Charles University in Prague

Faculty of Mathematics and Physics

DOCTORAL THESIS

Martin Pilát

Evolutionary Algorithms for Multiobjective Optimization

Institute of Computer Science, AS CR

Supervisor of the doctoral thesis:	Mgr. Roman Neruda, CSc.
Study programme:	Computer Science
Specialization:	4I1 – Theoretical Computer Science

Prague 2013

Essentially, all models are wrong, but some are useful.

— George E. P. Box (1987)

ACKNOWLEDGMENTS

I would like to thank to my supervisor, Roman Neruda, who introduced me to evolutionary algorithms and showed me how amazing this area is. He taught me how to do research and was with me during my first steps as a researcher. Without his willingness to work regardless of the time of day (or rather night) and regardless of the day of week(end) this thesis and most of our publications would be hardly possible.

A big thank you belongs also to the rest of our small team, Klára, Ondra, and Jakub. The inspiring meetings we had almost regularly made the work on the parts about meta-learning much easier and provided motivation to further improve the results. I am also very thankful to you for making most of the required changes to the Pikater system. Without them it would not be possible to make some of the experiments.

I am also grateful to Petra Novotná, who provided great help with all the administrative and financial matters and was able to answer all those strange questions I had during the years at the university. I would also like to thank to all the other people both at the Charles University and at the Institute of Computer Science who created a pleasant and inspiring working environment.

This thesis would also not be possible without my parents, grandparents, and my brother. They provided me both with support and motivation, and without them I would not even be able to study the university in the first place.

Finally, I would like to thank to my love, Petra, who supported me during all the years at the university and made my life happier. She patiently waited for me while I was on those numerous conferences and made me look forward to coming back home to see her and be with her. Thanks to her, I have always known there is someone to live for and someone to return to. Without her the world would not be the wonderful place it is.

ABSTRACT

TITLE:Evolutionary Algorithms for Multiobjective Optimization
AUTHOR: Mgr. Martin Pilát
DEPARTMENT: Institute of Computer Science, AS CR
SUPERVISOR: Mgr. Roman Neruda, CSc., Institute of Computer Science, AS CR
KEYWORDS: Multi-objective optimization, surrogate models, evolutionary algorithms, model selection, hyper-parameter tuning

Multi-objective evolutionary algorithms have gained a lot of attention in the recent years. They have proven to be among the best multi-objective optimizers and have been used in many industrial applications. However, their usability is hindered by the large number of evaluations of the objective functions they require. These can be expensive when solving practical tasks. In order to reduce the number of objective function evaluations, surrogate models can be used. These are a simple and fast approximations of the real objectives.

In this work we present the results of research made between the years 2009 and 2013. We present a multi-objective evolutionary algorithm with aggregate surrogate model, its newer version, which also uses a surrogate model for the pre-selection of individuals. In the next part we discuss the problem of selection of a particular type of model. We show which characteristics of the various models are important and desirable and provide a framework which combines surrogate modeling with meta-learning. Finally, in the last part, we apply multi-objective optimization to the problem of hyper-parameters tuning. We show that additional objectives can make finding of good parameters for classifiers faster.

ABSTRAKT

NÁZEV: Evoluční algoritmy pro vícekriteriální optimalizaci
AUTOR: Mgr. Martin Pilát
KATEDRA: Ústav informatiky AV ČR, v.v.i.
ŠKOLITEL: Mgr. Roman Neruda, CSc., Ústav informatiky AV ČR, v.v.i.
KLÍČOVÁ SLOVA: Vícekriteriální optimalizace, náhradní modely, evoluční algoritmy, výběr modelů, ladění hyper-parametrů

Vícekriteriální evoluční algoritmy se v posledních letech těší velké pozornosti. Dokázaly, že patří mezi nejlepší vícekriterální optimalizátory a byly použity v mnoha průmyslových aplikacích. Jejich použitelnost je ale omezována tím, že vyžadují velké množství vyhodnocení jednolivých účelových funkcí. Tyto mohou být v případě reálných problémů složité a jejich vyhodnocení může být drahé. Pro snížení počtu vyhodnocení jednotlivých účelových funkcí se používají tzv. náhradní modely. Ty jsou jednoduchou a rychlou aproximací skutečných účelových funkcí.

V této práci představujeme výsledky výzkumu prováděného mezi lety 2009 a 2013. Představujeme vícekriteriální evoluční algoritmus s agregovaným náhradním modelem a jeho verze, které používají další náhradní model pro předvýběr jedinců. V další části se zabýváme problémem výběru vhodného typu náhradního modelu. Diskutujeme o tom, které charakteristiky modelu jsou důležité a žádané, a navrhujeme propojení náhradního modelování s meta-učením. V poslední části se potom zabýváme využitím vícekriteriální optimalizace pro ladění parametrů klasifikátorů a ukazujeme, že přidání dalších účelových funkcí může urychlit nalezení vhodného nastavení.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

Balcar, Štěpán, Martin Pilát, and Roman Neruda (2012). "An evolutionary algorithm for 2D semi-guillotinable circular saw cutting." In: *IEEE Congress on Evolutionary Computation*. IEEE, pp. 1–5. ISBN: 978-1-4673-1510-4. DOI: 10.1109/CEC.2012.6256455.

Kazík, Ondřej, Klára Pešková, Martin Pilát, and Roman Neruda (2011a). "Implementation of Parameter Space Search for Meta Learning in a Data-Mining Multi-agent System." In: *ICMLA (2)*. Ed. by Xue-wen Chen, Tharam S. Dillon, Hisao Ishbuchi, Jian Pei, Haixun Wang, and M. Arif Wani. IEEE Computer Society, pp. 366–369. DOI: 10.1109/ICMLA.2011.161.

Kazík, Ondřej, Klára Pešková, Martin Pilát, and Roman Neruda (2011b). "Meta Learning in Multi-agent Systems for Data Mining." In: *IAT*. Ed. by Olivier Boissier, Jeffrey Bradshaw, Longbing Cao, Klaus Fischer, and Mohand-Said Hacid. IEEE Computer Society, pp. 433–434. ISBN: 978-0-7695-4513-4. DOI: 10.1109/WI-IAT.2011.233.

Kazík, Ondřej, Klára Pešková, Martin Pilát, and Roman Neruda (2012a). "A Novel Meta Learning System and Its Application to Optimization of Computing Agents' Results." In: *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on*. Vol. 2, pp. 170–174. DOI: 10.1109/WI-IAT.2012.250.

Kazík, Ondřej, Klára Pešková, Martin Pilát, and Roman Neruda (2012b). "Combining Parameter Space Search and Meta-learning for Data-Dependent Computational Agent Recommendation." In: *ICMLA (2)*. IEEE, pp. 36–41. ISBN: 978-1-4673-4651-1. DOI: 10.1109/ICMLA.2012.137.

Pilát, M. and R. Neruda (2012). "A Surrogate Based Multiobjective Evolution Strategy with Different Models for Local Search and Preselection." In: *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*. Vol. 1, pp. 215–222. DOI: 10.1109/ICTAI.2012.37.

Pilát, Martin (2010). "Evolutionary Multiobjective Optimization: A Short Survey of the State-of-the-art." In: *WDS 2010-Proceedings of Contributed Papers. Proceedings of the 19th Annual Conference of Doctoral Students, held 1-4 June 2010, in Prague.Part I-Mathematics and Computer Sciences Prague*. Ed. by Jana Safránková and Jirí Pavlû. Vol. 1. Matfyzpress, pp. 13–18. ISBN: 978-80-7378-139-2.

Pilát, Martin and Roman Neruda (2010). "Combining multiobjective and single-objective genetic algorithms in heterogeneous island model." In: *IEEE Congress on Evolutionary Computation*. IEEE, pp. 1–8. DOI: 10.1109/CEC.2010.5586075.

Pilát, Martin and Roman Neruda (2011a). "ASM-MOMA: Multiobjective memetic algorithm with aggregate surrogate model." In: *IEEE Congress on Evolutionary Computation*. IEEE, pp. 1202–1208. ISBN: 978-1-4244-7834-7. DOI: 10.1109/CEC.2011.5949753.

Pilát, Martin and Roman Neruda (2011b). "Improving many-objective optimizers with aggregate meta-models." In: *HIS*. Ed. by Ajith Abraham, Mohamed Kamel, and Ronald Yager. IEEE, pp. 555–560. ISBN: 978-1-4577-2151-9. DOI: 10.1109/HIS.2011.6122165.

Pilát, Martin and Roman Neruda (2011c). "LAMM-MMA: multiobjective memetic algorithm with local aggregate meta-model." In: *GECCO (Companion)*. Ed. by Natalio Krasnogor and Pier Luca Lanzi. ACM, pp. 79–80. ISBN: 978-1-4503-0690-4. DOI: 10.1145/2001858.2001905.

Pilát, Martin and Roman Neruda (2011d). "Local Meta-models for ASM-MOMA." In: *ICIC (1)*. Ed. by De-Shuang Huang, Yong Gan, Vitoantonio Bevilacqua, and Juan Carlos Figueroa García. Vol. 6838. Lecture Notes in Computer Science. Springer, pp. 147–152. ISBN: 978-3-642-24727-9. DOI: 10.1007/978-3-642-24728-6_20.

Pilát, Martin and Roman Neruda (2012a). "A surrogate multiobjective evolutionary strategy with local search and pre-selection." In: *GECCO (Companion)*. Ed. by Terence Soule and Jason H. Moore. ACM, pp. 633–634. ISBN: 978-1-4503-1178-6. DOI: 10.1145/2330784.2330895.

Pilát, Martin and Roman Neruda (2012b). "Aggregate meta-models for evolutionary multiobjective and many-objective optimization." In: *Neurocomputing*. In press. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2012.06.043.

Pilát, Martin and Roman Neruda (2012c). "An Evolutionary Strategy for Surrogate-Based Multiobjective Optimization." In: *IEEE Congress on Evolutionary Computation*. IEEE, pp. 1–7. ISBN: 978-1-4673-1510-4. DOI: 10.1109/CEC.2012.6256450.

Pilát, Martin and Roman Neruda (2012d). "Meta-learning and Model Selection in Multi-objective Evolutionary Algorithms." In: *11th International Conference on Machine Learning and Applications, ICMLA, Boca Raton, FL, USA, December 12-15, 2012. Volume 1*. IEEE, pp. 433–438. ISBN: 978-1-4673-4651-1. DOI: 10.1109/ICMLA.2012.78.

Pilát, Martin and Roman Neruda (2013a). "Multi-objectivization and Surrogate Modelling for Neural Network Hyper-parameters Tuning." In: *9th International Conference on Intelligent Computing, ICIC 2011, Nanning, China, July 28-31, 2013*. accepted. Springer.

Pilát, Martin and Roman Neruda (2013b). "Multiobjectivization for Classifier Parameter Tuning." In: *15th Annual Genetic and Evolution-*

ary Computation Conference, GECCO 2013, Companion Material Proceedings, Amsterdam, Neatherlands, July 6-10, 2013. accepted. ACM.

Pilát, Martin and Roman Neruda (2013c). “Surrogate Model Selection for Evolutionary Multiobjective Optimization.” In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2012, Cancun, Mexico, June 20-23, 2013*. accepted. IEEE.

CONTENTS

Part I

INTRODUCTION

This part contains the introduction to the problem of multi-objective optimization. The importance of the problem is discussed, as well as motivation for multi-objective optimization. The problem is defined in a formal manner. We also discuss the challenges one needs to face when comparing multi-objective optimizers and introduce some of the most often used benchmark problems.

1 INTRODUCTION

Optimization, the problem of finding the best possible solution to a given problem, is an important task. Most of the time optimization of only one objective is considered, although most problems can be more naturally defined in the framework of multi-objective optimization (i.e. optimization of more objective at once). For example, if a new product is designed by a company, one of the objectives would usually be the price, as most people would not buy a product which is extremely expensive just because it is slightly better according to some important criteria (e.g. a computer which is only slightly faster than a much cheaper one).

But multi-objective optimization (or rather multi-objective decision making) does not apply only to companies, in fact, most of the decisions we make in our lives are multi-objective. Even in the most common situations, like choosing how to get to work in the morning, we have several options, each of them with different properties, and some of them better according to one criteria, and the other better according to another one. We can, for example, go by car, which may be the fastest but also probably one of the more expensive options. Moreover, we have to park the car somewhere, which may be difficult. Another option may be to use public transportation, which may be slower, by also cheaper and easier to use. Every day in the morning, we have to make this multi-criteria decision and choose which mean of transport to take. To make things even more complicated, our preferences may change from time to time, and we may make a different decision every day. Therefore, it is important to have the whole set of solutions (in this case ways how to get to work) among which one can choose instead of having just the fastest one.

multi-objective example

And this is precisely the goal of multi-objective optimization – to provide a set of solutions, which are in a sense optimal in all of the objectives at once. The notion of optimality in this case comes from the notion of Pareto dominance i.e. one solution is better than another one, if it is better in all objectives. Although this notion is natural, it brings a problem, namely, there is a lot of solutions which are not comparable to each other. Thus, after a multi-objective optimizer returns a set of mutually incomparable solutions, the user has to decide which of them to use. The problem of choosing such a solution is called multi-objective decision making.

goal of multi-objective optimization

In this thesis, we focus on the multi-objective evolutionary algorithms, as in the last years they have proven to be among the best multi-objective optimizers. However, multi-objective evolutionary al-

problems of evolutionary algorithms

gorithms have one important disadvantage. They use a large number of objective function evaluations, and these evaluations can be expensive in practice. The expensiveness may come from two sources – either the function takes long to evaluate, or it is expensive in financial terms (e.g. because a real-world experiment has to be performed). It is not uncommon for evolutionary multi-objective optimizers to require tens of thousands of function evaluations to find a good set of solutions. And in such a case, even functions which take only a minute to evaluate may make the evolution unusable. This problem is even more pronounced when the function is expensive financially.

If the objective function is expensive "only" computationally, the situation can be improved to some extent by using the parallelization of the evaluations. However, a better approach is probably to reduce the required number of evaluations directly. To this end, so called surrogate models are often used. The surrogate model is an approximation of the real and expensive objective function. This approximation can be obtained in several ways, e.g. by using a simulation instead of running an experiment, or by using a less precise but faster simulation instead of a more precise one. However, one of the most common ways is to evaluate some points from the design space of the objectives precisely, and use these evaluated points to train a regression model. To this end, different types of models have been used, ranging from simple polynomial regression ones, to neural networks and Gaussian processes.

surrogate models

In this thesis, we study new surrogate models for multi-objective optimization. The important feature of the model we use is that it predicts only one value which expresses the overall quality of each individual instead of predicting the value for each of the objectives separately– it is a so called aggregated surrogate model. This makes the model suitable for exploitation by a much simpler single-objective evolutionary algorithm.

1.1 THE OUTLINE OF THE THESIS

introduction

The whole thesis is divided into four parts. The first part contains a general introduction, and the definition of multi-objective optimization. We also discuss the problems multiple objectives bring when two optimizers should be compared, and mention the most often used benchmark problems.

state of the art

The second part of the thesis contains three chapters which describe the current state of the art. Chapter 3 describes the models which are most often used in the field of surrogate-based multi-objective optimization, and which we also use in some of the experiments. In Chapter 2 we describe some of the most important state-of-the-art multi-objective evolutionary algorithms. Chapter 5 then describes the approaches others have taken to tackle the problem of surrogate modeling in evolutionary multi-objective optimization. All the chapters mention mostly the most important algorithms with respect to this thesis, they are not meant to be a complete survey of the state of the art. However, references to more complete surveys and overviews are given where appropriate.

The third part is the most important one and contains the main contribution of the thesis. In Chapter 6 we describe the new distance-based aggregate surrogate model and the way it can be used to augment the existing multi-objective evolutionary algorithms. The model is based on the distance of individuals from the current Pareto set and is exploited by a single-objective evolutionary algorithm. We try different types of models to predict the distance and show which of them work better. The algorithm is later augmented by the addition of pre-selection which decides whether a given individual is promising and should thus be evaluated using the real objective, or whether the individual should be rather dropped. We show that the pre-selection step should use a different model than the local search step in order to be able to provide more than one individual in each generation. This is important in practice as often more individuals can be evaluated at once with lower cost than if they were evaluated one by one. The pre-selection is described in Chapter 7. Throughout these two chapters we present the results of several preliminary experiments which show the performance of various versions of the algorithms on a limited set of benchmark functions. At the end of Chapter 7, we compare some of the versions on a much larger and challenging set of benchmark functions to show the performance under more realistic conditions.

new surrogate model

pre-selection

extensive comparison

Chapters 8 and 9 contain preliminary results which should lead to the integration of meta-learning and surrogate modeling in the future and especially Chapter 2 is focused more on an application of multi-objective optimization, rather than the study of new algorithms. Chapter 8 studies the types of model which are useful in evolutionary optimization. It turns out that models which have low mean squared error are not the best ones to use with an evolutionary algorithm which uses only comparisons between individuals during their run. Models which preserve the relations among the individuals better tend to provide better overall results. This is an important observation as it provides guide for the selection of better models, if meta-learning is used with surrogate modeling. In Chapter 9 we study the integration of multi-objective optimization and meta-learning from a different point of view. We apply multi-objective optimization to the problem of setting the hyper-parameters of a classifier in order to minimize its classification error. We show that additional objectives in this case can improve the results, even though the problem is essentially single-objective. The additional objectives help the algorithm to proceed to interesting parts of the search space as they provide di-

model selection

parameter tuning

rection in the areas of search space with constant classification error. These cases are quite common as the classification error is a discrete variable. Moreover, classifiers are usually robust in the sense than a small change of their hyper-parameters does not affect the final classification error, which makes the error function even more complicated.

Finally, the fourth part summarizes the main contributions of the thesis and provides some ideas for a future research.

2 MULTI-OBJECTIVE OPTIMIZATION

As we have already mentioned, multi-objective optimization provides framework for optimizing multiple objectives at once. Having multiple objectives brings new challenges to the field of optimization and even relatively simple concepts, like the comparison of two solutions and the decision which of them is better, gets more complicated in the multi-objective case. In this chapter, we define the most important concepts in multi-objective optimization and discuss some of the new challenges.

We shall start our review be the definition of multi-objective optimization problem itself.

multi-objective optimization problem

Definition 2.1. A *multi-objective optimization problem* is defined as a tuple $\langle D, O, F, C\rangle$, where D is the design (decision) space, $O \subseteq \mathbb{R}^n$ is the objective space, $F = \langle f_1, \dots, f_n\rangle$ with $f_i : D \to \mathbb{R}$ is the set of n objective functions, and $C = \{c_1, \dots, c_l\}$ is the set of l constraints.

In the case of continuous multi-objective optimization the design space is a subset of $\mathbb{R}^d$, most often a d-dimensional interval $[l_1, u_1] \times [l_2, u_2] \times \cdots \times [l_d, u_d]$, where l_i and u_i define the lower and upper bounds for each of the variables respectively. In a more general case, some of the variables may have discrete domains. Moreover, in case of combinatorial optimization, when e.g. the traveling salesman problem is solved, D may be the set of all permutations over a given set. Also, the objective space may in theory be more general, however, such situation is uncommon in practice.

In this thesis, we will only consider continuous optimization of multiple objectives $f_i : \mathbb{R}^n \to \mathbb{R}$ without constraints. We also assume without any loss of generality only minimization of all the objectives here.

Having multiple objectives causes several complications compared to single-objective optimization. One of them is the absence of any linear ordering of the solutions, as it is not obvious which of two solutions x_1 and x_2 is better in a case, where $f_1(x_1)$ has a better value than $f_1(x_2)$, but $f_2(x_2)$ is better than $f_2(x_1)$. The comparison of individuals in the multi-objective case is defined by the so called *Pareto dominance*. Unfortunately, Pareto dominance does not define complete order on the set of all individuals.

Pareto dominance

Definition 2.2. *Individual x dominates individual y* ($x \prec y$) (equivalently, *individual y is dominated by individual x*), if for each objective f_i, $f_i(x) \leq f_i(y)$, and there is at least one objective f_j for which $f_j(x) \neq f_j(y)$.

If neither $x \prec y$ nor $y \prec x$, we say that *x and y are (mutually) non-dominated*.

Having the definition of Pareto dominance, we can proceed with the definition of the goal of multi-objective optimization.

goal of multi-objective optimization

Definition 2.3. The *solution of a multi-objective optimization problem* $\langle D, O, F, C \rangle$ is a *Pareto set* $P \subset D$, such as for each $x \in D$ and $y \in P$, the individual y is not dominated by the individual x. The image of the Pareto set P under the objectives F is a subset of O called the *Pareto front*.

Pareto set

Thus, the Pareto set is the set of minimal points of the Pareto dominance relation in the decision space. In practice, this set is usually uncountable for continuous multi-objective optimization problems and thus no algorithm can provide the complete Pareto set. We will call the finite sets of non-dominated points returned by the multi-objective optimizers Pareto set approximations.

Definition 2.4. A *Pareto set approximation* $A \subset D$ is a finite set of points in the decision space such that for each two points $x, y \in A$, x and y are mutually non-dominated.

2.1 COMPARISON OF MULTI-OBJECTIVE OPTIMIZERS

good Pareto sets

The fact that the Pareto dominance relation is not a complete ordering of course translates to the absence of a natural complete ordering of the Pareto approximations. This is an important problem in practice, when new algorithms are designed and should be compared to other existing algorithms. However, there are at least two important features which are important for a Pareto set approximation to be considered "good". One of them is the spread of solutions – the solutions in the approximation should be evenly spread along the true Pareto set. The other is the convergence of the solutions – the solutions should be close to the true Pareto set[1].

indicators

Various indicator have been proposed which express these two features of Pareto set approximations numerically. Most of these indicators are binary and compare the quality of two Pareto set approximations. If these indicators are used in practice, one of the approximations may be the union of the approximations obtained by the compared algorithm, which makes the indicator in fact unary and the direct comparison of various algorithms easier

generational distance

The *generational distance* (Van Veldhuizen and Lamont, 1998) indicator measures the average distance of the points in the approximation to the true Pareto front. This metric should be minimized.

Definition 2.5. Let A be a Pareto set approximation and P a set of uniformly distributed points from the Pareto set. Let $\delta(x, y)$ be the

1 An attentive reader may notice we have just defined a multi-objective optimization problem of finding a good Pareto set approximation.

Euclidean distance of points x and y in the objective space. The *generational distance* metric is defined as

$$\mathrm{GD}(A, P) = \frac{1}{|A|} \sum_{x \in A} \min_{y \in P} \delta(x, y).$$

The main disadvantage of this metric is that it measures only the convergence of the Pareto set approximation and it completely ignores the spread of the solutions. If there are only a few points in the Pareto set approximation, which are additionally close to some points in P the generational distance may be quite low, regardless of the fact that some of the parts of the Pareto set are not covered.

This problem is partially solved by the use of *inverse generational distance* (Sato et al., 2004) which basically swaps the sets A and P in the definition.

inverse generational distance

Definition 2.6. Let A be a Pareto set approximation and P a set of uniformly distributed points from the Pareto set. Let $\delta(x, y)$ be the Euclidean distance of points x and y in the objective space. The *inverse generational distance* metric is defined as

$$\mathrm{IGD}(A, P) = \frac{1}{|P|} \sum_{x \in P} \min_{y \in A} \delta(x, y).$$

In this case, if some points of the set P are not covered by the Pareto set approximation, the approximation is penalized. However, using only the information provided by the IGD metric it is not obvious, whether a Pareto set optimization is worse than another one because of worse spread or because of poor converge. Thus, both GD and IGD are used at the same time in order to provide information both on spread and convergence.

One of the most often used indicators is the *hypervolume indicator* (also knows as the $\mathcal{S}$-metric). The hypervolume indicator expresses the hypervolume of the space dominated by a given Pareto set approximation (the space is bounded by a reference point from above).

hypervolume

Definition 2.7. Let A be a Pareto set approximation and $r \in O$ be a reference point. The hypervolume indicator

$$\mathrm{H}(A, r) = \int_O \mathbb{1}_{\{x \in O \, \exists a \in A \, a \preceq x \preceq r\}}(z) \, dz,$$

where $\mathbb{1}_X$ is the characteristic function of the set X.

The hypervolume indicator has some features which make it particularly popular in the literature. It is a unary operator, which makes the comparisons of two sets easier than when binary operators are used. It also expresses both the convergence and the spread of the approximation set A. It is one of the few indicators which are Pareto compliant (Zitzler, Brockhoff, et al., 2007), i.e. if a set A dominates set

B, than $\mathrm{H}(A, r) \geq \mathrm{H}(B, r)$. Thus, finding a set which maximizes the hypervolume implies that the set contains only Pareto optimal points and cannot be dominated by any other set.

optimal μ-distributions

An interesting feature of the hypervolume indicator was discovered while the optimal μ-distributions were studied (i.e. sets of μ points which have the optimal hypervolume). It turns out that the hypervolume indicator is biased towards the points on the Pareto front whose derivation is close to -1 (Auger, Bader, Brockhoff, and Zitzler, 2009). Thus, the density of points in these areas of the Pareto front is higher than in different areas. This may or may not be a good feature, depending on the application. Generally, such points are considered interesting, as they provide fair trade-off among the objectives. If such a feature is not desired, weighted hypervolume may be used (Zitzler, Brockhoff, et al., 2007). It is derived from the above defined formula by adding a weighting multiplicative term $w(z)$ inside the integral. Such weighting can change the bias of the hypervolume indicator taking into account the decision maker's preferences. More recently, the optimal μ-distributions were also studied in the three-objective case (Auger, Bader, and Brockhoff, 2010).

*#**P**-hardness*

However, hypervolume indicator also has a serious disadvantage – it cannot be computed effectively. In fact, its computation is known to be #**P**-hard (Bringmann and Friedrich, 2010) and thus no polynomial algorithm exists unless $\mathbf{P} = \mathbf{NP}$. The fastest algorithms for the computation of hypervolume have the worst case complexity $\mathcal{O}(n^{N/2})$ (Yang and Ding, 2007), $\mathcal{O}(N^{n-2} \log N)$ (C. Fonseca et al., 2006), or $\mathcal{O}(N \log N + N^{n/2})$ (Beume, 2009) (in all cases n is the number of objectives and N is the size of the set). To overcome this difficulty, Monte Carlo sampling may be used to approximate the value of the indicator for problems with higher numbers of objectives (Bader and Zitzler, 2011).

attainment function

An alternative approach to the use of indicators is the use of so called *empirical attainment functions* (C. M. Fonseca and Fleming, 1996). The attainment function for a set of Pareto approximation sets (i.e. the results of several runs of the evolutionary algorithm) expresses the probability that a given point in the objective space is dominated by the resulting Pareto set approximation.

Definition 2.8. Let $X_1, \ldots, X_p \subseteq O$ be p Pareto set approximations. The *empirical attainment function* $\alpha : O \to [0, 1]$ is defined as

$$\alpha(x) = \alpha(X_1, \ldots, X_p, x) = \frac{1}{p} \sum \mathrm{I}(X_1 \preceq x),$$

where $\mathrm{I}(b)$ for a condition b is the indicator function

$$\mathrm{I}(b) = \begin{cases} 1 & \text{if } b \text{ is true} \\ 0 & \text{otherwise.} \end{cases}$$

The empirical attainment functions are used mostly for visualization of performance in the bi-objective case, although they can also be used for a more detailed statistical analysis of the results (C. M. Fonseca, Guerreiro, et al., 2011). In the case of visualization, the p%-*attainment surfaces* are particularly interesting. The p% attainment surface is the boundary between the part of the objective space which are attained in more than p% of the runs of the optimizer and the rest of the objective space. Thus, the attainment surfaces can be seen as generalization of different percentiles to the multi-objective case.

attainment surface

2.2 BENCHMARK PROBLEMS

The need to compare various optimizers naturally leads to the need to define benchmark problems. In the last decades, there were several benchmark suites proposed and designed. Among the most well-known and most often used is the ZDT benchmark set (Zitzler, Deb, et al., 2000) (named after its authors, as is also the case with the other benchmarks mentioned here). The set contains six bi-objective problems which are scalable in the number of variables. ZDT5 is often ignored in comparisons, as it is defined for binary strings rather than real vectors. Although the ZDT benchmark is now considered rather simple, it is still used as the only benchmark in most of the publications. The simplicity of the benchmark set comes from the fact that most of the objectives are separable – which is a feature some of the multi-objective optimizers are able to exploit. Moreover, the Pareto optimal sets of the problems lay on the boundary of the search space and the first objective of most of the functions (except ZDT6) is linear.

ZDT benchmark

Another (very related) benchmark suite is called IHR (Igel et al., 2007). This test suite contains rotated variants of the ZDT problems. The rotation makes the problems harder, as they are no longer separable and also the Pareto optimal set does not lay on the boundary of the search space. It was designed to show the advantages of algorithms based on covariance matrix adaptation and differential evolution, which are invariant to the rotations of the search space (among other properties).

IHR benchmark

DTLZ benchmark set (Deb, L. Thiele, et al., 2002) is one of two benchmark suites which are scalable both in the number of variables and the number of objectives. This makes it extremely useful for the comparison of many-objective optimizers (with "many" meaning more than 4 in this case). It is also sometimes used to show the performance of multi-objective optimizers in the three-objective case.

DTLZ benchmark

Newer benchmark sets, like LZ09 (Li and Qingfu Zhang, 2009) and WFG (Huband et al., 2006) aim at the creation of more complicated problems. The LZ09 problems have complicated Pareto sets, while the WFG toolkit was designed to provide functions, which have wide

LZ09 and WFG benchmarks

variety of properties – multi-modality, non-separability, deceptiveness. The functions also have dissimilar domains and do not have Pareto optimal points on the boundary of the search space.

In this thesis, we use mostly the ZDT benchmark set. The many-objective experiments are run on the DTLZ benchmark. Some of the results were also validated on the IHR and WFG benchmark sets.

Part II

STATE OF THE ART

In this part, we first describe the motivation for multi-objective optimization. Later, we formally define the problem of multi-objective optimization. The description of some of the state-of-the-art multi-objective algorithms follows. Then, we describe the motivation for the use of surrogate models and also provide some insights to the current state of the art in this field.

3 MODELS

This chapter contains the description of the modeling techniques, we have used during the research of surrogate-based multi-objective optimization. We use these models mostly as black boxes, which after being presented with a training set, create its model. Therefore, we describe most of these models only briefly, with the emphasis on the structure of the model and its training. We pay more attention only to the support vector machines, which are in different versions used by some of the competing surrogate-based evolutionary algorithms.

3.1 LINEAR REGRESSION

Linear Regression (LR) Draper et al. (1966) is one the simplest and oldest regression models. It assumes linear dependence between the input variables and the output variable. For training set

$$T = \{(x_i, y_i), x_i \in \mathbb{R}^n, y_i \in \mathbb{R}\},$$

the goal is to find a linear function $f(x) = \langle w, x\rangle + b$ such, that the sum of the squared residual errors $\sum_{i=1}^{N}(y_i - f(x_i))^2$ is minimized. The bias b is often removed by adding a 1 in front each of the input vectors x_i (thus, a vector $\langle x_{i1}, \dots, x_{in}\rangle$ becomes $\langle 1, x_{i1}, \dots, x_{in}\rangle$. We will denote the added 1 as x_{i0} and b will be denoted as w_0. Using this trick, the function $f(x)$ becomes $f(x) = \langle w, x\rangle = \sum_{i=0}^{n} w_i x_i$.

The input vectors x_i are often written in a form of matrix

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \dots & x_{Nn} \end{pmatrix}$$

the parameters w_i as a column vector $w = \langle w_0, w_1, \dots, w_n\rangle^\top$. And the outputs as a column vector $y = \langle y_1, \dots, y_n\rangle^\top$. Thus, the problem becomes to minimize the value of $E(w) = (y - Xw)^\top (y - Xw) = \sum_{j=1}^{N}(y_j - \sum_{i=0}^{n} w_i x_{ij})^2$. To find the minimum, one takes the derivative of the error function and sets it equal to zero. Thus

error function

$$\frac{\partial E}{\partial w_k} = -2\sum_{j=1}^{N}(y_j - \sum_{i=0}^{n} w_i x_{ji}) x_{jk} = 0,$$

which in turn leads to the system of equations

system of equations

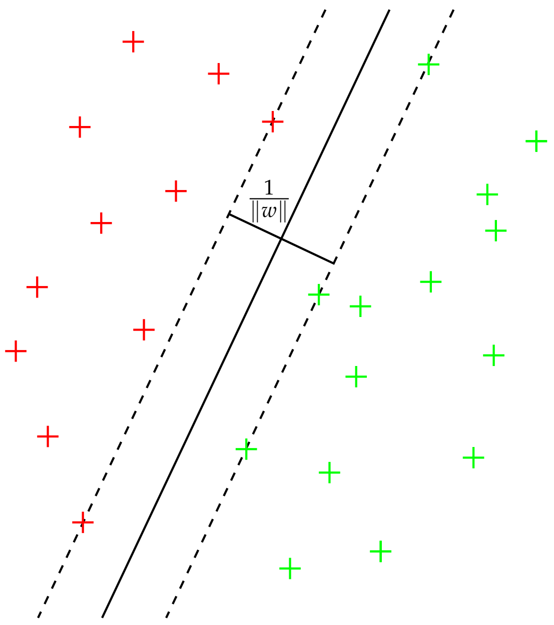


Figure 3.1: The maximum margin classifier, which separates the red and green crosses. The solid line represents the separating hyperplane, the two dashed lines represent the margin. The crosses, which lay on the dashed lines are the support vectors and completely define the separating hyperplane.

$$\sum_{j=1}^{N}\sum_{i=0}^{n} w_i x_{ji} x_{jk} = \sum_{j=1}^{N} y_j x_{jk},$$

for each k.

The system of equations can be expressed in the matrix form as

$$X^\top X w = X^\top y,$$

which leads to the expression for the vector of weights w

$$w = (X^\top X)^{-1} X^\top y.$$

features

An important observation regarding linear regression is that the model is linear in the weights w. However, the *features* x do not need to be linear. Therefore, the vectors x can be mapped into a feature space before the regression is made and the approach described above can thus be generalized to non-linear regression.

3.2 SUPPORT VECTOR MACHINES

Support Vector Machines (SVM) (Cortes and Vapnik, 1995) were originally created to deal with the problem of binary classification. In this section the *maximum margin classifier* is first derived in detail and later, the *kernel trick* is presented which allows for the classification of linearly non-separable data.

Given the finite training set

maximum margin classifier

$$T = \{(x_i, y_i) | x_i \in \mathbb{R}^n, y_i \in \{-1, 1\}\}$$

the goal of the maximum margin classifier is to find a separating hyperplane $\langle w, x \rangle + b$ such that

$$\begin{aligned} \langle w, x_i \rangle + b &\geq 1 \qquad y_i = 1 \\ \langle w, x_i \rangle + b &\leq -1 \qquad y_i = -1 \end{aligned}$$

These two constraints can be equivalently rewritten as a single constraint $y_i(\langle w, x_i \rangle + b) - 1 \geq 0$.

Moreover, the margin (i.e. the distance to the closest point on either sides of the separating hyperplane) should be maximized. The margin equals to $\frac{1}{\|w\|}$. Maximization of this quantity is equivalent to minimization of $\frac{1}{2}\|w\|^2$. Thus the problem of finding the normal vector of the separating hyperplane can be formulated as an optimization problem

primal problem definition

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(\langle w, x_i \rangle + b) - 1 \geq 0. \end{aligned}$$

In order to solve its problem, its Lagrangian dual form is constructed. The Lagrangian of the problem is

$$L(w, b, \mu) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \mu_i [(y_i \langle w, x_i \rangle + b) - 1],$$

where μ_i are the Lagrange multipliers. The objective function of the dual problem is obtained by minimization of the Lagrangian with respect to w and b.

To this end, we use the Karush-Kuhn-Tucker conditions (Karush, 1939; Kuhn and Tucker, 1951) which imply that for the optimal vector of the parameters (w^*, b^*) and for $\mu_i \geq 0$

Karush-Kuhn-Tucker conditions

$$\nabla L = \nabla \left(\frac{1}{2}\|w^*\|^2 \right) - \sum_{i=1}^{N} \mu_i \nabla \left(y_i \left(\langle w^*, x_i \rangle + b^* \right) - 1 \right) = 0.$$

Thus,

$$\begin{aligned} \frac{\partial L}{\partial w} &= w - \sum_{i=1}^{N} \mu_i y_i x_i = 0 \\ \frac{\partial L}{\partial b} &= -\sum_{i=1}^{N} \mu_i y_i = 0, \end{aligned}$$

which implies

$$\begin{aligned} w^* &= \sum_{i=1}^{N} \mu_i y_i x_i \\ \sum_{i=1}^{N} \mu_i y_i &= 0 \end{aligned}$$

and after substituting back to the Lagrangian, we get the objective function of the dual optimization problem

$$\theta(\mu) = -\frac{1}{2}\sum_{i,j=1}^{N} \mu_i \mu_j y_i y_j \langle x_i, x_j \rangle + \sum_{i=1}^{N} \mu_i$$

dual problem formulation

It can be further simplified by defining the matrix $H_{ij} = y_i y_j \langle x_i, x_j \rangle$, which yields the following formulation of the dual problem:

$$\begin{aligned} \max_{\mu} \quad & \sum_{i=1}^{N} \mu_i - \frac{1}{2}\mu^\top H \mu \\ \text{subject to} \quad & \mu_i \geq 0 \\ & \sum_{i=1}^{N} \mu_i y_i = 0. \end{aligned}$$

This is a quadratic optimization problem and can be solved by the means of quadratic programming. A popular choice is the Sequential Minimal Optimization algorithm (Platt, 1998), which is specifically designed to deal with the kind of optimization problems which arises during the training of SVMs.

calculation of the bias b

It remains to set the value of b which is not set by the optimization problem defined above as it was completely eliminated during the dual construction. To this end, we take one of the conditions for which the respective $\mu_i \neq 0$ (these are the so called active conditions, and these also define the support vectors – the vectors closest to the separating hyperplane). For this condition, it holds that $y_i(\langle w, x_i \rangle + b) = 1$, and thus the b can be expressed as

$$b = y_i - \langle w, x_i \rangle.$$

An alternative approach is to set the b as the average of the above expression over all the support vectors.

The main disadvantage of the maximum margin classifier is that it only works for linearly separable data. If the data are not linearly separable, there is no feasible solution to the problem. To overcome this difficulty, slack variables $\xi_i \geq 0$ are used to relax the conditions and express how much the conditions are violated. Of course, the violation is also minimized. This leads to the so called *soft margin classifier* .

soft margin classifier

Thus, the primal optimization problem becomes

$$\begin{aligned} \min_{w,b,\xi_i} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & y_i(\langle w, x_i \rangle + b) - 1 + \xi_i \geq 0. \end{aligned}$$

and the corresponding dual optimization problem (derived by similar steps as for the maximum margin classifier) is then

$$\begin{aligned} \max_{\mu} \quad & \sum_{i=1}^{N} \mu_i - \frac{1}{2}\mu^\top H \mu \\ \text{subject to} \quad & 0 \le \mu_i \le C \\ & \sum_{i=1}^{N} \mu_i y_i = 0. \end{aligned}$$

Note that the slack variables disappear in the dual form, and only the parameter C remains as an additional constraint of the Lagrange multipliers. The bias b is computed in the same way as for the maximum margin classifier.

Although the soft margin classifier works for linearly non-separable data, it is still only a linear classifier, and in lots of cases linear classification is too simple to grasp the complexity of the data. This problem may be solved by mapping the input space to a higher dimensional feature space through a mapping $\phi(x) : \mathbb{R}^n \to \mathbb{R}^m$, for some $m > n$, in which the original data would be linearly separable. Then, the maximum margin classifier may be used in the feature space. The disadvantage of this approach is the increased computational complexity which comes from the increased dimension.

feature space

In order to overcome this complexity, the so called *kernel trick* is used. A kernel is a positive semi-definite function $k(x, y) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, such that $k(x, y) = \langle \phi(x), \phi(y) \rangle$. Thus, the kernel computes the inner product of the two vectors in the feature space using only its values in the input space. In practice, the mapping ϕ does not have to be specified explicitly, and only the kernel is needed in order to be able to use the support vector machine. Note that the dual optimization problem for the maximum margin classifier depends only on the inner products of the vectors in the training set. Thus, the model may be generalized for non-linear classification by just replacing the dot products in the matrix H by a suitable kernel.

kernel trick

One of the popular choices for the kernel is the so called *polynomial kernel* $k(x, y) = (\langle x, y \rangle + b)^d$, where d is a positive integer. For example, for $d = 2$ and an n-dimensional input vector x, this kernel computes the dot products in a $\binom{n+2}{2}$-dimensional feature space given by the (implicitly defined) mapping

polynomial kernel

$$\begin{aligned} &\phi(\langle x_1, \ldots, x_n \rangle) = \\ &\quad \langle x_1^2, \ldots, x_n^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \ldots, \sqrt{2}x_{n-1}x_n, \sqrt{2c}x_1, \ldots, \sqrt{2c}x_n, c \rangle. \end{aligned}$$

Another popular choice of the kernel is the so called *Gaussian radial basis function*

Gaussian kernel

$$k(x, y) = e^{-\gamma \|x-y\|^2}.$$

This kernel maps the vectors into an infinite dimensional Hilbert space.

Although originally developed for classification, variants of support vector machines have been developed to deal with regression (Drucker et al., 1996). The ideas used in Support Vector Regression (SVR) are similar to those for classification. The goal is to find a linear model of the data which for each instance would have error less then ϵ. Again, slack variables are added to deal with data which cannot be approximated with such a precision. In this case, the variables are ξ_i^- and $\xi_i^+$for lower and upper bound respectively. The primal form of the optimization problem is

support vector regression

$$\begin{aligned} \min_{w,b,\xi_i^+,\xi_i^-} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i^- + \xi_i^+) \\ \text{subject to} \quad & \langle w, x_i\rangle + b - y_i \leq \epsilon + \xi_i^- \\ & y_i - \langle w, x_i\rangle + b \leq \epsilon + \xi_i^+ \\ & \xi_i^+, \xi_i^- \geq 0. \end{aligned}$$

Using again similar steps as above, we get the dual optimization problem

$$\begin{aligned} \max_{\mu^+,\mu^-} \quad & \sum_{i=1}^{N} y_i(\mu_i^- - \mu_i^+) - \epsilon\sum_{i=1}^{N}(\mu_i^- - \mu_i^+) \\ & -\frac{1}{2}(\mu^- - \mu^+)K(\mu^- - \mu^+) \\ \text{subject to} \quad & 0 \leq \mu_i^+, \mu_i^- \leq C \\ & \sum_{i=1}^{N}(\mu_i^- - \mu_i^+) = 0, \end{aligned}$$

where the matrix $K_{ij} = \langle x_i, x_j\rangle$ is the matrix of dot products between the pairs of input vectors.

The dual problem is again a quadratic optimization problem and can be solved by quadratic optimization techniques. Note, that the dual problem is again expressed only using the inner products of the input vectors, and thus the kernel trick may be used to extend the ideas of support vector regression to non-linear regression. It suffices to redefine the matrix K as $K_{ij} = k(x_i x_j)$.

3.3 MULTILAYER PERCEPTRON NETWORKS

Multilayer perceptron (MLP) networks (Haykin, 1999) are among the most popular types of artificial neural networks (ANN). An MLP consists of several layers of units – the *input layer*, the *output layer* and one or more *hidden layers*.

The task of the input layer is to feed the inputs to the first hidden layer. Thus, the units in the input layer perform an identity function and are connected to all the units in the first hidden layer.

In the hidden layer, each of the units makes a weighted sum of its inputs, and transforms the result through an *activation function* f with a bias b. Thus, the function performed by the unit is $f(b + \langle w, x \rangle)$, where n is the number of units in the previous (hidden or input) layer and w are the weights of these inputs. Similarly to the approach used in linear regression, the bias is often removed by adding and constant input $x_0 = 1$ to all the layer in the network. This simplifies the function performed by each of the units to $f(\langle w, x \rangle)$. Most commonly, the activation function in the hidden layer is a sigmoid function, usually, the logistic sigmoid

activation function

$$f(x) = \frac{1}{1 - e^{-\lambda x}}.$$

The outputs of each unit in the hidden layer are fed to the inputs of the units in the next layer – either hidden or output.

The units in the output layer work similarly to the units in the hidden layer. The only difference is that they often use a linear activation function instead of the sigmoid function, especially if the neural network is used for regression.

The weights of the MLP are traditionally adapted using the *back-propagation algorithm*[1]. Is is a gradient descend algorithm, thus it changes the weights of the network by subtracting a quantity proportional to the derivation of the error function by the particular weight. Thus, the weight after $k+1$ iterations of the algorithm is

back-propagation

$$w^{k+1} = w^k - \alpha \frac{\partial E}{\partial w^k},$$

for each weight in the MLP network, where w^k denotes the value of the weight w after k iterations of the back-propagation algorithm.

Usually, the error function E, is the mean squared error, the same as was optimized in the case of linear regression. However, in the case of MLP, the network can have more than one output. Thus, the function for a single vector v from the training set becomes $E_v = \frac{1}{2}\sum_{j=1}^{n}(y_j - y_j^*)^2$, where y_j is the actual output of the network and y_j^* is the desired output of the network. The error for the whole training set is then the sum over all vectors in the training set. In some cases additional terms are added to the function, especially if regularization of the neural network is performed.

error function

Let us compute the partial derivation of the error function E_v according to the weights between the last hidden layer and the output layer. Let w_{ij} denote the weight between the i-th neuron of the last hidden layer and the j-th neuron of the output layer, and x_i the output of the i-th neuron in the last hidden layer. Let $\xi_j = \sum_{i=0}^{l} w_{ij}x_i$, where l is the number of neurons in the last hidden layer, be the excitation of the j-th neuron in the output layer. The output of the network is then

adaptation of output layer

1 Nowadays, more advanced gradient algorithms are often used, e.g. the Levenberg-Marquardt algorithm (Levenberg, 1944; Marquardt, 1963).

$y_j = f(\xi_j)$, The partial derivation of the error E_v according to weight w_{ij} is

$$\frac{\partial E_v}{\partial w_{ij}} = \frac{\partial E_v}{\partial y_j}\frac{\partial y_j}{\partial \xi_j}\frac{\partial \xi_j}{\partial w_{ij}} = (y_j - y_j^*)x_i\frac{\partial f(\xi j)}{\partial \xi_j}.$$

Where the last term is the derivation of the activation function and is equal to 1 for the identity activation function, and to $\lambda y_j(1-y_j)$ for the logistic sigmoid function.

For reason which will become apparent soon, we define δ_j as the partial derivative

$$\delta_j = \frac{\partial E_v}{\partial \xi_j}.$$

adaptation in the hidden layer

For the adaptation of the weight between hidden layers (or input and hidden layer), the situation is a bit more complicated. We will derive the adaptation rule for the weights between the two last hidden layers and show that the ideas are also valid for the adaptation of weights between any two hidden layers. In the following, we will use the index$k$ to denote the neurons in the output layer, index j to denote the neurons in the last hidden layer and index i to denote the neurons in the layer before the last hidden layer. The inputs of the neurons will be denoted as x_k, x_j, and x_i respectively. The excitations of the neurons as ξ with the respective index and the activations functions as f with the respective index. With this definitions, the last two layer of the network perform the function $y_k = f_k(\sum_k w_{jk} f_j(\sum w_{ij} x_i))$. The error of the network is still defined in the same way as before. Thus $E_v = \sum_k (y_k - y_k*)^2$. The partial derivative

$$\frac{\partial E_v}{\partial w_{ij}} = \left[\sum_k \frac{\partial E_v}{\partial \xi_k}\frac{\partial \xi_k}{\partial y_j}\right]\frac{\partial f_j(\xi_j)}{\partial \xi_j}\frac{\partial \xi_j}{\partial w_{ij}} = \left[\sum_k \delta_k w_{jk}\right] y_i \frac{\partial f_j(\xi_j)}{\partial \xi_j}.$$

The same simplification of the last partial derivative for the special case of f_j being linear or logistic sigmoid still hold. We used the definition of δ_k we made earlier. At this point it shall be noted, that the above idea does not depend on the fact that k is the output layer, the important trick is to use the previously known partial derivative δ_k. It remains to define the partial derivative also for the hidden layers:

$$\delta_j = \frac{\partial E_v}{\partial \xi_j} = \frac{\partial f_j(\xi_j)}{\partial \xi_j}\sum \delta_k w_{jk}.$$

When combined together, we get the well-known adaptation rules for the back-propagation algorithm. For all the weights in the network

$$w_{ij}^{k+1} = w_{ij}^k - \alpha \delta_j y_i.$$

3.4 RADIAL BASIS FUNCTION NETWORKS

Radial basis function (RBF) networks (Haykin, 1999) are another type of artificial neural networks. They usually have one input layer, one hidden layer, and one output layer. The purpose of the input layer is again to feed the inputs to the hidden layer.

The hidden layer in this case computes a so called *radial basis function*. The values of such a function depend only on the distance of given input from the center of the function c_i. The Gaussian kernel described in the section on support vector machines (Section 3.2) is a popular choice of the radial basis function.

radial basis function

The output layer then computes a linear combination of the outputs of the hidden layer in the hidden layer in case of regression, and moreover applies a sigmoid function in case of classification (where the outputs of the network should be between 0 and 1).

Overall, the function performed by the network is

$$F(x) = \sum_{i=1}^{N} w_i \phi_i(\|c_i - x_i\|),$$

where N is the number of units in the hidden layer, ϕ_i is the radial basis function of the i-th hidden unit, c_i is the center of the function and w_i are weights between the hidden and output units.

training

The training of RBF networks is performed in two steps – first the centers of the radial basis functions are set, (either by clustering algorithm, or they randomly chosen among the vectors in the training set), and later, the weights between the hidden and output layers are set. In the case of linear activation function in the output layer, this corresponds to the problem of linear regression described earlier. In other cases, the weights are often set using a gradient based algorithm. Specifically, if the activation function is a logistic sigmoid, the problem of training the weights correspond to the problem of logistic regression.

3.5 GAUSSIAN PROCESS REGRESSION

Gaussian process regression GPR (Rasmussen and Williams, 2005) is based on the assumption that the observed values are realizations of a Gaussian process. A Gaussian process is a generalization of Gaussian distribution to functions.

Gaussian process

Definition 3.1. *Gaussian process* is a collection of random variables such that each finite set of them has a multivariate Gaussian distribution.

Each Gaussian process is completely defined by its mean $m(x, \theta_m)$ and covariance $k(x, x', \theta_k)$, where θ_m, and θ_k are the hyper-parameters of mean and covariance respectively. Let X be the training data points and f their corresponding observed values. Let X_*, and f_* be the same for testing data. Let us also define (for the sake of

prior distribution

simplicity) $\mu = m(X, \theta_m)$, $\Sigma = k(X, X, \theta_k)$, $\Sigma_* = k(X, X_*, \theta_k)$, and $\Sigma_{**} = k(X_*, X_*, \theta_k)$. Then, the prior distribution of

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma_* \\ \Sigma_*^\intercal & \Sigma_{**} \end{bmatrix} \right).$$

posterior distribution

We are mostly interested in the conditional distribution of the values of test data given the values of the training data. It can be expressed as

$$f_* | f \sim \mathcal{N}\left(\mu_* + \Sigma_*^\intercal \Sigma^{-1}(f - \mu), \Sigma_{**} - \Sigma_*^\intercal \Sigma^{-1} \Sigma_* \right).$$

expected improvement

The fact that Gaussian process regression provides the whole probabilistic distribution as the result of prediction is sometimes used to predict the expected improvement of a given point in the search space. The variance of the model can also be used to describe areas of the search space where the model is not trained well and where a new evaluated point may help to improve the quality of the model.

hyper-parameter setting

The training of Gaussian processes aims at the setting of the hyper-parameters θ_m and θ_k. To this end, the log-likelihood of the training values is maximized. It is expressed as

$$\log p(y | X, \theta_m, \theta_k) = -\frac{1}{2} \log |\Sigma| - \frac{1}{2}(f - \mu)^\intercal \Sigma^{-1} (f - \mu) - \frac{n}{2} \log(2\pi).$$

The hyper-parameters are hidden in the expressions μ and Σ as we have defined them above. The maximization of the log-likelihood is usually performed by a gradient algorithm, however different optimization techniques can also be used, e.g. Qingfu Zhang, Liu, et al. (2010) use differential evolution to tune the parameters.

4 MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

In this chapter, the most notable and state-of-the-art multi-objective evolutionary algorithms are reviewed. The list is definitely incomplete, and the interested reader is referred to (Zitzler, 1999) for an overview of the algorithms created in the 1990s, and to (Zhou et al., 2011) for a more recent survey of the state of the art in this rapidly evolving field. The presented algorithms were selected to provide an overview of the state of the art in the multi-objective optimization field, and to present the most interesting and/or promising ideas. The algorithms are described in their basic versions and their improvements are described at the end of the description of each of the algorithms. We deliberately ignore surrogate versions of the algorithms in this chapter, as the whole next chapter is dedicated to surrogate-based evolutionary multi-objective optimization.

4.1 EVOLUTIONARY ALGORITHMS

Before we explain how some of the multi-objective evolutionary algorithms work, let us present what a single-objective evolutionary algorithm is and how it works.

main loop

Evolutionary algorithms (EA) (Michalewicz, 1996) are an optimization technique inspired by the Darwinian evolution. They operate on a *population* (set) of *individuals* (vectors). Each individual encodes a solution of a given optimization problem. In the beginning, the initial population is randomly generated. Then, the algorithm runs in several iterations, called *generations*. In each iteration, the current population (*parents*) of individuals undergoes *mating selection* and *genetic operators* (*crossover* and *mutation*) are applied to those individuals that are selected. Both crossover and mutation generate new individuals (*offspring*). After new individuals are generated, they are all evaluated using a *fitness function*, which expresses their quality. Finally, at the end of each generation, *environmental selection* is performed. This selection decides which of the offspring and parents survive to the next generation and constitute the new set of parents. The algorithm ends, if a termination condition is met.

Let us now describe the various aspects of evolutionary algorithms in more detail. In continuous evolutionary optimization, the individual is usually a vector of real numbers, however, it may also contain other information (see for example Section 4.5 on multi-objective covariance matrix evolution strategy).

crossover

The crossover and mutation are randomized operations on the individuals. Crossover takes two individuals and create a new individual based on them. Among the most common crossover operators are one-point crossover and arithmetic crossover. The one-point crossover randomly selects a position in the individual and swaps the parts of the two parents after this position. The arithmetic crossover, on the other hand, generates a weight (random number between 0 and 1) and produces a weighted sum of the parents.

mutation

The mutation operator takes only one individual as the input and usually adds random number to some of the positions in the individual. This number is often drawn from the Gaussian distribution with a suitable standard deviation and zero mean. Another option is to generate some of the positions completely randomly, without taking into account their current values.

In evolution strategies (Beyer and Schwefel, 2002), the generation of new individuals is slightly different. Evolution strategies adapt the parameters of a normal distribution (mean and covariance) and new individuals are drawn randomly from this distribution. For more details and an example, see again the Section 4.5 on multi-objective covariance matrix adaptation evolution strategy).

mating selection

The mating selection selects the individuals which undergo the crossover operator. The original idea was that better individuals should have higher probability of being selected for crossover in order to provide offspring with better quality. To this end various selection operators were designed. Among the most popular one is the original roulette wheel selector, which selects individuals with probability proportional to their fitness, and the binary tournament selector, which draws two individuals from the population randomly and selects with high probability the one with better fitness.

environmental selection

The environmental selection, on the other hand decides which of the individuals survive to the next generation. The basic idea is the same as with the mating selection, i.e. better individuals should survive with higher probability. Both roulette wheel selection and tournament selection can again be used here. One of the important consideration is, whether the environmental selection selects only among the new offspring, or also among parents. Also, it is often useful for few of the best individuals to always survive, therefore, so called *elitism* often constitutes part of the environmental selection. In this case, often the best individuals among parents are selected first and only the rest of the population is filled using some random selection operator.

4.2 VECTOR EVALUATED GENETIC ALGORITHM

The Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1985) is one of the first attempts to use an evolutionary algorithm to solve the problem of multi-objective optimization. As such, it has gained a lot of attention and was widely used for comparison with new algorithms developed in the 1990s. However, nowadays, it is mostly interesting from the historical point of view, as the modern algorithms have much better performance.

The main idea of the algorithm is to use each of the objectives to select a proportional part of the population. If there are n objectives and the size of the population is μ, μ/n best individuals according to each of the objectives are selected for the next generation. This selection should ensure that trade-off solutions are found, however, very often the algorithm converges to the optima of the individual objectives after several generations. Although there usually are some trade-off solutions during the run of the algorithms, these are usually not preserved in the population for very long. This is also the main disadvantage of the algorithm – it does not provide a good spread of solutions on the Pareto front.

selection procedure

4.3 NON-DOMINATED SORTING GENETIC ALGORITHM II

The second version of Non-dominated Sorting Genetic Algorithm (NSGA-II) (Deb, Pratap, et al., 2002) is among the most popular and most often used algorithms. Compared to the first version (Srinivas and Deb, 1994), the second version contains a faster non-dominated sorting procedure and removes one niching parameter and replaces it with the crowding distance concept, and most importantly, it adds elitism to the evolutionary algorithm.

The basic idea of NSGA-II is to use the dominance relation in the environmental selection. To this end, each individual in population is assigned a non-dominated rank . The rank is computed as follows:

non-dominated sorting

1. Let $r = 1$.
2. Select the non-dominated individuals in the current population and assign them rank r.
3. $r = r + 1$
4. Remove temporarily the non-dominated individuals from the population.
5. Iterate until the whole population is ranked.

To ensure elitism (i.e. the fact that the best found solutions are not lost during the selection), NSGA-II first merges the parent and children population and the ranks are assigned based on the merged population.

After the rank assignment, the individuals with lower rank dominate the individuals with higher ranks (in the sense of set dominance).

Thus, individuals with lower rank are considered better than individuals with higher rank and are selected first. *selection* The selection proceeds by the numbers of the ranks and adds whole ranks to the population as long as the added rank fits. When the number of individuals of a given rank is larger than the number of free slots in the population a secondary criterion us used to select the individuals with next best rank.

secondary sorting criterion The secondary sorting criterion is traditionally the so called crowding distance. The crowding distance $\mathrm{CD}(I)$ of an individual I is computed using the following algorithm:

1. For each objective function f_i
    a) S = sort the population according to f_i in the descending order
    b) $\mathrm{CD}(S[0]) = \infty$
    c) $\mathrm{CD}(S[n]) = \infty$
    d) For $j = 1$ to $n - 1$
        i. $\mathrm{CD}(S[j]) = \mathrm{CD}(S[j]) + \frac{f_i(S[j-1]) - f_i(S[j+1])}{f_i(S[0]) - f_i(S[n])}$.

The crowding distances expresses how far from the neighboring solutions a given individual is. Individuals with higher crowding distance are selected first to encourage the diversity of the population. The individuals which are the best or the worst in a given objective have the highest crowding distance and are therefore always selected before any trade-off individuals are selected.

Another important feature of NSGA-II are the operators which are used. *simulated binary crossover* The usual crossover operator is the so called simulated binary crossover (SBX) (Deb and Agrawal, 1994). This operator performs arithmetic crossover (i.e. it makes a weighted average of two parents), but the weights are selected in such a way that the change in the values of the variables is similar to the change of variables when one-point crossover on binary encoded strings is used. Basically, it means that the variables of the offspring have higher probability to be closer to one of the parents than if the weights are selected uniformly. *polynomial mutation* The mutation operator – called Polynomial Mutation (Deb and Goyal, 1996) – uses a similar idea. The relative changes in the values of the variables should be similar to those of a bit-flip mutation on binary strings.

It shall be noted that the operators change the individual in directions parallel with the axis of the decision space. This works well for lots of the test problems, which are in fact separable (and thus the functions can be optimized one variable at a time), however, it is not invariant with respect to the rotation of the search space, and may cause problems with non-separable functions. In fact, such behavior is easily observed e.g. on the IHR benchmark set (Igel et al., 2007). Also the operators are not invariant with respect to the scaling of the decision space. Although these features may cause problems when NSGA-II is used to solve practical tasks, the algorithm is still one of the most often used ones in applications.

Another problem of NSGA-II comes from the use of crowding distance as the secondary selection criterion. The problem is that for higher number of objectives most of the individuals in the population are mutually non-dominated, and thus the secondary selection criterion effectively becomes the only selection criterion and the algorithm looses the selection pressure towards the Pareto optimal set and only the pressure for diversity remains (Ishibuchi, Tsukamoto, et al., 2008). Similar problem in a lesser extent can be observed even for lower numbers of objectives in the later phases of the evolution when the algorithm almost converges. In this case the population is also usually full of non-dominated individuals. The problem can be reduced by using a different secondary criteria – e.g. the hypervolume contribution (Emmerich, Beume, et al., 2005).

different secondary criteria

4.4 INDICATOR-BASED EVOLUTIONARY ALGORITHM

Indicator-Based Evolutionary Algorithm (IBEA) (Zitzler and Künzli, 2004) shows that performance indicators can be used directly in the fitness assignment inside the multi-objective evolutionary algorithms. The use of indicators, which in a sense generalize the dominance relation, also leads to better performance on problems with more objectives.

In the case of IBEA, two different binary indicators are used, the additive ϵ_+indicator or the hypervolume indicator.

indicators

Definition 4.1. Let A, B be two Pareto set approximations. The binary additive ϵ_+ indicator is defined as

$$I_{\epsilon^+}(A, B) = min_{\epsilon \in \mathbb{R}} A + \epsilon \preceq B,$$

where the $A + \epsilon$ is the Pareto set approximation obtained from A by adding ϵ to the values of all objectives for all of the individuals in the set.

The binary hypervolume indicator is defined as

$$I_{\mathrm{HD}}(A, B) = H(A, r) - H(B, r)$$

for a given reference point r.

We can notice in both cases that the value of the indicator is negative when $A \preceq B$, and is positive in the opposite case. In these two cases it also holds that $I(A, B) = -I(B, A)$. In the case the two sets are mutually non-dominated both the values $I(A, B)$ and $I(B, A)$ are greater than zero.

In IBEA the indicators are used to define the fitness values of an

fitness function

individual $F(i)$ as

$$F(i) = \sum_{j \in P \setminus \{i\}} e^{-\kappa I(\{i\},\{j\})},$$

where i and j denote two different individuals from the current population P. The reason for the exponential is to enlarge the differences between dominated and non-dominated individuals, and κ is a normalization coefficient. The authors recommend to scale the values of the indicators to the interval $[-1, 1]$ and use $\kappa = 1$.

environmental selection

The environmental selection in IBEA always removes the worst individual from the population and updates the fitness of all the remaining individuals

$$F(i) = F(i) - e^{-\kappa I(\{i\},\{i*\})},$$

where i^* is the removed individual.

The original version of IBEA uses the same operators as the NSGA-II algorithm and thus inherits its problems with non-separable functions. On the other hand, the use of indicators means that IBEA does not need any secondary sorting criterion, and the search is guided completely by the values of indicators. Another advantage of this approach is that the indicators can incorporate the decision maker's preferences to the search and and guide the algorithm towards interesting parts of the Pareto front.

4.5 MULTI-OBJECTIVE CMA-ES

Another popular algorithm, Multi-objective Covariance Matrix Adaptation Evolution Strategy (MO-CMA-ES) (Igel et al., 2007), is based on the idea of running multiple instances of Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier, 2001) at once. The main advantage this brings is that the algorithm is invariant to rotations of the search space, as well as to different scaling. This significantly improves the performance of the algorithm in the case of non-separable objectives.

To describe the MO-CMA-ES we have to start with the description of the single-objective $(1+\lambda)$-CMA-ES. The basic idea of CMA-ES is to learn the right coordinate system for a given problem. To this end, the algorithm adapts a covariance matrix, which is used to generate new individuals. Once a new individual is generated, it is checked against its parent, and in case it is better, the covariance matrix is updated in such a way to further encourage the generation of new individuals in the same direction. CMA-ES also adapts the size of the steps it performs in such a way that the step size is increased, if the probability of improving steps increases and decreases if the probability of good steps decreases.

More formally,$(1+\lambda)$-CMA-ES is defined by the covariance matrix C, the evolution path p_c, and the step size σ. Moreover, it also computes the probability of a successful step p_s (i.e. probability that the newly generated individual is better than the parent) which is used during the step-size update. The covariance matrix is used to generate new individuals while taking care of different scales of different variables and various rotations of the decision space. The evolution path records the directions of last successful steps and is used to increase the covariance in this direction, and the step size controls, how far from the parent the individuals will be generated. It is also used the control the probability of successful step.

offspring generation

The algorithm uses only one parent x and in each step generates λ new offspring by sampling the random distribution $\mathcal{N}(x, \sigma^2 C)$. After the offspring are generated, the step size is updated, as well as the probability of successful step p_s. The update rule for the probability p_s is in fact exponential smoothing of this value over the generations of the algorithm

$$p_s = (1 - c_p)p_s + c_p \frac{\lambda_{succ}}{\lambda},$$

where λ_{succ} is the number of offspring which have better objective value than their parent and c_p is the smoothing coefficient.

step size update

The update of the step size σ is based on the well known $\frac{1}{5}$ rule (Schumer and Steiglitz, 1968). Its goal is to make the probability of success close to a pre-defined value p_{target}. In each step, the step size is adapted as

$$\sigma = \sigma \exp\left(\frac{1}{d} \frac{p_s - p_{target}}{1 - p_{target}}\right),$$

where the parameter d controls the speed of the step size adaptation.

evolution path and covariance update

The last step of the algorithm is the update of the evolution path p_c and the covariance matrix C. The update of the evolution path is again a smoothing of the last steps of the algorithm.

$$p_c = (1 - c_c)p_c + \sqrt{c_c(2 - c_c)} \frac{o - x}{\sigma},$$

where o is the vector of the best offspring, and x is the vector of the parent; c_c is again a smoothing parameter and the expression $\sqrt{c_c(2 - c_c)}$ normalizes the variance of p_c. The evolution path is finally used to update the covariance matrix C.

$$C = (1 - c_{cov})C + c_{cov} p_c p_c^\intercal,$$

where (again) c_{cov} is a smoothing parameter and $p_c p_c^\intercal$ is the outer product of p_c and $p_c^\intercal$.

In case the probability of successful step is large, the update of the evolution path is stalled (the second term in the sum is missing) to avoid too fast increase of the vector. The adaptation of the covariance matrix C is changed accordingly to account for the stalled update of the evolution path.

Now, we can finally explain the MO-CMA-ES. The version we describe here is more specifically $\lambda_{\text{MO}} \times (1 + 1)$-MO-CMA-ES. The name *multi-objective CMA-ES*

implies that the algorithm uses λ_{MO} individuals, each of which represents all the parameters of a single $(1+1)$-CMA-ES (thus, each individual i is a vector $[x^i, p^i_s, p^i_c, \sigma^i, C^i]$). In each generation, each of the λ_{MO} parents generates one new individual. If the generated individual dominates the parent, or they are mutually non-dominated and the generated one is better according to the secondary criterion, the parameters of the individual are adapted in the same way we described above for the CMA-ES. At the end of each generation, non-dominated sorting, and either crowding distance, or hypervolume contribution is used to select the individuals which survive to the next generation, similarly to the way environmental selection is performed in NSGA-II.

More recently (Voss et al., 2009), the original MO-CMA-ES was improved. In this version of $(\lambda_{\text{MO}} + \lambda_{\text{MO}})$-MO-CMA-ES, each individual does not only affect the covariance matrix of his own and his parent, but also the covariance matrices of neighboring individuals (the neighborhood is defined by closeness in the decision space). This leads to a faster convergence, as the information obtained during the search is shared among the individuals. Also, strategies for the selection of parents were studied (Loshchilov et al., 2011) with the goal to improve the diversity in the population. The parents were selected based on the multi-armed bandit scenario in order to balance the exploration and exploitation of the algorithm and prevent premature convergence.

4.6 MOEA BASED ON DECOMPOSITION

decomposition

A different approach is used in the Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) (Qingfu Zhang and Li, 2007). The algorithm decomposes the multi-objective problem into a series of single-objective problems. The type of decomposition is not important from the point of view of the algorithm, in fact, three different decomposition strategies are studied in the original paper on MOEA/D – weighted objectives, Tchebychev approach, and normal boundary approach (all known in the classical multi-objective optimization field (Miettinen, 1999)). All of the decomposition strategies lead to a weighted sum of the objectives, and each setting of weights corresponds to a decomposed problem. Each of the decomposed problems are represented by a single individual in the algorithms.

neighborhood

All the decomposed problems are solved in one run, a neighborhood for each problem is defined based on the Euclidean distance of the weights – the closest weights define the neighborhood. The genetic operations are performed only on individuals from the same neighborhood. The motivation for this is that the decomposition is continuous in the weights, and thus solutions of one decomposition can benefit from the information obtained for similar decompositions.

In each generation, two individuals from each neighborhood are selected and the genetic operators are applied to them to create a new individual. This individual is then improved by a problem-specific heuristic. Finally, the individual is evaluated, and compared to the individuals in its neighborhood. All individuals in the neighborhood which are worse than the new individual are replaced by the new one.

operators

The algorithm uses the simulated binary crossover and polynomial mutation we have already discussed in the part on NSGA-II. This again means that the algorithm is not invariant with respect to most transformations of the decision space. Moreover, the decomposition approach affects the spread of the solutions on the Pareto front – the problem is that even spread of weights does not imply even spread of solutions in case the range of one of the functions is significantly different. The authors partially solve this problem by the scaling of the objectives into the same range.

Additionally, the authors claim that the computational complexity of MOEA/D is lower than the one of NSGA-II and that the obtained solutions have similar or better quality than those found by the NSGA-II.

improvements

Later (Sindhya et al., 2011), the algorithm was improved by the change of the operators to ones which use the ideas from differential evolution, thus providing some invariance with respect to transformations of the search space. MOEA/D was also used in a many-objective setting (Ishibuchi, Sakane, et al., 2009), and better decomposition strategies were proposed (Qingfu Zhang, Li, et al., 2010).

5 SURROGATE-BASED EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION

The idea of surrogate modeling is quite new in the area of multi-objective evolutionary algorithms. In this chapter, we first describe the possibilities, how surrogate models can be used to enhance the performance of multi-objective optimizers, and than we describe some of the approaches that can be found in the literature. We pay more attention to those, which affect our work.

Generally, there are at least two ways how to augment an evolutionary algorithm with surrogate models: one is a local-search operator based on the surrogate model, the other is surrogate-based pre-selection. The two approaches differ in the way the information provided by the model is exploited.

pre-selection

In the pre-selection scenario, the model is used to pre-screen individuals in the population. The individuals which seem to be promising according to the model are evaluated by the real objective function. Being promising may mean a few different things: an individual may be promising, because it is predicted to have a good value of the real objective. However, an individual may also be promising, because knowing his value would improve the quality of the model. The latter case is sometimes used in cases, where the model not only predicts the value, but it is additionally able to predict its own error for a given input. This is for example the case of Gaussian process regression, which can predict the variance of the model in each point, and points with high variance indicate areas with poor quality of the model (or not yet visited areas, which may be interesting to explore).

local search

In the local search case, the model is used to create new individuals directly – the model is trained and then a local search algorithm is executed to find the optima of the model. This optima is later added to the population as a new individual. Under the assumption that the model is at least partially correct, the new individual should be better than the individuals in the population, or at least interesting. Of course, the quality of the model is what affects the outcome the most. A special case of local search may the generational approach to surrogate modeling. In this case the model is trained and is then used instead of the real objective function for a number of generations. These generations can also be (equivalently) described as the run of a local search algorithm which seeks the optima of the surrogate model. In this case, the local search algorithm is the same as the external algorithm.

model exploitation

The performance of the surrogate assisted evolutionary algorithm is largely affected by the quality of the model, and by the amount of exploitation. Loshchilov et al. (2012) argue that the exploitation of the model should be inversely proportional to the error of the model, i.e. model with small error may be exploited longer than model with larger error. One can of course re-train the model in each generation, however, this requires the evaluation of new individuals, which may be expensive, and also the training of the model may be rather slow.

overhead

The overhead caused by the modeling may be quite large in some cases, however, it is mostly assumed negligible. The assumption in this case is that the evaluation of the objective function is expensive (say, in the order of minutes, or even hours), and thus the time required to evaluate the objective constitutes major part of the run time of the algorithm. In this sense, almost any overhead can be justified, as long as it leads to the reduction in the number of evaluations of the real objective function. Obviously, algorithms with higher overhead are not suited for solving problems with faster objectives. Often, there is a trade-off between the overhead (i.e. the time required to build and exploit the model), and the reduction in the number of function evaluations.

5.1 NSGA-II WITH SURROGATE MODELS

Voutchkov and Keane (2006) described one of the first uses of surrogate modeling in multi-objective optimization. They use NSGA-II and create a surrogate for each of the objectives separately. First, some points are generated in the search space and evaluated by the real objective function. These form an archive of evaluated individuals and are used for the training of surrogate models. After the surrogates are trained, NSGA-II is started and optimizes the surrogate models instead of the real objectives. After this evolution finishes, some evenly spaced points are selected from their final population and these are evaluated using the real objective function. These are added to the archive. If the size of the archive is larger than a pre-defined threshold, it is truncated using a procedure which selects the points closest to the Pareto front. The new archive is then used to train new surrogate models, and the steps are repeated until a termination condition is met.

The authors use several different types of surrogate models and show their performance. They also argue that deceptive and multimodal functions are difficult to optimize using surrogates, as the models exploit the information already known about the search space and it is thus easier to find a local optima instead of the global one. All the experiments are performed on a rather easy set of benchmark function with only two variables.

artificial neural networks

Nain and Deb (2005) propose a version of NSGA-II augmented by artificial neural networks (multilayer perceptrons). In this variant, the NSGA-II first optimizes the real objectives directly for n generations and than models of the objectives are created based on the individuals evaluated in the last n generations. The models are optimized for another m generations. Here, n and m are two parameters set in advance. The steps with the optimization of the real problem and the approximation are repeated until a termination condition is met. The authors show approximately 25%-50% reduction in the number of real objective evaluations compared to NSGA-II while the quality of the solutions is retained.

5.2 MOEA/D WITH GAUSSIAN PROCESS MODEL

Qingfu Zhang, Liu, et al. (2010) created a Gaussian process based version of MOEA/D, called MOEA/D-EGO. This algorithm is specifically designed to work with extremely low numbers of function evaluations (the authors use it for only 200 evaluations).

Similarly to MOEA/D, the multi-objective problem is decomposed into several single-objective problems and these problems are solved at once. The algorithm starts by generating some random individuals and evaluating them by the real objective functions to create the initial archive which is used during the training of the models. Then, a Gaussian processes based model for each of the objectives is trained. All the models are optimized using MOEA/D. The goal of the optimization is to find the point with maximum expected improvement. The advantage of using MOEA/D is that all the single-objective problems are solved at once, and that problems with similar decomposition share the information (in the same way as in original MOEA/D). After the optima are found, several of these individuals are selected for evaluation by the real objective function. These are then added to the archive of evaluated individuals and used for the model building in the next iteration. The algorithm runs until a termination criterion is met.

model building

The hyper-parameters of the Gaussian processes are estimated using a differential evolution based algorithm. To reduce the modeling overhead, the population is first clustered into several clusters using a fuzzy clustering technique (Bezdek, 1981), and a local model is build for each of the clusters. The overhead is further reduced by using the properties of the Gaussian processes models – in each of the clusters, one model is build for each of the objectives, and the models for the different decompositions are derived from these models directly.

reducing the overhead

5.3 AGGREGATED SURROGATE MODELS

Loshchilov et al. (2010a) came with the idea to use a single aggregated surrogate model instead of individual models for each of the objectives. The aggregated surrogate model is trained to discriminate between the dominated and non-dominated points in the decision space. The main idea is to map the current Pareto set to a small interval $[\rho - \epsilon, \rho + \epsilon]$ and the dominated points into the interval $[-\infty, \rho - \epsilon]$. The new non-dominated points should then be in the interval $[\rho + \epsilon, \infty]$. To this end ideas from Support Vector Regression, and OneClass SVM are used, and these conditions are formulated as an SVM training problem in the following way.

SVR and OneClass SVM

$$
\begin{array}{ll}
\min\limits_{w,\xi,\rho} & \frac{1}{2}\|w\|^2 + C\sum\limits_{i=1}^{l}(\xi_i^+ + \xi_i^-) + C\sum\limits_{i=l+1}^{m}\xi_i^+ + \rho \\
\text{subject to} & \begin{array}{ll}
\langle w, x_i\rangle \le \rho + \epsilon + \xi_i^+ & 1 \le i \le l \\
\langle w, x_i\rangle \ge \rho - \epsilon - \xi_i^- & 1 \le i \le l \\
\langle w, x_i\rangle \le \rho + \epsilon + \xi_i^- & l+1 \le i \le m \\
\xi_i^+ \ge 0 & 1 \le i \le l \\
\xi_i^- \ge 0 & 1 \le i \le m
\end{array}
\end{array}
$$

In this formulation x_1 to x_l are the individuals in the current Pareto set, and x_{l+1} to x_m are the dominated individuals, ξ^+ and ξ^-denote the vectors of slack variables.

The first two conditions correspond to the SVR conditions we have seen earlier (i.e. the mapping of current Pareto set to the interval $[\rho - \epsilon, \rho + \epsilon]$) and the third condition corresponds to the OneClass SVM condition (i.e. to the mapping of dominated individuals into the interval $[-\infty, \rho - \epsilon]$). This primal problem is transformed into a dual problem and solved, which leads to a training procedure similar to the one used in SVM.

The authors use the model to augment NSGA-II and MO-CMA-ES. In both cases, if the algorithm generates one offspring, λ_{pre} offspring are generated in the new algorithm and the best according to the surrogate model is selected and later evaluated using the real objective.

rank-based SVM

Later, Loshchilov et al. (2010c) proposed a new aggregated surrogate model, this time based on rank-based SVM (Joachims, 2005). The rank-based SVM uses constraints which ensure that better ranked individuals have larger values predicted by the model. Let $\mathcal{P}$ is the set of pairs (i, j), such that $(i, j) \in \mathcal{P}$ if x_i is preferred to x_j. The formulation of the primal problem is as follows:

$$
\begin{array}{ll}
\min\limits_{\xi,w} & \frac{1}{2}\|w\|^2 \quad + C\sum\limits_{i=1}^{|P|}\xi_i \\
\text{subject to} & \begin{array}{ll}
\langle w, x_i\rangle - \langle w, x_j\rangle \ge 1 - \xi_i & (i,j) \in \mathcal{P} \\
\xi_i \ge 0 & \forall i
\end{array}
\end{array} ,
$$

where ξ are again slack variables. As always with SVM-based models, the dual problem is constructed and solved to train the model. To make the optimization more effective, the constraints from the set are added iteratively, one constraint at a time. Each time, the most violated constraint is added.

The question which remains to answer is how the set $\mathcal{P}$ is chosen. Loshchilov et al. added the pair (i, j) to the set $\mathcal{P}$, if x_i is the closest individual to x_j such that, x_j is dominated by x_i – these form what they call *primary constraints*. There is also a set of *secondary constraints*, which are defined as $(i, j) \in \mathcal{P}$, if x_i is in the current Pareto set and x_j is not. The training of the model starts with the set of primary constraints only, and after a while, the secondary constraints are added one by one (the most violated first), until a pre-defined fraction of them (the authors used 10%) are added.

constraints

This model is then used to augment the NSGA-II and MO-CMA-ES in the same way as the first aggregated surrogate model. One of the advantages of the latter approach is that it can in theory contain any preference information, and can thus be used to guide the search in directions interesting for the decision maker.

The authors show that this kind of models reduces the number of function evaluations needed to attain a specific quality of solutions (defined by the hypervolume of the dominated space) by more than 50% compared to the non-surrogate versions of the algorithms.

5.4 GENERALIZED SURROGATE MOMA

The approach presented in (Lim et al., 2010) is interesting for a different reason. The author describe one of the few algorithms which deal with more than one surrogate model. They use two types of models – one of them is a low degree polynomial, the other is an ensemble of models. The intuition is that the ensemble model would be more precise, however it can overfit the training data and thus miss-lead the algorithm. On the other hand, the low order polynomial should be able to generalize better and it should also smooth multi-modal functions and make them easier to optimize.

ensembles and polynomials

The authors use the same approach both for single-objective and multi-objective optimization, however, we will only describe the multi-objective case here. In the multi-objective case, in each generation for each individual x a set of weights is generated randomly and an aggregate objective is created as the weighted sum of the real objectives. Then, the two surrogate models are trained on a training set created from the archive of previously evaluated individuals to predict the value of the aggregate objective. After that a local search is performed on both the models to yield individual x_1 and x_2 as the optima of the models, these are then evaluated. Finally, individuals x, x_1, and x_2 are compared and the non-dominated ones of them are added to the population so that they are considered for selection to the next generation.

local search

Part III

CONTRIBUTION

This part contains the main contribution of the thesis. First, we describe the steps in the creation of a multi-objective evolutionary algorithm with local search based on an aggregate surrogate model and with pre-selection. Later, we deal with the problem of model selection in the field of multi-objective optimization, and, finally, we show how multi-objective optimization may be used to solve some single-objective problems more effectively.

6 DISTANCE-BASED AGGREGATE SURROGATE MODEL

In this chapter we describe one of the most important contributions of this thesis – the distance-based aggregate surrogate model. The model was inspired by the SVR and OneClass SVM based model by Loshchilov et al. (2010a) (see also Section 5.3 of this thesis). From a historical point of view, it was created independently from the model based on ranking SVM based presented by the same authors in (Loshchilov et al., 2010c), although the two models share some similarities.

motivation

The main motivation for us was to augment the multi-objective optimizers with a local search, as it should improve the results. The intuition we had was that the local search would in a sense take some of the points in the search space and move them closer to the real Pareto set. To this end, the use of aggregates surrogate models seems interesting, as they provide only a single value, which can be more easily optimized compared to the multiple objective values of the original problem. Having a single value means one can use a single-objective evolutionary algorithm to exploit the model, and single-objective evolution is both faster and more simple to perform.

However, the model presented by (Loshchilov et al., 2010a) did not seem to be a good candidate for local search, as it only discriminates between the Pareto set and the rest of individuals. But the predicted values for dominated individuals are not constrained in any way and thus each of these individuals may have a different value, which may not correspond to the quality of the individual at all. This could miss-lead the local search, as was also discussed by the authors of the model. Therefore, we came with the idea to describe the search space using distance of the individuals to the current Pareto set. The distances are computed in the decision space, which makes the model invariant to any rank-preserving transformations of the objective functions.

historical remark

Historically, the algorithm described in this chapter was presented in two steps. We first created the multi-objective memetic algorithm with aggregate surrogate model (ASM-MOMA), and created the more local models used in LAMMA (multi-objective memetic algorithm with local aggregate meta-model) later. However, we do not make the distinction in this thesis and describe both of the algorithms at once.

Algorithm 6.1 The main loop of the algorithm

```
Require: $G_{init}$: the number of generations to build the archive
  $t \leftarrow 0$
  Initialize randomly new population $P_0$
  Initialize empty archive $A$
  while Termination criterion not met do
    Use crossover and mutation to create new population $P'$
    if $t \geq G_{init}$ then
      for Each individual $I$ in the population $P'$ do
        if Random number from $[0, 1] \leq p_{mem}$ then
          Use the individuals in the archive $A$ to train new model $M_{t,I}$
          Run memetic operator on individual $I$ and replace it in $P'$
    Compute the objective values for each new individual in $P'$
    Add all new individuals to the archive and truncate it
    $P_{t+1} \leftarrow$ selected individuals from $P_t \cup P'$
    $t \leftarrow t + 1$
  return The non-dominated individuals from the population
```

6.1 MULTI-OBJECTIVE MEMETIC ALGORITHM WITH AGGREGATE SURROGATE MODEL

overview Multi-objective memetic algorithm with aggregate surrogate model (ASM-MOMA) describes a general framework to augment existing multi-objective evolutionary algorithms and add a surrogate-based local search to them. The word "memetic" in its name refers to the new additional genetic operator which is added into an existing multi-objective algorithm. The new genetic operator essentially works as a mutation, in the sense that it only changes one individual, but it is not completely random. It uses the information provided by the surrogate model to guide the search. In fact, the operator runs its own internal evolutionary algorithm to find the optima of the model. The right place to add this operator into an evolutionary algorithm is right before the newly generated individuals are evaluated.

surrogate model The surrogate model used by the operator is constructed based on an archive of previously evaluated points from the decision space. The archive stores the values of the objective functions for these points. The surrogate model is trained to predict the distance to the currently known non-dominated solutions. Moreover, as an addition to ASM-MOMA, in LAMMA the points do not not have the same weight, as those that are closer to the locally optimized one are considered more important during the model building phase (see next section for details).

intuition The main idea is that points closer to the current Pareto front are more interesting during the run of the algorithm and the memetic operator moves the individuals closer to the Pareto front and hopefully even finds new non-dominated solutions. The purpose of the surrogate model is not to precisely predict the value but rather provide a general direction in which the memetic search should proceed.

training set

To obtain a training set for the meta-models we also added an external archive of individuals with known objective values. This archive is updated after each generation when new individuals are added and at the same time the archive is truncated to ensure it does not grow indefinitely.

The following sections detail the important parts of the algorithm. The main loop (see Algorithm 6.1) is essentially a generic Multiobjective Evolutionary Algorithm (MOEA) with an added memetic operator.

6.1.1 *Meta-model construction*

We train a dedicated model for each individual I which shall be locally optimized by the memetic operator. For such an individual I we create a weighted training set

$$T_I = \{((x_i, y_i), w_i) | y_i = -d(x_i, P), w_i = \frac{1}{1 + \lambda d(x_i, I)}\},$$

where $d(x, y)$ is the Euclidean distance of individuals x and y in the decision space, P is the set of non-dominated individuals in the archive and $d(x, P)$ is the distance of individual x to the closest point in the set P. λ is a parameter which controls the locality of the model, larger values of λ lead to more local model, whereas lower values lead to more global one.

model locality

The points which are closer to the individual I are more important during the training of the model. This distance weighting adds some locality to the models trained for each individual. The training set is constructed in such a way, that for the individuals closer to the currently known Pareto front the meta-model should return larger values. This fact is used during the local search phase (which uses the meta-model as a fitness function).

global model

The training set described above corresponds to the training set of LAMMA as it was originally described. In order to obtain the training set for ASM-MOMA, it is enough to set the locality parameter $\lambda = 0$. This also ensures that the model does not depend on the optimized individual I and, thus, it is enough to train the model only once per generation, instead of re-training it for each optimization.

The target value of the model depends only on the distance of the individual from the Pareto front. The non-dominated points have the value of 0.0 and any dominated points have negative values. Ideally, after the local search phase, there would be new non-dominated points in the population which should have positive values predicted by the model.

Algorithm 6.2 Meta model training

```
Require: The archive of evaluated individuals $A$
Require: The optimized individual $I$
Require: The locality parameter $\lambda$
  Initialize empty training set $T_I$ and meta model $M_I$
  $N \leftarrow$ non-dominated individuals in the archive
  for each individual $x_i$ in the archive $A$ do
    Let $d(x_i, N) \leftarrow$ distance of $x_i$ to the closest individual in $N$
    Let $d(x_i, I) \leftarrow$ distance of individual $x_i$ to the optimized individual $I$
    Add $\langle(i, -d(x_i, N)), 1/(1 + \lambda d(x_i, I))\rangle$ into $T_I$
  Train the model $M_I$ on data $T_I$
  return The trained meta model $M_I$
```

The model does not respect the dominance relation: a point dominated by a lot of others may have higher target value than another point further from the Pareto front. We also experimented with models based on the number of non-dominated front in which the particular individual is, but such models did not work well. The distance based models described here provide better information about the search space and guide the local search algorithm towards the Pareto front.

6.1.2 *Local search*

memetic operator

In the local search phase during the run of the memetic operator (see Algorithm 6.3) we use another evolutionary algorithm (this time it is only a single objective one) to find better points in the surroundings of each individual. The algorithm runs only for a few generations and it uses only meta-model evaluations. The newly found individuals are placed back to the population. During the initialization of the local search the individual which should be optimized is inserted into the initial population and its variables are perturbed to create the rest of the initial population. The perturbation adds random number with Gaussian distribution and the standard deviation of $\frac{1}{10}$ of the range of the variable.

Note, that any other optimization method could in theory be used for finding of the optima of the surrogate model, however, we chose the evolutionary algorithm for two reasons: it does not need any assumptions about the meta-model used, and due to its randomized nature, it provides multiple different points which lead to better diversity in the population of the external algorithm. Although the complexity of the evolutionary algorithm is larger than the complexity of other methods, the local search is still limited mainly by the

Algorithm 6.3 Memetic operator

```
Require: The trained meta model $M$
Require: The individual $I$ to be improved
  Perturb $I$ to create initial population $P$
  Add $I$ to the population $P$
  Use evolutionary algorithm with $M$ as fitness to improve $I$
  return The best individual found
```

time needed for the training of the model as we shall see in Section 6.2.

6.1.3 *Archive truncation*

The algorithm uses an archive of previously evaluated individuals. This archive is used in the model building phase.

filling the archive

Before the algorithm can build the surrogate model, it needs a few initial generations (G_{init}) to fill the archive with enough individuals for training. This number of generations depends on the number of variables, the number of individuals in population and the complexity of the chosen model.

truncation

Moreover, the size of the archive should be kept under a certain limit to prevent large memory usage, therefore archive is truncated after each generation. The truncation process is very simple: random individuals are selected for removal from the archive. In preliminary tests we tried different archive truncation techniques (e.g. the dominance based truncation like in NSGA-II), but the random approach appears to give the best results. In the random approach more recent individuals are more likely to stay in the archive, thus being used to training the meta-model.

Regarding the age of the individuals in the archive more specifically, let μ be the population size and A is the archive size. Each individual has the probability: $p = \frac{\mu}{\mu+A}$ of being removed from the archive during truncation. Therefore, the archive contains on average $I_t = \mu(1-p)^t$ individuals of age t (i.e. individuals which survived t archive truncations in the past). The average age of an individual in the archive is

$$\frac{\sum_{i=1}^{\infty} \mu i(i-p)^i}{A} = \frac{\mu}{A}\frac{1-p}{p} = \frac{A(A+\mu)}{\mu^2}.$$

The probability, that an individual survives exactly t archive truncations is: $p_t = (1-p)^{t-1}p$ and it has a geometric distribution. This implies that on average an individual survives $\frac{1}{p} = \frac{\mu+A}{\mu}$ generations in the archive.

In the experiments, we use the population size of 50 individuals and archive size of 400 individuals. For this configuration each individual survives on average 9 generations in the archive and half of

Table 6.1: The notation used in the equations

Symbol	Meaning
T_o	The time of an objective function evaluation
T_t	The time of meta-model training
T_m	The time of meta-model evaluation
G_e	Number of evaluated generations (external EA)
G_i	Number of evaluated generations (internal EA)
P_e	External EA population size
P_i	Internal EA population size
p_{mem}	Memetic operator probability
R	Reduction of the number of evaluations

the individuals in the archive was added during the last six generations. The observations above show the time locality of the archive – it contains mostly individuals from the last generations with higher amount of individuals from the more recent ones.

6.2 MODELING OVERHEAD

overhead Generally, when talking about surrogate modeling, the assumption is that the fitness evaluations take a long time and therefore the complexity of other parts of the algorithm is negligible. In this section we would like to discuss, how long it takes for the algorithm to create the model and make all the evaluations, and therefore how long must the evaluation of the real fitness function take in order to hide this overhead. The analysis provided bellow assumes the local models used in LAMMA which must be retrained for each optimized individual. Equations for ASM-MOMA would be similar, only the time needed to train the model would be accounted only once for each generation.

assumptions The proposed algorithm uses quite a large number of meta-model evaluations and even meta-model trainings. In this section we would like to discuss the usability of this approach. In the equations bellow we use the notation defined in Table 6.1. We ignore the time needed to run the selection of the multi-objective algorithm after the values of the objective functions are known, as well as the time consumed by the genetic operators. Adding this would make the difference between the memetic and original variants even larger as the time for these is constant in each generation and the memetic algorithm should need less generations. On the other hand we also ignore the time needed by the selection and operators of the internal evolutionary algorithm.

Note that both the initial and final populations need to be evaluated, therefore the factor G_e is one higher than the number of gener-

Table 6.2: Times needed for training and evaluation of selected meta-models, in seconds

Model	Training (T_t)	Evaluation (T_m)
Linear regression	0.142	8.46×10^{-7}
Support vector reg.	0.328	7.14×10^{-7}
Multilayer perceptron	3.75	1.80×10^{-5}

ations of the external evolutionary algorithm (the same holds for G_i and internal evolutionary algorithm). The original external algorithm takes the time

$$T_{orig} = G_e P_e T_o$$

to finish.

surrogate algorithm run time

The use of the meta-model reduces the number of generations needed, but on the other hand adds time to train and evaluate the local models. It takes

$$T_{meta} = RG_eP_eT_e + RG_eP_ep_{mem}(T_t + P_iG_iT_m)$$

to find the solution of the same quality. The first part is identical to the original algorithm with the reduced number of generations. The second part corresponds to the training and local search. We consider the local meta-models here, otherwise there is only one model training per generation which makes the algorithm more effective.

condition for speed up

Now, we would like to know, how much faster the meta-model training and evaluations need to be (compared to the original objective evaluation) for this method to speed up the optimization, i.e. under which conditions the inequality $T_{meta} < T_{orig}$ holds. After substituting the above expressions and solving for T_o we get

$$T_o > \frac{R}{1-R} p_{mem}[T_t + G_iP_iT_m].$$

This inequality holds for R between 0 and 1. R is always positive and if R is larger than 1, no reduction is made and therefore the memetic algorithm cannot work faster than the original one.

modeling time

Although the factor G_iP_i may be quite large the training of the meta-model is what usually dominates the time in this case. Table 6.2 shows results of measurements we made in order to find out how fast some of the models we used are. All the test were done on a computer with Intel Core i7 920 (2.87Ghz) processor and 6GB RAM. The size of the training set was set to 400 and the training set was obtained during the run of the described algorithm.

Based on these measurements and equations above we can compute the limit time of the objective function evaluation for which the use of our algorithm reduces the time needed to find a solution. Some of these are computed in Table 6.3. (The remaining parameters were

Table 6.3: Theoretical limit evaluation threshold T_0 for which the real time reduction is achieved considering various values of R, in milliseconds

R	0.1	0.2	0.5	0.8	0.9
Linear regression	4	8	30	144	324
Support vector reg.	9	21	82	330	742
Multilayer perceptron	105	237	949	3,800	8,540

Table 6.4: Parameters of the multi-objective algorithm

Parameter	MOEA value	Local search value
Stopping criterion	50,000 evaluations	30 generations
Population size	50	50
Crossover operator	SBX	SBX
Crossover probability	0.8	0.8
Mutation operator	Polynomial	Polynomial
Mutation probability	0.1	0.2
Archive size	400	–
Memetic operator prob.	0.25	–
Locality parameter λ	–	1

$p_{mem} = 0.25$, $G_i = 50$, $P_i = 50$.) We can see that especially with the faster models (linear regression and support vector regression) the algorithm is theoretically able to reduce the run time even for relatively cheap objective functions given rather small reduction in the number of evaluations (e.g. for $R = 0.9$, which translated to only 10% reduction in the number of evaluations, speed up is achieved even for objectives which evaluate in under a second). We can also see the effect of the relative slowness of multilayer perceptrons which indicate that they might not be very advantageous unless they are able to provide much better reductions than the other models.

Later in this chapter, we show that the values of $R = 0.2$ and even lower are possible to obtain. This means, that LAMMA is usable even for problems with relatively fast objective functions which take only milliseconds to evaluate.

6.3 EXPERIMENTS – BI-OBJECTIVE PROBLEMS

benchmark sets

To evaluate the performance of ASM-MOMA and LAMMA in the bi-objective case, we tested our approach on the widely used ZDT Zitzler, Deb, et al. (2000) benchmark problems. These problems are all two dimensional, and we used 30 variables for ZDT1 and 15 variables for the other problems. In the local search phase we used various types of models: namely multilayer perceptron, support vector regression, and linear regression. All the models use default parameters from the Weka framework Hall et al. (2009) (which we used to run the experiments), i.e. polynomial kernels and normalization for the support vector regression and learning rate of 0.2 and momentum of 0.3 for multilayer perceptron, together with 4 neurons in the hidden layer, the instances are again normalized.

models

See Table 6.4 for the parameters of the main multi-objective algorithm and the internal single-objective algorithm.

We used the NSGA-II and ϵ-IBEA with Simulated Binary Crossover Deb and Agrawal (1994) and Polynomial Mutation Deb and Goyal (1996) as the external multi-objective evolutionary algorithm. In the local search phase we used a simple single objective evolutionary algorithm with the same operators and the surrogate model served as its fitness function.

algorithm settings

6.3.1 *Performance measure*

To compare the results we use a measure we call H_{ratio}, it is defined as the

$$H_{ratio} = \frac{H_{real}}{H_{optimal}}$$

where H_{real} is the hypervolume of the dominated space attained by the algorithm and $H_{optimal}$ is the hypervolume of the real Pareto set of the solutions. As the Pareto set is known for all the ZDT problems, we can compute this number directly. We use the vector $\vec{2} = (2, 2)$ as the reference point in the hypervolume computation. All points that do not dominate the reference point are excluded from the hypervolume computation.

We compare the median number of function evaluations needed to attain the H_{ratio} of 0.5, 0.75, 0.9, 0.95, and 0.99 respectively. This methodology allows for easy comparison of the speed up obtained by the algorithm.

6.3.2 *Results*

Tables 6.5 and 6.6 show the results of ASM-MOMA and LAMMA compared to original NSGA-II and ϵ-IBEA. In all the tables NSGA is the original NSGA-II, IBEA denotes the original ϵ-IBEA. LR, SVM, and MLP stands for the model used: linear regression, support vector regression and multilayer perceptron respectively. G denotes the single global model of ASM-MOMA and L stands for the local models of LAMMA.

tables

The numbers in the table represent the median number of objective function evaluations needed to reach the specified H_{ratio} value.

Table 6.5: Median number of function evaluations needed to reach the specified H_{ratio} on ZDT1 and ZDT2 test problems

			ZDT1		
H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	5600	18600	19850	20750	21850
NSGA-LR-G	1500	2000	2400	2800	**12750**
NSGA-SVM-G	1450	2050	2350	2850	13550
NSGA-MLP-G	2100	2800	3850	4500	15200
NSGA-LR-L	**1300**	1750	2250	2600	13100
NSGA-SVM-L	1350	**1650**	**2150**	**2450**	14150
NSGA-MLP-L	1600	2100	2700	3250	15700
IBEA	7400	13750	18200	20000	25550
IBEA-LR-G	1450	2500	2800	2950	7450
IBEA-SVM-G	1400	2050	2700	3100	**6850**
IBEA-MLP-G	1800	2550	4000	4600	10100
IBEA-LR-L	**1300**	1900	2400	2750	7500
IBEA-SVM-L	1350	1900	**2350**	**2600**	7100
IBEA-MLP-L	1400	**1850**	2450	3250	9650

			ZDT2		
H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	650	1650	3550	5050	7900
NSGA-LR-G	**350**	550	750	950	1250
NSGA-SVM-G	**350**	**450**	700	1050	1750
NSGA-MLP-G	400	550	800	1000	1500
NSGA-LR-L	**350**	**450**	**600**	**850**	**1100**
NSGA-SVM-L	**350**	550	750	900	1250
NSGA-MLP-L	**350**	500	750	**850**	1250
IBEA	750	2050	5150	7800	13000
IBEA-LR-G	350	550	750	900	1650
IBEA-SVM-G	350	550	850	1050	1550
IBEA-MLP-G	450	650	950	1200	2700
IBEA-LR-L	**300**	**500**	**700**	**850**	**1350**
IBEA-SVM-L	350	550	800	1000	1450
IBEA-MLP-L	350	550	750	900	1400

Table 6.6: Median number of function evaluations needed to reach the specified H_{ratio} on ZDT3 and ZDT6 test problems

ZDT3

H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	600	1250	4150	7250	-
NSGA-LR-G	**300**	500	700	800	1150
NSGA-SVM-G	350	500	700	**750**	1100
NSGA-MLP-G	450	700	1000	1150	1750
NSGA-LR-L	**300**	**450**	**650**	800	1050
NSGA-SVM-L	350	550	700	850	**1000**
NSGA-MLP-L	350	550	850	950	1300
IBEA	650	1550	5400	8150	33350
IBEA-LR-G	**350**	550	850	950	**1300**
IBEA-SVM-G	**350**	550	850	1000	**1300**
IBEA-MLP-G	450	800	1100	1250	1800
IBEA-LR-L	**350**	**450**	**750**	**900**	**1300**
IBEA-SVM-L	400	650	850	1050	1450
IBEA-MLP-L	400	650	950	1150	1600

ZDT6

H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	7950	10200	13950	17700	28650
NSGA-LR-G	2750	5950	11100	15750	30500
NSGA-SVM-G	**2500**	**4950**	**8650**	**12500**	**23500**
NSGA-MLP-G	3300	5850	10350	14650	26800
NSGA-LR-L	2850	5850	10550	15350	29200
NSGA-SVM-L	2600	**4950**	9100	12900	25300
NSGA-MLP-L	3350	6050	10300	13950	27150
IBEA	10300	13650	18400	23150	34050
IBEA-LR-G	3050	**6500**	13400	**17600**	32100
IBEA-SVM-G	**3000**	7250	14100	19250	34150
IBEA-MLP-G	3500	7250	13250	18900	32450
IBEA-LR-L	3050	6850	13050	18750	**31400**
IBEA-SVM-L	**3000**	**6500**	**12650**	17850	32550
IBEA-MLP-L	3400	7050	13300	18200	32950

Table 6.7: The effect of different number of neurons in the hidden layer on the performance of LAMMA on the ZDT1 test problem.

Number of neurons	0.5	0.75	0.9	0.95	0.99
1	1300	2900	3150	3300	8750
3	1400	2100	2800	3250	9600
5	1450	1800	2200	2950	8500
10	1500	1900	2550	3250	9350

Twenty runs for each configuration were made. A "-" symbol means that the particular configuration was not able to attain the specified H_{ratio} within the limit of 50,000 evaluations of the objective functions.

general observations From the results, we can see that the global models significantly decrease the number of required function evaluations, and the local models are even better than the global ones. Moreover, we can see that linear regression gives better results than support vector regression and multilayer perceptrons. It probably creates simpler models which indicate the right general direction in which the local search should proceed. Moreover, the distances based models are quite easy to train and linear regression can provide better extrapolation of the values. Furthermore, we can see that the results of local models are almost always better than those of a single global model (see the following paragraphs for more detailed discussion). Comparing the differences between the chosen models (linear regression, support vector regression and multilayer perceptron) we can note that within local models these differences are smaller. This implies that the choice of the type of the model is less important when the local models are used. Following from the discussion in section 6.2, we could recommend using the faster models, i.e. linear regression or support vector regression instead of multilayer perceptrons.

different hyper-parameters To rule out the possibility of improperly set parameters of the multilayer perceptrons, we tried changing the number of neurons in the hidden layer, which is the most important parameter for this model. The results of this experiment on ZDT1 are in Table 6.7. We can see that the results of this model can indeed be improved when more attention is paid to its settings, however LAMMA with multilayer perceptrons still does not outperform the results of LAMMA with other meta-models.

specific observation On ZDT1 the global model decreased the number of function evaluations by the factor of 7.4 for the $H_{ratio} = 0.95$ (NSGA-II and linear regression), the local model decreased this number by another almost 8%, yielding a combined factor of 8. The numbers for $H_{ratio} = 0.99$ are not that good, although the number of function evaluations dropped to approximately a half with the global model and remains practically

Table 6.8: Parameters of the multi-objective algorithm

Parameter	MOEA value	Local search value
Stopping criterion	10,000 evaluations	30 generations
Population size	50	50
Crossover operator	SBX	SBX
Crossover probability	0.8	0.8
Mutation operator	Polynomial	Polynomial
Mutation probability	0.1	0.2
Archive size	400	–
Memetic operator prob.	0.25	–
Locality parameter	–	1.0 and 4.0

unchanged with the use of local models. The results for other combinations of MOEA and types of meta-models are similar on ZDT1.

On ZDT2 (again NSGA-II and linear regression), the global model reduced the required number of evaluations (to reach the $H_{ratio} = 0.99$) by the factor of 6.3 with the local model lowering the number by another 12%, yielding the overall reduction factor of 7.2. Again, the results are similar for other combinations of MOEA and type of the meta-model, the only exception being the behavior of multilayer perceptron in the combination with ϵ-IBEA, where the local model decreased the number of evaluations to 1,400 compared to the 2,700 of the global model and 13,000 of plain ϵ-IBEA yielding a reduction by the factor of 9.3.

On ZDT3, both ASM-MOMA and LAMMA were able to reach the $H_{ratio} = 0.99$ while the original NSGA-II was not. Moreover, LAMMA needed only 1000 evaluations (with support vector regression as the meta-model), ASM-MOMA needed 100 more evaluations. The original ϵ-IBEA needed over 30,000 function evaluations to attain the $H_{ratio} = 0.99$, the ASM-MOMA and LAMMA both needed only 1,300 evaluations, thus reducing the number of evaluations almost 26 times. This reduction ratio is rally large, however it is mainly caused by the fact that the external algorithm itself is not able to find such a good solution effectively.

ZDT6 difficulties

ZDT6 is the most difficult problem among those we used for comparison. Although the number of evaluations needed to reach the $H_{ratio} = 0.5$ dropped approximately to a third of the original, this difference gets lower as the H_{ratio} grows, and the results for $H_{ratio} = 0.99$ are almost identical. In this case, the local models helped to reduce the number of evaluations slightly, and for most configurations of LAMMA they were lower than those needed by the original algorithms. We believe the poor results are partially caused by premature convergence (as some preliminary tests showed that the results for higher percentage of individuals which are locally improved are even worse, and ZDT6 is multi-modal), together with the difficulty of modeling this particular function. On ZDT6 the Pareto front is biased for solutions with one of the functions close to 1, which yields training set with low diversity and that could be the reason for poorly trained models.

6.4 EXPERIMENTS – MANY-OBJECTIVE PROBLEMS

As we have shown both ASM-MOMA and LAMMA greatly reduce the number of objective function evaluations needed to find a good solution to a multi-objective problem with two objectives. However the question remains how these two algorithms would scale with respect to the number of objectives and how they will perform on many-objective problem.

The idea is that both algorithms use some kind of scalarization during the creation of the meta-model, which is similar to some of the many-objective optimization algorithms. This could provide non-dominated individuals and have similar performance without any additional objective function evaluation. The aggregate meta-models thus provide a natural hybridization of scalarization techniques with other many-objective optimization techniques (in our case with the indicator based algorithms, as we used the IBEA as the main many-objective optimizer).

In this scenario we tested only ϵ-IBEA based ASM-MOMA and LAMMA as NSGA-II is known to have poor performance in the many-objective case (Ishibuchi, Tsukamoto, et al., 2008). We also did not test the performance of multilayer perceptrons as the meta-model as the results in bi-objective case indicated they do not work well and are much slower than linear regression and support vector regression.

benchmark functions

To assess the performance of aggregate meta-models in the many-objective case we tested ASM-MOMA and LAMMA (with locality parameter $\lambda = 1$ and $\lambda = 4$) on the well-known DTLZ1 to DTLZ4 test problems with 5, 10, and 15 objectives, we used 20 variables in all cases. See Table 6.8 for the parameters of the algorithms used.

performance measure

We chose the hypervolume as the performance measure and report its values after 1,000, 2,000, 5,000 and 10,000 evaluations of the objective functions. The hypervolume is normalized in such a way that the point $\vec{0}$ has the hypervolume of 1.0. As the exact hypervolume cannot be computed effectively for this large number of objectives, we used Monte Carlo sampling to find its approximation, 100,000 samples were used. Again, 20 runs for each configuration were performed and the average values are reported. We do not use the H_{ratio} in this case, as we do not know the real hypervolume for all of the test problems. Moreover the hypervolumes differ largely for different numbers of objectives.

Although we believe that the comparison methodology used in the multi-objective case is better, as it allows direct comparison of the number of needed objective function evaluations, the complexity of the hypervolume computation for these large number of objectives makes it impractical, as it requires the evaluation of the hypervolume after each generation. Also due to the slight deviations in the computed hypervolume caused by the Monte-Carlo sampling the reported number of evaluations might have been incorrect in the case that the estimated hypervolume would be larger than the real hypervolume.

Tables 6.9 to 6.12 show the results of aggregate meta-models on many-objective optimization problems. Best results for each configuration are in italics, if the result is significantly better (one sided t-test, p-value < 0.05) than other results it is in bold and the worse result is in the parentheses (N for no model, S for support vector regression, and L for linear regression).

Generally, we can see that the aggregate meta-models improve the results compared to the plain ϵ-IBEA. The following sections discuss the results in more detail.

6.4.1 *ASM-MOMA*

general observations

With ASM-MOMA, we can see that the results were improved in almost all cases (the only exception being DTLZ1 with 5 objective functions after 1000 and 2000 function evaluations). Moreover, the results for linear regression are generally better than those where support vector regression was used as the underlying surrogate model.

specific discussion

On the DTLZ1 problem, ASM-MOMA with linear regression as the meta-model reached the best hypervolume for 5, 10 and 15 objective function after 5,000 and 10,000 objective function evaluations. However, for the configuration with only 5 objective functions it was beaten by ϵ-IBEA after 1,000 and 2,000 function evaluations. This might mean that the aggregate meta-models are able to provide new non-dominated solutions even in the later phases of the evolution. On the other hand, in the earlier phases the models are not well trained and may slow the evolution down, as they may provide wrong direction for the search.

On DTLZ2 with 5 and 10 objective function ASM-MOMA with support vector regression performed better than ASM-MOMA with linear regression as meta-model. Both performed better than ϵ-IBEA, however only support vector regression ASM-MOMA was significantly better. When the number of objectives is increased to 15, the situation changes. ASM-MOMA with linear regression works the best of the compared algorithms and is significantly better than both ϵ-IBEA and ASM-MOMA with support vector regression model. We can also note that ASM-MOMA reduces the number of evaluations: the same result

Table 6.9: Results of ASM-MOMA and LAMMA ($\lambda = 1$ and $\lambda = 4$) with linear regression (LR) and support vector regression (SVR) on the test problems. Average hypervolume attained during 20 runs. Best values are in italics, significantly better values are in bold with the appended letters denoting the variants which are significantly worse (N for no model of IBEA, S for support vector regression, and L for linear regression), only the different models for the same variant of the algorithm are compared.

DTLZ1

Dim.	Algorithm	Objective function evaluations			
		1,000	2,000	5,000	10,000
5	IBEA	*0.294*	*0.413*	0.569	0.711
	LR-ASM-MOMA	0.265	0.386	*0.580*	*0.753*
	SVR-ASM-MOMA	0.270	0.384	0.573	0.737
	LR-LAMMA-1	0.283	0.400	*0.596*	0.747
	SVR-LAMMA-1	0.260	0.383	0.595	*0.749*
	LR-LAMMA-4	0.246	0.374	0.549	0.702
	SVR-LAMMA-4	0.262	0.375	0.566	*0.731*
10	IBEA	0.244	0.271	0.304	0.325
	LR-ASM-MOMA	*0.257*	**0.305NS**	*0.318*	*0.339*
	SVR-ASM-MOMA	0.245	0.270	0.308	0.336
	LR-LAMMA-1	*0.260*	*0.291*	0.316	0.346
	SVR-LAMMA-1	0.249	0.27	*0.325*	*0.352*
	LR-LAMMA-4	*0.250*	*0.277*	0.303	0.322
	SVR-LAMMA-4	0.244	0.268	*0.311*	0.311
15	IBEA	0.193	0.210	0.224	0.235
	LR-ASM-MOMA	**0.198S**	*0.221*	**0.248S**	**0.263N**
	SVR-ASM-MOMA	0.178	0.209	0.223	0.239
	LR-LAMMA-1	*0.199*	*0.217*	*0.240*	**0.256N**
	SVR-LAMMA-1	0.193	0.212	0.226	0.240
	LR-LAMMA-4	*0.196*	0.214	0.238	0.254
	SVR-LAMMA-4	0.191	*0.222*	**0.256N**	**0.259N**

Table 6.10: Results of ASM-MOMA and LAMMA ($\lambda = 1$ and $\lambda = 4$) with linear regression (LR) and support vector regression (SVR) on the test problems. Average hypervolume attained during 20 runs. Best values are in italics, significantly better values are in bold with the appended letters denoting the variants which are significantly worse (N for no model of IBEA, S for support vector regression, and L for linear regression), only the different models for the same variant of the algorithm are compared.

DTLZ2

Dim.	Algorithm	Objective function evaluations			
		1,000	2,000	5,000	10,000
5	IBEA	0.650	0.685	0.732	0.742
	LR-ASM-MOMA	0.654	0.699	0.745	0.783
	SVR-ASM-MOMA	*0.660*	*0.720*	**0.779N**	**0.821NL**
	LR-LAMMA-1	0.640	*0.700*	**0.768N**	**0.817N**
	SVR-LAMMA-1	0.639	0.688	**0.772N**	**0.822N**
	LR-LAMMA-4	*0.653*	*0.710*	**0.773N**	**0.815N**
	SVR-LAMMA-4	0.652	0.708	**0.771N**	**0.812N**
10	IBEA	0.726	0.730	0.738	0.744
	LR-ASM-MOMA	0.732	0.739	0.748	0.756
	SVR-ASM-MOMA	*0.739*	*0.743*	*0.754*	*0.761*
	LR-LAMMA-1	*0.747*	*0.755*	*0.760*	**0.771S**
	SVR-LAMMA-1	0.727	0.735	0.740	0.745
	LR-LAMMA-4	0.726	*0.737*	*0.745*	*0.748*
	SVR-LAMMA-4	0.717	0.725	0.737	0.741
15	IBEA	0.83	0.832	0.836	0.836
	LR-ASM-MOMA	**0.849NS**	**0.856NS**	**0.864NS**	**0.868NS**
	SVR-ASM-MOMA	0.836	0.837	0.837	0.839
	LR-LAMMA-1	0.845	0.846	**0.854N**	**0.859N**
	SVR-LAMMA-1	**0.853N**	**0.853N**	**0.860N**	**0.862N**
	LR-LAMMA-4	**0.851N**	**0.851N**	**0.857N**	**0.863N**
	SVR-LAMMA-4	0.840	0.844	0.846	0.852

Table 6.11: Results of ASM-MOMA and LAMMA ($\lambda = 1$ and $\lambda = 4$) with linear regression (LR) and support vector regression (SVR) on the test problems. Average hypervolume attained during 20 runs. Best values are in italics, significantly better values are in bold with the appended letters denoting the variants which are significantly worse (N for no model of IBEA, S for support vector regression, and L for linear regression), only the different models for the same variant of the algorithm are compared.

DTLZ3

Dim.	Algorithm	Objective function evaluations			
		1,000	2,000	5,000	10,000
5	IBEA	0.732	0.79	0.839	0.872
	LR-ASM-MOMA	*0.752*	*0.799*	*0.860*	*0.888*
	SVR-ASM-MOMA	0.738	0.788	0.84	0.867
	LR-LAMMA-1	0.734	0.78	0.834	0.865
	SVR-LAMMA-1	*0.752*	*0.798*	*0.854*	*0.883*
	LR-LAMMA-4	0.734	0.784	0.828	0.866
	SVR-LAMMA-4	*0.756*	*0.804*	*0.845*	*0.872*
10	IBEA	0.519	0.519	0.527	0.531
	LR-ASM-MOMA	**0.570N**	**0.572N**	**0.581NS**	**0.585N**
	SVR-ASM-MOMA	0.545	0.549	0.548	0.555
	LR-LAMMA-1	0.542	**0.551N**	**0.560N**	**0.565N**
	SVR-LAMMA-1	*0.546*	0.55	0.56	**0.566N**
	LR-LAMMA-4	*0.540*	*0.549*	*0.556*	*0.562*
	SVR-LAMMA-4	0.525	0.537	0.547	0.55
15	IBEA	0.447	0.452	0.457	0.458
	LR-ASM-MOMA	*0.458*	*0.463*	0.466	0.469
	SVR-ASM-MOMA	0.458	0.462	*0.468*	*0.474*
	LR-LAMMA-1	0.464	**0.472N**	**0.477N**	*0.477*
	SVR-LAMMA-1	*0.467*	0.472	0.473	0.47
	LR-LAMMA-4	**0.479N**	**0.484N**	**0.489N**	**0.491N**
	SVR-LAMMA-4	**0.477N**	**0.480N**	**0.490N**	**0.491N**

Table 6.12: Results of ASM-MOMA and LAMMA ($\lambda = 1$ and $\lambda = 4$) with linear regression (LR) and support vector regression (SVR) on the test problems. Average hypervolume attained during 20 runs. Best values are in italics, significantly better values are in bold with the appended letters denoting the variants which are significantly worse (N for no model of IBEA, S for support vector regression, and L for linear regression), only the different models for the same variant of the algorithm are compared.

DTLZ4

Dim.	Algorithm	Objective function evaluations			
		1,000	2,000	5,000	10,000
5	IBEA	0.092	0.147	0.184	0.208
	LR-ASM-MOMA	**0.172N**	**0.290N**	**0.386N**	**0.421N**
	SVR-ASM-MOMA	**0.202N**	**0.347N**	**0.449N**	**0.484N**
	LR-LAMMA-1	**0.217N**	**0.328N**	**0.428N**	**0.470N**
	SVR-LAMMA-1	**0.230N**	**0.346N**	**0.431N**	**0.460N**
	LR-LAMMA-4	**0.169N**	**0.289N**	**0.380N**	**0.407N**
	SVR-LAMMA-4	**0.166N**	**0.272N**	**0.388N**	**0.434N**
10	IBEA	0.557	0.558	0.559	0.559
	LR-ASM-MOMA	**0.626N**	**0.637N**	**0.637N**	**0.636N**
	SVR-ASM-MOMA	**0.610N**	**0.625N**	**0.627N**	**0.626N**
	LR-LAMMA-1	**0.627NS**	**0.640NS**	**0.641NS**	**0.642NS**
	SVR-LAMMA-1	**0.597N**	**0.605N**	**0.605N**	**0.604N**
	LR-LAMMA-4	**0.613N**	**0.618N**	**0.621N**	**0.621N**
	SVR-LAMMA-4	**0.599N**	**0.617N**	**0.642N**	*0.674*
15	IBEA	0.96	0.961	0.962	0.962
	LR-ASM-MOMA	**0.973N**	**0.973N**	**0.973N**	**0.973N**
	SVR-ASM-MOMA	**0.973N**	**0.973N**	**0.973N**	**0.973N**
	LR-LAMMA-1	**0.974N**	**0.974N**	**0.974N**	**0.974N**
	SVR-LAMMA-1	**0.972N**	**0.972N**	**0.972N**	**0.972N**
	LR-LAMMA-4	**0.976N**	**0.976N**	**0.975N**	**0.974N**
	SVR-LAMMA-4	**0.974N**	**0.976N**	**0.976N**	**0.976N**

ϵ-IBEA reached after 10,000 evaluations, was reached by ASM-MOMA after 5,000 evaluations in the 5 objective case, 2,000 evaluations in the 10 objective case, and only 1,000 evaluations in the 15 objective case.

Linear regression seems to be the best model for DTLZ3 also, for almost all of the configurations (except 15 objectives after 5,000 and 10,000 generations, where support vector regression wins). For the configuration with 10 objectives, the linear regression ASM-MOMA provides significantly better results than ϵ-IBEA. Moreover, we can notice that even after 1,000 evaluation the ASM-MOMA already has better result than ϵ-IBEA after 10,000 evaluations, thus reducing the required number of evaluations more than 10 times.

On DTLZ4, ASM-MOMA with either model is significantly better than ϵ-IBEA in all cases. Similarly to DTLZ2, support vector regression works better in the 5 objective case and linear regression is slightly better for higher number of objectives. Moreover, in this case the performance of ϵ-IBEA after 10,000 evaluations is reached by ASM-MOMA after only 1,000 evaluations in all cases. The number of objective function evaluations is again decreased more than 10 times.

6.4.2 *LAMMA*

The results for LAMMA on the selected benchmark functions are similar to those of ASM-MOMA.

On DTLZ1 with 5 objectives the convergence is even slower than for ASM-MOMA, but the results after 10,000 evaluations are almost the same. The convergence slows down even more, when more local ($\lambda = 4$) are used. In this case locality of the models does not help and a single global meta-model is better. When the number of objectives increases the differences tend to get smaller. LAMMA provides less significant improvements than ASM-MOMA on this test problems.

On DTLZ2 we see more significant results with LAMMA, compared to ASM-MOMA. Also the differences between the two types of meta-models are lower both for $\lambda = 1$ and $\lambda = 4$. LAMMA provides similar speed-ups (in the terms of the function evaluations) as ASM-MOMA (5 to 10 times in this case).

On DTLZ3 LAMMA again provides better results than ϵ-IBEA. In the case of 15 objectives, the results are even significantly better regardless of the meta-model used (for $\lambda = 4$).

On DTLZ4 we again see the significant improvements we observed with ASM-MOMA in all situations and cases. In this case LAMMA with $\lambda = 1$ provides the best results. The speed ups are again more than 10 times, as LAMMA after 1,000 evaluations reaches better hypervolumes than those obtained by ϵ-IBEA after 10,000 evaluations.

6.5 CONCLUSION

In this chapter we presented a memetic evolutionary algorithm for multi-objective optimization with local distance-based aggregate surrogate model. We showed that the local models give better results than a single global model, usually reducing the number of needed function evaluations by 10%, with occasional reductions over 40%. Although this difference may seem rather small, it may greatly reduce the associated costs in practical tasks.

We also showed that the algorithm is usable even for problems with quite simple objective functions, which take only milliseconds to evaluate, thus making it more widely usable. In fact, ASM-MOMA could be used even in the case of some of the ZDT benchmarks (namely ZDT3), as the reduction was so large, that the $H_{ratio} = 0.99$ was reached faster (in terms of time) by ASM-MOMA that NSGA-II.

However, we saw that some problems are still difficult to solve with LAMMA, and these provide the motivation for further research. The question is, how well the benchmark problems correspond to the real-life ones, and also how to decide, whether a given problem belongs to a class of problems which can be easily solved by the presented evolutionary algorithm.

We have also shown that aggregate meta-models can be used to speed up the search of evolutionary algorithms for many-objective optimization problems. These meta-models are able to provide new non-dominated individuals and thus speed up the search.

The use of aggregate meta-models in many-objective optimization can also be seen as a combination of two approaches: indicator based algorithm and scalarization. In this case the scalarization is incorporated as a memetic operator and does not use new objective function evaluations, however it still provides new non-dominated individuals which can be used by the main many-objective algorithm.

The results indicate that using aggregate meta-models might be a promising direction in the field of many-objective optimization, however some questions remain. One of them is, how would these approaches scale with the number of variables, as this is what directly affects the dimension of the space in which the model is built.

7 PRE-SELECTION IN ASM-MOMA

In this chapter, we continue to investigate ASM-MOMA further. One of the areas, where the algorithm can be further improved is the selection of individuals for evaluation by the real objective function. If the individuals which are not prospective are not evaluated at all, the performance of the algorithm can be further increased. However, we will see that one must proceed very carefully when selecting such individuals in order to avoid the pre-mature convergence of the algorithm. The new selection scheme is derived in two steps – first, we show that selecting one best individual improves the performance of the optimizer, and than we provide a (different) way to select more than one individual in each generation.

motivation

The motivation for pre-selection comes from the fact that offspring are generated randomly, and as such there is a certain probability that the new offspring will be worse than its parents. If we were able to detect these offspring before they are evaluated by the real expensive objective function, we might be able to reduce the number of evaluations further.

Another motivation comes from the fact that steady state algorithms (i.e. algorithms which produce only one individual per generation) are often used for optimization of expensive objectives – the faster feedback from the fitness function leads to a more effective search.

steady-state ASM-MOMA

Inspired by the steady-state algorithms we first propose a steady-state variant of ASM-MOMA. This variant is a simple augmentation of the original ASM-MOMA. The only change is the addition of a new pre-selection operator right after the local search operator. The pre-selection operator evaluates all the individuals using a surrogate model and selects only the best one (according to the model). For the first experiments, we use the same model we used during the local search phase. Originally (Pilát and Neruda, 2012), we called this algorithm with this kind of pre-selection “surrogate based multi-objective evolution strategy” (SBMO-ES). The term “evolution strategy” is used in a more general manner in this case, as the only thing that the algorithm has in common with evolution strategies is the $(\lambda + 1)$ selection scheme.

problem of SBMO-ES

The main problem of SBMO-ES is the large overhead all the modeling and local searches have. The worst part is that most of the work done during one generation of the evolutionary algorithm is dropped as only one of the generated individuals is selected and evaluated. We may do a similar discussion of the modeling overhead we did in the previous chapter and as long as there is a reduction in the number of evaluations SBMO-ES may still be more effective than ASM-MOMA for sufficiently expensive objectives. But the large overhead makes SBMO-ES less useful in situations where the objectives are not so expensive and it does not provide the speed up for relatively cheap problems like ASM-MOMA does.

7.1 PRE-SELECTION IN ASM-MOMA

algorithm description

The surrogate-based multi-objective evolution strategy (SBMO-ES) is a quite simple modification of LAMMA. The main difference is the additional pre-selection step, which trains a global surrogate model (model with the locality parameter $\lambda = 0$) and uses it to select the best offspring from those newly generated. Only this offspring is then evaluated and added to the population. This pre-selection step is executed after the new individuals are generated using the local search step in LAMMA and before the environmental selection of the external algorithm is performed.

The pre-selection step ensures that poor individuals are never evaluated and as such should improve the convergence speed. The main disadvantage is that only one individual is selected in each generation which makes the parallelization of this algorithm more complicated. Also, the overhead of the new algorithm is much larger, as after each generation only one individual is created instead of multiple individuals created by ASM-MOMA and LAMMA.

7.2 EXPERIMENTS – ONE PRE-SELECTED INDIVIDUAL

comparison

To assess the performance of SBMO-ES we compare it to two algorithms, which we presented earlier, namely LAMMA and ASM-MOMA and to plain versions of NSGA-II (Deb, Pratap, et al., 2002) and ϵ-IBEA (Zitzler and Künzli, 2004). Different configurations are tested – we used NSGA-II and ϵ-IBEA as the external evolutionary algorithms, both of them use the Simulated Binary Crossover (SBX) (Deb and Agrawal, 1994) and Polynomial Mutation (PM) (Deb and Goyal, 1996) as their genetic operators.

model configuration

We tested linear regression and support vector regression as the surrogate models used during the local search phase and the selection procedure. The local and global models were based on the same model in all cases (i.e. they both used linear regression, or they both used support vector regression). The support vector regression uses polynomial kernels with the degree 1, i.e. the dot product of the two vectors. The parameters of the methods are the same as those we used in the previous chapter, but for the sake of self-containedness of this chapter we repeat them here (Table 7.1). We again use the H_{ratio} metric we defined in the previous chapter to make the comparison.

Table 7.1: The parameters of the method

Parameter	External EA	Internal EA
Termination criterion	50,000 evaluations	30 generations
Population size	50	50
Crossover probability	0.8	0.8
Crossover operator	SBX	SBX
Mutation probability	0.1	0.2
Mutation operator	PM	PM
Memetic operator prob.	0.25	-
Maximum archive size	400	-
Model locality parameter λ		1.0

benchmark problems

The algorithm was tested on selected functions from the ZDT (Zitzler, Deb, et al., 2000) benchmark set, namely ZDT1 with 30 variables, and ZDT2, ZDT3, and ZDT6 with 15 variables. The median number of function evaluations needed to attain the specified H_{ratio} is reported, 20 independent runs were made for each of the configurations. The results for ASM-MOMA and LAMMA are also repeated here to provide comparison for the newly developed method.

7.2.1 *General observations*

general results

The results indicate that SBMO-ES is generally able to outperform both ASM-MOMA and LAMMA. In most cases even the worst variant of SBMO-ES is better, or comparable with, the best variant of ASM-MOMA and LAMMA. When the same variants (i.e. the same external algorithm and type of model) are compared, the evolutionary strategy based one is better in all cases except on ZDT3 with NSGA-II as external algorithm and SVR as the surrogate model. However, even in this case, the results are not significantly worse.

Generally, the version which uses NSGA-II as the external algorithm tends to show smaller improvements. There is almost none improvement on ZDT2 and ZDT3 test problems, where SVR is used as the surrogate model. This may be caused by the inability of the NSGA-II selection procedure to provide good spread of solutions and therefore it is stuck in some local optima. ϵ-IBEA seems not to have such problems and the improvements tend to be quite large.

The best improvement can be found on ZDT1 with NSGA-II as external algorithm and SVR as the surrogate model, here, the number of evaluations dropped to less than a half of those needed by ASM-MOMA, the second best configuration. Another reduction to approximately a half of the original can be observed for ϵ-IBEA on the ZDT2 (both types of surrogate models) and ZDT3 (SVR) problems. On ZDT6,

Table 7.2: Median number (20 runs) of function evaluations needed to reach the specified H_{ratio} on ZDT1 and ZDT2 test problems. The best values are in bold. The name of the method is encoded as follows: NSGA and IBEA indicate the type of the external evolutionary algorithm (NSGA-II and ϵ-IBEA respectively), LR and SVM stand for the type of the surrogate model used (linear regression and support vector regression), G, L, and ES indicate the algorithm (ASM-MOMA, LAMMA and SBMO-ES).

ZDT1

H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA-LR-G	1500	2000	2400	2800	12750
NSGA-SVM-G	1450	2050	2350	2850	13550
NSGA-LR-L	1300	1750	2250	2600	13100
NSGA-SVM-L	1350	1650	2150	2450	14150
NSGA-LR-ES	**949**	**1293**	**1692**	1985	5097
NSGA-SVM-ES	1019	1442	1479	**1751**	**4551**
IBEA-LR-G	1450	2500	2800	2950	7450
IBEA-SVM-G	1400	2050	2700	3100	6850
IBEA-LR-L	1300	1900	2400	2750	7500
IBEA-SVM-L	1350	1900	2350	2600	7100
IBEA-LR-ES	**798**	**1197**	**1456**	**1759**	5639
IBEA-SVM-ES	1115	1474	1723	1813	**5072**

ZDT2

H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA-LR-G	350	550	750	950	1250
NSGA-SVM-G	350	450	700	1050	1750
NSGA-LR-L	350	450	600	850	1100
NSGA-SVM-L	350	550	750	900	1250
NSGA-LR-ES	**156**	**257**	**367**	**719**	**916**
NSGA-SVM-ES	206	294	733	809	1178
IBEA-LR-G	350	550	750	900	1650
IBEA-SVM-G	350	550	850	1050	1550
IBEA-LR-L	300	500	700	850	1350
IBEA-SVM-L	350	550	800	1000	1450
IBEA-LR-ES	**150**	**246**	380	486	788
IBEA-SVM-ES	153	251	**314**	**394**	**678**

Table 7.3: Median number (20 runs) of function evaluations needed to reach the specified H_{ratio} on ZDT3 and ZDT6 test problems. The best values are in bold. The name of the method is encoded as follows: NSGA and IBEA indicate the type of the external evolutionary algorithm (NSGA-II and ϵ-IBEA respectively), LR and SVM stand for the type of the surrogate model used (linear regression and support vector regression), G, L, and ES indicate the algorithm (ASM-MOMA, LAMMA and SBMO-ES respectively).

ZDT3

H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA-LR-G	300	500	700	**800**	1150
NSGA-SVM-G	350	500	700	750	1100
NSGA-LR-L	300	450	650	**800**	1050
NSGA-SVM-L	350	550	700	850	1000
NSGA-LR-ES	166	295	**631**	831	**831**
NSGA-SVM-ES	**165**	**293**	744	857	1064
IBEA-LR-G	350	550	850	950	1300
IBEA-SVM-G	350	550	850	1000	1300
IBEA-LR-L	350	450	750	900	1300
IBEA-SVM-L	400	650	850	1050	1450
IBEA-LR-ES	187	272	450	553	901
IBEA-SVM-ES	**172**	**259**	**422**	**536**	**722**

ZDT6

H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA-LR-G	2750	5950	11100	15750	30500
NSGA-SVM-G	2500	4950	8650	12500	23500
NSGA-LR-L	2850	5850	10550	15350	29200
NSGA-SVM-L	2600	4950	9100	12900	25300
NSGA-LR-ES	1348	3096	6558	9623	19581
NSGA-SVM-ES	**1327**	**3021**	**5652**	**8026**	**17284**
IBEA-LR-G	3050	6500	13400	17600	32100
IBEA-SVM-G	3000	7250	14100	19250	34150
IBEA-LR-L	3050	6850	13050	18750	31400
IBEA-SVM-L	3000	6500	12650	17850	32550
IBEA-LR-ES	**1629**	**4059**	**8468**	**12170**	**21816**
IBEA-SVM-ES	1651	4477	8824	12498	23515

which was the hardest problem for ASM-MOMA and LAMMA, the number of evaluations dropped by 20% for NSGA-II based external algorithm and by a third for ϵ-IBEA based one.

recommendation

In almost all cases support vector regression provides better models than linear regression. As the training and evaluation time for both these models is similar, SVR seems to be better and can be recommended as the first choice for unknown problems, especially in combination with ϵ-IBEA.

7.2.2 *Detailed analysis*

On ZDT1, LAMMA and ASM-MOMA show differences between NSGA-II and ϵ-IBEA based algorithms. However, this difference is much smaller when SBMO-ES is used. Both variants need about 5,000 function evaluations to attain the H_{ratio} of 0.99. For comparison: this value is 4-5 times lower than what plain NSGA-II or ϵ-IBEA needs.

ZDT2 shows two different results: for NSGA-II based SBMO-ES there is almost none reduction and the new evolutionary algorithm is comparable to LAMMA. For ϵ-IBEA based SBMO-ES, we observe the reduction to approximately a half. In this case, the crowding distance used in the NSGA-II selection procedure was probably not able to provide sufficient diversity in the population and thus slowed down the search. Compared to plain NSGA-II and ϵ-IBEA, SBMO-ES uses approximately 8 and 20 times lower number of function evaluations respectively.

ZDT3 is similar to ZDT2 regarding the results: almost no improvement for NSGA-II based algorithm and significant improvement for ϵ-IBEA based one. However, we can see quite a large difference between SVR based and linear regression based SBMO-ES with NSGA-II selection procedure. This is another clue, that NSGA-II selection procedure might suffer from convergence to local minima or poor spread of the individuals – the SVR model should have lower error than the linear regression and thus should guide the search more aggressively towards a local Pareto front. NSGA-II selection can deal with the pressure of linear regression, however, it has problems with the support vector regression. Compared to plain ϵ-IBEA, SBMO-ES uses 46 times lower number of objective function evaluations.

improvement on ZDT6

ZDT6 has been the most difficult problem both for ASM-MOMA and LAMMA. Regarding the number of function evaluation it is still the most difficult even for SBMO-ES, however SBMO-ES is able to reduce the number of evaluations significantly compared to both ASM-MOMA and LAMMA. The two latter algorithms had problems to even decrease the number of evaluations compared to plain NSGA-II and ϵ-IBEA. SBMO-ES can decrease the number of evaluations by almost 25% with NSGA-II and almost a third with ϵ-IBEA (compared to best configurations of

Table 7.4: Comparison of SBMO-ES and its version with 5 individuals selected in each generation (SMBO-ES-5). Median number (20 runs) of function evaluations needed to reach the specified H_{ratio} on ZDT1, ZDT2, ZDT3 and ZDT6 test problems. NSGA-II was used as the external algorithm and linear regression as the meta-model.

ZDT1

H_{ratio}	0.5	0.75	0.9	0.95	0.99
SBMO-ES	949	1293	1692	1985	5097
SBMO-ES (5)	2210	2590	2850	2990	4095

ZDT2

H_{ratio}	0.5	0.75	0.9	0.95	0.99
SBMO-ES	156	257	367	719	916
SBMO-ES (5)	110	1050	1250	1955	2745

ZDT3

H_{ratio}	0.5	0.75	0.9	0.95	0.99
SBMO-ES	166	295	631	831	831
SBMO-ES (5)	125	765	1420	2165	7475

ZDT6

H_{ratio}	0.5	0.75	0.9	0.95	0.99
SBMO-ES	1348	3096	6558	9623	19581
SBMO-ES (5)	2780	8735	-	-	-

LAMMA and ASM-MOMA). Compared to plain NSGA-II and ASM-MOMA, the reduction is approximately by one third in both cases.

7.3 PRE-SELECTION OF MORE INDIVIDUALS

We have seen that the pre-selection of a single individual improves the performance of LAMMA. However, the overhead of such an algorithm is large. Moreover, such an algorithm is hard to parallelize as the evaluations of the real objectives need to be done by one. Additionally, in practice, it is sometimes possible to evaluate more individuals at once for a cost which is lower than evaluating each of the individuals by one. Therefore, our next goal is to provide a pre-selection scheme, which would ensure similar performance to the one described above, but would also provide more individuals in each generation. This would also reduce the overhead of the evolution (more precisely, it would spread the overhead over more individuals) and thus make the algorithm usable in cases with less expensive objectives. Therefore, our goal for this section is to find a pre-selection scheme which would provide more individuals in each generation and have at the same time at least the same performance as SBMO-ES.

towards better pre-selection

Table 7.5: Comparison of SBMO-ES and its version with K-Means to divide the non-dominated individuals (SBMO-ES-K). Median number (20 runs) of function evaluations needed to reach the specified H_{ratio} on ZDT1, ZDT2, ZDT3 and ZDT6 test problems. ϵ-IBEA was used as the external algorithm and linear regression as the meta-model.

ZDT1

H_{ratio}	0.5	0.75	0.9	0.95	0.99
SBMO-ES	798	1197	1456	1759	5639
SBMO-ES-K	868	5561	7390	9871	-

ZDT2

H_{ratio}	0.5	0.75	0.9	0.95	0.99
SBMO-ES	150	246	380	486	788
SBMO-ES-K	271	352	459	557	1214

ZDT3

H_{ratio}	0.5	0.75	0.9	0.95	0.99
SBMO-ES	187	272	450	553	901
SBMO-ES-K	289	469	739	1180	2311

ZDT6

H_{ratio}	0.5	0.75	0.9	0.95	0.99
SBMO-ES	1629	4059	8468	12170	21816
SBMO-ES-K	66823	-	-	-	-

simple modifications fail

The easiest way how to change SBMO-ES to provide more than one individual per generation would be to simply select more individuals in the pre-selection phase. Although this provides more individuals per generation, the experiments show that it also requires larger number of objective function evaluations. The results of preliminary experiments with such algorithm are presented in Table 7.4. In these experiments, the memetic phase is the same as described above, and the pre-selection phase selects five best individuals as evaluated by the same model, which was used during the local search.

The results show that in most situations the algorithm required more evaluations, the only exception being the ZDT1 task (Zitzler, Deb, et al., 2000), where the increased number of selected individuals in fact led to a slight decrease in the number of evaluations. By studying the algorithm more carefully we noticed that most of the selected individuals are close together, which decreases the diversity in the population and in the archive, which in turn leads to poorly trained surrogate models. Moreover, as the training set does not change much between the two consecutive generations, the model is also likely very similar and this may generate a lot of similar individuals as the algorithm proceeds.

attempts with clustering

With this information in mind, we tried to make the algorithm to create a more diverse set of offspring after the local search phase. To this end we decided to use the k-means algorithm to cluster the non-dominated front of the archive, and to optimize the individuals towards each of the clusters separately (we computed the distance to the cluster instead of the whole non-dominated set during the creation of the training set). We set the algorithm to create 5 clusters, and trained the model. According to this model we selected 1/5 of the best offspring created by the operators of the external algorithm and we run the local search phase.

The preliminary results of this version are presented in Table 7.5. We can see that the results are again not better than those of SBMO-ES with the same configuration, especially in the case of the ZDT6 problem the algorithm suffered from premature convergence and was not able to converge to the global Pareto set at all.

different approaches

We also tried other approaches which used clustering in different ways, with the goal to increase the diversity of the population and the training set for the meta-models. As an example, we tried to cluster the set of generated individuals and select only one from each cluster. We also used different clustering techniques, tried different population sizes etc. but neither of these approaches led to better convergence than the one presented above as the new algorithm. In all cases the results of the other attempts were significantly worse at least on some of the test problems. We do not present these results here due to space reasons.

important message

The important message here is that the mutual effects of the local search phase and the pre-selection phase are more complicated than what was originally expected, and changing the pre-selection can have dramatic effects on the performance of the algorithm as a whole.

7.3.1 *Different meta-model for pre-selection*

The problem described above indicates that a different model should be used in the local search and pre-selection phases. To this end, we propose a pre-selection based on surrogate models trained for each of the objectives individually. The resulting algorithm is called *multi-objective evolutionary algorithm with local search and pre-selection* (LSPS-MOEA).

new pre-selection operator

The new pre-selection operator (see Algorithm 7.1) thus creates a surrogate model for each of the objectives and selects only the offspring which are not dominated by any of the parents. There is also a condition which further reduces the strength of the pre-selection: if only a small number of individuals is selected, some of the offspring

Algorithm 7.1 The pre-selection algorithm

```
Require: The meta-model type
Require: The population of parents $P$
Require: The population of pre-offspring $O$
Require: The archive of evaluated individuals $A$
  $T_i$ := the training set for objective $i$
  $M_i$ := the meta-model trained on set $T_i$
  $S$ := initialize empty set of pre-selected individuals
  for all pre-offspring $PO \in O$ do
    if $PO$ is not dominated by any parent in $P$ according to models
    $M_i$ then
      $S := S \cup PO$
  if size of $S$ is less than $m$ then
    Add $a$ randomly selected pre-offspring from $O$ to $S$
  return The set of pre-selected individuals $S$
```

are added randomly (in our test we added 10% of the population randomly if less than 20% of the population was selected, these numbers were chosen after a series of preliminary tests). There may be at least two reasons why only a few individuals are selected: either the model is trained poorly (e.g. because it is biased to a part of the search space as there are a lot of individuals in the archive in this region), or the algorithm has found a local minimum. In both of these cases increasing the diversity of the population (and also of the archive and training set) should improve the performance. These are also the reasons for such a rule in the pre-selection step.

The pre-selection described here is similar to the one used by Emmerich and Naujoks (2004), however, in our algorithm we only select the non-dominated individuals while they always selected a given number of individuals according to non-dominated sorting.

The surrogate model is again based on existing regression models (e.g. support vector regression or linear regression). The training set for these models directly maps the variables of the individuals on the values of the objective function. Thus, the training set T_i for the i-th objective contains pair $\langle x, f_i(x)\rangle$ for each individual x from the archive.

training set

7.4 EXPERIMENTS – MORE PRE-SELECTED INDIVIDUALS

The performance of the new pre-selection scheme was again compared to SBMO-ES on the ZDT (Zitzler, Deb, et al., 2000) test suite. We compared different types of models in both local search and pre-selection and also different combinations of these types of models. The settings of the algorithm and the models match those we used for SBMO-ES.

Table 7.6: Median number (20 runs) of function evaluations needed to reach the specified H_{ratio} on ZDT1 test problem. The best values are in bold. The name of the method is encoded as follows: NSGA and IBEA indicate the type of the external evolutionary algorithm (NSGA-II and ϵ-IBEA respectively), LR, SVM, and MLP stand for the type of the surrogate model used (linear regression, support vector regression, and multilayer perceptron), ES indicate the SBMO-ES, and two surrogate models indicate the two models used in the proposed algorithm.

ZDT1

H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	5600	18600	19850	20750	21850
NSGA-LR-ES	949	**1293**	1692	1985	5097
NSGA-SVM-ES	1019	1442	**1479**	**1751**	**4551**
NSGA-LR-LR	603	3897	3972	4143	8465
NSGA-LR-SVM	**500**	1420	1904	2053	5285
NSGA-LR-MLP	867	1896	2380	3101	5512
NSGA-SVM-LR	809	1794	2173	2591	5300
NSGA-SVM-SVM	617	1565	2152	2355	5639
NSGA-SVM-MLP	861	2266	3284	3494	5821
IBEA	7400	13750	18200	20000	25550
IBEA-LR-ES	798	**1197**	**1456**	**1759**	5639
IBEA-SVM-ES	1115	1474	1723	1813	5072
IBEA-LR-LR	606	2234	2343	2414	**3588**
IBEA-LR-SVM	**453**	3815	4367	4625	5694
IBEA-LR-MLP	613	1340	1607	2025	3884
IBEA-SVM-LR	587	2180	2695	2886	5230
IBEA-SVM-SVM	470	2206	2484	2601	3798
IBEA-SVM-MLP	801	1542	2748	3371	4125

Table 7.7: Median number (20 runs) of function evaluations needed to reach the specified H_{ratio} on the ZDT2 test problem. The best values are in bold. The name of the method is encoded as follows: NSGA and IBEA indicate the type of the external evolutionary algorithm (NSGA-II and ϵ-IBEA respectively), LR, SVM, and MLP stand for the type of the surrogate model used (linear regression, support vector regression, and multilayer perceptron), ES indicate the SBMO-ES, and two surrogate models indicate the two models used in the proposed algorithm.

ZDT2

H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	650	1650	3550	5050	7900
NSGA-LR-ES	156	257	367	719	916
NSGA-SVM-ES	206	294	733	809	1178
NSGA-LR-LR	**145**	**196**	293	388	630
NSGA-LR-SVM	162	217	**274**	**320**	**517**
NSGA-LR-MLP	177	234	348	718	1610
NSGA-SVM-LR	150	201	299	342	617
NSGA-SVM-SVM	156	207	259	424	719
NSGA-SVM-MLP	177	246	359	377	573
IBEA	750	2050	5150	7800	13000
IBEA-LR-ES	**150**	246	380	486	788
IBEA-SVM-ES	153	251	314	394	678
IBEA-LR-LR	151	**204**	275	332	620
IBEA-LR-SVM	153	211	267	312	**522**
IBEA-LR-MLP	177	246	359	377	573
IBEA-SVM-LR	151	214	262	**295**	603
IBEA-SVM-SVM	153	212	275	324	543
IBEA-SVM-MLP	151	216	**261**	319	470

Table 7.8: Median number (20 runs) of function evaluations needed to reach the specified H_{ratio} on the ZDT3 test problem. The best values are in bold. The name of the method is encoded as follows: NSGA and IBEA indicate the type of the external evolutionary algorithm (NSGA-II and ϵ-IBEA respectively), LR, SVM, and MLP stand for the type of the surrogate model used (linear regression, support vector regression, and multilayer perceptron), ES indicate the SBMO-ES, and two surrogate models indicate the two models used in the proposed algorithm.

ZDT3

H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	600	1250	4150	7250	-
NSGA-LR-ES	166	295	631	831	831
NSGA-SVM-ES	165	293	744	857	1064
NSGA-LR-LR	159	259	340	428	727
NSGA-LR-SVM	**151**	**223**	**317**	**379**	699
NSGA-LR-MLP	183	259	439	585	681
NSGA-SVM-LR	167	251	344	430	561
NSGA-SVM-SVM	165	250	340	382	**552**
NSGA-SVM-MLP	201	270	434	681	869
IBEA	650	1550	5400	8150	33350
IBEA-LR-ES	187	272	450	553	901
IBEA-SVM-ES	172	259	422	536	722
IBEA-LR-LR	159	223	292	367	478
IBEA-LR-SVM	156	**209**	318	367	488
IBEA-LR-MLP	193	295	375	416	562
IBEA-SVM-LR	**155**	211	288	**328**	531
IBEA-SVM-SVM	162	223	**283**	358	**452**
IBEA-SVM-MLP	178	262	371	424	592

Table 7.9: Median number (20 runs) of function evaluations needed to reach the specified H_{ratio} on the ZDT6 test problem. The best values are in bold. The name of the method is encoded as follows: NSGA and IBEA indicate the type of the external evolutionary algorithm (NSGA-II and ϵ-IBEA respectively), LR, SVM, and MLP stand for the type of the surrogate model used (linear regression, support vector regression, and multilayer perceptron), ES indicate the SBMO-ES, and two surrogate models indicate the two models used in the proposed algorithm.

ZDT6

H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	7950	10200	13950	17700	28650
NSGA-LR-ES	1348	3096	6558	9623	19581
NSGA-SVM-ES	**1327**	**3021**	**5652**	**8026**	**17284**
NSGA-LR-LR	1883	5897	10774	14714	28126
NSGA-LR-SVM	2946	5610	9747	13856	24635
NSGA-LR-MLP	1526	3646	6913	9750	19865
NSGA-SVM-LR	2076	6907	11936	14428	36085
NSGA-SVM-SVM	2122	7136	10688	13331	24859
NSGA-SVM-MLP	1429	3164	6399	9632	19046
IBEA	10300	13650	18400	23150	34050
IBEA-LR-ES	1629	4059	8468	12170	21816
IBEA-SVM-ES	1651	4477	8824	12498	23515
IBEA-LR-LR	2247	5119	10904	15140	29474
IBEA-LR-SVM	2298	4606	8961	11745	21844
IBEA-LR-MLP	**1444**	**3509**	7446	**10543**	**18297**
IBEA-SVM-LR	1999	5769	11471	15269	27479
IBEA-SVM-SVM	2012	6073	10778	14619	23034
IBEA-SVM-MLP	1634	3717	**7233**	10823	18894

Table 7.6 shows the results of our algorithm compared to original NSGA-II (Deb, Pratap, et al., 2002), and ϵ-IBEA (Zitzler and Künzli, 2004), and to SBMO-ES.

table headers To shorten the headers of the table the names of the various tested algorithms and their variants are encoded as follows: the models used in the comparison are designated as "LR", "SVM", and "MLP" for linear regression, support vector regression and multilayer perceptron respectively. The original versions of NSGA-II and ϵ-IBEA are denoted as "NSGA" and "IBEA" respectively. **SMBO-ES!** (**SMBO-ES!**) is distinguished by the "ES" suffix, preceded by the type of surrogate model and the type of selection. The variations of the proposed algorithm are expressed as the type of selection followed by the type of local search model and the type of pre-selection model (i.e. IBEA-LR-SVM is the new algorithm with ϵ-IBEA selection, linear regression as the local search model and support vector regression as the pre-selection model).

The numbers in the table represent the median number of objective function evaluations needed to reach the specified H_{ratio} value. Twenty runs for each configuration were made. A "-" symbol means that the particular configuration was not able to attain the specified H_{ratio}.

7.4.1 *General observations*

general observations Generally, the results indicate that the proposed algorithm works at least as well as SBMO-ES while producing more individuals per generation and thus allowing for easy parallelization of the evaluations of the real and costly objective functions as well as reducing the overhead of modeling.

NSGA-II vs. ε-IBEA Another observation is that ϵ-IBEA selection seems to work better in this context than NSGA-II selection. This may be the case that NSGA-II selection in unable to deal with the large number of new non-dominated individuals and the discriminative power of the non-dominated sorting diminishes. Thus, the selection is not able to effectively select good individuals, and the only criterion which guides the search is the crowding distance. This is similar to the problems NSGA-II has with tasks with more objective functions, although in this case the reason is different.

performance of different models Moreover, the results indicate that more complex models (MLP) do not necessarily provide better results than simpler ones, although the more complex models usually have lower mean square error. For the local search models this corresponds to our previous findings. However, for the pre-selection models the results are new. One could intuitively expect more precise models to provide better results, however this may not be the case, as the mean square error (which is the measure minimized during the training of such models) need not correspond to the suitability of the model to be used during the evolution (similar conclusion was also made by others (Diaz-Manriquez et al., 2011)). Overall, the best results are often achieved by the combination of linear regression as the local search model and support vector regression as the pre-selection model.

7.4.2 *Specific observations*

On the ZDT1 test problem, an interesting behavior can be observed. While SBMO-ES needs a large number of evaluations to improve from the $H_{ratio} = 0.95$ to $H_{ratio} = 0.99$, the proposed algorithm, together with the original non-surrogate versions of the algorithm (especially NSGA-II) require a large number of evaluations for improvement from $H_{ratio} = 0.5$ to $H_{ratio} = 0.75$. This might indicate that both of these approaches have problems with different local optima, and provides an opportunity for further research of an adaptive selection. Interesting results are those of IBEA-LR-LR, IBEA-SVM-SVM and both configurations with the MLP pre-selection which were able to escape the local optima quite quickly and thus reduced the required number of evaluations considerably compared to SBMO-ES.

For ZDT2 and ZDT3, the results compared to SBMO-ES are better for all configurations except NSGA-LR-MLP on ZDT2. The number of evaluations is decreased significantly, in some cases by almost 50%.

ZDT6 is the most complex of the test problems – both of its objective functions are multi-modal. This is the reason, why the differences among the different configurations of the models are most pronounced here. In this test, the more complex models of multilayer perceptrons improve the results significantly compared to the simpler models created by linear regression or support vector regression. This might be caused by the more complex objectives in ZDT6 compared to the other problems. Although the best result for this algorithm was obtained by SBMO-ES, the results obtained by IBEA-LR-MLP are comparable (only 6% worse) and the more parallel nature of this algorithm can prove to be more suited for practical tasks.

7.5 COMPARISON OF ASM-MOMA AND LSPS-MOEA

Although we have provided comparisons of most of the methods in the chapters where they were presented, these comparisons were mostly on the ZDT benchmark set which is rather easy. It contains mostly separable functions, with the Pareto set on the boundary of the search space. In this section, we compare selected methods on a wider set of benchmark functions and we also let the methods run longer to provide a more realistic comparison. In this section, the algorithms are compared on the ZDT (Zitzler, Deb, et al., 2000), IHR (Igel et al., 2007), and WFG (Huband et al., 2006) benchmark sets.

sNSGA-II

We compare NSGA-II with the hypervolume contribution as the secondary sorting criterion (sNSGA-II), to ASM-MOMA with support vector regression based models ($\epsilon = 0.01$, $C = 5$, RBF kernel with $\sigma = 0.5$) and to LSPS-MOEA with the same type of models. As the WFG benchmarks have different domains for different variables, we also added a variant of ASM-MOMA which divides the value of the each of the variables by the range of its domain (ASM-MOMA-S). For ZDT benchmarks this does not change anything, as all the variables are in the interval $[0, 1]$. For IHR problems, this transforms the variables to interval $[-0.5, 0.5]$, and for WFG all variables are again scaled into $[0, 1]$. LSPS-MOEA also uses this scaling in these experiments. LSPS-MOEA uses only global model (locality $\lambda = 0$), and both ASM-MOMA and LSPS-MOEA use NSGA-II with Hypervolume as Secondary Sorting Criterion (sNSGA-II) as the external algorithm.

The algorithms were selected as representatives of the previous approaches. sNSGA-II is a better variant of NSGA-II and serves as the baseline and external algorithm here, ASM-MOMA is the first algorithm we proposed, and LSPS-MOEA is the most recent version of the algorithm. The support vector regression based model was chosen as it provided consistently good results in the previous experiments.

benchmarks

The ZDT1, ZDT2, and ZDT3 problems have 30 variables, the ZDT6 and all IHR problems have 10 variables, and the WFG problems have 24 variables (in all cases, these are the numbers suggested in the original publications on these problems).

settings

All algorithms have population of 100 individuals and use the SBX crossover (with 0.9 probability) and polynomial mutation (0.03). ASM-MOMA runs the local search with probability 0.25 and the internal evolutionary algorithm has 52 individuals and runs for 30 generations. It uses the same operators as the external algorithm (sNSGA-II is used in all cases). The algorithm is given the limit of 100,000 objective function evaluations.

hypervolume difference

In this case, we used the ΔH indicator – the difference between the hypervolume of the optimal μ-distribution (Auger, Bader, Brockhoff, and Zitzler, 2009) for the given problem and the hypervolume of Pareto front found by the algorithm. This measure allows for easier visualization of the convergence when plotted on the logarithmic scale, compared to the measures we used in the previous chapters. Most of the algorithms are able to obtain $\Delta H = 0.001$ at least on some of the problems and in this cases the values of the H_{ratio} would be close to one, and it would be difficult to visualize the progress.

plots

The results of the experiments are summarized in Figures 7.1 to 7.6. The plots show the median ΔH obtained by the algorithms in 15 independent runs as the function of the number of objectives evaluations. The error-bars (plotted after each 2,000 evaluations) show the first and third quartile of this value. Although all the runs were stopped after 100,000 function evaluations, the algorithms usually converged

Figure 7.1: Comparison of sNSGA-II, ASM-MOMA, and LSPS-MOEA on ZDT1, ZDT2, and ZDT3 benchmarks

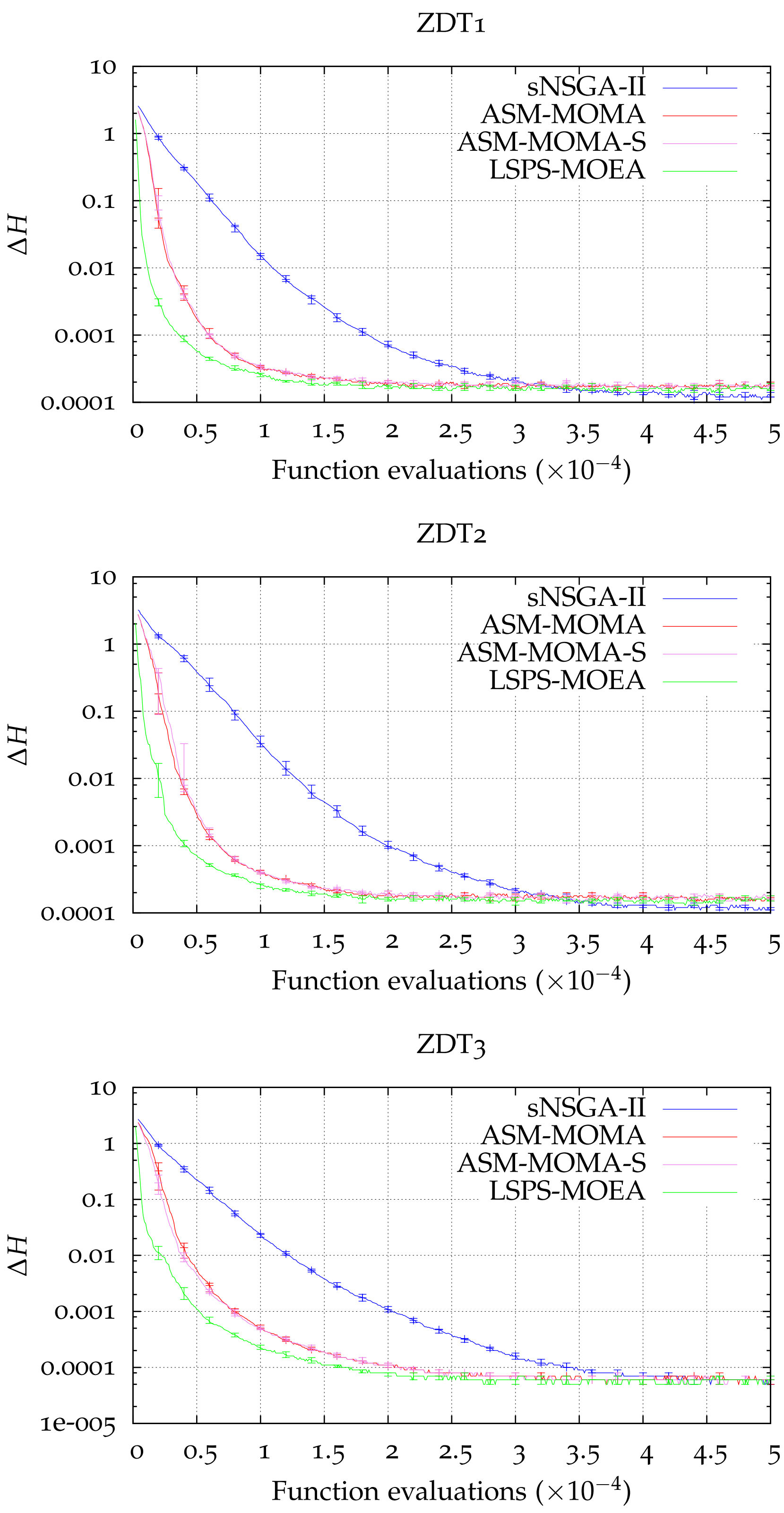


Figure 7.2: Comparison of sNSGA-II, ASM-MOMA, and LSPS-MOEA on ZDT6, IHR1, and IHR2 benchmarks

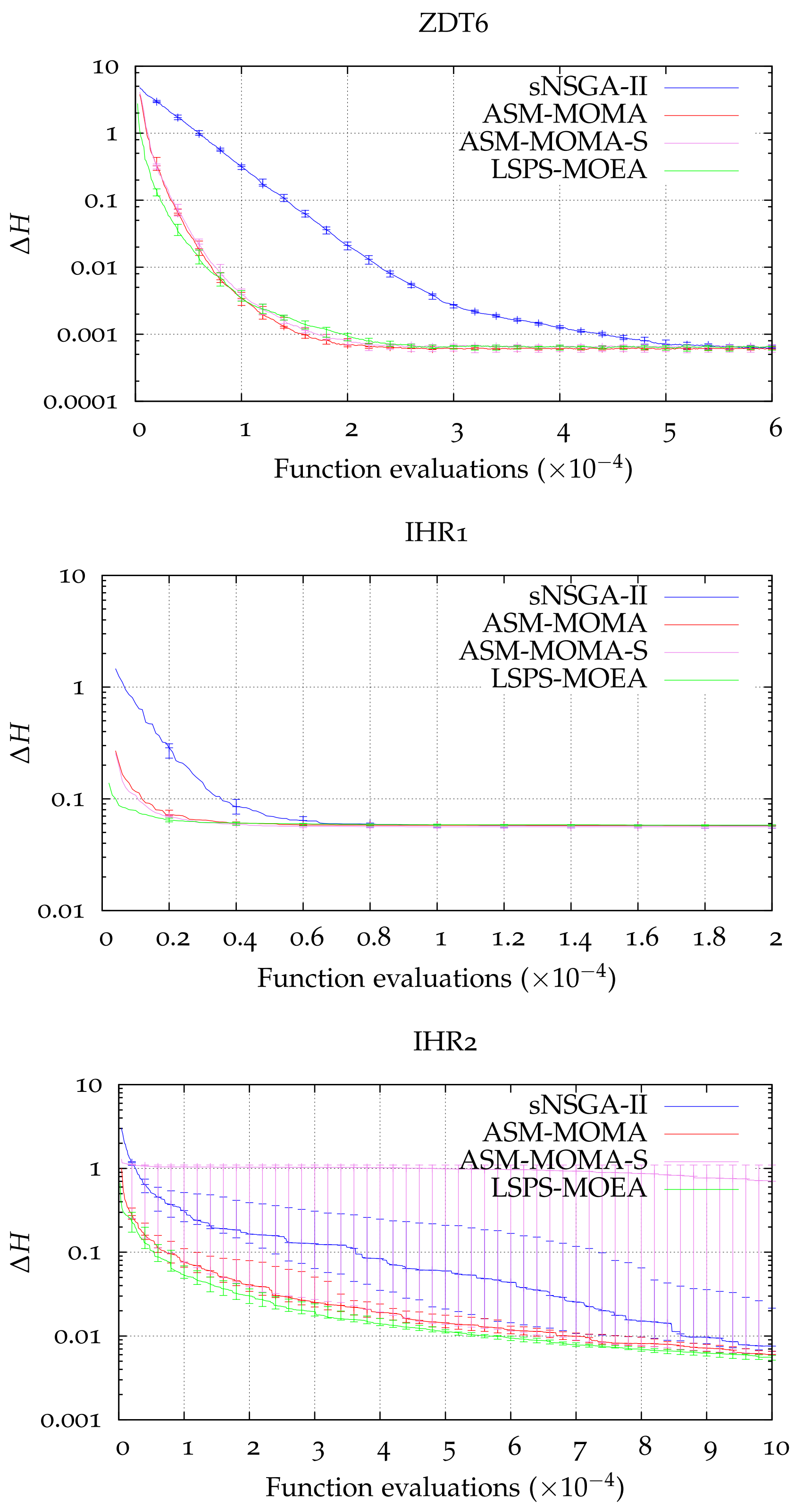
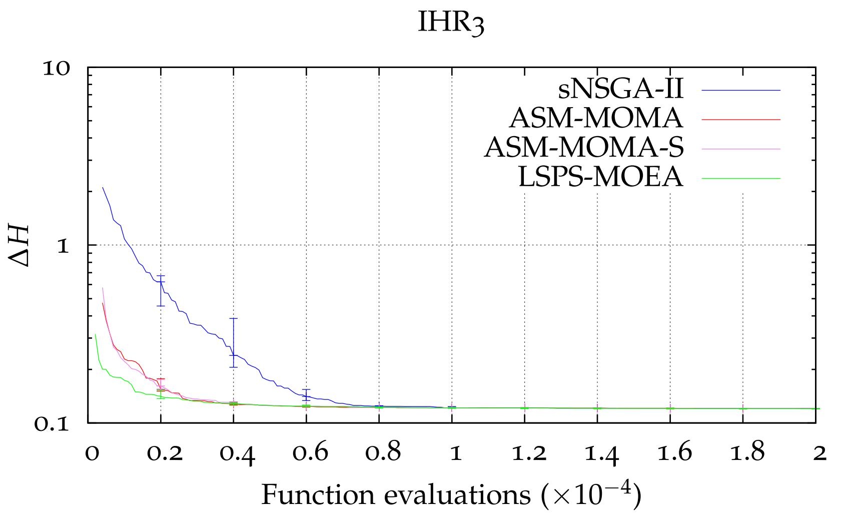
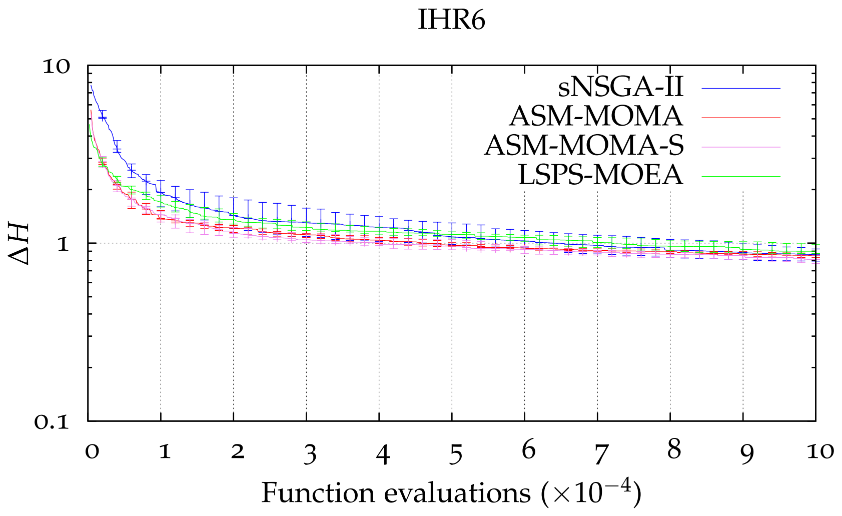


Figure 7.3: Comparison of sNSGA-II, ASM-MOMA, and LSPS-MOEA on IHR3, IHR6, and WFG1 benchmarks





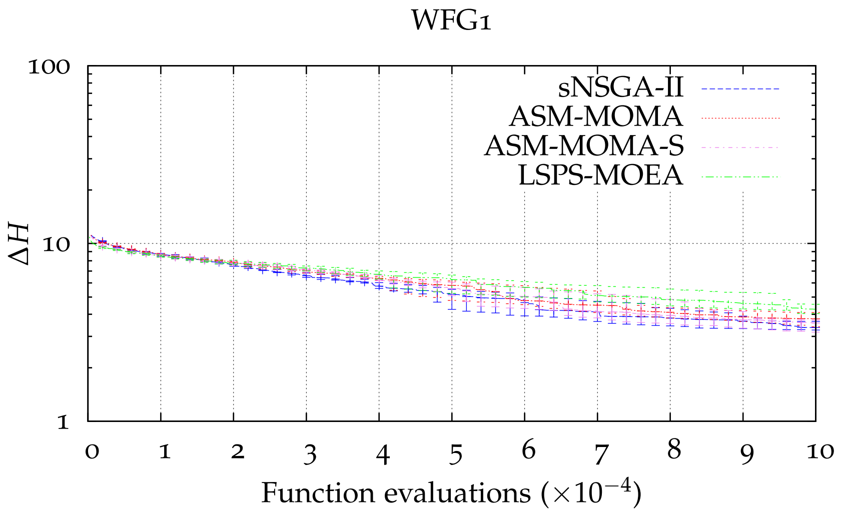
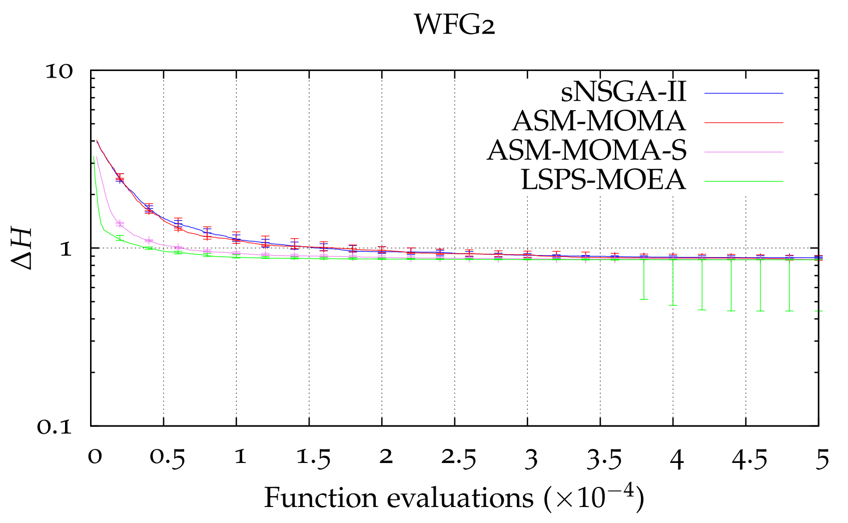
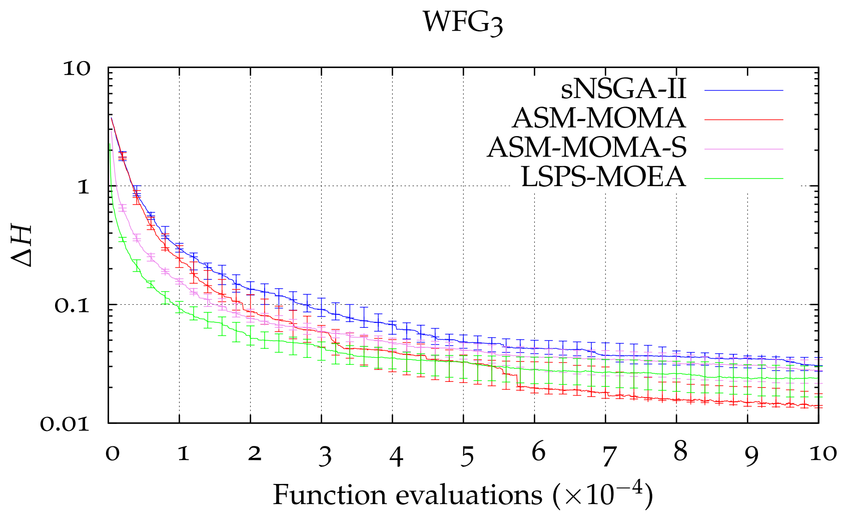


Figure 7.4: Comparison of sNSGA-II, ASM-MOMA, and LSPS-MOEA on WFG2, WFG3, and WFG4 benchmarks





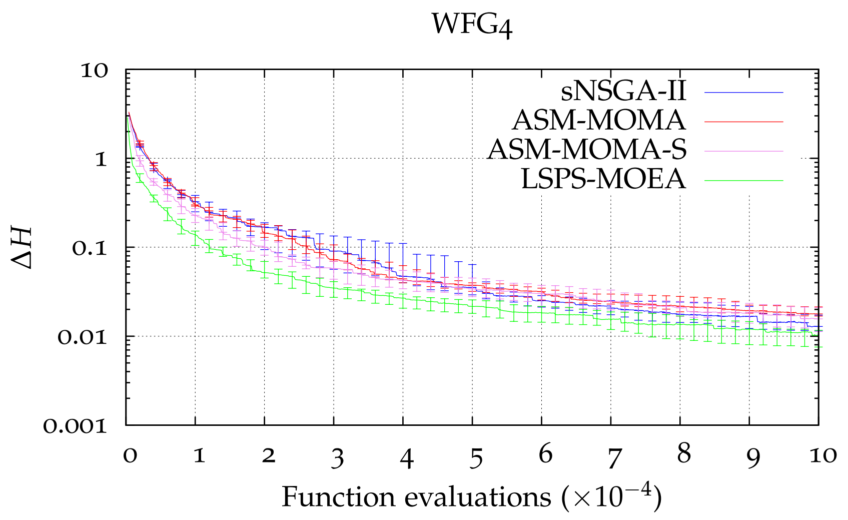
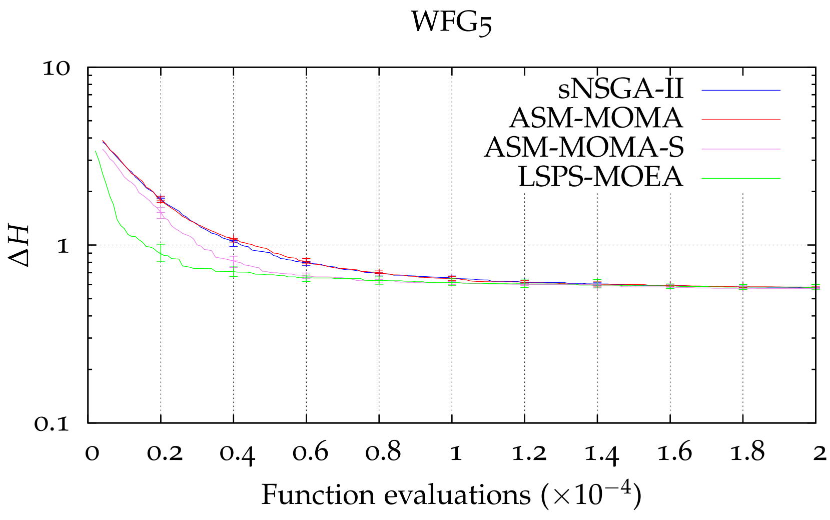
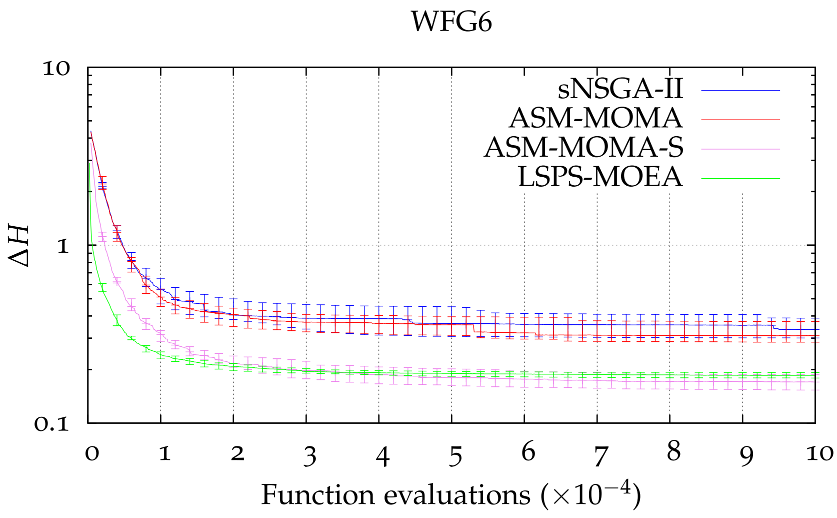


Figure 7.5: Comparison of sNSGA-II, ASM-MOMA, and LSPS-MOEA on WFG5, WFG6, and WFG7 benchmarks





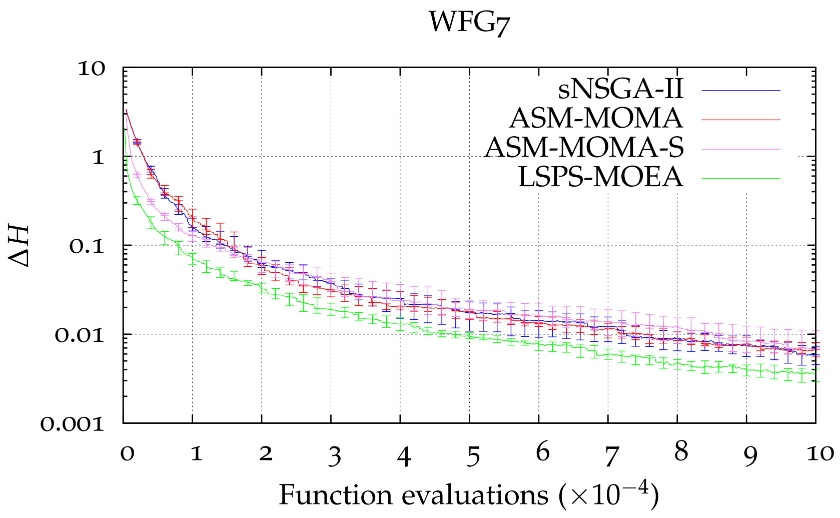
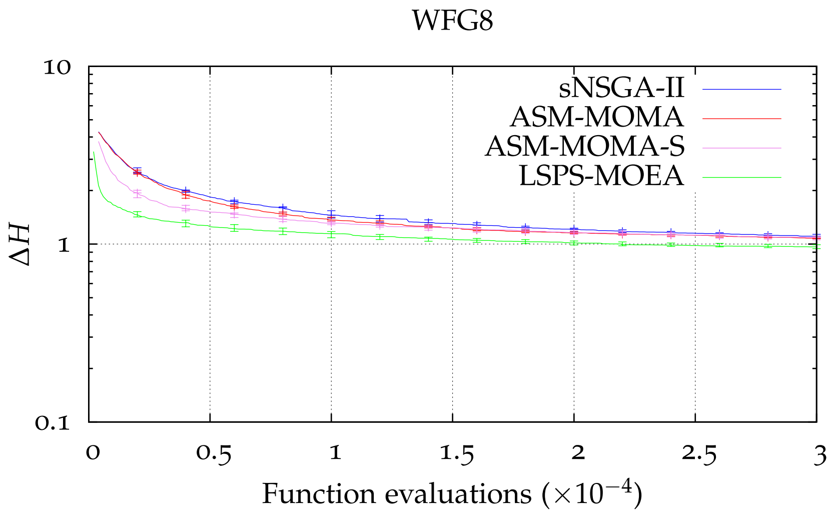


Figure 7.6: Comparison of sNSGA-II, ASM-MOMA, and LSPS-MOEA on WFG8 and WFG9 benchmarks



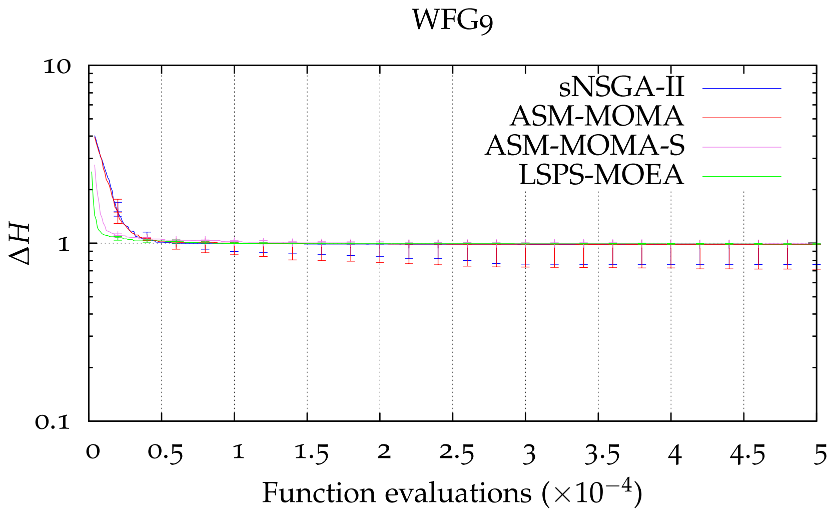


Table 7.10: Comparison of the algorithms on the ZDT and IHR benchmarks. The superscript indicates which of the other algorithms were significantly worse (Mann-Whitney U-test $p < 0.001$; "N" for sNSGA-II, "A" for ASM-MOMA, "S" for ASM-MOMA with scaling, and "L" for LSPS-MOEA).

Prob.	Algorithm	Function evaluations					
		1000	5000	10000	25000	50000	100000
ZDT1	sNSGA-II	1.70890	0.18617	0.01518	0.00035	0.00012^{ASL}	0.00013^{AL}
	ASM-MOMA	0.88151^{N}	0.00175^{N}	0.00033^{N}	0.00019^{N}	0.00018	0.00017
	ASM-MOMA-S	0.88466^{N}	0.00193^{N}	0.00034^{N}	0.00018^{N}	0.00017	0.00016
	LSPS-MOEA	0.01444^{NAS}	0.00058^{NAS}	0.00026^{NAS}	0.00015^{N}	0.00018	0.00016
ZDT2	sNSGA-II	2.25572	0.37916	0.03315	0.00040	0.00011^{ASL}	0.00011^{ASL}
	ASM-MOMA	1.19259^{N}	0.00292^{N}	0.00040^{N}	0.00019^{N}	0.00016	0.00016
	ASM-MOMA-S	1.22651^{N}	0.00325^{N}	0.00040^{N}	0.00018^{N}	0.00016	0.00015
	LSPS-MOEA	0.04609^{NAS}	0.00072^{NAS}	0.00026^{NAS}	0.00016^{N}	0.00016	0.00016
ZDT3	sNSGA-II	1.81398	0.22630	0.02371	0.00038	0.00006	0.00005
	ASM-MOMA	1.29742^{N}	0.00563^{N}	0.00050^{N}	0.00008	0.00006	0.00005
	ASM-MOMA-S	1.09130^{N}	0.00449^{N}	0.00049^{N}	0.00008	0.00006	0.00005
	LSPS-MOEA	0.03684^{NAS}	0.00109^{NAS}	0.00022^{NAS}	0.00006	0.00006	0.00005
ZDT6	sNSGA-II	3.81687	1.30253	0.31670	0.00669	0.00071	0.00057
	ASM-MOMA	1.15168^{N}	0.03239^{N}	0.00342^{N}	0.00064^{N}	0.00061	0.00057
	ASM-MOMA-S	1.25898^{N}	0.03811^{N}	0.0039^{N}	0.00067^{N}	0.00064	0.00062
	LSPS-MOEA	0.38956^{NAS}	0.02181^{NAS}	0.00348^{N}	0.00071^{N}	0.00064	0.00061
IHR1	sNSGA-II	0.70932	0.06992	0.05824	0.05734	0.05711	0.05649
	ASM-MOMA	0.11616^{N}	0.05978^{N}	0.05794	0.05757	0.05726	0.05695
	ASM-MOMA-S	0.10758^{N}	0.05724^{N}	0.05624	0.05593	0.05576	0.05517
	LSPS-MOEA	0.07865^{NAS}	0.06019^{N}	0.05865	0.05799	0.05753	0.05738
IHR2	sNSGA-II	1.95942	0.50976	0.31471	0.15387	0.05985	0.00757
	ASM-MOMA	0.41365^{N}	0.13860^{N}	0.07623^{N}	0.02987^{N}	0.01435	0.00603
	ASM-MOMA-S	1.14490^{N}	1.05561	1.04198	1.03247	0.98743	0.70423
	LSPS-MOEA	0.27965^{NAS}	0.10404^{N}	0.05339^{NS}	0.02248^{NS}	0.01127^{NS}	0.00558^{N}
IHR3	sNSGA-II	1.08313	0.17258	0.12176	0.12031	0.11981	0.11957
	ASM-MOMA	0.22870^{N}	0.12536^{N}	0.12170	0.12018	0.11989	0.11968
	ASM-MOMA-S	0.22190^{N}	0.12548^{N}	0.12182	0.12023	0.11988	0.11964
	LSPS-MOEA	0.17366^{NAS}	0.12550^{N}	0.12139	0.12002	0.11965	0.11951^{AS}
IHR6	sNSGA-II	6.42174	2.93054	1.92699	1.32843	1.08368	0.86082
	ASM-MOMA	3.74191^{N}	1.91325^{N}	1.37783^{NL}	1.14771^{L}	0.97041^{L}	0.85406^{L}
	ASM-MOMA-S	3.78398^{N}	1.87897^{N}	1.44271^{NL}	1.07767^{NL}	0.96284^{L}	0.81122^{L}
	LSPS-MOEA	3.43330^{NA}	2.08986^{N}	1.69939	1.29656	1.11967	0.91169

Table 7.11: Comparison of the algorithms on the WFG benchmark. The superscript indicates which of the other algorithms were significantly worse (Mann-Whitney U-test $p < 0.001$; "N" for sNSGA-II, "A" for ASM-MOMA, "S" for ASM-MOMA with scaling, and "L" for LSPS-MOEA).

Test	Algorithm	Function evaluations					
		1000	5000	10000	25000	50000	100000
WFG1	sNSGA-II	10.70700	9.35227	8.70460	7.03192	5.17987	3.36809^{L}
	ASM-MOMA	10.51760	9.44238	8.68383	7.41937	5.81645	3.77332
	ASM-MOMA-S	10.28680^{NA}	9.10285^{NA}	8.53707	7.48011	5.58442	3.56751^{L}
	LSPS-MOEA	9.86843^{NAS}	9.07836^{NA}	8.53448	7.57294	6.26300	4.26715
WFG2	sNSGA-II	3.26656	1.46850	1.12326	0.93374	0.88305	0.86558
	ASM-MOMA	3.26224	1.43741	1.10853	0.92990	0.86332	0.84710^{N}
	ASM-MOMA-S	2.00426^{NA}	1.03803^{NA}	0.93445^{NA}	0.87653^{NA}	0.86343^{N}	0.86110
	LSPS-MOEA	1.25058^{NAS}	0.95989^{NAS}	0.88789^{NA}	0.86402^{NA}	0.85885^{N}	0.85040
WFG3	sNSGA-II	2.78777	0.68498	0.29435	0.11441	0.04831	0.03046
	ASM-MOMA	2.78861	0.59993	0.24546	0.07229	0.03272^{N}	0.01407^{NSL}
	ASM-MOMA-S	1.16110^{NA}	0.28443^{NA}	0.15653^{NA}	0.06570^{N}	0.04104	0.02771
	LSPS-MOEA	0.54880^{NAS}	0.16679^{NAS}	0.09173^{NAS}	0.04865^{NAS}	0.03270^{N}	0.02395
WFG4	sNSGA-II	2.19325	0.63526	0.32519	0.12998	0.03537	0.01288
	ASM-MOMA	2.23614	0.67815	0.29909	0.10870	0.03735	0.01778
	ASM-MOMA-S	1.82587^{NA}	0.46778^{NA}	0.22830^{A}	0.06738^{A}	0.03402	0.01565
	LSPS-MOEA	0.81912^{NAS}	0.29346^{NAS}	0.13820^{NAS}	0.04287^{NAS}	0.02165^{NAS}	0.01029^{A}
WFG5	sNSGA-II	2.81432	0.90662	0.65068	0.56572	0.55109	0.54776
	ASM-MOMA	2.81188	0.93021	0.64657	0.56734	0.55322	0.54885
	ASM-MOMA-S	2.41022^{NA}	0.70094^{NA}	0.61300^{A}	0.56416	0.55176	0.54857
	LSPS-MOEA	1.23550^{NAS}	0.68109^{NA}	0.61480	0.56187	0.55415	0.54870
WFG6	sNSGA-II	3.36046	0.95998	0.56237	0.39629	0.36284	0.33590
	ASM-MOMA	3.37579	0.94875	0.51098	0.37985	0.35655	0.30983
	ASM-MOMA-S	2.07042^{NA}	0.51959^{NA}	0.31521^{NA}	0.20699^{NA}	0.18005^{NA}	0.17006^{NA}
	LSPS-MOEA	0.81183^{NAS}	0.32613^{NAS}	0.24228^{NAS}	0.20148^{NA}	0.18993^{NA}	0.18528^{NA}
WFG7	sNSGA-II	2.41298	0.52473	0.15836	0.04965	0.01754	0.00586
	ASM-MOMA	2.34310	0.50924	0.20848	0.03985	0.01801	0.00667
	ASM-MOMA-S	1.15943^{NA}	0.24937^{NA}	0.12868^{A}	0.04809	0.01887	0.00684
	LSPS-MOEA	0.49530^{NAS}	0.14261^{NAS}	0.07301^{NAS}	0.02512^{NAS}	0.00943^{NAS}	0.00357^{NAS}
WFG8	sNSGA-II	3.38109	1.83913	1.45536	1.15209	1.02516	0.96183
	ASM-MOMA	3.30802	1.74878	1.36596^{N}	1.12165	1.01803	0.95857
	ASM-MOMA-S	2.48120^{NA}	1.51600^{NA}	1.31268^{N}	1.12447	1.00793	0.94049
	LSPS-MOEA	1.66216^{NAS}	1.25356^{NAS}	1.14571^{NAS}	0.98191^{NAS}	0.91208^{NAS}	0.87040^{NAS}
WFG9	sNSGA-II	2.79457	1.01242	0.99700^{S}	0.98701^{SL}	0.98454^{SL}	0.98380^{SL}
	ASM-MOMA	2.63249	1.01875^{S}	1.00149	0.98741^{S}	0.98522^{S}	0.98450^{S}
	ASM-MOMA-S	1.29456^{NA}	1.04947	1.02054	0.99975	0.99323	0.99006
	LSPS-MOEA	1.12061^{NAS}	1.03013	0.99693	0.99357	0.98916	0.98634

much earlier, therefore we show only the parts of the plots until convergence is achieved and the hypervolume (including the quartiles) does not change. All the plots use logarithmic scale on the vertical axis.

Moreover, the results are also numerically summarized in Tables 7.10 and 7.11. In this tables, we present the ΔH after a 1000, 5000, 10000, 25000, 50000, and 100000 objective function evaluations. Additionally, superscripts above some of the values indicate, that the given algorithm was significantly better than the algorithm indicated by the superscript. The significance was tested by Mann-Whitney U-test (also known as Wilcoxon rank-sum test) (Mann and Whitney, 1947; Wilcoxon, 1945), and results are considered significant if the p-value of the test is less than 0.001. The superscripts encode the algorithm in the following way - "N" stands for sNSGA-II, "A" stands for ASM-MOMA, "S" denotes the ASM-MOMA with scaling before model training, and "L" is for LSPS-MOEA.

general results Generally, the results confirm what we have already seen on the ZDT benchmark set in the previous chapters. Both ASM-MOMA and LSPS-MOEA generally outperform the plain sNSGA-II and lead to a significant speed up, at least in the beginning of the evolution. The long runs also allow us to see that in most cases all the algorithms find a similar optima and the main difference is in the number of evaluations it takes them. For some of the problems, sNSGA-II can slightly outperform the surrogate based algorithms after a large number of evaluations, although the difference is quite small. We can also see that in most cases LSPS-MOEA is at least as good as ASM-MOMA, and it is usually better. Also, the scaled version of ASM-MOMA gives better results than the unscaled one for the WFG problems. They have the same results on the ZDT benchmark, as the scaling does nothing in this case.

ZDT benchmark The results on the ZDT benchmark set should not be surprising, as it is the set we used in all of the comparisons before. The only new information here is that sNSGA-II can slightly outperform the surrogate version on the ZDT1 and ZDT2 problems after approximately 40,000 function evaluations. Another interesting observation is that the performance of ASM-MOMA and LSPS-MOEA is almost the same on the ZDT6 problem. However, this is expected and it matches the results we have seen earlier with the SVR-based model.

IHR benchmark On the other hand, the results for the IHR benchmark are new. We can see that in the case of IHR1 and IHR3 all the compared algorithms converge to a ΔH value after a few thousand evaluations and nothing changes afterward. On the other hand, for the IHR2 problem, slow convergence can be observed during the whole budget of 100,000 evaluations. Also, the behavior of ASM-MOMA with scaling before model training is interesting. The algorithm suffers much more from pre-mature convergence than the others and is not able (in the median case) to converge very well, all the other algorithms perform much better. However, sNSGA-II shows quite large inter-quartile difference which also indicates that it has problems with convergence. On the other hand ASM-MOMA without scaling and LSPS-MOEA both converge relatively well, with LSPS-MOEA showing better results. On IHR6, which is multi-modal, LSPS-MOEA has some problems to converge, and although it outperforms sNSGA-II, it is not able to outperform ASM-MOMA.

WFG benchmark

The WFG set of benchmark functions is more challenging. One can observe that for most of the functions even the $\Delta H = 0.1$ is not attained by the algorithms. This is partially due to the nature of the external algorithm which have problem with these functions itself. In most cases the surrogate modeling improves the speed of convergence, however, the overall result is not improved much. We can also see that the scaling improves the results significantly, and that models used without the scaling in fact do not improve the performance compared to sNSGA-II. This effect is most pronounced in the results for the WFG6 problem, where the scaled ASM-MOMA and LSPS-MOEA (which also uses the scaling) significantly outperforms the sNSGA-II and ASM-MOMA without scaling and are even able to obtain better results in the long run. The reason for the large effect of scaling is that the WFG problems have very dissimilar domains, the domain of i-th variable is $[0, 2i]$. This makes it difficult for the model to approximate the function. Also, the distances can be quite large in this case, which may be another complication. The scaling reduces both these problems.

An interesting behavior can be observed on the WFG2 problem. All the algorithms after approximately 20,000 evaluation converge to a similar value and the results do not change until around 37,000 of evaluations where some of the runs of LSPS-MOEA suddenly improve the value quite dramatically as can be seen by the drop in the first quartile, however the median run is still the same. WFG2 has disconnected Pareto front and LSPS-MOEA in this case finds a new part of the front. However, it manages to do so only in a small fraction of runs, so the median value does not change. Despite this fact it may be interesting to study this behavior further and try to find a way which would increase the probability of this happening. We can see a similar behavior on WFG9. However, in this case LSPS-MOEA is the algorithm which does not improve further, and ASM-MOMA and even sNSGA-II are able to find better solutions at least in a few of the runs (again, medians are still the same).

Overall, although the results on the WFG toolkit mostly confirm that ASM-MOMA and LSPS-MOEA both outperform the baseline sNSGA-II and that LSPS-MOEA outperforms ASM-MOMA, the WFG problems are still challenging even for the surrogate based algorithms and provide strong motivation for future research.

comparison to Loshchilov at al.

The methodology chosen in this section matches the one adopted by Loshchilov et al. (2010c) in their paper on aggregate surrogate models based on ranking SVMs. Thus, we can now compare these two approaches directly. The comparison shows, that ASM-MOMA outperforms the algorithm based in ranking SVM by approximately 20%-40% function evaluations needed to attain a specified target ΔH. An interesting observation is that the difference between the two algorithms gets smaller for smaller targets of ΔH. This can indicate that the approach presented by Loshchilov at al. is able to provide better convergence at the later phases of the evolution, while ASM-MOMA can find some solution faster. However, the number of differences between these two algorithms is quite large (e.g. different models, different type of exploitation of the model, different archive truncation technique), thus it is not clear, where does the difference between the two approaches come from, and whether they can be combined in a way which would provide even better performance. This is left as a future work.

7.6 CONCLUSIONS

We have shown that the use of different types of meta-models in local search phase and pre-selection phase is essential and leads to improved convergence speed.

We presented a new version of SBMO-ES, LSPS-MOEA, which is capable of producing more individuals in each generation and still provides comparable performance in terms of the number of evaluations of the real objective function. This may be important in practice as some of the simulations or experiments may be parallelized (and in some cases may be cheaper when run in small batches).

During preliminary testing, we have shown that configurations with more simple meta-models (linear regression, or support vector regression) in the local search phase and more complex models (support vector regression, or multilayered perceptrons) in the pre-selection phase, combined with ϵ-IBEA selection work particularly well and can be used in most situations. We have also encountered interesting behavior on one of the test problems (ZDT1), which provides a possibility for future research – e.g. the convergence speed might be monitored and the type of selection (or the number of selected individuals) could be tuned adaptively.

Future work should also be focused on the way surrogate models are selected and trained. The traditional approaches which use the mean square error might not be the most suitable for the use in conjunction with evolutionary algorithms. There is a possibility to train multiple models and select one of them based on another criterion.

8 SURROGATE MODEL SELECTION

In the previous chapter, each time we tested an algorithm we tried several different types of surrogate models. We have also seen that the results of the algorithm depend on the choice of model and that a wrongly chosen model can significantly affect the performance. In this chapter, we look at the problem of model selection more closely, and study which features of models are important for their good performance in evolutionary algorithms. To this end, we define four different criteria based on the performance of the model on a validation set and use them to select the best model in each run. The goal of this chapter is mostly to compare the effect of different model selectors rather than the performance of the algorithm.

8.1 MOTIVATION

criticism

In most cases, the authors of a surrogate evolutionary algorithm do not discuss why they use a particular type of surrogate models. The model is usually selected in advance and is never changed during the evolution. Often, the same model is used for different tasks. However, different tasks may require different types of models, as each type of model provides good performance for different types of training sets. Some are good if there are more variables, some generalize well, some are more simple, with less local optima, and thus can be easily optimized in a kind of local search.

MSE and other errors

The problem of the selection of a suitable model is quite complex. The most precise model (in terms of mean square error, MSE) may not perform well when used in an evolutionary algorithm which uses only comparisons to select individuals. The reason is that although the model has low MSE it may still predict comparisons between individuals incorrectly. If the model predicts higher value for one individual and lower value for the other the ordering of the individuals according to the model may be different than it is in reality. On the other hand, if the model predicts consistently higher values for all individuals the comparisons may be correct, even though the MSE is large. To solve this problem, we define four different model selectors and use them to select one of the trained models. These selectors decide based on different criteria. One of them is the most obvious approach which selects the model with the lowest mean square error on a validation set. Other selectors use the bias and variance of the error on the validation set, one of the selectors uses the so called relation preservation. It is a metric, which describes how well a given model preserves relations (comparisons) between the individuals in the validation set. This is a feature which may be better suited than the others for model selection in evolutionary algorithms that only use comparisons during their run.

There are not many references to algorithms which would deal with more than one model. One of the exceptions is (Lim et al., 2010), where the authors use two different local meta-models. Both are trained to approximate a weighted sum of the objectives. One is an ensemble model, the other is a low order polynomial. Two single-objective algorithms are run to find optima of the respective models, which are then precisely evaluated. A selection procedure is used to decide which of the individuals (if any) is added to the population.

complementary approach

The approach used in Diaz-Manriquez et al. (2011) is in a sense complementary to the one presented here. Authors compare different models and monitor various features of the models during the evolutionary search. One of the results of the work is, that mean square error might not be the best measure of the suitability of the model for the use with evolutionary algorithms. The authors also argue that even model with large mean square error performs well when it preserves the ordering of the individuals well.

Loshchilov et al. (2010b) also argue that comparison based optimizers should use comparison based surrogates and demonstrate this fact by using ranking SVM as the surrogate model in ACM-ES.

8.2 SURROGATE MODEL SELECTION

ultimate goal

Our ultimate goal is to integrate surrogate evolutionary algorithms with meta-learning system (Kazík et al., 2012). The meta-learning system would recommend a set of types of surrogate models based on the features of the training set (i.e. the set of previously evaluated individuals). All of the recommended models would be trained and then a model would be selected and used during the evolution.

relation preservation

To this end, we define four different features of the surrogate models below. These are then used to select the best model from the set using the four selectors defined later.

Definition 8.1. Let I be the indicator function (1 if its argument is true and 0 if it is false). Let $F : \mathbb{R}^n \to \mathbb{R}$ be a modeled function, and let M is its model. Let $V \subseteq \mathbb{R}^n$ be a finite validation set. The *relation preservation (RP) measure* is defined as

$$RP(M) = \frac{1}{|V|(|V|-1)} \sum_{x,y \in V} I(F(x) \leq F(y) \& M(x) \leq M(y))$$

The relation preservation measure might be more important when the use of a model as a surrogate function is considered in evolutionary algorithms, because most EAs compare two individuals rather than using the value of the function directly, thus errors in the absolute value of the function are not important.

bias

Definition 8.2. Let V, F, and M be defined as above. The *bias* is defined as

$$BIAS(M) = \frac{1}{|V|} \sum_{x \in V} (M(x) - F(x)).$$

The bias on the other hand is something that should not affect the performance of the evolutionary algorithms at all. A model, which consistently predicts $f + 10$ instead of f, is from the point of view of a comparison-based evolutionary algorithm as good as a model which would consistently predict f. We define this criteria, and the respective model selector, mainly to validate this claim and for the sake of completeness. In fact, we expect the performance of the algorithm which use this selector to be rather poor.

variance

Definition 8.3. Let V, F, and M be defined as above, and b is the bias of M as defined above. The *variance* is defined as

$$VAR(M) = \frac{1}{|V|} \sum_{x \in V} (b - (M(x) - F(x))^2.$$

The variance of the model expresses how consistently the model predicts the values. Large variance means, that the model often overestimates or underestimates the function F, and moreover the error is different for each point x. This may lead to a large number of wrongly predicted comparisons. The motivation for this criterion is that its performance should be similar to the one of the RP criteria, but this one is continuous and thus more easily usable for training of models.

For the sake of completeness we also define the well known mean square error here.

mean square error

Definition 8.4. Let V, F, and M be defined as above. The *mean square error* is defined as

$$MSE(M) = \frac{1}{|V|} \sum_{x \in V} (F(x) - M(x))^2.$$

selectors

Now, we can define four types of selectors. The role of the selector is to select a model from a set of possible (recommended) models based on one of the features defined above.

Definition 8.5. Let R be a set of models. Then

- $\mathcal{S}_M(R) = \arg\min_{m \in R} MSE(m)$ is the *mean square error selector*,
- $\mathcal{S}_B(R) = \arg\min_{m \in R} BIAS(m)$ is the *bias selector*,
- $\mathcal{S}_V(R) = \arg\min_{m \in R} VAR(m)$ is the *variance selector*,
- $\mathcal{S}_P(R) = \arg\max_{m \in R} RP(m)$ is the *relation preservation selector*.

Note that the relation preservation criteria is the only one which should be maximized.

8.3 EXPERIMENTS

algorithm To test the performance of different selectors we used a variant of LSPS-MOEA. In this variant, each time a surrogate models should be used, several models are trained and evaluated using the given model selector. The best model (according to the selector) is than used for the predictions in the given generation. The models are re-trained and re-selected in each generation. All the models are trained using their respective training algorithms which optimize the MSE. This is an important feature which shall be taken into account while interpreting the results. In this case we use the fact that all the criteria are correlated in the sense that when MSE is zero, the other criteria also have the best possible values, and thus optimizing the MSE also leads to good values of the other criteria (although generally not the best possible). Moreover, models which optimize the MSE also should generally have very low value of BIAS, as this can be removed by a simple additive term which most of the models contain.

benchmark We tested our approach on the widely used ZDT (Zitzler, Deb, et al., 2000) benchmark problems. These problems are all two dimensional, and we used 15 variables for each of them, except the ZDT1 where 30 variables were used.

The main multi-objective algorithm (NSGA-II, or ϵ-IBEA) used a population of 50 individuals and stopped after 50,000 objective function evaluations. The SBX crossover (Deb and Agrawal, 1994) (probability 0.8) and polynomial mutation (Deb and Goyal, 1996) (probability 0.1) were used. During the local search we used an evolutionary algorithm which run with 50 individuals in the population for 30 generations, again SBX crossover and polynomial mutation were used.

To compare the results we again use the H_{ratio} measure, it is defined as the ratio of the hyper-volume (Zitzler and Lothar Thiele, 1998) of the dominated space attained by the algorithm divided by the hyper-volume of the global Pareto front.

types of models Although our ultimate goal is to integrate the evolutionary algorithm with a meta-learning system (Kazík et al., 2012), we are not yet ready to recommend the models to be used automatically, so we manually selected ten different types of models and used them during the tests. These are a linear regression model estimated using the least squares method, 3 variants of support vector regression – with polynomial kernels of degrees one and two, and with the RBF kernel, 3 architectures of multilayer perceptrons – with 2, 5, and 10 neurons in the single hidden layer, 2 variants of RBF networks – with 2 and 5 RBF units, and Gaussian processes regression. All the other options were set to their default values from the Weka framework Hall et al. (2009). The models were selected as a representative set of the various types of models used in the literature dealing with the surrogate evolutionary algorithms.

8.3.1 *Types of Behavior*

dynamic behavior

The types of selected models change as the evolution proceeds and converges. The different behaviors can be roughly divided into three groups:

- Undecided (U) – there is no best model during the evolution. It is characterized by the fact, that none of the models is selected in more than 30% of the runs. This happens when more (all) of the models are equally good given the selection criteria.
- Constant (C) – there is a single model which is selected most often during the evolution. The model is usually selected in more than 80% of the runs. This typically happens when there is a model, which can predict the data precisely (e.g. the modeled function is linear), but there are cases, when this occurs even for more difficult problems.
- Dynamic (D) – similar to the constant behavior, but the best model changes during the run of the evolution. This is the most interesting type of behavior as it clearly shows the dynamic nature of the evolution. We expect this behavior to occur, when the complexity of the modeling of the functions change as the evolution progresses towards the optima.

The 30%, and 80% limits are rather arbitrary, derived from the observations of different behaviors. The main intuition is that for the undecided behavior, none of the models clearly wins, or is used very often. For the Constant behavior, the opposite is true. This leads to the choice of the 30% and 80% limits.

complex behaviors

Often, the behavior is more complicated than the three groups outlined above can capture. Thus, we can imagine groups which combine the features of more of them, these are denoted as A/B where A and B are one of the groups defined. For example a group denoted as D/U is a group, where there are e.g. three models clearly used more often the others (around 25% each) but none of them wins.

Also the behavior sometimes changes as the evolution proceeds. These are than denoted as A-B, where the behavior changes from A to B, and A and B are as defined above (either the basic groups, or the combined ones).

undecided and constant behavior

Figure 8.1 shows an example of the undecided and the constant behavior. It shows the relative frequency of each of the types of models as a function of the generation number of the algorithm. In the left sub-figure we can see the undecided behavior produced by the BIAS model selector on the aggregated function used in the local search phase on the ZDT6 problem with the ϵ-IBEA as the external algorithm. In this case the selector does not choose any of the model very often, probably because most of the models have low bias. On the

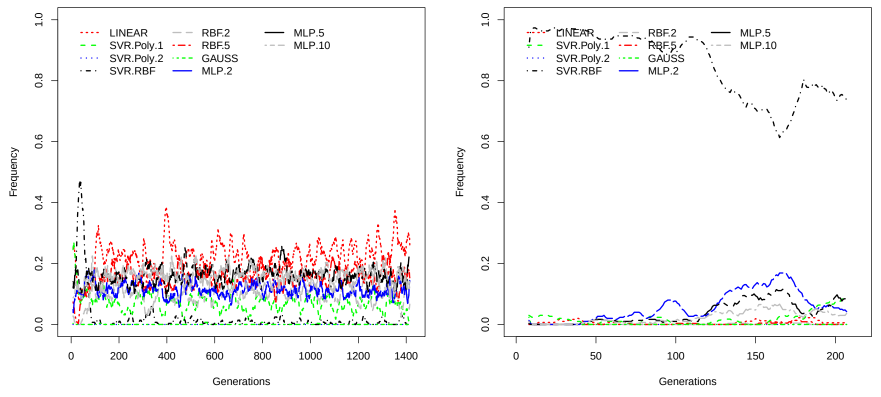


Figure 8.1: An example of undecided (left) and constant (right) behavior. The undecided behavior was obtained on the aggregated model of the ZDT6 test problem with the BIAS meta-model selector. The constant behavior was obtained on the aggregated model of the ZDT1 test problem with the MSE selector. Both configuration use ϵ-IBEA as the external algorithm.

other hand, in the right sub-figure we can see a typical example of the constant behavior as produced by the MSE selector on the aggregate function of ZDT1 with the ϵ-IBEA as the external algorithm. In this case the support vector regression with the RBF kernels provides the lowest MSE consistently over the 20 runs and is selected in most of them.

dynamic behavior

The dynamic behavior may be more interesting. We provide two examples in Figure 8.2. Further examples are provided later. In the left sub-figure, we can see the result of running the MSE model selector for the F1 objective function of the ZDT1 test problem. Again, the ϵ-IBEA is used as the external algorithm. In this case, the linear regression was used in the beginning, later it was replaced by support vector regression with polynomial kernel, and in the end they were both replaced by the support vector regression with RBF kernels. The right sub-figure shows the dynamics of the RP selector when used for the aggregated model of the ZDT1 test problem with NSGA-II as the external algorithm. In this case, the support vector regression with RBF kernels is replaced by the SVR with polynomial kernels early during the evolution and in the later phases none of the models clearly dominates.

Although the distinction between the types of behavior is rather subjective, it allows us to discuss and describe the dynamic nature more easily. The presented figures also show the importance of having the option to select from more models during the run of the evolution, because the optimal models can change as the evolution proceeds.

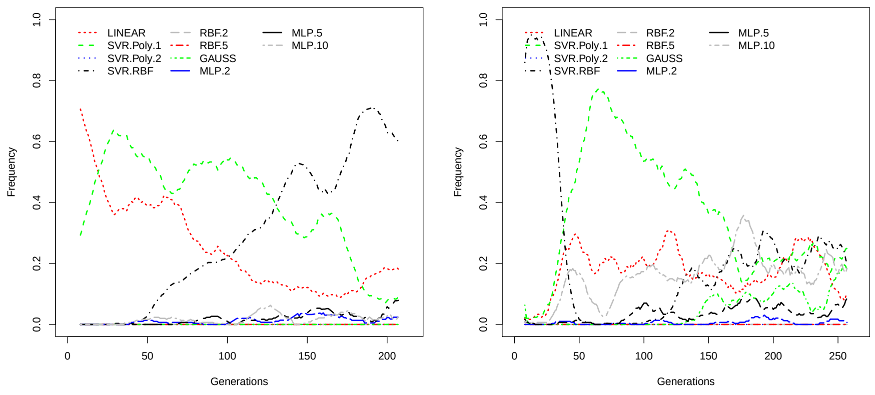


Figure 8.2: Two examples of dynamic behavior. F1 objective of the ZDT1 test problem with MSE model selector and ϵ-IBEA as the external algorithm on the left and the aggregate meta-model of the ZDT1 with the RP model selector and NSGA-II as the external algorithm on the right.

8.3.2 *Performance*

description of tables

Tables 8.1 and 8.2 show the performance of the different meta-model selector on a series of test from the ZDT Zitzler, Deb, et al. (2000) benchmark – ZDT1, ZDT2, ZDT3, and ZDT6. The numbers represent the median number of fitness evaluations needed to attain the specified H_{ratio}. For comparison, we also provide the results of plain NSGA-II and ϵ-IBEA for comparison, as well as the results we obtained previously with fixed models during the whole evolution for comparison. Tables 8.3 and 8.4 show the behaviors we observed on that particular test for each of the functions which were modeled – the aggregate function used during the local search (A), and both the objective functions (F0, F1).

performance comparison

We can clearly see, that the use of surrogate models reduce the number of objective function evaluations significantly compared to the versions of NSGA-II and ϵ-IBEA, this is an expected result. A more interesting observation is that for ZDT1 and ZDT6 the dynamic selection of surrogate models also improves the results significantly compared the to the versions with statically assigned model types. For ZDT3 the results of statically assigned and dynamically selected models are comparable. On the other hand, the results for ZDT2 are better with the statically selected models. This is an interesting observation and should be further studied in the future. However, note that the best results with statically selected models were not obtained by one fixed configuration, but with different configurations for different values of H_{ratio}. Also, choosing the models statically requires much more tuning and testing before the experiment is run than the dynamic model selection. Moreover, there are also configurations of

Table 8.1: Median number (20 runs) of function evaluations needed to reach the specified H_{ratio} on ZDT1, and ZDT2 test problems. The name of the method is encoded as follows: NSGA and IBEA indicate the type of the external evolutionary algorithm (NSGA-II and ϵ-IBEA respectively) and MSE and RP encode the type of selector used to select the best model from the set.

	ZDT1				
H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	5600	18600	19850	20750	21850
NSGA-STATIC	500	1420	1904	2053	5285
NSGA-MSE	544	**816**	1207	1300	6089
NSGA-BIAS	502	837	1172	1402	5789
NSGA-VAR	555	852	**1054**	**1198**	5777
NSGA-RP	**485**	1069	1228	1389	**4843**
IBEA	7400	13750	18200	20000	25550
IBEA-STATIC	**453**	1340	1607	2025	3588
IBEA-MSE	487	1194	1575	1660	2499
IBEA-BIAS	475	**783**	**1095**	**1154**	2382
IBEA-VAR	493	1214	1499	1591	**2337**
IBEA-RP	514	1541	1799	1842	2507

	ZDT2				
H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	650	1650	3550	5050	7900
NSGA-STATIC	**145**	**196**	**274**	**320**	**517**
NSGA-MSE	209	267	367	483	866
NSGA-BIAS	211	275	338	445	843
NSGA-VAR	196	273	335	468	965
NSGA-RP	229	268	342	417	794
IBEA	750	2050	5150	7800	13000
IBEA-STATIC	**151**	**204**	**261**	**295**	**522**
IBEA-MSE	196	236	332	429	990
IBEA-BIAS	203	248	351	415	685
IBEA-VAR	198	257	337	399	823
IBEA-RP	204	259	335	410	983

Table 8.2: Median number (20 runs) of function evaluations needed to reach the specified H_{ratio} on ZDT3 and ZDT6 test problems. The name of the method is encoded as follows: NSGA and IBEA indicate the type of the external evolutionary algorithm (NSGA-II and ϵ-IBEA respectively) and MSE, BIAS, VAR, and RP encode the type of selector used to select the best model from the set.

	ZDT3				
H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	600	1250	4150	7250	–
NSGA-STATIC	**151**	**223**	**317**	**379**	**552**
NSGA-MSE	202	265	366	402	578
NSGA-BIAS	230	289	386	425	621
NSGA-VAR	207	281	355	434	587
NSGA-RP	221	301	413	490	612
IBEA	650	1550	5400	8150	33350
IBEA-STATIC	**155**	**209**	**283**	**328**	**452**
IBEA-MSE	212	284	349	402	549
IBEA-BIAS	219	308	401	493	635
IBEA-VAR	200	282	338	378	547
IBEA-RP	236	303	419	469	632

	ZDT6				
H_{ratio}	0.5	0.75	0.9	0.95	0.99
NSGA	7950	10200	13950	17700	28650
NSGA-STATIC	**1429**	3164	6399	9632	19046
NSGA-MSE	1658	3079	5280	7847	14889
NSGA-BIAS	1724	3436	6636	9442	18663
NSGA-VAR	1854	2964	5384	7948	15166
NSGA-RP	1626	**2602**	**5023**	**7202**	**12756**
IBEA	10300	13650	18400	23150	34050
IBEA-STATIC	**1444**	3509	7233	10543	18297
IBEA-MSE	1952	3961	7053	9522	18208
IBEA-BIAS	2067	5003	10285	13882	25864
IBEA-VAR	1740	3995	6970	9937	18220
IBEA-RP	1555	**3123**	**5599**	**7213**	**11773**

Table 8.3: The behavior of the different methods on the ZDT1 and ZDT2 problem sets. The name of the method is encoded as follows: NSGA and IBEA indicate the type of the external evolutionary algorithm (NSGA-II and ϵ-IBEA respectively) and MSE, BIAS, VAR, and RP encode the type of selector used to select the best model from the set.

		ZDT1	
Function	A Behav.	F0 Behav.	F1 Behav.
NSGA-MSE	D-D/U	C	D
NSGA-BIAS	D-U	C	D-U
NSGA-VAR	D	C	D
NSGA-RP	D	C	D
IBEA-MSE	C	C	D
IBEA-BIAS	D-U	C	D-D/U
IBEA-VAR	C	C	D
IBEA-RP	C	C	D

		ZDT2	
Function	A Behav.	F0 Behav.	F1 Behav.
NSGA-MSE	D-U	C	D/U
NSGA-BIAS	U	C	D/U
NSGA-VAR	D	C	D
NSGA-RP	D	C	D
IBEA-MSE	C	C	D/U
IBEA-BIAS	D/U	C	C/U
IBEA-VAR	C	C	D
IBEA-RP	D	C	D-C

Table 8.4: The behavior of the different methods on the ZDT3 and ZDT6 problem sets. The name of the method is encoded as follows: NSGA and IBEA indicate the type of the external evolutionary algorithm (NSGA-II and ϵ-IBEA respectively) and MSE, BIAS, VAR, and RP encode the type of selector used to select the best model from the set.

		ZDT3	
H_{ratio}	A Behav.	F0 Behav.	F1 Behav.
NSGA-MSE	D	C	D
NSGA-BIAS	U	C	C/U
NSGA-VAR	C	C	D
NSGA-RP	D	C	C/D
IBEA-MSE	D	C	D
IBEA-BIAS	D/U	C	D/U
IBEA-VAR	C	C	D-D/C
IBEA-RP	D	C	D

		ZDT6	
H_{ratio}	A Behav.	F0 Behav.	F1 Behav.
NSGA-MSE	D/U	D	D/U
NSGA-BIAS	U	U	D/U
NSGA-VAR	D/U	C/U	D-D/U
NSGA-RP	D-U	U	D/U
IBEA-MSE	D-U	D/U	D-C/D
IBEA-BIAS	U	U	U
IBEA-VAR	D-U	D/U	D-D/U
IBEA-RP	D-U	D/U	D-C

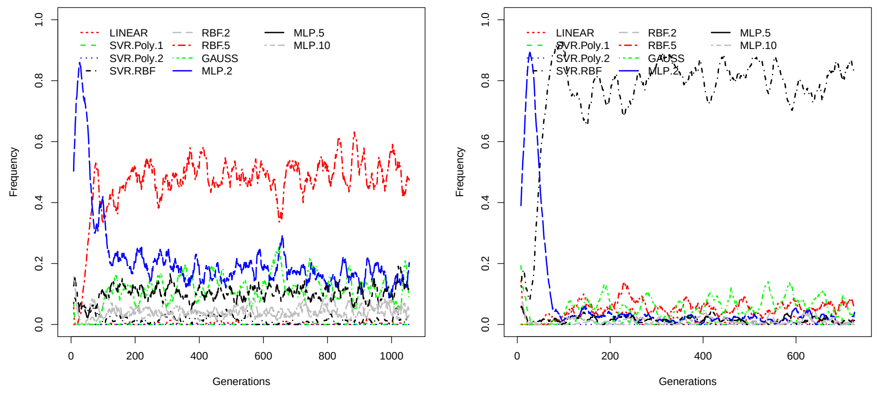


Figure 8.3: The relative frequency of the types of models selected based on the MSE (left) and RP (right) selector. The F1 objective function of the ZDT6 test problem with ϵ-IBEA fitness

statically selected models which require much more objective function evaluations than the dynamically selected ones. In this sense, we can say that the dynamically selected models are more robust than the statically assigned ones.

behaviors for different functions

Regarding the types of behavior, there is an obvious observation: as the F0 objective of the ZDT1, ZDT2, and ZDT3 test is a linear function, the behavior on these is constant – the linear regression model is the best according to all the selectors. For the other modeled functions, the behavior is usually dynamic, often combined with undecided, as there seems to be around three different models which work well and compete with each other, each of them being used with similar frequency. Probably each of the models wins in some of the individual runs and it is used during most of the evolution. This also explains the fact that the behavior changes from dynamic to undecided in a number of runs – in the beginning, there is a single model which is used most often (as the evolution progresses in a similar way in the beginning of each run), but as the evolution progresses, different models are good for each of the runs.

We can also see that the bias selector almost always creates undecided behavior – this selector does not discriminate between the qualities of different models very well, as all the models are able to provide a low bias. This should be expected, as any bias adds to the MSE which all the models should minimize during their training.

The high number of cases, in which we observed the dynamic behavior, indicates the importance of the use of a meta-model selector, as the optimal meta-model changes during the run of the evolution it is impossible to select a single model to be used during the entire run.

numbers of evaluations

Let us now compare the median number of function evaluations needed to attain a specified quality of the solutions expressed as the

Table 8.5: The number of times the specified selector provided the best median value for specified H_{ratio}

H_{ratio}	0.5	0.75	0.9	0.95	0.99
MSE	2	4	1	1	1
BIAS	1	1	1	1	1
VAR	2	1	4	3	2
RP	3	2	2	3	4

H_{ratio}. First, we can see that the bias selector does not perform well, it is significantly better than the other selectors only in one of the eight configurations. This further confirms what we discussed earlier, that this selector does not discriminate among the different types of models, and the different models are selected mostly randomly.

comparison of selectors

The variance and MSE based selectors provide similar performance for most of the test problems. This is again closely connected with the fact that the models have very low bias and therefore the MSE is given mostly by the variance of the error. We originally hoped that the performance of the variance selector would be more similar to the one of the relation preservation selector. In such a case the variance selector could be used instead of the RP one. An algorithm which would minimize the variance of a model should be easier to create as the variance is a continuous number, whereas the RP is discrete.

The RP selector provides the best results for the $H_{ratio} = 0.99$ in half of the cases. These results, especially for ZDT6 are much better than those of the other selectors. ZDT6 is the most difficult of the test problems, and the results we obtained within this work are better than those we were able to obtain in our previous work (ASM-MOMA, LSPS-MOEA) with a single statically selected model.

Table 8.5 shows the number of times each of the selectors provided best results out of the eight tested configurations. We can see that for the lowest H_{ratio} all of the selectors are comparable. In this case the numbers of evaluations are quite low in most of the cases (except ZDT6) and other effects (like the random initialization) are probably more important than the type of selection. On the ZDT6 test problem, the best result is provided by the RP selector. For $H_{ratio} = 0.75$ the MSE selector provides the best result most often but the differences are still quite small. For the higher values of H_{ratio}, the RP and variance selectors provide the best results in six out of the eight configurations.

dynamics

To provide a closer look on the dynamics of the evolution and the types of models selected by different model selectors, we present Figure 8.3. It compares the types of models selected by the MSE and RP selectors. We can see that both selectors select the multilayered perceptron with two neurons in the hidden layer in the beginning of the population, however in the later phases the RBF network with 5 neurons has the lowest MSE in most of the runs, as is therefore selected by the MSE selector, whereas the RP selector chooses a different type of model – the support vector regression with RBF kernels. If we compare the overall results of these two methods, we can see that the model selected by the RP selector leads to much lower number of objective function evaluations that the one selected by the MSE selector.

selected models

Tables 8.6 and 8.7 compare the frequency with which the different models were selected for the modeling of different functions during the evolution, when NSGA-II is used as the external algorithm. We compare the traditional MSE selector with the RP selector here, to show the difference between the traditional and proposed approach. In this table, F_0 and F_1 denote the two objectives of the ZDT problems and AGR denotes the aggregated distance based surrogate model used during the local search.

There is an obvious observation: as the objective F_0 is linear for the ZDT1 to ZDT3 functions, the linear regression model is selected every time as it can learn the function precisely.

Regarding the other function, we can see that the RP selector tends to select simpler models – like the support vector regression or linear regression, rather than the multilayer perceptron. These are more often selected by the MSE selector. This indicates that although the more complicated models can learn the functions more precisely they are not able to predict the comparisons well – they underestimate for some of the points and overestimate for others.

It is also interesting to note that Gaussian process regression, which is quite often used in applications, is only rarely selected in these cases, except for the more complicated ZDT6 problem. Moreover, it is almost never selected to model the aggregate surrogate model.

8.4 CONCLUSIONS

We have shown that the way in which meta-models are selected for the use with evolutionary algorithm greatly affects the performance. Moreover, the traditional way of minimizing the MSE may not be the best, as other criteria, like the relation preservation, may be more important and express the suitability of a particular model for use with evolutionary algorithm better.

We have also shown that the best model may change wildly during the run of the evolution, and there may be a different suitable model in each of the phases of the evolution, based not only on the progress towards the optima, but also on the particular state of the algorithm. The best model may be different in different independent runs of the algorithm. Therefore a kind of automatic selection is necessary to ensure good convergence speed and also the quality of the individuals.

Table 8.6: Relative frequencies of selected models when NSGA-II is used as the external algorithm for the ZDT1 and ZDT2 problems.

ZDT1

Function	AGR		F0		F1	
Model sel.	MSE	RP	MSE	RP	MSE	RP
LINEAR	0.15	0.16	1.00	1.00	0.20	0.27
SVR-Poly-1	0.17	0.33	0.00	0.00	0.18	0.13
SVR-Poly-2	0.00	0.00	0.00	0.00	0.00	0.00
SVR-RBF	0.20	0.23	0.00	0.00	0.01	0.00
MLP-2	0.15	0.00	0.00	0.00	0.24	0.24
MLP-5	0.15	0.03	0.00	0.00	0.20	0.17
MLP-10	0.14	0.15	0.00	0.00	0.14	0.15
RBF-2	0.00	0.00	0.00	0.00	0.00	0.00
RBF-5	0.00	0.00	0.00	0.00	0.00	0.00
GAUSS	0.00	0.06	0.00	0.00	0.00	0.00

ZDT2

Function	AGR		F0		F1	
Model sel.	MSE	RP	MSE	RP	MSE	RP
LINEAR	0.15	0.16	1.00	1.00	0.22	0.21
SVR-Poly-1	0.11	0.35	0.00	0.00	0.14	0.13
SVR-Poly-2	0.00	0.00	0.00	0.00	0.00	0.00
SVR-RBF	0.30	0.29	0.00	0.00	0.09	0.08
MLP-2	0.06	0.04	0.00	0.00	0.10	0.13
MLP-5	0.11	0.02	0.00	0.00	0.17	0.12
MLP-10	0.22	0.08	0.00	0.00	0.25	0.30
RBF-2	0.00	0.00	0.00	0.00	0.00	0.00
RBF-5	0.00	0.00	0.00	0.00	0.00	0.00
GAUSS	0.01	0.01	0.00	0.00	0.00	0.00

Table 8.7: Relative frequencies of selected models when NSGA-II is used as the external algorithm for the ZDT3 and ZDT6 problems.

ZDT3

Function	AGR		F0		F1	
Model sel.	MSE	RP	MSE	RP	MSE	RP
LINEAR	0.14	0.17	1.00	1.00	0.22	0.30
SVR-Poly-1	0.17	0.28	0.00	0.00	0.16	0.39
SVR-Poly-2	0.00	0.00	0.00	0.00	0.00	0.00
SVR-RBF	0.24	0.31	0.00	0.00	0.32	0.15
MLP-2	0.14	0.02	0.00	0.00	0.10	0.05
MLP-5	0.18	0.03	0.00	0.00	0.11	0.02
MLP-10	0.08	0.12	0.00	0.00	0.04	0.02
RBF-2	0.00	0.00	0.00	0.00	0.00	0.00
RBF-5	0.00	0.00	0.00	0.00	0.00	0.00
GAUSS	0.01	0.04	0.00	0.00	0.01	0.03

ZDT6

Function	AGR		F0		F1	
Model sel.	MSE	RP	MSE	RP	MSE	RP
LINEAR	0.14	0.05	0.00	0.10	0.01	0.27
SVR-Poly-1	0.31	0.16	0.00	0.03	0.00	0.12
SVR-Poly-2	0.00	0.00	0.00	0.00	0.00	0.00
SVR-RBF	0.08	0.22	0.00	0.05	0.09	0.31
MLP-2	0.15	0.06	0.35	0.15	0.30	0.04
MLP-5	0.12	0.13	0.24	0.23	0.06	0.02
MLP-10	0.13	0.17	0.17	0.24	0.00	0.02
RBF-2	0.00	0.03	0.01	0.00	0.05	0.02
RBF-5	0.01	0.06	0.02	0.01	0.32	0.01
GAUSS	0.01	0.08	0.18	0.16	0.12	0.14

In the future, the proposed framework shall use a meta-learning tool, which would provide a list of models that shall be considered during the evolution. This list can be recommended either once in the beginning of the evolution based on some information about the objective function, numbers of variables and objective etc., or after each generation, with the additional knowledge of the precise values of previously evaluated individuals (i.e. the training set, and its statistical properties).

integration with meta-learning

It may also be beneficial to create training algorithm which would optimize the relation preservation or the variance of the given model directly, instead of optimizing the models for the lowest MSE and then selecting according to a different criteria.

9 MULTI-OBJECTIVIZATION FOR HYPER-PARAMETER TUNING

This chapter is in a sense complementary to the previous one. In the previous chapter, we studied how meta-learning can improve the performance of multi-objective tuners by recommending better models. In this chapter, we show how multi-objective optimization can improve the performance of machine learning models by the tuning of their parameters. We focus on the case of classifier models in this chapter as that is what we mostly studied while developing the Pikater system (Kazík et al., 2012), however most of the ideas can be applied even for regression models.

9.1 MOTIVATION

parameter tuning

Most classifiers have at least a few parameters, which more or less affect their performance on a given data set. The values of these parameters need to be set before the classifier can be used to solve a particular task. The problem is, that the best values of the parameters differ for different data sets. Although there are usually some guidelines for the parameter settings, finding the optimal values often requires expert knowledge, intuition, and trial-and-error approach. If we also add the task of selecting the right type of classifier, the problem is even more complicated.

meta-learning

Meta-learning is the part of machine learning which deals with this problem. It can be usually divided into two parts as already indicated above: the selection of a good classifier type, and the tuning of the parameters of the selected classifier to the data set at hand.

classifier selection

The selection of a good classifier for a particular data set is often performed based on a prior knowledge. Usually, the performance of a set of classifiers on different data sets is known in advance, as well as some meta-data about the data sets (e.g. number of instances, number of classes, types of attributes). It is assumed that classifiers perform similarly on data sets with similar meta-data, thus a good classifier (or a set of promising classifiers) is selected based on the performance on similar data sets (Brazdil et al., 1994; Kazík et al., 2012). This step can also additionally provide some initial values (or ranges) for the parameters of the classifier.

However, having selected the right classifier is only part of the task. The next vital step is to find the values of the parameters of the classifier, which provide the best performance on the given data set. And this is the role of the parameter tuner.

fair comparison

Parameter tuning is not only important in the field of meta-learning, but also when a new classifier is developed. In this case the classifier should be compared to different competing classifiers. We argue that for the comparison to be fair, both classifiers have to be compared with the values of their parameters that ensure the best results on a given data set.

In this chapter we deal with the problem of tuning of the parameters of Radial Basis Function (RBF) Networks and Multilayer Perceptron (MLP) (Haykin, 1999) networks with the goal to provide settings which minimize the classification error. To this end we use the concept of multi-objectivization – solving the single-objective problem by means of multi-objective optimization. We describe our multi-objectivization approach, and compare it to the performance of single-objective evolutionary algorithm. Moreover, surrogate versions of both the single-objective and multi-objective algorithms are discussed.

chapter outline

The rest of the chapter is organized as follows: in the next section we discuss the largest challenges posed on evolutionary algorithms when they are used to solve the problem of parameter tuning, then we provide an overview of existing approaches from the literature. Section 9.4 describes our multi-objectivization approach. and in Section 9.5 we present the experiment setup and the data sets we used for testing. Section 9.6 provides the results of the different methods and, finally, Section 9.7 concludes this chapter.

9.2 THE CHALLENGES

Using evolutionary algorithms to optimize the parameters of classifiers is not an easy task. In fact, this can easily be one of the most challenging fields for the application of evolutionary algorithms. Despite this fact, quite a lot of researchers have tried to use evolutionary algorithms to solve this task.

What are the most difficult challenges the evolutionary algorithm must face to tackle this problem? First of all, most of the classifiers provide similar (or even the same) results for lots of different settings of their parameters, this leads to a search space, where most of the points (i.e. different parameter settings) yield the same error rate Reif et al. (2012). This problem is even more pronounced if the given classification problem has only a small number of instances. In this case, there are only a few different error rate values possible. Thus, the fitness function is piecewise constant, sometimes with only a small area of a local optima – exactly the type of fitness function which is difficult to optimize using the evolutionary algorithms. The magnitude of the problem can also be seen in Table 9.1 in Section 9.4.

plateaus in search space

training complexity

These properties of the search space are not the only problem. Machine learning algorithms also require quite a long time to train (especially on larger data sets), and evolutionary algorithms often need a large number of fitness function evaluations to find a good point in the feature space. Moreover, parameter tuning often calls for the use of cross-validation to improve the generalization properties of the given machine learning techniques. And it increases the number of trainings and evaluations even more. Thus, the whole parameter tuning process may require large amount of computational resources, if this aspect is not taken into account. To this end we can also apply the idea of surrogate modeling from the previous chapters.

The most straightforward way to apply the surrogate modeling for the problem of parameter tuning would be to model the classification error of the classifier based on its parameters. However, as we have already discussed (the first challenge above), most of the parameter settings yield similar values. This can be problem for some of the surrogate models.

9.3 RELATED WORK

The first attempts at parameter tuning were specifically designed for a given type of classifier, for example there are several algorithms to optimize the parameters of SVM (Chapelle et al., 2002; Qilong Zhang et al., 2009; Kapp et al., 2009).

tuned data mining

Despite the challenges described in the previous section, several researchers have approached the problem of optimizing the parameters of a given classifier using surrogate-based optimization techniques. For example, Konen et al. (2011) proposed a framework for Tuned Data Mining. The framework contains both feature selection and parameter tuning. Both of these tasks are performed at once. It also supports cases when there are different costs for different classification errors. The parameter tuning is done in the SPOT (Bartz-Beielstein et al., 2005) framework, which also uses surrogate modeling. The framework only deals with the problem of parameter tuning, no recommending is performed.

deep networks tuning

Bergstra et al. (2011) have used parameter tuning to enhance the performance of Deep Belief Networks (DBN). They used surrogate assisted evolutionary algorithm and showed that it outperforms both manual setting of parameters and random search on two data sets from the image recognition domain. They use Gaussian processes to model the error and compare them to their proposed Tree Structured Parzen Estimator. The main difference between the two modeling techniques is that Gaussian processes predict the probability distribution of $P(y|x)$ (y is the target value and x are the parameters) whereas the Tree Structured Parzen Estimator predicts the distributions of $P(x|y)$ and $P(y)$. Also the latter model contains information about the structure of the DBN.

multi-objective regularization

Multi-objective optimization has also been used to tune the parameters of optimizers for various reasons. One of them is regularization.

Traditionally, regularization was performed by adding an additive term with a pre-specified weight to the error function of the classifier. This additional term ensures some desirable properties of the classifier – usually good generalization, or (similarly) low complexity of the model. In the framework of multi-objective optimization such regularizing term can be considered as another objective, and both the objectives (the error rate of the classifier and its complexity) are optimized at once (Jin, 2006). Multi-objective optimization is also used if there are conflicting error measures, all of which shall be optimized at once. One example of such conflicting measures may be sensitivity and specificity.

multi-objectivization

Another possibility to use multi-objective optimization for parameter tuning is so called multi-objectivization. Brockhoff et al. (2009) showed that adding another objective may lead to a better performance even if only one objective is important. This is especially true for functions with plateaus, where the additional objective may increase the selection pressure. If the additional objective is correlated with the original one, this can lead to improved convergence rate and results. Hohm and Zitzler (2009) used a similar idea to optimize a model of a gene regulative network.

meta-learning for initialization

Reif et al. (2012) used a completely different technique to enhance the performance of evolutionary algorithms for parameter tuning. In their case, they used the the ideas from meta-learning to create the initial population. Namely, they have used the several best performing settings on similar datasets as the individuals in the initial population. They claim that this helps to start the evolution in the regions of the search space, where the more promising individuals are located and show that the performance of the tuned classifiers is similar to the performance of the classifiers tuned by a grid search algorithm, which uses much more iterations of the tuning. The disadvantage is that the method needs a database of results of the given classifier on several datasets and thus cannot be used without any prior knowledge.

9.4 MULTI-OBJECTIVIZATION

Our goal is to provide good parameter settings for a given classifier. The quality of the settings is measured by the classification error of the classifier. As discussed above, the classification error (the number of incorrectly classified instances) of a classifier is a function which is difficult to optimize using evolutionary algorithms.

In order to improve the convergence rate of the algorithm, we use the idea of multi-objectivization. We add two more objectives whose values are not important for our task, but which are correlated to the error rate of the optimizer. Then, a multi-objective evolutionary algorithm is used to solve the multi-objective optimization problem.

The three objectives we optimize are:

1. classification error (minimization) – expresses the percentage of incorrectly classified instances, i.e. of the instances whose class predicted by the model is different from their class as assigned in the training set, *classification error*
$$CE = \frac{A - C}{A},$$
where C is the number of correctly classified instances and A is the number of all instances;

2. kappa statistic (Di Eugenio and Glass, 2004) (maximization) – expresses the inter-classifier (or classifier and training set in our case) agreement, it is computed as *kappa statistic*
$$\kappa = \frac{P(a) - P(e)}{1 - P(e)},$$
where $P(a)$ is the observed agreement between the two classifiers and $P(e)$ is the calculated chance agreement (i.e. the agreement two random classifiers would have if they had the same distributions of classes as the actual classifiers);

3. root mean square error (minimization) – is traditionally used for regression tasks. For classification, the class indices are encoded as binary vectors with just one 1 on the position of the class index (e.g. if there are 5 classes in total, the vector for class 3 would be $\langle 0, 0, 1, 0, 0 \rangle$); *root mean square error*
$$R = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(c_{ij}^{*} - c_{ij})^2},$$
where n is the number of instances in the training set, m is the number of classes (numbered 1 to m), and $c_{ij}^{*} = 1$ if individual i belongs to the class j and 0 otherwise, and c_{ij} is the output of the classifier.

The error rate and kappa statistic are highly correlated, especially in cases where all the classes are represented by the same number of instances in the training set. This is also the reason to add the third objective, which may seem unrelated to classification.

Root mean square error (RMSE) is traditionally used as the objective which is minimized in regression tasks. As it is implemented here (i.e. the model is trained to predict the unary representation of the class label), the number does not tell much about the quality of the classifier itself. However, it is more sensitive to changes in the parameter settings. It is important to note that the only important measure is the classification error, so whenever two models have the same classification error we may choose any of them and they are

Table 9.1: Number of different values of the error measures compared to the different sets of options (data obtained during the experiments described in Section 9.5).

	Number of distinct values of			
	CE	κ	R	options
balance-scale	253	3255	23953	27636
breast-w	174	671	24223	27639
car	206	9843	12655	18417
haberman	33	640	22572	27591
iris	98	98	24995	27629

equivalent for the given task, thus choosing the one with lower RMSE does not hurt the result. Moreover, RMSE guides the optimization to an optima of the classification error – if the RMSE is zero the classification error is also zero. This is the reason why RMSE provides a sensible guide for the search.

discriminative power

The relative discriminative power of the different error measures is summarized in Table 9.1. We can see that more than 25,000 different sets of options produce only hundreds different values of classification error. The other error measures provide much more different values and thus have more discriminative power, which helps to guide the evolutionary algorithm. The numbers in the table were obtained during some of the experiments which are described in the Section 9.5.

9.5 EXPERIMENTS

algorithms

To investigate the effect of multi-objectivization on the performance of hyper-parameter tuners we compare a simple single-objective tuner to NSGA-II (Deb, Pratap, et al., 2002) based multi-objective tuner. Moreover, we also compare surrogate versions of both the tuners. In the single-objective case the surrogate model is trained to predict the classification error given the parameters. In the multi-objective case we use the LSPS-MOEA we described earlier.

Pikater system

To run the experiments, we used the Pikater (Kazík et al., 2012) system. It is a multi-agent system for meta-learning and parameter tuning. It contains several types of agents, among them computational agents, which encapsulate the classifiers from the Weka (Hall et al., 2009) framework. These agents are able to train the method they encapsulate on a given data set and report the performance (different error measures) on that particular data set. Moreover, Pikater also contains a general search agent, which is able to search the options of the computational agents.

We implemented a specialized search agent, which runs the algorithms described above. Moreover, it is able to run a simple evolutionary algorithm without any surrogates, single-objective surrogate algorithm, NSGA-II and LSPS-MOEA.

optimized classifiers

We used this search agent to tune the parameters of the methods given bellow, together with their parameter settings.

- RBF network – an artificial neural network with (in our case) normalized Gaussians as the activation function, the output is the value of logistic function applied to the linear combination of the outputs of individual neurons. We optimized four parameters:
  - the number of clusters – integer between 2 and 10,
  - the minimal width of the Gaussians – real number between 0.01 - 1.0,
  - the ridge parameter for the logistic regression – real number between 0.000000001 and 10,
  - and the maximum number of iterations for the logistic regression – integer between -1 and 50, -1 meaning "until convergence",
- MLP network – a feed-forward artificial neural network, consisting of individual units with sigmoid function as the activation function. We again optimized four parameters:
  - the learning rate – real number between 0.001 and 1.0,
  - the momentum – real number between 0.0 and 0.9,
  - the number of epochs to train through – integer between 1 and 10,000,
  - the percentual size of the validation set, which is used to terminate the training – integer between 0 and 100.

  The other parameters of the multilayer perceptron were fixed, especially the architecture. The neural network had one hidden layer. The number of neurons in this layer is the arithmetic average of the number of attributes and number of classes.

rationale for options

The ranges for the number of clusters of the RBF network was set in a range which corresponds to our experience with this particular model. The number of neurons in the hidden layer of the multilayer perceptron corresponds to the defaults of the Weka framework. Other ranges contain all (meaningful) values for the given parameters.

datasets

The performance of the tuners was tested on the following five datasets which are available from the UCI Machine Learning Repository (A. Asuncion, 2007).

- balance-scale – the goal it to predict the direction of the balance scale tip (left, right, balanced), given four numerical attributes, the data set was created to model a psychological experiment,
- breast-w – the goal is to predict the type of tumor (benign, malign) given ten numerical attributes, the features are computed from digitized images,
- car – the goal is to predict the car acceptability (in four different levels) given six nominal attributes, the data set is derived form a simple hierarchical model
- haberman – the goal is to predict the survival status of cancer patients, given three numerical attributes, the data set contains cases of cancer from a hospital
- iris – the goal is to predict one of three types of iris plants, given four of its attributes, this is probably the most well-known data set.

experimental settings

To obtain the results, the tuners were given the computational budget of 300 objective function evaluations. One evaluation is a 10-fold cross-validation with the parameters given by the tuner on the respective training set. All evaluated individuals are saved in an archive and if the same individual is generated again in the same run it is not re-evaluated. The archive of evaluated individuals is also used during the training of the surrogate model in the case of surrogate evolutionary algorithm.

The single-objective evolutionary algorithm (both with and without surrogate modeling) uses population of 10 individuals, with one point crossover and Gaussian mutation (a random number from the normal distribution with standard deviation equal to 30% of the range of the particular parameter is added to the parameter). Moreover, the evolutionary algorithm uses tournament selection and 10% elitism (one individual). The version with surrogate model uses Gaussian processes as the surrogate model. The local search operator is applied with the probability of 0.25 and uses another evolutionary algorithm with the same population size and operators which is run for 10 generations (only the standard deviation of the Gaussian mutation is reduced to 10% of the range to promote exploitation). The surrogate operator is not used, if there are less than 20 evaluated individuals in the archive, as low number of individuals would lead to poorly trained models.

The non-surrogate multi-objective evolutionary algorithm uses the same parameters, the only difference being in the selection phase, where it uses the NSGA-II selection based on dominance and crowding distance.

The parameters of LSPS-MOEA match those of the surrogate evolutionary algorithm, it also uses the automated model selection we described in the previous chapter. Models with good relation preservation are used. The surrogate modeling is not used, if there are less than 50 evaluated individuals. In this case, the surrogate modeling seems to be more sensitive to the lack of training data and larger training sets improve the results.

9.6 RESULTS

general observations

Generally, the results (the last generation shown numerically in Tables 9.2 and 9.3). correspond to the discussion we presented earlier. It is difficult for the surrogate model to precisely approximate the classification error of the optimizer, thus the surrogate model does not improve the performance of the optimizer much. In some instances during preliminary testing, we have observed that the local search operator guides the search towards the boundaries of the search space, thus generating the same individuals in each run. This problem was partially reduced when we lowered the number of generations for the internal algorithm to 10 (the value used in final testing, we started with 30). We also experimented with different types of surrogate models for the single-objective EA, like RBF networks, and linear regression, however the Gaussian processes seemed to give the best and most robust results.

However, we can observe, that the results of the surrogate algorithm are often better with respect to the worst and average run – this indicates that the surrogate versions of the algorithms have more robust performance, thus providing better guarantee on the optimality of the found solution. This is a feature which may be of importance in practice, where running the experiment several times to get better results may be unacceptable or intractable.

On the other hand, as can be seen from the table, the difference between the single-objective and multi-objective version of the optimizers is significant. The multi-objective optimizers perform better than their single-objective version in almost all the cases.

9.6.1 *Optimizing the RBF networks*

RBF network results

For the tests with the optimization of RBF networks, we can observe, that the multi-objective tuners work better than both versions of the single-objective tuners. The surrogate tuners usually outperform the non-surrogate one, however the difference between the surrogate and non-surrogate versions is rather small in the case of the best found setting. On the other hand, the performance in the average and worst cases is usually better for the surrogate based tuners. On two of the five datasets the worst performance of the surrogate version is better than the average performance of the non-surrogate one.

Table 9.2: Results of the four different methods after 300 evaluations. Optimization of the parameters of the RBF network. Numbers are aggregated over 10 runs.

	Simple EA			Surrogate EA		
	best	average	worst	best	average	worst
balance-scale	0.0512	0.0712	0.1056	0.0528	0.0739	0.1040
breast-w	0.0286	0.0308	0.0329	0.0300	0.0315	0.0329
car	0.0741	0.0779	0.0839	0.0723	0.0802	0.0938
haberman	0.2386	0.2474	0.2516	0.2320	0.2412	0.2484
iris	0.0200	0.0300	0.0333	0.0267	0.0287	0.0333
	Multi-objective EA			**LSPS-MOEA**		
	best	average	worst	best	average	worst
balance-scale	**0.0464**	0.0498	0.0528	**0.0464**	**0.0494**	**0.0512**
breast-w	0.0286	0.0296	0.0315	**0.0272**	**0.0285**	**0.0286**
car	0.0712	0.0737	0.0787	**0.0700**	**0.0719**	**0.0723**
haberman	**0.2288**	**0.2395**	0.2516	0.2353	**0.2395**	**0.2451**
iris	**0.0133**	**0.0200**	**0.0267**	0.0200	0.0220	**0.0267**

More specifically, on the balance-scale data set we can observe the largest difference between the single-objective and the multi-objective optimizers. For this data set the multi-objective optimizers clearly wins and, moreover, closer inspection of the convergence speed reveals, that after approximately 50 evaluations the multi-objective optimizers provide results, which are not provided by the single-objective optimizers even after 300 evaluations.

The haberman data set is the only one, where the surrogate optimizers perform much better (in the best case) than the non-surrogate optimizers and the multi-objective optimizers gets the best results overall. However, it is outperformed by LSPS-MOEA in the average and worst cases.

On the breast-w data set all the methods work similarly well during the first 50 evaluations of the objective function, however, surrogate based EA finds a local optima soon after that mark and does not improve much later, the non-surrogate EA improves only slowly. The multi-objective evolutionary algorithms can, thanks to the other objectives, avoid the local optima and are able to improve the results further. In this case, the worst performance of LSPS-MOEA matches the best performance of the non-surrogate multi-objective evolutionary algorithm.

Finally, on the iris data set, the multi-objective optimizer was able to find a setting which was better than all settings we were able to

Table 9.3: Results of the four different methods after 300 evaluations. Optimization of the parameters of the MLP network. Numbers are aggregated over 10 runs.

	Simple EA			Surrogate EA		
	best	average	worst	best	average	worst
balance-scale	0.0752	0.0776	0.0816	0.0736	0.0787	0.0848
breast-w	0.0300	0.0316	**0.0329**	0.0300	0.0319	0.0343
haberman	0.2288	0.2337	0.2386	0.2320	0.2343	0.2418
iris	**0.0200**	**0.0207**	**0.0267**	**0.0200**	0.0247	**0.0267**
	Multi-objective EA			**LSPS-MOEA**		
	best	average	worst	best	average	worst
balance-scale	**0.0688**	0.0746	**0.0784**	**0.0688**	0.0747	0.0800
breast-w	**0.0286**	**0.0313**	**0.0329**	0.0315	0.0319	**0.0329**
haberman	0.2288	0.2330	0.2386	**0.2255**	**0.2314**	**0.2353**
iris	**0.0200**	0.0213	**0.0267**	**0.0200**	**0.0207**	**0.0267**

obtain previously (using different optimizers like simulated annealing, grid search and random search). In the later phases, the average performance of the multi-objective optimizer in this case is the same as the best performance of the single objective optimizers, and it is even better than the performance of LSPS-MOEA.

9.6.2 *Optimizing the MLP networks*

MLP networks results

MLP networks take much longer to train than RBF networks. This is also the reason, why we omit the results on the largest data set – car – here. Single cross-validation in this case takes several minutes, thus we would need several days to obtain the results (as a side note, it took approximately two hours to get all the results for all datasets and optimizers for the RBF network).

In the case of MLP networks the differences among the methods are much lower than in the case of RBF networks. It may be the case that MLP networks are less sensitive to the particular settings of the parameters we tried to optimize. Table 9.3 also indicates that all the optimizers were able to find similar optima on some of the datasets, which may indicate these are the true optima given the fixed parameters.

When comparing the convergence speed of the different optimizers more closely, we can see that on the iris data set the multi-objective surrogate was able to provide the best results for almost all of the time, only in the end the single-objective non-surrogate optimizer got a slightly better result. However, the differences are so small we can neglect them. Similar, but reversed, situation may be observed on the haberman data set, where the multi-objective optimizer was able to find the best results (on average) despite the fact it was between the two single-objective optimizers for most of the run.

On the other hand, for the balance-scale data set we observe relatively large differences among the different parameter tuners. Again, as we have already seen in the case of RBF networks, the multi-objective optimizers provide by far the best results, with the two single-objective optimizers having similar performance.

9.7 CONCLUSION AND FUTURE WORK

We have shown that using multi-objectivization for the parameter tuning may be more useful than using surrogate modeling. It is caused mainly by the specific type of the fitness function to be modeled, which leads to poorly trained models. On the other hand, adding more objectives, which are not in fact directly important for the optimization task at hand, may improve the results.

We have also shown that surrogate modeling does not improve the results much, but it is able to provide better results in the average and worst cases, which implies better robustness of the results – this may also be important in practice, where running each experiment several times to get good results is not acceptable.

Some of the ranges of the parameters we used were quite limited. This corresponds to our future usage of parameter tuning – we would like to use meta-learning which would provide us with the model type, together with the region of interest of its parameters, and then we will use the parameter tuning to find the best possible settings of its parameters.

Part IV

CONCLUSION

In this last part of the thesis we again summarize the most important results and provide ideas and directions for future research.

10 CONCLUSIONS AND FUTURE WORK

In this thesis, we studied surrogate based multi-objective algorithms. The main contributions are:

1. the new distance-based surrogate model – the experiments show that multi-objective evolutionary algorithms based on this model perform well on the large set of benchmark functions and are able to significantly reduce the number of function evaluations needed to attain a specified quality of solutions; or, from the complementary point of view, produce significantly better solutions within a given budget of objective function evaluations,

distance-based models

2. the new pre-selection scheme – the pre-selection scheme based on a different types of models reduces the number of function evaluations further, thanks to the fact that non-promising individuals are ignored and are never evaluated by the real expensive objective function; we have demonstrated that different types of models in each of the phases are essential to ensure better convergence and also to provide more than one individual in each generation; this is important in practice, as it allows for easy parallelization of the evaluations and also, in some cases, evaluating more solutions at once may be more economical,

pre-selection

3. the study of different model selectors – we have shown that the traditionally important feature of good models, low mean square error, may not be the best feature for the model selection in evolutionary algorithms which use only comparisons among individuals during their run; the ranking preservation criterion provides better results and should thus be preferred, and

model selection

4. the multi-objectivization approach for hyper-parameter tuning – we have demonstrated that the multi-objectivization approach improves the hyper-parameter tuning of two classifiers when the goal is to find the hyper-parameters which minimize the classification error; the additional objectives provide direction for the evolutionary algorithm in the areas of search space where the classification error is constant – and there are a lot of such areas as most classifiers were designed in such a way to be robust and provide good performance without being too sensitive to the particular values of the hyper-parameters.

parameter tuning

There are several directions which can be researched to improve on the results we presented here. One of them is derived from the fact that the distance based models may have problems in cases, when

better models

some of the directions in the search space are more important than others. We have seen demonstration of these problems in the experiments on the WFG benchmark, where the different variables have differently scaled domains and we could see, that even the scaling of all the variables before the use of the model can dramatically improve the performance. MO-CMA-ES goes a bit further in solving this problem, it adapts the covariance matrix used to generate new individuals. The distance-based model may profit from a similar idea. If the Euclidean distance we used is replaced by Mahalonobis distance, the covariance matrix (in case the distance based model is used in conjunction with an algorithm which uses one) can be used even for the computation of distances.

Another interesting possibility is to study the options how to reduce of the number of times the model is trained. This is a problem especially for more expensive local models. These are trained multiple times in each generation. One possibility could be to cluster the individuals before the model is constructed and to create a single local model for all the individuals in the cluster.

Another open question is the effect of the degree of locality (represented by the λ parameter) on the evolution convergence speed and the possibility to change this parameter adaptively. Moreover, local and global meta-models might be combined: the global meta-model may be used to pre-evaluate the individuals, and some of them might then be locally improved with a memetic operator based on a local model. The question is, whether it would be more beneficial to improve those individuals which already have good values, or those, which are worse.

better models

The models we used in the pre-selection phase were rather simple and in the light of the experiments we did with model selection, it may be interesting to try models which would optimize the ranking preservation directly, like the one used by Loshchilov et al., 2010b. Also, different criteria for the selection of individuals should be considered. We selected those individuals which are not dominated by any of the parents. This criteria may be rather strong, and weaker criteria, which would select more individuals would decrease the overhead of the algorithm. On the other hand, they may reduce the convergence speed. It may also be interesting to incorporate the Gaussian processes in this case and select individuals similarly to the way they are selected in MOEA/D-EGO – such that they maximize the expected improvement.

expected improvement

Moreover, one of the largest disadvantages of the pre-selection step is that it selects different number of individuals in each generations. The pre-selection step only guarantees (in the way we had it implemented) that at least 10% of the population size are selected in each step. Although it is probably better than selecting only one individ-

better pre-selection

ual, being able to select a pre-defined number of individuals would be more practical.

The experiments with model selectors show that preference preserving is an important feature of surrogate models which are used inside an evolutionary algorithm. Unfortunately, there are not many models, except the ranking SVM, which would try to learn the ranking among the individuals. It may be an interesting research idea to develop more such models. It may be possible to change the training algorithms of some of the existing models to favor the relation preservation, although it may be quite difficult, as the measure is non-differentiable and thus gradient-based techniques cannot be used. An interesting option may be to use an approach similar to the multi-objectivization one, with the primary objective being the relation preservation, and the mean square error would be in this case only a secondary objective (the one most model can optimize using their respective training algorithms).

preference-preserving models

The multi-objectivization idea also provides a prospective direction for research. It may be interesting to come up with different types of secondary objectives. An interesting option may be to use different costs for different miss-classification, and using the errors provided by these to guide the search. It would certainly be more natural in the case of classification than the use of mean squared error. Another interesting direction would be to study the application of multi-objectivization for regression tasks. In this case, the error function provides better direction for the search, as it is continuous, but it may still be possible to improve the convergence with some additional objectives.

multi-objectivization for regression

The main motivation for the multi-objectivization and model selection research was the integration of surrogate modeling and meta-learning. The meta-learning should be able to recommend types of models with good performance to the multi-objective optimizer given some metrics calculated on the training sets. On the other hand, the multi-objective algorithm, with the multi-objectivization approach, can improve the recommendation of the models by integrating the parameter tuning into the meta-learning process. The work on the integration of surrogate multi-objective optimization and meta-learning is currently on-going and should continue in the future.

integration with meta-learning

BIBLIOGRAPHY

A. Asuncion, D.J. Newman (2007). *UCI Machine Learning Repository* (cit. on p. 117).

Auger, Anne, Johannes Bader, and Dimo Brockhoff (2010). "Theoretically Investigating Optimal μ-Distributions for the Hypervolume Indicator: First Results for Three Objectives." In: *Parallel Problem Solving from Nature, PPSN XI*. Ed. by Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph. Vol. 6238. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 586–596. ISBN: 978-3-642-15843-8. DOI: 10.1007/978-3-642-15844-5_59 (cit. on p. 10).

Auger, Anne, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler (2009). "Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point." In: *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*. FOGA '09. Orlando, Florida, USA: ACM, pp. 87–102. ISBN: 978-1-60558-414-0. DOI: 10.1145/1527125.1527138 (cit. on pp. 10, 81).

Bader, Johannes and Eckart Zitzler (2011). "HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization." In: *Evolutionary Computation* 19.1, pp. 45–76. DOI: 10.1162/EVCO_a_00009 (cit. on p. 10).

Bartz-Beielstein, Thomas, Christian Lasarczyk, and Mike Preuss (2005). "Sequential parameter optimization." In: *Congress on Evolutionary Computation*. IEEE, pp. 773–780. ISBN: 0-7803-9363-5. DOI: 10.1109/CEC.2005.1554761 (cit. on p. 113).

Bergstra, James, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl (2011). "Algorithms for Hyper-Parameter Optimization." In: *NIPS*. Ed. by John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, pp. 2546–2554. DOI: http://books.nips.cc/papers/files/nips24/NIPS2011_1385.pdf (cit. on p. 113).

Beume, Nicola (Dec. 2009). "S-metric calculation by considering dominated hypervolume as klee's measure problem." In: *Evol. Comput.* 17.4, pp. 477–492. ISSN: 1063-6560. DOI: 10.1162/evco.2009.17.4.17402 (cit. on p. 10).

Beyer, Hans-Georg and Hans-Paul Schwefel (2002). "Evolution strategies - A comprehensive introduction." In: *Natural Computing* 1.1, pp. 3–52. DOI: 10.1023/A:1015059928466 (cit. on p. 26).

Bezdek, James C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers. ISBN: 0306406713 (cit. on p. 37).

Brazdil, Pavel, João Gama, and Bob Henery (1994). "Characterizing the applicability of classification algorithms using meta-level learning." In: *Proceedings of the European conference on machine learning on Machine Learning*. ECML-94. Catania, Italy: Springer-Verlag New York, Inc., pp. 83–102. ISBN: 3-540-57868-4 (cit. on p. 111).

Bringmann, Karl and Tobias Friedrich (2010). "Approximating the volume of unions and intersections of high-dimensional geometric objects." In: *Comput. Geom.* 43.6-7, pp. 601–610. DOI: 10.1016/j.comgeo.2010.03.004 (cit. on p. 10).

Brockhoff, Dimo, Tobias Friedrich, Nils Hebbinghaus, Christian Klein, Frank Neumann, and Eckart Zitzler (June 2009). "On the effects of adding objectives to plateau functions." In: *Trans. Evol. Comp* 13.3, pp. 591–603. ISSN: 1089-778X. DOI: 10.1109/TEVC.2008.2009064 (cit. on p. 114).

Chapelle, Olivier, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee (Mar. 2002). "Choosing Multiple Parameters for Support Vector Machines." In: *Mach. Learn.* 46.1-3, pp. 131–159. ISSN: 0885-6125. DOI: 10.1023/A:1012450327387 (cit. on p. 113).

Cortes, Corinna and Vladimir Vapnik (1995). "Support-vector networks." English. In: *Machine Learning* 20 (3), pp. 273–297. ISSN: 0885-6125. DOI: 10.1007/BF00994018 (cit. on p. 16).

Deb, Kalyanmoy and Ram B Agrawal (1994). "Simulated Binary Crossover for Continuous Search Space." In: *Complex Systems* 9, pp. 1–34 (cit. on pp. 28, 51, 66, 96).

Deb, Kalyanmoy and Mayank Goyal (1996). "A combined genetic adaptive search (GeneAS) for engineering design." In: *Computer Science and Informatics* 26.4, pp. 30–45 (cit. on pp. 28, 51, 66, 96).

Deb, Kalyanmoy, A. Pratap, S. Agarwal, and T. Meyarivan (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II." In: *Evolutionary Computation, IEEE Transactions on* 6.2, pp. 182–197. ISSN: 1089-778X. DOI: 10.1109/4235.996017 (cit. on pp. 27, 66, 79, 116).

Deb, Kalyanmoy, L. Thiele, M. Laumanns, and E. Zitzler (2002). "Scalable multi-objective optimization test problems." In: *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*. Vol. 1, pp. 825–830. DOI: 10.1109/CEC.2002.1007032 (cit. on p. 11).

Di Eugenio, Barbara and Michael Glass (Mar. 2004). "The kappa statistic: a second look." In: *Comput. Linguist.* 30.1, pp. 95–101. ISSN: 0891-2017. DOI: 10.1162/089120104773633402 (cit. on p. 115).

Diaz-Manriquez, A., G. Toscano-Pulido, and W. Gomez-Flores (2011). "On the selection of surrogate models in evolutionary optimization algorithms." In: *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 2155–2162. DOI: 10.1109/CEC.2011.5949881 (cit. on pp. 80, 94).

Draper, Norman Richard, Harry Smith, and Elizabeth Pownell (1966). *Applied regression analysis*. Vol. 3. Wiley New York (cit. on p. 15).

Drucker, Harris, Christopher J. C. Burges, Linda Kaufman, Alex J. Smola, and Vladimir Vapnik (1996). "Support Vector Regression Machines." In: *NIPS*. Ed. by Michael Mozer, Michael I. Jordan, and Thomas Petsche. MIT Press, pp. 155–161 (cit. on p. 20).

Emmerich, Michael, Nicola Beume, and Boris Naujoks (2005). "An EMO Algorithm Using the Hypervolume Measure as Selection Criterion." In: *Evolutionary Multi-Criterion Optimization*. Ed. by CarlosA. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler. Vol. 3410. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 62–76. ISBN: 978-3-540-24983-2. DOI: 10.1007/978-3-540-31880-4_5 (cit. on p. 29).

Emmerich, Michael and Boris Naujoks (2004). "Metamodel Assisted Multiobjective Optimisation Strategies and their Application in Airfoil Design." English. In: *Adaptive Computing in Design and Manufacture VI*. Ed. by I.C. Parmee. Springer London, pp. 249–260. ISBN: 978-1-85233-829-9. DOI: 10.1007/978-0-85729-338-1_21 (cit. on p. 74).

Fonseca, Carlos M. and Peter J. Fleming (1996). "On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers." In: *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*. PPSN IV. London, UK, UK: Springer-Verlag, pp. 584–593. ISBN: 3-540-61723-X (cit. on p. 10).

Fonseca, Carlos M., Andreia P. Guerreiro, Manuel López-Ibánez, and Luís Paquete (2011). "On the Computation of the Empirical Attainment Function." In: *Evolutionary Multi-Criterion Optimization*. Ed. by RicardoH.C. Takahashi, Kalyanmoy Deb, ElizabethF. Wanner, and Salvatore Greco. Vol. 6576. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 106–120. ISBN: 978-3-642-19892-2. DOI: 10.1007/978-3-642-19893-9_8 (cit. on p. 11).

Fonseca, C.M., L. Paquete, and Manuel López-Ibńez (2006). "An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator." In: *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pp. 1157–1163. DOI: 10.1109/CEC.2006.1688440 (cit. on p. 10).

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten (Nov. 2009). "The WEKA data mining software: an update." In: *SIGKDD Explor. Newsl.* 11.1, pp. 10–18. ISSN: 1931-0145. DOI: 10.1145/1656274.1656278 (cit. on pp. 51, 96, 116).

Hansen, Nikolaus and Andreas Ostermeier (June 2001). "Completely Derandomized Self-Adaptation in Evolution Strategies." In: *Evol. Comput.* 9.2, pp. 159–195. ISSN: 1063-6560. DOI: 10.1162/106365601750190398 (cit. on p. 30).

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. 2nd ed. Prentice Hall. ISBN: 978-0139083853 (cit. on pp. 20, 22, 112).

Hohm, Tim and Eckart Zitzler (2009). "Multiobjectivization for parameter estimation: a case-study on the segment polarity network of drosophila." In: *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. GECCO '09. Montreal, Qu&#233;bec, Canada: ACM, pp. 209–216. ISBN: 978-1-60558-325-9. DOI: 10.1145/1569901.1569931 (cit. on p. 114).

Huband, S., P. Hingston, L. Barone, and L. While (2006). "A review of multiobjective test problems and a scalable test problem toolkit." In: *Evolutionary Computation, IEEE Transactions on* 10.5, pp. 477–506. ISSN: 1089-778X. DOI: 10.1109/TEVC.2005.861417 (cit. on pp. 11, 80).

Igel, Christian, Nikolaus Hansen, and Stefan Roth (Mar. 2007). "Covariance Matrix Adaptation for Multi-objective Optimization." In: *Evol. Comput.* 15.1, pp. 1–28. ISSN: 1063-6560. DOI: 10.1162/evco.2007.15.1.1 (cit. on pp. 11, 28, 30, 80).

Ishibuchi, H., Y. Sakane, N. Tsukamoto, and Yusuke Nojima (2009). "Evolutionary many-objective optimization by NSGA-II and MOEA/D with large populations." In: *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pp. 1758–1763. DOI: 10.1109/ICSMC.2009.5346628 (cit. on p. 33).

Ishibuchi, H., N. Tsukamoto, and Y. Nojima (2008). "Evolutionary many-objective optimization: A short review." In: *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 2419–2426. DOI: 10.1109/CEC.2008.4631121 (cit. on pp. 29, 56).

Jin, Y. (2006). *Multi-objective machine learning*. Vol. 16. Springer (cit. on p. 114).

Joachims, Thorsten (2005). "A support vector method for multivariate performance measures." In: *Proceedings of the 22nd international conference on Machine learning*. ICML '05. Bonn, Germany: ACM, pp. 377–384. ISBN: 1-59593-180-5. DOI: 10.1145/1102351.1102399 (cit. on p. 38).

Kapp, Marcelo N., Robert Sabourin, and Patrick Maupin (2009). "A PSO-based framework for dynamic SVM model selection." In: *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. GECCO '09. Montreal, Qu&#233;bec, Canada: ACM, pp. 1227–1234. ISBN: 978-1-60558-325-9. DOI: 10.1145/1569901.1570066 (cit. on p. 113).

Karush, W. (1939). "Minima of Functions of Several Variables with Inequalities as Side Constraints." MA thesis. Dept.~of Mathematics, Univ.~of Chicago (cit. on p. 17).

Kazík, Ondřej, Klára Pešková, Martin Pilát, and Roman Neruda (2012). "Combining Parameter Space Search and Meta-learning for Data-Dependent Computational Agent Recommendation." In: *ICMLA (2)*. IEEE, pp. 36–41. ISBN: 978-1-4673-4651-1. DOI: 10.1109/ICMLA.2012.137 (cit. on pp. 94, 96, 111, 116).

Konen, Wolfgang, Patrick Koch, Oliver Flasch, Thomas Bartz-Beielstein, Martina Friese, and Boris Naujoks (2011). "Tuned data mining: a benchmark study on different tuners." In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. GECCO '11. Dublin, Ireland: ACM, pp. 1995–2002. ISBN: 978-1-4503-0557-0. DOI: 10.1145/2001576.2001844 (cit. on p. 113).

Kuhn, H.W. and A.W. Tucker (1951). "Nonlinear programming." In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. Ed. by J. Neyman. University of California Press, Berkeley, California, pp. 481–492 (cit. on p. 17).

Levenberg, K. (1944). "A method for the solution of certain non-linear problems in least squares." In: *Quart. J. Appl. Maths.* II.2, pp. 164–168 (cit. on p. 21).

Li, Hui and Qingfu Zhang (2009). "Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II." In: *Evolutionary Computation, IEEE Transactions on* 13.2, pp. 284–302. ISSN: 1089-778X. DOI: 10.1109/TEVC.2008.925798 (cit. on p. 11).

Lim, Dudy, Yaochu Jin, Yew-Soon Ong, and Bernhard Sendhoff (June 2010). "Generalizing surrogate-assisted evolutionary computation." In: *Trans. Evol. Comp* 14.3, pp. 329–355. ISSN: 1089-778X. DOI: 10.1109/TEVC.2009.2027359 (cit. on pp. 39, 94).

Loshchilov, Ilya, Marc Schoenauer, and Michele Sebag (2010a). "A mono surrogate for multiobjective optimization." In: *GECCO*. Ed. by Martin Pelikan and Jürgen Branke. ACM, pp. 471–478. ISBN: 978-1-4503-0072-5. DOI: 10.1145/1830483.1830571 (cit. on pp. 37, 43).

Loshchilov, Ilya, Marc Schoenauer, and Michele Sebag (2010b). "Comparison-Based Optimizers Need Comparison-Based Surrogates." In: *PPSN (1)*. Ed. by Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph. Vol. 6238. Lecture Notes in Computer Science. Springer, pp. 364–373. ISBN: 978-3-642-15843-8. DOI: 10.1007/978-3-642-15844-5_37 (cit. on pp. 94, 126).

Loshchilov, Ilya, Marc Schoenauer, and Michele Sebag (2010c). "Dominance-Based Pareto-Surrogate for Multi-Objective Optimization." In: *SEAL*. Ed. by Kalyanmoy Deb et al. Vol. 6457. Lecture Notes in Computer Science. Springer, pp. 230–239. ISBN: 978-3-642-17297-7. DOI: 10.1007/978-3-642-17298-4_24 (cit. on pp. 38, 43, 92).

Loshchilov, Ilya, Marc Schoenauer, and Michele Sebag (2011). "Not All Parents Are Equal for MO-CMA-ES." In: *EMO*. Ed. by Ricardo H. C. Takahashi, Kalyanmoy Deb, Elizabeth F. Wanner, and Salvatore Greco. Vol. 6576. Lecture Notes in Computer Science. Springer, pp. 31–45. ISBN: 978-3-642-19892-2. DOI: 10.1007/978-3-642-19893-9_3 (cit. on p. 32).

Loshchilov, Ilya, Marc Schoenauer, and Michele Sebag (2012). "Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy." In: *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*. GECCO '12. Philadelphia, Pennsylvania, USA: ACM, pp. 321–328. ISBN: 978-1-4503-1177-9. DOI: 10.1145/2330163.2330210 (cit. on p. 36).

Mann, H. B. and D. R. Whitney (1947). "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other." In: *The Annals of Mathematical Statistics* 18.1, pp. 50–60. ISSN: 00034851. DOI: 10.2307/2236101 (cit. on p. 90).

Marquardt, Donald W. (1963). "An Algorithm for Least-Squares Estimation of Nonlinear Parameters." In: *SIAM Journal on Applied Mathematics* 11.2, pp. 431–441 (cit. on p. 21).

Michalewicz, Zbigniew (1996). *Genetic algorithms + data structures = evolution programs (3rd ed.)* London, UK, UK: Springer-Verlag. ISBN: 3-540-60676-9 (cit. on p. 25).

Miettinen, Kaisa (1999). *Nonlinear multiobjective optimization*. Vol. 12. Springer (cit. on p. 32).

Nain, Pawan KS and Kalyanmoy Deb (2005). "A multi-objective optimization procedure with successive approximate models." In: *KanGAL report* 2005002 (cit. on p. 36).

Pilát, Martin and Roman Neruda (2012). "An Evolutionary Strategy for Surrogate-Based Multiobjective Optimization." In: *IEEE Congress on Evolutionary Computation*. IEEE, pp. 1–7. ISBN: 978-1-4673-1510-4. DOI: 10.1109/CEC.2012.6256450 (cit. on p. 65).

Platt, John C. (1998). *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines* (cit. on p. 18).

Rasmussen, Carl Edward and Christopher K. I. Williams (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press. ISBN: 026218253X (cit. on p. 23).

Reif, Matthias, Faisal Shafait, and Andreas Dengel (June 2012). "Meta-learning for evolutionary parameter optimization of classifiers." In: *Mach. Learn.* 87.3, pp. 357–380. ISSN: 0885-6125. DOI: 10.1007/s10994-012-5286-7 (cit. on pp. 112, 114).

Sato, H., H.E. Aguirre, and Kiyoshi Tanaka (2004). "Local dominance using polar coordinates to enhance multiobjective evolutionary algorithms." In: *Evolutionary Computation, 2004. CEC2004. Congress on*. Vol. 1, 188–195 Vol.1. DOI: 10.1109/CEC.2004.1330856 (cit. on p. 9).

Schaffer, J. David (1985). "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms." In: *Proceedings of the 1st International Conference on Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., pp. 93–100. ISBN: 0-8058-0426-9 (cit. on p. 26).

Schumer, M. and K. Steiglitz (1968). "Adaptive step size random search." In: *Automatic Control, IEEE Transactions on* 13.3, pp. 270–276. ISSN: 0018-9286. DOI: 10.1109/TAC.1968.1098903 (cit. on p. 31).

Sindhya, Karthik, Sauli Ruuska, Tomi Haanpää, and Kaisa Miettinen (2011). "A new hybrid mutation operator for multiobjective optimization with differential evolution." English. In: *Soft Computing* 15.10, pp. 2041–2055. ISSN: 1432-7643. DOI: 10.1007/s00500-011-0704-5 (cit. on p. 33).

Srinivas, N. and Kalyanmoy Deb (Sept. 1994). “Muiltiobjective optimization using nondominated sorting in genetic algorithms.” In: *Evol. Comput.* 2.3, pp. 221–248. ISSN: 1063-6560. DOI: 10.1162/evco.1994.2.3.221 (cit. on p. 27).

Van Veldhuizen, David A. and Gary B. Lamont (22-25 July 1998). “Evolutionary Computation and Convergence to a Pareto Front.” In: *Late Breaking Papers at the Genetic Programming 1998 Conference*. Ed. by John R. Koza. University of Wisconsin, Madison, Wisconsin, USA: Stanford University Bookstore (cit. on p. 8).

Voss, Thomas, Nikolaus Hansen, and Christian Igel (2009). “Recombination for Learning Strategy Parameters in the MO-CMA-ES.” In: *Evolutionary Multi-Criterion Optimization*. Ed. by Matthias Ehrgott, CarlosM. Fonseca, Xavier Gandibleux, Jin-Kao Hao, and Marc Sevaux. Vol. 5467. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 155–168. ISBN: 978-3-642-01019-4. DOI: 10.1007/978-3-642-01020-0_16 (cit. on p. 32).

Voutchkov, I. and A.J. Keane (Apr. 2006). “Multiobjective optimization using surrogates.” In: *Proceedings of the 7th International Conference on Adaptive Computing in Design and Manufacture*. The M.C.Escher Company, pp. 167–175 (cit. on p. 36).

Wilcoxon, Frank (1945). “Individual Comparisons by Ranking Methods.” In: *Biometrics Bulletin* 1.6, pp. 80–83. ISSN: 00994987. DOI: 10.2307/3001968 (cit. on p. 90).

Yang, Qing and Shengchao Ding (2007). “Novel Algorithm to Calculate Hypervolume Indicator of Pareto Approximation Set.” In: *Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques*. Ed. by De-Shuang Huang, Laurent Heutte, and Marco Loog. Vol. 2. Communications in Computer and Information Science. Springer Berlin Heidelberg, pp. 235–244. ISBN: 978-3-540-74281-4. DOI: 10.1007/978-3-540-74282-1_27 (cit. on p. 10).

Zhang, Qilong, Ganlin Shan, Xiusheng Duan, and Zining Zhang (2009). “Parameters optimization of support vector machine based on simulated annealing and genetic algorithm.” In: *Proceedings of the 2009 international conference on Robotics and biomimetics*. ROBIO’09. Guilin, China: IEEE Press, pp. 1302–1306. ISBN: 978-1-4244-4774-9 (cit. on p. 113).

Zhang, Qingfu and Hui Li (2007). “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition.” In: *Evolutionary Computation, IEEE Transactions on* 11.6, pp. 712–731. ISSN: 1089-778X. DOI: 10.1109/TEVC.2007.892759 (cit. on p. 32).

Zhang, Qingfu, Hui Li, D. Maringer, and E. Tsang (2010). “MOEA/D with NBI-style Tchebycheff approach for portfolio management.” In: *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1–8. DOI: 10.1109/CEC.2010.5586185 (cit. on p. 33).

Zhang, Qingfu, Wudong Liu, E. Tsang, and B. Virginas (2010). “Expensive Multiobjective Optimization by MOEA/D With Gaussian Process Model.” In: *Evolutionary Computation, IEEE Transactions on* 14.3, pp. 456–474. ISSN: 1089-778X. DOI: 10.1109/TEVC.2009.2033671 (cit. on pp. 24, 37).

Zhou, Aimin, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang (2011). “Multiobjective evolutionary algorithms: A survey of the state of the art.” In: *Swarm and Evolutionary Computation* 1.1, pp. 32–49. ISSN: 2210-6502. DOI: 10.1016/j.swevo.2011.03.001 (cit. on p. 25).

Zitzler, Eckart (1999). “Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications.” PhD thesis. ETH Zurich, Switzerland (cit. on p. 25).

Zitzler, Eckart, Dimo Brockhoff, and Lothar Thiele (2007). “The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration.” In: *Evolutionary Multi-Criterion Optimization*. Ed. by Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata. Vol. 4403. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 862–876. ISBN: 978-3-540-70927-5. DOI: 10.1007/978-3-540-70928-2_64 (cit. on pp. 9, 10).

Zitzler, Eckart, Kalyanmoy Deb, and Lothar Thiele (June 2000). “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results.” In: *Evol. Comput.* 8.2, pp. 173–195. ISSN: 1063-6560. DOI: 10.1162/106365600568202 (cit. on pp. 11, 50, 67, 72, 74, 80, 96, 99).

Zitzler, Eckart and Simon Künzli (2004). “Indicator-Based Selection in Multiobjective Search.” In: *Parallel Problem Solving from Nature - PPSN VIII*. Ed. by Xin Yao, EdmundK. Burke, JoséA. Lozano, Jim Smith, JuanJulián Merelo-Guervós, JohnA. Bullinaria, JonathanE. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel. Vol. 3242. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 832–842. ISBN: 978-3-540-23092-2. DOI: 10.1007/978-3-540-30217-9_84 (cit. on pp. 29, 66, 79).

Zitzler, Eckart and Lothar Thiele (1998). “Multiobjective optimization using evolutionary algorithms – A comparative case study.” In: *Parallel Problem Solving from Nature – PPSN V*. Ed. by AgostonE. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel. Vol. 1498. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 292–301. ISBN: 978-3-540-65078-2. DOI: 10.1007/BFb0056872 (cit. on p. 96).

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

SVM	Support Vector Machine
MOEA	Multiobjective Evolutionary Algorithm
SVR	Support Vector Regression
LR	Linear Regression
MLP	Multilayer Perceptron
RBF	Radial Basis Function
ANN	Artificial Neural Network
GPR	Gaussian Process Regression
NSGA-II	Non-dominated Sorting Genetic Algorithm II
SBX	Simulated Binary Crossover
MOEA/D	MOEA Based on Decomposition
IBEA	Indicator Based Evolutionary Algorithm
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
MO-CMA-ES	Multi-objective CMA-ES
ASM-MOMA	Multi-objective Memetic Algorithm with Aggregate Surrogate Model
LAMMA	Multi-objective Memetic Algorithm with Local Meta-Model
SBMO-ES	Surrogate-based Multi-objective Evolution Strategy
LSPS-MOEA	MOEA with Local Search and Pre-selection
sNSGA-II	NSGA-II with Hypervolume as Secondary Sorting Criterion
MSE	Mean Squared Error
RP	Relation Preservation

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured at:

http://postcards.miede.de/

Final Version as of May 15, 2013 (`classicthesis` version 0.1).