Methods to characterize the behaviour of optimization algorithms

by

Olaf Mersmann

Presented in partial fulfillment of
the requirements for the degree of
Doktor der Naturwissenschaften

Referees:
Prof. Dr. Claus Weihs
Prof. Dr. Jörg Rahnenführer

Technische Universität Dortmund · Fakultät Statistik
Dortmund, Dezember 2014

Contents

Contents **1**

1 Motivation **3**

2 Characterization of Continuous Optimization Problems **9**
2.1 Contributed Material 9
2.2 Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis 9
2.3 Exploratory Landscape Analysis 14
2.4 Algorithm Selection Based on Exploratory Landscape Analysis and Cost-Sensitive Learning 16
2.5 Outlook 17

3 Travelling Salesperson Problem Generation **19**
3.1 Contributed Material 19
3.2 Local Search and the Traveling Salesman Problem: A Feature-Based Characterization of Problem Hardness 19
3.3 A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem 22
3.4 Outlook 23

4 Understanding unexpected Hypervolume Reductions **25**
4.1 Contributed Material 25
4.2 On the Distribution of EMOA Hypervolumes 25
4.3 Non-monotonicity of Observed Hypervolume in 1-greedy S-Metric Selection 27
4.4 Effect of SMS-EMOA Parameterizations on Hypervolume Decreases 28

4.5 Outlook 28

5 Accompanying Software **31**
5.1 ela 31
5.2 emoa 32
5.3 sendmailR 32
5.4 soobench 33
5.5 microbenchmark 34
5.6 BatchJobs and BatchExperiments 35
5.7 tspmeta 37

Bibliography **39**

A Contributed Material **43**

Chapter 1

Motivation

> *The worthwhile problems are the ones you can really solve or help solve, the ones you can really contribute something to.*
>
> — Richard Feynman

When I began the work on this thesis, my aim was to develop tools and techniques to better understand the structure of optimization problems. Before I discuss what that structure might be and how it might be used to better solve an optimization problem, I will fix a minimal amount of notation. Given $f\colon \mathcal{X} \to \mathcal{Y}$ and the partial order $\preceq$ over $\mathcal{Y}$ I will call the 2-tuple $(f, \preceq)$ an *optimization problem*. The function f is called the *objective* or *fitness function*, $\mathcal{X}$ the *search space* and $\mathcal{Y}$ the *objective space*. Depending on the research community, $\mathcal{X}$ may also be called the *parameter space* or the *domain* of the optimization problem. Given an element x of $\mathcal{X}$, its associated *objective value* $f(x)$ is often referred to as the *fitness* of x. Many of these terms reflect that I mostly work with researchers from the field of evolutionary computation. Other communities have different terms for the above concepts.

The aim of any optimization procedure, given a problem $(f, \preceq)$ is to find an $x^* \in \mathcal{X}$ such that

$$\forall x \in \mathcal{X}\colon f(x^*) \preceq f(x).$$

That is, the aim of the optimization process is to find an element of the search space that maps to the minimal element of $\{f(x)\colon x \in \mathcal{X}\} \subseteq \mathcal{Y}$ with respect to the given partial order $\preceq$. To simplify the following discussion, I implicitly assume that such an element exists. If in fact the problem is not bounded below, an ideal optimization procedure would deduce this somehow and return this fact to the user. Clearly this is not possible in the general setting described

above. Also note that my choice of minimizing (finding a minimal element) is arbitrary. By redefining the ordering any maximization problem (finding the maximal element) can be reformulated into a minimization problem and vice-versa.

The above definition looks rather abstract at first. When someone thinks of an optimization problem, they often implicitly assume that the objective space is the real numbers and that the natural order is used to judge the fitness of a solution. While this is often the case, the above definition also includes the class multi-objective optimization problems. Similarly, many assume that $\mathcal{X}$ is a metric space and again this does not have to be the case. A well known class of an optimization problems where $\mathcal{X}$ is not a metric space are travelling salesperson problems. There $\mathcal{X}$ is the set of all routes the salesperson may travel.

Using the mathematical structure of the domain $\mathcal{X}$ and the objective space $\mathcal{Y}$ an optimization problem can be assigned to any one of a broad set of problem classes. Some examples of problem classes are

Discrete problems: the search space is a finite set such as in the travelling salesperson problem mentioned above.

Continuous problems: the search space is a metric space. An example would be optimizing the proportions of certain assets in a portfolio or the variance parameter of the Gauss kernel in kernel logistic regression.

Multi-objective problems: the objective space is a product space. In feature selection one might want to minimize the number of features and the model error at the same time. The objective space would then be the product of the natural numbers less than the maximal number of features and the positive real numbers.

A problem may belong to more than one of these classes and there are problems which do not belong to any of the above classes. Nevertheless, the above classes are roughly speaking the main subdivisions of optimization. Methods which are applicable to discrete problems will in most cases fail to work on continuous problems. Solvers for continuous problems often exploit the neighborhood structure induced by the metric over $\mathcal{X}$ and are therefore not usable for all classes of discrete problems. Both continuous and discrete solvers usually assume that the objective space is a subset of the real numbers. Therefore

specialized methods are needed for multi-objective problems or problems where no total order of the objective space is given.

Within these broad classes there are further subdivisions based upon characteristics of the problems. In discrete optimization it is customary to classify problems based on their concrete domain. There are the already mentioned travelling salesperson problems, the more general vehicle routing problems, problems related to the satisfiability of Boolean clauses and many more such classes. Given that a discrete optimization problem can always be solved using exhaustive search, it is not surprising that the difficulty of a subclass of discrete optimization problems is most often judged by the complexity class in which the corresponding decision problem lies. Much theoretical work has therefore been performed to establish the complexity of many common and not so common discrete optimization problems and to find approximation schemes for these problems.

Continuous problems on the other hand are often further classified based on the structure of the mapping between the search space and the objective space. The most prominent subclass of continuous optimization problems are convex problems. This is the class of all optimization problems in which both the search space $\mathcal{X}$ and the fitness function f are convex. The reason this class is so prominent is that it is possible to derive simple to check necessary and sufficient conditions to prove that a point x is a (local) minimum of f using convex analysis. Combined with the fact that any local minimum is also a global minimum. In addition, if f is strictly convex then the global minimum is unique. It is therefore possible, given convex $\mathcal{X}$ and f and a point $x \in \mathcal{X}$, to prove that x is a local minimizer of f and therefore a global minimum of f and a solution to the continuous convex optimization problem $(f, \leq)$. The structure of a convex function f can often be exploited further to derive efficient algorithms to solve the associated optimization problem. In fact, in some cases the solution can be derived analytically, something that is not possible for many other types of optimization problems. Convex optimization plays a key role in statistical estimation. Many classical estimation procedures are formulated or can be reformulated as convex optimization problems. These include least squares and least absolute deviation estimation and many maximum likelihood estimation problems but also more modern methods such as support vector classification or regression.

For mutli-objective optimization I know no established further subclasses

that are prominent. Some authors define the subclass of many-objective optimization as any multi-objective optimization task where the objective space is the product of more than a few sets. This subdivision is motivated less by theoretical consideration and more by practical necessity because most established multi-objective optimization algorithms do not scale to tens or hundreds of objectives. Other characterizations exists that are based on the solution strategy. An example of such a subdivision would be the so called scalarization methods which transform the multi-objective problem into a single objective problem by combining the objectives into a single new objective that is then minimized using a single objective optimization algorithm. Much of the current research focuses on finding structures in the objective space that can be exploited by an optimization strategy. It is odd that little research focuses on exploiting the structure of the search space. The work done in this area largely relies on the existing body of work from single objective discrete or continuous optimization.

Given that there is already an extensive body of research into the structure and characteristics of many subclasses of optimization problems, what prompted me to work on this topic in my thesis? My initial interest was sparked by a class of problems called black-box continuous optimization problems. This subclass is characterized by the fact that f is considered to be a black box, that is apart from the domain and objective space nothing else is known about the structure of f. These types of problems often arise in engineering or other natural sciences where f might be a simulation or even a real experiment. Much of the mathematical community ignores this problem class because little can be proven about them theoretically. An exception is the result by Auger and Teytaud (2010) which applies to all continuous optimization problems and states that the well known No-Free-Lunch theorems do not apply to this domain.

Given the practical relevance of the problem class, I set out to use experimental methods to empirically characterize the structure of a black-box function. The aim of my work was to derive a characteristic fingerprint for a function. In turn, such a fingerprint is a useful tool in several different settings:

Function grouping: finding similar functions with almost the same characteristics is essential for validation purposes in benchmark studies.

Algorithm selection: instead of relying on theoretical properties of a func-

tion, the empirically determined characteristics can be used to select a set of candidate optimization algorithms for the task at hand.

User insight: By comparing the characteristics of an unknown function with those of functions with known theoretical properties it should be possible to gain insight into the true theoretical properties of the unknown function without its analytic form.

Together with my coworkers I have made progress in all three areas. The corresponding articles are summarized in Chapter 2.

Based on the promising results obtained characterizing and grouping black-box continuous optimization problems, I developed the idea of generating similar optimization problems to a given problem. The core idea was to search for functions with similar characteristics and assume that they should also show similar behaviour during the optimization. In the continuous domain this is a daunting task. The search space would be the set of all computable functions from the parameter space to the objective space. Adding strong assumptions such as continuity or differentiability do not make the search space significantly smaller. Therefore my attention turned to discrete problems. More specifically I chose to work with travelling salesperson problems (TSPs) because I had colleagues with expert knowledge in this domain. Here a problem instance is well defined and it is trivial to generate new instances. There are also plenty of characteristics for TSPs in the literature so no new features had to be developed. Using a guided search procedure, I, together with my colleagues, created instances with varying characteristics spanning the space complete feature space. Sadly it turned out that the characteristics and the performance of a simple TSP solver heuristic on these generated instances did not correlate. We did however manage to generate "easy" and "hard" instances. Based on this observation we reworked our guided search procedure to specifically generate both easy and hard instances for the employed search heuristic. More details on the procedure and the obtained results we published are given in Chapter 3.

Finally in Chapter 4 I present joint work with a former student on rare anomalies discovered during the systematic evaluation of a multi-objective optimization algorithm. While it may seem that this is not related to the previous work, it builds on the same principles. The algorithm under test, the so called SMS-EMOA, is theoretically well studied. We were able to experimentally ver-

ify many properties of the algorithm that had theoretically been derived for simpler versions of the algorithm. One crucial property however was violated. The dominated hypervolume, a measure for the progress of the optimization, is supposed to monotonically decrease over the course of an optimization run when employing the SMS-EMOA. In our dataset we found sporadic increases. Our initial suspicion was that there was an implementation error or that numerical instabilities were the root cause. One by one we could eliminate these until we had a small and concise test case that could reliably reproduce the observed increase[1]. Starting from there, we found the true root cause and studied it. The results of this work can be found in Chapter 4.

The rest of this thesis is structured as follows. The following three Chapters cover three distinct areas of my thesis work. There is a chapter on the characterization of continuous black box optimization problems (Chapter 2). Following this there is a body of work on the systematic generation travelling salesperson problems (Chapter 3). Finally there is a chapter that covers the study of hypervolume decreases in SMS-EMOA runs (Chapter 4). Before the conclusion, Chapter 5 covers the accompanying software I (co)developed during the course of my thesis work. After the conclusion there is an appendix containing copies of all the peer-reviewed articles that I submit as part of my thesis work.

[1]This may sound simpler than it is given that the SMS-EMOA is a randomized algorithm

Chapter 2

Characterization of Continuous Optimization Problems

2.1 Contributed Material

- *Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis* (Mersmann et al., 2010a)
- *Exploratory Landscape Analysis* (Mersmann et al., 2011)
- *Algorithm selection based on exploratory landscape analysis and cost-sensitive learning* (Bischl et al., 2012a)

2.2 Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis

In Mersmann et al. (2010a) we laid out a generic framework to systematically benchmark evolutionary algorithms for black-box continuous optimization. The design of this framework is based on the work presented in (Mersmann et al., 2010c). We then applied this methodology to results of the 2009 Workshop on Black-Box Optimization Benchmarking (BBOB) (Hansen et al., 2009). The organizers of the BBOB workshop selected a testbed of 24 noise free continuous functions for the benchmark. Each function was chosen for a particular challenge that it poses for an optimization algorithm. For example, for the sphere function (function 1 in the BBOB function set), we would like

to observe quadratic convergence towards the global minimum. In addition the functions are assigned to one of five subgroups (Finck et al., 2009):

- Separable functions (Functions 1 – 5)
- Functions with low or moderate condition (Functions 6 – 9)
- Functions with high condition and unimodal (Functions 10 – 14)
- Multi-modal functions with adequate global structure (Functions 15 – 19)
- Multi-modal functions with weak global structure (Functions 20 – 24)

The idea behind these groups is that they are somewhat homogeneous and it is expected that an algorithm performs equally well on all candidate functions in a subgroup.

In addition, each function is scalable, that is, the dimensionality of its parameter space can be chosen at will. In the BBOB setting, the sizes of the parameter space were fixed to 2, 3, 5, 10, 20 and optionally 40 dimensions. This gives 120 (optionally 144) base functions which an algorithm has to solve to participate in the benchmark. Furthermore, each function in composed with a random transformation (rotations and/or affine mappings) of the input parameters to generate a so called function instance. This is done 5 times to obtain a total of 600 (720) unique function instances that have to be solved. These transformations are applied under the assumption that an algorithm should be invariant under these transformations of the input space. Finally, because most of the optimization algorithms for these types of problems are randomized, a participant is expected to submit three runs on each function instance[1].

Compared to the work in Mersmann et al. (2010c) we extend ranking based approach to the analysis of benchmark experiments in Mersmann et al. (2010a) with several novel visualizations. Given the structured nature of the function set used in BBOB, we also advocate the analysis of subsets of the benchmark data instead of solely relying on consensus rankings as described in Mersmann et al. (2010c). For this we combine the results of an optimization algorithm on functions with the 2 and 3 dimensional parameter spaces into a group of

[1] In later workshops the three repetitions were replaced and instead 15 function instances had to be solved.

low dimensionality and the 5, 10 and 20 dimensional functions into a group of *moderate dimensionality*. Then a ranking is computed for each function and parameter space size combination based on the expected running time (ERT) of each optimization algorithm. From this we obtain 120 ranks, 48 for the low dimensionality group and 72 for the moderate dimensionality group, for each participating optimization algorithm. The distribution of these ranks is shown in Figure 2.1. The algorithms are sorted according to their mean rank which corresponds to their position in a Borda consensus ranking. From the figure it is clear that, depending on the dimensionality of the search space, a different class of algorithms should be chosen. In low dimensions, classical algorithms such as Nelder-Mead variants do exceedingly well. For higher dimensional problems these methods fail. Instead randomized search heuristics, in particular variants of the covariance matrix adapting evolutionary strategy (CMA-ES) are very competitive. An influence of the search space dimensionality on the choice of algorithm is to be expected.

We can also deduce that for the moderate dimensional group of functions, if we chose to always use the BIPOP-CMA-ES, we would never do worse than an average ranked algorithm, i.e. the worst case performance of the BIPOP-CMA-ES places it around the 15th place in the ranking. On the other hand we can see some very specialized algorithms that excel at some functions but fail on the majority of functions. Examples for these are the LSstep and LSminbnd procedures.

In the second half of Mersmann et al. (2010a) we tackled the problem of finding subsets of the functions on which the observed performance of the optimization algorithms is similar. Here we defined similar behavior to mean that the rankings induced by the functions are similar. To measure the similarity between the rankings we employed the symmetric difference (SD) between the binary relations induced by the rankings. We focused on the moderately dimensional problem instances because we feel this is the current sweet spot for black-box optimization. Lower dimensional problems can often be solved using a combination of visualization and existing conventional algorithms and higher dimensional problems are often intractable with current methods.

By calculating the pairwise distance between all rankings we obtain a distance matrix that captures the global structure of the set of all rankings. We apply the PAM clustering technique to the distance matrix to find groups of similar rankings in the data. The resulting clusters are visualized using MDS

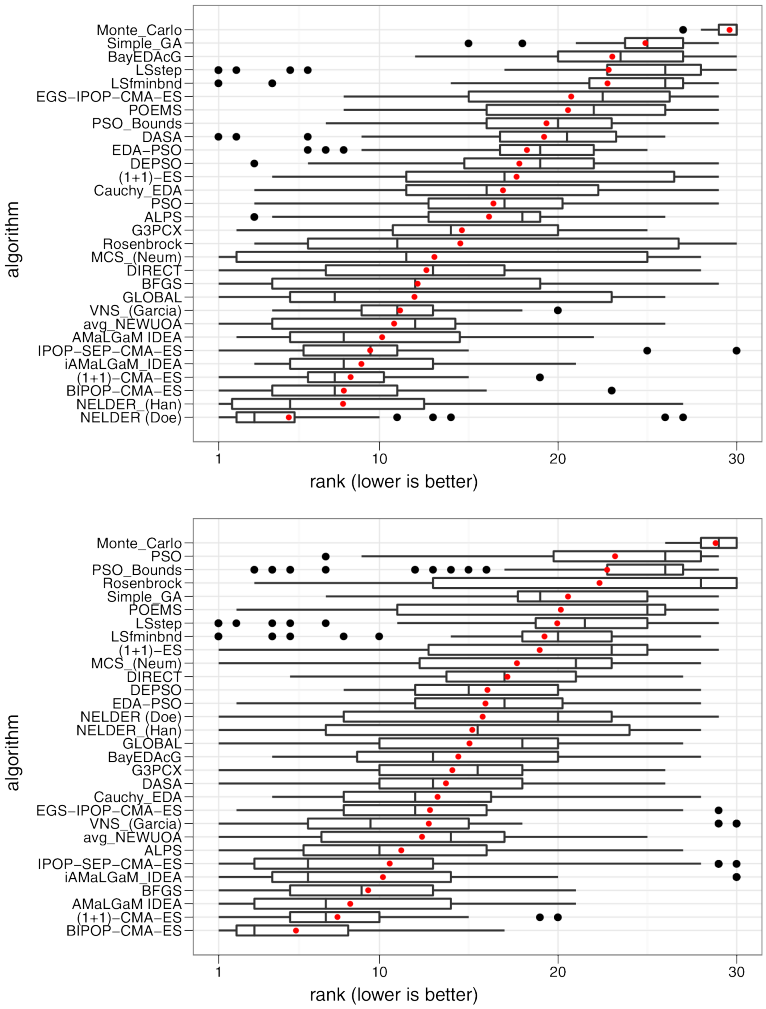

Figure 2.1: Distribution of the rank each algorithm achieved in the low dimensionality (top) and moderate dimensionality (bottom) group. The red dot denotes the mean rank of each algorithm. Figure taken from Mersmann et al. (2010a).

to generate a two dimensional point cloud that has a similar distance structure as our ranking induced distance matrix. We observe that rankings for the same function but with different parameter space sizes are closely grouped. The two clusters we identified are clearly separated in the plot of the MDS results. Since we know that all the rankings within a cluster are similar, we can sensibly choose a default algorithm for each cluster by calculating a consensus ranking among all the functions assigned to one cluster and picking the best or one of the top performers from the consensus ranking. While this will work, it is of little practical value. Given a new black-box optimization problem, we would have to run all algorithms to determine the ERT of each algorithm. From the calculated ERTs we could then derive a ranking for the algorithms and then, based on this ranking, assign the function to one of our clusters. Given the cluster, we would now use the algorithm that performed best in the cluster consensus ranking to solve our optimization problem. Clearly this is nonsense! From the ranking we calculated to assign the function to one of our clusters we already know which optimization algorithm is best for this function.

What we want is a method to assign a function to one of the clusters without having to perform a benchmark experiment. We proposed the following approach. First, each (benchmark) function is assigned certain high-level characteristics by a domain expert. A summary of these high-level characteristics and their values for each of the 24 functions in the BBOB function set is shown in Table 2.1. Next, we use the CART[2] methodology to create a simple classification rule which reproduces the boundary between the two clusters given the high level characteristics of the functions. For this we combine the data in Table 2.1 with the mapping of function to cluster we obtained using PAM. The classifier is then trained to predict the cluster given the high-level properties of the function. Given a new function, a practitioner would now have to derive the high-level characteristics of the function and could then predict the cluster it belongs to using the decision tree. While this is more realistic and much less labour intensive than rerunning the benchmark with all algorithms, it is still cumbersome and prone to error. The assignment of the high-level characteristics is not an exact science.

A remedy for this is what we call *Exploratory Landscape Analysis* (ELA). It is introduced in Mersmann et al. (2011) and its main results are summarized

[2]Classification and Regression Trees

Table 2.1: 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	m.-mod.	g.-str.	sep.	scal.	hom.	basins	g.-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

in the following section.

2.3 Exploratory Landscape Analysis

Starting from our initial success in characterizing the BBOB function set, we extended the approach in Mersmann et al. (2011). Because, as detailed in the previous section, it is difficult to obtain the high-level features for a black-box function, we set out to predict the high-level features of a function using easily derived properties of the function. We call these properties *low-level features* or *ELA features*. Given a classification rule which predicts the high-level properties of a function and given the (computable) ELA features of the function, we can then construct a chain of decision rules that, given the ELA features, predicts the cluster a (new) function belongs to. From the cluster we can derive a good optimization strategy for the function. Therefore, if the proposed ELA features provide adequate information to predict the high-level

characteristics, we have a method that, without expert knowledge, can give a practitioner a good algorithm recommendation using only function evaluations of his black-box function.

The design of the *ELA features* was very much driven by intuition and experimentation. Among the many proposed features are all kinds of summary statistics that characterize the distribution of the function values. To approximate this distribution we generated a uniform random sample from the parameter space and evaluated this sample. For many features, this is enough to calculate their value. However, some features require additional function values to estimate some property. One example of such a feature is the convexity feature. It estimates the probability that a function is (locally) convex on a randomly chosen line segment within the parameter space. To estimate this probability, two points $\vec{x}_i$ and $\vec{x}_j$ from the uniform random sample and a random $\alpha \in (0, 1)$ are chosen. We then check if $f(\alpha\vec{x}_i + (1-\alpha)\vec{x}_j)$ is greater than, less than or equal to $\alpha f(\vec{x}_i) + (1-\alpha)f(\vec{x}_j)$. This procedure is repeated many times and the probability of observing a "greater than", "less than" or "equal" function value is estimated from the results.

Clearly this type of feature requires additional function evaluations. We would therefore prefer not to use it in the prediction if we can obtain similar results using only features derived from the initial uniform sample. On the other hand, once we have added the feature describing the probability of observing a "greater than" result, adding the feature for the "less than" probability is free in terms of additional function evaluations. The cost of a classification rule in terms of required function evaluations to calculate the features required by said classification rule is non-trivial to calculate. Yet it is important to minimize this cost because it is part of the overall cost, again in terms of number of function evaluations, of the optimization procedure.

To minimize both the missclassification error (MCE) and the number of function evaluations used to calculate the feature vector required by the classification rule, we employed a multi-objective optimization algorithm – namely an SMS-GA[3]. Because under certain circumstances no additional function evaluations are required to add a feature, we also added an artificial third objective: the number of features used by the classification rule. With this objective, we wanted to avoid that our solutions become bloated with irrelevant but cost-free

[3]S-Metric Selection genetic algorithm. A multi-objective optimization algorithm that maximizes the dominated hypervolume of a set of solutions.

features that do not harm the MCE.

Using the SMS-GA and a random forest as our classification procedure, we built two sets of models. One set to predict the high-level features of the 24 BBOB functions and a second set to predict to which of the five function groups a given function belongs. We discuss and visualize the results of the two optimizations in detail. Overall we were able to predict both the high-level features as well as the function group with high accuracy and few additional function evaluations using several different combinations of our proposed ELA features.

2.4 Algorithm Selection Based on Exploratory Landscape Analysis and Cost-Sensitive Learning

Given the success we had in predicting the high-level features of BBOB test functions in Mersmann et al. (2011) and choosing a good optimization algorithm from a small portfolio for a given BBOB test function based solely on its high-level features (Mersmann et al., 2010b), the next logical step is to use the ELA features introduced in Mersmann et al. (2011) to directly predict the best algorithm for a given BBOB test function. In (Bischl et al., 2012b) we combined the 2009 and 2010 BBOB datasets and then pruned the list of algorithms in our portfolio down for this task. First we eliminated all algorithms that were not best w.r.t the expected running time (ERT) for at least one function. This left the algorithms listed in Table 2.2. From the median and maximum relative ERT[4] values we can deduce that if we chose to always rely on the IPOP-ACTCMA-ES algorithm, our median ERT would rise by a factor of 3.72 and we would never need more than a factor of 2655.55 function evaluations compared to the best algorithm in expectation.

We could attempt to use the 14 algorithms in Table 2.2 for our portfolio. Instead, we elected to simplify the task by selecting the optimal subset of four algorithms from the 14 algorithms. These are Line Search-fminbdn, Line Search - STEP, BFGS and BIPOP-CMA-ES. The two line search variants are specialists for some of the functions, BFGS does well on smooth unimodal functions (i.e. the convex functions in the set) and BIPOP-CMA-ES is a

[4]The relative ERT is obtained by dividing the ERT of an algorithm by the best observed (minimal) ERT on that function over all algorithms.

Table 2.2: List of algorithms which are best w.r.t ERT on at least one of the 24 BBOB test functions. In addition to the algorithm name, the functions it performed best on, and its median and maximum relative ERT over all functions is given.

Algorithm	**Best on Function**	**med (rel. ERT)**	**max (rel. ERT)**
AVGNEWUOA	8	29.67	58731.09
BFGS	1, 10	31.55	58731.09
BIPOP-CMA-ES	17	3.83	5873.11
DE-F-AUC	18	41.13	11867.07
FULLNEWUOA	6	39.36	58731.09
GLOBAL	12, 21, 22	22.60	58731.09
iAMALGAM	23	10.02	2287.43
IPOP-ACTCMA-ES	7, 11, 13	3.72	2655.55
IPOP-CMA-ES	15, 16, 19	3.40	5743.86
Line Search-fminbnd	2	172.23	58731.09
Line Search - STEP	3, 4	417.47	13494.28
MCS	5, 9	39.55	58731.09
MOS	20, 24	4.27	1186.71
NEWUOA	14	28.01	58731.09

generally robust strategy from the CMA-ES family. If we had an oracle which always picked the best algorithm from this subset, we would obtain a median relative ERT of about 1.45 and a worst case relative ERT of 4.169 – both much better than any single algorithm in the above table.

So how do we approximate such an oracle? This is generally coined the *algorithm selection problem* and here we used a cost sensitive one-sided regression approach introduced by Tu and Lin (2010). Using this approach, we were able to obtain an approximation of the oracle that had a median relative ERT of between 1.54 and 2.70 depending on the feature set used and the granularity of the cross-validation.

2.5 Outlook

My work on the characterization of black-box optimization benchmarking started with the wish to formalize the benchmarking procedures used. From there, it evolved into characterizing continuous black-box functions using algorithmically defined instead of analytic properties. These low-level or ELA

features were then used to successfully estimate the high-level properties of black-box functions and select an appropriate optimization algorithm from a small portfolio of algorithms. All of this work is based on the tremendous work done by the BBOB team to assemble a large and diverse dataset with different test functions and many different algorithms and algorithm variants.

One might think that after this work, there is little else which needs to be researched in this area. The exact opposite is true! While the original 24 functions in the BBOB test set were picked with great care to cover different typical functions, the set is far from complete. Looking at Table 2.1, it is obvious that some combinations of high-level features are under represented or not present at all in the current function set. In addition, during the training and verification of the different classification rules, it became clear that a more diverse function set would make hyperparameter tuning, cross-validation and verification in general much easier. On the other hand, I observed that many real world problems from statistics and engineering are not similar to any of the BBOB functions when compared using the ELA features. All of this suggests that it would be nice to have a generator that is able to create a collection of functions which have similar given ELA properties. Given such a procedure, it would be possible to use classical design of experiment methodologies to systematically explore the space of all black-box optimization problems by generating functions which fill the ELA feature space. This project is my main research focus for the foreseeable future.

The work my colleagues and I have published has encouraged others to adopt this data-driven approach to problem characterization (see Abell et al., 2012; Morgan and Gallagher, 2012; Malan and Engelbrecht, 2013; Munoz et al., 2013). I sincerely hope that other groups continue their work in this area.

Chapter 3

Travelling Salesperson Problem Generation

I am a travelling salesman. I deal in ideas.

— Martin Kippenberger

3.1 Contributed Material

- *Local Search and the Traveling Salesman Problem: A Feature-Based Characterization of Problem Hardness* (Mersmann et al., 2012)
- *A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem* (Mersmann et al., 2013)

3.2 Local Search and the Traveling Salesman Problem: A Feature-Based Characterization of Problem Hardness

In Mersmann et al. (2012), my coauthors and I applied an idea similar to exploratory landscape analysis to travelling salesperson problems (TSPs). In the discrete optimization community, the idea of characterizing problem instances using (empirical) features of the instance has already established itself. We were therefore not burdened with the development of novel features but relied on those introduced in the literature. At the same time, solving a (large) TSP

to optimality is difficult or impossible using current techniques[1]. This has led to the development of many different heuristics that return approximations of the optimal route of a TSP. But instead of choosing the optimal heuristic for a given problem instance from a portfolio, as done in my study of continuous black-box optimization algorithms in the previous chapter, our aim in this paper was to *generate* TSP problem instances which a given heuristic, here 2-opt, solves well and instances on which it fails. Generating instances that 2-opt solves well, also called *easy* instances, as well as instances on which it fails, also called *hard* instances, requires a definition of difficulty. In previous studies, authors have chosen to measure the runtime of the heuristic algorithms to characterize the difficulty of a problem. We felt that this approach falls short of the intended goal and instead opted to use the approximation quality, i.e. the length of the tour returned by the heuristic divided by the optimal tour length, as our criterion to judge the difficulty of a problem instance. An easy instance would have an approximation ratio of close to 1, i.e. the heuristic almost solves the TSP instance optimaly. Large approximation ratios on the other hand indicate that the heuristic is not able to find a reasonable route for this instance.

Compared to black-box continuous problem instances, generating a TSP instance is trivial, one only needs to generate a symmetric distance matrix which gives the distance from each city to every other city. In our case, we went one step further and opted to generate Euclidean TSP instances. For these, we merely pick the coordinates of the cities within the unit square. One approach to generate *easy* and *hard* instances would be to pick a large collection of instances at random, calculate their optimal tour and the expected 2-opt tour length and then sort them by their approximation ratio. It is unlikely that this approach will yield extreme instances. We therefore opted to use a customized genetic algorithm (GA) to evolve both easy and hard instances. Even this approach has a considerable computational burden because the optimal tour must be calculated for each generated instance. This also limited the size of the instances which we were able to study.

Nevertheless this approach allowed us to generate a set of *easy* and *hard* instances for the 2-opt heuristic. Characterizing these instances using the empirical features from the literature allowed us accurately predict the problem

[1]Recall that the TSP is an NP-hard problem and that no efficient deterministic solver is known for general TSPs.

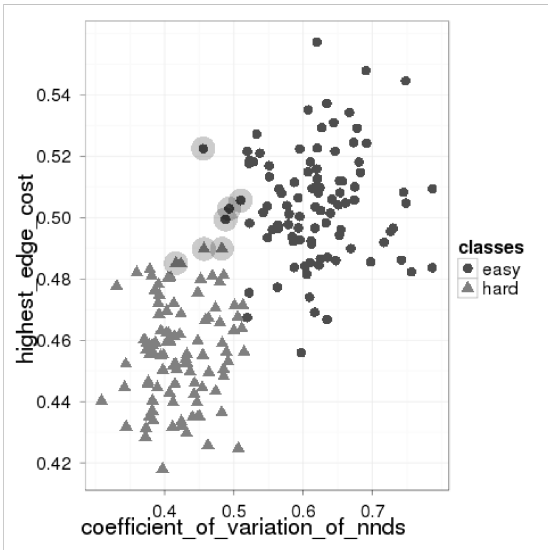

Figure 3.1: Figure taken from Mersmann et al. (2012).

class (*easy*, *hard*) for each instance using a simple decision tree. Figure 3.1 shows a scatter plot of the instances by their two main features. The seven misclassified examples are marked by an underlying circle. To extend our result to other instance types that are neither easy nor hard, we devised a scheme to generate many instances *between* two extreme instances. For this we associated each city in an easy instance with a city in a hard instance. Then we defined new problem instances where the coordinates of each city are the convex combination of the coordinates of the easy and hard instance city. Using this approach, we are slowly moving each city from its easy to its hard position. Our conjecture was that these "in between" instances should be of average hardness. We were able to verify this and also show that the feature values of such instances smoothly interpolate between the feature vectors of an easy and a hard instance. The method therefore allows us to smoothly vary the difficulty of a problem instance and observe the changes in algorithm behaviour, coupled with the changes in the instances features.

All of the software required for the experiments was published in the R package tspmeta. See also the description of the package in Section 5.7.

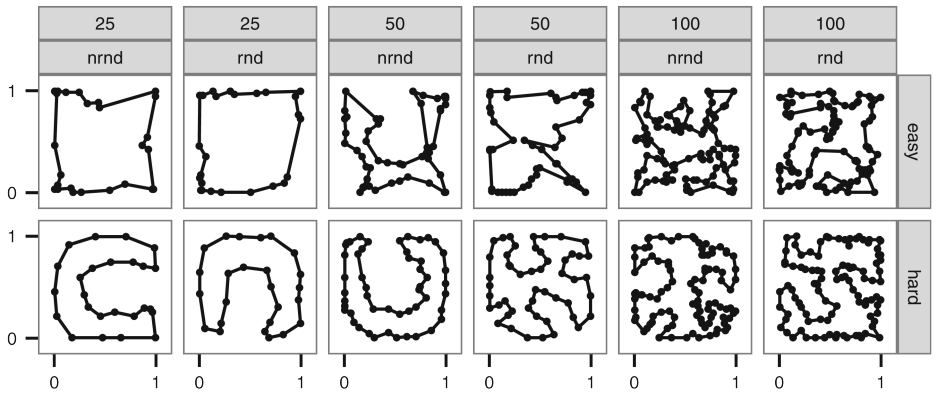

Figure 3.2: Example easy and hard instances for the two rounding schemes studied in Mersmann et al. (2013). The line connecting the cities of each instance depicts the optimal tour. Figure taken from Mersmann et al. (2013).

3.3 A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem

Given our success in creating novel easy and hard to solve instances in Mersmann et al. (2012) and based on feedback from domain experts, we extended the ideas in Mersmann et al. (2013). The changes compared to our previous work include a revised genetic algorithm (GA) that has a modified normalization step as well as an additional optional rounding step. The normalization is necessary to ensure the cities in an instance are all within the unit square and at the same time their convex hull should cover the maximal area of the unit square. The reason we normalize at all is that some features are not scale invariant, a property we would prefer in a feature but that is not given for some of the features found in the literature. The other major change in the GA was the addition of an optional rounding step. It was motivated by the observation from a domain expert that in practice, the cities of a TSP instance often lie on a grid. Examples of such instances are problems that arise in the area of VLSI design and PCB production.

As in Mersmann et al. (2012), we were able to train an almost perfect classifier for the instance type (easy, hard), regardless of the rounding scheme used or the size of the instance. Example instances are shown in Figure 3.2. The morphing procedure was also enhanced by including a better point matching

scheme. Here we chose to use a greedy scheme that matches points to their closest neighbor in the other instance. In addition we also normalized the instances w.r.t. their rotation by rotating each instance around its centroid such that the eigenvector of the covariance matrix of the coordinates is parallel to the X-axis and there are more points above and the right of the centroid. This transformation does not change the optimal tour but may require a rescaling of the instance because points may be rotated such that they lie outside of the unit square. Using the morphing procedure we produce intermediate instances of medium difficulty and study the influence of the features on the performance for this larger dataset fitting a regression model that predicts the approximation quality based on the feature vectors using the MARS method. The results are analyzed and presented visually using partial dependency plots and residual plots.

Finally we validated our results by applying a similar analysis to instances of the TSPLIB library of standard TSP instances. Again we were able to make an accurate prediction of the approximation quality which validates our approach.

All source code used for the article is contained in the R package tspmeta and can be downloaded from CRAN for further use by other researchers.

3.4 Outlook

After characterizing continuous black-box optimization problems using empirically features, I turned my attention to the discrete domain more specifically to the domain of travelling salesperson problems. The empirical characterization of problem instances is already an established idea in this field and it was therefore not necessary to invent novel features. Instead I focused my research around a procedure to generate new problem instances that exhibited certain properties. This lead to the development of a genetic algorithm that is able to find instances that are either easy or hard w.r.t. the approximation quality for a given heuristic. Here the methodology differs from my approach in the continuous case. Instead of using a set of instances that covers the feature space, an algorithm specific set of instances was created to showcase the strengths (easy instances) and weaknesses (hard instances) of the algorithm. To generalize the results we devised a morphing strategy that is able to generate many instances that lie in between the easy and hard set. These instances were then

used to model the general behaviour of the algorithm and test.

The method presented here can be employed to analyze other TSP heuristics. Nallaperuma et al. (2013) study an ant colony algorithm using the ideas presented in Mersmann et al. (2012, 2013). Hutter et al. (2014) also extend the work to the general problem of algorithm runtime prediction. In principle similar approaches should be feasible for almost all discrete optimization problems. For these it is usually much easier to generate instances and find meaningful mutation and cross-over operations. In the future I would like to extend this work to the continuous domain where it is less clear how one would "generate" a function for an optimization problem.

Chapter 4

Understanding unexpected Hypervolume Reductions

Facts are the air of scientists. Without them you can never fly.

— Linus Pauling

4.1 Contributed Material

- *On the Distribution of EMOA Hypervolumes* (Mersmann et al., 2010b)
- *Effect of SMS-EMOA Parameterizations on Hypervolume Decreases* (Judt et al., 2012a)
- *Non-monotonicity of Observed Hypervolume in 1-greedy S-Metric Selection* (Judt et al., 2012b)

4.2 On the Distribution of EMOA Hypervolumes

In recent years the dominated hypervolume of a Pareto front approximation has become the de facto standard to gauge the performance of multi-objective optimization algorithms. This led to the investigation which is at the heart of Mersmann et al. (2010b). For a benchmarking study, we performed many repeated runs of two popular multi-objective optimization algorithms, NSGA-II and the SMS-EMOA, on three common 2D test problems (ZDT1 to ZDT3). The partial results of the benchmark study are published in Mersmann et al.

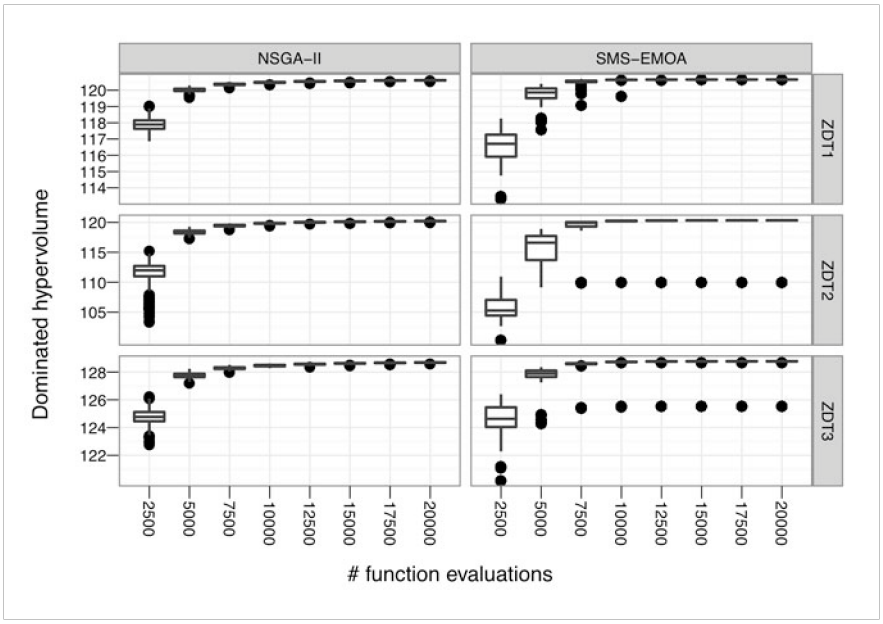

Figure 4.1: Figure taken from Mersmann et al. (2010b).

(2010c), here we are only interested in an exploratory analysis of the distribution of the obtained dominated hypervolumes. In an ideal setting, we would expect the distributions to be unimodal and fairly concentrated around the dominated hypervolume of the true Pareto front. Looking at Figure 4.1 we can see that this is not always the case. For ZDT2 and ZDT3 there are outliers in the distribution for the SMS-EMOA. Since the SMS-EMOA optimizes the dominated hypervolume directly, it is plausible that these outliers are local optima from which the algorithm cannot escape.

We were also able to show that the SMS-EMOA has a harder time in the initial phase of the optimization to approximate the Pareto front. It is only after around 10 000 function evaluations that the SMS-EMOA is able to compete with the NSGA-II results. Again, this is surprising because the NSGA-II does not care about the dominated hypervolume of the Pareto front approximation and instead optimizes the non Pareto compliant crowding distance of the individual solutions.

While these are not great scientific discoveries, it is nevertheless interesting. In particular it is surprising that no research team had observed these phenomena previously. Not because they lacked the experimental data but because exploratory data analysis is often neglected in these types of studies.

The real value of these experiments will become apparent in the following two articles.

4.3 Non-monotonicity of Observed Hypervolume in 1-greedy S-Metric Selection

Spurred by the observations made in Mersmann et al. (2010b), a larger benchmark study was performed that included a systematic variation of the hyperparameters of the SMS-EMOA. The aim of the study was to identify factors that could inhibit the observed early convergence. During this study, further data analysis revealed another surprising phenomenon. The core principle of the SMS-EMO algorithm is the steady increase of the dominated hypervolume. In every iteration of the algorithm a new solution is created and added to the population. Then the solution with the smallest hypervolume contribution is pruned from the population. This procedure should not result in decreases in the dominated hypervolume. Nevertheless we were able to observe relatively frequent small drops in the dominated hypervolume w.r.t. a fixed reference point.

In Judt et al. (2012b) the magnitude of these drops is correlated with the parameterization of the SMS-EMOA and different root causes for the drops are identified and studied. For the 2D case most drops can be attributed to a "trick" the algorithm employs. The corners of a Pareto front approximation are well defined in 2D, they are the two points that are extreme w.r.t. the two objectives. These solutions are considered to be so important that they are never eligible for selection during the population reduction. As such, it is possible that the dominated hypervolume decreases but the spread of the Pareto front increases. This is a trade-off that the original algorithm designers chose in order to obtain more meaningful approximations. In the case of a 3D objective space no such special case exists. Here there is a different cause but it is again related to the points on the edge of the Pareto front approximation. Their contribution to the total dominated hypervolume is governed by the location of the reference point which is used for the hypervolume calculation. For numerical reasons this reference point is constantly adapted. This entails that the SMS-EMOA is in fact solving a dynamic optimization problem, each change of the reference point entails a change in the fitness function the SMS-EMOA is maximizing. While informal discussion of this property appears to

have taken place between the algorithm designers[1], its effect had been considered irrelevant. The practical implications may seem dire at first, but we conjectured that they may indeed be one of the reasons that the elitist $(\mu + 1)$ SMS-EMOA is so successful in practice even though theoretically it has been proven that an elitist selection strategy cannot escape from local optima.

4.4 Effect of SMS-EMOA Parameterizations on Hypervolume Decreases

To further characterize the observed hypervolume decreases described in Judt et al. (2012b), a systematic analysis of the SMS-EMOAs hyper-parameters was performed in Judt et al. (2012a). We were able to show that the mean decrease in hypervolume is correlated with the mean increase in hypervolume in successful steps of the algorithm. This suggests that tuning for "maximal progress" also allows the algorithm to escape local optima by allowing for seldom large decreases in hypervolume. At the same time we were able to show that empirically smaller population sizes lead to a larger number of decreases which is consistent with our expectations. Finally we were able to show that the probability of a decrease is governed to a large extent by the parameterization of the mutation operator. The cross-over operator has little or no influence on the number of observed decreases. The biggest difficulty in extending these results further is the lack of a numerically stable and accurate implementation of an algorithm to calculate the hypervolume of an arbitrary point set. We conjecture but cannot prove that the overall impact of the decreases, while measurable is marginal on the final result because of the highly non-linear scale on which the hypervolume measures progress. Once the algorithm achieves a good approximation of the Pareto front any progress w.r.t. the hypervolume indicator is barely measurable.

4.5 Outlook

While studying the distribution of the dominated hypervolume of solutions returned by different multi-objective optimization algorithms, I showed that under certain conditions outliers could be observed. These outliers are artifacts

[1]Note that one of the coauthors of the article, Boris Naujoks, is one of the original designers of the algorithm.

which indicate early convergence to a local optimum by the algorithm. In turn, this behaviour led to a more detailed study of the SMS-EMOAs behaviour under different parameterizations. Here another unexpected phenomenon was observed, namely decreases in the supposedly monotonically increasing hypervolume of successive Pareto front approximations. Root causes for these decreases were identified and the influence of the parameterization on their occurrence studied. Further work in this area is hampered by the lack of numerically stable algorithms for the hypervolume calculation.

Chapter 5

Accompanying Software

> Publishing a paper without the code is not enough.
>
> — L. Amber Wilcox-O'Hearn

In the course of my studies I have co-authored several free and open source R packages. They can be categorized into two groups. On the one hand there are helper packages which contain utility functions to ease software development and help manage the large result sets produced by the experimental studies presented previously. Packages in this category include microbenchmark, sendmailR, soobench, BatchJobs and BatchExperiments. The other class of packages contains core functions used during the experiments. These are the packages ela, emoa and tspmeta. Often their development started as a loose collection of R scripts which later turned into a package so that others can reproduce and extend the work of my co-authors and my own work.

In what follows, each software package will be briefly described and its relevance to my research will be highlighted.

5.1 ela

The ela package is, as of writing, unpublished because it is still under heavy development. It contains robust implementations of the ELA (exploratory landscape anaylsis) features as used in Mersmann et al. (2010a, 2011) and Bischl et al. (2012a). The reason it has not been published to CRAN is that work is underway to extend the ELA features to multi-objective problems. For this, some of the public interfaces of the package needed and need to be adjusted or extended in incompatible ways. Since I strive to not break

backwards compatibility once one of my packages is publicly released, I opted against an early release. Instead snapshots are available from on my website[1].

5.2 emoa

Most evolutionary multi-objective optimization algorithms contain some common building blocks. The purpose of the emoa package is to provide these building blocks so that authors may quickly implement new optimization algorithms or create problem specific variants of an existing algorithm. It contains functions to calculate different quality measures such as the hypervolume or the R2 indicator, a routine to perform efficient non-dominated sorting, different selection operators and a small collection of multi-objective test functions. The package was instrumental for the work done in Mersmann et al. (2010b) and Judt et al. (2012a,b). Initially we attributed the observed hypervolume decreases to a bug in the software but through careful testing we were able to construct a test case which produced the same effect. Instrumental for this was the modular design of emoa which allowed us to focus on the selection procedure instead of viewing the optimization algorithm as a black box.

5.3 sendmailR

Many of the experiments conducted during my thesis work require long running simulations or numerous repetitions of some expensive calculation. Traditionally I started these on a cluster, waited and checked every once in a while if they were finished. This was both tedious and time consuming. Sometimes a calculation would abort because of an error and I would "waste" a day or more of time because I checked too infrequently if it was finished. Out of this need the sendmailR package was born. It can send e-mails from within R without any external utilities. I use it regularly to notify myself of errors or when a compute job has completed. It is also the package I receive the most feedback on.

[1] `http://git.p-value.net/p/ela.git`

5.4 soobench

The contemporary literature is filled with collections of single objective optimization problems. What was missing, at least for R, was a package that collected implementations of all these functions and their associated metadata such as the number of parameters, the global optima, its name and the original paper in which the function was proposed. In addition soobench contains functions that make it easy to conduct large scale experiments. These include

- functions to collect all evaluated parameter combinations
- functions to measure the first hitting time for a given list of target levels
- functions which transform the parameter space to generate variants of existing functions
- functions to perform an arbitrary optimization run with a fixed budget

Combined with the functions in the ela package, these functions and benchmark problem implementations were used to produce the results published in Mersmann et al. (2010b, 2011) and Bischl et al. (2012a).

While it is a nice feature that the soobench package makes it easier to perform large benchmark studies I also want to highlight another positive side effect. If a large fraction of the scientific community decides to standardize on the functions implemented in soobench results become comparable. Often times benchmark functions are chosen at random and implemented without any testing or verification. When one wants to compare results from different publications one often finds that the used functions have subtle differences. Even worse, some authors only use their three or four favourite functions instead of a large and diverse collection. I therefore hope that soobench or a similar package becomes a de facto standard in the scientific community. To a certain extent the BBOB[2] and NumBBO[3] projects, to which I have contributed, have achieved this for the comparison of black-box optimizers. The scope of soobench is broader. It also includes experiments such as those described in the papers from Chapter 2.

[2] `http://coco.gforge.inria.fr/doku.php?id=bbob-2013`
[3] `http://numbbo.gforge.inria.fr/doku.php`

5.5 microbenchmark

When optimizing R functions for large simulations studies it is not enough to use appropriate algorithms and data structures, the implementations must also be tuned. Often even small changes to the code can lead to drastic reductions in runtime. This is best illustrated by an example. Assume that `x` is a one million element numeric vector filled with (uniform) random numbers between 0 and 1 and we want extract all elements less than or equal 0.1. Two simple approaches are the R expressions `x[x < 0.1]` and `x[which(x < 0.1)]`. One would think that the first expression would be faster since it avoids the allocation of an additional result vector by the `which` function. Using microbenchmark I can easily check this

```
> library("microbenchmark")
> x <- runif(1000000)
> microbenchmark(x[x < 0.1], x[which(x < 0.1)], times=100L)
Unit: milliseconds
              expr     min      lq  median      uq      max neval
        x[x < 0.1] 76.1717 77.2544 78.3319 79.7721 106.3057   100
 x[which(x < 0.1)] 26.1653 27.1944 27.8998 29.0252  55.8956   100
```

Surprisingly the second expression is much faster. While it is possible to conduct similar timing experiments using standard R facilities[4], they lack precision. microbenchmark on the other hand tries hard to use the most precise time source provided by the respective operating system. This allows precise sub-millisecond timing and on some platforms even nanosecond accurate runtime measurements. While this may seem extreme, it is often much easier to benchmark small chunks of code with very short runtimes to understand its performance characteristics instead of running benchmarks on larger chunks of code.

microbenchmark played a crucial role in optimizing the implementations of the benchmark functions and helpers in the soobench package. It was also used to improve the performance of some functions in the ela and tspmeta packages.

[4]For example using the function `system.time` or the rbenchmark package which is based on this function.

5.6 BatchJobs and BatchExperiments

The computational experiments performed to obtain the results described in Chapters 2 to 4 consume anywhere from several weeks to more than a year of CPU time. Clearly these cannot be run on a local workstation or laptop. Instead they require large high performance computing (HPC) clusters. While HPC clusters have become a common tool and we are fortunate to have such a cluster in Dortmund, they are also complex to use. Instead of writing a single large program the experiment needs to be subdivided into many smaller tasks that can be run in parallel. These tasks are then submitted on the HPC system for execution. Some of them may fail for various reasons and need to be rescheduled. Once all jobs have terminated, the results need to be aggregated and analyzed. For all of these operations specialized knowledge of the software used on the HPC cluster is required.

To aid fellow researchers I developed a series of shell scripts that could submit, suspend, resume and terminate a series of R jobs on the LiDO cluster in Dortmund. Soon the limitations of these scripts became apparent. They were specific to our cluster in Dortmund and still required substantial modifications of the user's R code. At the same time my coauthors, Bernd Bischl and Michel Lang, decided to implement similar functionality to my shell scripts in the form of an R package. This resulted in the development of the package BatchJobs.

The core insight that lead to the design of BatchJobs is that almost all computational experiments in statistics can be modeled as a series of map and reduce steps. That is, a function is applied to every element of a list and the result is stored in a new list. This is the *map* step. Then another function reduces the result list to a final result value. This is the *reduce* step. More formally, given functions $m\colon I \to O$ and $a\colon R \times O \to R$ and an input set $\{i_1, \ldots, i_n\} \subset I$ we can apply m to each element of our input set to obtain the output set $\{o_1, \ldots, o_n\} \subset O$. This is the *map* step of the computation and can be performed in parallel by the cluster. To compute the final result, we recursively apply a to the output set. That is, we compute $r_i = a(r_{i-1}, o_i)$ for $i = 1$ to n and take r_n as our final result. Note that we need to specify an initial value r_0 for this computation. The recursive application of a to the output set is called the *reduce* step. It cannot be executed in parallel because in order to calculate r_i we must first calculate r_{i-1}. However, in most experiments the computationally expensive operations are performed in the map step. For

example the map step might train and test a particular parameterization of an SVM on a dataset and return the observed missclassification error. Here our input set would consist of the different parameter settings for the SVM and our output set would be the observed missclassification errors. In the reduce step we would want to find the minimum missclassification error so our function a would return the smaller of r_{i-1} and o_i and we would set $r_0 = \infty$. The vast majority of the computational burden lies in the map step where for each parameterization an SVM must be trained. The fact that we cannot parallelize the reduction step of calculating the minimum will have minimal effect on the total wall time of the calculation.

Another feature of the BatchJobs package is that it is not specific to one cluster setup or a single HPC software environment. Instead the user, or more likely the administrator, provides a series of simple templates that describe how jobs are submitted to the cluster manager and how they are removed. Using these templates, BatchJobs generates all the required files and folders to run the experiment, checks for jobs that terminated prematurely and provides error reports and diagnostic outputs. All of these facilities do not require the user to leave the R environment.

BatchExperiments builds on the infrastructure provided by BatchJobs. It provides utilities to design and analyze certain types of statistical experiments in a more natural form. Instead of defining an input set and mapping a function over it, a set of algorithms, algorithm settings, data sets and hyperparameters are defined and a (subset) of their Cartesian product is evaluated. That is each algorithm configuration is applied to each data set and the result stored. In the reduce step quality indicators or other measures are computed to asses the fitness of each algorithm configuration. Because my particular experiments do not quite fit the BatchExperiments regime I have not used the package outside of toy examples.

We published a detailed description of both packages in the Journal of Statistical Software (Bischl et al., 2015). Judging by the amount of feedback and the number of e-mails we receive the packages seem to have been very well received. Most of the heavy lifting when implementing both packages was performed by my coauthors Bernd Bischl and Michel Lang. I merely implemented some of the template functions in BatchJobs and consulted with them on design decisions given my prior experience implementing the shell script solution.

5.7 tspmeta

tspmeta is an R package that contains all domain specific functions used for the research presented in Chapter 3. Apart from the functions used to calculate all the features of a TSP instance it also contains a fast 2-opt implementation and the various normalization procedures used in Mersmann et al. (2012, 2013). For a detailed description of their purpose the reader is referred to the above chapter or the original publications. The intended purpose of the publication as a package instead of supplementary material is to enable other researchers to more easily extend our work and improve on the existing ideas.

Bibliography

Tinus Abell, Yuri Malitsky, and Kevin Tierney. Fitness landscape based features for exploiting black-box optimization problem structure. Technical report, Technical Report TR-2012-162, IT University of Copenhagen, 2012.

Anne Auger and Olivier Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 57(1):121–146, 2010.

Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Mike Preuß. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In Terence Soule and Jason H. Moore, editors, *GECCO*, pages 313–320. ACM, 2012a.

Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Claus Weihs. Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, 20(2):249–275, June 2012b.

Bernd Bischl, Michel Lang, Olaf Mersmann, Jörg Rahnenführer, and Claus Weihs. BatchJobs and BatchExperiments: Abstraction mechanisms for using R in batch environments. *Journal of Statistical Software*, 64:1–15, 2015.

Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Presentation of the Noiseless Functions, 2009. URL `http://coco.gforge.inria.fr/lib/exe/fetch.php?media=download3.6:bbobdocfunctions.pdf`.

Youssef Hamadi and Marc Schoenauer, editors. *Learning and Intelligent Optimization, 6th International Conference, LION 6, Paris, France, January 16-20, 2012. Selected Papers*, 2012. Springer.

Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noisy Functions Definitions. Technical Report RR-6869, INRIA, 2009.

Frank Hutter, Lin Xu, Holger H Hoos, and Kevin Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111, 2014.

Leonard Judt, Olaf Mersmann, and Boris Naujoks. Effect of SMS-EMOA parameterizations on hypervolume decreases. In Hamadi and Schoenauer (2012) (see above).

Leonard Judt, Olaf Mersmann, and Boris Naujoks. Non-monotonicity of observed hypervolume in 1-greedy s-metric selection. *Journal of Multi-Criteria Decision Analysis*, 2012b.

Katherine Malan and Andries Petrus Engelbrecht. Ruggedness, funnels and gradients in fitness landscapes and the effect on PSO performance. In *IEEE Congress on Evolutionary Computation*, pages 963–970. IEEE, 2013.

Olaf Mersmann, Mike Preuss, and Heike Trautmann. Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph, editors, *PPSN (1)*, volume 6238 of *Lecture Notes in Computer Science*, pages 73–82. Springer, 2010a.

Olaf Mersmann, Heike Trautmann, Boris Naujoks, and Claus Weihs. On the Distribution of EMOA Hypervolumes. In Christian Blum and Roberto Battiti, editors, *LION*, volume 6073 of *Lecture Notes in Computer Science*, pages 333–337. Springer, 2010b.

Olaf Mersmann, Heike Trautmann, Boris Naujoks, and Claus Weihs. Benchmarking evolutionary multiobjective optimization algorithms. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010c.

Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory Landscape Analysis. In Natalio Krasnogor and Pier Luca Lanzi, editors, *GECCO*, pages 829–836. ACM, 2011.

Olaf Mersmann, Bernd Bischl, Jakob Bossek, Heike Trautmann, Markus Wagner, and Frank Neumann. Local Search and the Traveling Salesman Problem: A Feature-Based Characterization of Problem Hardness. In Hamadi and Schoenauer (2012) (see above).

Olaf Mersmann, Bernd Bischl, Heike Trautmann, Markus Wagner, Jakob Bossek, and Frank Neumann. A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. *Annals of Mathematics and Artificial Intelligence*, pages 1–32, 2013.

Rachael Morgan and Marcus Gallagher. Length Scale for Characterising Continuous Optimization Problems. In Carlos A. Coello Coello, Vincenzo Cutello, Kalyanmoy Deb, Stephanie Forrest, Giuseppe Nicosia, and Mario Pavone, editors, *PPSN (1)*, volume 7491 of *Lecture Notes in Computer Science*, pages 407–416. Springer, 2012.

Mario A. Munoz, Michael Kirley, and Saman K. Halgamuge. The Algorithm Selection Problem on the Continuous Optimization Domain. In Christian Moewes and Andreas Nürnberger, editors, *Computational Intelligence in Intelligent Data Analysis*, volume 445 of *Studies in Computational Intelligence*, pages 75–89. Springer Berlin Heidelberg, 2013.

Samadhi Nallaperuma, Markus Wagner, and Frank Neumann. Ant colony optimisation and the traveling salesperson problem: hardness, features and parameter settings. In *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion*, pages 13–14. ACM, 2013.

Han-Hsing Tu and Hsuan-Tien Lin. One-sided support vector regression for multiclass cost-sensitive classification. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1095–1102, 2010.

Appendix A

Contributed Material

List of all contributed material to this dissertation:

- Olaf Mersmann, Mike Preuss, and Heike Trautmann. Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph, editors, *PPSN (1)*, volume 6238 of *Lecture Notes in Computer Science*, pages 73–82. Springer, 2010.

- Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory Landscape Analysis. In Natalio Krasnogor and Pier Luca Lanzi, editors, *GECCO*, pages 829–836. ACM, 2011.

- Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Mike Preuß. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In Terence Soule and Jason H. Moore, editors, *GECCO*, pages 313–320. ACM, 2012.

- Olaf Mersmann, Bernd Bischl, Jakob Bossek, Heike Trautmann, Markus Wagner, and Frank Neumann. Local Search and the Traveling Salesman Problem: A Feature-Based Characterization of Problem Hardness. In Youssef Hamadi and Marc Schoenauer, editors. *Learning and Intelligent Optimization, 6th International Conference, LION 6, Paris, France, January 16-20, 2012. Selected Papers*, Springer, 2012.

- Olaf Mersmann, Bernd Bischl, Heike Trautmann, Markus Wagner, Jakob Bossek, and Frank Neumann. A novel feature-based approach to char-

acterize algorithm performance for the traveling salesperson problem. *Annals of Mathematics and Artificial Intelligence*, pages 1–32, 2013.

- Olaf Mersmann, Heike Trautmann, Boris Naujoks, and Claus Weihs. On the Distribution of EMOA Hypervolumes. In Christian Blum and Roberto Battiti, editors, *LION*, volume 6073 of *Lecture Notes in Computer Science*, pages 333–337. Springer, 2010.

- Leonard Judt, Olaf Mersmann, and Boris Naujoks. Effect of SMS-EMOA parameterizations on hypervolume decreases. In Youssef Hamadi and Marc Schoenauer, editors. *Learning and Intelligent Optimization, 6th International Conference, LION 6, Paris, France, January 16-20, 2012. Selected Papers*, Springer, 2012.

- Leonard Judt, Olaf Mersmann, and Boris Naujoks. Non-monotonicity of observed hypervolume in 1-greedy s-metric selection. *Journal of Multi-Criteria Decision Analysis*, 2012.