Benchmarking Evolutionary Algorithms: Towards Exploratorybenchmarking optimization algori ÔbestÕone? and the second one: which algorithm should problem? Both are connected and neither is easy to answ which can be used to analyse the raw data of a benchmar some insight regarding the answers to these questions. W methods to analyse the BBOBÕ09 benchmark results and Indings.

Keywords: evolutionary optimization, benchmarking, BB mensional scaling, consensus ranking.

1 Introduction

The last years have seen several competitions for optimization algorithms at *evolutionary computation* (EC) conferences, possibly starting with the CEC'05 [11], and currently most notably continued with the black-box optimization algorithm benchmarking (BBOB) competitions at GECCO 2009 and 2010 [2]. However, this bears two main questions: a) Given a number of comparison results on different functions, what is the 'best' algorithm? and b) How can we transfer benchmarking results onto real-world situations?

To answer the first question, we turn to existing benchmarking theory, and especially to consensus ranking procedures. Answering the second question is surely harder. While it is relatively easy to control the settings of a benchmark experiment and to enforce every problem property we can possibly imagine, not much is usually known about a real-world problem we may have to deal with. The only possible solution for achieving a good algorithm-problem matching may be to extract meaningful high-level empirical properties and approach the matching from two sides: a) find out which algorithms perform especially well on certain property combinations and b), develop ways to cheaply and automatically extract problem properties from a concrete problem instance. The latter may be termed *Exploratory Landscape Analysis* (ELA) and is our long-term objective which is not explicitly attempted in this work. However, in order to achieve any progress in this directions, we need a good set of properties and to resolve the first issue of detecting and evaluating algorithm-property dependencies.

R. Schaefer et al. (Eds.): PPSN XI, Part I, LNCS 6238, pp. 73–82, 2010.

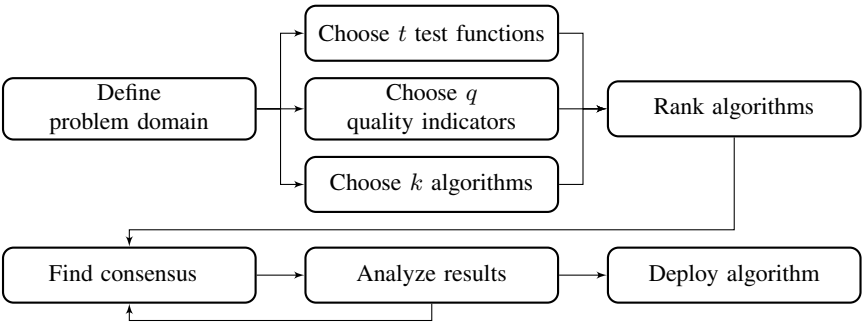

Fig. 1. Flow chart describing the steps involved in a benchmark experiment

After introducing benchmarking theory in Sec. 2 we will focus on the BBOB'09 test in Sec. 3 which is grouped into 5 categories using predefined properties. We will analyse if the measured performance of the benchmarked algorithms is in line with this grouping and will include additional properties into the analysis. Furthermore, we will explicitly make the distinction of low (2 and 3) and high (5-20) dimensions[1] and analyse how similar or different the algorithms behave with respect to these groups. Conclusions are given in Sec. 4.

2 Benchmarking Theory

The outcome of benchmarking experiments and competitions strongly depends on chosen performance measures and ranking procedures ([7],[8]). The general setup of a benchmark experiment is shown in Fig. 1.

Initially a *problem domain* needs to be defined. This step is crucial since it restricts the domain to which any conclusions made in later steps can possibly generalize. Next, t test functions with a known global optimum value or a known bound that should be achieved are chosen. Finding good test functions is a hard problem since they should be distributed in a 'uniform' fashion in the space of all possible functions from the problem domain. To judge the performance of an optimization result, q (ideally independent) quality indicators are chosen. Finally, a set of candidate optimization procedures needs to be determined.

The next step is to determine the number of independent runs of each of the k algorithms on the t test functions with respect to the trade-off between speed and accuracy. In practice 10 to 25 repetitions are a good rule of thumb. The result of this are $t \times q \times k$ quality indicator estimates. Using these, individual rankings of the algorithm for each quality indicator and test function will be constructed.

2.1 Individual Ranking

Benchmarking theory is mainly based on the theoretical framework of relations and orders [4] from which a formal definition of a *ranking* of a set of items can be derived

[1] The BBOB'09 data for 40 dimensions is not complete and therefore discarded.

[8]. We will use $\succ$, where $a \succ b$ reads as "a better than or equal to b", to denote this ranking and the underlying relation.

Initially, without loss of generality, we will consider the case of a fixed test function f and quality criterion I to be maximized. 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 straight forward. We use the order induced by I on the algorithms as our ranking. The main disadvantage of our simple definition of $\succ$ is that we must *estimate* the value of a quality criterion from several runs of the optimization algorithm. This means that $I(a, f)$ is a random variable whose distribution is unknown. One way to deal with this is to use classical statistical hypothesis tests to define $\succ$. Details of this procedure are given in [7] and [8]. The main obstacle is that we need $\succ$ to be transitive and antisymmetric in order for it to define a meaningful ordering.

2.2 Consensus Ranking

In case the *best* algorithm out of a given set should be determined the problem of building a consensus of a number of individual rankings of the considered algorithm arises. Ideally, a consensus should be non-dictatorial, universal, Pareto efficient and fulfill the *Independence of irrelevant alternatives* (IIA) criterion as well as the majority criterion (see [7] or [8] for formal definitions). Unfortunately, all criteria cannot be met simultaneously. Thus, consensus approaches yield different results with respect to the criteria chosen to be fulfilled.

Generally, we can differentiate between positional and optimization based methods. It can be shown that the Borda count method [1], as one of the oldest consensus methods, is optimal under all positional consensus methods [9]. An algorithm is assigned one point for each algorithm that it weakly dominates, i.e. the algorithms that are not better than the algorithm considered. The Borda score results by taking the sum of these values. Optimization based methods transform the consensus ranking task into an optimization problem based on a suitable distance measure between the individual rankings. The consensus ranking comes out as the ranking which minimizes the median or mean distance to all individual rankings (*SD approach*, [6]). Advantages and drawbacks of the introduced consensus methods are summarized in [10].

3 Benchmarking Results for the BBOB09-Testset

3.1 Suggested Problem Properties

The BBOB'09 test set is built according to problem properties, a) separable problems, b) low or moderate conditioned problems, c) high conditioned and unimodal problems, d) multi-modal problems with adequate global structure, and e) multi-modal problems with weak global structure. We suggest to add more properties applied to the BBOB'09 test set in Table 1. It gets evident that some property values are sparsely populated, e.g. only one test function has plateaus.

Multi-modality refers to the number of local optima of a problem. In practical applications, many problems are not unimodal (convex) as favoured by most classical optimization algorithms.

Global structure is what remains after deleting all non-optimal points. For Rastrigins problem, we obtain a perforated parabola which is unimodal. Problems without global basin structure are more difficult because one virtually needs to look in every corner.

Separability means a problem may be partitioned into subproblems which are then of lower dimensionality and should be considerably easier to solve. However, for an unknown problem, information about its separability may be scarce.

Variable scaling can make a problem behave differently in each dimension. It can be essential to perform small steps in some dimensions, and large ones in others, which is due to the non-spherical form of basins of attraction. Note that scaling may differ between different basins of attraction.

Search space homogeneity refers to a search space without phase transitions. Its overall appearance is similar in different search space areas. Most benchmark problems are of this type.

Basin size homogeneity means the size relation (largest to smallest) of all basins of attraction (e.g. [12] postulated that size differences influence problem hardness).

Global to local optima contrast refers to the difference between global and local peaks in comparison to the average fitness level of a problem. It thus determines if very good peaks are easily recognized as such.

Plateaus can make the life of optimization algorithms a lot harder as they do not provide any information about good directions to turn to. However, in the BBOB'09 test set, this property is largely unused.

3.2 Algorithm Analysis

Initially, a ranking for each test function / dimension combination is calculated using the *expected running time* (ERT, [2]) which is shown in Fig. 2 and 3. It is evident that there is a change in the general performance of the algorithms going from three to five dimensions. Especially in the parallel coordinate plot we recognize a deterioration of performance for some algorithms as the dimension rises.

Looking at the distribution of these ranks separately for the two and three as well as the 5 to 20 dimensional functions as shown in Fig. 4, we see two very different consensus rankings of the algorithms. The order of the algorithms was chosen by decreasing mean rank, this coincides with the Borda consensus for the two dimension groups. In lower dimensions the Nelder-Mead type algorithms perform best while only ranking in the middle for higher dimensional problems. The same behaviour can be observed for some other classical algorithms such as the Rosenbrock procedure. On the other hand, BFGS performs better for higher dimensional problems. Here, the effort invested for estimating the gradient obviously pay off. Generally, the order of the algorithms gets more stable for higher dimensions, so that the differences between dimensions 5/10/20 are much smaller than the ones between 2/3/5 (see Fig. 2). We therefore do not present a consensus ranking over all dimensions.

3.3 Function-Set Analysis

We have seen that the algorithms perform quite differently in different dimensions. Can we expect similar behaviour for different classes of functions from the test function set?

Table 1. 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.

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	none	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	high	low	med.	none	none	none
7: Step Ellipsoidal	none	none	high	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

And more importantly, can we use any insight gained from the benchmark experiment to choose a good algorithm for a real world optimization problem? For this, we will use the distance measure SD introduced for the SD/L consensus method to calculate the distances between the 144 rankings as a way to quantify how similar the algorithms performed. We focus our analysis on the 5 to 20 dimensional functions, as they are more challenging than the low dimensional ones. Additionally, algorithm ranks are much more consistent on these functions, which should simplify the analysis.

Ideally, we would like to identify groups from this reduced function set that lead to similar algorithm rankings. We then could reduce the size of the function set by only including one prototype from each group. In addition, new functions similar to functions in this group would probably result in similar algorithm performance. We could therefore try to guess a good algorithm from the function group the new problem belongs to.

Two approaches are used to retrieve groups or clusters from the distance matrix. First, we project the high-dimensional data onto a lower dimensional space by means of multidimensional scaling (MDS, [3]) in order to visualize the relationship between observations. The MDS embeds the observations into the lower dimensional space while attempting to retain the distance between data points. Thus, the starting point for any MDS algorithm is a distance matrix[2] over all observations of a dataset. Starting from

[2] Usually this is a similarity matrix, but it is trivial to compute one out of the other.

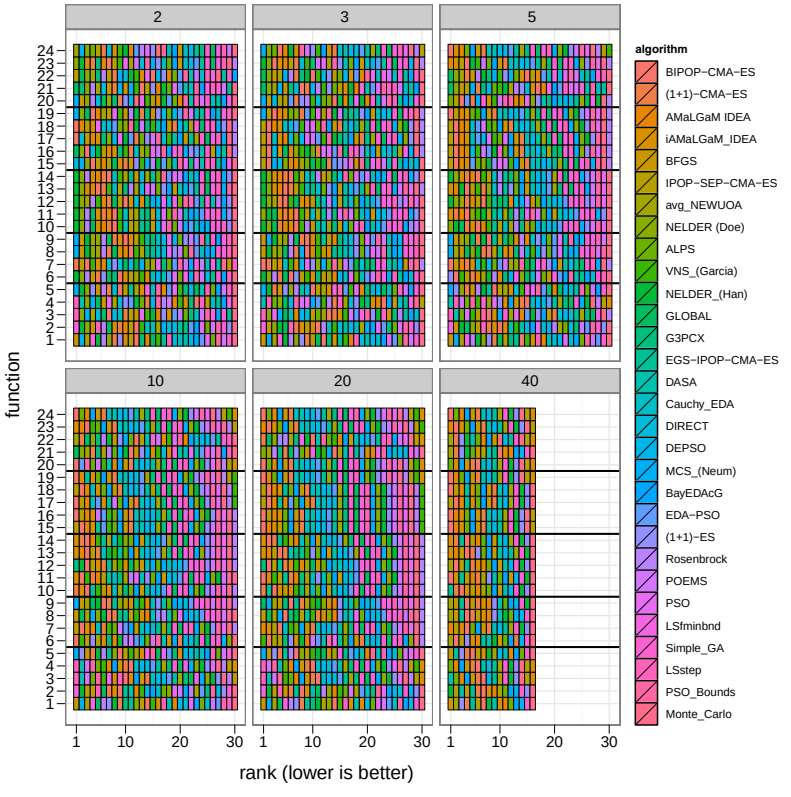

Fig. 2. Benchplot for each dimension by function. The color indicates the algorithm and the position of its rank in the ranking. For higher dimensions, a clear structure emerges because many algorithms are not able to reliably solve such problems with the given number of FEs. The black lines separate the 5 function groups as defined by the BBOB'09 organizers.

this matrix, a geometrical representation of the relationship is created while preserving the input distances or distance as accurately as possible. This is formalized as the optimization problem of minimizing a loss function over the difference between the input and output distance matrix. Next, the clustering algorithm PAM [5] is employed to find clusters in the data, based on the distance matrix. For this dataset 2 clusters were determined to be optimal, yielding an average silhouette width of ≈ 0.44. These clusters were then correlated with the identified function properties (see Table 1). Note that this silhouette width value is not exceptionally high, meaning that the clustering is not seen as very good (that would be the case for values near 1).

The two dimensional MDS of the distance matrix in Fig. 5 shows that a test function will generally lead to similar performance (close proximity) regardless of the dimension. The first MDS component seems to be adequate to describe the clustering. Observations with a value smaller than -50 are assigned to cluster one, the rest to cluster two. This however does not indicate how the clustering relates to the properties of the test

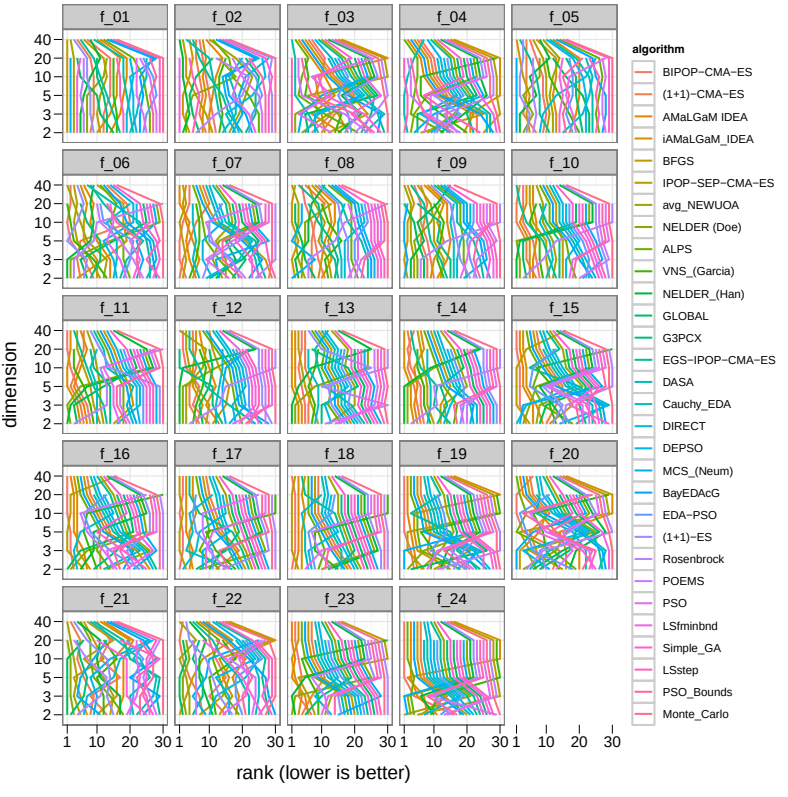

Fig. 3. Parallel coordinate plot for each function, showing the rank of each algorithm as the number of dimensions rise. Some algorithms dramatically loose ground as the number of dimensions rises (diagonal lines in upper right direction).

functions. Therefore, we look at the cross tabulation of the cluster against the function properties. To determine if there is any correlation between the property and the clustering, a Fisher Test was performed for the null hypothesis that the two are independent. For the four properties shown in Table 2 this hypothesis was rejected at the 1% level.

To further model the unknown decision boundary of the clustering with respect to the function properties, a classification tree (Fig. 6) was constructed using the four properties identified by the cross tabulation as having a relationship with the clustering. It comes as no surprise that the first split separates the highly multi-modal problems from all others. The matched 9 problems (27 observations in 5/10/20 dimensions) obviously require a specific type of algorithm, whether no, low, and medium multi-modality go together. Interestingly, the second split considers low variable scaling as a separate function group. Possibly, this reflects that some algorithms have basic means to adapt to rescaled variables where others do not possess such means. The third split between none and deceptive (Schwefel function) global structure is not that surprising. Global structure needs multi-modality, and after removing highly multi-modal functions, the Schwefel function is the only one that possesses a global structure (according to our

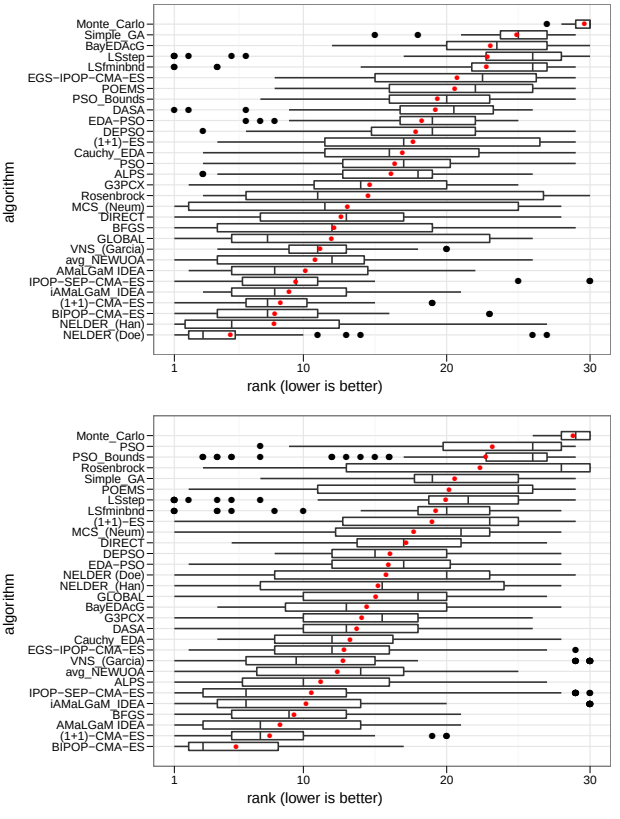

Fig. 4. Boxplot of the ranks for each algorithm in {2,3} (top) and {5,10,20} dimensions (bottom). The algorithms are ordered by mean rank which is marked by a red dot. This corresponds to the Borda ranking method. In low dimensions, the classical Nelder-Mead algorithm performs quite well.

Table 2. Cross tabulation of Multi-modality (a), Global Structure (b), Variable Scaling (c), Global / Local contrast (d) against the clusters found via PAM

(a)

	1	2
none	14	16
low	9	0
medium	3	3
high	0	27

(b)

	1	2
none	26	19
weak	0	3
medium	0	9
strong	0	12
deceptive	0	3

(c)

	1	2
none	12	9
low	1	26
medium	6	3
high	7	8

(d)

	1	2
none	14	16
low	9	24
medium	3	0
high	0	6

classification). It could be interesting to add more functions with low multi-modality but some global structure to the set. Basically, we obtain four different classes of functions with respect to algorithm performance.

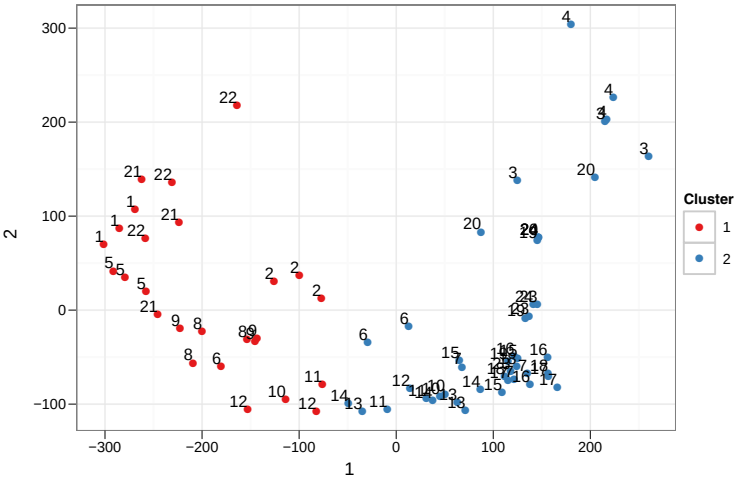

Fig. 5. MDS of the distance matrix obtained from the 5 to 20 dimensional test problems. Color denotes assigned clusters, numbers refer to function numbers. The 5, 10 and 20 dimensional instances of a test function tend to lie in close proximity.

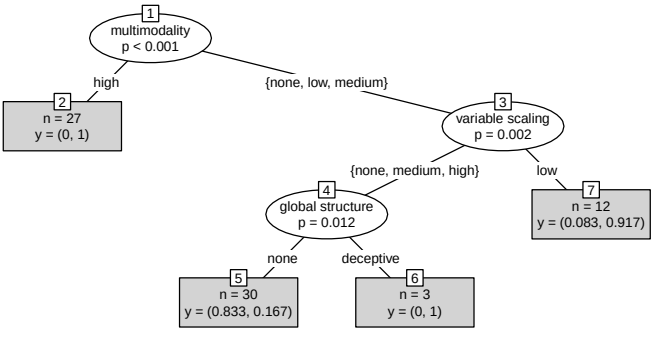

Fig. 6. Decision tree modeling the cluster boundary for the 5 to 20 dimensional function set. The vector y reflects the proportion of observations in each cluster class.

4 Conclusions and Outlook

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 true performance of a set of algorithms under test. Furthermore, we demonstrated how the structure of the function set can be explored. This leads us to define different groups or clusters of functions for which the ranking of the algorithms was essentially the same or very similar. To describe these groups we used decision trees for modeling the unknown cluster boundary. In the future, we would like to extend this work by including additional, measured features, and using the generated decision trees to develop heuristics

for supporting practitioners in the choice of an optimization procedure that works well for their problem type.

Acknowledgements. This work was partly supported by the Collaborative Research Center SFB 823 of the German Research Foundation.

References

1. de Borda, J.C.: Mémoire sur les élections au scrutin. Historie de l'Académie Royale des Sciences (1781)
2. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2009: Experimental setup. Tech. Rep. RR-6828, INRIA (2009), http://hal.inria.fr/inria-00362649/en/