Analyzing the BBOB Results by Means of Benchmarking Concepts

O. Mersmann olafm@statistik.tu-dortmund.de
Chair of Computational Statistics, TU Dortmund University, Germany

M. Preuss mike.preuss@uni-muenster.de
Chair of Information Systems and Statistics, University of Muenster, Germany

H. Trautmann trautmann@wi.uni-muenster.de
Chair of Information Systems and Statistics, University of Muenster, Germany

B. Bischl bischl@statistik.tu-dortmund.de
Chair of Computational Statistics, TU Dortmund University, Germany

C. Weihs weihs@statistik.tu-dortmund.de
Chair of Computational Statistics, TU Dortmund University, Germany

Abstract

We present methods to answer two basic questions that arise when benchmarking optimization algorithms. The first one is: which algorithm is the "best" one? and the second one: which algorithm should I use for my real world problem? Both are connected and neither is easy to answer. We present a theoretical framework for designing and analyzing the raw data of such benchmark experiments. This presents a first step in answering the aforementioned questions. The 2009 and 2010 BBOB benchmark results are analysed by means of this framework and we derive insights regarding the answers to the two questions. Furthermore, we discuss how to properly aggregate rankings from algorithm evaluations on individual problems into a consensus, its theoretical background and which common pitfalls should be avoided. Finally, we address the grouping of test problems into sets with similar optimizer rankings and investigate whether these are reflected by already proposed test problem characteristics, finding that this is not always the case.

Keywords

evolutionary optimization, benchmarking, exploratory landscape analysis, BBOB test set, multidimensional scaling, consensus ranking

1 Introduction

In the domain of stochastic optimization, the available theory is still too limited to predict the performance of different algorithms on more than the most simple objective functions. Therefore, benchmarking plays a vital role in developing and comparing these types of algorithms. The BBOB 2009 and 2010 (Hansen et al. (2009a) and Hansen et al. (2009b) and follow-ups for 2010) results provide the most sophisticated benchmarking data currently available, and their specific strength lies in the inclusion of many classic and modern optimization algorithms from fields outside of evolutionary computation. Although a result summary has already been published by the organizers (Auger et al. (2010)), much more can be learned from the available data when inspected from a slightly different angle, namely the one of benchmarking theory (see Sec. 2), a long since existing branch of statistics. A first attempt in this direction has been Mersmann et al. (2010a), solely considering the BBOB 2009 data. In this work, we aggregate

the data from 2009 and 2010 and expand the portfolio of applied statistical techniques. Due to the high number of algorithms from both conferences (64) it also became inevitable to group the algorithms according to their main idea, choosing only one representative of each group for comparison as documented in Sec. 3.3. Such grouping is of course subjective, however we have tried to be as fair as possible, so as to obtain a good overview of the capabilities of the different classes of algorithms currently available on the BBOB test problems.

Who is the *winner* of an optimization competition and does this question make any sense? Without clearly qualifying one method as the overall winner, Hansen et al. (2010) suggest to look at such measures as the number of problems solved satisfactorily over the number of evaluations. This perspective implies that the most successful algorithms can solve the largest possible fraction of problems in the shortest time, thereby suggesting a default method which shall be employed if no problem knowledge is available. There may however be algorithms which are especially well suited for problems that cannot be solved well by the winning method. One may further argue that some problem properties are available even for a black-box problem of unknown structure, e.g. its dimensionality. Or preliminary runs with standard (gradient based) methods might have hinted at a unimodal or multimodal structure of the function landscape.

Thus, we can state that: a) looking at different classes of problems makes sense, and b) choosing a best algorithm (possibly for only a subgroup), requires to find a ranking for the aggregation of the data of the according test functions. The latter is made possible by using consensus ranking methods which we introduce in Sec. 2.2. Unfortunately, there cannot be a consensus ranking method that satisfies a complete set of reasonable, intuitive prerequisites simultaneously. Instead, we will have to make a subjective choice which will influence our interpretation of the resulting consensus. These methods, while not particularly well known in the EC community, are often used in other disciplines and should not be dismissed as exotic. Examples of the applications of consensus methods in everyday life are sporting events where several rankings are produced by a panel of judges, races etc. and then averaged to find the winner of the competition.

The ultimate goal of a comparison of methods on a benchmark suite should be to detect which optimization algorithm to use for a given practical problem, but clearly, not every imaginable problem can be represented by a benchmark. Additionally, benchmarking also suits the purpose of algorithm development as one may try to enhance a method that does not work too well on some problems. Therefore it is essential that we are able to determine the difficulty that a problem poses for an algorithm. In any case, some generalization from single problems to problem groups is necessary. The BBOB test set is partitioned into five groups according to very general properties. But do these problem groups match the abilities of the optimization algorithms under test well? Or shall they be restructured according to other criteria? We treat this question in Sec. 3.5, coming to the conclusion that the groups are largely well chosen, but with some exceptions.

The assignment of problems to groups is also interesting from another perspective: If one is confronted with a new problem of which not much is known, it would be desirable to automatically detect some of its properties and then sort it into an existing group for which good optimization algorithms are available. This is our long term goal, and we have coined the term *exploratory landscape analysis* (ELA) to describe this methodology. First approaches are presented in Mersmann et al. (2010a) and further work in this direction can be found in Mersmann et al. (2011). In this work however, we concentrate on the manual analysis of the BBOB workshop data to obtain a "ground truth", to which future ELA approaches can be compared to (Sec. 3). Finally, we provide a summary and some conclusions in Sec. 5.

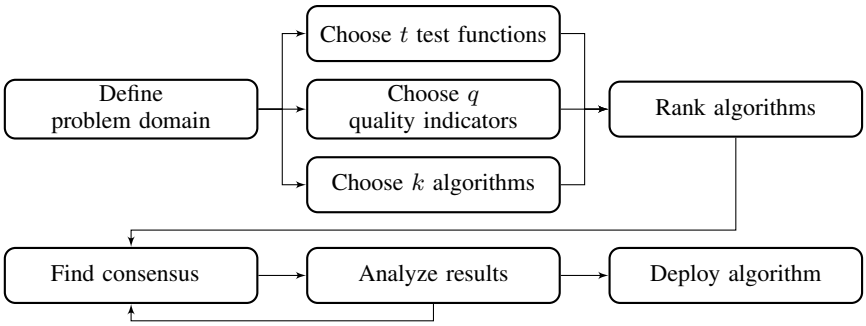

Figure 1: Flow chart describing the steps involved in a benchmark experiment.

2 Benchmarking

Benchmarking experiments and comparisons are set up in order to evaluate the performance of different algorithms on given problem classes. To derive an overall ranking of all the algorithms in a benchmark experiment is one of the common goals. This has several uses, the most prominent one being the identification of an "overall best" algorithm. Without loss of generality we will focus on optimization algorithms in the following description. A specific position in an overall ranking will in general be dependent on the rank aggregation method employed and there is no consensus method which can be considered optimal as has been known in econometrics for several decades (Arrow, 1950). So all methods presented here will have to be a trade-off between three basic properties which any optimal consensus method should satisfy.

Fig. 1 schematically visualizes the general setup of what we call a benchmark experiment. We will see that its outcome strongly depends on the chosen performance measures and ranking procedures (Hornik and Meyer, 2007; Mersmann, 2009; Mersmann et al., 2010b). The definition of a problem domain together with a set of t test functions is a key step. Benchmarking results can only be generalized to the considered domain, and even this is not admissible if the functions are not chosen systematically and therefore represent the problem domain properly. Ideally, this would be guaranteed by applying the design of experiments methodology to the most important features which characterize the function types, but this is a nontrivial task for several reasons. First of all, the relationship between these features and the measured performance of an algorithm is generally not linear. Secondly and more importantly, this would require that we have a method to randomly sample from the set of all functions which satisfy some constraints on their characteristics. This is currently an unsolved issue.

To do this performance assessment, q (ideally stochastically independent) quality indicators or performance measures are chosen to judge different aspects of algorithm performance and optimization quality. Then, a set of k algorithms is chosen for the benchmark. This set should be diverse and care should be taken not to over-represent a certain class of algorithms since this may bias the consensus rankings derived later on.

Since the outcome of the algorithms under test is usually stochastic, they are compared by aggregating r independent runs. Appropriate choices of r depend on the expected difference in quality compared to the variance of the observed quality indicator values. A good rule of thumb would be a value between 10 and 25 repetitions, but during the analysis it may become apparent that more runs are required to adequately differentiate between the algorithms (Mersmann, 2009). Here the trade-off is between the speed of running such an experiment and the accuracy of the results.

Our k algorithms will now be run r times for each of the t test functions resulting in $r \times t$

Scenario	Summary statistic (s_i)	R
Best case quality	$\max\{I_{i,1}, \ldots, I_{i,r}\}$	$\geq$
Average case quality	$r^{-1} \sum_{j=1}^{r} I_{i,j}$	$\geq$
Median quality	$median(I_{i,1}, \ldots, I_{i,r})$	$\geq$
Worst case quality	$\min\{I_{i,1}, \ldots, I_{i,r}\}$	$\geq$
Consistent quality	$(r-1)^{-1} \sum_{j=1}^{r} (I_{i,j} - \bar{I}_i)^2$	$\leq$

Table 1: List of summary statistics and order relations R.

values of the q quality indicators for each algorithm. Using these values we can derive individual rankings of the algorithms for each combination of a quality indicator and a test function (see Sec. 2.1). An overall (consensus) ranking can then be generated from the individual rankings or from subsets of these. Different methods for this are introduced and discussed in Sec. 2.2.

2.1 Individual Ranking

Benchmarking theory is often based on the theoretical framework of relations and orders (Hunter (2008)) from which a formal definition of a *ranking* R of a set of items $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ can be derived as a weak order over the set $\mathcal{A}$. For $a_i, a_j \in \mathcal{A}$ we say a_i is better than or equal to a_j and denote this by $a_i \succ a_j$ iff $a_j \, R \, a_i$. If $a_i \, R \, a_j$ and $a_j \, R \, a_i$, then we say a_i and a_j are tied and denote this by $a_i \sim a_j$. If R is a linear order, we refer to the corresponding ranking as a *strict ranking*.

Initially, without loss of generality, we will consider the case of a fixed test function f and quality criterion I to be maximized. For example, I could be the dominated hypervolume indicator (HV, Zitzler and Thiele (1998)) in multiobjective evolutionary optimization or the accuracy a single-objective evolutionary algorithm has reached after a fixed number of function evaluations. If we only had two algorithms, a_1 and a_2, then we could define $\succ$ by saying $a_1 \succ a_2$ (a_1 is better than or equal to a_2) if $I(a_1(f)) \geq I(a_2(f))$. Generalizing this result to a higher number of algorithms is straightforward in that we use the order induced by I on the algorithms as our ranking. However, due to the inherent stochastic nature of the algorithms, $I(a_i(f))$ becomes a random variable with unknown distribution. We will therefore estimate some properties of this distribution, usually the expected value or some quantile, from the r repetitions we performed. Choosing a good summary statistics requires an initial study of the distribution of the quality indicator. An example of such an analysis is given in Mersmann et al. (2010c) where the distribution of the HV is studied for different evolutionary multiobjective algorithms. Among other characteristics it is shown to be unimodal in most cases.

The usage of a summary statistic of the indicator distribution together with the classic "greater than or equal" ($\geq$) or "less than or equal" ($\leq$) relation is a straightforward approach to generate a linear order on the algorithms under test. Table 1 lists suitable summary statistics assuming fixed samples of r quality indicator values $I_i = (I_{i,1}, \ldots, I_{i,r})$. However, any uncertainty in these statistics is neglected which could lead to perturbed ranking results where the extent of the error depends on the variance of the indicator distribution. Another possibility is to use statistical hypothesis tests (Mood et al., 1974) to decide if the locations of two quality indicator distributions are different in some sense. Caution is in order here because this approach can lead to a relation $\succ$ that is not transitive and antisymmetric. More details are given in Mersmann (2009) and Mersmann et al. (2010b).

2.2 Consensus Ranking

If we want to find the best algorithm out of a given set by using the individual rankings obtained using the methodology described above, we need to aggregate these into a single ranking. This is called finding a *consensus* among the individual rankings and the result is a *consensus ranking*. As mentioned earlier, there is no single best consensus method for rankings and we will elaborate on this here.

One can postulate several criteria that a "best" consensus method cm should fulfill (Arrow (1950)):

1. A consensus method cm that takes into account all rankings instead of mimicking one predetermined ranking is said to be *non-dictatorial*.

2. A cm that, given a fixed set of rankings, deterministically returns a complete ranking is called a *universal* consensus method or is said to have a *universal domain*.

3. A cm fulfills the *independence of irrelevant alternatives criterion*, short *IIA criterion*, if given two sets of rankings $\mathcal{R} = \{r_1, \ldots, r_n\}$ and $\mathcal{T} = \{t_1, \ldots, t_n\}$ in which for every $i \in \{1, \ldots, n\}$ the order of two algorithms a_1 and a_2 in r_i and t_i is the same, the resulting consensus rankings rank a_1 and a_2 in the same order. IIA means that introducing a further algorithm does not lead to a rank reversal between any of the already ranked algorithms which is a very strict requirement.

4. A cm which ranks an algorithm higher than another algorithm if it is ranked higher in a majority of the individual rankings for which the consensus ranking is sought, fulfills the *majority criterion*.

5. A cm is called *Pareto efficient* if given a set of rankings in which for every ranking an algorithm a_i is ranked higher than an algorithm a_j, the consensus also ranks a_i higher than a_j.

Unfortunately, all criteria cannot be met simultaneously because the IIA criterion and the majority criterion are incompatible if we assume a non-dictatorial consensus method. Thus, consensus approaches will yield different results with respect to the criteria chosen to be fulfilled. At this point one might ask why we even bother to find a consensus if it will always be a trade-off between the above criteria. The reason is, that while it may not be optimal in some sense, it still gives us insight into which algorithms might be worth further investigation and which algorithms perform rather poorly. However, we will have to take care that no (accidental or even intentional) manipulation of the consensus takes place. This might easily happen if the IIA is not fulfilled - which it usually is not. Therefore simply adding similar algorithms to the benchmark can increase the chance of a rank reversal.

Generally, we can differentiate between positional and optimization based methods. Positional methods calculate sums of scores s for each algorithm a_i over all rankings and result in an ordering as follows: for t test functions and q quality indicators considered where $r_{i,j}$ denotes the ranking induced by the i-th function and the j-th comparison or quality indicator:

$$a_i \succ a_j \iff s_i > s_j, \quad a_i \sim a_j \iff s_i = s_j, \quad \text{with } s_i = \sum_{k=1}^{t} \sum_{\ell=1}^{q} s(a_i, r_{k,\ell}). \tag{1}$$

The simplest score function assigns a value of one to the best algorithm in each ranking while all other algorithms get a value of zero. Though this is somehow intuitive, undesirable consensus rankings can occur. Consider the situation with two different rankings of three algorithms, i.e.

$R_1 \colon a_1 \succ a_2 \succ a_3$ as well as $R_2 \colon a_3 \succ a_2 \succ a_1$. After calculating the scores the consensus would be $[a_1 \sim a_3] \succ a_2$ although the rankings are directly opposed and we would intuitively expect to have $[a_1 \sim a_2 \sim a_3]$.

The Borda count method (de Borda (1781)) accounts for this drawback and assigns an algorithm one point for each algorithm that is not better than the algorithm considered, i.e. $s^{BC}(a_i, r) = \sum_{i \neq j} \mathbf{I}(a_i \succ a_j)$ which in the case of no ties reduces to the ranks of the data. Unfortunately, the Borda method does not fulfill the majority or the IIA criterion. It is still a popular consensus method because it can be easily implemented and understood. The main criticism voiced in the literature is that it implicitly, like all positional consensus methods, assumes a distance between the positions of a ranking – which is equally spaced in case of the Borda method.

Optimization based methods on the other hand require a distance function (see Cook and Kress (1992) for on overview) between different rankings which quantifies how much two rankings deviate from each other. Central to this is a notion of *betweenness*, expressed by pairwise comparisons. I.e. a ranking r_2 lies between r_1 and r_3 if for all pairs of algorithms either r_2 agrees with r_1 or r_3 on the relative order of the pair, or r_1 and r_3 have conflicting orderings for the pair and r_2 declares the pair to be tied. From this, we obtain a geometry on the set of all possible rankings and can ask for something like a mean (minimizing squared loss) or median (minimizing absolute loss) consensus ranking.

If we add the requirement of non-negativity, symmetry and the triangle inequality to the list of properties our distance function should fulfill, we can formalize the above notion of a mean or median consensus by taking the distance measure as our loss function and then minimizing over all admissible consensus rankings $\mathcal{C}$:

$$\arg\min_{c \in \mathcal{C}} L(c) = \sum_{i=1}^{t} \sum_{j=1}^{q} d(r_{i,j}, c)^{\ell} \quad \ell \geq 1. \tag{2}$$

The consensus ranking is then given by the ranking whose loss L is minimal. Setting $\ell = 1$ results in what is called a median consensus ranking and $\ell = 2$ results in a mean consensus ranking.

Kemeny and Snell (1972) postulated meaningful axioms for distance functions which can be proven to uniquely lead to the symmetric difference (*SD*) which counts the cases where $a_i \succ a_j$ is contained in one of the relations but not the other:

$$d^{SD}(r_1, r_2) := \vec{1}' |\vec{I}^{r_1} - \vec{I}^{r_2}| \vec{1} \tag{3}$$

where $\vec{1}$ is the k dimensional one vector, $\vec{I}^{r_i}$ the incidence matrix belonging to the relation that corresponds to the ranking r_i and $|\cdot|$ denotes the element wise absolute value. SD/L denotes the *SD* approach for the set of all linear and SD/O for the set of all partial orders.

Unfortunately, we cannot give a general recommendation regarding the introduced consensus methods (Saari and Merlin (1997)) as each method offers a different trade-off of the consensus criteria. The SD/L and SD/O methods meet the *majority criterion* and thus cannot meet the *IIA criterion* simultaneously. However, on real data they rarely result in rank reversals if algorithms are added or dropped. The Borda count (BC) method does not fulfill either of these criteria. Saari and Merlin (1997) show that the SD method always ranks the Borda winner above the Borda loser and that the Borda method always ranks the SD winner above the SD loser.

It is important to notice that consensus rankings generally do not admit nesting in a hierarchical structure. For example, separate consensus rankings could be of interest for test functions with specific features, e.g. high multimodality or convexity. While this certainly is a valid and

meaningful approach one has to keep in mind that an overall consensus of these separate consensus rankings does not necessarily have to equal the consensus ranking directly generated based on all individual rankings.

3 BBOB Analysis

In the following sections we will apply the benchmarking framework which was presented in the previous section to the joint results of the 2009 (Hansen et al., 2010) and 2010 BBOB open benchmark (Auger et al., 2010). Our aim will not be to identify the "best" algorithm but try to characterize the performance of different algorithm classes. This will allow us to deduce a small subset of algorithms in Sec. 3.3 which together might form the basis of a practitioner's black-box optimization toolbox.

3.1 Benchmark Setup

In this section we will give a short overview of the BBOB 2009 and 2010 open benchmarks.For a detailed description of the experimental setup see Hansen et al. (2009a). The general setup used by the BBOB team follows the methodology depicted in Fig. 1 but instead of choosing the k algorithms themselves, since this is an open benchmark, researchers are invited to submit results for their algorithms. It uses a balanced and unbiased sample of the published set of test functions from the field of black-box optimization. Their characteristics have been studied by the BBOB team to ensure that different aspects and difficulties are covered.

To assess the performance of an algorithm, the BBOB team proposes the use of the so called *expected running time*. This measure estimates the expected number of function evaluations required to achieve an accuracy of $\varepsilon > 0$. For a given ε the ERT is defined as

$$\mathbf{E}\left\{RT(\varepsilon)\right\} := \mathbf{E}\left\{N_{\text{eval}}^{\text{succ}}(\varepsilon)\right\} + \frac{1-\pi^{\text{succ}}(\varepsilon)}{\pi^{\text{succ}}(\varepsilon)}\,\mathbf{E}\left\{N_{\text{eval}}^{\text{fail}}(\varepsilon)\right\}, \tag{4}$$

where $N_{\text{eval}}^{\text{succ}}(\varepsilon)$ denotes the number of function evaluations until the algorithm reaches the desired accuracy of ε, $N_{\text{eval}}^{\text{fail}}(\varepsilon)$ denotes the number of function evaluations until the algorithm terminates without reaching the desired accuracy level (unsuccessful run) and $\pi^{\text{succ}}(\varepsilon)$ is the probability of a successful run. We will estimate the ERT from the r runs performed by every algorithm on each test function by plugging in the empirical equivalents of the unknown parameters. For a thorough motivation of the ERT see Hansen et al. (2005). There are certainly other measures and ways to characterize algorithm performance. One could for example ask for the accuracy attained after a fixed budget of function evaluations. In this analysis we will however focus on the ERT and restrict ourselves to the performance metric originally suggested by the BBOB team.

In order to estimate the ERT, each contestant is required to submit 15 runs of his or her algorithm for each of the 24 test functions. It was required to submit results for 2, 3, 5, 10 and 20 dimensional parameter spaces. Results for 40 dimensional parameter spaces were optional. The experimenter may therefore have to perform up to $15 \times 24 \times 6 = 2160$ runs. The results of each run are automatically stored in a file by the BBOB framework. From this file it is possible to infer the number of function evaluations used for almost any accuracy level ε. There is one small difference in the way the 15 runs are composed between the 2009 and 2010 BBOB instance. In 2009, the 15 runs were divided among 5 different test function instances[1] for each of which 3 runs had to be performed. In 2010, instead of 5 instances, 15 instances with just one run per instance were required.

[1] Slight reparameterizations of the test function obtained by rescaling, rotating or otherwise transforming the parameter vector before applying the function.

Algorithm	Number of runs
ALPS	15 – 45
BFGS	15 – 30
(1+1)-CMA-ES	15 – 30
DIRECT	5 – 5
AVGNEWUOA	15 – 30
MA-LS-CHAIN	15 – 29
MCS	15 – 30
(1+1) ES	15 – 26
Artificial Bee Colony	15 – 33

Table 2: Algorithms for which there are deviations from the 15 runs per test function / dimension rule. The second column shows the minimal and maximal number of runs performed per test function / dimension combination.

The way the BBOB team analyses the results turned in by the contestants is different from what we propose in Sec. 2. We will therefore refer the reader to Hansen et al. (2010) for a description of their methodology and the results they obtain. Instead we will use the raw data, graciously provided by BBOB team on their website[2], to conduct an analysis based on the methods proposed in Mersmann et al. (2010b).

Before we begin with the analysis we would like to mention a few discrepancies between the actual data available and what should be included in a contestant's submission. Not all contestants have turned in results for each dimension or test function. This is legitimate but hinders some analysis. For example very few algorithm runs in 40D are available. We can only speculate if the other contestants did not turn in results because their algorithms did not perform well in this dimension or because they did not have enough time / resources to perform the additional runs. Another problem is that some results do not contain the required 15 runs per test function / dimension combination. In fact, some authors turned in *more* runs than required! These findings are summarized in Tab. 2. One should note that DIRECT is a deterministic method which was submitted in 2009 when three replications for each function were required by the BBOB rules. It therefore does abide by the rules since each repetition would have produced the same result. To avoid any biasing of the results we have opted to use a form of stratified sampling that either chose 15 unique instances (2010 submission) or three runs from five instances (2009 submission) for those submissions with more than 15 runs. Other issues encountered include backup files in the submitted archives and differing directory structures between contestants. In the future it would be desirable to standardize the layout of the submitted data to ease external analysis.

3.2 Individual Rankings

Initially, algorithm rankings for each test problem and dimension combination are generated for all accuracy levels 10^{-i} for $i = 3, 4, 5, 6, 7, 8$. An exemplary visualization of rankings obtained for the maximum accuracy level of 10^{-8} is shown in Fig. 2. All rankings are obtained by ranking based on the ERT using the $\leq$ (i.e. smaller is better) relation.

From the plot, we can see that it is not advisable to simultaneously analyze all competing algorithms. Obviously, figures are not very meaningful for 55 algorithms due to information overload. Due to this limitation and the inherent risk of including multiple variants of one algorithm in an analysis, as described in the previous section, algorithm groups are defined which consist of a number of very similar algorithms, better denoted algorithm variants. For instance, there are 17 variants of the CMA-ES which differ only in population size and restart strategy. In

[2] `http://coco.gforge.inria.fr/doku.php`

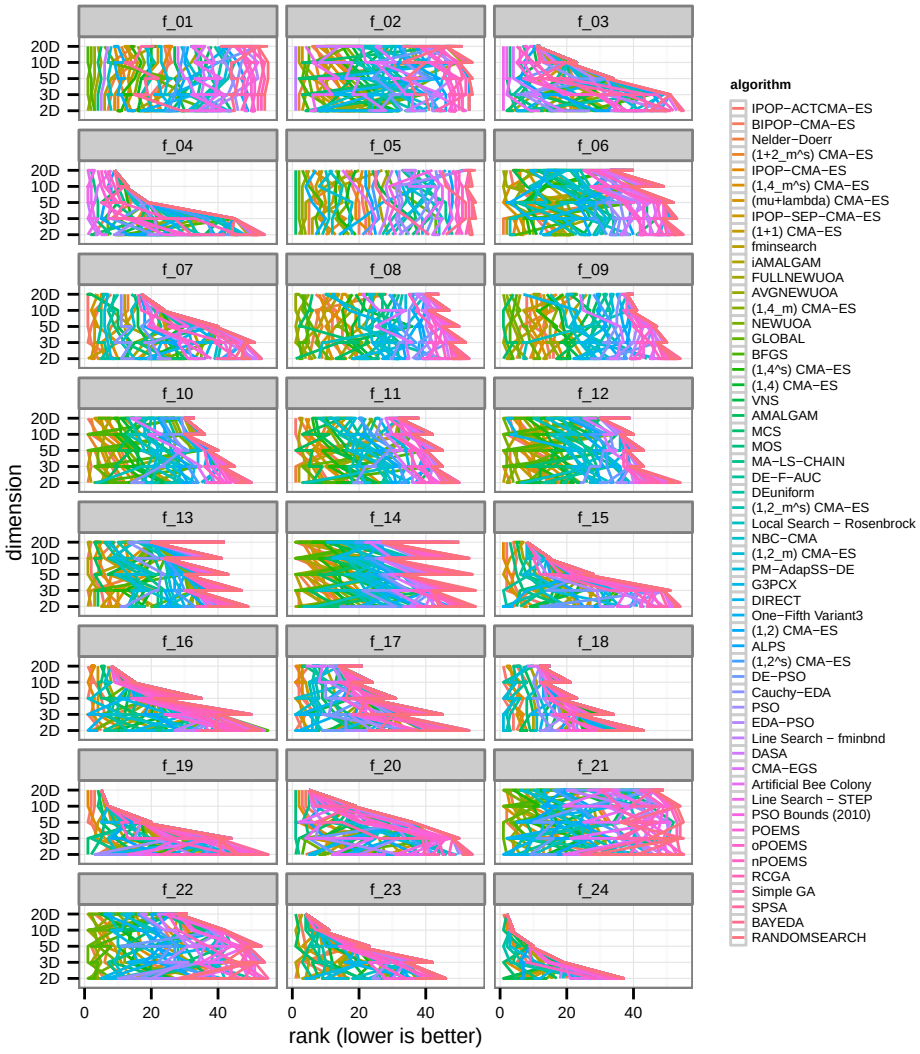

Figure 2: Parallel coordinate plot for each function, showing the rank of each algorithm as the number of dimensions rise for the accuracy level of 10^{-8}.

Sec. 3.3 we discuss why an inclusion of all such algorithm variants can extremely bias resulting consensus rankings. Therefore we will choose a "best" variant from each algorithm group and use this algorithm as a representative for further analysis and especially the final consensus rankings.

An aggregation on the test problem level would be possible as well. In Hansen et al. (2009b) the BBOB test problems are grouped into predefined problem classes with specific properties as a) unimodal separable problems (UM - Sep., f_1-f_4) , b) unimodal low or moderate conditioned problems (UM - Low C, f_6-f_9), c) high conditioned and unimodal problems (UM - High C, f_{10}-f_{14}), d) multi-modal problems with adequate global structure (MM - adeq. GS, f_{15}-f_{19}), and e) multi-modal problems with weak global structure (MM - weak GS, f_{20}-f_{24}). As already shown in Mersmann et al. (2010a) and obvious from Fig. 2 the algorithm performance is not consistent enough within the specified function groups to justify the selection of a reduced set of representative test problems.

Some general statements about the algorithms' performance can however be extracted from Fig. 2. It is evident [3] that there is a change in the general performance of the algorithms as the dimension rises, e.g. not surprisingly the performance of gradient based methods like BFGS in most cases deteriorates with increasing dimension. On the other hand, there is an opposite tendency for the CMA-ES variants. Only rarely the set of the best ranked algorithms is quite stable across all dimensions as for the sphere problem f_1 and the rotated Rosenbrock problem f_9 for instance. Interestingly, a few algorithms, e.g. Harmony Search, perform even worse than Monte Carlo search for some combinations of dimension and test problem.

Specifically, the differences within dimensions 5 – 20 and within 2 and 3 are much smaller than the differences between these two groups. This has already been pointed out by Mersmann et al. (2010a). Therefore, detailed performance analysis of a carefully selected set of best performing algorithms should be conducted separately for these two dimension classes. In Sec. 3.5 we perform such an analysis for the higher dimensional class as this is the most interesting one in our view.

3.3 Algorithm groups and representatives

The high number of more than 50 algorithms (including variants) present in the combined BBOB data set of 2009 and 2010 over-strains the capacities of tables and figures, as can easily be seen in Fig. 2. We therefore need to group the algorithms and continue only with the ones ranked best in each category. This requires a grouping criterion, for which we chose algorithmic similarity, so that all optimizers relying on the same base mechanism will be put into the same group. While this appears straightforward e.g. for the different variants of the CMA-ES, it is much less self-evident for the group of so called hybrid optimization methods. In this case, the use of multiple search paradigms itself is the underlying principle. Another problem is the different size of the obtained groups. For this reason, and also because there is a difference in the use of non-trivial restart heuristics, we have cut the largest group (CMA-ES) in two: the simple CMA-ES variants and the more complex ones (dubbed CMA-hybrid). Even so, the former still has 11 members. In stark contrast to that, the gradient and random search groups contain only one. This may seem a bit unfair, but it is a necessary step in the analysis to mitigate the possibility of inadvertently changing the ranking of two algorithms by just adding another variant of one of them. This risk was previously pointed out in the section on benchmarking. Moreover, we finally strive for a small number of best algorithms which is sufficient to, in some sense, cover all test problems. We expect that on very different problems, very different algorithms perform best, and not variants of the same algorithm. If this were the case, we would not call these

[3] Note, that because of the difficulty of visualizing the results at this stage on just one page, we have made larger individual plots available at `http://ptr.p-value.net/ecj13`

algorithms variants but assume that the effect we see stems from tuning. Finally, we think that answers to more general questions such as: "Shall I use an estimation of distribution algorithm (EDA) or a gradient method on problem x?" are of higher interest than the choice of the exact variant, which may also be seen as a different parametrization in many cases. By selecting a representative out of each class we would like to support this view. This position shall however not be misunderstood as discouragement of algorithm variant development. We would merely like to point out that a new variant should initially be benchmarked against the other available ones before competing against an algorithm from a different class.

In the following, we list the chosen groups and their algorithms, together with a short explanation concerning the conjunctive concept. Fig. 3 and Fig. 4 display consensus rankings within the algorithm groups over the required target precision and BBOB function groups as described in Sec. 3.2. Note that for selecting the representative, we rely on the Borda count consensus as the most intuitive approach. In case the rankings over precisions and function groups lead to different results, we allow for two representatives. Tab. 3 summarizes the Borda and SD/L consensus results for the algorithm groups for the lowest and highest precision value considered.

CMA-ES: (1,2_m) CMA-ES, (1,2_mˆs) CMA-ES, (1,2) CMA-ES, (1,2ˆs) CMA-ES, (1,4_m) CMA-ES, (1,4_mˆs) CMA-ES, (1,4) CMA-ES, (1,4ˆs) CMA-ES, (1+1)-CMA-ES, (1+2_mˆs) ES, $(\mu + \lambda)$ CMA-ES

This group contains all simple variants of the CMA-ES that do not employ a specific restart heuristic (other than randomly placed restarts). They can be seen as reparametrizations of the original CMA-ES (which is not in the test set). The representative of this group is the (1,2_mˆs) CMA-ES that strongly dominates the other CMA-ES versions on the unimodal function groups.

CMA-hybrid: BIPOP-CMA-ES, CMA-EGS, IPOP-ACTCMA-ES, IPOP-CMA-ES, IPOP-SEP-CMA-ES, NBC-CMA

In contrary to the group above, these CMA variants all employ non-trivial restart heuristics that change the population size or determine the search space positions for restarts. The group is represented by the IPOP-ACTCMA-ES.

DE: DE-F-AUC, DE-PSO, DEuniform, PM-AdapSS-DE

Here we find all methods that may largely be considered as differential evolution (DE) algorithms. It is represented by DE-F-AUC which clearly dominates over most precision values.

EDA: AMALGAM, BAYEDA, Cauchy-EDA, iAMALGAM

This group comprises the estimation of distribution (EDA) methods, and its representative is the iAMALGAM which ranks best over 4 of the 5 function groups.

GA/ES: ALPS, DASA, G3PCX, One-Fifth Variant3, RCGA, Simple GA, SPSA

Algorithms which largely follow the design principles of a genetic algorithm or an evolution strategy and do not use the covariance matrix adaptation are collected here. This group is represented by G3PCX as it ranks best over 3 of the 5 problem groups.

Global Search: AVGNEWUOA, DIRECT, FULLNEWUOA, GLOBAL, MCS, NEWUOA

These algorithms take explicit measures to aim for a good covering of the search space. Its representative is FULLNEWUOA because it is a good average performer.

Gradient: BFGS

This group contains the only 'pure' gradient method of all competing algorithms, BFGS, which also represents it.

Hybrid: EDA-PSO, MA-LS-CHAIN, MOS, nPOEMS, oPOEMS, POEMS, VNS

This group consists of methods that rely on multiple different search paradigms and can therefore be considered hybrid. We also put memetic algorithms here. The group is repre-

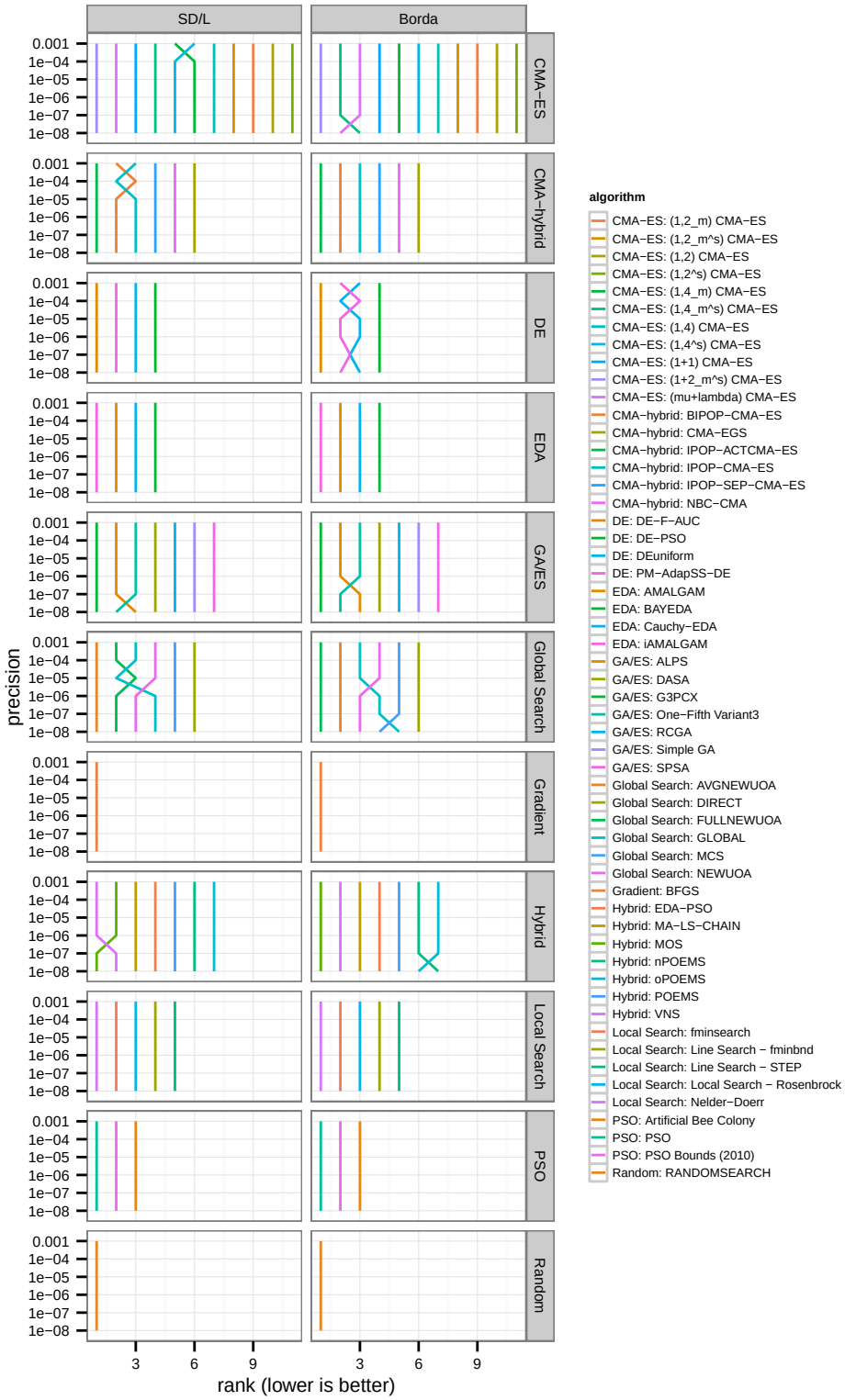

Figure 3: Consensus rankings within the chosen algorithm groups, according to target value precision

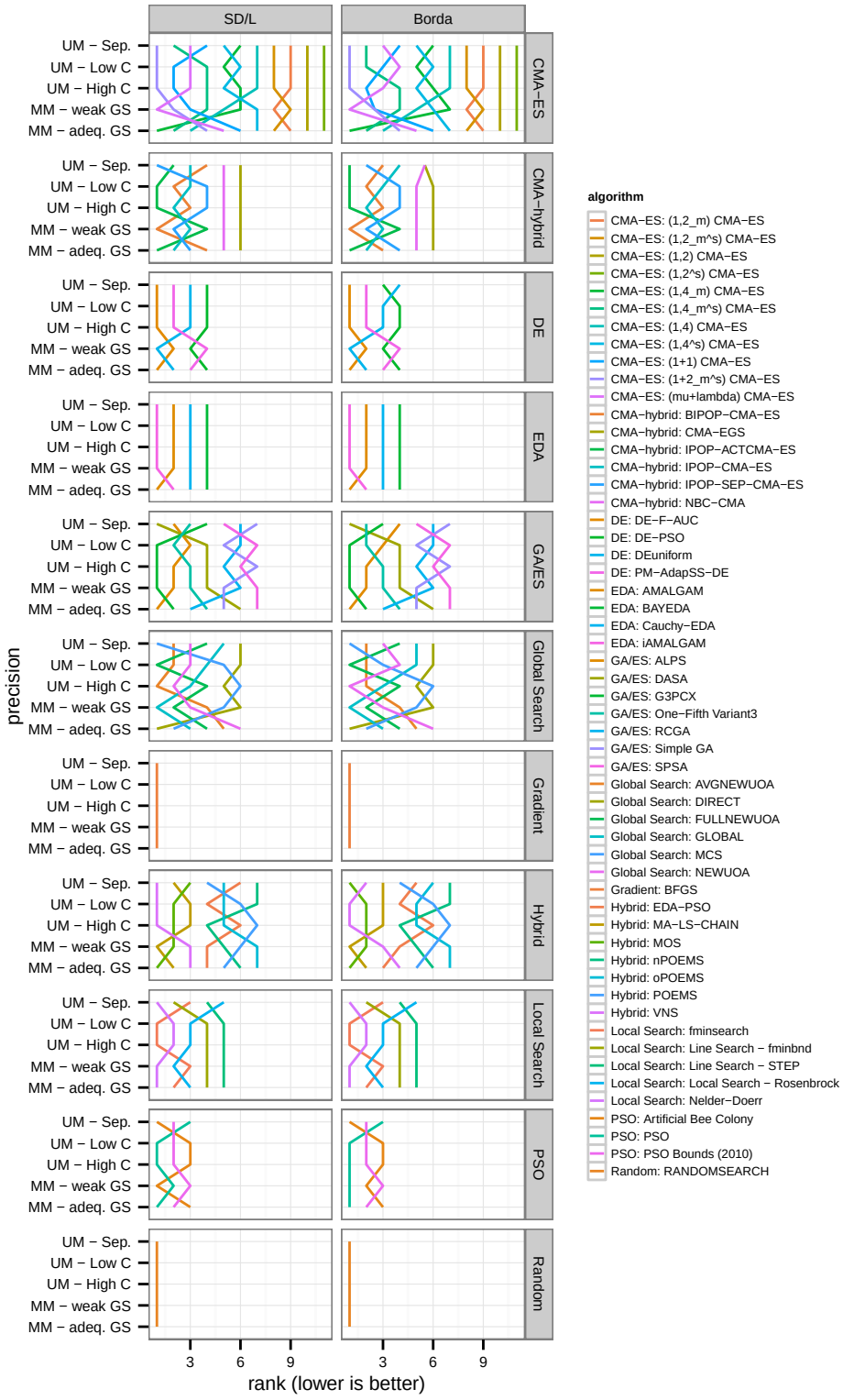

Figure 4: Consensus rankings within the chosen algorithm groups, according to function group

sented by the MOS algorithm which achieved the top rank for 2 of 5 cases.

Local Search: fminsearch, Line Search - fminbnd, Line Search - STEP, Local Search - Rosenbrock, Local Search: Nelder-Doerr

Here, the direct search and related methods are collected that explicitly aim for best possible local optimization and do not possess an explicit global search mechanism. However, Nelder-Doerr employs an evolutionary component by selecting the best of a population of local search instances. Nevertheless, it appears more similar to a (repeated) local search algorithm than to a hybrid or global optimization algorithm. It is also the representative of this group as it ranks best over 3 of the 5 problem groups.

PSO: Artificial Bee Colony, PSO, PSO Bounds (2010)

This group consists of nature-inspired swarm algorithms, namely different particle swarm optimization methods (PSO) and the artificial bee colony (ABC) method. It is represented by the PSO method which ranks best for most precisions and problem groups.

Random: RANDOMSEARCH

Only one random search algorithm has entered the BBOB instances, and its search paradigm is different enough from all others to justify its own group. It also represents the group.

Group	Consensus Method	# Alg.	Best Algorithm Precision 10^{-3}	Best Algorithm Precision 10^{-8}
CMA-ES	SD/L	11	(1+2_mˆs) CMA-ES	(1+2_mˆs) CMA-ES
CMA-ES	Borda	11	(1+2_mˆs) CMA-ES	(1+2_mˆs) CMA-ES
CMA-hybrid	SD/L	6	IPOP-ACTCMA-ES	IPOP-ACTCMA-ES
CMA-hybrid	Borda	6	IPOP-ACTCMA-ES	IPOP-ACTCMA-ES
DE	SD/L	4	DE-F-AUC	DE-F-AUC
DE	Borda	4	DE-F-AUC	DE-F-AUC
EDA	SD/L	4	iAMALGAM	iAMALGAM
EDA	Borda	4	iAMALGAM	iAMALGAM
GA/ES	SD/L	7	G3PCX	G3PCX
GA/ES	Borda	7	G3PCX	G3PCX
Global Search	SD/L	6	AVGNEWUOA	AVGNEWUOA
Global Search	Borda	6	FULLNEWUOA	FULLNEWUOA
Gradient	SD/L	1	BFGS	BFGS
Gradient	Borda	1	BFGS	BFGS
Hybrid	SD/L	7	MOS	VNS
Hybrid	Borda	7	MOS	MOS
Local Search	SD/L	5	Nelder-Doerr	Nelder-Doerr
Local Search	Borda	5	Nelder-Doerr	Nelder-Doerr
PSO	SD/L	3	PSO	PSO
PSO	Borda	3	PSO	PSO
Random	SD/L	1	RANDOMSEARCH	RANDOMSEARCH
Random	Borda	1	RANDOMSEARCH	RANDOMSEARCH

Table 3: Consensus ranking results for each algorithm group at the lowest / highest studied precision. Only the best algorithm is shown for each consensus method and as can be seen there are both groups in which the two consensus methods are in agreement and those where they do not agree.

Representative algorithms for groups

As motivated in the previous section, we need to choose a few representatives from each algorithm group. This was done by calculating the Borda consensus over all algorithms in each group for the accuracy levels 10^{-3} and 10^{-8} and choosing the best algorithm in each consensus as one of the representatives. All further analysis will also restrict itself to the two afore mentioned

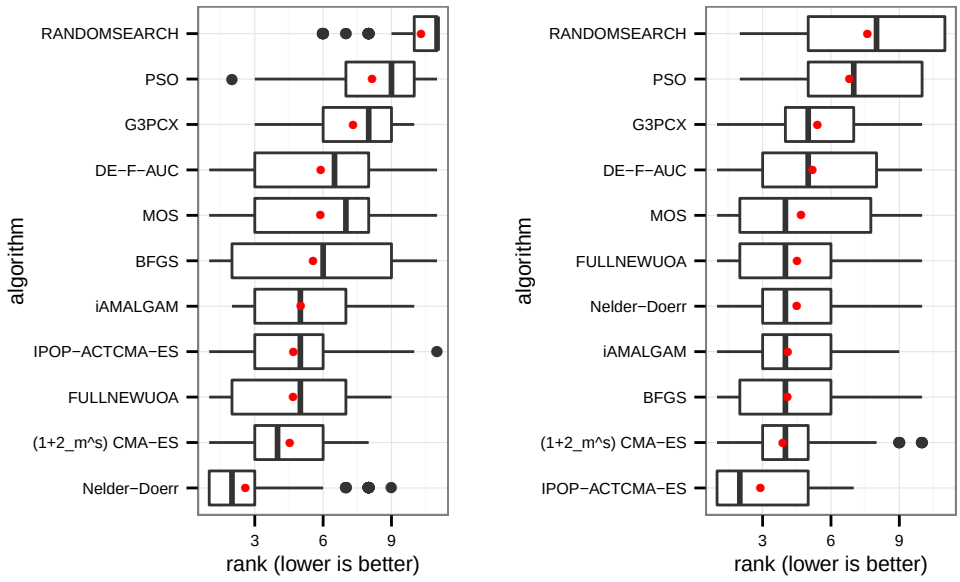

Figure 5: Box-plots showing the distribution of the rank of each of best algorithms for all 2D and 3D (left) and 5D to 20D (right) rankings. The red dot marks the mean of the ranks which coincides with the Borda score of the algorithm if a consensus was formed. The algorithms are therefore ordered by the Borda consensus. The vertical lines mark the median of the ranks.

precisions. Why do we use the Borda and not the SD/L consensus for this decision? The Borda consensus method captures our intent to find an algorithm that performs "above average" over all functions when compared to the other algorithms in the group. The SD/L method would prefer an algorithm that performs well on the majority of the test functions but may fail catastrophically on a minority. The chosen 11 algorithms for the further analysis are therefore (1+2_mˆs) CMA-ES, IPOP-ACTCMA-ES, DE-F-AUC, iAMALGAM, G3PCX, FULLNEWUOA, BFGS, MOS, Nelder-Doerr, PSO and RANDOMSEARCH. We will call this group of algorithms the *best algorithms* in the following sections. Do not confuse this with the mythical best algorithm mentioned in Sec. 2.

To get an initial idea of how these best algorithms compare to each other we can look at the distribution of their ranks. For each test function and dimension we obtain two rankings of the best algorithms. One for an accuracy level of 10^{-3} and one for an accuracy level of 10^{-8}. We can now extract the rank (i.e. the position) of each algorithm in each of these rankings and then look at this distribution. Two things we would want to see in a good algorithm are a low mean rank which means it is often one of the better algorithms and, as a secondary goal, a low variance of the ranks implying that its performance does not vary much with the test function. Since we cannot expect the algorithms to perform equally well in 2D and 3D when compared to higher dimensions we will look at these distributions separately. The results can be seen as boxplots in Fig. 5. We can instantly assess that there are differences between the 2D – 3D and the 5D – 20D distributions, as expected. In fact, in low dimensions it appears that a different set of algorithms performs well when compared to the high dimensional set.

Finally we can also look at an overall (Borda) consensus of the algorithms over all test

functions, dimensions and the two precision levels. This gives us

IPOP-ACTCMA-ES $\succ$ Nelder-Doerr $\succ$ (1+2_mˆs) CMA-ES $\succ$ iAMALGAM $\succ$ FULLNEWUOA $\succ$ BFGS $\succ$ MOS $\succ$ DE-F-AUC $\succ$ G3PCX $\succ$ PSO $\succ$ RANDOMSEARCH

More interesting are the Borda consensus rankings for the same scenario as above but not over all test functions. Instead for each function group as defined in Hansen et al. (2009b) a consensus is calculated over all dimensions and accuracy levels. This gives the following consensus rankings:

Unimodal - Separable:

Nelder-Doerr $\succ$ BFGS $\succ$ (1+2_mˆs) CMA-ES $\succ$ FULLNEWUOA $\succ$ IPOP-ACTCMA-ES $\succ$ iAMALGAM $\succ$ MOS $\succ$ DE-F-AUC $\succ$ PSO $\succ$ G3PCX $\succ$ RANDOMSEARCH

Unimodal - Low Contrast:

FULLNEWUOA $\succ$ IPOP-ACTCMA-ES $\succ$ BFGS $\succ$ (1+2_mˆs) CMA-ES $\succ$ Nelder-Doerr $\succ$ iAMALGAM $\succ$ G3PCX $\succ$ DE-F-AUC $\succ$ MOS $\succ$ PSO $\succ$ RANDOMSEARCH

Unimodal - High Contrast:

iAMALGAM $\succ$ IPOP-ACTCMA-ES $\succ$ (1+2_mˆs) CMA-ES $\succ$ Nelder-Doerr $\succ$ BFGS $\succ$ DE-F-AUC $\succ$ FULLNEWUOA $\succ$ MOS $\succ$ G3PCX $\succ$ PSO $\succ$ RANDOMSEARCH

Multimodal - weak Global Structure:

Nelder-Doerr $\succ$ FULLNEWUOA $\succ$ BFGS $\succ$ (1+2_mˆs) CMA-ES $\succ$ G3PCX $\succ$ MOS $\succ$ iAMALGAM $\succ$ IPOP-ACTCMA-ES $\succ$ DE-F-AUC $\succ$ PSO $\succ$ RANDOMSEARCH

Multimodal - adequate Global Structure:

IPOP-ACTCMA-ES $\succ$ MOS $\succ$ DE-F-AUC $\succ$ iAMALGAM $\succ$ Nelder-Doerr $\succ$ (1+2_mˆs) CMA-ES $\succ$ PSO $\succ$ FULLNEWUOA $\succ$ G3PCX $\succ$ BFGS $\succ$ RANDOMSEARCH

Here we can see that while Nelder-Doerr is only second in the overall ranking, it dominates two function groups, whether the IPOP-ACTCMA-ES leads only one. Additionally, FULLNEWUOA and iAMALGAM, who do not play an important role in the overall ranking, each dominate one function group. It is obvious that knowing the problem type is very important for selecting the right algorithm.

After looking at these different ways to aggregate the results we might ask if there is a natural grouping of functions that arises from the performance of the algorithms. To answer this question and gain further insight into the performance characteristics of the set of best algorithms we will need some tools from statistics which will be introduced next.

3.4 Statistical Methods

In the following section we apply different methods from data analysis and statistics to visualize and interpret the BBOB results. These statistical tools will be briefly introduced in this section. Much more detailed explanations can be found in the cited literature, and we recommend Hastie et al. (2001) as a general reference, which covers all the required material.

Multidimensional scaling (MDS) is a visualization technique, embedding objects x_i for $i \in \{1, \ldots, n\}$, from a high-dimensional into a lower dimensional euclidean space (usually 2D or 3D). MDS operates only on a matrix of given distances (e.g. euclidean) or dissimilarities $\delta_{i,j}$ between the objects. Note, that MDS can therefore be applied even when only the $\delta_{i,j}$ are

known, but not the x_i themselves. The embedding is performed in such a way that the distances are maintained as closely as possible by solving the following optimization problem:

$$\min_{z_1,\ldots,z_n} \sum_{i \neq j} \left(||z_i - z_j|| - \delta_{i,j} \right)^2 \ .$$

Here, the $z_i \in \mathbb{R}^k$ are the low-dimensional mappings of the original x_i. The optimization problem is usually solved by a gradient descent algorithm.

Partitioning around medoids (PAM) is an unsupervised data mining method, which clusters unlabeled objects x_i into sets of neighbored items. Again, we assume to have a matrix of distances $\delta_{i,j}$ between all objects x_i and x_j available. At each stage of the algorithm a set of k representatives (the medoids) is maintained. As this is just a subset of the original data points, their set of indices $\{i_1, \ldots, i_k\}$ suffices. The target function to minimize is defined as

$$\sum_{j=1}^{k} \sum_{C(i)=j} \delta_{i,i_j} \ ,$$

where $C(i) = \arg\min_j \delta_{i,i_j}$ is the index of the nearest representative to observation x_i.

In the initial phase the medoids are chosen in a greedy, iterative way to reduce the target function value. Afterwards, the algorithm exchanges in each step until convergence a medoid with a non-medoid so that the target function is maximally reduced.

The number of clusters can be decided by running PAM several times with different values for k and calculating the so called average silhouette width. This width defines a numerical measure for each observation and reflects how well it fits into its selected cluster $C(i)$ instead of any other cluster. For a formal definition see Kaufman and Rousseeuw (1990). The parameter k is then chosen by selecting the number of clusters with the highest silhouette width.

Classification and Regression Trees (CART) Breiman et al. (1984) try to find a mapping between a d-dimensional input space $X = X_1 \times \ldots \times X_d$ (where the individual features are usually measured on a metric, ordinal or nominal scale) and a set of finite labels $Y = \{y_1, \ldots, y_g\}$. The mapping is of the form of a binary tree, where every node represents a univariate rule $x_i < c$ (or $x_i = c$ if X_i is nominal). These rules are generated in a greedy, top-down fashion by analyzing a finite set of learning examples $(x^i, y^i) \in X \times Y$. The best rule for the current node k is found by first considering the so-called impurity

$$i(k) = 1 - \sum_{j=1}^{g} p_k(j)^2$$

of the node. This is maximal, if all classes in the data which reach node k occur with equal relative frequency $p_k(j)$. Note, that our definition above is also called the Gini index and alternative measures of node impurity are available. The rule for node k is selected now in such a way that the difference in impurity between k and its subnodes (created by this new rule) is maximal:

$$\delta(k) = i(k) - p_l i(l) - p_r i(r)$$

Here $i(l)$ and $i(r)$ are the impurities of the left and right subnode of k, and p_l and p_r are the percentages of data which move from the parent node r to its children l and r respectively.

Usually, a large tree is grown w.r.t. some stopping criterion, such as the minimal node size, and then simplified in a second stage by cost-complexity pruning.

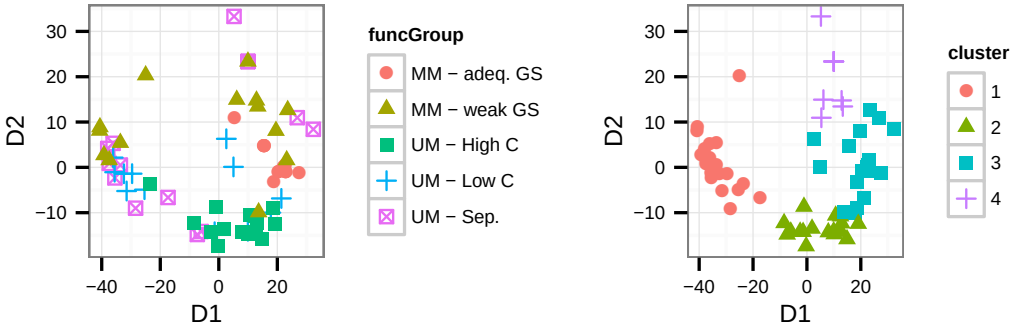

Figure 6: Plot of the two dimensional projection recovered from the dissimilarity matrix of the rankings of the best algorithms in 5 to 20 dimensions. Each point represents a single test function / dimension combination. On the left the points are colored according to the function group the test function belongs to and on the right the color shows which cluster was assigned to the test function by the PAM procedure.

3.5 Characterizing performance of the "best group"

In Sec. 3.3 we chose a subset of algorithms from the original dataset called the best algorithms and used consensus methods to find a best algorithm for some dimension or some subset of the test functions. In this section we will focus on understanding the structure the algorithms reveal in our set of test functions. Consequently, we will restrict ourselves to what we consider the current sweet spot of black-box optimization, that is to 5-20 dimensional problems.

For these we would like to know if the similarity between test functions that is implied in the five function groups defined by the BBOB team is also present in the rankings. Therefore, we need means to characterize similarity or distance between rankings. Recall that we introduced such a distance metric for the optimization based consensus method in Sec. 2. Using the SD metric proposed there we can calculate a similarity matrix between all the individual rankings. We can visualize this matrix using multi-dimensional scaling and also try to cluster the rankings based on their relative distance to each other using PAM. Using the average silhouette width as the cluster index, the optimal number of clusters into which to partition the rankings is 4. This is already a departure from the 5 function groups proposed. To see how well the two groupings of the test functions agree we plot the 2D representation of the dissimilarity matrix as recovered by the MDS in Fig. 6, and color the points according to their group memberships.

Even though the MDS is only an approximate representation of the distance matrix, we can see that the clusters are also visible in the plots. On the other hand, the function groups do not seem to be reflected in the clustering or the MDS plot. We therefore infer that the function grouping which was provided by the BBOB team does not coincide with similar algorithm behavior. Instead we have empirically determined the four new groups shown in Table 4.

What is missing is a set of rules describing these clusters. From the list above it is almost impossible to deduce anything about what defines a cluster. Some functions are spread between different clusters, all clusters contain some 5, some 10 and some 20 dimensional functions. So instead of trying to describe the clusters based on the function name and dimension, we will try to abstract away from the concrete test function and replace each test function by a set of features that describe the function. These were first proposed by Mersmann et al. (2010a) and are shown in Tab. 5. They are again a subjective way of categorizing each function. We then train a classification tree on the obtained data set by using the function properties as the features and the cluster as our target class. The resulting tree is displayed in Fig. 7 and permits two

Function	5D	10D	20D	Function	5D	10D	20D
f_{01}	1	1	1	f_{13}	2	3	2
f_{02}	1	2	2	f_{14}	2	2	2
f_{03}	3	3	4	f_{15}	3	3	3
f_{04}	4	4	4	f_{16}	3	3	3
f_{05}	1	1	1	f_{17}	3	3	3
f_{06}	2	3	3	f_{18}	3	3	3
f_{07}	2	2	2	f_{19}	3	3	4
f_{08}	1	1	1	f_{20}	3	3	4
f_{09}	1	1	1	f_{21}	1	1	1
f_{10}	2	2	2	f_{22}	1	1	1
f_{11}	2	2	2	f_{23}	3	4	4
f_{12}	1	2	2	f_{24}	3	4	–

Table 4: Table showing the cluster that was assigned to each function / dimension combination. Note that for f_{24} in 20 dimensions we do not have enough data to assign a cluster.

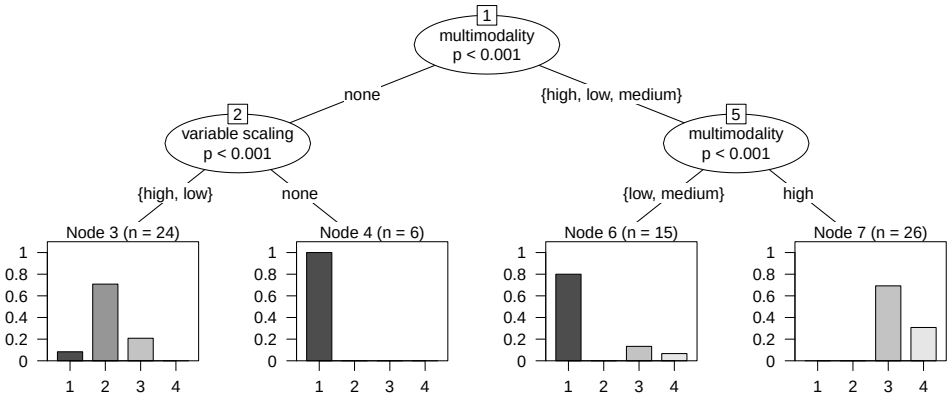

Figure 7: Decision tree describing the relationship between the features defined in Tab. 5 and the clusters found by PAM. Each node of the tree describes a decision. The variable on which the decision is based is given in the node and the values are the labels on each edge. The leaves of the tree describe the relative frequence of each cluster in the data that, according to the decision rules, belong in that leaf. High values mean that it is likely that a function with these properties would belong to the respective cluster. For example, all test functions which are in leaf node 4, i.e. that have no multimodality, and no variable scaling, belong to cluster 1.

interesting observations.

First of all, we have no way of differentiating between cluster 3 and cluster 4 using the defined features. This is surprising and might lead to the discovery of new characteristics of the test functions which might explain the difference between the two clusters. The main difference between the remaining cluster one and cluster two is that cluster one has no variable scaling and cluster two is unimodal.

We conclude by presenting a Borda consensus over the four groups discovered using the cluster analysis. Recall that these four groups are already fairly homogeneous w.r.t. the algorithm rankings since their distance to each other, in the SD metric, is small. The resulting consensus rankings are:

Cluster 1:

FULLNEWUOA $\succ$ BFGS $\succ$ Nelder-Doerr $\succ$ (1+2_mˆs) CMA-ES $\succ$ IPOP-ACTCMA-ES $\succ$ G3PCX $\succ$ iAMALGAM $\succ$ MOS $\succ$ DE-F-AUC $\succ$ PSO

Function	multim.	gl.-struc.	separ.	scaling	homog.	basins	gl.-loc.	plat.
1: Sphere	none	none	high	none	high	none	none	none
2: Ellipsoidal separable	none	none	high	high	high	none	none	none
3: Rastrigin separable	high	strong	high	low	high	low	low	none
4: Büche-Rastrigin	high	strong	high	low	high	med.	low	none
5: Linear Slope	none	none	high	none	high	none	none	none
6: Attractive Sector	none	none	none	low	med.	none	none	none
7: Step Ellipsoidal	none	none	none	low	high	none	none	small
8: Rosenbrock	low	none	none	none	med.	low	low	none
9: Rosenbrock rotated	low	none	none	none	med.	low	low	none
10: Ellipsoidal high conditioned	none	none	none	high	high	none	none	none
11: Discus	none	none	none	high	high	none	none	none
12: Bent Cigar	none	none	none	high	high	none	none	none
13: Sharp Ridge	none	none	none	low	med.	none	none	none
14: Different Powers	none	none	none	low	med.	none	none	none
15: Rastrigin multimodal	high	strong	none	low	high	low	low	none
16: Weierstrass	high	med.	none	med.	high	med.	low	none
17: Schaffer F7	high	med.	none	low	med.	med.	high	none
18: Schaffer F7 moderately ill-cond.	high	med.	none	high	med.	med.	high	none
19: Griewank-Rosenbrock	high	strong	none	none	high	low	low	none
20: Schwefel	med.	deceptive	none	none	high	low	low	none
21: Gallagher 101 Peaks	med.	none	none	med.	high	med.	low	none
22: Gallagher 21 Peaks	low	none	none	med.	high	med.	med.	none
23: Katsuura	high	none	none	none	high	low	low	none
24: Lunacek bi-Rastrigin	high	weak	none	low	high	low	low	none

Table 5: Classification of the noiseless functions based on their properties (multi-modality, global structure, separability, variable scaling, homogeneity, basin-sizes, global-local contrast, plateaus). Predefined groups are separated by horizontal lines. Note that we did not assign global structure to the unimodal problems. This is disputable and in contrast to the opinion of the BBOB organizers. However, is has not been formally defined in the BBOB setup

Cluster 2:

IPOP-ACTCMA-ES $\succ$ (1+2_mˆs) CMA-ES $\succ$ iAMALGAM $\succ$ MOS $\succ$ DE-F-AUC $\succ$ Nelder-Doerr $\succ$ G3PCX $\succ$ FULLNEWUOA $\succ$ BFGS $\succ$ PSO

Cluster 3:

IPOP-ACTCMA-ES $\succ$ MOS $\succ$ DE-F-AUC $\succ$ iAMALGAM $\succ$ FULLNEWUOA $\succ$ PSO $\succ$ G3PCX $\succ$ (1+2_mˆs) CMA-ES $\succ$ Nelder-Doerr $\succ$ BFGS

Cluster 4:

MOS $\succ$ iAMALGAM $\succ$ IPOP-ACTCMA-ES $\succ$ Nelder-Doerr $\succ$ PSO $\succ$ (1+2_mˆs) CMA-ES $\succ$ BFGS $\succ$ DE-F-AUC $\succ$ FULLNEWUOA $\succ$ G3PCX

The consensus rankings allow for making several interesting observations. We discover that the difference between clusters 1 and 2 is that classical optimization approaches work well on functions in the first cluster while evolutionary strategies outperform them on the second cluster of functions. These are the two clusters we can characterize fairly well, as can be seen in Fig. 7. The last two clusters consist of a large amount of functions which are some of the hardest in the test function set because on these RANDOMSEARCH outperforms some more advanced methods. Nearly all functions in 3 and 4 are multimodal, most of them highly multimodal. Whether the functions in 3 can usually be solved by the IPOP-ACTCMA-ES, this is often not the case for cluster 4, so that many of these are not solved by any method.

Comparing the results obtained on the newly found four clusters and the clusters defined by the BBOB team based on human experience, we come to very different conclusions. However, we have to admit that we do not yet have adequate features to describe our groups.

4 Outlook on Future Work

Analyzing the benchmark data as done in Sec. 3 already generates many valuable insights into the performance ranking of optimizers and algorithmic groups under various problem conditions. However, it does not provide one with a satisfying answer to the urgent problem of what optimization algorithm should be selected in practice for a given, unknown problem or problem domain. It is a well known fact from practice – and also clearly visible in our presented results – that no current optimization algorithm solves all problems equally well. Although the No-Free-Lunch theorem does not hold for the case of continuous spaces as shown by Auger and Teytaud (2007), it is very unlikely that one optimization algorithm will completely dominate all others in the near future. Therefore, a general set of rules which guide a practitioner in choosing an appropriate algorithm for the problem at hand from the vast pool of available optimizers would be a highly useful tool. At the same time, the knowledge used to construct such a ruleset might be used to construct even better or more robust optimizers.

Of course, the construction of such a set of rules requires the definition of test problem characteristics and relating them in a meaningful way to the expected performance of an optimizer. This will in general be a very challenging problem. A very similar task has been considered in the machine learning community under the term of meta learning (see for example Brazdil et al. (1994)), where one tries to predict which learner is most appropriate given a feature vector of data set characteristics. In Mersmann et al. (2010a) we already proposed a set of manually constructed test set properties. These have two major disadvantages: They are discrete (e.g. low, medium or high multi-modality) and therefore somewhat ambiguous. And they obviously require their definition by an expert, which limits their practical usefulness, if one wants to move beyond solving and analyzing the BBOB problem set.

We are currently working on the definition of an extensive set of numerical, computable test problem characteristics, which contain, among other numerical properties, "average" gradients and convexity, landmarking by simple and fast optimizers and techniques from regression and classification to capture general landscape shapes. For first results see Mersmann et al. (2011).

In our future work we will try to demonstrate that:

1. These features can be constructed for any test problem without help from a human expert.
2. For the BBOB set these at least contain all the information gained by the manually crafted features.
3. At least in an exploratory sense they can be used to relate test problem characteristics to optimizer performance or ranking.

The biggest remaining challenge then will be to efficiently calculate a relevant subset of these features in an on-line fashion and use them to select or switch to an appropriate optimization algorithm set for a considered, unknown target problem.

5 Conclusions

In this article, we have shown how a combination of benchmarking methods and classical statistical exploratory data analysis can be used to gain insight into the performance characteristics of a set of algorithms under test. For this we introduced a novel approach to aggregate the results of black-box optimization benchmarks. The approach requires a carefully chosen set of test functions and performance measures. We apply this approach to the combined 2009 and 2010 BBOB results. After reducing the number of algorithms by partitioning the set of algorithms into groups of similar algorithm designs and reducing these partitions to one or two representative algorithms, we are able to show that the relative performance of these algorithms is far from uniform over all test functions. Even within the predefined groups of functions the algorithm performance varies widely. Using the similarity between the individual rankings we used cluster

methods to find four groups within which the relative performance of the algorithms is homogeneous. We then set out to explain the cluster memberships by using properties of the functions as features in order to build a decision tree which describes the relationship between clusters and function properties. We will continue this line of work by developing tools to automatically characterize functions using empirical features. Based upon these, rules can be constructed which allow practitioners the selection of a reasonable set of algorithms as a starting point for a new optimization problem. We call this new line of research *Exploratory Landscape Analysis* (ELA).

Acknowledgments

This work was partly supported by the Collaborative Research Center SFB 823, the Graduate School of Energy Efficient Production and Logistics and the Research Training Group "Statistical Modelling" of the German Research Foundation.

Supplementary Material

The complete source code used to produce the figures, tables and consensus rankings in this paper, and all figures, in color, as well some additional figures which might be useful to better understand how some of the conclusions were derived, especially for Fig. 2, are available at `http://ptr.p-value.net/ecj13`.

References

Arrow, K. J. (1950). A difficulty in the concept of social welfare. *Journal of Political Economy*, 58:328.

Auger, A., Finck, S., Hansen, N., and Ros, R. (2010). BBOB 2010: Comparison Tables of All Algorithms on All Noiseless Functions. Technical Report RT-388, INRIA.

Auger, A. and Teytaud, O. (2007). Continuous lunches are free! In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 916–922, New York, NY, USA. ACM.

Brazdil, P., Gama, Jo a., and Henery, B. (1994). Characterizing the applicability of classification algorithms using meta-level learning. In *ECML-94: Proceedings of the European conference on machine learning on Machine Learning*, pages 83–102, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Chapman & Hall/CRC.

Cook, W. D. and Kress, M. (1992). *Ordianl Information & Preference Structures*. Prentice Hall, Upper Saddle River, NJ.

de Borda, J. C. (1781). Mémoire sur les élections au scrutin. *Historie de l'Académie Royale des Sciences*.

Hansen, N., Auger, A., and Auger, A. (2005). Performance evaluation of an advanced local search evolutionary algorithm. In *In Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1777–1784. IEEE Press.

Hansen, N., Auger, A., Finck, S., and Ros, R. (2009a). Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA.

Hansen, N., Auger, A., Ros, R., Finck, S., and Pošík, P. (2010). Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *GECCO '10: Proceedings of the 12th annual conference comp on Genetic and evolutionary computation*, pages 1689–1696, New York, NY, USA. ACM.

Hansen, N., Finck, S., Ros, R., and Auger, A. (2009b). Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer, New York.

Hornik, K. and Meyer, D. (2007). Deriving consensus rankings from benchmarking experiments. In Decker, R. and Lenz, H.-J., editors, *Advances in Data Analysis (Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation e.V., Freie Universität Berlin, March 8–10, 2006*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 163–170. Springer.

Hunter, D. J. (2008). *Essentials of Discrete Mathematics*. Jones and Bartlett Publishers, Boston, MA.

Kaufman, L. and Rousseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Interscience, New York.

Kemeny, J. G. and Snell, J. L. (1972). *Mathematical Models in the Social Sciences*. MIT Press, Cambridge, MA.

Mersmann, O. (2009). Benchmarking evolutionary multiobjective optimization algorithms using R. Bachelor Thesis, http://www.statistik.tu-dortmund.de/∼olafm/ files/ba.pdf, TU Dortmund.

Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., and Rudolph, G. (2011). Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 829–836, New York, NY, USA. ACM.

Mersmann, O., Preuss, M., and Trautmann, H. (2010a). Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In Schaefer, R. et al., editors, *PPSN XI: Proceedings of the 11th International Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science 6238, pages 71–80. Springer.

Mersmann, O., Trautmann, H., Naujoks, B., and Weihs, C. (2010b). Benchmarking evolutionary multiobjective optimization algorithms. In Ishibuchi, H. et al., editors, *Congress on Evolutionary Computation (CEC)*. IEEE Press, Piscataway NJ.

Mersmann, O., Trautmann, H., Naujoks, B., and Weihs, C. (2010c). On the Distribution of EMOA Hypervolumes. In Blum, C. and Battiti, R., editors, *LION*, volume 6073 of *Lecture Notes in Computer Science*, pages 333–337. Springer.

Mood, A., Graybill, F., and Boes, D. (1974). *Introduction to the Theory of Statistics*. McGraw-Hill, New York.

Saari, D. G. and Merlin, V. R. (1997). A geometric examination of Kemeny's Rule. *Social Choice and Welfare*, 17:2000.

Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms - a comparative case study. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 292–304, London, UK. Lecture Notes in Computer Science, Springer-Verlag.