for the first experiment comparing the two MOEA/D methods. The instances were randomly sampled from the available set. *x.est* refers to the δ point estimate of the difference of means of the two algorithms for each instance, in this case ["MOEA/D-DE" - "MOEA/D"]. *x.se* refers to the estimated standard error, which was controlled exactly at the nominal level for all cases. *x.CIl* and *x.CIu* refer to the lower and upper limits of the 95% confidence interval for each estimate, and *x.n.1* and *x.n.2* refer to the number of runs required, on each instance, for the MOEA/D and MOEA/D-DE, respectively.

The results presented in Table 4 deserve some attention. Notice that the number of repetitions performed for the MOEA/D was generally superior to that of MOEA/D-DE, suggesting that the former algorithm presented generally larger standard deviations. Also, notice that the standard error was controlled at the nominal threshold in all cases, suggesting that the proposed method does indeed generate the repetitions needed for obtaining the desired ac-

be much more severe when researching, for instance, heuristics for optimizing numerical models in engineering applications, or other expensive optimization scenarios [73].

curacy in the estimation of these values. Figure 5.1 presents the confidence intervals for the percent differences of means observed in each instance.

To compare the two methods in terms of mean differences in IGD for the problem class, a paired t test was performed using the paired difference estimators (column *x.est* in Table z5.1) as the individual observations. The results of this test were statistically significant at the 95% confidence level ($p = 5.9 \times 10^{-7}$; $N = 8$; $CI_{.95} = [-0.456, -0.345]$, providing evidence that the second algorithm presents IGDs that are systematically lower than those yielded by the first.

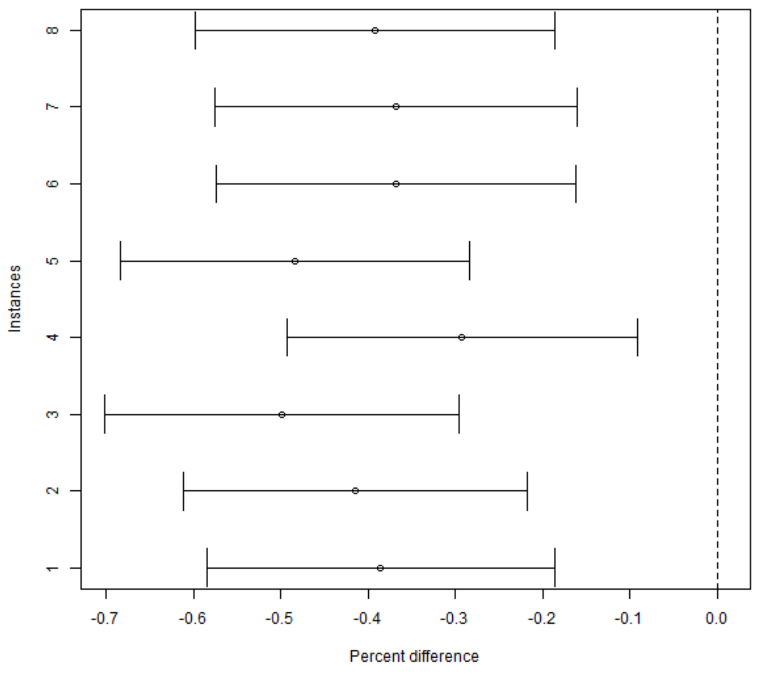

Figure 5.1: 95% confidence intervals on the percent differences of means for each instance, for the first comparative experiment.

A second experiment was performed to illustrate the behaviour of the proposed method when facing algorithms with an expected smaller *real* difference in performance. In this experiment, we compared two configurations of the MOEA/D-DE: the standard one, described in Appendix B, and another MOEA/D-DE in which only the population size was changed, increasing it by 5% (by changing the parameter H in the decomposition strategy from 99 to 104). Considering an interest in mean percent gains, the effect size of minimal practical relevance was defined in this experiment as $d^* = 0.5$ (leading to an estimated sample size of $N = 32$ instances), and the standard error threshold was selected as $d_{max} = 0.1$.

Table 5.2 displays the summary of the resulting experimental design after applying the proposed method. In this case, a paired t-test performed on the point estimates for the percent differences in means of each instance returned a non-statistically significant result ($p = 0.2562$; $N = 32$; $CI_{.95} = [-0.0440.0122]$, suggesting the absence of differences equal to, or greater than the effect size of minimal practical relevance. Whatever difference in mean percent differences may exist between these two algorithms, it is probably smaller than the predefined relevance threshold and, consequently, not interesting from the practical perspective of the experimenter.

	Instance	x.est	x.se	x.CIl	x.CIu	x.n.1	x.n.2
1	UF_6_59	-0.07	0.07	-0.23	0.09	10	10
2	UF_7_26	-0.03	0.08	-0.20	0.15	10	10
3	UF_7_51	-0.00	0.09	-0.20	0.19	10	10
4	UF_2_43	-0.13	0.07	-0.28	0.01	10	10
5	UF_6_48	0.16	0.10	-0.04	0.35	13	12
6	UF_2_36	-0.02	0.08	-0.19	0.15	10	10
7	UF_5_28	-0.10	0.08	-0.27	0.07	10	10
8	UF_6_24	-0.01	0.09	-0.21	0.19	10	10
9	UF_3_48	-0.07	0.08	-0.23	0.10	10	10
10	UF_4_31	-0.08	0.07	-0.23	0.08	10	10
11	UF_4_35	-0.06	0.10	-0.26	0.15	17	10
12	UF_3_28	-0.00	0.10	-0.20	0.20	10	11
13	UF_4_53	-0.10	0.06	-0.23	0.03	10	10
14	UF_6_33	0.01	0.07	-0.14	0.16	10	10
15	UF_1_52	-0.16	0.08	-0.32	-0.00	10	10
16	UF_6_52	0.08	0.08	-0.10	0.25	10	10
17	UF_4_28	0.04	0.09	-0.14	0.23	10	10
18	UF_3_36	-0.07	0.10	-0.27	0.13	10	11
19	UF_3_26	-0.07	0.08	-0.24	0.10	10	10
20	UF_4_56	0.13	0.07	-0.03	0.28	10	10
21	UF_1_47	-0.07	0.10	-0.27	0.14	10	12
22	UF_1_43	0.06	0.09	-0.13	0.25	10	10
23	UF_2_44	-0.02	0.06	-0.15	0.11	10	10
24	UF_1_59	0.02	0.07	-0.13	0.17	10	10

25	UF_5_57	0.06	0.10	-0.14	0.26	10	10
26	UF_2_58	0.11	0.10	-0.09	0.31	11	10
27	UF_4_43	0.08	0.07	-0.07	0.23	10	10
28	UF_4_27	-0.09	0.07	-0.23	0.06	10	10
29	UF_6_21	0.05	0.06	-0.07	0.17	10	10
30	UF_4_33	-0.12	0.06	-0.24	0.01	10	10
31	UF_4_34	0.00	0.09	-0.18	0.18	10	10
32	UF_5_22	-0.04	0.07	-0.19	0.11	10	10

Table 5.2: Results returned by the proposed method for the second experiment comparing the two MOEA/D-DE with different population sizes. The instances were randomly sampled from the available set. *x.est* refers to the point estimate of the difference of means of the two algorithms for each instance, in this case ["MOEA/D-DE" with $h = 104$ - "MOEA/D-DE" with $h = 99$]. *x.se* refers to the estimated standard error, which was controlled exactly at the nominal level for all cases. *x.CIl* and *x.CIu* refer to the lower and upper limits of the 95% confidence interval for each estimate, and *x.n.1* and *x.n.2* refer to the number of runs required, on each instance, for the "MOEA/D-DE" with $h = 99$ and "MOEA/D-DE" with $h = 104$, respectively.

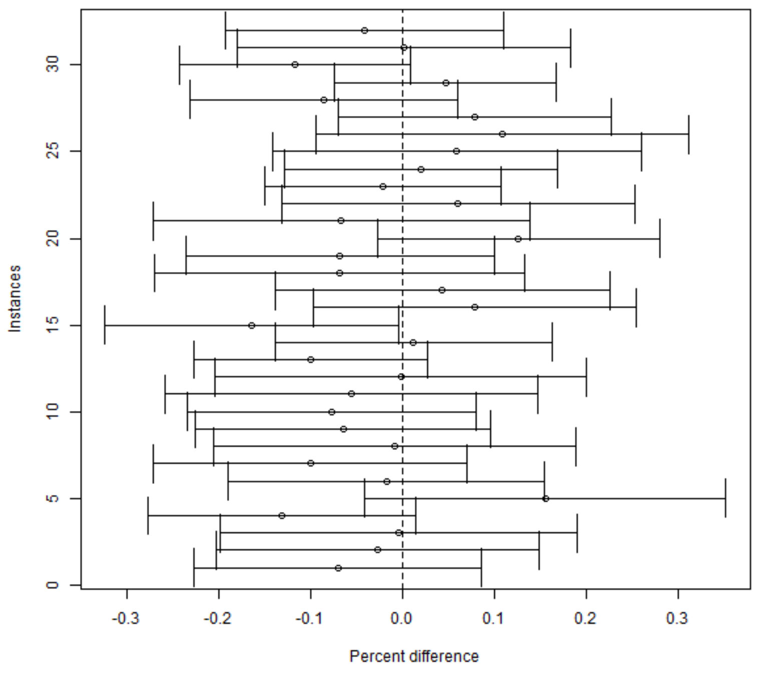

Figure 5.2: 95% confidence intervals on the percent differences of means for each instance, for the second comparative experiment.

A third test was done considering the common situation in which a limited number of instances is available - in our case, only 7 instances (UF1 - UF7, with dimension 25). In this case, since N is fixed, the experiment exists with a tradeoff between power and effect size, as illustrated in Figure 5.3.

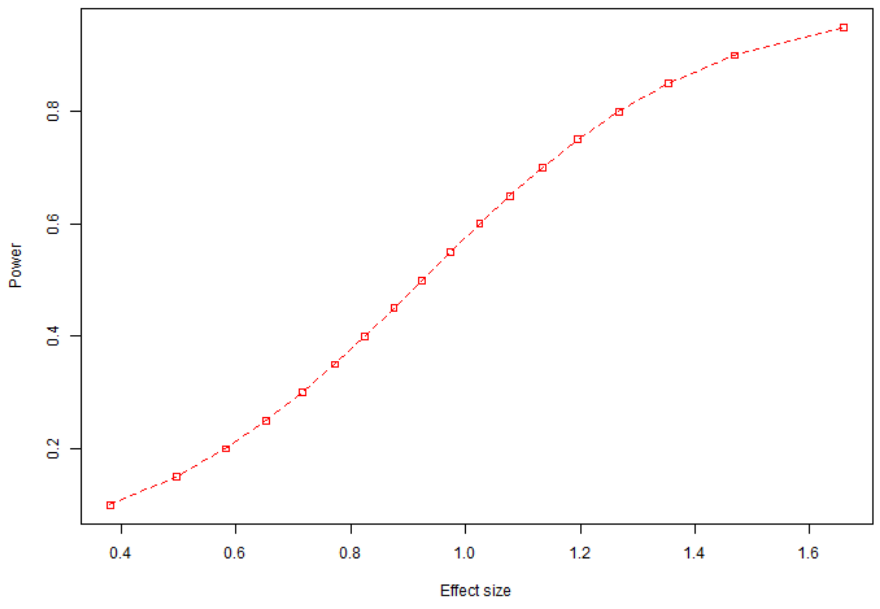

Figure 5.3: Power profile for an experiment with a fixed sample size of 7 instances

To perform this experiment, we compared the standard version of the MOEA/D-DE (described in Appendix B) against another version of the same algorithm, in which the polynomial mutation operator was removed (i.e., a version in which the only variation operator is the differential mutation). The standard error tolerance was set as $d_{max} = 0.1$ in this case.

Table 5.3 shows the summary of the resulting experiment. As in the first test using the MOEA/D variants, the number of repetitions was relatively small (in many cases only the minimum sample, $N_{start} = 10$, was required) due to the less strict requirements regarding the standard errors. In this experiment, the paired t test yielded a statistically significant result at the 95% confidence level ($p = 2.9 \times 10^{-10}$; $N = 7$; $CI_{.95} = [-0.891, -0.837]$, which suggests that the effect of the removed operator (polynomial mutation) is indeed important in terms of the algorithm's ability to yield results with superior performance.

	Instance	x.est	x.se	x.CIl	x.CIu	x.n.1	x.n.2
1	UF_1_25	-0.82	0.10	-1.03	-0.62	10	13
2	UF_2_25	-0.83	0.10	-1.03	-0.62	25	13
3	UF_3_25	-0.86	0.10	-1.07	-0.66	16	10
4	UF_4_25	-0.90	0.10	-1.10	-0.70	14	10
5	UF_5_25	-0.89	0.09	-1.08	-0.70	10	10
6	UF_6_25	-0.88	0.09	-1.08	-0.69	10	10
7	UF_7_25	-0.87	0.10	-1.07	-0.67	16	10

Table 5.3: Results returned by the proposed method for the third experiment comparing the two MOEA/D methods. The instances were the 7 UF functions fixed with parameters dimension of 25. *x.est* refers to the point estimate of the difference of means of the two algorithms for each instance, in this case ["MOEA/D-DE" - "MOEA/D"]. *x.se* refers to the estimated standard error, which was controlled exactly at the nominal level for all cases. *x.CIl* and *x.CIu* refer to the lower and upper limits of the 95% confidence interval for each estimate, and *x.n.1* and *x.n.2* refer to the number of runs required, on each instance, for the MOEA/D and MOEA/D-DE, respectively.

The individual differences in this experiment, as in the previous one, show a greater distance to zero than the first one, as illustrated in figure 5.4. As mentioned above, this provides evidence that the polynomial mutation operator is important as a variation heuristic within the MOEA/D-DE algorithm, at least for the (hypothetical) class of instances for which functions UF1-UF7 with 25 variables represent a reasonable sample.

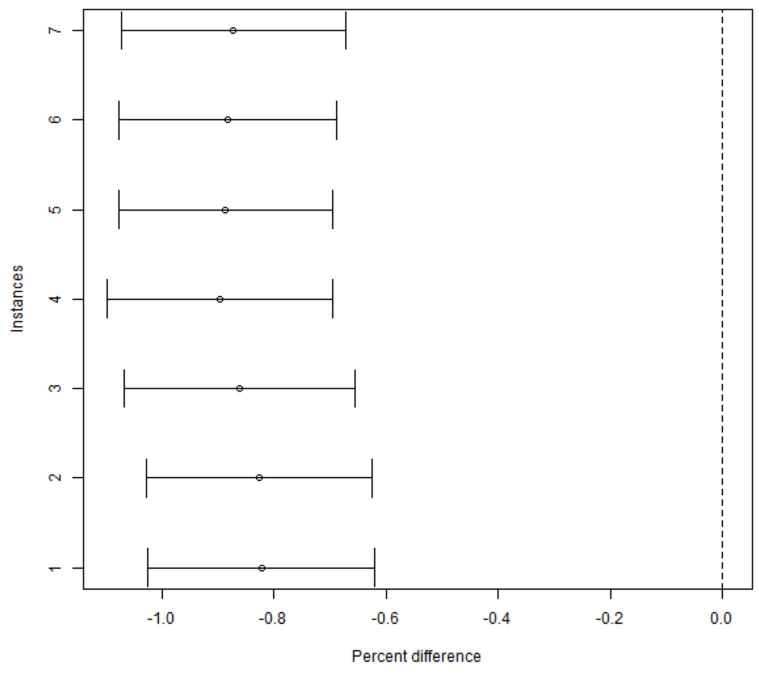

Figure 5.4: 95% confidence intervals on the percent differences of means for each instance, for the third comparative experiment.

5.1.1 Comparison with bootstrapping

These examples, however, are merely illustrative. When inspecting the assumption for executing such tests it is possible to notice that the data is not normal, for instance the Shapiro-Wilk test for normality returns a p-value of 1.55×10^{-5}, for the percent differences of 32 random instances and other extremely low values for data sampled from the execution of the tested algorithms. Therefore, this experiment would be more correctly handled using a non parametric approach, as follows.

A more realistic approach is to use bootstrapping due to the characteristics of the data. Therefore, the proposed methodology was, then, used to investigate the mean *percent* differences of performance between the two algorithms summarized in Table 5.4 on the problem class of interest. The parameters used for this experiment were defined as follows: $\alpha = 0.05$, $\beta^* = 0.2$, $d^* = 0.5$, $n_0 = 15$, $n_{max} = 200$, $se^* = 0.05$. The standard errors were calculated using the bootstrap approach (Algorithm 3), using $R = 999$; and the number of instances was calculated assuming the use of a t test and a bilateral alternative hypothesis (21).[2]The specific parameters of these two algorithms are summarized in Table 5.4, and a detailed explanation can be found in the relevant literature [12, 80, 44].

Table 5.4: Algorithms and parameters. **Boldface** entries highlight differences. n_v denotes the dimension of the problem instance being solved.

Component	Alg. 1: MOEA/D	Alg. 2: MOEA/D-DE
Decomposition strategy	SLD ($H = 99$)	SLD ($H = 99$)
Neighborhood strategy	By weight vectors ($T = 20, \delta = 1.0$)	By weight vectors ($T = 20, \delta = \mathbf{0.9}$)
Aggregation function	Weighted Tchebycheff	Weighted Tchebycheff
Variation Operators	SBX ($\eta = 20, p_c = 1$)	**Differential mutation** `/rand/1` ($F = 0.5$) **Binomial recombination** ($CR = 1.0$)
	Polynomial mutation ($\eta = 20, p_m = 1/n_v$)	Polynomial mutation ($\eta = 20, p_m = 1/n_v$)
Update strategy	Standard update	**Restricted update** ($n_r = 2$)
Stop criterion	$2000n_v$ function calls	$2000n_v$ function calls

Following the procedure outlined in Section 3.2, the proposed methodology indicated that the required number of instances in this case was $N^* = 34$. This amount of instances was randomly sampled (without replacement) from the set of available instances, and the two algorithms were run on each instance according to the procedure defined in Algorithm 2. The results of this process are summarized in Table 5.5.

Some interesting remarks can be made regarding the results summarized in Table 5.5. First, we observed negative values of $\widehat{\delta_j}$ in the majority of instances tested, suggesting an advantage of the MOEA/D-DE over the original MOEA/D (recall that smaller IGD is better). MOEA/D-DE also seems to require smaller sample sizes in most instances, which indicates lower vari-

2 The full replication script for this experiment is available in the Vignette "*Adapting Algorithms for CAISEr*" of the `CAISEr` package [14].

Table 5.5: Summary of results obtained in Experiment 1. Instances marked in **boldface** were sampled up to the maximum allowed budget, $n_{max} = 200$.

Instance (dim.)	$\widehat{\delta}_j$	$\widehat{se}_{\widehat{\delta}_j}$	n_{1j}	n_{2j}	Instance (dim.)	$\widehat{\delta}_j$	$\widehat{se}_{\widehat{\delta}_j}$	n_{1j}	n_{2j}
UF4 (13)	-0.14	0.02	15	15	**UF5 (17)**	0.46	0.05	83	117
UF2 (29)	-0.36	0.05	65	15	UF3 (15)	-0.08	0.05	40	53
UF5 (28)	0.69	0.05	80	120	UF4 (16)	-0.11	0.03	15	15
UF1 (29)	-0.63	0.05	25	15	UF7 (18)	-0.89	0.05	41	33
UF2 (36)	-0.29	0.05	71	16	UF7 (38)	-0.86	0.05	32	16
UF3 (29)	0.05	0.05	99	101	UF4 (14)	-0.21	0.02	15	15
UF3 (10)	0.07	0.05	57	58	UF1 (11)	-0.82	0.02	15	15
UF7 (16)	-0.95	0.00	15	15	UF1 (16)	-0.75	0.02	15	15
UF7 (29)	-0.90	0.04	15	15	UF2 (32)	-0.35	0.05	51	15
UF2 (25)	-0.38	0.05	66	15	UF3 (24)	-0.19	0.05	42	47
UF4 (30)	-0.04	0.02	15	15	UF6 (34)	-0.74	0.05	15	15
UF1 (26)	-0.65	0.05	15	15	UF4 (32)	-0.00	0.03	15	15
UF2 (18)	-0.46	0.05	40	15	UF2 (11)	-0.47	0.05	46	15
UF7 (36)	-0.92	0.02	15	15	UF2 (22)	-0.54	0.05	44	15
UF4 (18)	-0.17	0.02	15	15	UF1 (17)	-0.71	0.03	15	15
UF2 (34)	-0.40	0.05	44	15	UF1 (18)	-0.69	0.03	15	15
UF2 (39)	-0.29	0.05	71	15	UF3 (23)	-0.18	0.05	33	58

ance on several instances, a desirable feature since it means that the algorithm tends to return more consistent performance values across repeated runs.

Another noteworthy point is that in three of the 34 instances sampled – UF3 (29), UF5 (17), and UF5 (28), boldfaced in the table – the maximum allocated budget ($n_{max} = 200$) was not enough to reduce the standard error $\widehat{se}_{\widehat{\delta}_j}$ below the predefined threshold of $se^* = 0.05$. In these three cases the second algorithm, MOEA/D-DE, seems to present an unusually high variance (evidenced by the large number of runs attributed to it by the proposed sampling methodology), resulting in the need for a larger number of repeated runs to reduce the uncertainty on the estimate of $\widehat{\delta}_j$. However, since the resulting standard errors in these three cases were not particularly high[3], their effect on the total residual variance is likely negligible.

Continuing with the experimental procedure, a t-test performed on our sample of estimated paired differences of performance yields statistically significant results ($p = 2.90 \times 10^{-6}, df = 33$) with an estimated paired mean difference in IGD of $\widehat{\mu}_D = -0.379$ ($CI_{0.95} = [-0.517, -0.242]$), which means an expected value of IGD for the MOEA/D-DE that is (37.9 ± 13.7) % better than that of the original MOEA/D for our problem class of interest.

The normality assumption of the t-test can be easily validated using the normal QQ-plot shown in Figure 5.5. The plot indicates that no expressive deviations of normality are present, which gives us confidence in using the t test as our inferential procedure of choice, since the sampling distribution of the means will be even closer to a Normal variable than the data distribution, diluting whatever small deviations from normality may be present.

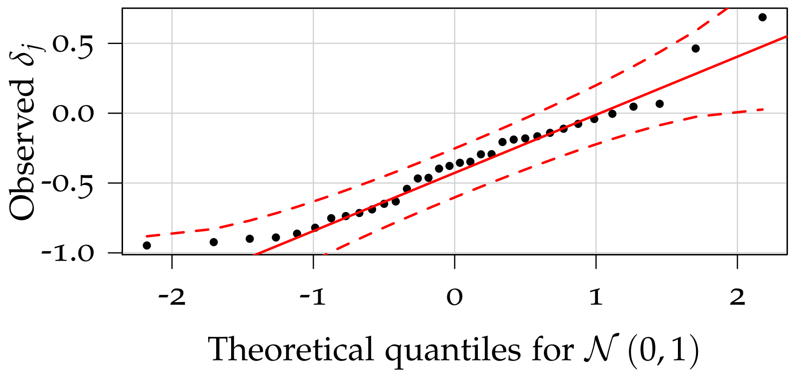

Figure 5.5: Normal quantile-quantile plot for observations $\widehat{\delta}_j$ in Experiment 1.

Finally, it is important to reinforce that these results could also be used to motivate further analyses of the performance of these two algorithms for problems belonging to the problem class of interest, even before proceeding to the full, exhaustive test on all available instance. For example, the individual IGD distributions and mean values of each algorithm on each instance, presented in Figure 5.6, suggest that both algorithms encounter difficulties when solving *UF5* (and, to a lesser extent, *UF3*) instances, which could motivate a more focused

3 More specifically: $\widehat{se}_{\widehat{\delta}_j} = 0.0518$ for *UF5 (28)*; $\widehat{se}_{\widehat{\delta}_j} = 0.0544$ for *UF3 (29); and* $\widehat{se}_{\widehat{\delta}_j} = 0.0536$ *for UF5 (17)*

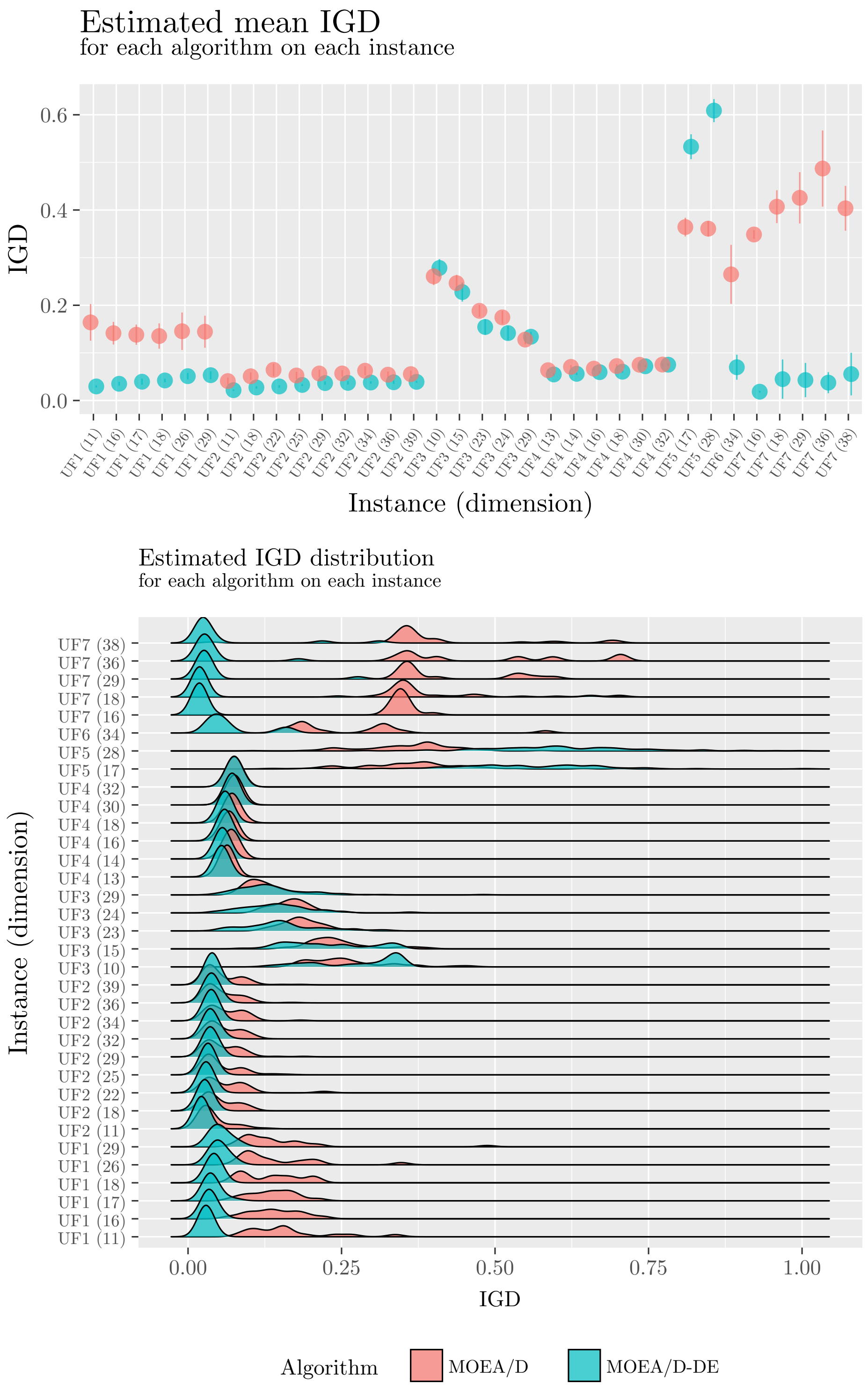

Figure 5.6: Top: 95% confidence intervals on the means of individual IGD values of each algorithm on each sampled instance. **Bottom**: Density estimates of IGD for MOEA/D and MOEA/D-DE on each sampled instance. Notice the discrepant performance of MOEA/D-DE on instance *UF5*.

investigation into the reasons for these poor performance profiles, and on possible algorithmic improvements to remedy this problem. A natural follow-up to the experiment presented in this first example would be to broaden the investigation to include the full available test set, in which case the proposed methodology could still be useful in defining the number of repetitions to be performed for each algorithm on each test instance, as well as the expected statistical power of whatever subgroup comparisons the researcher could deem interesting.

5.2 PARALLEL MACHINES PROBLEM WITH SEQUENCE DEPENDENT SETUP TIMES

The proposed methodology can also be useful in situations when the researcher uses a predefined set of benchmark instances to compare two algorithms. To illustrate this case, we used a set of 200 large instances of the unrelated parallel machines problem with sequence dependent setup times, provided by Vallada and Ruiz [75] for calibration experiments.[4] Currently the best results for this problem are those presented by Santos *et al.* [62] using a simulated annealing algorithm with six neighbourhood structures (*Shift*, *Switch*, *Task move*, *Swap*, *Two-Shift*, and *Direct swap*), randomly selected at each trial move.[5].

Preliminary results by Maravilha *et al.* [49] suggest that the most influential neighborhood structure for this case is *Task move*, which presents the largest expected improvement value across a wide range of problem sizes. To isolate and quantify the effect of this specific neighbourhood structure to the performance of the method, two versions of the algorithm were compared: a *full version*, which is the original algorithm equipped with all six neighbourhood structures; and a *no-task-move* version, which uses exactly the same structure but does not include the *Task move* neighbourhood. As mentioned above, these two versions were tested on the calibration test set proposed by Vallada and Ruiz [75], which features 200 large instances with $M \in \{10, 15, 20, 25, 30\}$ machines and $N \in \{50, 100, 150, 200, 250\}$ jobs. All algorithmic aspects were set exactly as in Santos *et al.*'s work [62], with the stop criteria employed at each instance being the total run time, calculated as a function of instance size following guidelines from the original references [75, 62].

Given that the number of instances is predefined, there is no need to calculate it using the approach presented in Section 3.2. Instead, we used the proposed methodology to estimate the power curve of the experiment, that is, the expected sensitivity of this comparison to detect effects of different magnitudes. This is illustrated in Figure 5.7, which was derived assuming that the desired significance of the experiment is $\alpha = 0.05$, and that a t-test will be performed using a one-sided alternative hypothesis, since we have a prior expectation that the *full version* algorithm should be better than the *no-task-move*, and are interested in testing and quantifying this effect.

4 The instance files can be retrieved from `http://soa.iti.es/problem-instances`

5 The source codes used for this experiment can be retrieved from `http://github.com/andremaravilha/upmsp-scheduling`

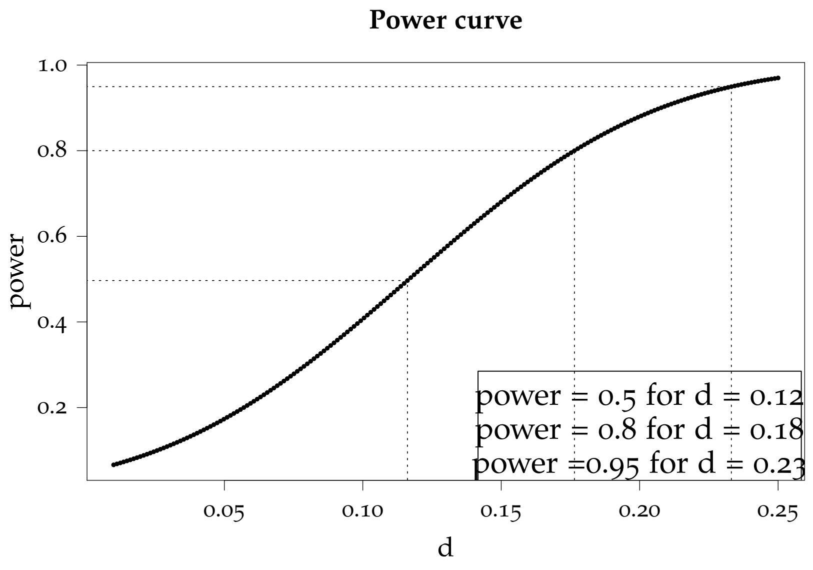

Figure 5.7: Expected sensitivity of experiment 2 to different effect sizes, for a t-test test with a one-sided alternative hypothesis. With 200 instances we can be fairly confident that the experiment will be able to identify mean performance gains greater than approximately 0.2 standard deviations.

As suggested in the figure, this experiment has a reasonable probability of detecting mean performance gains due to the use of the *Task move* neighbourhood structure greater than about 0.2 standard deviations. Smaller differences in mean performance, particularly under about 0.1 standard deviations, can go undetected, but in terms of impact on the expected behaviour of the algorithm these would be really minor effects.

The experiment was performed using the proposed method for iteratively estimating the required number of repetitions for each algorithm on each of the 200 instances. The experimental parameters were set as $se^* = 0.05$ on the percent differences, $n_0 = 15$ and $n_{max} = 150$. The standard errors were calculated using the parametric formulas provided in Section 3.1.2. A t test performed on the resulting data suggested significant differences at the 95% confidence level ($p < 2 \times 10^{-16}, df = 199$, against a one-sided, lower H_1) with an estimated paired mean difference of $\widehat{\mu}_D = -0.361$ ($CI_{0.95} = [-0.380, -0.342]$), which means that the expected impact of the *Task move* neighbourhood on the performance of the algorithm, for an instance belonging to the same problem family defined by the test, set is a reduction of $(36.1 \pm 1.9)\,\%$ in the makespan of the final solution returned.[6]

6 The graphical analysis of the residuals did not suggest expressive deviations of normality. The results table and residual analysis are provided in the Supplemental Materials.

Notice that further analyses could (and should) be performed on this same data, to refine the conclusions and, possibly, suggest new lines of inquiry. For instance, while the overall expected improvement due to the use of *Task move* in the pool of possible movements is quantified as $(36.1 \pm 1.9)\,\%$, knowledge, e.g., of instance size can improve the estimation accuracy of performance gains. This is illustrated in Figure 5.8, which suggests that, while the use of *Task move* provides relevant improvements across all problem sizes tested, its effect increases with the number of machines (M) and decreases with the number of jobs (N). A detailed quantification of these effects and the reasons behind them is, however, outside the scope of the present work.

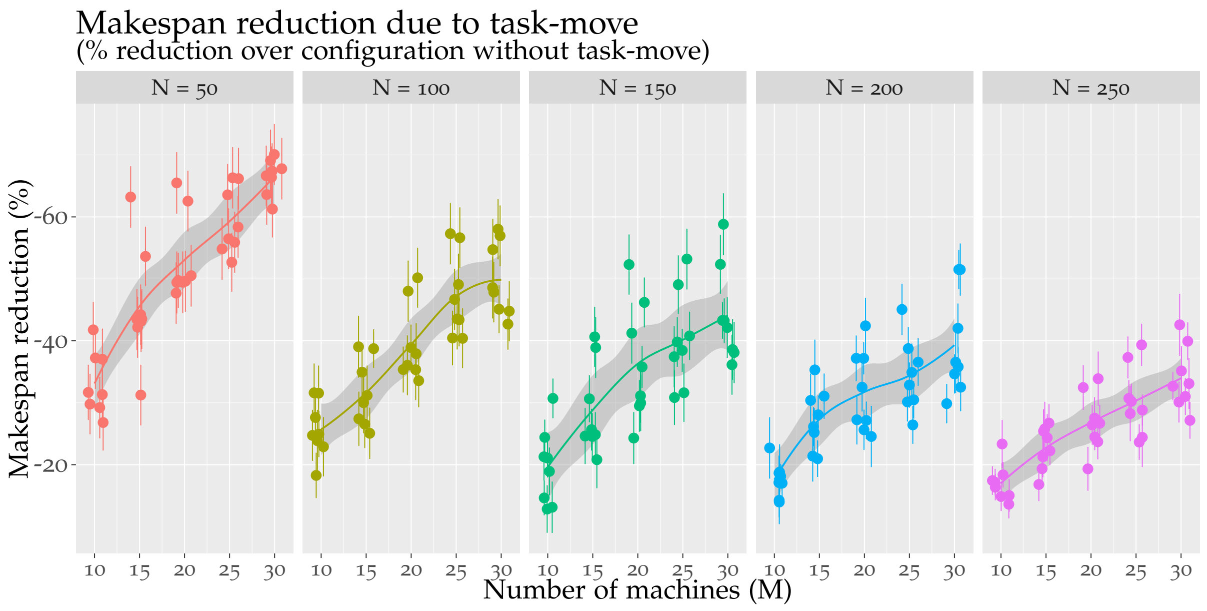

Figure 5.8: Percent gains in performance attributable to using the *Task-move* neighbourhood. The y-axis indicates how much lower the average makespan was for the full algorithm in comparison to the *no-task-move* version (notice that the y-axis is vertically reversed). Vertical lines represent the standard errors of each observation. The x-coordinates of the observations were perturbed slightly around their true values ($M = \{10, 15, 20, 25, 30\}$), for visualization purposes.

5.3 SUMMARY

In this chapter, three case studies were presented, using variations of the MOEA/D as our test algorithms, and the CEC2009 benchmark set for dimensions between 20 and 60 as our problem class of interest. In these experiments we were able to observe situations which we believe reflect usual applications for our proposed methodology. The two first experiments considered different specifications for the calculation of the sample size, and the third considered a scenario in which the number of instances was fixed, and the method was used only to determine the number of repetitions. A power curve for this third experiment was also provided, as an example of the type of reflection that researchers and practitioners should considerate when designing experiments using fixed benchmark sets. These tests were however, made with a parametric approach, which should not be used with the experimented data, another example was made using bootstrap to compare other two variations of the same algorithm.

By the end of the chapter a last experiment is shown. It consists in testing the effect of a neighbourhood in a scheduling parallel machines problem. Particularly, it was allowed to quantify the improvement of a *Task move* neighbourhood structure to the performance of the method, when large instances of unrelated parallel machines problem with sequence dependent setup times are used.

6 CONCLUSION

One never notices what has been done; one can only see what remains to be done.

Marie Curie, letter to her brother (1894)

In this work we presented a methodology for sample size calculation (in terms of number of instances and of repeated runs on each instance) for the performance comparison of two algorithms on a class of problems. While this may not have been adequately stressed throughout the text, a critical aspect of such comparisons is the concept of a problem class for which the conclusions of any inference based on a finite sample can be drawn. Even when the problem class is not explicit in the available test instances, the experimenter should always be aware that his or her conclusions are valid for a (possibly hypothetical) population of problems for which the instances used can be seen as a representative sample.

If the algorithm comparison is made with a representative sample of instances of the considered family, appropriate statistical procedures can be used the to infer the quality of the algorithms performances when solving this type of problem. These procedures can tell if a given pair of algorithms has similar performance, or how much better one is than the other.

One aspect of the proposed methodology is the generation of observations for each algorithm, in order to make the data comparable within a controlled margin of error. Which is interesting not only from the perspective of statistical inference (e.g., maintaining desired power levels for a given effect size of practical interest), but also from an *accuracy in parameter estimation* perspective, in which precise estimates of the differences within each instance can also inform the experimenter about different aspects of the algorithms under investigation.

This number of runs of each algorithm on each instance is determined iteratively, approximating optimal sample size ratios for the reduction of the uncertainty associated with the estimation of paired differences in performance. Experiments performed using a controlled simulation model show that the number of executions to estimate the average behaviour of an algorithm for a problem can vary from a minimal of few dozens to thousands of repetitions, depending mainly on the relationship between the stated tolerance for the standard error (or the confidence interval half-width) and the residual variance within each instance. These results point to the need of properly determining the number of repetitions, instead of using a predefined value as a rule. Considering that if an arbitrary number of repetitions is used, the estimate performance of an algorithm on instances that require much more observations might be so far off that could lead to flawed conclusions on the algorithms performances.

Besides the generating the observations, the determination of the number of instances was also done. It is based on considerations of *practically relevant differences* and on the desired statistical power to detect them. For this, analytic solution were presented for the parametric case, for both the one sided and two sided cases. For the non parametric case, approximations based on the asymptotic relative efficiency of the Wilcoxon signed-ranks and the binomial sign test were provided.

The proposed method addresses the comparison of algorithms trying to facilitate the use of correct experimental methodologies for algorithm experimentation, in particular comparison of optimization metaheuristics. While there is a recognized need of a good experimental design, not many studies have been made in order to respond to this demand in a simple way, this lack of scientific production on this theme, helped to consolidate the current experimental "short-cuts" in the field. It is important to reinforce that the proposed methodology is by no means an universal way to test algorithms: when the goal of the experiment is to characterize an heuristic, how robust it is and its best / worst case performance behaviour, different methodologies can and should be employed. However, such extensive experimentation is prohibitive in a number of scenarios, such as in several cases of applied engineering optimization [73] or when comparing heuristics on very large, time-consuming instances.

One of the main aims of this work was to lay the statistical and methodological groundwork for the calculation of required sample sizes in the experimental comparison of algorithms. While the developments and results presented do fulfill this particular goal, there are a number of limitations and possibilities of continuity that can be explored. We finish this work by examining a few of the most promising ones.

6.1 FUTURE WORKS

Some major limitations of the methodology developed in this work are: (i) the fact that it is only defined for the comparison of two algorithms; (ii) the fact that the definition of the number of instances is performed *a priori*, using a fixed sample size methodology; and (iii) the fact that only centrality statistics (the mean and, to a certain extent, the median) were considered.

Regarding the number of algorithms considered in the comparison, a natural next step of this work is to extend the sample size estimation methodologies for multiple algorithms. This can be achieved in a relatively straightforward manner for the estimation of the number of instances, using standard formulas for either omnibus tests (e.g., ANOVA, Friedman) or planning directly for the eventual post-hoc pairwise comparisons [51]. Estimating the number of runs, however, will require greater improvements on the method proposed in this work, probably based on the definition of standard error thresholds for each individual algorithm on each instance, instead of on the standard error of the differences.

While the *a priori* definition of the number of instances provides a reasonable expectation of statistical power for a given MRES, the required sample size may be considerably smaller if the actual effect size is much larger than the predefined d^*. Using sequential analysis methodologies, such as the ones commonly employed in clinical trials or industrial settings [9, 4], may result in a reduced number of instances being necessary to determine the existence of differences between two (or eventually more) algorithms, and represent another possibility of continuity for this work.

The possibility of using the methodology defined in this work as a framework for comparisons of algorithms according to different statistics – e.g., variances, rates of convergence, regression coefficients, or best/worst cases – is yet another promising direction. While most experiments still focus on average (mean/median) cases, the need for methodologically sound comparisons of other quantities has long been recognized [36, 23], and we believe the methodology presented in this paper can be easily adapted for such comparisons. First, the bootstrap approach for the calculation of the number of runs can be extended to different measures of paired differences in performance - medians, quantiles, or other statistics - in a relatively straightforward manner (using balanced samples if needed, or deriving optimal ratios for these other statistics). Moreover, standard statistical tests for other quantities tend to be readily available, and analytic formulas for power and sample size are in most cases available

[51], providing a rich basis upon which better, more comprehensive protocols for algorithmic comparisons can be built.

Appendix

A STATISTICAL CONCEPTS

To aid the comprehension of this work, some statistical concepts needed to introduce the proposed method will be discussed in this appendix.

A.1 DATA SAMPLES

In statistics a data sample is a set of observations selected from the statistical population being studied. This population is composed by several entities with similar characteristics used to explain or describe the studied event or object [61]. For instance, in this work, the samples are the result from an execution of an algorithm, the population would be all the outcomes of a possibly infinite number of executions, and the studied object is the performance of this algorithm when compared against others.

The question of interest addressed in the current work deals with the performance of the algorithms: *"Which algorithm has the best average performance on a given problem class, among those under investigation?"*. This question is tackled using statistical inference tools, so that information contained in a given sample can be used to draw conclusions about the general behaviour of the population.

Frequentist statistical inference involves the test of competing hypotheses and the estimation of certain populational parameters (both in terms of point estimates and statistical intervals) from the available data. [61].

A.2 POPULATIONAL PARAMETERS

Point estimates are single values expected to serve as some populational parameter, such as the mean, or the median. However, given the inherently random nature of inferential procedures, point estimates are likely not to represent the actual value of the parameter, but instead aim at providing a best estimate conditional on the available information. For a given parameter θ, the usual notation for its point estimator is $\hat{\Theta}$, and a specific value of this point estimator is a point estimate, $\hat{\theta}$ [54].

To quantify the uncertainty associated with a given estimation result, statistical intervals provide a range within which the true value of the parameter being estimated is likely to exist. They represent the *accuracy* of the parameter estimates. Two measures of this uncertainty are of particular interest to this study: *standard errors*, which represent the square root of the variance of the sampling distribution of a given test statistic (e.g., the mean); and *closed confidence intervals*, which provide a finite range (a, b) that contains the true value of the parameter of interest with a quantifiable degree of confidence.

For the sample mean, for instance, the standard error is given as $se_{\bar{X}} = \sigma/\sqrt{n}$, and can be generally interpreted as analogous to a "measurement error" of the parameter being estimated, in this case the true mean μ. Since in most cases the populational standard deviation is not known, it must be estimated from the data, which results in the calculation of the *sample standard error*,

$$\widehat{se}_{\bar{X}} = s/\sqrt{n},$$

Differently, the confidence interval is associated with a *confidence level* used in *hypothesis testing*. The calculation of this interval for the mean, for a confidence level of 0.95, for example, can be done by

$$\left(\bar{X} - 1.96\frac{s}{\sqrt{n}}, \bar{X} + 1.96\frac{s}{\sqrt{n}}\right)$$

A.3 HYPOTHESIS TESTING

A hypothesis testing consists in deciding between two competing statements, the first one, called the null hypothesis, sets a value (μ_0) representing the best current estimate for the parameter. Null hypotheses are commonly defined using one of the following formulations:

$$H_0 : \mu = \mu_0 \tag{30}$$

for the two sided case, or

$$H_0 : \mu \geq \mu_0, \quad H_0 : \mu \leq \mu_0 \quad \text{or} \tag{31}$$

for the directional case, which are contrasted against a second statement, called the alternative hypothesis. This hypotheses represents the presence of some deviation from the explanation proposed in the null hypothesis. It will be denoted H_1, and expresses the complementary explanation to the definition under H_0:

$$H_1 : \mu \neq \mu_0 \tag{32}$$

the two-sided alternative hypothesis, or

$$H_1 : \mu < \mu_0, \quad H_1 : \mu > \mu_0 \quad \text{or} \tag{33}$$

in the directional case. If the null hypothesis turns out to be true, it is said that the null hypothesis was accepted, however if it was not, it is rejected.

A.3.1 Statistical errors and effect size

In the context of hypothesis testing, a *false positive* (also known as a type-I error) occurs whenever a true null hypothesis is rejected, and a *false negative* (or type-II error) when a false null hypothesis is not rejected. Type-I errors can usually be easily controlled, as they depend only on the value of the parameter of interest under the null hypothesis, which is known to the experimenter. The probability of a false positive is usually denoted as α, and referred to as the *significance level* of a test. The complement of α, i.e., $1 - \alpha$, is called the *confidence level* of the test. In the context of this work, the control of Type-I errors is more complex when dealing with multiple comparisons, which occurs in the scope of this thesis. Different error rates have been used for multiple testing, such as the *error rate per family*, the *error rate per hypothesis*, and the *familywise error rate*. The later one, used in this work, is defined as the probability of at least one error happening on the family of tests, making a set of comparisons whose errors must be jointly controlled [65].

Type-II errors, on the other hand, are dependent not only on the null hypothesis, but also on the magnitude of the difference between the actual value of the parameter being tested and the value under H_0, which we will call the *simple effect size*, δ, which can be written as:

$$\delta = \mu - \mu_0. \tag{34}$$

The probability of a false negative result in a test of hypotheses is generally denoted β, and its complement $\pi = 1 - \beta$ is usually known as the *power* of the test, which is a measure of its *sensitivity* in detecting effects of a certain magnitude.

Since the power of a test is conditional to the (unknown) effect size, its control is generally harder than the confidence level. To circumvent this limitation it is important, when designing an experiment, to define what is called a *minimally relevant effect size* (MRES) or the *smallest practically significant effect size*, which is given by:

$$d^* = |\delta^*|/\sigma \tag{35}$$

And is defined as the smallest value the ratio d can achieve that the experimenter is interested in detecting. Where

$$d = |\delta|/\sigma \tag{36}$$

is the *standardized effect size* or the *Cohen's d* coefficient.

Power calculations, which include the determination of the required sample sizes, can then be performed with this value in mind. The resulting study will still have its own actual power, which may be greater than the nominal (if $d > d^*$), or smaller (if $d < d^*$). In the latter case, however, there will be no harm in running an underpowered study, given that any effects smaller than δ^* will be, by definition, below the threshold of practical relevance.

A.3.2 Parameter estimation and accuracy

One of the most common uses of statistics is parameter estimation, i.e., the use of information contained in a finite sample to estimate, with a certain accuracy, the value of a given parameter. For any parameter θ, the usual notation for its point estimator is $\widehat{\Theta}$, and a specific value of this point estimator is a point estimate, $\widehat{\theta}$ [54]. Two common examples of point estimators, which have their own specific notations, are the sample mean and the sample standard deviation,

$$\bar{X} = \widehat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{37}$$

$$S = \widehat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}, \tag{38}$$

where $\bar{x}$ is a realization of $\bar{X}$.

While point estimators return the value of greatest likelihood for a parameter given a sample, their values are also subject to uncertainties due to the randomness of their inputs. More specifically, a point estimator $\widehat{\Theta}$ has a given *sampling distribution* [54], which is a function of populational parameters and the sample size used in its calculation. The sample mean, for instance, has a distribution $P(\bar{X})$ with mean $E[\bar{X}] = E[X] = \mu$ and variance $V[\bar{X}] = V[X]/n = \sigma^2/n$.

Given these aspects of parameter estimation, an important point to consider is the *accuracy* of parameter estimates. A simple way of measuring this accuracy is using the *standard error* $se_{\widehat{\theta}}$, which represents the standard deviation of the sampling distribution of the estimator [54]. For the sample mean, for instance, the standard error is given as $se_{\bar{X}} = \sigma/\sqrt{n}$, and can be generally interpreted as analogous to a "measurement error" of the parameter being estimated, in this case the true mean μ. Since in most cases the populational standard deviation is not known, it must be estimated from the data, which results in the calculation of the *sample standard error*,

$$\widehat{se}_{\bar{X}} = s/\sqrt{n}, \tag{39}$$

Notice that it is straightforward to solve for n in (39), which allows us to predefine a desired level of accuracy (i.e., a desired upper limit for $\widehat{se}_{\bar{X}}$) for the estimation and calculate the required sample size to obtain it. Since we need some data to estimate s in the first place, an iterative approach can be used for this calculation, as will be presented in Section 3.1.

A.3.3 Parametric vs. Non parametric methods

Amongst the statistical tests, the parametric ones are those people are more familiar with, such as the t-test or one way ANOVA. They assume the data follows a specific distribution in order to test the hypothesis, therefore this data need to meet some assumptions regarding its ability to fit the model distribution. If the data does not meet these assumptions, it means that the distribution used is not good to draw any conclusion about this data.

When it is not possible to meet the assumptions of the parametric methods with the data, non parametric alternatives are to be used. Unlike the parametric ones, they do not follow any specific distribution, however they tend to have a smaller statistical power and have their own assumptions to meet.

B CAISER: PACKAGE USAGE

B.1 THE PACKAGE

The CAISEr package can be obtained from CRAN, `https://CRAN.R-project.org/package=CAISEr`, or using the command *install.packages("CAISEr")*.

It offers functions to perform experimental comparisons of algorithms with adequate sample sizes for power and accuracy. However, while in this version of the package only fixed sample(FSS) size experiments can be done, on the "under development" version, it is possible to perform both the FSS experimentation and the sequential one.

They are calculated by the routines *run_experiment* and *run_sequential_experiment* respectively. Both have the same inputs and outputs:

Required inputs

- Instance.list(list) - a list containing the instance functions.
- Algorithm.list(list) - a list containing the algorithm functions.
- power(numeric) - desired power.
- d(numeric) - MRES that detects differences greater d standard deviations.
- se.max(numeric) - desired maximum standard error
- dif(character) (**options:** *"simple", "perc"*) - choice of within instances difference if it is either simple or percentage.

Optional inputs

- sig.level(numeric) (**default:** *0.05*) - significance level for the confidence interval.

- direction(character) (**default:** *"two.sided"* **options:** *"one.sided", "two.sided"*) - type of H1. (**only for the FSS**)
- test.type(character) (**default:** *"t.test"* **options:** *"t.test", "wilcoxon", "binomial"*) - type of test. (**only for the FSS**)
- method(character) (**default:** *"param"* **options:** *"param", "boot"*) - estimation of parameters using a parametric method or bootstrap.
- nstart(integer) (**default:** *10*) - initial number of samples.
- nmax(integer) (**default:** *1000*) - maximum allowed sample size.
- seed(numeric) (**default:** *NULL*) - seed for PRNG.
- boot.R(integer) (**default:** *999*) - number of bootstrap resamples.
- force.balanced(logic) (**default:** *FALSE*) - force balanced sampling.

Output

The function returns a *list* containing:

- $Configuration - the inputed configuration
- $data.raw - a data frame with all observations generated
- $data.summary - a data frame containing the means, standard errors and sample sizes of each algorithm on each problem instance.
- $N - the number of instances used
- $N.star - the number of instances needed considering a FSS approach
- $instances.sampled - list with instances used
- $Underpowered - logic indicator if the experiment is underpowered.

B.2 EXAMPLE

An example of its fixed sample size usage and how to adapt functions to run on this method can be seen in https://cran.r-project.org/web/packages/CAISEr/vignettes/Adapting_Algorithm_for_CAISEr.html.

The execution of a sequential experiment is very similar, and is exemplified by the following code:

```
################################################################################
################################################################################
######################### Testing chase.r with MOEADr ##########################
################################################################################
################################################################################
library(MOEADr)
library(smoof)

#Generating Instances
### Build function names (instances: UF1 - UF7, dimensions 10 - 40)
fname   <- paste0("UF_", 1:7)
dims    <- c(10:40)
allfuns <- expand.grid(fname, dims, stringsAsFactors = FALSE)
# Assemble instances list
Instance.list <- vector(nrow(allfuns), mode = "list")
for (i in 1:length(Instance.list)){
    Instance.list[[i]]$FUN <- paste0(allfuns[i,1], "_", allfuns[i,2])
}
### Build the functions listed in Instance.list
# (so that they can be properly used)
for (i in 1:nrow(allfuns)){
    assign(x          = Instance.list[[i]]$FUN,
           value      = MOEADr::make_vectorized_smoof(prob.name  = "UF",
           dimensions = allfuns[i, 2],
           id         = as.numeric(strsplit(allfuns[i, 1], "_")[[1]][2])))
}

#Defining algorithms
## 1. MOEA/D-DE
moead.99 <- function(type, instance){
    # Input parameters:
    #    - type (variant to use: "original" or "moead.de")
    #     – instance (instance to be solved, e.g., instance = Instance.list[[i]])
    # All other parameters are set internally

    ## Extract instance information to build the MOEADr problem format
    fdef  ← unlist(strsplit(instance$FUN, split = "_"))
    uffun ← smoof::makeUFFunction(dimensions = as.numeric(fdef[3]),
                                  id         = as.numeric(fdef[2]))
    fattr    ← attr(uffun, "par.set")
    prob.dim ← fattr$pars$x$len

    ## Build MOEADr problem list
    problem ← list(name = instance$FUN,
                   xmin = fattr$pars$x$lower,
                   xmax = fattr$pars$x$upper,
                   m    = attr(uffun, "n.objectives"))

    ## Load presets for the algorithm provided in input 'type' and
    ## modify whatever is needed for this particular experiment
    algo.preset ← MOEADr::preset_moead(type)
    algo.preset$decomp$H ← 99 # ← set population size
    algo.preset$stopcrit[[1]]$name ← "maxeval" # ← type of stop criterion
    algo.preset$stopcrit[[1]]$maxeval ← 2000 * prob.dim # stop crit.
    poly.ind ← which(sapply(algo.preset$variation,function(x){x$name == "polymut"}))
    algo.preset$variation[[poly.ind]]$pm ← 1 / prob.dim # ←— pm = 1/d

    ## Run algorithm on "instance"
    out ← MOEADr::moead(preset   = algo.preset,
                        problem  = problem,
                        showpars = list(show.iters = "none"))

    ## Read reference data to calculate the IGD
    Yref  ← as.matrix(read.table(pasteo("../data/pf_data/",fdef[1], fdef[2], ".dat")))
    IGD = MOEADr::calcIGD(Y = out$Y, Yref = Yref)

    ## Return IGD as field "value" in the output list
    return(list(value = IGD))
}
## 2. MOEA/D-DE
moead.104 ← function(type, instance){
    # Input parameters:
    #     – type (variant to use: "original" or "moead.de")
    #     – instance (instance to be solved, e.g., instance = Instance.list[[i]])
    # All other parameters are set internally

    ## Extract instance information to build the MOEADr problem format
    fdef  ← unlist(strsplit(instance$FUN, split = "_"))
    uffun ← smoof::makeUFFunction(dimensions = as.numeric(fdef[3]),
                                   id         = as.numeric(fdef[2]))
    fattr    ← attr(uffun, "par.set")
    prob.dim ← fattr$pars$x$len

    ## Build MOEADr problem list
    problem ← list(name = instance$FUN,
                   xmin = fattr$pars$x$lower,
                   xmax = fattr$pars$x$upper,
                   m    = attr(uffun, "n.objectives"))

    ## Load presets for the algorithm provided in input 'type' and
    ## modify whatever is needed for this particular experiment
    algo.preset ← MOEADr::preset_moead(type)
    algo.preset$decomp$H ← 104 # ← set population size
    algo.preset$stopcrit[[1]]$name ← "maxeval" # ← type of stop criterion
    algo.preset$stopcrit[[1]]$maxeval ← 2000 * prob.dim # stop crit.
    poly.ind ← which(sapply(algo.preset$variation,function(x){x$name == "polymut"}))
    algo.preset$variation[[poly.ind]]$pm ← 1 / prob.dim # ←— pm = 1/d

    ## Run algorithm on "instance"
    out ← MOEADr::moead(preset   = algo.preset,
                        problem  = problem,
                        showpars = list(show.iters = "none"))

    ## Read reference data to calculate the IGD
    Yref  ← as.matrix(read.table(paste0("../data/pf_data/",fdef[1], fdef[2], ".dat")))
    IGD = MOEADr::calcIGD(Y = out$Y, Yref = Yref)

    ## Return IGD as field "value" in the output list
    return(list(value = IGD))
}
algo.d.de  ← list(list(FUN   = "moead.104",
                       alias = "MOEAD-DE-104",
                       type  = "moead.de"),
                  list(FUN   = "moead.99",
                         alias  = "MOEAD-DE-99",
                         type   = "moead.de"))

out <- run_sequential_experiment(Instance.list  = Instance.list,
                                 Algorithm.list = algo.d.de,
                                 power          = 0.9,
                                 d              = 0.5,
                                 se.max         = 0.1,
                                 dif            ="perc")
```

References

BIBLIOGRAPHY

[1] M. Z. Ali et al. "Leveraged Neighborhood Restructuring in Cultural Algorithms for Solving Real-World Numerical Optimization Problems". In: *IEEE Transactions on Evolutionary Computation* 20.2 (Apr. 2016), pp. 218–231. ISSN: 1089-778X. DOI: 10.1109/TEVC.2015.2450018.

[2] Ignacio G. del Amo et al. "An algorithm comparison for dynamic optimization problems". In: *Applied Soft Computing* 12.10 (Oct. 2012), pp. 3176–3192. DOI: 10.1016/j.asoc.2012.05.021. URL: http://dx.doi.org/10.1016/j.asoc.2012.05.021.

[3] Richard S. Barr et al. "Designing and reporting on computational experiments with heuristic methods". In: *Journal of Heuristics* 1.1 (Sept. 1995), pp. 9–32. DOI: 10.1007/bf02430363. URL: http://dx.doi.org/10.1007/BF02430363.

[4] Jay Bartroff, Tze Leung Lai, and Mei-Chiung Shih. *Sequential Experimentation in Clinical Trials*. Springer New York, 2013. DOI: 10.1007/978-1-4614-6114-2. URL: https://doi.org/10.1007/978-1-4614-6114-2.

[5] Thomas Bartz-Beielstein. *Experimental Research in Evolutionary Computation*. Springer, 2006. 232 Seiten. ISBN: 3540320261. URL: http://www.ebook.de/de/product/5329328/thomas_bartz_beielstein_experimental_research_in_evolutionary_computation.html.

[6] Thomas Bartz-Beielstein. *Meaningful Problem Instances and Generalizable Results*. Jan. 2015.

[7] Leonardo C. T. Bezerra, Manuel Lopez-Ibanez, and Thomas Stutzle. "Automatic Component-Wise Design of Multiobjective Evolutionary Algorithms". In: *IEEE Transactions on Evolutionary Computation* 20.3 (June 2016), pp. 403–417. DOI: 10.1109/tevc.2015.2474158. URL: https://doi.org/10.1109%2Ftevc.2015.2474158.

[8] Mauro Birattari. *On the estimation of the expected performance of a metaheuristic on a class of instances. How many instances, how many runs?* Tech. rep. Brussels, Belgium: Université Libre de Bruxelles, 2004.

[9] Juan Botella et al. "Optimization of sample size in controlled experiments: The CLAST rule". In: *Behavior Research Methods* 38.1 (Feb. 2006), pp. 65–76. DOI: 10.3758/bf03192751. URL: https://doi.org/10.3758/bf03192751.

[10] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. "A survey on optimization metaheuristics". In: *Information Sciences* 237 (July 2013), pp. 82–117. DOI: 10.1016/j.ins.2013.02.041. URL: http://dx.doi.org/10.1016/j.ins.2013.02.041.

[11] R.J. Tibshirani Bradley Efron. *An Introduction to the Bootstrap*. 1st ed. Chapman and Hall, 1994.

[12] F. Campelo, Lucas S. Batista, and Claus Aranha. "The MOEADr Package – A Component-Based Framework for Multiobjective Evolutionary Algorithms Based on Decomposition". In: *Submitted: Journal of Statistical Software* (2017).

[13] Felipe Campelo and Claus Aranha. *MOEADr: Component-Wise MOEA/D Implementation*. R package version 0.2.1. 2017. URL: https://CRAN.R-project.org/package=MOEADr.

[14] Felipe Campelo and Fernanda Takahashi. *CAISEr: Comparison of Algorithms with Iterative Sample Size Estimation*. 2017. URL: https://CRAN.R-project.org/package=CAISEr.

[15] Eduardo G. Carrano, Elizabeth F. Wanner, and Ricardo H. C. Takahashi. "A Multicriteria Statistical Based Comparison Methodology for Evaluating Evolutionary Algorithms". In: *IEEE Transactions on Evolutionary Computation* 15.6 (Dec. 2011), pp. 848–870. DOI: 10.1109/tevc.2010.2069567. URL: http://dx.doi.org/10.1109/TEVC.2010.2069567.

[16] Marie Coffin and Matthew J. Saltzman. "Statistical Analysis of Computational Tests of Algorithms and Heuristics". In: *INFORMS Journal on Computing* 12.1 (Feb. 2000), pp. 24–44. DOI: 10.1287/ijoc.12.1.24.11899. URL: http://dx.doi.org/10.1287/ijoc.12.1.24.11899.

[17] M.J. Crawley. *The R Book*. 2nd. Wiley, 2013.

[18] A. C. Davison and D. V. Hinkley. *Bootstrap methods and their application*. Cambridge University Press, 1997.

[19] Janez Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *Journal of Machine Learning Research* 7 (Dec. 2006), pp. 1–30. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1248547.1248548.

[20] J. Derrac et al. "Statistical analysis of convergence performance throughout the evolutionary search: A case study with SaDE-MMTS and Sa-EPSDE-MMTS". In: *2013 IEEE Symposium on Differential Evolution (SDE)*. Apr. 2013, pp. 151–156. DOI: 10.1109/SDE.2013.6601455.

[21] Joaquín Derrac et al. "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms". In: *Swarm and Evolutionary Computation* 1.1 (Mar. 2011), pp. 3–18. DOI: 10.1016/j.swevo.2011.02.002. URL: http://dx.doi.org/10.1016/j.swevo.2011.02.002.

[22] Bryony DuPont and Jonathan Cagan. "A hybrid extended pattern search/genetic algorithm for multi-stage wind farm optimization". In: *Optimization and Engineering* 17.1 (Jan. 2016), pp. 77–103. DOI: 10.1007/s11081-016-9308-3. URL: http://dx.doi.org/10.1007/s11081-016-9308-3.

[23] A.E. Eiben and M. Jelasity. "A critical note on experimental research methodology in EC". In: *Proceedings of the 2002 IEEECongress on Evolutionary Computation*. Institute of Electrical & Electronics Engineers (IEEE), 2002. DOI: 10.1109/cec.2002.1006991. URL: http://dx.doi.org/10.1109/CEC.2002.1006991.

[24] José M. Ferrer, M. Teresa Ortuño, and Gregorio Tirado. "A GRASP metaheuristic for humanitarian aid distribution". In: *Journal of Heuristics* 22.1 (Oct. 2015), pp. 55–87. DOI: 10.1007/s10732-015-9302-5. URL: http://dx.doi.org/10.1007/s10732-015-9302-5.

[25] E. C. Fieller. "Some Problems in Interval Estimation". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 16.2 (1954), pp. 175–185. ISSN: 00359246. URL: http://www.jstor.org/stable/2984043.

[26] Martina Fischetti and Michele Monaci. "Proximity search heuristics for wind farm optimal layout". In: *Journal of Heuristics* 22.4 (Feb. 2015), pp. 459–474. DOI: 10.1007/s10732-015-9283-4. URL: http://dx.doi.org/10.1007/s10732-015-9283-4.

[27] Volker Franz. *Ratios: A short guide to confidence limits and proper use*. 2007.

[28] S. García et al. "A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability". In: *Soft Computing* 13.10 (Dec. 2009), pp. 959–977. DOI: 10.1007/s00500-008-0392-y. URL: http://dx.doi.org/10.1007/s00500-008-0392-y.

[29] Salvador García et al. "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization". In: *Journal of Heuristics* 15.6 (May 2008), pp. 617–644. DOI: 10.1007/s10732-008-9080-4. URL: http://dx.doi.org/10.1007/s10732-008-9080-4.

[30] Salvador García et al. "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power". In: *Information Sciences* 180.10 (May 2010), pp. 2044–2064. DOI: 10.1016/j.ins.2009.12.010. URL: http://dx.doi.org/10.1016/j.ins.2009.12.010.

[31] Nikolaus Hansen et al. "COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting". In: *CoRR* abs/1603.08785 (2016). URL: http://arxiv.org/abs/1603.08785.

[32] D. A. Harrison and A. R. Brady. "Sample size and power calculations using the non-central t-distribution". In: *Stata Journal* 4.2 (2004), 142–153(12). URL: http://www.stata-journal.com/article.html?article=st0062.

[33] Megan L. Head et al. "The Extent and Consequences of P-Hacking in Science". In: *PLOS Biology* 13.3 (Mar. 2015), pp. 1–15. DOI: 10.1371/journal.pbio.1002106. URL: https://doi.org/10.1371/journal.pbio.1002106.

[34] J. N. Hooker. "Toward unification of exact and heuristic optimization methods". In: *International Transactions in Operational Research* 22.1 (May 2013), pp. 19–48. DOI: 10.1111/itor.12020. URL: http://dx.doi.org/10.1111/itor.12020.

[35] John N Hooker. "Needed: An empirical science of algorithms". In: *Operations Research* 42.2 (1994), pp. 201–212.

[36] John N Hooker. "Testing heuristics: We have it all wrong." In: *Journal of Heuristics* 1.1 (1996), pp. 33–42.

[37] Stuart H. Hurlbert. "Pseudoreplication and the Design of Ecological Field Experiments". In: *Ecological Monographs* 54.2 (Feb. 1984), pp. 187–211. DOI: 10.2307/1942661. URL: https://doi.org/10.2307%2F1942661.

[38] R. K. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons Ltd, 1991. 720 Seiten. ISBN: 0471503363. URL: http://www.ebook.de/de/product/4415910/r_k_jain_the_art_of_computer_systems_performance_analysis.html.

[39] Donald R. Jones, Matthias Schonlau, and William J. Welch. "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492.

[40] Ahmed Kattan et al. "GP made faster with semantic surrogate modelling". In: *Information Sciences* 355 (2016), pp. 169–185.

[41] Renato A. Krohling, Rodolfo Lourenzutti, and Mauro Campos. "Ranking and comparing evolutionary algorithms with Hellinger-TOPSIS". In: *Applied Soft Computing* 37 (Dec. 2015), pp. 217–226. DOI: 10.1016/j.asoc.2015.08.012. URL: http://dx.doi.org/10.1016/j.asoc.2015.08.012.

[42] Manuel Laguna and Rafael Martí. "Heuristics". In: *Encyclopedia of operations research and management science*. Ed. by Saul I. Gass and Michael C. Fu. New York London: Springer, 2013, pp. 695 –703. ISBN: 978-1-4419-1154-4.

[43] S. E. Lazic. "The problem of pseudoreplication in neuroscientific studies: is it affecting your analysis?" In: *BMC Neuroscience* 11.5 (2010), pp. 397–407. DOI: 10.1186/1471-2202-11-5.

[44] Hui Li and Qingfu Zhang. "Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 13.2 (Apr. 2009), pp. 284–302. DOI: 10.1109/tevc.2008.925798. URL: https://doi.org/10.1109%2Ftevc.2008.925798.

[45] Y. C. Lin, M. Clauß, and M. Middendorf. "Simple Probabilistic Population-Based Optimization". In: *IEEE Transactions on Evolutionary Computation* 20.2 (Apr. 2016), pp. 245–262. ISSN: 1089-778X. DOI: 10.1109/TEVC.2015.2451701.

[46] Jiaxiang Luo and Lixin Tang. "A hybrid approach of ordinal optimization and iterated local search for manufacturing cell formation". In: *The International Journal of Advanced Manufacturing Technology* 40.3-4 (Jan. 2008), pp. 362–372. DOI: 10.1007/s00170-007-1346-8. URL: http://dx.doi.org/10.1007/s00170-007-1346-8.

[47] Ulrike Von Luxburg and Volker H. Franz. "A geometric approach to confidence sets for ratios: Fieller's theorem, generalizations, and bootstrap". In: *Statistica Sinica* (2009).

[48] X. Ma et al. "A Multiobjective Evolutionary Algorithm Based on Decision Variable Analyses for Multiobjective Optimization Problems With Large-Scale Variables". In: *IEEE*

Transactions on Evolutionary Computation 20.2 (Apr. 2016), pp. 275–298. ISSN: 1089-778X. DOI: 10.1109/TEVC.2015.2455812.

[49] André L. Maravilha, Letícia M. Pereira, and Felipe Campelo. *Statistical characterization of neighborhood structures for the unrelated parallel machine problem with sequence-dependent setup times*. In preparation.

[50] Dietmar G. Maringer. *Portfolio Management with Heuristic Optimization*. Springer US, 2006. URL: http://www.ebook.de/de/product/11430917/dietmar_g_maringer_portfolio_management_with_heuristic_optimization.html.

[51] Paul Mathews. *Sample Size Calculations: Practical Methods for Engineers and Scientists*. Mathews Malnar& Bailey Inc., 2010. 338 Seiten. ISBN: 0615324614. URL: http://www.ebook.de/de/product/11350637/paul_mathews_sample_size_calculations_practical_methods_for_engineers_and_scientists.html.

[52] Scott E. Maxwell, Ken Kelley, and Joseph R. Rausch. “Sample Size Planning for Statistical Power and Accuracy in Parameter Estimation”. In: *Annual Review of Psychology* 59.1 (Jan. 2008), pp. 537–563. DOI: 10.1146/annurev.psych.59.103006.093735. URL: http://dx.doi.org/10.1146/annurev.psych.59.103006.093735.

[53] Catherine C. McGeoch. “Feature Article—Toward an Experimental Method for Algorithm Simulation”. In: *INFORMS Journal on Computing* 8.1 (Feb. 1996), pp. 1–15. DOI: 10.1287/ijoc.8.1.1. URL: http://dx.doi.org/10.1287/ijoc.8.1.1.

[54] Douglas C. Montgomery and George C. Runger. *Applied Statistics and Probability for Engineers*. 6th. Wiley, 2013.

[55] J. Toby Mordkoff. *The Assumption(s) of Normality*. 2011. URL: http://www2.psychology.uiowa.edu/faculty/mordkoff/GradStats/part%201/I.07%20normal.pdf.

[56] J. Mukund Nilakantan et al. “Bio-inspired search algorithms to solve robotic assembly line balancing problems”. In: *Neural Computing and Applications* 26.6 (Jan. 2015), pp. 1379–1393. DOI: 10.1007/s00521-014-1811-x. URL: http://dx.doi.org/10.1007/s00521-014-1811-x.

[57] Gottfried E. Noether. “Sample Size Determination for Some Common Nonparametric Tests”. In: *Journal of the American Statistical Association* 82.398 (June 1987), pp. 645–647. DOI: 10.1080/01621459.1987.10478478. URL: http://dx.doi.org/10.1080/01621459.1987.10478478.

[58] Mônica S Pais et al. “Factorial design analysis applied to the performance of parallel evolutionary algorithms”. In: *Journal of the Brazilian Computer Society* 20.1 (2014), p. 6. DOI: 10.1186/1678-4804-20-6. URL: http://dx.doi.org/10.1186/1678-4804-20-6.

[59] X. Qiu et al. “Adaptive Cross-Generation Differential Evolution Operators for Multiobjective Optimization”. In: *IEEE Transactions on Evolutionary Computation* 20.2 (Apr. 2016), pp. 232–244. ISSN: 1089-778X. DOI: 10.1109/TEVC.2015.2433672.

[60] Herbert Robbins and Sutton Monro. “A Stochastic Approximation Method”. In: *Annals Mathematical Statistics* 22.3 (Sept. 1951), pp. 400–407. DOI: 10.1214/aoms/1177729586. URL: http://dx.doi.org/10.1214/aoms/1177729586.

[61] George G. Roussas. *An Introduction to Probability and Statistical Inference, Second Edition*. 2nd ed. Academic Press, 2014. ISBN: 0128001143,9780128001141.

[62] Haroldo G. Santos et al. “Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem”. In: *International Transactions in Operational Research* 00 (June 2016). URL: https://doi.org/10.1111/itor.12316.

[63] Shlomo S. Sawilowsky. “New Effect Size Rules of Thumb”. In: *Journal of Modern Applied Statistical Methods* 8.2 (2009), pp. 597–599.

[64] Marc Sevaux et al. “GRASP with ejection chains for the dynamic memory allocation in embedded systems”. In: *Soft Computing* 18.8 (Oct. 2013), pp. 1515–1527. DOI: 10.1007/s00500-013-1157-9. URL: http://dx.doi.org/10.1007/s00500-013-1157-9.

[65] Juliet Popper Shaffer. “Multiple Hypothesis Testing”. In: *Annual Review of Psychology* 46.1 (1995), pp. 561–584. DOI: 10.1146/annurev.ps.46.020195.003021. eprint: https://doi.org/10.1146/annurev.ps.46.020195.003021. URL: https://doi.org/10.1146/annurev.ps.46.020195.003021.

[66] David J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Taylor & Francis Ltd, 2011. 1926 Seiten. ISBN: 1439858012. URL: http://www.ebook.de/de/product/19785465/david_j_sheskin_handbook_of_parametric_and_nonparametric_statistical_procedures.html.

[67] A. Sinha et al. “Solving Bilevel Multicriterion Optimization Problems With Lower Level Decision Uncertainty”. In: *IEEE Transactions on Evolutionary Computation* 20.2 (Apr. 2016), pp. 199–217. ISSN: 1089-778X. DOI: 10.1109/TEVC.2015.2443057.

[68] María Soto, André Rossi, and Marc Sevaux. “A multiple neighborhood search for dynamic memory allocation in embedded systems”. In: *Journal of Heuristics* 21.6 (July 2015), pp. 719–749. DOI: 10.1007/s10732-015-9297-y. URL: http://dx.doi.org/10.1007/s10732-015-9297-y.

[69] Nils Egil Søvde et al. “A semi-greedy metaheuristic for the European cableway location problem”. In: *Journal of Heuristics* 21.5 (June 2015), pp. 641–662. DOI: 10.1007/s10732-015-9294-1. URL: http://dx.doi.org/10.1007/s10732-015-9294-1.

[70] Kenneth Sörensen. “Metaheuristics-the metaphor exposed”. In: *International Transactions in Operational Research* 22.1 (Feb. 2015), pp. 3–18. DOI: 10.1111/itor.12001. URL: http://dx.doi.org/10.1111/itor.12001.

[71] Kenneth Sörensen and Fred Glover. “Metaheuristics”. In: *Encyclopedia of operations research and management science*. Ed. by Saul I. Gass and Michael C. Fu. New York London: Springer, 2013, pp. 960 –970. ISBN: 978-1-4419-1154-4.

[72] KP Suresh and S Chandrashekara. “Sample size estimation and power analysis for clinical research studies”. In: *Journal of Human Reproductive Sciences* 5.1 (2012), p. 7. DOI: 10.4103/0974-1208.97779. URL: https://doi.org/10.4103%2F0974-1208.97779.

[73] Y. Tenne and C.-K. Goh. *Computational Intelligence in Expensive Optimization Problems*. Springer, 2010.

[74] Patrizio E. Tressoldi et al. “High Impact = High Statistical Standards? Not Necessarily So”. In: *PLoS ONE* 8.2 (Feb. 2013). Ed. by Robert K. Hills, e56180. DOI: 10.1371/journal.pone.0056180. URL: http://dx.doi.org/10.1371/journal.pone.0056180.

[75] Eva Vallada and Rubén Ruiz. “A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times”. In: *European Journal of Operational Research* 211.3 (June 2011), pp. 612–622.

[76] Matej Črepinšek, Shih Hsi Liu, and Marjan Mernik. “Replication and comparison of computational experiments in applied evolutionary computing: Common pitfalls and guidelines to avoid them”. In: *Applied Soft Computing Journal* 19 (2014), pp. 161–170.

[77] Frank Wilcoxon. “Individual Comparisons by Ranking Methods”. In: *Biometrics Bulletin* 1.6 (Dec. 1945), p. 80. DOI: 10.2307/3001968. URL: http://dx.doi.org/10.2307/3001968.

[78] Bo Yuan and Marcus Gallagher. "An improved small-sample statistical test for comparing the success rates of evolutionary algorithms". In: *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO09*. Association for Computing Machinery (ACM), 2009. DOI: 10.1145/1569901.1570213. URL: http://dx.doi.org/10.1145/1569901.1570213.

[79] Y. Yuan et al. "Balancing Convergence and Diversity in Decomposition-Based Many-Objective Optimizers". In: *IEEE Transactions on Evolutionary Computation* 20.2 (Apr. 2016), pp. 180–198. ISSN: 1089-778X. DOI: 10.1109/TEVC.2015.2443001.

[80] Qingfu Zhang and Hui Li. "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition". In: *IEEE Transactions on Evolutionary Computation* 11.6 (Dec. 2007), pp. 712–731. DOI: 10.1109/tevc.2007.892759. URL: https://doi.org/10.1109%2Ftevc.2007.892759.

[81] C. Zhu, L. Xu, and E. D. Goodman. "Generalization of Pareto-Optimality for Many-Objective Evolutionary Optimization". In: *IEEE Transactions on Evolutionary Computation* 20.2 (Apr. 2016), pp. 299–315. ISSN: 1089-778X. DOI: 10.1109/TEVC.2015.2457245.

[82] Eckart Zitzler et al. "Performance assessment of multiobjective optimizers: An analysis and review". In: *IEEE Transactions on Evolutionary Computation* 7.2 (2003), pp. 117–132.