Probabilistic Performance Profiles for the Experimental Evaluation of Stochastic Algorithms

André M. S. Barreto
Laboratório Nacional de Computação Científica
Petrópolis, RJ, Brazil
amsb@lncc.br

Heder S. Bernardino
Laboratório Nacional de Computação Científica
Petrópolis, RJ, Brazil
hedersb@lncc.br

Helio J. C. Barbosa
Laboratório Nacional de Computação Científica
Petrópolis, RJ, Brazil
hcbm@lncc.br

ABSTRACT

One of the many difficulties that arise in the empirical evaluation of new computational techniques is the analysis and reporting of experiments involving a large number of test-problems and algorithms. The performance profiles are a methodology specifically developed for this purpose which provides a simple means of visualizing and interpreting the results of large-scale benchmarking experiments. However good, performance profiles do not take into account the uncertainty present in most experimental settings. This paper presents an extension of this analytic tool called probabilistic performance profiles. The basic idea is to endow the original performance profiles with a probabilistic interpretation, which makes it possible to represent the expected performance of a stochastic algorithm in a convenient way. The benefits of the new method are demonstrated with data from a real benchmark experiment involving several problems and algorithms.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Performance evaluation

General Terms

Experimentation, Measurement, Performance

Keywords

Experimental Evaluation, Performance Profiles

1. INTRODUCTION

The development of new methods to solve a given class of problems requires a clear strategy to evaluate the quality of alternative candidate algorithms. In some cases such evaluation can be done in a formal way, with probabilistic analyses of worst- or average-case scenarios [13]. Usually, though, the analytical difficulties of such analyses make it hard or even impossible to obtain meaningful results for more realistic settings. In these cases, the usual strategy is to resort to an empirical evaluation of the candidate methods: the algorithms are applied to a set of representative problems and have their results compared against each other or against some baseline defined in advance.

Despite its apparent simplicity, the experimental evaluation of algorithms raises some difficulties in practice. First, one must define a set of problems that is heterogeneous enough to represent well the target domain and yet small enough to allow the carrying out of the experiments. Second, it is necessary to determine the performance measures used to evaluate the algorithms, a subject for which there is no consensus in the literature [10, 9, 2]. Finally, one must decide how to interpret, analyze, and report the results obtained in the experiments.

This paper focuses on the third issue above. In particular, we discuss the difficulties involved in the analysis of benchmark results when two complicating factors are present:

1. The number of algorithms and/or problems is too large to allow for the reporting and analysis of all the results or even of the main descriptive statistics;
2. The methods being evaluated are "stochastic" in nature, *i.e.*, there is some level of unpredictability concerning their behavior.

Recently, Dolan and Moré [8] proposed a tool for analyzing benchmark experiments which specifically addresses item 1 above. *Performance profiles* constitute a methodology that makes it easy to summarize, visualize, and interpret the results of large experiments. However good, performance profiles have themselves a serious limitation: they do not account for the variability inherent to most experimental setups. This is particularly problematic in the scenario where the methods under evaluation are stochastic, since in this case the algorithms themselves are a source of uncertainty.

In contrast with deterministic algorithms, which always provide the same results under the same conditions, stochastic algorithms may perform differently in distinct executions even if all of its parameters are kept fixed across the runs. The level of variability in an algorithm's behavior that is considered acceptable varies from application to application. It should be obvious, though, that a good tool for analyzing benchmark experiments must provide some information as to how much uncertainty is associated with the performance of a given candidate method.

In this paper we propose an extension of Dolan and Moré's performance profiles which is able to incorporate and thus to represent the variance associated with the functioning

GECCO'10, July 7–11, 2010, Portland, Oregon, USA.

of a stochastic algorithm. The basic idea is to endow performance profiles with a probabilistic interpretation, which makes it possible to represent the expected performance of a method in a convenient way. As will be shown, the *probabilistic performance profiles* unveil trends that are hard to detect in the raw data and that would be masked by the use of the standard performance profiles.

To make the discussion more concrete, we will be using the terminology usually adopted by the evolutionary-computation community—that is, we will consider that the problems at hand concern the minimization or maximization of a given objective function. It should be noted, however, that the issues discussed here extend straightforwardly to any scenario where candidate methods can be evaluated empirically, even if the problems considered in such scenarios are not normally cast as optimization tasks [3].

The paper is organized as follows. In Section 2 we discuss the steps involved in the design of a good experimental-evaluation setup. We then focus on the issue of how to analyze and to report the experiments' results. We start by presenting the performance profiles in Section 3. Then, in Section 4, a probabilistic version of this analytic tool is proposed. The advantages of the new method are illustrated in Section 5 with a few examples. Finally, we present in Section 6 the main conclusions regarding this investigation.

2. EXPERIMENTAL EVALUATION OF STOCHASTIC ALGORITHMS

The empirical evaluation of stochastic algorithms is a challenging task which constitutes by itself an active area of research [13]. The design of a good experimental setup involves a series of interconnected decisions that can be roughly divided into three steps: selection of problems, definition of performance measures, and analysis of the results. In what follows we briefly discuss each one of these steps.

2.1 Selection of problems

The first stage in carrying out an experimental investigation is to find a set of problems which is representative of the domain of interest. This is not an easy task, since it involves two conflicting objectives: on the one hand, one wants to keep the problem set as small as possible, since this alleviates the computational burden associated with the experiments. On the other hand, one seeks a collection of tasks that spans all problem characteristics relevant to the future application of the algorithm—which usually requires a large number of problem instances.

There have been innumerous attempts to build suites of problems representing specific classes of tasks. From operations research [7] to machine learning [4] and bioinformatics [14], the list is simply too long to be explicitly enumerated. In the context of evolutionary computation, the first attempt to put together a test suite was probably that of De Jong in his Ph.D thesis [6]. De Jong's test suite was composed by five functions with different characteristics, in an effort to provide a sample of the various features one might find in solving real-world optimization problems. Such 5-function suite was used for quite some time before being gradually replaced by larger test-function suites.

2.2 Definition of performance metrics

The second step in the design of an experimental evaluation is the definition of which measures should be taken in order to assess the quality of the algorithms. In some contexts this comes out as a fairly obvious choice: if all the algorithms are guaranteed to find the optimal solutions of the problems, for example, it is clear that the performance metric should be some measure of time efficiency [13]. In the case of stochastic algorithms, however, the situation is usually more complicated than that. Since these algorithms are normally applied to problems for which optimal solutions are very hard to find, one must weight the efficiency of the methods against the quality of the solutions obtained.

There are basically two approaches to evaluate the quality of a stochastic algorithm. One possible strategy is to define a meaningful goal and to measure the amount of computational resources, such as computing time or number of iterations, required by the algorithm to achieve this goal. Examples of goals are: to find a solution within a certain tolerance of the best known solution or, in the case of constrained optimization problems, to simply find a feasible result. Another way to assess the quality of a stochastic algorithm is to fix the amount of computational resources and then check the quality of the solutions delivered by the algorithms within this budget.

2.3 Analysis of the results

Once the problem set has been defined and the performance indexes have been collected, one must decide how to analyze and to report the results generated by the experiments. The main difficulties here are related to the amount of data generated, which may be very large depending on the extension of the experiments performed. If the number of algorithms, problems, and performance metrics involved in the experimental evaluation is such that it is possible to report all the main descriptive statistics in a tabular form, and to easily interpret them, this should probably be the preferred way to present the results. In most realistic scenarios, however, the amount of information available makes it very hard, if not impossible, to analyze the results looking at a huge data table. In this case, one must decide how to best summarize the results, which inevitably involves several tradeoffs.

One source of disagreement refers to what statistics should be reported in the evaluation of algorithms. For example, in a recent paper Birattari and Dorigo [2] argue that it makes no sense to report the best results obtained by a stochastic method in a series of experiments, a practice which is supported by other researches [9]. In a related work, Fleming and Wallace [10] discuss why the geometric mean, and not the commonly used arithmetic mean, is the appropriate statistics to report benchmark results which have been normalized.

Another critical point which is a breeding ground for disagreements is the criterion used to compare the algorithms. One way to do so is to look at the algorithms' average or cumulative total for each performance metric over all the problems considered [5]. An obvious drawback of this strategy is that a small number of the most difficult problems tend to dominate the results [8]. In order to circumvent this issue, some researches have suggested that the algorithms should be ranked according to each performance metric and then a final rank should be computed as the average of the partial ones [12, 15]. This strategy has its disadvantages, too. First, the ranks effectively hide the magnitude of the difference on the algorithms' results. Thus, an algorithm

that performs slightly better than the average on most of the problems and very badly on a small subset of the test suite may be preferred over an algorithm which has a more consistent behavior over the entire problem set. This clearly biases the evaluation against the less specialized and more robust methods. Another drawback of ranking the algorithms is that the rank is overly sensitive to small differences on the algorithms' performances. Again, this is particularly problematic in the context of stochastic algorithms, in which the dispersion of the results is an integral part of the game.

Performance profiles were proposed to overcome the shortcomings associated with the approaches mentioned above. As will be shown, this analytic tool provides information on the relative magnitude of the performance metrics and at the same time avoids that a small portion of the results distort the overall conclusion.

3. PERFORMANCE PROFILES

In order to introduce the performance profiles we consider the following scenario: there is a set P of test problems p_j, with $j = 1, 2, \ldots, n_p$, and a set A of algorithms a_i, $i = 1, 2, \ldots, n_a$. If a specific algorithm is being compared against itself with different parameter settings, each configuration should be considered as a distinct algorithm a_i.

Benchmark results are generated by running algorithm a_i on problem p_j and recording the performance metrics of interest. For each algorithm a_i and each problem p_j we define $c_{ij} > 0$ as a "cost" associated with the execution of a_i on p_j (that is, the smaller c_{ij}, the better the performance of a_i). The variable c_{ij} may be the performance indexes themselves or some value derived from them. Thus, if the experiments consist in the algorithms trying to achieve a specific goal, the variable c_{ij} would be some measure of time efficiency, such as the processing time or the number of iterations performed before reaching the objective (if algorithm a_i does not solve problem p_j, we abuse notation slightly and write $c_{ij} = \infty$). When one is interested in the quality of the solutions found within a limited amount of computational resources, the variable c_{ij} must be defined in such a way that it is always a positive number.

Given the definition of the cost c_{ij}, we define the *performance ratio* of algorithm a_i on problem p_j as

$$r_{ij} \equiv \frac{c_{ij}}{b_j},$$

where b_j is the "baseline" against which the performance of the algorithms are compared. A sensible way to determine such baseline is to simply set $b_j = \min_k c_{kj}$. Then, r_{ij} indicates how many times the cost of algorithm a_i on problem p_j is bigger than the cost of the best performing algorithm on the same problem. The purpose of the variable r_{ij} is to represent the performance metrics in a common scale without losing information regarding the relative magnitude of the original indexes.

In order to present the performance profiles we need one last definition. Let δ be a function given by

$$\delta(r_{ij}, x) = \begin{cases} 1, \text{if } r_{ij} \leq x, \\ 0, \text{ otherwise.} \end{cases}$$

The interpretation of $\delta(r_{ij}, x)$ will depend on the type of experimental evaluation performed. If we specify a goal and measure the resources taken by the algorithms to achieve this goal, $\delta(r_{ij}, x) = 1$ indicates that a_i was able to reach the objective on problem p_j with a cost at most x times bigger than the smallest cost on this problem. If we fix a limited amount of computational resources and measure the performance of a_i, $\delta(r_{ij}, x) = 1$ indicates that a_i returned a solution whose quality is up to x times worse than that of the best-performing algorithm on problem p_j. To make things easier, we will interpret $\delta(r_{ij}, x) = 1$ as simply "algorithm a_i solved problem p_j at a tolerance level of x".

We are finally ready to present the performance profiles. Given a set of problems P and a set of algorithms A, the *performance profile* of a given algorithm a_i on P is a function $\rho_i : \mathbb{R}^*_+ \mapsto [0, 1]$ given by

$$\rho_i(x) \equiv \frac{1}{n_p} \sum_{j=1}^{n_p} \delta(r_{ij}, x).$$

The function $\rho_i(x)$ gives the fraction of the problems in P solved by a_i at a tolerance level of x. Thus, at any point $x = \tau$ we have a partial rank of the algorithms, in which algorithms with larger values for $\rho_i(\tau)$ are ranked higher.

To fix ideas, suppose we set $\tau = 5$. This is equivalent to saying that we are willing to accept a performance on each problem p_i up to five times worse than the best performance on that same problem. Thus, if $\rho_i(5) = 0.7$, for example, this means algorithm a_i is able to solve 70% of the problems within this tolerance. Two particular values of τ allow for interesting interpretations:

- If $\tau = 1$, $\rho_i(\tau)$ equals the proportion of times a_i "wins" over the rest of the algorithms, that is, the fraction of the problems in P in which a_i presents the best performance.
- If $\tau = \infty$, $\rho_i(\tau)$ is the fraction of problems algorithm a_i is able to eventually solve.

Another interesting measure concerns the "reliability" of the algorithms, which can be easily derived from the performance profiles as

$$\zeta(a_i) = \sup\{\tau : \rho_i(\tau) < 1\}.$$

Simply put, $\zeta(a_i)$ is the worst performance of algorithm a_i over the entire test suite (if a_i does not solve all the problems, $\zeta(a_i) = \infty$). Therefore, the most reliable algorithm is the one with the smallest value for $\zeta(a_i)$.

The measures above in isolation may be of interest in particular circumstances, but it is in the overall analysis of the benchmark results that performance profiles show themselves to be really useful. In particular, by looking at the curves $\rho_i(x)$ over the entire interval of interest, we get an easy-to-interpret summary of the measures discussed that allows for a throughout comparison of the methods under different criteria. In addition, the generation of such summary keeps the intervention of the analyst at a minimum.

To illustrate the points above, consider the following example. Suppose we are comparing 3 algorithms on a test suite composed by 5 problems and the costs associated with each algorithm-problem pair are those shown in Table 1. The first thing that stands out when we look at the referred table is the fact that the variable c_{21} is not well defined. This simply means that algorithm a_2 was not able to solve problem p_1 in the experiments performed. Notice though that setting $c_{21} = \infty$ is just an artifice to allow for an easy interpretation of the data; since any empirical evaluation must

use up a finite amount of resources, there is no way to guarantee that algorithm a_2 will not eventually solve p_1—unless, of course, this algorithm halts or stagnates at some point. Therefore, the results of an experiment may leave the functions $\rho_i(x)$ undefined over an interval $(\tau_{\max}, \infty)$. When this is the case, the analysis of the performance profiles should be restricted to the interval $(0, \tau_{\max}]$.

	p_1	p_2	p_3	p_4	p_5
a_1	1.0	1.0	1.0	5.0	3.0
a_2	∞	5.5	5.5	1.0	1.0
a_3	2.0	4.0	4.0	6.5	8.0

Table 1: Algorithms' costs c_{ij}

Suppose $\tau_{\max} = 10$ in the experiment that originated Table 1. By plotting ρ_i for all algorithms a_i over the interval $(0, 10]$ we obtain the graphic shown in Figure 1. It is clear that the performance profile $\rho_i(x)$ is a nondecreasing, piecewise constant function, continuous from the right at each discontinuity point. The discontinuity points are the entries in Table 1.

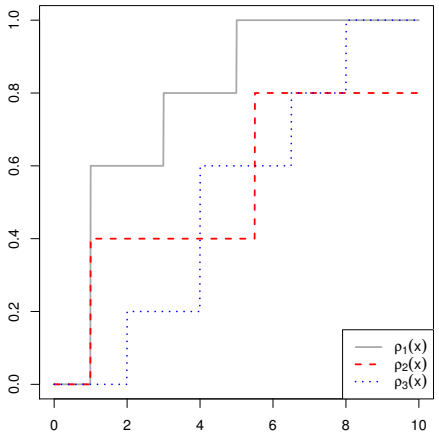


Figure 1: Examples of performance profiles for $n_a = 3$ and $n_p = 5$. The values of the variables c_{ij} used to generate the plots are shown in Table 1.

Several interesting observations can be made regarding the generation and analysis of Figure 1. First, note that the fact that $c_{21} = \infty$ does not preclude the generation of the plot, nor does it require arbitrary interventions of the analyst such as excluding problem p_1 from the test set or defining a penalty for algorithm a_2 for not solving this problem. As for the interpretation of the graphic, one can state the following:

- By looking at the curves $\rho_i(\tau)$ at $\tau = 1$, it is clear that algorithm a_1 presented the best performance on 60% of the problems, whereas algorithm a_2 was the winner at the remaining 40%. Hence, algorithm a_3 was never the best-performing method.
- When we check the performance profiles at $\tau = 10$ we see that both a_1 and a_3 were able to solve all the problems, while algorithm a_2 solved only 80% of the test suite (in this example, 4 out of 5 problems).
- Algorithm a_1 is the most reliable one ($\zeta(a_1) = 5$), followed by algorithm a_3 ($\zeta(a_3) = 8$). Algorithm a_2 is the least reliable, with $\zeta(a_2) = \infty$.
- As $\rho_1(x) \geq \rho_i(x)$ for $i = 2, 3$ over the interval considered, and since $\zeta(a_1) < \tau_{\max}$, algorithm a_1 clearly outperforms a_2 and a_3 on the test set P. This is an example of a situation in which performance profiles give a clearcut answer as to which algorithm is the best choice.
- When we compare algorithms a_2 and a_3, it is not so clear which one should be preferred. In fact, this choice will depend on what exactly is expected from the algorithms. If we are interested in a more specialized method which performs very well on a subset of the problems in P, then a_2 is clearly a better choice than a_3. On the other hand, if we are looking for a reliable algorithm, a_3 should be the selected one. Finally, if we are concerned with the performance of the algorithms at a specific level of tolerance τ, the choice will depend on the value of τ.

The example given shows how performance profiles provide a simple means of analyzing the results of an experimental evaluation under different criteria. Nevertheless, they are not able to unveil all the information contained in the data. In particular, performance profiles fail to represent the variance inherent to most experimental setups. To illustrate this point, suppose we are comparing algorithms a_2 and a_3 at $\tau = 6$. By checking the plot shown in Figure 1, it is clear that a_2 should be the preferred choice in this case. However, when we observe that the curves $\rho_2(x)$ and $\rho_3(x)$ cross each other at $x = 5.5$ and $x = 6.5$, this seems like a questionable choice. Looking back at Table 1, one can see that fluctuations of less than 10% in the performance of the algorithms could change this scenario completely. Small variations like this may happen even if the algorithms being analyzed are deterministic, but it is when the algorithms are stochastic that these fluctuations become a serious issue. In order to circumvent this potential source of uncertainty, it is necessary to somehow incorporate the variance present in the experiments into the performance measures. This is the purpose of the probabilistic performance profiles, discussed in the next section.

4. PROBABILISTIC PERFORMANCE PROFILES

The proposal of this paper is to extend performance profiles to a probabilistic framework. Thus, instead of talking about *numbers* c_{ij} representing the costs of the algorithms, we will be talking about *random variables* C_{ij} distributed according to a given distribution $\mathcal{D}_{ij}$, that is, $C_{ij} \sim \mathcal{D}_{ij}$. Since the definition of the performance ratios r_{ij} depends on the costs c_{ij}, by turning the latter into random variables we automatically do the same with the former. In particular, we define the *probabilistic performance-ratio* as

$$R_{ij} \equiv \frac{C_{ij}}{b_j},$$

where b_j is again a baseline against which the performance of the algorithms is compared. Here we set

$$b_j = \min_k E[C_{kj}],$$

where $E[\cdot]$ indicates expectation according to the underlying probability distribution (we are assuming that the costs were defined in such a way that $\min_k E[C_{kj}] > 0$).

The variables C_{ij} and R_{ij} induce the definition of the *probabilistic performance profiles*, given by

$$\bar{\rho}_i(x) \equiv \frac{1}{n_p} \sum_{j=1}^{n_p} E[\delta(R_{ij}, x)]. \quad (1)$$

Notice that $\delta(R_{ij}, x)$ can be seen as Bernoulli random variable indicating whether algorithm a_i solved problem p_j at a tolerance level of x. Since

$$\sum_{j=1}^{n_p} E[\delta(R_{ij}, x)] = E\left[\sum_{j=1}^{n_p} \delta(R_{ij}, x)\right],$$

it is easy to see that the function $\bar{\rho}_i(\tau)$ gives the expected fraction of the problems algorithm a_i solves at a tolerance level of x. To conclude, observe that the expected value of $\delta(R_{ij}, x)$ is simply the probability that $R_{ij} \leq x$. Thus, definition (1) can be rewritten as

$$\bar{\rho}_i(x) \equiv \frac{1}{n_p} \sum_{j=1}^{n_p} P(R_{ij} \leq x). \quad (2)$$

If there is no uncertainty regarding the value of R_{ij}—that is, if R_{ij} is a constant rather than a random variable—the probabilistic performance profile reduces to its standard form. Therefore, $\bar{\rho}_i(x)$ is a generalization of performance profiles that makes it possible to represent the performance of an algorithm even if its behavior is not completely predictable. Notice that all the analyses that can be made with $\rho_i(x)$ can also be made with $\bar{\rho}_i(x)$; the only difference is that now we are talking about *expected* performance instead of discussing a single execution of the algorithms.

The discussion above makes it clear that probabilistic performance profiles should be preferred over their standard counterparts when there is a certain level of uncertainty regarding the experimental setup considered. However, in order to use this tool one must be able to determine the value of $\bar{\rho}_i(x)$, which amounts to computing expression (2) at any given point x. The computation of $P(R_{ij} \leq x)$ appearing in (2) requires the knowledge of the distribution of R_{ij}, which in turn depends on the distribution $\mathcal{D}_{ij}$. Usually, though, we do not know how the variables C_{ij} are distributed, and the standard approach to deal with this situation is to assume a certain parametric form for $\mathcal{D}_{ij}$ and try to estimate its parameters based on a sample of the corresponding random variable. In the next section we discuss how this can be done when we assume that the variables C_{ij} are normally distributed.

4.1 Probabilistic performance profiles for normal distributions

Usually the distribution of the variables C_{ij} is not known. When this is the case, the most common approach is to assume these variables are normally distributed [11]. One justification for this choice is the central limit theorem, which states that the sum of a large number of independent random variables (normal or not) has approximately a normal distribution [11]. Thus, by considering that C_{ij} are normally distributed we are implicitly assuming that the performance of an algorithm on a problem is the sum of several random effects acting independently—which sounds reasonable, at least in principle.

In our model each variable C_{ij} comes from a distinct normal distribution $\mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$, where μ_{ij} is the mean of the distribution and σ_{ij} is its standard deviation. Since the parameters μ_{ij} and σ_{ij} are unknown, we must estimate their values from a sample of the variable C_{ij}. It is well known that the maximum likelihood estimates of the mean and the standard deviation of a normal distribution are given by [11]

$$\hat{\mu}_{ij} = \frac{1}{n_{ij}} \sum_{k=1}^{n_{ij}} c_{ij}^k \quad \text{and} \quad \hat{\sigma}_{ij}^2 = \frac{1}{n_{ij} - 1} \sum_{k=1}^{n_{ij}} \left(c_{ij}^k - \hat{\mu}_{ij}\right)^2,$$

where n_{ij} is the number of times algorithm a_i was executed on problem p_j and c_{ij}^k is the cost associated with the k-th such execution. As $n_{ij} \to \infty$, $\hat{\mu}_{ij} \to \mu_{ij}$ and $\hat{\sigma}_{ij} \to \sigma_{ij}$.

Using the properties of the normal distribution, it is easy to show that $C_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$ implies that

$$R_{ij} \sim \mathcal{N}\left(\frac{\mu_{ij}}{b_j}, \frac{\sigma_{ij}^2}{b_j^2}\right).$$

Thus, we can use $\hat{\mu}_{ij}$ and $\hat{\sigma}_{ij}$ to approximate the distribution of R_{ij}. The computation of (2) then becomes easy: in order to determine $P(R_{ij} \leq x)$, it suffices to use the cumulative distribution function of a normal distribution with mean $\hat{\mu_{ij}}/b_j$ and standard deviation $\hat{\sigma}_{ij}/b_j$. In particular,

$$P(R_{ij} \leq x) = F\left(x; \frac{\mu_{ij}}{b_j}, \frac{\sigma_{ij}^2}{b_j^2}\right),$$

where $F(\cdot; \mu, \sigma^2)$ is the cumulative distribution function of $\mathcal{N}(\mu, \sigma^2)$, given by:

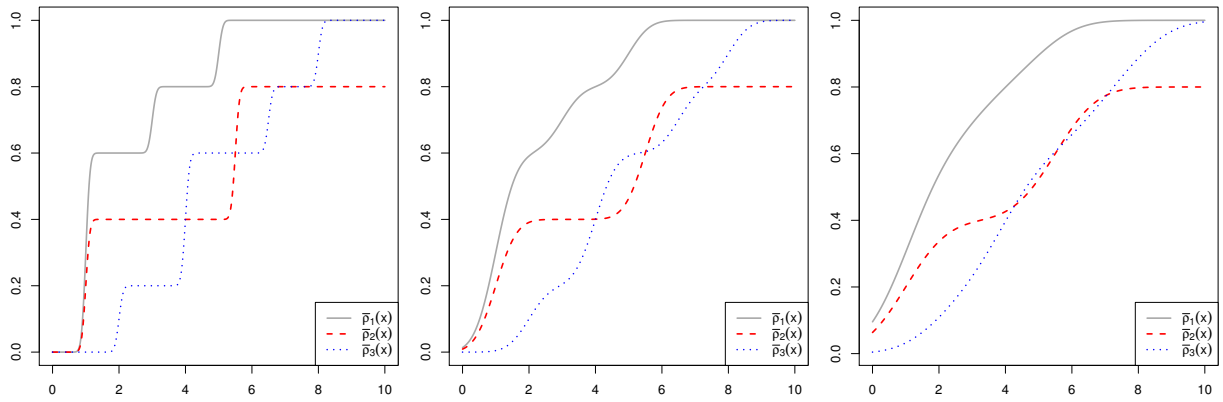
$$F(x; \mu, \sigma^2) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(z-\mu)^2}{2\sigma^2}\right) dz.$$

Notice that the means $\hat{\mu}_{ij}$ and the standard deviations $\hat{\sigma}_{ij}$ are the only two statistics necessary to generate the probabilistic performance profiles. Since these values are usually reported in the literature, it is easy to analyze the results of previous benchmark experiments even when the raw data is not available. To conclude, it should be pointed out that the normal distribution may not always be the most appropriate way to describe the behavior of a stochastic algorithm. When this is the case, one must work out the steps above using the distribution selected to represent C_{ij}. Obviously, as long as it is possible to compute (2), probabilistic performance profiles are still applicable.

5. ILLUSTRATIVE EXAMPLES

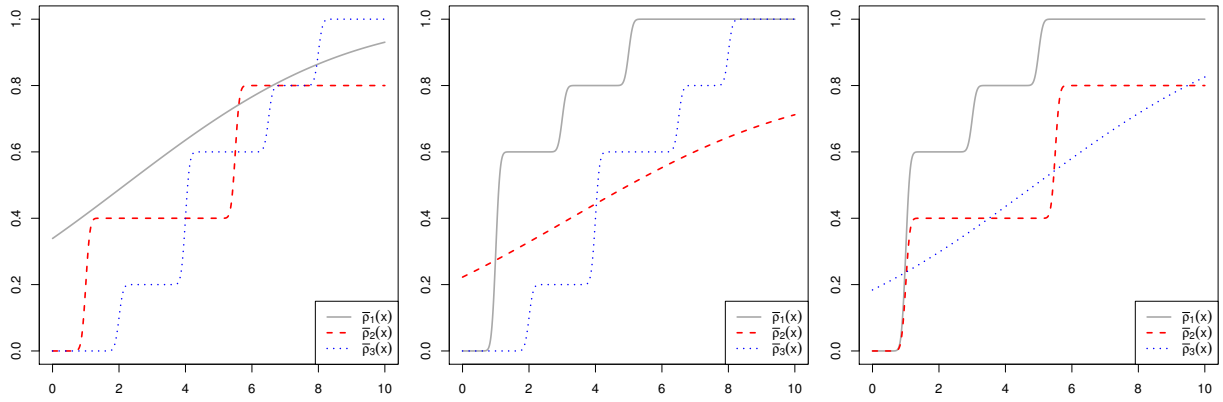
In this section we present a few examples showing how probabilistic performance profiles are able to reveal trends in benchmark results that would be masked by the standard version of this tool. For all the examples discussed we assume that the variables C_{ij} are normally distributed.

We start out by showing the consequences of incorporating several levels of variance to the curves shown in Figure 1. This is done in Figure 2. Observe how the effect of increasing the standard deviation is to progressively smooth out and "disentangle" the performance profile curves. In particular, notice how $\bar{\rho}_2(x)$ and $\bar{\rho}_3(x)$ get closer to each other until they almost coincide in the interval $[4, 7]$ when $\sigma_{ij} = 1$.



(a) $\sigma_{ij} = 0.1$ (b) $\sigma_{ij} = 0.5$ (c) $\sigma_{ij} = 1$

Figure 2: Probabilistic performance profiles in which the algorithms' costs C_{ij} have the same standard deviation. The values $\hat{\mu}_{ij}$ used to generate the plots are the same used for the variables c_{ij} to create Figure 1 (see Table 1).



(a) $\sigma_{1j} = 5$ (b) $\sigma_{2j} = 5$ (c) $\sigma_{3j} = 5$

Figure 3: Probabilistic performance profiles in which the algorithms' costs C_{ij} may have different standard deviations. Unless otherwise noted in the plots' captions, $\sigma_{ij} = 0.1$.

This is in accordance with our intuition that small differences on the performance of stochastic algorithms should not be taken too seriously. Also, notice how the reliability of a_3 decreases as σ_{ij} grows. This means that by using the standard performance profiles to compute $\zeta(a_3)$ one would be overestimating the real capability of this algorithm.

Things start to get more interesting when the level of variance added to the curves varies across the algorithms. This represents the situation in which some of the methods present a larger dispersion in the results than the others. Figure 3 illustrates what happens with the curves of Figure 2a when we increase the standard deviations of the algorithms one at a time. Observe in Figure 3a how the performance of algorithm a_1 degenerates when we change σ_{1j} from 0.1 to 5. Now the curve $\bar{\rho}_1(x)$ crosses both $\bar{\rho}_2(x)$ and $\bar{\rho}_3(x)$, leaving room for doubt where there was none. In fact, after this change any of the three algorithms could be selected as the best one, depending on the region of the x axis we decided to focus our analysis. Similar phenomena occur when we increase the standard deviations of a_2 and a_3, as shown in Figures 3b and 3c, respectively. As an example, observe in Figure 3c how algorithm a_3, considered as a reliable method so far, solves only about 80% of the problems when $\sigma_{3j} = 5$. Other examples could be constructed by changing the variances of a subset of the algorithms or, alternatively, by setting the standard deviations σ_{ij} independently of each other. In order to have a more realistic analysis of probabilistic performance profiles, however, lets turn to a real example.

During the 2009 Genetic and Evolutionary Computation Conference, a Black Box Optimization Benchmarking Workshop was organized in order to quantify and to compare in a rigorous way the performance of real-parameter optimization algorithms [1]. Among other things, the organizers of the workshop provided a well-motivated 24-function testbed that should be used by the participants to evaluate their chosen algorithms. The functions were presented in different versions, varying in both dimensionality and in the presence or not of noise. In order to illustrate the use of probabilistic performance profiles in a real scenario, we adopted the noise-free subset of 40-dimensional functions. Fifteen algorithms were compared under this setting. Figure 4 shows the performance profiles of these algorithms with and without the information regarding their variances. In both cases, the

performance metric is the number of function evaluations necessary to achieve an absolute difference to the optimal objective-function value that is smaller than 10^{-3}.

As shown in Figure 4, the same trend observed in the artificial data takes place here: the overall effect of incorporating the variance into the performance profile curves is to smooth them out and thus separate them. This makes it easier to see trends that would be hard to identify otherwise. As an example, observe how some of the curves shown in Figure 4a coincide over subintervals of the x axis. When these curves are replaced by their probabilistic versions, such overlaps almost never happen.

In order to have a better picture of the results presented in Figure 4, we show in Figure 5 the performance profiles of only a subset of the algorithms, which we will refer to as a_1, a_2, ..., a_6. Several interesting observations can be made regarding this picture. First, note how the ranks induced by both versions of the performance profiles at $x = 10$ are slightly different. In particular, observe in Figure 5b how the probabilistic version of this tool manages to brake a tie between algorithms a_1 and a_2, which present identical results in Figure 5a. Perhaps more surprisingly, the order of algorithms a_3 and a_4 in the rank is inverted, which suggests that by not considering the variability of the methods one is overestimating the performance of the former. Another interesting observation concerns the behavior of algorithm a_5. Notice how the curve $\rho_5(x)$ touches $\rho_3(x)$, $\rho_6(x)$, and $\rho_4(x)$ at different points of the x axis, overlapping with the first two over short intervals. When we look at $\bar{\rho}_5(x)$ instead of $\rho_5(x)$, it is much clear that algorithm a_5 presents an intermediary performance between that of a_6, which is slightly worse, and that of a_3 and a_4. Finally, if we check the curves at $x = 1$—which gives the (expected) number of wins of a given algorithm—the probabilistic performance profiles help again to clarify small differences on the algorithms' results. Since it is hard to see these values at the scale of Figure 5, we show $\rho_i(1)$ and $\bar{\rho}_i(1)$ in Table 2. Observe how the latter is much more effective in discriminating the performance of the methods.

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
$\rho_i(1)$	0.042	0.042	0.000	0.125	0.083	0.042
$\bar{\rho}_i(1)$	0.052	0.064	0.008	0.098	0.057	0.028

Table 2: Value of $\rho_i(x)$ and $\bar{\rho}_i(x)$ at $x = 1$

6. CONCLUSION

This paper introduced the probabilistic performance profiles, a tool for analyzing benchmark results which extends the functionality of its predecessor by allowing the representation of the uncertainty inherent to many experimental setups. This ability is particularly important in the evaluation of stochastic methods, since in this case ignoring the variability of the algorithms may lead to deceptive conclusions. Probabilistic performance profiles are a general framework that can be specialized to many particular situations. As a standard approach, we assume the behavior of the algorithms is the additive effect of many random variables, which amounts to modeling the indexes derived from the performance metrics as normal variables. As shown, this approach succeeds in unveiling hidden trends present in data from real benchmark experiments.

Acknowledgments

We would like to thank the support provided by the Brazilian agencies CAPES, CNPq (grants 311651/2006-2 and 1405-51/2008-5), and FAPERJ (grant E-26/102.825/2008). We also thank the reviewers from GECCO, whose suggestions were of great value for the preparation of the final manuscript.

7. REFERENCES

[1] A. Auger. Black box optimization benchmarking (BBOB), 2009. http://coco.gforge.inria.fr.

[2] M. Birattari and M. Dorigo. How to assess and report the performance of a stochastic algorithm on a benchmark problem: Mean or best result on a number of runs? *Optimization Letters*, 1(3):309–311, 2007.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag Inc., 2006.

[4] C. Blake and C. Merz. UCI repository of machine learning databases, 1998. URL: http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[5] A. R. Conn, N. Gould, and P. L. Toint. Numerical experiments with the LANCELOT package (release A) for large-scale nonlinear optimization. *Mathematical Programming*, 73(1):73–110, 1996.

[6] K. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, The University of Michigan, 1975.

[7] E. Demirkol, S. Mehta, and R. Uzsoy. Benchmarks for shop scheduling problems. *European Journal of Operational Research*, 109(1):137–141, August 1998.

[8] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[9] A. Eiben and M. Jelasity. A critical note on experimental research methodology in EC. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 582–587. IEEE.

[10] P. J. Fleming and J. J. Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *Comm. of the ACM*, 29(3):218–221, 1986.

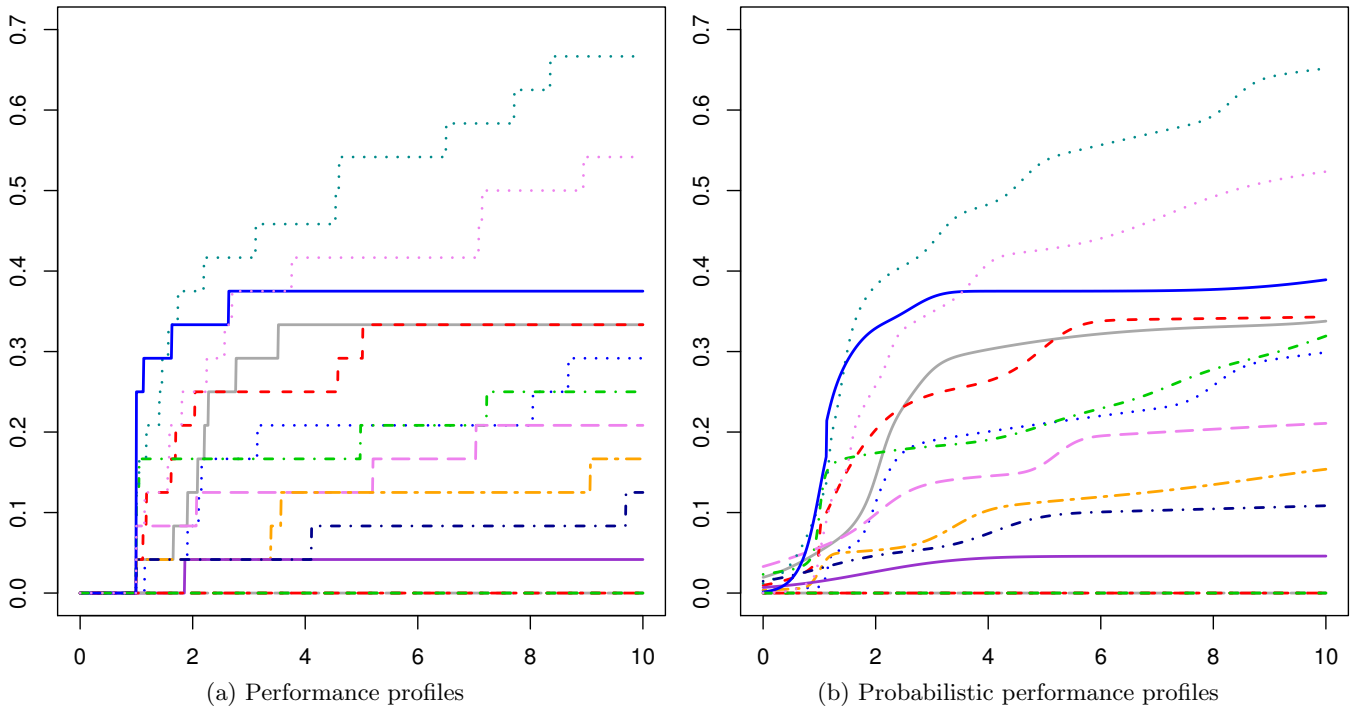
[11] P. G. Hoel. *Introduction to mathematical statistics*. Wiley, New York, 1963.

[12] J. J. Liang, T. P. Runarsson, E. M. M. Clerc, P. N. Suganthan, C. A. C. Coello, and K. Deb. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Technical report, School of EEE, Nanyang Technological University, Singapure, 2006.

[13] R. L. Rardin and R. Uzsoy. Experimental evaluation of heuristic optimization algorithms: A tutorial. *Journal of Heuristics*, 7(3):261–304, 2001.

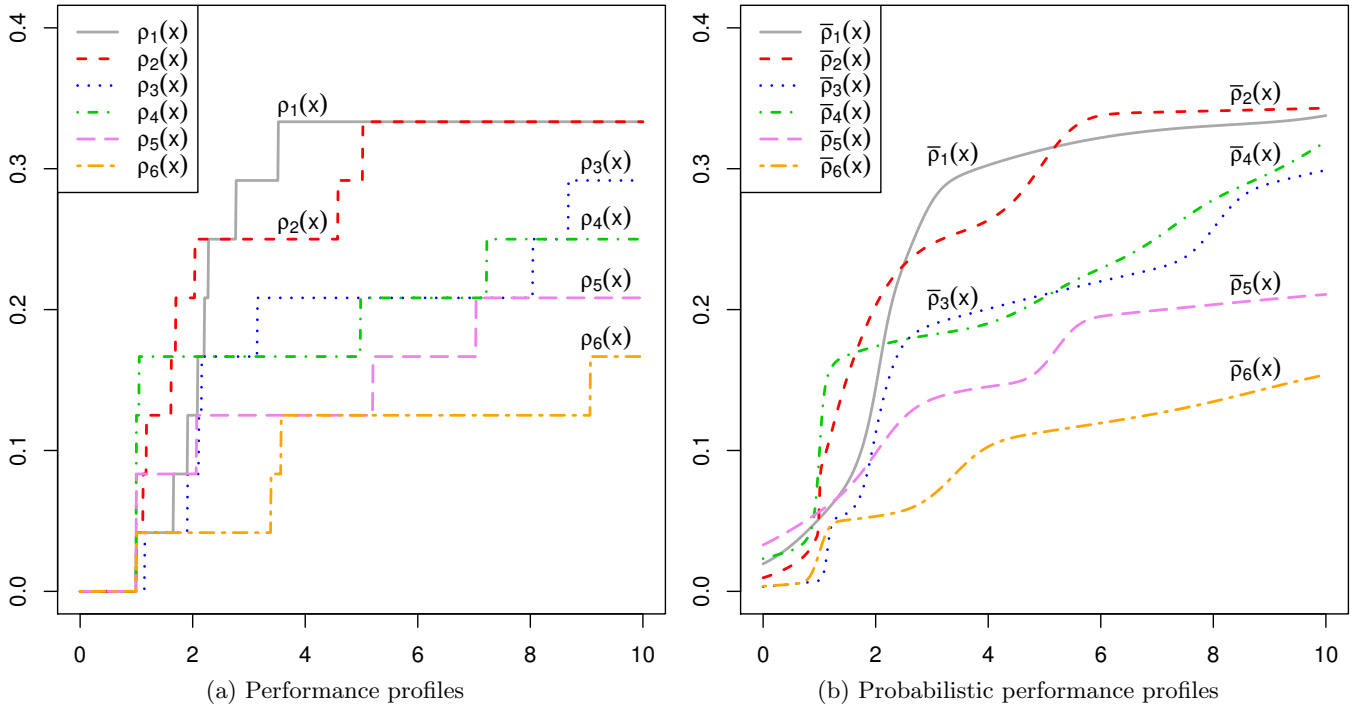
[14] A. Shmygelska and H. Hoos. An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem. *BMC Bioinformatics*, 6(1):30, 2005.

[15] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang. Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, November 2007.



(a) Performance profiles (b) Probabilistic performance profiles

Figure 4: Results of 2009 Black Box Optimization Benchmarking Workshop. The data correspond to 15 executions of the methods. The algorithms' names were omitted to improve the readability of the graphics.



(a) Performance profiles (b) Probabilistic performance profiles

Figure 5: Replot of some of the performance profiles shown in Figure 4.