Chapter 4
SRCS: A Technique for Comparing Multiple Algorithms under Several Factors in Dynamic Optimization Problems

Ignacio G. del Amo and David A. Pelta

Abstract. Performance comparison among several algorithms is an essential task. This is already a difficult process when dealing with stationary problems where the researcher usually tests many algorithms, with several parameters, under different problems. The situation is even more complex when dynamic optimization problems are considered, since additional dynamism-specific configurations should also be analyzed (e.g. severity, frequency and type of the changes, etc). In this work, we present a technique to compact those results in a visual way, improving their understanding and providing an easy way to detect algorithms' behavioral patterns. However, as every form of compression, it implies the loss of part of the information. The pros and cons of this technique are explained, with a special emphasis on some statistical issues that commonly arise when dealing with random-nature algorithms.

4.1 Introduction

An essential task in the optimization area is to evaluate an algorithm against variations of different factors, either to determine the best combination of parameters for it (step size, population, etc) or to verify its robustness over several settings of a problem (number of local optima, dimensionality, etc). When dealing with Dynamic Optimization Problems (DOPs) [6], this situation is even harder, since these problems have some extra features that need to be analyzed (frequency of changes in the environment, severity of the change, etc). Moreover, it is also usual to compare

Ignacio G. del Amo · David A. Pelta
Models of Decision and Optimization Research Group (MODO),
Dept. of Computer Sciences and Artificial Intelligence, University of Granada,
I.C.T. Research Centre (CITIC-UGR), C/ Periodista Rafael Gómez, 2,
E-18071, Granada, Spain
e-mail: ngdelamo@ugr.es, dpelta@decsai.ugr.es
http://modo.ugr.es

E. Alba et al. (Eds.): Metaheuristics for Dynamic Optimization, SCI 433, pp. 61–77.
springerlink.com

multiple algorithms at the same time, for example to verify if a new proposal outperforms previous state-of-the-art techniques (1-vs-all), or to determine the best one of a set of methods in a competition when no prior knowledge about their performance exists (all-vs-all).

Apart from that, if metaheuristics and non-exact algorithms are used to solve these problems, their random nature will make it necessary to perform several independent repetitions of each experiment in order to obtain a set of result samples representative enough of its underlying distribution. There are several ways of presenting these samples, ranging from giving the mean or the median, to include statistics like the standard deviation, the first and third quartile, etc. In general, the more the information provided of the sample, the better the analysis that can be performed on the results. On the other hand, if there is too much information, it will be more complex to present it to the reader, and also, it will be increasingly difficult to manage it and grasp its meaning. Furthermore, this set of result samples is a random variable, and it is no longer a question of comparing two single values for deciding which algorithm is better; it is necessary to use statistical tools. One of the most used techniques is hypothesis testing, where a null hypothesis is stated (for example, that the underlying distribution of the results of two algorithms is the same) against an alternative hypothesis (that the distributions are not the same), and it is checked if the data from the samples support the null hypothesis at a certain significance level. If the data are too unlikely at that level, the null hypothesis is rejected.

Depending on the approach used for the design of the experiments (fractional, 2^k, full-factorial, etc. [1, 2]), the amount of obtained results can vary greatly. There are several techniques commonly used when presenting these data, ranging from the traditional numerical tables to specifically designed graphs (line charts, barplots, boxplots, qqplots, etc). When there are few results, the use of numerical tables is probably one of the best options, since they provide a complete and precise description of the data. However, if the amount of results increases, tables become rapidly intractable, due to their extension, along with the difficulty of comprehending the meaning of so much numerical data. Graphs allow to alleviate this situation, summarizing the data in a visual way. But again, if several factors are analyzed at the same time in a single experiment, it may be necessary to further compress the information, since the number of plots may grow to unsustainable levels (an example of this will be shown in Sect. 4.2). Several special-purpose graphs can be used in this situation to cope with high amounts of data (dendograms, combinations of pie-charts and scatter plots at the same time), although the type of graph that better suits each case tends to be dependent on the specific problem at hand. For an extensive showcase of visualization techniques, the interested reader is referred to [5].

The goal of this chapter is to introduce a technique named SRCS (Statistical Ranking Color Scheme) specifically designed to analyze the performance of multiple algorithms in DOPs over variations of several factors. This technique is based on the creation of a ranking of the results for each algorithm using statistical tests, and then presents this ranking using a color scheme that allows to compress it. It should be noted that other authors have already addressed this topic in related

areas with similar approaches. For example, in the machine learning field, Demšar [8] uses statistical tests for comparing several classifiers over multiple data sets, proposing also a graphical way for presenting statistically non-differentiable classifiers by means of connected lines. And in the multi-objective optimization area, Fonseca et al. [9] devise a technique based on statistical tests to determine the level of attainment of each objective by several algorithms, while López-Ibáñez et al. [17] use this technique along with a graphical color-plotting scheme to visually remark the differences of the algorithms along their obtained Pareto-front.

Before giving the details of the SRCS technique, we will illustrate the application scenario with a practical example.

4.2 Typical Research Case: Comparing Multiple Algorithms over Several Configurations of a Problem

Let us suppose that we want to compare the performance of a set of metaheuristic algorithms over a DOP, for example, the Moving Peaks Benchmark (MPB).

The MPB is a test benchmark for DOP's originally proposed in [3]. It is a maximization problem consisting in the superposition of **m** peaks, each one characterized by its own height (**h**), width (**w**), and location of its centre (**p**). The fitness function of the MPB is defined as follows: Moving Peaks Benchmark

$$\mathbf{MPB}(\mathbf{x}) = \max_j \left(h^j - w^j \sqrt{\textstyle\sum_{i=1}^{n} (x_i - p_i^j)^2} \right),\ j = 1, ..., m, \tag{4.1}$$

where n is the dimensionality of the problem. The highest point of each peak corresponds to its centre, and therefore, the global optimum is the centre of the peak with the highest parameter **h**.

Dynamism is introduced in the MPB by periodically changing the parameters of each peak j after a certain number of function evaluations (ω):

$$\mathbf{h}_j(t+1) = \mathbf{h}_j(t) + \mathbf{h_s} \cdot \mathbf{N}(0,1) \tag{4.2}$$

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \mathbf{w_s} \cdot \mathbf{N}(0,1) \tag{4.3}$$

$$\mathbf{p}_j(t+1) = \mathbf{p}_j(t) + \mathbf{v}_j(t+1) \tag{4.4}$$

$$\mathbf{v}_j(t+1) = \frac{\mathbf{s}}{|\mathbf{r} + \mathbf{v}_j(t)|} ((1-\lambda)\mathbf{r} + \lambda \mathbf{v}_j(t)). \tag{4.5}$$

Changes to both width and height parameters depend on a given severity for each of them ($\mathbf{w_s}$ and $\mathbf{h_s}$). Changes to the centre position depend on a shift vector $\mathbf{v}_j(t+1)$, which is a linear combination of a random vector **r** and the previous shift vector $\mathbf{v}_j(t)$ for the peak, normalized to length **s** (position severity, shift distance, or simply *severity*). Finally, parameter λ indicates the linear correlation with respect to the previous shift, where a value of 1 indicates "total correlation" and a value of 0 "pure randomness".

The MPB has been widely used as a test suite for different algorithms in the presence of dynamism. One of the most used configurations for this purpose is Scenario 2, which consists of the set of parameters indicated in Table 4.1.

Table 4.1 Standard settings for Scenario 2 of the Moving Peaks Benchmark

Parameter	Value
Number of peaks (m)	$\in [10, 200]$
Number of dimensions (d)	5
Peaks heights (h_i)	$\in [30, 70]$
Peaks widths (w_i)	$\in [1, 12]$
Change frequency (ω)	5000
Height severity (h_s)	7.0
Width severity (w_s)	1.0
Shift distance (s)	$\in [0.0, 3.0]$
Correlation coefficient (λ)	$\in [0.0, 1.0]$

Once the problem has been defined, we need to decide which performance measures are we going to use for evaluating and comparing the algorithms. There are different options, like using directly the fitness of the algorithm at every moment in time, or the absolute error in case the optimum is known. However, these measures have the problem that they are expressed in absolute units, and they do not give an idea of how close was an algorithm of reaching the optimum, nor allow us to easily compare the results between changes in the environment. For example, if at a given instant in time the fitness of an algorithm can be in the interval $[0, 10]$, an absolute error of 9 units is a very bad result, while if, in another instant, the fitness can be in the interval $[0, 1000]$, the same absolute error of 9 units is a remarkably good result. In order to allow an easier comparison of the results, a *relative* performance measure would be desirable.

Therefore, for the examples of this chapter, we will assume that the optimum is known, and we will use Weicker's definition [21] of the *accuracy* performance measures!accuracy of an algorithm A over a function F at a given instant in time t, as the basic performance measure:

$$accuracy_{F,A}^{(t)} = \frac{F(sol_A^{(t)}) - Min_F^{(t)}}{Max_F^{(t)} - Min_F^{(t)}}, \tag{4.6}$$

where $sol_A^{(t)}$ is the solution generated by the algorithm A at the instant t, and $Min_F^{(t)}$ and $Max_F^{(t)}$ are, respectively, the minimum and maximum values of the function F at the instant t. This measure has the advantage of always being bounded between 0 and 1, 0 being the worst possible value, and 1 the best (note that this is true independently of whether the problem is of the maximization or minimization type). In Weicker's original definition, $best_A^{(t)}$ is used instead of $sol_A^{(t)}$, referring to the best value of the algorithm at time t. This definition assumes that the algorithm is of an

evolutionary type, where a population of solutions is evaluated at once, and $best_A^{(t)}$ refers to the best individual in that population. However, this cannot be always assumed, since the algorithm may not be population-based, or the problem may not allow that type of concurrent evaluation, thus forcing us to evaluate each solution sequentially. With the aim of not restricting the study to any given implementation of the algorithm nor the problem, we will use $sol_A^{(t)}$ without loss of generality over the *accuracy*. In return, we will define, independently of the algorithm, the *bestAccuracy* measure performance measures!best accuracy as:

$$bestAccuracy(t,t_0) = \begin{cases} accuracy(t) & \text{if } t = t_0 \\ \max\{accuracy(t), bestAccuracy(t-1)\} & \text{if } t > t_0 \end{cases} \quad (4.7)$$

where $t = t_0$ indicates the instant of time immediately after a change in the environment (variable t is "reset" in every change), such that the *bestAccuracy* refers only to the time elapsed since the last change.

We will now extend the *accuracy* to its *offline* and *average offline* performance measures!offline accuracy versions for several consecutive performance measures!average offline accuracy changes in the environment, using De Jong [7] and Branke [4] definitions:

$$offlineAccuracy(t_0,T) = \frac{1}{T-t_0}\sum_{t=t_0}^{T} bestAccuracy(t,t_0) \quad (4.8)$$

$$avgOfflineAccuracy(N_c) = \frac{1}{N_c}\sum_{n=0}^{N_c} offlineAccuracy(\tau_0(n),\tau_T(n)), \quad (4.9)$$

where N_c is the total number of changes considered, and $\tau_0(n)$ and $\tau_T(n)$ are functions that return, respectively, the first and last instant of time t of the stationary period n. A graphical explanation of these measures can be seen in Figs. 4.1 and 4.2.

At this point we can summarize an execution or *run* of an algorithm with the *avg. offline accuracy*. However, as it has been mentioned in the previous section, when dealing with stochastic algorithms it is necessary to perform a series of independent repetitions of the experiments in order to obtain a representative sample of its performance. Therefore, we will execute N_r runs of the algorithm, thus obtaining N_r measurements of the *avg. offline accuracy*.

Now that we have already defined how are we going to measure the performance, let us suppose that we want to compare 4 hypothetical algorithms for a given configuration of the MPB (for example, the widely used Scenario 2). In Sect. 4.4 we will analyze in more detail the influence of N_r in the results of the statistical tests, but for the moment, let us just assume that we perform a fixed amount of independent repetitions, say $N_r = 30$, for each algorithm. An example of the results that could be obtained is presented in Table 4.2 and Fig. 4.3.

In order to determine the existence of statistically significant differences in the results, we need to perform a series of hypothesis tests. Several authors have already

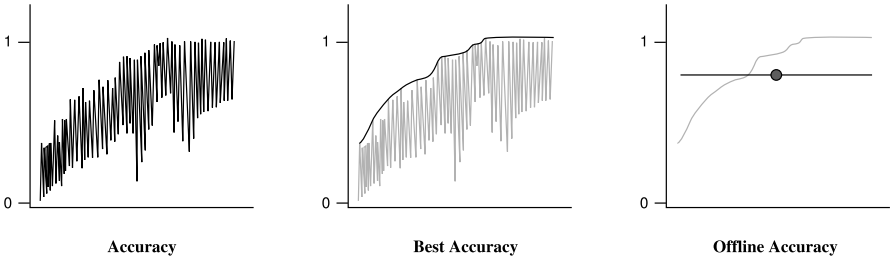

Fig. 4.1 Performance measure of an algorithm using different versions of *accuracy*

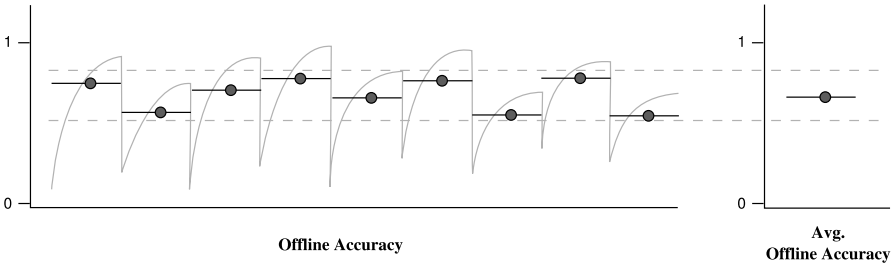

Fig. 4.2 Performance measure of an algorithm over several consecutive changes in the environment using the *offline accuracy* (the *best accuracy* is displayed in the background). The sudden drops of the *best accuracy* values indicate a change in the environment. The average of all *offline accuracy* measures is displayed in the right

pointed out that these results, in general, do not follow a normal distribution [11], therefore recommending the use of non-parametric tests for their analysis [13, 19]. We will use a significance level $\alpha = 0.05$, meaning that we are willing to assume a probability of mistakenly rejecting the null hypothesis of, at most, 0.05. The first issue that needs to be addressed is the fact that we are comparing multiple algorithms at the same time. Therefore, we need to use a test that allows to compare more than 2 groups simultaneously. For this example, we will perform a Kruskal-Wallis (KW) test [15], among all the samples of the 4 algorithms to check if there are global differences at the 0.05 significance level. If the KW test concludes that there are statistically significant differences, we will then perform a series of pair-wise tests between each pair of algorithms, to see if we can determine which are the ones that are causing those differences. In this case, we will use the Mann-Whitney-Wilcoxon test Mann-Whitney-Wilcoxon (MWW) [18, 22] test to compare each pair of algorithms. The combination of these tests is suitable, since the KW test can be considered as the natural extension of the MWW test to multiple groups. It is important to note that in order to guarantee the α-level achieved by the KW test (global), we need to adjust the α-level of each MWW test (pair-wise) to a certain value, usually much smaller than the first one. For this purporse, we will use Holm'shypothesis

testing!Holm correction correction [14], although other techniques are also available (for example, Hochberg's, Hommel's, etc; for an in-depth comparison on the use of these techniques, the interested reader is referred to [8, 10, 11]). The results of the tests are shown in Table 4.3, where individual comparisons between each pair of algorithms can be seen, along with the sign of the comparison.

Table 4.2 Performance results of several algorithms on a single problem configuration (mean and standard deviation values of the *avg. offline accuracy*)

	Algorithm 1	**Algorithm 2**	**Algorithm 3**	**Algorithm 4**
Avg. Offline Accuracy	0.78 ±0.05	0.84 ±0.02	0.95 ±0.01	0.89 ±0.03

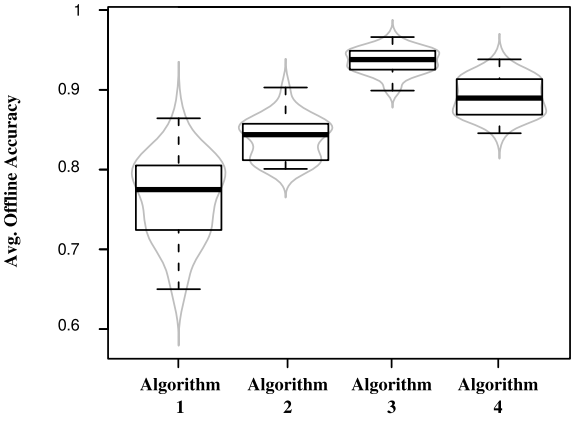

Fig. 4.3 Graphical representation of the results in Table 4.2. The distributions are displayed using a boxplot (dark lines) in combination with a kernel estimation of the distribution density (light lines)

Until now, the way of presenting the results (numerical data tables, boxplot graphs, and statistical tests tables) has been appropriated, and the data are comprehensible. We now contemplate extending the experimental framework. We want to know if the conclusions obtained for the algorithms follow any kind of pattern related to some characteristic of the problem (e.g., whether algorithm 3 is good only for Scenario 2 of the MPB, or if this is a general behaviour linked to, for example, low change frequencies). In order to answer this question, we perform more experiments, keeping all problem parameters constant, except for the change frequency, which we vary progressively.

We can see now that the number of results increases, and its presentation begins to be a problem, both at a table level, because of its extension and difficulty to comprehend the data (Table 4.4), and at a graphical level, because of its complexity (Fig. 4.4). However, it is still feasible to show the results this way, since, although data are

Table 4.3 Pairwise statistical differences among the *avg. offline accuracy* distribution of the algorithms. A '+' sign indicates that there are statistically significant differences between the algorithm in the row and the algorithm in the column, and that the sign of the comparison favors the algorithm in the row (i.e., is "better"). A '-' sign indicates the opposite, that the algorithm in the row is "worse". Finally, the word 'no' indicates no statistically significant differences

	Algorithm 1	**Algorithm 2**	**Algorithm 3**	**Algorithm 4**
Algorithm 1	no	-	-	-
Algorithm 2	+	no	-	-
Algorithm 3	+	+	no	+
Algorithm 4	+	+	-	no

Table 4.4 Performance results of several algorithms on multiple problem configurations (mean and standard deviation values of the *avg. offline accuracy*). The different configurations are based on systematic variations of one factor, the problem's change frequency, expressed in the number of evaluations. Boldface values indicate the best algorithm for the given configuration

Change Frequency	**Algorithm 1**	**Algorithm 2**	**Algorithm 3**	**Algorithm 4**
200	**0.630 ±0.03**	**0.632 ±0.03**	**0.631 ±0.03**	**0.630 ±0.03**
500	0.750 ±0.02	0.751 ±0.02	**0.783 ±0.02**	**0.781 ±0.02**
1000	0.811 ±0.02	0.798 ±0.02	**0.886 ±0.02**	**0.886 ±0.02**
1500	0.825 ±0.02	0.854 ±0.02	**0.922 ±0.02**	0.871 ±0.02
2000	0.840 ±0.02	0.859 ±0.01	**0.939 ±0.01**	0.862 ±0.02
2500	0.843 ±0.01	0.871 ±0.01	**0.943 ±0.01**	0.889 ±0.01
3000	0.852 ±0.01	0.880 ±0.01	**0.950 ±0.01**	0.913 ±0.01
3500	0.871 ±0.01	0.901 ±0.01	**0.959 ±0.01**	0.921 ±0.01
4000	0.860 ±0.01	0.906 ±0.01	**0.964 ±0.01**	0.932 ±0.01
4500	0.863 ±0.01	0.910 ±0.01	**0.968 ±0.01**	0.939 ±0.01
5000	0.869 ±0.01	0.911 ±0.01	**0.970 ±0.01**	0.941 ±0.01

now more difficult to grasp and manage, it is nevertheless still understandable (in Fig. 4.4 it is reasonably easy to see which algorithm is the best, and this can also be accomplished in Table 4.4 by enhancing the best algorithm's result using a boldface type). Anyway, it is worth noting that individual differences between each pair of algorithms in the statistical tests are now too lengthy to be shown, since they imply a comparison of the type *all against all* for each problem configuration, which, in general, is not practical for a publication (we are talking of 11 tables like Table 4.3).

Finally, when we consider to simultaneously analyze several factors (e.g., change frequency and severity of change), data grows exponentially, and the presentation in the form of tables and figures becomes intractable. In the literature, some examples of works can be found, where the magnitude of the study and the amount of obtained results force the authors to use such a high number of tables and graphs that the comprehension of the data gets obscured:

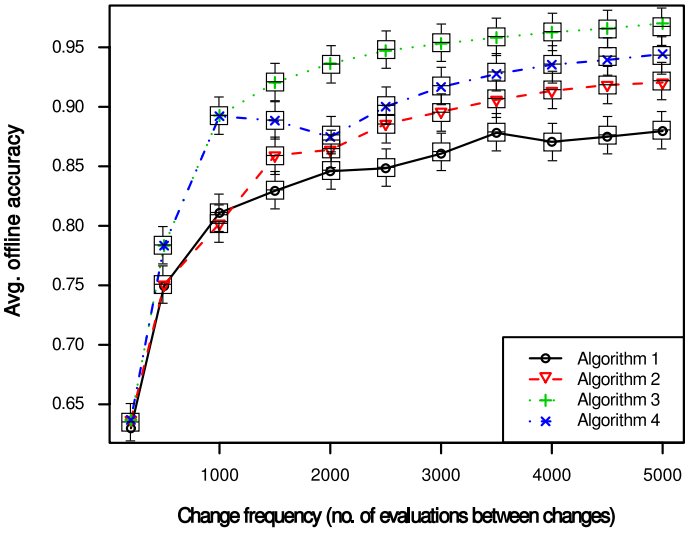

Fig. 4.4 Graphical representation of the results in Table 4.4, where each point corresponds to the results of an algorithm on a configuration of the problem. The results for each configuration are shown using a boxplot of the distribution

- In [23], the author uses 48 graphs and 45 tables of statistical comparisons to analyze the behaviour of 8 different versions of algorithms.
- In [12] a huge number of results are presented in the form of boxplot graphs for several algorithms on a single problem configuration, using 42 graphs for that purpose, although without statistical comparisons tables.
- In [20], the authors compare up to 19 different techniques in 32 tables full of numerical results, using for it an ad-hoc solution based in vectors that compress the information, since they explicitly admit the difficulty in performing so many comparisons.

These are only a small sample of the difficulties that a researcher may find when presenting the results of comparing multiple algorithms, multiple versions of them, multiple problem configurations, or combinations of all the previous. It should be pointed out, however, that each particular case is different from the rest, and that not always such a high number of tables and graphs must imply a bigger difficulty in the understanding of the data. In many cases, the skill of the author for grouping and presenting data is crucial to facilitate its comprehension. However, in general, it is easier for the reader to understand some concise results than some extensive ones.

In order to better confront this situation, we are going to introduce a proposal for compressing the information to be presented using color schemes obtained from the results of the statistical tests.

4.3 SRCS: Statistical Ranking Color Scheme

As it has already been justified in the previous sections, the presentation of the results of several algorithms over variations of multiple factors in a DOP can be problematic. It is necessary to somehow compress the information in order for the reader to be able to capture it and understand it.

The technique we present here, SRCS (Statistical Ranking Color Scheme), has been designed for those situations in which the main interest is to analyze the *relative* performance of the algorithms, rather than the *absolute* one. That is, when we want to establish an ordering or ranking of the algorithms for a given problem configuration.

The first obstacle appears precisely at the moment of establishing that ranking. Ordering the algorithms would be easy if we had a single value to measure their performance, but instead, we have a set of values (one for each independent execution). In order to solve this, we use the output of the statistical tests that tells us if there are significant differences between each pair of samples (for example, using the KW + MWW tests combination of Sect. 4.2).

The way of doing this will be as follows: for a given DOP configuration, all the algorithms begin with an initial ranking of 0. We first compare the results of all the algorithms using a multiple comparison test (e.g., the KW test) in order to determine if there are global differences. In case there are no differences among all, that would be the end of the process, and the algorithms would finish with their initial 0 rank. If, however, significant differences were found, an adjusted pair-wise test (e.g., MWW + Holm) would be performed between each pair of algorithms, in order to assess individual differences. If the pair-wise test says there are significant differences for a given pair of algorithms, the one with the best performance value (the median of the sample) adds $+1$ to its ranking, and the one with the worst value, -1. If there were no differences according to the pair-wise test (a tie), neither algorithm adds anything, but both maintain their previous ranking. At the end, every algorithm will have an associated ranking value, ranging in the interval $[-(N_a-1),+(N_a-1)]$, where N_a is the number of algorithms to be compared. A ranking value of $+r$ for a given algorithm indicates that its performance is significantly better than r algorithms, and a value of $-r$, that it is significantly worse than r algorithms (at the end of this chapter, in the Appendix, we provide an implementation of this ranking calculation using the R programming language).

However, until now we have only shifted the problem, since we have a ranking, but it is still numerical, and therefore, difficult to fully understand when presented in the form of tables if there are too many data. The solution to this comes from human's ability to better manage images and colors than numbers. Starting off from this ranking, we associate a color (for example white) to the maximum ranking value that can be obtained, $+(N_a-1)$, and another very different color (a dark one preferably) to the minimum ranking value that can be obtained, $-(N_a-1)$. All the intermediate ranking values are associated to an interpolated color between the two

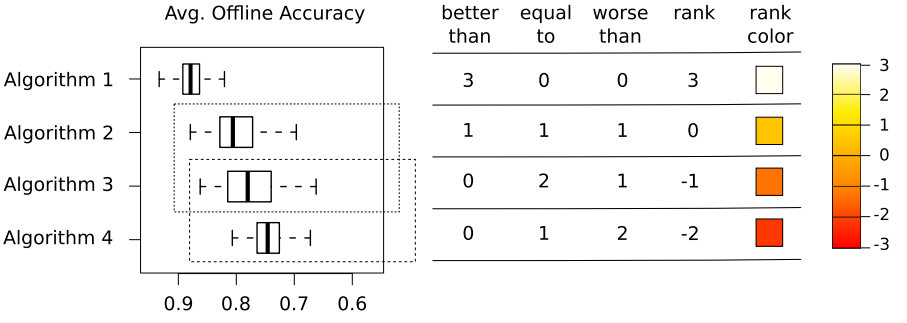

Fig. 4.5 Rank explanation. The boxplot shows the distribution of the performance measures of every algorithm, ordered by its median value. Dotted rectangles indicate those algorithms for which no statistical differences were found at the specified significance level (algorithms 2-3 and 3-4). The table in the right displays, for every algorithm, how many times it shows a significantly better performance ("better than"), no significant differences ("equal to") or significantly worse performance ("worse than") when it is compared with respect to the other 3 algorithms, and its final rank with the correspondent color key.

previous ones. Figure 4.5 explains the calculation of the ranking and the color association of the 4 algorithms we have been using previously, for a given problem configuration.

Color codes obtained from the ranking can now be used to represent the *relative* performance of each algorithm with respect to the others in a graphical way. This representation allows us to visualize the results of many configurations at once, giving the researcher the possibility to identify behavioural patterns of the algorithms more easily.

For example, let us suppose that we have the 4 algorithms of the previous examples, and we want to extend the study of their performance in the MPB with different variations of two factors: *severity*, and *change frequency*. As it has already been justified, presenting the results of these experiments in the form of tables may not be feasible. However, using the SRCS technique, we can arrange the rank colors of each configuration to create the images shown in Fig. 4.6. In this figure, the same color scheme as the one appearing in the explanation in Fig. 4.5 has been used, where a darker color indicates a worse performance, and a lighter one a better. Taking a quick glance at Fig. 4.6, and without having to examine any type of numerical data, we can obtain valuable overall information, like:

- in general, *algorithm 1* is the worst in almost all configurations
- *algorithm 3* has, in almost all configurations, a good or very good performance
- for higher change frequencies (higher number of evaluations between changes), *algorithm 3* is the best
- for lower change frequencies, *algorithm 4* is the best
- variations of the severity have, in general, less influence in the performance of the algorithms than variations of the change frequency

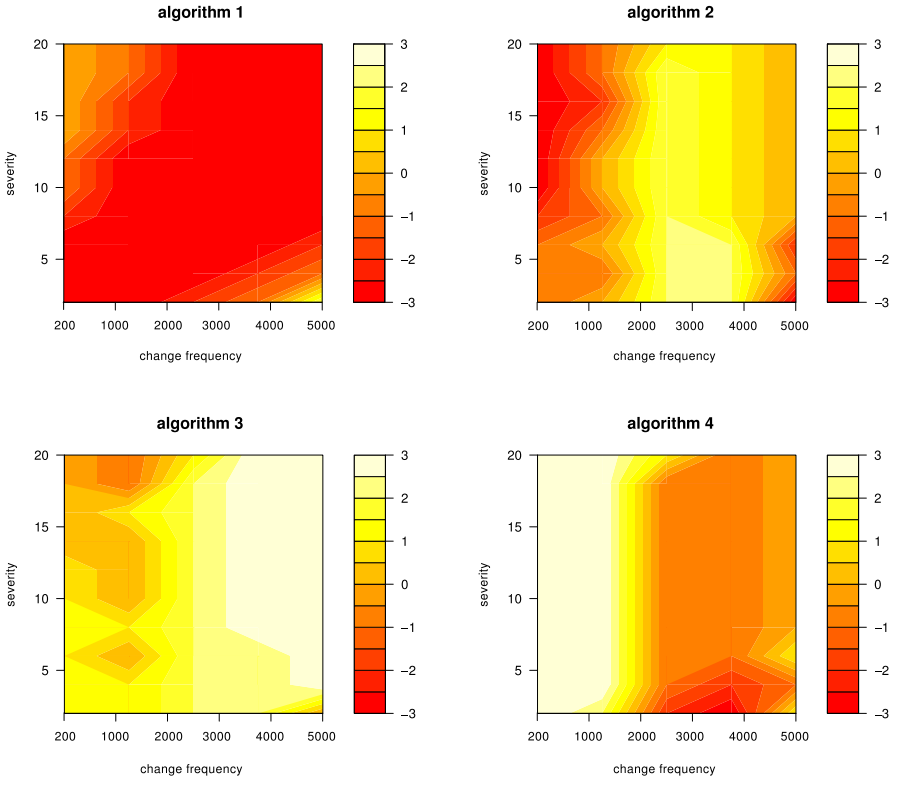

Fig. 4.6 An example of a graphical visualization of the rank-based color scheme for 4 hypothetical algorithms. The visualization shows a comparison of the results of 4 algorithms for different configurations of the factors *severity* and *change frequency* for a problem

Also, figures created using SRCS can be arranged to visualize variations of more than 2 factors (see Fig. 4.7), depending on the practitioner's creativity. These figures can help us to further detect behavioral patterns of the algorithms, and increase our understanding of them.

Finally, although the examples in this chapter used the *avg. offline accuracy* as performance measure, and the KW + MWW combination as statistical tests, the SRCS technique is not restricted to these methods. Other performance measures (avg. offline error, reactivity, etc.) and statistical tests (Friedman, Iman-Davenport, etc) are also valid, as long as their usage is appropriated. In-depth examples of the use of non-parametric statistical tests for comparing optimization algorithms can be found in [8, 10, 11].

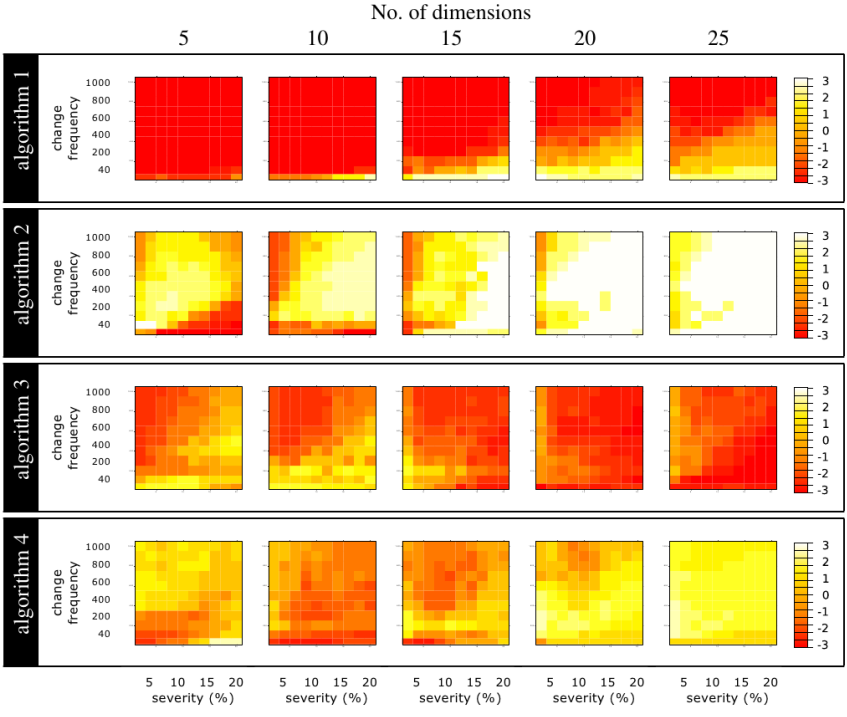

Fig. 4.7 An arrangement of the graphical color scheme of the rankings for visualizing variations of 3 different factors: severity, change frequency, and dimensionality.

4.4 Some Considerations on Statistical Tests

In a statistical hypothesis test, some of the main parameters that determine its outcome are:

- n, the sample size used.
- α, the significance level, or the probability of making a Type I error (*false positive*), i.e., rejecting the null hypothesis H_0 when it is true.
- θ, the effect size, or the minimum difference that can be detected by the test in order to be considered significant, in absolute units.
- π, the power of the test or the sensitivity level, equal to $1-\beta$, where β is the probability of making a Type II error (*false negative*), i.e., accepting the null hypothesis H_0 when it is false.

These parameters are interrelated, and the values of some of them are usually determined from those of the rest, which may be fixed as a result of the experiment's requirements. For example, in a clinical essay for a drug, it could be determined that a minimum increase in blood pressure of 15 mm Hg must be observed in order to

consider its effect of practical significance, with a 99% confidence. Although not explicitely stated, in these types of experiments a minimum power is usually expected (a typical value is 80%). Therefore, in this case, the effect size ($\theta = 15$), the significance level ($\alpha = 0.01$), and the power ($\pi = 0.8$) are fixed, and the sample size should be adjusted in order to obtain those values.

However, when comparing the performance of some algorithms in a synthetic problem (like the MPB used in the examples of previous sections), there is usually no concern about the effect size, since, in practice, it has no real meaning. With no additional information, it cannot be determined if a difference of 0.1 fitness units between two algorithms is significant or not, and therefore, the sample size (N_r in the examples) is not constrained. In this case, unless some external requirements limit the maximun amount of executions, we recommend to use the higher N_r possible, as it will increase the power of the test.

For a more detailed introduction to the use of non-parametric tests and questions on the factors that determine them, the interested reader is referred to [13, 16, 19].

4.5 Conclusions

In this chapter we have presented a new technique, SRCS (Statistical Ranking Color Scheme), specifically designed to analyze the performance of multiple algorithms in DOPs over variations of several factors (e.g., change frequency, severity, dimensionality, etc). This technique is especially well-suited when we want to compare algorithms in a all-vs-all manner, for example, when we want to determine which are the best performing ones in a wide range of scenarios.

SRCS uses statistical tests to compare the performance of the algorithms for a given problem configuration, producing a ranking. Since the results of metaheuristics and non-exact algorithms do not generally follow a normal distribution, non-parametric tests are usually preferred. As a practical guideline, a multiple-comparison test must be performed first, like the Kruskal-Wallis test, in order to determine if there are global differences in the performance of the algorithms. Then, a pair-wise test is used, in order to assess individual differences between algorithm pairs, like the Mann-Whitney-Wilcoxon test. This pair-wise test must be adjusted in order to compensate for the family-wise error derived from the performance of multiple comparisons, using, for example, Holm's method. However, these tests are only suggestions that do not affect the way in which SRCS works, and other options can be used (Friedman's test, Iman-Davenport, etc).

The ranking produced is later used to associate color codes to each algorithm result, such that the *relative* performance of each algorithm with respect to the others can be represented in a graphical way. This representation allows us to visualize the results of many algorithms on many configurations in a much more compact way by enhancing differences between the results, and giving thus the researcher the possibility of identifying behavioural patterns more easily.

Like any information compressing technique, SRCS left out part of the information, so its use, either isolated or as a complement to other traditional ways for displaying results (tables and plots), should be evaluated in each case. With SRCS, using rankings for stressing out the differences among algorithms implies not displaying absolute performance values.

Acknowledgements. This work has been partially funded by the project TIN2008-01948 from the Spanish Ministry of Science and Innovation, and P07-TIC-02970 from the Andalusian Government.

Appendix

In this Appendix we provide an implementation of the ranking method of Sect. 4.3, using the R programming language [1].

```
#-----------------------------------------------------------------------------------------------------------
# The function for calculating the rank of a set of algorithms using their performance results on a
# *single* problem configuration. Returns a list indexed by algorithm, with the corresponding rank value
# for each of them.
# Parameters:
# - data: a vector with the results of all the algorithms, that is, the 'n' independent repetitions of
#     the performance measure, for each algorithm.
# - group: a vector of factors for the data, indicating, for each corresponding entry of the data vector,
#     the algorithm it belongs to.
# - alpha: the minimum p-value for the tests to assess a significant difference. Defaults to 0.05
# - max: TRUE or FALSE. TRUE means the higher the performance measure, the better. FALSE means the
#     opposite. For example, if the performance measure is the error, set max = FALSE; if the performance
#     measure is the accuracy, set max = TRUE. Defaults to TRUE.
# Example input:
# - data <- c( 2.5, 2.3, ..., 1.2, 0.7, ..., 3.5, 4.1 )
# - group <- factor( "alg1", "alg1", ..., "alg2", "alg2", ..., "alg3", "alg3" )
# Example output:
# - rankList : [ ["alg1"][0], ["alg2"][2], ["alg3"][-1] ]
#-----------------------------------------------------------------------------------------------------------
rank <- function( data, group, alpha=0.05, max=TRUE ) {
    # initialize the ranks to 0
    algorithms <- unique( group )
    rankList <- list()
    for( algorithm in algorithms ) {
        rankList[[algorithm]] <- 0
    }

    # calculate the vector of medians for all the algorithms' measures
    medians <- tapply( data, group, median )

    # perform a Kruskal-Wallis test to assess if there are differences among all the results
    dataframe <- data.frame( group, data )
    kruskal <- kruskal.test( data ~ group, data=dataframe )
    if( !is.na( kruskal$p.value ) && kruskal$p.value < alpha ) {

        # post-hoc test: perform a pairwise Mann-Whitney-Wilcoxon (MWW) rank sum test
        # with Holm correction to assess individual differences
        wilcoxon <- pairwise.wilcox.test( data, group, p.adj="holm", exact=FALSE )

        for( algorithm1 in rownames( wilcoxon$p.value ) ) {
            for( algorithm2 in colnames( wilcoxon$p.value ) ) {
                if( !is.na( wilcoxon$p.value[algorithm1,algorithm2] ) &&
                    wilcoxon$p.value[algorithm1,algorithm2] < alpha ) {

                    # there is a significant difference between algorithm1 and algorithm2;
                    # we need to identify which one is the best and which one the worst,
                    # we'll use the median for that purpose, since it is coherent with the
                    # use of the MWW method, which also uses medians
                    if( medians[algorithm1] > medians[algorithm2] ) {
                        best <- algorithm1
                        worst <- algorithm2
                    } else {
                        best <- algorithm2
```

[1] http://www.r-project.org/

```
                        worst <- algorithm1
                    }

                    # if max==FALSE, swap best and worst
                    if( !max ) {
                        tmp <- best
                        best <- worst
                        worst <- tmp
                    }

                    # update ranks
                    rankList[[best]] <- rankList[[best]] + 1
                    rankList[[worst]] <- rankList[[worst]] - 1
                }
            }
        }
    }

    return( rankList )
}
```

References

[1] Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation: The New Experimentalism. Natural Computing Series. Springer, Heidelberg (2006), doi:10.1007/3-540-32027-X

[2] Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.): Experimental Methods for the Analysis of Optimization Algorithms. Springer, Heidelberg (2010), doi:10.1007/978-3-642-02538-9

[3] Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the 1999 IEEE Congress on Evolutionary Computation (CEC 1999), vol. 3, pp. 1875–1882. IEEE (1999), doi:10.1109/CEC.1999.785502

[4] Branke, J.: Evolutionary Optimization in Dynamic Environments. Genetic algorithms and evolutionary computation, vol. 3. Kluwer Academic Publishers, Massachusetts (2001)

[5] Chen, C.-H., Härdle, W., Unwin, A., Friendly, M.: Handbook of Data Visualization. Springer Handbooks of Computational Statistics. Springer, Heidelberg (2008), doi:10.1007/978-3-540-33037-0

[6] Cruz, C., González, J., Pelta, D.: Optimization in dynamic environments: a survey on problems, methods and measures. In: Soft Computing, pp. 1–22 (2010), doi:10.1007/s00500-010-0681-0

[7] De Jong, K.: An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, Ann Arbor, MI, USA (1975)

[8] Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7(1) (2006)

[9] Fonseca, V.G., Fonseca, C.M.: The attainment-function approach to stochastic multiobjective optimizer assessment and comparison. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.) Experimental Methods for the Analysis of Optimization Algorithms, pp. 103–130. Springer, Heidelberg (2010), doi:10.1007/978-3-642-02538-9_5

[10] García, S., Herrera, F.: An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. Journal of Machine Learning Research 9, 2677–2694 (2008)

[11] García, S., Molina, D., Lozano, M., Herrera, F.: 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. Journal of Heuristics 15(6), 617–644 (2009), doi:10.1007/s10732-008-9080-4

[12] Gräning, L., Jin, Y., Sendhoff, B.: Individual-based management of meta-models for evolutionary optimization with application to three-dimensional blade optimization. In: Yang, S., Ong, Y.-S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. SCI, vol. 51, pp. 225–250. Springer, Heidelberg (2007), doi:10.1007/978-3-540-49774-5_10

[13] Hollander, M., Wolfe, D.: Nonparametric Statistical Methods, 2nd edn. John Wiley & Sons, Inc. (1999)

[14] Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6(2), 65–70 (1979)

[15] Kruskal, W.H., Allen Wallis, W.: Use of ranks in one-criterion variance analysis. Journal of the American Statistical Association 47(260), 583–621 (1952)

[16] Russell, V.: Lenth. Some practical guidelines for effective sample size determination. The American Statistician 55(3), 187–193 (2001), doi:10.1198/000313001317098149

[17] López-Ibáñez, M., Paquete, L., Stützle, T.: Exploratory analysis of stochastic local search algorithms in biobjective optimization. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.) Experimental Methods for the Analysis of Optimization Algorithms, pp. 209–222. Springer, Heidelberg (2010), doi:10.1007/978-3-642-02538-9_9

[18] Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. The Annals of Mathematical Statistics 18(1), 50–60 (1947), doi:10.1214/aoms/1177730491

[19] Randles, R.H., Wolfe, D.: Introduction to the Theory of Nonparametric Statistics. John Wiley & Sons, Inc. (1979)

[20] Reyes-Sierra, M., Coello, C.: A study of techniques to improve the efficiency of a multi-objective particle swarm optimizer. In: Yang, S., Ong, Y.-S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. SCI, vol. 51, pp. 269–296. Springer, Heidelberg (2007), doi:10.1007/978-3-540-49774-5_12

[21] Weicker, K.: Performance Measures for Dynamic Environments. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 64–73. Springer, Heidelberg (2002), doi:10.1007/3-540-45712-7_7

[22] Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin 1(6), 80–83 (1945), doi:10.2307/3001968

[23] Yang, S.: Explicit memory schemes for evolutionary algorithms in dynamic environments. In: Yang, S., Ong, Y.-S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. SCI, vol. 51, pp. 3–28. Springer, Heidelberg (2007), doi:10.1007/978-3-540-49774-5_1