Few guidelines for analyzing methods

Éric D. Taillard*

*Department of Electrical & Computer Engineering, University of Applied Sciences of Western Switzerland
Route de Cheseaux 1, Case postale, CH-1401 Yverdon, Switzerland
eric.taillard@eivd.ch

1 Introduction

Designing a solving method that is either exact or heuristic or deterministic or stochastic is a multi-objective process. The main dimensions that are commonly considered are:

- The computational resources
- The problem instance space
- The quality of solutions obtained, in case of an optimization problem

All these dimensions are not necessarily simultaneously present. For instance, if we are dealing with a simple problem for which an exact polynomial algorithm is known, the problem instance space is often restricted to the positive integers, corresponding to the instance size. Indeed, in this case, we are mainly interested by the evolution of the computational effort with the problem size growth. Since the algorithm produces exact solutions, the "quality" dimension doesn't exist.

In this presentation, we first propose few guidelines on the way all these dimensions can be measured or treated. We then present few statistical techniques that can help the practitioner to evaluate confidence intervals for measures made when designing a new method or for comparing methods one another. Indeed, decisions must be taken while designing a new method. Typically, there are parameters to tune for a method based on metaheuristic principles. Often, the method is stochastic, meaning that its result depends on the run. The set of instances cannot be completely examined. So, the instances on which a method is tested must be chosen. This can also be viewed as an additional stochastic component. Therefore, the designer must take decisions in an uncertain context. Statistical tests are precisely a way to help decision-maker working under uncertainty. Finally we present few possibilities of a software developed in our Institute for comparing solving methods via a web interface.

2 Measuring computational resources

There are several computational resources that can be considered, typically the computational effort, the number of processors and the memory requirement. In this presentation, we restrict ourselves to sequential algorithms, so, we suppose that the number of processors is 1. Since measuring the computational effort and the memory requirement can be done in a similar way, we will only consider the computational effort in this section, to simplify.

It is possible to perform absolute or relative measures. An absolute computational effort can be obtained by counting the number of characteristic operations of an algorithm as a function of the problem size (n) or others of its characteristics. As it is difficult to count all the operations (and to express this under the form of a function $a(n)$), we are mainly interested in the order of magnitude of the number of operations. This is the reason of the introduction of the $O(\cdot)$ notation.

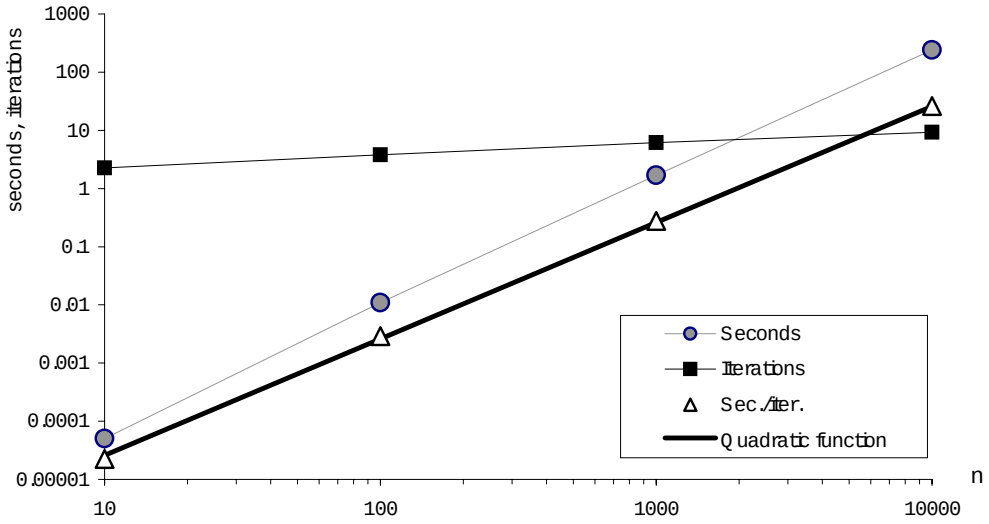

Figure 1: Evolution of 2-opt computational effort as a function of problem size for Euclidean TSP instances uniformly distributed in a unit square

Generally, but not always, it is easy to derive a function $f(n)$ that is an upper bound to the true value $a(n)$ of operations needed by an algorithm *for the worst instance* of size n. If there is a constant $c > 0$ such that $\lim_{n\to\infty} c \cdot f(n) \geq a(n)$ then it is said that $a(n)$ is in $O(f(n))$ or $a(n) \in O(f(n))$. When we have a lower bound (*best case*), the notation is $\Omega(\cdot)$. When we have both a lower and upper bound (*in any case*), the notation is $\Theta(\cdot)$, which is not an "average complexity", since this notion is not well defined. These notations allow to compare algorithms in an exact and theoretical way. Indeed if algorithm A is in $O(f(n))$ and algorithm B in $\Omega(g(n))$ and $g(n) \notin O(f(n))$, then it can be said that, asymptotically, A is faster than B. However, for the practitioner, these notions could be problematic. On the one hand, if both algorithm are in $\Theta(f(n))$, we have no idea about their relative efficiency. On the second hand, the lower and upper bound $\Omega(f(n))$ and $O(g(n))$ that can be theoretically derived could be far one another. For instance, it is easy to show that finding local optima to a travelling salesman problem using 2-opt or Or-opt neighbourhood is in $\Omega(n^2)$ and in $O(n!)$.

The gap between n^2 and $n!$ is huge and not easy to reduce.

The practitioner is certainly not interested to consider the worst theoretical case but is rather interested to limit himself to problem instances with a special structure. In this case, the practitioner may find an empirical function $h(n)$ that provides an *approximation* of the computational effort of the algorithm, for a *specified class* of problem instances. For instance, $h(n) = n^\alpha$, with α a positive constant, could be a good model. Figure 2 show the evolution of various measures of computational effort observed for a local search procedure based on 2-opt neighbourhood as a function of problem size, for Euclidean TSP instances uniformly distributed in a unit square. In this figure, it seems clear that the time per 2-opt iteration (i.e. examining once the entire neighbourhood; improving moves are immediately performed) grows quadratically, as expected theoretically. This figure shows that the number of times the entire neighbourhood is evaluated also grows, but very slowly. In [Taillard (2003)], we have proposed to use the notation $\hat{O}(h(n))$ to clearly show that $h(n)$ is an estimate of the complexity and not a theoretical, exact complexity. In the above example, the empirical complexity seems to be $\hat{O}(n^{2.22})$, which is far from the $O(n!)$ theoretical complexity.

We must stress here that it can be difficult to get a good estimate of the α constant in the above example. Indeed, the measures of computational times suffer from large imprecision for various reasons. Extrapolating computational times on another computer, for instance by using "Dongarra's factor" [Dongarra (2005)] can be even more imprecise. Therefore, we strongly recommend, when it is possible, to publish computational effort using an absolute measure (e.g. number of iterations of a local search (together with the theoretical complexity for performing one iteration), number of nodes evaluated, number of solutions evaluated, etc.) in addition to a relative measure like the computational time.

For a decision problem (solution found or not, in a large meaning: an optimization problem with a fixed objective value to reach can be considered as a decision problem), only the dimension "computational resources" can be considered. In this case, for comparing different methods, it is possible to consider the proportion of successes of the methods. Section 5.1 proposes a guideline to proceed to comparisons in this case.

3 Problem instances

We insist on complexity notions because we think that this dimension is often overwhelm in the metaheuristic community. Indeed, practitioners often focus on instance sets with relatively small variation in problem size (typically, there is a factor of less than 10^2 between the largest and the smallest instance). Focusing on such problem instance sets doesn't allow to analyze the factor of the problem size (The example given above in Figure 2 could be criticized, since the factor is only 10^3). There are however notorious exceptions, exemplified by travelling salesman problem sets, for which the factor is rather 10^5 or 10^6.

To analyze a method, it is a good practice to deal with instances of very different size, but one has to refrain providing statistics that are too aggregated and that can hide some effects (for instance, increasing the number of small, easy instances can artificially improve the statistics). We recommend to *stratify* the problem instances (i.e. to group problem instances with the same structure, like the size or the generation procedure) in order to diminish the

variance of the observations (time, quality) so that more accurate results can be provided. The exceptionally small dispersion that can be observed in Figure 2 is due to the fact that the problem instances considered have all the same structure and 10^2 to 10^6 problem instances for each size have been considered.

4 Quality measure

In this section, we suppose that we have only one quality measure. So, in case of multi-objective optimization we suppose that one of the numerous metrics for measuring the efficiency of methods has been used. When the problem instances are finely stratified (especially regarding the values of the input data), it is possible to use the objective function as a quality measure. For instance it is possible to take the length of travelling salesman problems instances of a given size, randomly, uniformly generated in the unit square. Indeed, in this case, the variance of optimal tour length is very low for such instances. But, more frequently, it is required to "normalize" the data, for instance by measuring the quality in % of deviation from a reference value. The best reference value is the optimum but it is generally unknown. Practically, one chooses either a bound on this value or the value of the best feasible solution known. Both of these values are supposed to evolve. Therefore, it is necessary to carefully report on the reference value considered in order to avoid confusion. Measuring the deviation from a reference value cannot be done in case the objective can be 0. In this case, one has to project the objective in the interval $[0, 1]$ by taking two reference values corresponding to lower and upper bounds to the objective. Finally, let us mention that ranking various solutions is perhaps the simplest and safest way to normalize data. Indeed, statistics based on ranks are very robust, insensitive to outliers and do not require large number of observations. Proceeding like this, we must be aware that other information is available. We loose the information of how large is the quality gap between two solutions having adjacent ranks. This can lead to paradoxical situations. For instance, the mean of a sample can be lower than the mean of another sample, while it can be the reverse for the ranks.

5 Few statistical methods for comparing algorithms

In order to proceed to comparisons on statistical basis, we have to start by making hypothesis. So, we are going to make the following hypothesis:

- The practitioner is honest. The problem instances are not selected in a strange way; if computational times are considered, sufficiently large incertitude margins are taken so that the conclusions are not biased. A factor of 10 in computational times shouldn't be exceptional. The number of instances considered should be sufficient (generally, one can work comfortably with 20 observations; in case of the comparison of stochastic algorithms, one can multiply the number of runs if the problem set is restricted).
- The practitioner is not in immoderate love with statistics and has almost no idea about the distribution functions of the samples studied. So, a limited panel of statistical tests is convenient, provided that they are robust, easy to use and based on weak hypothesis.

Table 1: A contingency table counting successes and failures of two methods

	Success	Failure	Total
Method A	a	$n-a$	n
Method B	b	$m-b$	m

In this section, we are going to propose a "rescue toolkit" for such a practitioner, supposed to be randomly chosen in the metaheuristic community. We will not recall standard tests that can be found in every college's level book, that are based on very strong assumptions such as Gaussian distribution of the samples, assumptions that are often violated.

5.1 Comparing two methods with a fixed goal

We suppose here that we want to compare the success rates of two methods. These methods can be stochastic (in this case, a comparison on a single instance can be done, provided that such an information has a meaning!) or deterministic (in this case, the problem instances are considered to be randomly chosen). If both methods are iterative, we suppose that the comparisons are repeated for different computational efforts. In this case a 2×2 contingency table can be built (see Table 1).

In order to test whether both methods are behaving similarly or not, a test known under the name of "Fisher's exact test" can be conducted. This test can be viewed as a *permutation test* [Good 2005]. We recently developed a similar test [Taillard et al. (2004)]. This test is more general and more powerful than the sign test [Arbuthnott(1710)] for comparing proportions, also known as McNemar test.

The article of [Taillard et al. (2004)] is interesting for the practitioner since it tabulates a, b, n and m values for which success rates of a/n are significantly higher than success rates of b/m, for significance levels of 5% and 1%. So, the practitioner can compare two methods just by counting, without further computations. Below is the table published in this reference, for significance level of 1% (or confidence level of 99%) for small samples. We see that 7 observations may be sufficient in the best case to reach the 1% significance level.

There are numerous statistical tests (based on Chi-Square distribution, see e.g. [Conover (1999)]) that are able to directly treat $r \times c$ contingency tables (more than the two (Success, Failure) categories, more than 2 methods). However, the information given by such tests might be not so interesting for the practitioner (either at least one method behave differently from the others, but we don't know which one, or it cannot be excluded that all the methods behave identically).

5.2 Confidence interval

In this section, we suppose that the practitioner has executed an optimization algorithm n times and observed the solutions qualities $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. The observations are supposed to be independent, of finite variance, but there is no assumption about the distribution

m \ n	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2						(7,0)	(8,0)	(9,0)	(10,0)	(11,0)	(12,0)	(12,0)	(13,0)	(14,0)
3			(4,0)	(5,0)	(6,0)	(7,0)	(7,0)	(8,0)	(9,0)	(9,0)	(10,0)	(11,0)	(11,0)	(12,0)
											(12,1)	(13,1)	(14,1)	(15,1)
4		(3,0)	(4,0)	(5,0)	(5,0)	(6,0)	(6,0)	(7,0)	(8,0)	(8,0)	(9,0)	(9,0)	(10,0)	(11,0)
					(6,1)	(7,1)	(8,1)	(9,1)	(10,1)	(11,1)	(11,1)	(12,1)	(13,1)	(14,1)
5		(3,0)	(4,0)	(4,0)	(5,0)	(5,0)	(6,0)	(6,0)	(7,0)	(7,0)	(8,0)	(8,0)	(9,0)	(9,0)
				(5,1)	(6,1)	(7,1)	(7,1)	(8,1)	(9,1)	(10,1)	(10,1)	(11,1)	(12,1)	(12,1)
								(9,2)	(10,2)	(11,2)	(12,2)	(13,2)	(14,2)	(14,2)
6		(3,0)	(3,0)	(4,0)	(4,0)	(5,0)	(5,0)	(6,0)	(6,0)	(7,0)	(7,0)	(8,0)	(8,0)	(9,0)
			(4,1)	(5,1)	(6,1)	(6,1)	(7,1)	(8,1)	(8,1)	(9,1)	(9,1)	(10,1)	(11,1)	(11,1)
					(6,2)	(7,2)	(8,2)	(9,2)	(10,2)	(10,2)	(11,2)	(12,2)	(13,2)	(13,2)
										(11,3)	(12,3)	(13,3)	(14,3)	(15,3)
7	(2,0)	(3,0)	(3,0)	(4,0)	(4,0)	(5,0)	(5,0)	(5,0)	(6,0)	(6,0)	(7,0)	(7,0)	(7,0)	(8,0)
			(4,1)	(5,1)	(5,1)	(6,1)	(6,1)	(7,1)	(8,1)	(8,1)	(9,1)	(9,1)	(10,1)	(10,1)
				(5,2)	(6,2)	(7,2)	(8,2)	(8,2)	(9,2)	(10,2)	(10,2)	(11,2)	(12,2)	(12,2)
							(8,3)	(9,3)	(10,3)	(11,3)	(12,3)	(12,3)	(13,3)	(14,3)
												(13,4)	(14,4)	(15,4)
8	(2,0)	(3,0)	(3,0)	(4,0)	(4,0)	(4,0)	(5,0)	(5,0)	(5,0)	(6,0)	(6,0)	(6,0)	(7,0)	(7,0)
		(3,1)	(4,1)	(4,1)	(5,1)	(6,1)	(6,1)	(7,1)	(7,1)	(8,1)	(8,1)	(9,1)	(9,1)	(10,1)
			(4,2)	(5,2)	(6,2)	(6,2)	(7,2)	(8,2)	(8,2)	(9,2)	(10,2)	(10,2)	(11,2)	(12,2)
					(6,3)	(7,3)	(8,3)	(9,3)	(9,3)	(10,3)	(11,3)	(12,3)	(12,3)	(13,3)
								(9,4)	(10,4)	(11,4)	(12,4)	(13,4)	(14,4)	(14,4)
														(15,5)
9	(2,0)	(3,0)	(3,0)	(3,0)	(4,0)	(4,0)	(4,0)	(5,0)	(5,0)	(5,0)	(6,0)	(6,0)	(6,0)	(7,0)
		(3,1)	(4,1)	(4,1)	(5,1)	(5,1)	(6,1)	(6,1)	(7,1)	(7,1)	(8,1)	(8,1)	(9,1)	(9,1)
			(4,2)	(5,2)	(6,2)	(6,2)	(7,2)	(7,2)	(8,2)	(9,2)	(9,2)	(10,2)	(10,2)	(11,2)
				(5,3)	(6,3)	(7,3)	(8,3)	(8,3)	(9,3)	(10,3)	(10,3)	(11,3)	(12,3)	(12,3)
						(7,4)	(8,4)	(9,4)	(10,4)	(11,4)	(11,4)	(12,4)	(13,4)	(14,4)
									(10,5)	(11,5)	(12,5)	(13,5)	(14,5)	(15,5)
10	(2,0)	(3,0)	(3,0)	(3,0)	(4,0)	(4,0)	(4,0)	(4,0)	(5,0)	(5,0)	(5,0)	(6,0)	(6,0)	(6,0)
		(3,1)	(4,1)	(4,1)	(5,1)	(5,1)	(5,1)	(6,1)	(6,1)	(7,1)	(7,1)	(8,1)	(8,1)	(9,1)
			(4,2)	(5,2)	(5,2)	(6,2)	(6,2)	(7,2)	(8,2)	(8,2)	(9,2)	(9,2)	(10,2)	(10,2)
				(5,3)	(6,3)	(7,3)	(7,3)	(8,3)	(9,3)	(9,3)	(10,3)	(10,3)	(11,3)	(12,3)
					(6,4)	(7,4)	(8,4)	(9,4)	(9,4)	(10,4)	(11,4)	(12,4)	(12,4)	(13,4)
							(8,5)	(9,5)	(10,5)	(11,5)	(12,5)	(13,5)	(13,5)	(14,5)
											(12,6)	(13,6)	(14,6)	(15,6)
11	(2,0)	(3,0)	(3,0)	(3,0)	(3,0)	(4,0)	(4,0)	(4,0)	(5,0)	(5,0)	(5,0)	(5,0)	(6,0)	(6,0)
		(3,1)	(4,1)	(4,1)	(4,1)	(5,1)	(5,1)	(6,1)	(6,1)	(6,1)	(7,1)	(7,1)	(8,1)	(8,1)
		(3,2)	(4,2)	(5,2)	(5,2)	(6,2)	(6,2)	(7,2)	(7,2)	(8,2)	(8,2)	(9,2)	(9,2)	(10,2)
			(4,3)	(5,3)	(6,3)	(6,3)	(7,3)	(8,3)	(8,3)	(9,3)	(9,3)	(10,3)	(11,3)	(11,3)
				(5,4)	(6,4)	(7,4)	(8,4)	(8,4)	(9,4)	(10,4)	(10,4)	(11,4)	(12,4)	(12,4)
						(7,5)	(8,5)	(9,5)	(10,5)	(10,5)	(11,5)	(12,5)	(13,5)	(13,5)
								(9,6)	(10,6)	(11,6)	(12,6)	(13,6)	(14,6)	(14,6)
												(13,7)	(14,7)	(15,7)
12	(2,0)	(2,0)	(3,0)	(3,0)	(3,0)	(4,0)	(4,0)	(4,0)	(4,0)	(5,0)	(5,0)	(5,0)	(5,0)	(6,0)
		(3,1)	(3,1)	(4,1)	(4,1)	(5,1)	(5,1)	(5,1)	(6,1)	(6,1)	(7,1)	(7,1)	(7,1)	(8,1)
		(3,2)	(4,2)	(4,2)	(5,2)	(5,2)	(6,2)	(6,2)	(7,2)	(7,2)	(8,2)	(8,2)	(9,2)	(9,2)
			(4,3)	(5,3)	(5,3)	(6,3)	(7,3)	(7,3)	(8,3)	(8,3)	(9,3)	(9,3)	(10,3)	(11,3)
				(5,4)	(6,4)	(7,4)	(7,4)	(8,4)	(9,4)	(9,4)	(10,4)	(11,4)	(11,4)	(12,4)
					(6,5)	(7,5)	(8,5)	(9,5)	(9,5)	(10,5)	(11,5)	(11,5)	(12,5)	(13,5)
							(8,6)	(9,6)	(10,6)	(11,6)	(12,6)	(12,6)	(13,6)	(14,6)
									(10,7)	(11,7)	(12,7)	(13,7)	(14,7)	(15,7)
													(14,8)	(15,8)
13	(2,0)	(2,0)	(3,0)	(3,0)	(3,0)	(3,0)	(4,0)	(4,0)	(4,0)	(4,0)	(5,0)	(5,0)	(5,0)	(5,0)
	(2,1)	(3,1)	(3,1)	(4,1)	(4,1)	(4,1)	(5,1)	(5,1)	(6,1)	(6,1)	(6,1)	(7,1)	(7,1)	(7,1)
		(3,2)	(4,2)	(4,2)	(5,2)	(5,2)	(6,2)	(6,2)	(7,2)	(7,2)	(7,2)	(8,2)	(8,2)	(9,2)
			(4,3)	(5,3)	(5,3)	(6,3)	(6,3)	(7,3)	(7,3)	(8,3)	(9,3)	(9,3)	(10,3)	(10,3)
			(4,4)	(5,4)	(6,4)	(6,4)	(7,4)	(8,4)	(8,4)	(9,4)	(9,4)	(10,4)	(11,4)	(11,4)
				(5,5)	(6,5)	(7,5)	(8,5)	(8,5)	(9,5)	(10,5)	(10,5)	(11,5)	(12,5)	(12,5)
						(7,6)	(8,6)	(9,6)	(10,6)	(10,6)	(11,6)	(12,6)	(13,6)	(13,6)
							(8,7)	(9,7)	(10,7)	(11,7)	(12,7)	(13,7)	(13,7)	(14,7)
										(11,8)	(12,8)	(13,8)	(14,8)	(15,8)
14	(2,0)	(2,0)	(3,0)	(3,0)	(3,0)	(3,0)	(4,0)	(4,0)	(4,0)	(4,0)	(4,0)	(5,0)	(5,0)	(5,0)
	(2,1)	(3,1)	(3,1)	(4,1)	(4,1)	(4,1)	(5,1)	(5,1)	(5,1)	(6,1)	(6,1)	(6,1)	(7,1)	(7,1)
		(3,2)	(4,2)	(4,2)	(5,2)	(5,2)	(5,2)	(6,2)	(6,2)	(7,2)	(7,2)	(8,2)	(8,2)	(8,2)
		(3,3)	(4,3)	(5,3)	(5,3)	(6,3)	(6,3)	(7,3)	(7,3)	(8,3)	(8,3)	(9,3)	(9,3)	(10,3)
			(4,4)	(5,4)	(6,4)	(6,4)	(7,4)	(7,4)	(8,4)	(9,4)	(9,4)	(10,4)	(10,4)	(11,4)
				(5,5)	(6,5)	(7,5)	(7,5)	(8,5)	(9,5)	(9,5)	(10,5)	(11,5)	(11,5)	(12,5)
					(6,6)	(7,6)	(8,6)	(9,6)	(9,6)	(10,6)	(11,6)	(11,6)	(12,6)	(13,6)
						(7,7)	(8,7)	(9,7)	(10,7)	(11,7)	(11,7)	(12,7)	(13,7)	(14,7)
								(9,8)	(10,8)	(11,8)	(12,8)	(13,8)	(14,8)	(14,8)
											(12,9)	(13,9)	(14,9)	(15,9)
15	(2,0)	(2,0)	(3,0)	(3,0)	(3,0)	(3,0)	(3,0)	(4,0)	(4,0)	(4,0)	(4,0)	(5,0)	(5,0)	(5,0)
	(2,1)	(3,1)	(3,1)	(3,1)	(4,1)	(4,1)	(4,1)	(5,1)	(5,1)	(5,1)	(6,1)	(6,1)	(6,1)	(7,1)
		(3,2)	(4,2)	(4,2)	(4,2)	(5,2)	(5,2)	(6,2)	(6,2)	(6,2)	(7,2)	(7,2)	(8,2)	(8,2)
		(3,3)	(4,3)	(4,3)	(5,3)	(5,3)	(6,3)	(6,3)	(7,3)	(7,3)	(8,3)	(8,3)	(9,3)	(9,3)
			(4,4)	(5,4)	(5,4)	(6,4)	(7,4)	(7,4)	(8,4)	(8,4)	(9,4)	(9,4)	(10,4)	(10,4)
				(5,5)	(6,5)	(6,5)	(7,5)	(8,5)	(8,5)	(9,5)	(10,5)	(10,5)	(11,5)	(11,5)
				(5,6)	(6,6)	(7,6)	(8,6)	(8,6)	(9,6)	(10,6)	(10,6)	(11,6)	(12,6)	(12,6)
						(7,7)	(8,7)	(9,7)	(10,7)	(10,7)	(11,7)	(12,7)	(12,7)	(13,7)
							(8,8)	(9,8)	(10,8)	(11,8)	(12,8)	(13,8)	(13,8)	(14,8)
									(10,9)	(11,9)	(12,9)	(13,9)	(14,9)	(15,9)
												(13,10)	(14,10)	(15,10)

Table 2: Couples (a, b) for which a success rate $\geq a/n$ is significantly higher than a successe rate $\leq b/m$, for a confidence level of 99%.

function or its symmetry. This situation often occurs in metaheuristics: For instance the observations are limited by the value of the optimum, implying a truncated and asymmetric distribution. Typically, the practitioner want a confidence interval for a statistical function of interest $s(\mathbf{x})$, such as the mean or the median. One of the best approach for dealing with such unknown distributions is the *bootstrap* technique. The idea is to obtain the information of interest by re-sampling (with replacement) a large number B of times from the observations and to count the number of new samples that significantly deviate from the original observations. This technique is simple and robust, but straightforward implementation, such as those given below for a simple illustration, may be less efficient than more advanced (bootstrap) techniques. Reference books for bootstrap techniques are [Efron and Tibshirani (1993)] and [Davison and Hinkley (2003)]. Here is a first way to proceed to obtain a confidence interval for $s(\mathbf{x})$.

- Generate B vectors $\mathbf{x^b}, (b = 1, \ldots, B)$ of size n by choosing each component randomly, uniformly and with replacement among the observed values $(x_1, \ldots, x_n)$
- Compute the values of interest $s(\mathbf{x^1}), \ldots, s(\mathbf{x^B})$ for the B generated vectors and sort them by increasing value.
- It is considered that $s(\mathbf{x})$ has a probability $1 - 2\alpha$ to belong to the interval $[s_1, s_2]$ by taking $s_1 = 100 \cdot \alpha$th percentile of the sorted $s(\mathbf{x^b})$ and $s_2 = 100 \cdot (1 - \alpha)$th percentile of the sorted $s(\mathbf{x^b})$.

Practically, one can choose $B = 2000$ re-samples, $\alpha = 2.5\%$, $s_1 = 50$th and $s_2 = 1950$th values of the sorted list. We see that this method is very simple and adapted to metaheuristic practitioners who are familiar with simulation. Let us mention that this simple method generally not produces the narrowest possible interval and an interval that is centred on the value of interest. Methods like *BCa* (bias corrected and accelerated bootstrap) or *t-Bootstrap* can provide better confidence intervals with lower n and B. *BCa* is a little bit more complicated, but still relatively easy to implement. If the symmetry of the distribution can be assumed, *permutation* tests, that also work with re-sampling (but without replacement), seem to be a [Good 2005] alternative.

5.3 Comparing the quality of two methods

In this section, we suppose that the practitioner has executed method A n times, observed solutions qualities $\mathbf{x} = (x_1, \ldots, x_n)$ and method B m times and observed the quantities $\mathbf{y} = (y_1, \ldots, y_m)$ It is supposed that the observations are independent, but the distribution functions of methods A and B may differ. This case often arises in metaheuristic context. Typically the practitioner is interested in knowing if the average quality of a method is different than the average quality of the other. Answering this question can be done by comparing the confidence intervals of both samples with the techniques described above. A more efficient technique, also based on bootstrapping is the following:

- Compute:
  the averages $\bar{x}$ and $\bar{y}$ of both samples,

the respective variances v_x and v_y,
the average $\bar{z}$ of all the $n+m$ observations,
the value $t_{obs} = t(\mathbf{x}, \mathbf{y}) = \frac{\bar{x}-\bar{y}}{\sqrt{v_x/n+v_y/m}}$.

- Build vectors $\mathbf{x}'$ and $\mathbf{y}'$ with components $x'_i = x_i - \bar{x} + \bar{z}$ and $y'_i = y_i - \bar{y} + \bar{z}$
- Generate B bootstrap samples $\mathbf{x^b}$ and $\mathbf{y^b}, (b = 1, \ldots, B)$ with the modified $\mathbf{x}'$ and $\mathbf{y}'$ observations and compute the associated values $t(\mathbf{x^b}, \mathbf{y^b})$.
- The archived significance level is given by the number of bootstrap samples for which $t(\mathbf{x^b}, \mathbf{y^b}) \leq t_{obs}$ divided by B.

The reader can refer to [Hall (1992)] for further discussions and better bootstrap variants. In order to obtain safe results, especially if a high confidence is needed, bootstrap techniques may require heavy simulations and relatively large samples. If only small samples are available, it is certainly more convenient to use a nonparametric test based on the ranks, such as the Mann-Whitney test (see, e.g. [Conover (1999)]). Finally, let us mention that other techniques exist for comparing more than 2 methods simultaneously. For instance, the analysis of variance (ANOVA) can be used if we want to compare several algorithms on several instances, but we have to make strong assumptions such as normality of the distribution of the samples. If such assumptions cannot be done, then rank-based tests, such as Friedman test can be performed. As noted above, the information provided by these tests can be deceptive for the practitioner who wants to know which method is better than which other. In this case, multiple pairwise comparisons must be performed, using for instance techniques quoted above.

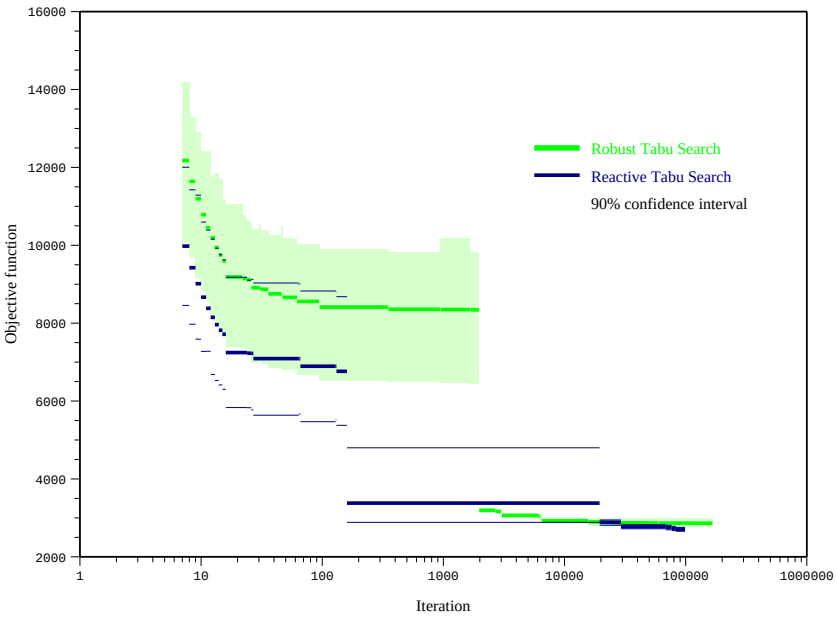

Figure 2: Example of a diagram comparing the evolution of the objective function of two methods with 90% confidence interval

6 Comparison of iterative methods

A simple way to compare iterative methods is to repeat a statistical test for each computational effort. However, instead of accepting or rejecting the test with a given significance level α, we suggest to consider the p-value, i.e. the probability to reject a true hypothesis. Moreover, this allows providing graphical results that are easier to read. A first diagram can provide the evolution of the solutions qualities as a function of computational effort (see Figure 2) and a second diagram can provide the p-value associated to a statistical test comparing 2 methods (see Figure 3).

A logarithmic scale helps the reader to represent the uncertainty in measuring the computational effort. Figures 2 and 3 give examples of such diagrams, that can be automatically generated through the web, by a software developped in our institute, available at the address `http://ina.eivd.ch/projects/stamp`. The same link allows performing few of the statistical tests presented above.

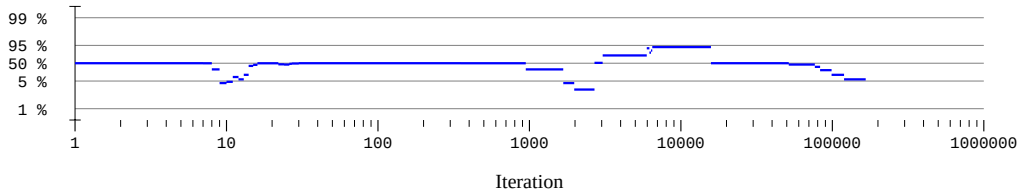

Figure 3: Example of a diagram providing the p-value for a method being better than another (Mann-Whitney rank-based statistics). Note that the p-value is not plotted on a linear scale, to be able to see the interesting portion between 1% and 5%.

References

[Arbuthnott(1710)] J. Arbuthnott, An argument for divine providence, taken from the constant regularity observed in the births of both sexes. *Philosophical Transactions*, **27**, 186–190.

[Conover (1999)] W. J. Conover, *Practical Nonparametric Statistics*, Wiley, Weinheim, third edition.

[Davison and Hinkley (2003)] A. C. Davison, D.V. Hinkley *Bootstrap Methods and their Application*, Cambridge University Press, 2003, 5th reprint.

[Dongarra (2005)] J.J. Dongarra, "Performances of various computers using standard linear equation software", *Technical report CS-89-85*, Univ. of Tennessee, Knoxville, USA.

[Efron and Tibshirani (1993)] B. Efron, R. J. Tibshirani, *An Introduction to the Bootstrap*, Chapman-Hall.

[Good 2005] P. Good. (2005): *Permutation, Parametric, and Bootstrap Tests of Hypotheses*. Springer Verlag, New York. Third edition.

[Hall (1992)] P. Hall, *The bootstrap and edgeworth expansions*, Springer Verlag, New-York.

[Taillard (2003)] É. Taillard, "Heuristic Methods for Large Centroid Clustering Problems", *Journal of Heuristics* **9** (1), 51–73.

[Taillard et al. (2004)] É. Taillard, Ph. Waelti, J. Zuber "Few statistical tests for proportions comparisons". *Technical Report*, EIVD, 2004.