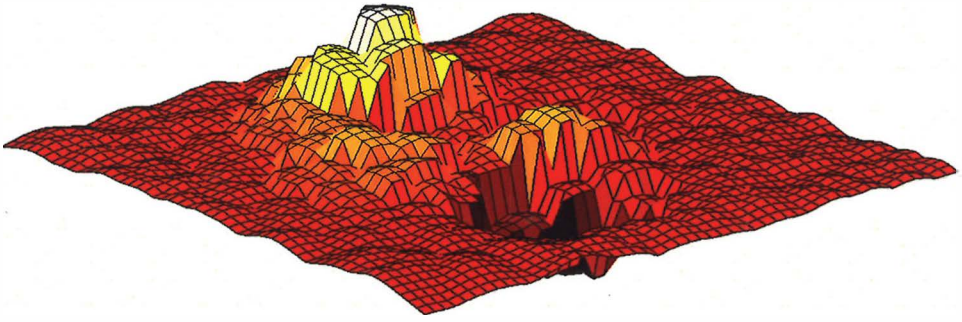
JYVÄSKYLÄ STUDIES IN COMPUTING
102

Timo Aittokoski

On Challenges of Simulation-Based Global and Multiobjective Optimization



JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 102

Timo Aittokoski

On Challenges of Simulation-Based Global and Multiobjective Optimization

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Villa Ranan Blomstedtin salissa
tammikuun 31. päivänä 2009 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in the Building Villa Rana, Blomstedt Hall, on January 31, 2009 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2009

On Challenges of Simulation-Based Global and Multiobjective Optimization

JYVÄSKYLÄ STUDIES IN COMPUTING 102

Timo Aittokoski

On Challenges of Simulation-Based Global and Multiobjective Optimization

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2009

Editors
Timo Männikkö
Department of Mathematical Information Technology, University of Jyväskylä
Pekka Olsbo, Marja-Leena Tynkkynen
Publishing Unit, University Library of Jyväskylä

Cover image by Timo Aittokoski: A predicted surface of the Matlab Peaks function as seen by the FDE algorithm

URN:ISBN:978-951-39-9034-3
ISBN 978-951-39-9034-3 (PDF)
ISSN 1456-5390

Jyväskylän yliopisto, 2022

ISBN 978-951-39-3457-6
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä 2009

ABSTRACT

Aittokoski, Timo
On Challenges of Simulation-Based Global and Multiobjective Optimization
Jyväskylä: University of Jyväskylä, 2009, 80 p.(+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; 102)
ISBN 978-951-39-3457-6
Finnish summary
Diss.

In this thesis, we address some challenges arising when solving real life simulation based optimization problems, for example, in system, device or process design. Often problems of this type are highly nonlinear, nonconvex, computationally expensive, gradient information is not available at a reasonable cost, and often there are several conflicting objectives to be considered simultaneously. These facts suggest that we should use carefully constructed optimization systems utilizing global, efficient and multiobjective approaches to solve these problems comfortably.

Multiobjective optimization problems can be solved in several different ways. In this thesis, we concentrate on interactive scalarization based approaches and on evolutionary multiobjective optimization (EMO) approaches. Our main emphasis in this thesis lays on dealing with two issues, problems due to the *computational complexity* of objective functions (running the simulator may be very time consuming), and difficulties of choosing the final solution among a possibly large set of Pareto optimal solutions.

In response to the above mentioned challenges, in this thesis we first construct a heterogenous optimization system capable of accommodating virtually any optimization algorithm and simulator combination. Then, to save in objective function evaluations, we propose an interactive approach with an adjustable solution accuracy during the process. On the algorithmic level, we propose a new and efficient single objective global optimization algorithm, which may be, for example, used in conjunction with the interactive approach to solve scalarized problems. On the other hand, to create an approximation of the Pareto optimal set in a single run, we propose a new and efficient EMO algorithm, that overcomes some drawbacks (e.g., lack of convergence) of the current approaches. In order to select the most preferred Pareto optimal solution, we propose an approach where the characteristics of the Pareto optimal set are condensed by using advanced clustering algorithms, thus reducing the cognitive burden of the decision maker.

Keywords: Global optimization, multiobjective optimization, evolutionary optimization, black box simulation, Pareto optimality

Author

Timo Aittokoski
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Supervisor

Professor Kaisa Miettinen
Department of Mathematical Information Technology
University of Jyväskylä
Finland

Reviewers

D.Sc. Jouni Lampinen
Department of Computer Science
University of Vaasa
Finland

Professor Eckart Zitzler
Computer Engineering and Networks Laboratory (TIK)
ETH Zürich
Switzerland

Opponent

Associate Professor Kyriakos Giannakoglou
Laboratory of Thermal Turbomachines
School of Mechanical Engineering
National Technical University of Athens
Greece

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor, professor Kaisa Miettinen for her encouragement, support, expertise and insights during the work of this thesis.

For their interest to my work, and for evaluating my thesis, I would like to thank Doctor Jouni Lampinen and Professor Eckart Zitzler. I have also been privileged to work with, and have some good advice from and fruitful discussions with (in no special order) Dr. Sami Äyrämö, Dr. Jussi Hakanen, Mr. Sauli Ruuska and Mr. Saku Kukkonen. For some technical issues related to the NIMBUS software I am indebted to Mr. Vesa Ojalehto. My lifetime fascination with engine design and improvement, which has also motivated the work of this thesis to some extent, originates from my boyhood years and writings of S. Tiittanen and late and widely appreciated Gordon Jennings.

Further, I would like to thank my parents for creation of my essence, and allowing me to pursue several seemingly precarious goals during the years, which, of course, were retrospectively necessary to get me where I am now. Last but not least, I also want to thank all my loved ones, friends and rest of my colleagues for being there, and once again, Fourmyle of Ceres for his necessary and sufficient optimality conditions for life.

This study was financially supported by COMAS graduate school, Academy of Finland (grant number 104641), Ellen & Artturi Nyyssönen Foundation, Jenny & Antti Wihuri Foundation and University of Jyväskylä's Grant for Doctoral Studies.

Jyväskylä, 18th December 2008
Timo Aittokoski

LIST OF FIGURES

CONTENTS

LIST OF INCLUDED ARTICLES

PI Timo Aittokoski and Kaisa Miettinen. Cost Effective Simulation-Based Multiobjective Optimization. *Engineering Optimization 40(7), 593-612*, 2008.

PII Timo Aittokoski and Kaisa Miettinen. Decreasing Computational Cost of Simulation Based Interactive Multiobjective Optimization with Adjustable Solution Accuracy. *Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing, No. B 19/2008, University of Jyväskylä*, 2008.

PIII Timo Aittokoski, Sami Äyrämö and Kaisa Miettinen. Clustering Aided Approach for Decision Making in Computationally Expensive Multiobjective Optimization. *Optimization Methods and Software, to appear.*

PIV Timo Aittokoski and Kaisa Miettinen. Efficient Evolutionary Method to Approximate the Pareto Optimal Set in Multiobjective Optimization. *Proceedings of International Conference on Engineering Optimization EngOpt 2008, Rio de Janeiro, Brazil, June 1-5*, 2008.

PV Timo Aittokoski. Efficient Evolutionary Optimization Algorithm: Filtered Differential Evolution. *Reports of the Department of Mathematical Information Technology, Series B. Scientific Computing, No. B 20/2008, University of Jyväskylä*, 2008.

1 INTRODUCTION

In many industrial design or control problems it is imperative to be able to adjust features (like quality, production cost, strength, etc.) of the end product or a system. Usually there are *design variables* in the design, which affect the end result, *objective*. For example, in the very simple case of a rectangular container design, we may have three design variables, width, height, and depth, which determine the objective value of the design, which is the volume of the container in this case. In many design and control problems the main problem can be simply posed as: what design variable values will produce the best end result?

Traditionally, a design engineering team has been responsible for answering the question stated above. By some careful selection of design variable values they have to create a functional and satisfactory design for the given product. Obviously, the quality of the product is relative to the expertise of the design engineering team, and also to the amount of real world trial-and-error-testing and redesign performed. It is easy to see that this kind of an iteratively progressing procedure of design-testing cycles is time consuming, expensive, may be even dangerous, and worst of all, the end result may be suboptimal.

Nowadays, the behavior of many real-world systems and devices can be expressed with mathematical *models*. If such a model is implemented using a computer, it can simulate the behavior of the system, and thus such a software is called a *simulator*. The simulator evaluates the behavior of the system using specific values for the input (or design) variables and calculates corresponding values for the output variables. For a review of simulation methodology, we refer, for example, to [6] and [54].

In industry, simulators are increasingly used to study the behavior of some system instead of extensive testing with small scale models, simplified mechanical models, or with the actual system itself. There are different simulators ranging from simulation of chemical processes of a paper making line [13] to flows around aircraft [139] and internal combustion engine processes [84]. The behavior of different structures, flows etc. is often simulated using the finite element method (FEM), see, for example, [24, 86]. There are many reasons for the growing

popularity of simulator use; for example, the ever increasing speed of computing and inexpensive computers make their use more appealing in terms of invested time and money. Running the simulator is usually far more cheaper, faster and in some cases also safer than implementing the prototype of a real system or device. Simulation, among other things, allows the designer to rapidly try several different designs and to explore their advantages and disadvantages.

Although the benefits of simulator usage are undisputed, as they may drasticly cut down the product design time and especially the testing time, certain problems still remain. A simulator merely mimics the behavior of some system. For example, an engine simulator may predict the output of a given engine configuration. Nevertheless, the simulator cannot give information of how exactly that behavior can be improved, or how some desired property could be attained. The core of the design process is essentially the same as with the traditional trial-and-error approach: what design variable values will produce the best end result? The designer must try to find an answer to this question, and decide what kind of design to evaluate next, but in this case, instead of the real world testing, the end result of the design is evaluated using the simulator.

In practice, the task of the designer is not easy, because there usually are too many variables for a human to control systematically while pursuing some predefined property for the end result. It is also often impossible to see the delicate interactions between the separate design variables with regard to the pursued objective.

Further, with real life engineering problems, the designer is rarely so lucky that he/she needs to deal with only a single objective. Often there are several objectives that should be simultaneously improved and they usually are conflicting. For example, in the field of engine design, the designer may want to create a powerful engine, while he/she must keep its fuel consumption as low as possible, and further, the production costs should be minimized. These three objectives are obviously conflicting. As another example, in the trivial case of a container design, we may have two objectives: the volume of the container should be as large as possible, while the amount of material needed to build the container should be kept as low as possible. It may be useful to point out here that in many cases, there may be more objectives than only two or three.

As it seems, for the human designer it may prove to be extremely difficult to gain results that he/she is aiming for by manually adjusting the design variable values. Further, if some acceptable design is found with a trial-and-error method, it is by no means guaranteed to be optimal; it could be possible to find even better design by some other means. This is where the optimization steps in and provides tools to generate optimal solutions with only minimal human effort. In the following paragraphs, we discuss shortly, on a general level, some essential concepts of optimization.

We all are familiar with the basic concepts of optimization, probably without even being aware of it. In our everyday life, we continually come across with optimization problems. For example, while planning a visit to our friend on the

other side of the city, we may plan our route so that it is the shortest or the fastest possible. While driving on a highway and noticing that we are a little bit short of fuel, we may adjust our way of driving in order to save fuel and to get to the next gas station. In both of these situations we are optimizing some quantity of the system. In the first example we are optimizing our route, and our objective (function) is to minimize either the distance or the time to the destination. In the second example, we are optimizing our fuel consumption, and the objective is to maximize the length of travel before the fuel runs out.

In both of these cases, some single property of the system is identified as an objective, and then optimized (either minimized or maximized). Because the value of the chosen objective is a function of the values of design variables, we refer to the mechanism that produces objective values as an *objective function.*

Sometimes it may be necessary to apply some constraints to the design variable values for the mathematical model to be meaningful and for the design to be implementable. For example, the length of some physical item may not be negative, or the volume of some object must be sufficient to house some specified apparatus: e.g., the nose cone of the aircraft may need to house a radar disc. Acceptable values for design variables define a region, where the optimum value can be searched for, i.e., *the search space* (also known as a design space).

If there is only one objective to be improved, we have a single objective optimization problem. However, in real life, as seen above, we often face problems with several, often conflicting objectives. Problems of this type are called *multi-objective optimization problems*.

Design (as well as any other) optimization problems can be solved using appropriate optimization (a.k.a search) algorithm, which can be described as a routine where the aim is to find the best possible value for a given objective by iteratively and systematically manipulating design variable values. Optimization algorithms automatize the search procedure using different means in judging what values for the design variables should produce good objective function values. In some sense, an optimization algorithm explores the objective function surface, and uses intelligent means to deduce where peaks (maxima) and valleys (minima) are located.

In contrast to testing several hundred different design variable combinations manually, with optimization algorithms the designer needs only to construct an objective function (which reflects the goodness of a particular design) to meet his/her needs and define ranges and other possible constraints for the design variables, i.e., the boundaries for the search space. After this, the optimization algorithm searches for the optimum by relentlessly testing possibly a myriad number of different design variable value combinations, until the optimal design is found, or some other stopping condition is met. There are lots of different optimization algorithms available for different types of optimization problems. A thorough discussion of these can be found, for example, in [10], [52] and [90] and references therein.

Although optimization tools may essentially speed up the design process,

and also produce optimal results, the implementation of the *optimization system* consisting of the optimization algorithm, objective function(s) (and related design variables and constraints), and interfaces between these two, is not necessarily an easy task. *Simulation based optimization*, where the objective function values (as well as possible constraint values) are derived from the output (files) of the simulator run, places some special requirements on the optimization system.

In simulation based optimization, the objective function value is often produced by some simulation software, and, thus, it can be the result of a complex sequence of calculations. Due to the complex nature of calculations, relationships between the decision variables and objective function values do not necessarily exist in a closed form, but the objective function is a so-called *black box function*. In this case, the internal mechanism and structure of the function is unknown or unavailable, and only the input and output characteristics can be utilized. For this reason, we cannot usually assume that the objective function is *convex* or unimodal, i.e., that it contains only one unambiguous optimum. Often there are several local optima, and the problem is to avoid poor local optima and to find the global one among them. Also *computational complexity* of objective function evaluations may be a problem. In contrast to easy closed form functions, running the simulation is often time consuming; one run may take time from a few seconds to hours or even days in the worst case, for example, with some complex flow model. To cope with computational complexity, it is necessary to use algorithms with as high *efficiency* as possible, i.e., algorithms which produce good objective function values using as few objective function evaluations as possible.

Further, derivatives which are commonly used to guide the optimization process are usually unavailable with black-box problems. Derivatives can be estimated using *finite differences*, i.e., by calculating differences in objective function values close to the current point along each dimension, but this, in turn, would increase computational load significantly. Also the selection of the optimal step length to calculate the finite differences is not a straightforward matter.

Another method to calculate gradient information numerically is a so-called *automatic differentiation* (AD), see, for example, [7]. The basic process of AD is to take the underlying program which calculates a numerical function value, and to transform it into a transformed program which calculates the desired derivative values. Obviously, the use of automatic differentiation may be very difficult, or even impossible (if the source code is not available), with black-box type software. For the reasons above, neither finite differences nor AD is used in this thesis. Instead, it is assumed that gradient information is not available at a reasonable cost. For a general discussion about simulation based optimization, we refer, for example, to [56].

All the facts above suggest that we are not able to use traditional and efficient optimization algorithms such as the steepest descent (gradient) method or the sequential quadratic programming method (SQP), which require gradient information and are local search methods. In other words, these methods will find the nearest optimum from the starting point, which could be quite far from the real global optimum.

To summarize, when solving real life simulation based black-box engineering problems, we need to use global, multiobjective and efficient (in terms of objective function evaluations) approaches to tackle problems caused by several local optima, several conflicting objectives, and computational cost of objective function evaluation, respectively. In this thesis, we discuss some special requirements of and difficulties in the aforementioned problems and possible responses to them. We concentrate on dealing with two issues mainly, namely problems due to the computational complexity of objective function(s), and difficulties of choosing the final solution among a large set of mathematically equivalent (Pareto optimal) solutions in the multiobjective case. In [PI] we show an example of how a simulation based optimization system can be constructed. To handle costly objective functions, we propose a method to reduce the number of required objective function evaluations in an interactive multiobjective approach in [PII], and further we propose efficient single and multiobjective optimization algorithms in [PV] and [PIV], respectively. To help in choosing the final solution among the set of Pareto optimal solutions, we propose a clustering based approach in [PIII].

In this thesis, we restrict our consideration to box-constrained black-box problems of global nature, with a reasonable number of both design variables (in the order of dozens) and objective functions (less than one dozen) to keep problems at a solvable level, although the characteristics of objective function(s) involved may strongly affect this, as discussed later in Section 2.1. Other types of approaches would be needed for larger problems. Throughout the text we assume that no gradient information is available at a reasonable cost. Further, we assume that a single objective function evaluation is somewhat costly (ranging from seconds to a few minutes), but not extremely costly (hours or days). Thus, we can solve problems using evaluations in the range of hundreds or thousands, instead of only tens, as would be the case with extremely costly problems.

The rest of this study is organized as follows. In Chapter 2 we discuss optimization methods and tools from our perspective in the field of global single objective optimization, and also proceed to a multiobjective treatment. In Chapter 3 we discuss some challenges originating mainly from the computational complexity of objective function evaluations and the difficulty in choosing the final solution among the mathematically equivalent set of Pareto optimal solutions in the multiobjective case. In the same chapter, with regard to challenges discussed, we describe the main contribution of this thesis, that is, we propose some possible treatments to the above mentioned problems. In Chapter 4 we describe the author's contribution in this study. Finally, in Chapter 5 we draw some conclusions and discuss some ideas for future work.

2 OPTIMIZING BLACK-BOX ENGINEERING PROBLEMS

Optimization can be regarded as a means to find the best solution to a problem among all allowed solutions, i.e., the optimum of one or more objective functions must be found by varying the design variable values with respect to some constraints. Several different methods have been developed for different types of optimizations problems, usually based on the type and number of design variables and on the properties of the objective and possible constraint functions [52].

Let us first consider cases where we are interested in solving of a global single objective optimization problem formulated as

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in S. \end{array} \tag{1}$$

Objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is minimized by altering values of the decision or design variables forming a vector $\mathbf{x} \in \mathbb{R}^n$. The *points* (or vectors) defined by values of decision variables will lie then within the search (a.k.a. decision or design) space, i.e., in a box constrained domain in $\mathbb{R}^n$ in our case. Sometimes all the points in the search space are not acceptable, and an acceptable subset of the search space is called *feasible region* S. If only box constraints are used, the search space equals to the feasible region. Point $\mathbf{x}^*$ is a global minimum, if $f(\mathbf{x}^*) \leq f(\mathbf{x})$ with all $\mathbf{x} \in S$. If there exists $\delta > 0$ so that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ with all $\mathbf{x} \in S$, for which is valid $\|\mathbf{x} - \mathbf{x}^*\| \leq \delta$, point $\mathbf{x}^*$ is the local optimum.

The set is *convex* if all the points in the line between any two points of the set belong to the set. Similarly, a function f is convex if $f(t\mathbf{x} + (1-t)\mathbf{y}) \leq tf(\mathbf{x}) + (1-t)(f(\mathbf{y})$ for any $t \in [0,1]$ and $\mathbf{x} \neq \mathbf{y}$. Optimization problem (1) is convex if the feasible region S and the function f are convex. Otherwise, the problem is nonconvex. If the problem is convex, it has only one optimal solution, i.e., local and global optima are the same, and it can be solved in a local manner, see, e.g., [10]. *Unimodality* is a less restrictive concept than convexity, but a unimodal function has also only one optimal solution. A function f is said to be unimodal (over S) if there exists a path from $\mathbf{x}$ to $\mathbf{x}^*$ (global optimum) over which f is strictly increasing, for all $\mathbf{x} \in S$ [125].

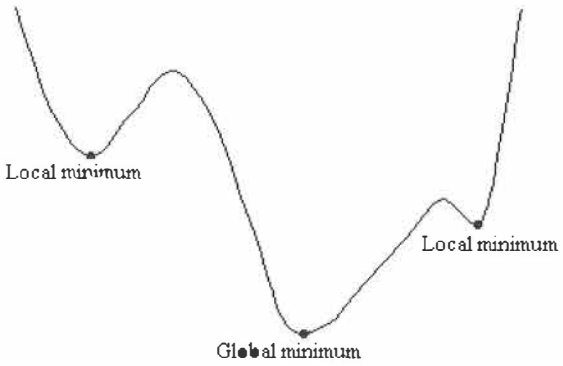


FIGURE 1 An example of a non-convex function with two local and one global minima.

On the other hand, if the problem is nonconvex, it may contain several local optima, and the aim is to find the best of these, the global optimum [1]. A function with several local optima is called *multimodal*.

Conventionally, problems of the form (1) have been considered in a local manner [10, 52], but in the recent decades there has been a growing interest to handle problems in a global manner [14, 53, 62, 105, 106, 133]. Further, problems with several objectives have drawn wide attention [22, 26, 32, 66, 90, 126], and we shall return to multiobjective optimization in Section 2.2. The trend towards proliferation of different methods is easily explained by the fact that local single objective algorithms have only limited applicability, especially in the field of real life optimization problems.

In Figure 1 differences of local and global minima are illustrated. When some *local optimization algorithm* is used, one of the local minima is found, but it depends completely on a given starting point which one. Thus there is a good chance to miss the global optimum, and get only a poor objective function value. On the other hand, with global optimization algorithms, a global optimum is usually found.

As our topic is simulation based optimization, we restrict our consideration to global and multiobjective optimization, which both are discussed in the following sections. We also assume that we have nonlinear, typically multimodal objective functions in the box-constrained search space involving continuous design variable values.

---

[1] It may be useful to point out that the global optimum is not necessarily unique; in some cases there may exist several global optima in the different parts of the search space. However, in the literature it is often implicitly assumed that the global optimum is a single point. We rely on a similar assumption also in this thesis.

2.1 Global optimization

The need to use global optimization algorithms arises from the characteristics of the objective function. If the objective function is unimodal and differentiable, there exists a plethora of local optimization algorithms to solve it efficiently and accurately, see, e.g., [10, 52]. On the other hand, if the objective function is multimodal and possibly otherwise difficult, i.e., non-differentiable, discontinuous, and probably even having discrete variables, some global optimization algorithm is needed in order to solve the problem. There are several different factors contributing to the difficulty of finding the optimum for the certain objective function. In the following we discuss effects of the characteristics of minima (sizes of basins of attractions, number of them, dispersion), characteristics of objective function landscape (variation, ruggedness, neutrality, deceptiveness), and general issues such as dimensionality and the cost of evaluation of objective function.

Each of the minima $\mathbf{x}_i^*$ has a *basin of attraction*, which is defined as the largest set of points in the search space such that the steepest descent algorithm which uses infinitely small steps will converge to $\mathbf{x}_i^*$ for any starting point inside that particular set of points [133]. The size of each basin may be related to the size of the whole search space, and thus each of the minima $\mathbf{x}_i^*$ has a probability p_i to be reached from a random starting point by using a local optimization algorithm. Thus, p_i is an important factor to characterize the optimization problem. If some $p_i = 1$, the function is unimodal, and thus rather easy to solve. If $p_i < 1$, there must exist several local minima, and the smaller the p_i for a global minimum is, the more difficult it is to find. An obvious consequence of a high number of minima is that each basin is generally smaller, although as a rule of thumb it seems (at least for some test problems) that the basin of attraction of the global minimum is the largest basin of all [132]. One possible explanation to this could be that if the function has almost the same Lipschitz constant (rate of change) around all the minima, then the one with the best value will have the largest basin.

If minima are clustered around a certain area of the search space, exploring the neighborhood of one minimum may lead to the detection of a neighboring and better minimum, thus improving the efficiency of the search. On the contrary, if the minima are dispersed all around the search space, lots of search effort may be wasted exploring neighborhoods of each of them.

In [141], some difficult properties of the objective function landscape are discussed, and here we mention some of them. Although global optimization algorithms do not usually utilize gradient information, they nevertheless depend on information about trends in the objective function landscape, based on objective function values. Thus, functions that are easy to optimize are continuous and exhibit low total variation in the objective function landscape, i.e., there is no fluctuation in function values. If the objective function landscape has lots of ups and downs, it is rugged, and *ruggedness* can be defined as a multimodality accompanied with steep ascends and descends. In that case, small changes in

decision variable values may cause large changes in the objective function value, the optimization algorithm cannot find reliable trends in the objective function surface, and it becomes harder to decide what part of the search space should be explored. Thus, efficiency of the search is decreased due to ruggedness.

Another difficulty related to information about trends in the objective function landscape is *neutrality*, i.e., objective functions with large plateaus. In a plateau, there is no information available to guide the search, and in an extreme case of neutrality, efficiency of the search is similar to that of random sampling [141]. One very difficult objective function landscape is a so called needle-in-a-haystack scenario, where the global minimum is located in a small and deep pit, which is isolated and surrounded by a large plateau. In this case, the extreme local ruggedness is combined with a general lack of information.

With a *deceptive* objective function landscape, the trend in objective function values tend to lead the search away from the global minimum. For example, a hill sloping down to the right leads the search to that direction, while the real global minimum may be located to the left, on the other side of the peak. In this case, the basin of attraction of the global minimum is rather small, and difficult to detect.

In addition to the above aspects, problem dimension, i.e., the number of design variables n, also affects the problem difficulty. In general, higher dimensional problems are more difficult to solve than lower dimensional ones. This is due to the *curse of dimensionality*, i.e., the fact that the volume of the higher dimensional space increases exponentially with the dimension n. This causes difficulties particularly if the search space is to be covered with some specified density. For example, if a one-dimensional line of length l is to be covered with a density of points located $l/10$ apart, 10 points are needed. In a two-dimensional case, i.e., a square, for the same density, 100 points are needed. In a three dimensional case of a cube, 1000 points are needed, etc. In this case, by adding one dimension, the number of points must be tenfold. It is obvious that with high dimensional problems extensive coverage becomes next to an impossible task.

Another possible difficulty with problems of higher dimensionality is that the number of local minima may be increasing with n. In this case, relative sizes of basins of attractions may be reduced, thus leading to a more difficult detection of the global minimum. Naturally, contemplation about an increasing dimensionality is reasonable only with artificial test problems. In other cases, the problem has a fixed dimensionality *per se*, and if more variables are included, the whole objective function landscape may change completely.

One concept loosely related to the dimensionality of the problem is *decomposability* of the problem. In [122], a function f is defined as decomposable if it can be written as a sum of n one-dimensional functions. In this case, according to [110], it is possible to replace the task of optimizing one function having n dimensions with the task of optimizing n one-dimensional functions. Thus, each variable can be optimized independently. Decomposable functions are also known as *separable* functions. If decomposability of the problem is known or it can be assumed, this feature should be utilized in the selection of the solution

method or the parameter values of it to make the optimization run more efficient.

Rotationally invariant search is such that its performance does not depend on the orientation of the coordinate system in which the objective function is evaluated [110]. For example, if the search is not rotationally invariant, more search effort may be directed along the coordinate axes. In relation to decomposable problems, it may be noted that they can be solved more efficiently if the search is not rotationally invariant. In general case, where decomposability of the problem cannot be assumed, it is more reasonable to use rotationally invariant search.

One obvious feature, in addition to the characteristics of the objective function landscape itself that may hinder the solution process is the computational cost (time) of the evaluation of the objective function value. If evaluations are very inexpensive, a huge amount of them can be afforded, and a mathematically difficult problem becomes in this sense "easy". On the contrary, even a relatively simple objective function landscape may be difficult to optimize if the evaluations are very expensive, e.g., taking hours or even days.

In general, a global optimization problem is not solvable. By this we mean that there exists no algorithm that can solve every global optimization problem using only a finite number of objective function evaluations [132]. On the other hand, if it was possible to use an infinite number of objective function evaluations, every problem could be solved with certainty using for example random sampling. Thus, with a limited budget for objective function evaluations, correctness of the final solution can be guaranteed only when very restrictive mathematical assumptions are valid [133], and for this reason, in methods with probabilistic features, the guaranteed correctness must be traded with a shorter runtime of the algorithm.

With methods with probabilistic features, increasing the number of objective function evaluations usually improves the quality of the solution up to a certain phase of the process, i.e., until the *convergence* is reached. When the algorithm has converged, it cannot reach new unseen solutions anymore, or solutions produced are located in a very small subset of the search space (stagnation) [141]. In convergence, there is usually no means to determine whether the algorithm has converged to a some local optimum, or to a global optimum, i.e., whether the convergence was premature or not. Convergence properties of some evolutionary algorithms have been discussed for example in [115] and [118].

To prevent premature convergence, and also to allow efficient and reliable search, global optimization algorithms consist of *global* and *local* techniques. The global technique is responsible for finding areas of the search space that have not been explored yet, thus the global technique is often referred to also as *exploration*. It is obvious that to detect the region of attraction of the global minimum, the search space should be sufficiently covered. With the local technique the already found good solutions are exploited by incorporating small changes into them to further improve the solution quality. The local technique is also referred to as *exploitation*. The division between local and global techniques is not necessarily explicit, as it may be in the simplest forms of so called hybrid methods,

where promising regions of the search space may be identified by some sampling procedure, and the best solutions found are then refined using some local optimization algorithm. Instead, the transition from a global technique to a local technique may be gradual and implicit, i.e., an *adaptive* technique. In this case, increasingly more points are sampled in the regions of the search space where promising solutions have already been found.

Different search operators may emphasize more either global or local search properties of the algorithm. It is obvious that algorithms emphasizing more the local search have a higher convergence speed, but they also have a higher risk of premature convergence to some local minimum. On the other hand, an algorithm lacking local search properties converges very slowly, or may even fail to converge to a global minimum completely. These two aspects of the search, local and global, lead to a compromise between convergence speed and reliability, often referred to as the *exploration-exploitation* dilemma: what would be the optimal rate to transition from global sampling to a more local procedure? Usually, emphasis between local and global search may be changed by tuning the parameters of the optimization algorithm before the optimization run. For example, in population based algorithms a larger population leads to a slower convergence but a higher reliability. Similarly, in simulated annealing the slow cooling schedule results with a more reliable search. In evolutionary algorithms, there are often parameters to control mutation and crossover rates, which affect the local-global search balance.

Unfortunately, tweaking the parameters of the optimization algorithm for fast and reliable search is an optimization problem in itself, and for the user of the algorithm it may be very difficult to find the proper values for a certain problem. This is especially the case if objective function evaluations are expensive, and the optimization run can be executed only once. Often, the only reasonable choice is to select some frequently used parameter values from the literature, but presumably these are not the best ones for the problem at hand.

The facts above suggest that it is beneficial if the optimization algorithm has only few parameters, and if the algorithm is not very sensitive to the choice of parameter values. In the optimal case, there would not be even the first parameter value for the end user to set. Attempts towards completely self-adaptive algorithms have been made, for example, by embedding the parameters of the algorithm inside the optimization problem, see, e.g., [17]. However, with this kind of approaches it may be reasonable to assume that the price to be paid for the lower number of parameters is a slower convergence rate.

As discussed above, the optimization problems may pose several difficulties for the optimization algorithm. By tweaking parameters of the algorithm it may be tuned more suitable for that particular problem, or it is even possible that the algorithm itself may tune itself to the problem at hand. Anyhow, it seems clear that for every problem it is possible to construct an algorithm that solves that particular problem most efficiently, but performs poorly on different types of problems. Mathematically, Wolpert and Macready have defined this behavior

in their No-Free-Lunch theorem (NFL) [146], postulating that over all possible problems the average performance of all algorithms is similar. With regard to NFL, no single algorithm can be superior in solving all given problems. Anyhow, it seems questionable how applicable NFL is with regard to real life optimization problems, or even with regard to problems which bear even remote resemblance to the physical world where we can assume some trends and logical coherence of the objective function landscapes.

In NFL, an optimization problem is considered as any arbitrary mapping from the discretized design space to the discretized objective space. By this definition, a vast majority of the objective functions considered have no noticeable structure, rather they are depicted by extreme ruggedness and apparent look of pure, drastic noise. We deem functions of this type senseless with regard to the physical reality, and beyond any hope to be solved efficiently.

In this light, we allow ourselves the possibility to believe that all "solvable" real life problems form a distinct subset of all possible problems, and for this subset the NFL is not necessarily applicable. The fact that optimization algorithms are used, instead of pure random sampling, supports our belief. If the NFL was applicable to a subset of real life problems, random sampling as a solution method should work as well as optimization algorithms.

Further, if the NFL was considered applicable for real life problems, all performance comparisons conducted in thousands of research publications would be rendered completely useless. By our reasoning, if the NFL was applicable, all algorithms should pose almost exactly the same performance, which does not seem to be the case. In performance comparisons differences are almost always detected, which, if the NFL was applicable, readily suggests that the set of test problems applied was insufficient. In this light, applicability of the NFL would render all performance testing useless, at least for general solvers, because the results would either be similar, or in case of differences the test setup should be deemed insufficient.

To tackle the difficulties posed by global optimization problems, many different methods have been suggested since the early years of the discipline. These methods share common ideas thus making it possible to define classes covering most of the methods. Striving for any complete classification is beyond the scope of this study, but instead we use a crude classification given in [133]. Methods are primarily divided into two non-overlapping classes with respect to the accuracy of the solution: those with guaranteed accuracy, *deterministic methods*, and those without one *probabilistic methods*. Deterministic, i.e., exact methods are covering methods (e.g., branch and bound), that iteratively detect and exclude regions of the search space which can be judged not to contain the global optimum. Although these methods guarantee that a solution with a given accuracy is obtained, the price to be paid for this guarantee is that some a priori information of the objective function must be available or some rather restricting mathematical assumptions must be valid [133]. In the case of simulation based optimization, these conditions cannot typically be met, and thus these methods are not of interest to us.

Among probabilistic methods, we devote our interest to two classes, namely metaheuristics and Bayesian methods. Both of these are often cited, and both have been successfully applied to real life engineering problems [104], the latter ones excelling in extremely costly problems [16, 49]. The concept of metaheuristics is not well defined, but in general applies to methods that contain some sort of metastrategy to guide the heuristic search method towards the global optimum. The list of metaheuristic methods may contain according to [53, 124], among others, simulated annealing, scatter search, tabu search, genetic algorithms, ant colony optimization, particle swarm optimization, controlled random search and differential evolution.

2.1.1 Metaheuristics

Metaheuristics may be further divided into single solution (e.g., simulated annealing and tabu search) and population based methods (e.g., genetic algorithms and differential evolution). In single solution methods, the neighborhood of the current solution is under consideration, and by certain rules a transition to the next location is allowed. For example, in simulated annealing, the new point is directly accepted if it is better than the current point, but in order to escape a local optimum during the solution process it is also possible to accept occasionally a worse point with a decreasing probability.

In population based algorithms, a population of points is initially scattered all around the search space. By certain rules, points in the population are replaced by new points, and a child population (i.e., the next generation) is produced, usually with a better average objective function value than in the parent population. With an increasing number of generations the population is expected to concentrate around the global optimum.

It is interesting to notice that essentially all population based algorithms have similar basic operations, although they use very colorful terminology, often inspired by some phenomena of the nature, i.e., evolution, or the flocking behavior of ants, birds, or bees. Sometimes it seems that the terminology itself may hinder the understanding of the algorithm behavior. Basically these algorithms have some mechanism to generate new points around the current points, and some mechanism to select points for the next population. If the functioning of a point generation mechanisms of these algorithms is studied geometrically, it can be seen that there are only subtle differences, although one could imagine the contrary based on the complex terminology used.

As we have utilized two often referred and rather effective metaheuristic algorithms in our work, Controlled Random Search in [PI] and [PII], and Differential Evolution in [PIV] and [PV], it is in order to introduce them in more detail.

Controlled Random Search, CRS

The Controlled Random Search (CRS) algorithm was presented originally by W.L. Price [108] already in 1977. Price proposed several versions of the algorithm, the widely cited method being CRS2 [109].

In general, CRS is a population based search algorithm. In the Price's original version the search space is randomly *sampled* (objective function values evaluated at the given locations) to form a population P of size NP, which is much larger than the number of design variables n. The suggested value for NP is $10 * n$ [2]. In the next step, a simplex is formed by selecting $n + 1$ points randomly from P. A new trial point is generated by selecting one of the points in the simplex which is reflected through the centroid of the remaining points (as in the Nelder and Mead simplex method [102]). If the objective function value of the trial point is better than the current worst point in P, the worst point is replaced in the population by the new trial point. The process of forming a new random simplex and generating the trial point is then repeated until some stopping criterion is met.

Price himself made the first two improvements to the original CRS algorithm, producing version CRS2 and version CRS3. In the second version (CRS2) a different mechanism to obtain new trial points is proposed. The difference between CRS1 and CRS2 lies in the way the simplex is formed. In CRS2, the first point of the simplex is always the current best point in the population P (others are randomly chosen), whereas the first point is randomly chosen in CRS1. CRS3 is otherwise similar to CRS2, but it also incorporates a Nelder-Mead type local search from the best $n + 1$ points of the set P.

After Price's initial work, the ideas of CRS algorithms have been further extended for example by M.M. Ali and C. Storey [3], who produced the variants called CRS4 and CRS5. Experiments have proved CRS4 to be superior to CRS5 [4]. Both CRS4 and CRS5 employ a local search phase, which is gradient based in CRS5.

Unlike in CRS3, in CRS4 there is no Nelder-Mead type local search. Instead, whenever a new best point x_{min} is found by a manner similar to that in CRS2, it is "rewarded" by an additional search around it by sampling a predetermined number r of points (e.g., $r = 4$) from the beta-distribution using the current x_{min} as its mean and the distance between x_{min} and x_{max} (the worst point in the population) as the standard deviation. This method is reported to be very efficient [2]. In our studies [PI] and [PII] we used version CRS2 based on a comparison made in [1], and a Pascal implementation for the algorithm was supplied by M.M. Ali.

Differential Evolution, DE

The Differential Evolution algorithm [110, 128] is a member in the family of evolutionary algorithms, and, to be more accurate, a form of evolution strategies (ES) [11]. As such, differential evolution is a simple, population based stochastic i.e., probabilistic optimization algorithm. It was developed and successfully applied to the optimization of some well known nonlinear, non-differentiable and non-convex functions by Storn and Price [128] in 1997. DE combines simple arithmetic

operators with the genetic operators (familiar from evolutionary algorithms) of selection, crossover and mutation. The basic principle of the DE is that a randomly generated starting population evolves to a final population concentrated around the global optimum. From the final population an individual with the best objective function value is picked up as the final solution when the search procedure terminates.

Genetic Algorithms (GAs) [55] and ESs have both similarities and differences. The crossover (also known as mating or recombination) processes of both methods are similar in that they facilitate the search process by mixing the successful information contained in more fit members, i.e., points in the search space, of the population to create new members. In GAs, the crossover step is the main search step, while ESs uses it as a secondary operator, or not at all.

In GAs, the mutation operator ensures that the genetic material (different design variable values) contained within a population between successive generations is sufficiently diverse to prevent premature convergence to some local optimum. In ESs, mutation is the main search step, and was originally implemented as a Gaussian-distributed move away from the current solution. This technique is effective when the average *mutation step length*, i.e., the amount of change in design variables, away from the current solution is comparable to the standard deviation of the actual distribution of the design variable values in the population. In this way, the extent of the search scales relatively to the scatter of the population. However, according to [12], the Gaussian distribution approach is computationally expensive to implement.

Ideally, the mutation step length is a function of the design variable (range of variable values) in question and the state of the evolutionary process, thus allowing large steps in the beginning of the process, and only small steps in the final phase of the search procedure. DE avoids the problem of selecting a proper mutation step length explicitly by using difference vectors formed from design variable values in the generation by generation evolving population as a convenient and appropriately scaled source of perturbations. Therefore, as the region of the search space which is occupied by current population contracts and expands over generations, the random step length in each dimension adapts accordingly. This crucial idea differs from the idea of a mutation operator as used by traditional ESs in which predetermined probability distribution functions determine vector perturbations.

As the execution of the DE algorithm starts, the initial population P of DE is formed and consists of NP individuals (vectors), each with n components. NP does not change during the optimization process. The initial population of vectors (of real coded design variables) is chosen randomly and should cover the entire search space to encompass sufficient diversity. If some already known good design is available, the initial population might be generated by disturbing its coordinates by adding normally distributed random deviations to them.

After the initialization phase, where the first parent population is created, the main loop of the DE algorithm is started with its two basic tasks: point generation and survivor selection mechanisms. Point generation mechanism encom-

passes two distinctive operators, mutation and crossover. In the mutation phase DE generates the same number of mutated i.e., perturbed vectors as there are members in the parent (current) population. These mutated vectors are later used in the crossover phase as mates for each member in the parent population. The mutation process begins by choosing three distinct vectors, $\mathbf{x}_{r_0}$, $\mathbf{x}_{r_1}$, and $\mathbf{x}_{r_2}$, randomly from the parent population, each with a uniform selection probability. The first selected vector forms the base value for the mutated vector. The other two vectors are paired to create a difference vector, whose components represent a random mutation step length for each dimension. The difference vector is multiplied by a mutation scale factor F, and a mutated point $\mathbf{v}$ is created as

$$\mathbf{v} = \mathbf{x}_{r_0} + F \cdot (\mathbf{x}_{r_1} - \mathbf{x}_{r_2}). \tag{2}$$

The whole mutation process is repeated in each DE iteration NP times so that a new mate for each member in the parent population is created.

The rationale of the mutation process above is that the length of the mutation step in each dimension will evolve proportionally over generations, taking small steps when the variation in the values of a given design variable within a population is small, and large steps when that variation is large.

In the crossover phase (parameter mixing), the mutated point $\mathbf{v}$ is recombined with another predetermined vector from the parent population, the target (i.e., the parent) vector $\mathbf{x}$, to yield a so-called trial vector. In DE, each member of the parent population serves as a target vector one after another. Thus, each parent is allowed to undergo recombination exactly once per iteration of DE by mating with a mutated vector. The crossover process in DE thus creates a child population of the same size as the parent population.

In recombination, the crossover constant, CR, is used to control the rate at which trial vector components are taken from the target vector or the mutated vector. If $rand(0,1) < CR$, the component is taken from the mutated vector, otherwise from the target vector. If $CR = 1$, then the trial vector is identical to the mutated vector.

In the survivor selection phase, each vector in the child population is evaluated for fitness (measured as the objective function value), on a competitive basis, against the fitness of its parent vector. The one with a better fitness of the two survives into the next generation. Thus, the trial vector replaces the target vector in the following generation if the trial vector yields a better objective function value than the target vector. The primary benefit of this scheme is that it resists loss of diversity by forbidding both the parent vector and its respective child vector to survive. The parent and the child vectors typically have some identical variable values, and if they both survive, the population may be driven to a homogenous state, where the diversity of individuals will be low and, thus, the DE will be unable to continue the search.

After the selection phase, the new population becomes a new parent population and the evolutionary process with mutation, crossover and selection continues until some termination criterion is valid, e.g., the best objective function value in the population converges to some specified value (e.g., to satisfactory

level), the predetermined budget for generations or function evaluations is exhausted, or the difference between the population's worst and best objective function values falls below some predetermined limit.

The behavior of the DE can be altered by using different values for parameters F and CR. As summarized in [110], the role of CR is to provide the means to exploit decomposability of the problem, if it exists, and to provide extra diversity to the pool of possible trial vectors, especially near $CR = 1$. In the general case of nondecomposable (parameter dependent functions), CR should be close to 1 so that the performance losses associated with using a low mutation rate (elements from mutated vector are utilized more) are minimized. When $CR = 1$, the DE performs rotationally invariant search, i.e., it does not generate more trial vectors in the direction of coordinate axes.

The parameter F is used to scale the mutation step length. In [110] it is stated that in DE the selection operator tends to reduce the diversity of a population, whereas mutation operator increases it. As a consequence, when F gets smaller, the convergence speeds up, but also the risk of premature convergence is increased, and eventually the population can converge even if the selection pressure is absent if F is too small. For this reason, it is crucial that F is of sufficient magnitude to counteract this selection pressure.

Usual parameter values are $CR = 0.9$ and $F = 0.8$, as given in [127]. Anyhow, it may require some serious parameter tuning to achieve the best performance for some certain problem. See [110] and references therein for details.

2.1.2 Bayesian algorithms

The approach of using Bayesian methods in global optimization aims at producing algorithms that despite having a rather poor efficiency in the worst case analysis can be used to solve average case problems efficiently [98]. Bayesian methods are based on a meta-modelling scheme, where the computationally expensive high fidelity objective function is replaced with a lower fidelity, and less expensive *surrogate model* (a.k.a meta model), and this model is used with the optimization algorithm instead of the original objective function. The surrogate model may be implemented for example by kriging [29], artificial neural networks (ANN) [59], radial basis function networks (RBFN) [19], support vector machines (SVM) [27, 135] etc. The surrogate is created by sampling the initial set of points within the search space, and after the initial sampling, a stochastic model of the objective function based on all sampled points is computed.

The simplest meta-modelling schemes utilize merely a static surrogate [8, 68], which is build once in the beginning of the optimization process, and no further updates are made. The selected optimization algorithm then evaluates values of the surrogate instead of the original objective function until the stopping condition is met. The use of a static surrogate, which may not describe the behavior of the original objective function very accurately, may obviously and very easily lead the algorithm to converge to a some false optimum, i.e., optimum of the surrogate, which is not the optimum of the original objective function.

A more realistic approach is to update the surrogate during the process, which leads to an improved accuracy of the surrogate, and thus the algorithm should avoid converging to a false optimum. Several algorithms with some surrogate update scheme have been proposed, to mention a few, assisting GA with ANN [20], GA with kriging [111], GA with RBFN [51, 100], and ES with kriging [40, 41]. It is also possible to use an ensemble of different surrogate schemes in unison, as in [85]. With a surrogate updating approach it is not a trivial task to decide when and how should the surrogate be updated so that the optimization algorithm would converge correctly with as few expensive objective function evaluations as possible. These issues are referred to as *model management* or in the context of evolutionary algorithms as *evolution control*, and they are discussed for example in [37], [69] and [112]. For a more profound discussion about the meta-model assisted evolutionary algorithms, the reader is referred to [68] and [50]. Useful information about meta-modelling can be found also in [76], although emphasis is on a multiobjective approach.

The highest level of sophistication within a meta-modelling scheme is achieved with *utility function based methods*, i.e., with methods that use the meta-model and uncertainty of it to determine at what location should the next expensive objective function evaluation be made in order to improve the surrogate model, and thus exploit all the information available to the full extent. The location is determined by maximizing a *utility function* (known also as a figure of merit) reflecting the rewards of taking more samples in a particular region. The purpose of the utility function is to balance local and global search by finding a trade-off between taking samples in known, promising regions versus taking samples in under-explored regions or regions where the variation in function values and uncertainty of the objective function value prediction are both high.

It may be worth to mention that *computational overhead* (required to run the algorithm itself) of algorithms employing utility functions may be high as the number of evaluated points increases. This is due to fact that maximizing the utility function is itself a global optimization problem, as is the fitting of the surrogate model. So, in order to select a location for the next sample, two global optimization problems must be solved, and this must be iterated as many times as samples are taken. Due to the large overhead in fitting a sampled dataset to the surrogate model and in selection of sample points, these methods are best suited for problems where the original objective function is very expensive to evaluate.

Probably the two most well known algorithms employing utility functions are the Efficient Global Optimization (EGO) [70] and *a radial basis function method for global optimization* [57], which both bear some resemblance to a method introduced more than a decade earlier, namely the P-algorithm [148]. Roots of methods of this type can be traced back as far as to 1964, to the work of H. Kushner [82]. Both algorithms have been shown to perform efficiently in terms of objective function evaluations with known test problems [57].

As we have utilized ideas borrowed from Bayesian algorithms in our work in [PV], it is in order to introduce the often referred and efficient EGO [70] algorithm in more detail.

Efficient Global Optimization, EGO

The Efficient Global Optimization (EGO) algorithm to solve optimization problems involving expensive black-box functions was first introduced and described in [70]. In EGO, the original objective function is sampled only in those points where the surrogate model with the utility function suggests that the value of the objective function could improve the most, and the surrogate model is updated accordingly. In this way, the number of expensive original objective function calls should be reduced, because only after a modest number of evaluations the surrogate model should describe the behavior of the original objective function quite accurately in the neighborhood of the global optimum. However, it is worth mentioning that fitting the surrogate model to the existing data may be very time consuming as fitting the model itself is an optimization task, as already discussed. In some cases, this computational load may result in prohibitive costs, as was the case in the comparison made in [1].

TO form a surrogate, EGO makes use of kriging [29], which is originally a geostatistical technique to interpolate the value of some quantity at unobserved locations based on values of nearby observations. In EGO, kriging is used to model the objective function surface based on the points sampled during the optimization procedure. More specifically, EGO exploits a version of the design and analysis of a computer experiments (DACE) model [121], based on Gaussian processes. The DACE model has such a favorable property that it estimates its own uncertainty in predicting objective function values. Knowledge of the possible error in the surrogate model is a useful property while trying to locate an optimum on the objective function surface, and EGO makes use of this property explicitly, as described in the following paragraphs.

In a box-constrained search space the EGO algorithm begins by first generating a number of sampling points (approximately ten times the number of design variables [70]) within the search space. Locations for these points are determined using a Latin hypercube (i.e., space filling) [142] design. A square grid containing sample point positions is a Latin square if and only if, there is only one sample point in each row and each column of the grid. In a two dimensional case (i.e., $n = 2$) it means that there cannot be two sample points with the same x or y coordinate values. A Latin hypercube is a generalization of the Latin square concept to an arbitrary number of dimensions i.e., design variables. By this arrangement, sample points are distributed at different locations along each axis instead of lumping them together, and as a result of that a maximum amount of information about the objective function surface is extracted using a minimum number of sample points.

In the next step, a DACE model is fitted to the points obtained by the Latin hypercube sampling. In this phase, some optimization algorithm is needed to fit the DACE model parameters to the existing data so that the model follows the given data as accurately as possible.

To generate a new sample point to evaluate, EGO optimizes the *figure of merit* (known also as an infill sampling criterion, ISC) to decide what point in the

search space should be included in the sampled points next. In this phase the EGO algorithm searches (and this is another optimization task within EGO itself in addition to the surrogate model fitting task) for the point that maximizes what is called in [70] "the expected improvement".

The idea of the expected improvement lies in the fact that the surrogate model provides confidence intervals on the predicted function values at unsampled points. Thus, different amounts of possible improvement are associated with different unexplored regions in the search space. The use of the expected improvement effectively means that EGO weighs up both the predicted objective function values and the error in this prediction in order to find the point that has the greatest potential to improve so far the best original objective function value. EGO does not just choose the solution that the model predicts would minimize the objective function value. Rather, it automatically balances exploitation and exploration (global and local search). A solution which has a good predicted objective function value and low error may not be as desirable as a solution whose predicted objective function value is worse but whose error of prediction is also higher. This is due to the fact that the worse solution with a higher prediction error may actually produce better values when evaluated with the original objective function.

Every time a new sampling point has been chosen and evaluated using the original expensive objective function (e.g., simulator run) the DACE model is updated with this new information and again the next sampling point is chosen from the location that maximizes the expected improvement using the updated DACE model, and this loop is iterated until some stopping criterion is met. The steps of the EGO algorithm as given in [123] are the following

1. Use a space-filling design of experiments to obtain an initial sample of the true expensive objective function.

2. Fit a surrogate model (kriging, DACE) to the data of points sampled so far.

3. Numerically maximize an infill sampling criterion (ISC) known as the expected improvement function to determine where to sample the next point.

4. Evaluate the value of the original expensive objective function at the given sample point and update the surrogate model.

5. Stop if the expected improvement function has become sufficiently small. Otherwise return to 2.

For a thorough discussion of the EGO algorithm, see [70] and [123].

2.1.3 Performance assessment

In a glimpse, performance assessment or comparison of different global optimization algorithms may seem a trivial task. With a more careful examination we may identify several issues that should be addressed. As stated in [30], performance

measures have tended to cluster in three areas: solution quality, computational effort required, and robustness of the algorithm.

Solution quality can be measured as the difference between the known optimal value and value reached by the algorithm before some stopping condition is met. Obviously, this approach is suitable only for the test problems where the global optimum is known in advance.

Computational effort required to solve the optimization problem arises usually from two sources: the computational cost of running the optimization algorithm itself (computational overhead) and the computational cost of evaluating the objective and/or constraint functions of the problem. Usually, with simulation based optimization problems, the computational effort due to running the algorithm is negligible compared to that of evaluating the objective functions. For this reason, it is often reasonable to measure the computational effort required to solve a certain problem (i.e., efficiency of the algorithm) merely by the number of objective function evaluations required. A common approach in performance testing is to choose some predefined target level of objective function value, and then record the number of objective function evaluations required to reach the target level. This approach can also be reversed. In this case the algorithm has some predefined budget of objective function evaluations to use, and after the budget has been exhausted, the best objective function value achieved so far is recorded.

Robustness may be defined as the ability of an algorithm to perform well over a wide range of test problems [30], and it is usually captured through measures of variability in solution quality, computational effort required etc. As algorithms perform differently with different problems, test functions possessing qualities that cause different difficulties to optimization algorithms should be selected to complement each other, to get as wide a perspective as possible to the performance of the algorithms. If the algorithm performs well over a wide variety of test problems, it is more likely that it will perform well also with some real life problems. Further, varying the number of design variables should be used in order to see how well the algorithm scales up with growing problem dimensions.

Another perspective to robustness is reliability. As global optimization algorithms contain very often stochastic elements, the result of the optimization process varies from run to run, and it is necessary to repeat each run several times to reach some level of statistical credibility in estimating the reliability of the algorithm. The number of required runs to achieve statistical reliability could probably be mathematically determined, but in the literature the tests are often repeated between 20 and 200 times, and the results are averaged. Sometimes an optimization algorithm may completely miss the global optimum and converge to some local optimum instead, or otherwise stagnate. These occurrences can also be recorded as a part of a reliability assessment.

After the performance testing has been carried out with a sufficient number of test functions, with a variation in design variable dimensions and with enough repetitions of a single setup (function, dimension, algorithm parameters) it is time to interpret the results. This might prove difficult: for example, algo-

rithm A may outperform algorithm B with some test functions, but with other functions B may outperform A. Further, with the same test function, algorithm A may converge more rapidly in the beginning of the process, but after a certain number of objective function evaluations algorithm B may catch up with it, and eventually bypass it. Which algorithm performed better in this case? If only a limited number of objective function evaluations were to be used, A would be better, but without limitations B would be better. In this sense, the results should be interpreted with regard to the purpose of use, and the optimization algorithm itself should also be selected with regard to the problem at hand. These issues are discussed in Section 3.1.

In the field of global optimization there exists no single test suite of test problems. Instead, there are different sources [38, 46, 47, 60] containing a variety of test problems. In [132], an attempt to classify test problems based on their difficulty is given. For a more thorough discussion about performance assessment, the reader is referred to [9, 15, 30, 61].

2.2 Multiobjective optimization

Multiobjective optimization is needed whenever there are several conflicting objective functions to be optimized simultaneously. A general form of a multiobjective minimization problem is

$$\begin{array}{ll} \text{minimize} & \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ \text{subject to} & \mathbf{x} \in S \end{array} \tag{3}$$

involving k (≥ 2) conflicting *objective functions* $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, ..., k$. Here, the *design variable vector* $\mathbf{x} \in \mathbb{R}^n$ and the search space are defined as in (1). An *objective vector* $\mathbf{z} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x}))^T$ in the *objective space* $\mathbb{R}^k$ consists of k objective function values calculated in the *design (variable) vector* $\mathbf{x}$.

In multiobjective optimization, we want to optimize the values of several objectives at the same time, but usually there exists no single point within the search space where all the objectives reach their individual optima. Instead, we have a set of solutions that we can regard as optimal. This set of so-called Pareto optimal solutions is called a Pareto optimal set [2]. A solution belongs to the Pareto optimal set if none of the objective function values can be improved without impairing the value of, at least, one other objective. This behavior is referred to as *tradeoff* between objectives, and it indicates that we are dealing with compromise solutions. In other words, if we want to gain something, we must give away something else. To be more specific, in (3), a design variable vector $\mathbf{x}' \in S$ and the corresponding objective vector $\mathbf{z}$ are called *Pareto optimal* if there does not exist another $\mathbf{x} \subset S$ such that $f_i(\mathbf{x}) \leq f_i(\mathbf{x}')$ for all $i = 1, \ldots, k$ and $f_j(\mathbf{x}) < f_j(\mathbf{x}')$ for

[2] In this thesis we assume that all interesting characteristics of the problem at hand are contained in the objective functions. In this case, we may restrict our consideration purely to the solutions in the Pareto optimal set.

at least one index j. The previous definition applies for global Pareto optimality. Also local Pareto optimality may be defined. A decision vector $\mathbf{x}^* \in S$ is locally Pareto optimal if there exists a neighborhood $N(\mathbf{x}^*)$ of $\mathbf{x}^*$ such that $\mathbf{x}^*$ is Pareto optimal in $N(\mathbf{x}^*) \cap S$. The objective vector $\mathbf{f}(\mathbf{x}^*)$ is locally Pareto optimal if the corresponding point $\mathbf{x}^*$ is locally Pareto optimal. Corresponding to the case of single objective optimization, the multiobjective optimization problem is convex if all the objective functions and the feasible region are convex [90].

The concept of dominance is related to Pareto optimality. In (3), an objective vector $\mathbf{z}^1$ is said to *dominate* another vector $\mathbf{z}^2$ if $z_i^1 \leq z_i^2$ for all $i = 1, \ldots, k$, and the inequality is strict for at least one index j. Furthermore, an objective vector $\mathbf{z}^1$ is *non-dominated* if there does not exist another objective vector $\mathbf{z}^2$ such that $\mathbf{z}^2$ dominates $\mathbf{z}^1$. Obviously, Pareto optimal points are non-dominated points, whereas non-dominated points are not necessarily Pareto optimal.

As the solutions in the Pareto optimal set cannot be further ordered without some additional preference information, it is usually up to the *decision maker* (also known as the designer) to provide such information. With this information, it is possible to select the most preferred solution, to be called the *final solution*, for the problem in question.

In several multiobjective optimization methods, the original multiobjective optimization problem is converted into a single objective optimization problem by using a scalarization method involving a *scalarizing function* [90] (see Section 2.2.1). The resulting *subproblem* is then solved using an appropriate single objective solver. It is important to emphasize that depending on whether the solver is local or global, the resulting solutions are either locally or globally Pareto optimal. With this respect, in addition to challenges posed by several objectives, also the same difficulties apply as with global optimization in general, as discussed in Section 2.1.

For many scalarization methods, some information about the ranges of solutions in the Pareto optimal set is needed. The lower bounds are defined by an *ideal objective vector* $\mathbf{z}^*$, whose components are obtained by minimizing each of the objective functions individually in the search space. With conflicting objectives, the ideal objective vector is not reachable, but it can be considered as a reference point, something to go for, and it dominates any Pareto optimal solution. A vector strictly better than $\mathbf{z}^*$ is called a *utopian objective vector* $\mathbf{z}^{**}$, and it is used for computational reasons, for example to avoid division by zero.

The upper bounds of the Pareto optimal set are much more difficult to obtain. A vector in the objective space containing the upper bounds is called a *nadir objective vector* $\mathbf{z}^{nad}$, and its components can be estimated from a payoff table, which is formed by using information obtained when calculating the ideal objective vector. The row i of the payoff table displays the values of all the objective functions calculated at the point where the objective function f_i obtains its minimal value. Hence, the components of the ideal objective vector are at the main diagonal of the table. The maximal value of the column i in the payoff table can be selected as an estimate of the upper bound of the objective function f_i. This estimated nadir objective vector may not be very good in all cases. For further

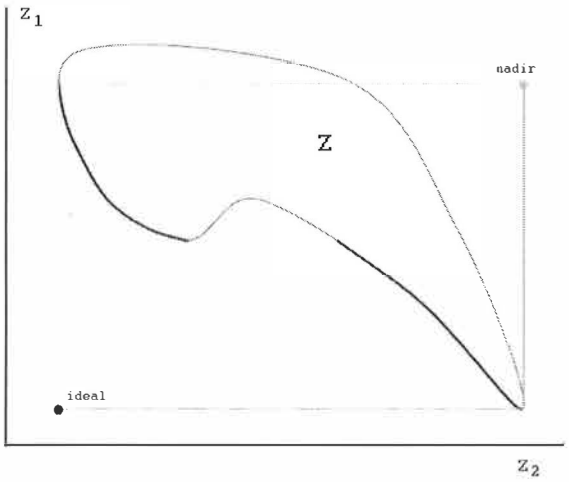


FIGURE 2 Illustration of Pareto optimal set (bold lines), ideal and nadir objective vectors in case of two objectives.

details, see [90] and the references therein.

For a survey of multiobjective optimization methods for engineering problems, see, for example, [87]. For more general discussion about the multiobjective optimization, see monographs [22, 66, 67, 90, 126].

To illustrate the key concepts of multiobjective optimization, in Figure 2 there are two objectives z_1 and z_2 and the feasible objective region Z (the set of all possible solutions of an optimization problem in the objective function space). The ideal objective vector is represented by a black point, the grey point represents the nadir objective vector, and the bold line is the Pareto optimal set.

The aim of multiobjective optimization can be regarded to be supporting a decision maker in finding the most preferred solution to be implemented among the Pareto optimal ones [77]. In this process, the preference information specified by the decision maker is required. Methods can be classified according to the role of the decision maker in the solution process [66, 90]. The classification given in [90] is as follows:

1. Methods where no articulation of preference information is used (no-preference methods).
2. Methods where a posteriori articulation of preference information is used (a posteriori methods).
3. Methods where a priori articulation of preference information is used (a priori methods).
4. Methods where progressive articulation of preference information is used (interactive methods).

In no-preference methods, the opinions of the decision maker are not taken into consideration. The decision maker may either accept or reject the solution. It seems reasonable to assume that the solution best satisfying the decision maker cannot be found using methods in this category. This is why these methods are suitable only for situations where no decision maker is available or the decision maker does not have any special expectations for the solution, and he/she is satisfied simply with some Pareto optimal solution.

In a posteriori methods (see, e.g., [120]), the Pareto optimal set or at least a representation of it is generated and presented to the decision maker, who then selects a solution that pleases him/her most. Methods of this type are also called approximation methods. One widely used but not a very good representative of a posteriori methods is the weighting method, see e.g., [48], where the idea is to associate an evenly distributed set of weighting vectors to objectives and minimize each of these weighted sums. Among the many problems of the weighting method let us mention that it does not work with nonconvex problems, and an evenly distributed set of weights does not necessarily produce evenly distributed set of Pareto optimal solutions [31]. Further, the weighting method requires high number of objective function evaluations, as for each weighting vector one optimization problem must be solved. It is worth mentioning that the weighting method can be used also as an a priori method if the decision maker specifies a weighting vector representing his/her preference information.

Most evolutionary multiobjective optimization (EMO) algorithms (e.g., [34, 75, 153]) used especially in the field of engineering [25, 32] also fall by this classification in the category of a posteriori methods; the algorithm produces a representative set of solutions aiming at approximating the Pareto optimal set (although solutions can only be guaranteed to be non-dominated ones), and the decision maker can afterwards choose the final solution. With EMO methods, problems may occur as generating the representation of a Pareto optimal set may prove to be computationally expensive, and visualization of the resulting representation is innate only in the case of two objective functions, which may hinder the selection of the final solution. With three objective functions, visualizations can be produced, for example, as projections to two dimensions, but with four or more objectives intuitive and easily understandable visualization is practically impossible, and thus it gets difficult to represent the information to the decision maker. Strangely, the latter aspect of visualization and decision making has been largely neglected by the EMO community, and a problem is often considered solved when the approximation of the Pareto optimal set is produced. In this thesis we have touched the issue of computational efficiency in [PIV], and problems of selecting the final solution in [PIII].

In a priori methods, the decision maker must specify his/her preferences before the solution process. The difficulty here is that the decision maker does not necessarily know beforehand what kinds of solutions it is possible to attain or how realistic his/her expectations are. Some known a priori methods are

value function method [73] (where the decision maker should be able to give an accurate and explicit mathematical form for the value function that represents his/her preferences globally), lexicographic ordering [44] (where the decision maker should arrange the objectives according to their absolute importance) and goal programming [23] (where the decision maker specifies goals or optimistic aspiration levels for the objective functions and deviations from these levels are minimized).

In interactive methods, a solution pattern is formed and repeated iteratively, overcoming many weak points of the three classes above. Only part of the Pareto optimal set has to be generated and evaluated, and based on this data the decision maker can further adjust his/her preferences as the solution process continues. In contrast to some other classes, the decision maker does not need to have knowledge about one's global preference structure. Due to the interactive solution process he/she will learn about the nature of the problem and will probably have more confidence in the final solution. Further, the amount of information that the decision maker should process at a time remains at a reasonable level, thus keeping the cognitive load acceptable.

From the perspective of simulation based optimization, interactive methods and EMO approaches from a posteriori class seem most appealing. With interactive methods the designer should be able to solve the problem at hand with a reasonable number of objective function evaluations, because in principle only those regions of the Pareto optimal set are explored that are of interest. Yet, the interactive nature of the process is largely lost with computationally expensive problems, because the decision maker may have to wait for a possibly long period of time after expressing his/her preference information for a new (or a set of new) Pareto optimal solutions to be generated. During the waiting period, the decision maker may forget what he/she was aiming at, or get otherwise confused. On the other hand, with EMO approaches the decision maker is involved in the solution process only after the most time consuming off-line computation is finished, and thus he/she can afterwards explore the Pareto optimal set rapidly once its representation has been generated. This approach seems attractive, if a high number of off-line objective function evaluations needed (feature often associated with EMO approaches) is tolerable. Further, the EMO field is an active research area, and approaches of this type are widely utilized because they do not pose special restrictions for the objective functions to be considered. In this thesis we have proposed an approach to overcome computational burden with the interactive NIMBUS approach in [PII]. In [PIV] we have addressed the issue of computational efficiency by proposing a new EMO algorithm, and in [PIII] we proposed a clustering based approach to aid the decision maker in the selection of the final solution.

In the following sections we discuss, in more detail, different scalarization functions, the interactive NIMBUS method employing them, and some evolutionary multiobjective optimization algorithms which are often used to solve engineering problems.

2.2.1 Scalarization methods

As mentioned earlier, in the case of an optimization problem with multiple objectives, one common approach is to scalarize it, i.e., convert it into a single objective optimization problem. For an ideal scalarization method there are two requirements: it must be able to find any Pareto optimal solution, and every solution it gives must be Pareto optimal [90, p 62]. Due to the nature of multiobjective optimization, some sort of a preference information over mathematically equivalued Pareto optimal solutions is needed. Typically, we assume that we have a human decision maker available who can give the preference information, and we can then define a Pareto optimal solution that satisfies the decision maker most as a final solution. For example, driving on a highway in a hurry and with little fuel in the tank, the driver has to balance between fuel consumed and time spent as driving faster consumes more fuel. One driver/decision maker may prefer the certainty of arriving to the destination to being there on time. Another driver, under the same circumstances, might be anxious to get to the destination on time, accepting the risk of not getting there at all. This example clarifies how the final solution depends on the decision maker.

In this subsection we consider some scalarization functions and methods. The first of the functions (neutral compromise solution) belongs to the class where no articulation of preference information is used. The next three functions (achievement scalarizing function, GUESS and STOM) are taken from interactive methods where a priori articulation of preference information is given in a recurrent fashion, and they are based on a decision maker specified *reference point*, i.e., a vector consisting of *aspiration levels* (function values that are satisfactory or desirable) for each objective function. A reference point is a natural and intuitive way to express preference information, because it directly utilizes objective function values, which have certain and clear meaning to the decision maker. Finally we discuss the interactive NIMBUS method, which is based on the decision maker's continuous involvement in the search process via classification of objective functions and utilizes the previous scalarization functions.

These methods have been selected to give a short overview of the different methods available, and the three reference point based functions are used because different scalarization functions tend to produce slightly different solutions (see Figure 3). All these functions were selected for consideration, because they are used in the synchronous version of the NIMBUS method [97], which we utilized in [PI] and [PII]. Their inclusion in NIMBUS is based on the results of [96], where 15 scalarizing functions were numerically and theoretically compared with regard to relation between the preference information provided and the results obtained. Further, in [PIII] we used the achievement scalarizing function.

Neutral compromise solution

The first of our scalarization functions belongs to the class of methods with no-preference information. It produces a neutral compromise solution (NCS) [145], which is Pareto optimal, by solving the problem

$$\text{minimize } \max_{1\le i\le k}\left[\frac{f_i(\mathbf{x})-\bar{z}_{i,mid}}{z_i^{nad}-z_i^{**}}\right]+\rho\sum_{i=1}^{k}\frac{f_i(\mathbf{x})-\bar{z}_{i,mid}}{z_i^{nad}-z_i^{**}}, \tag{4}$$

$$\text{subject to } \mathbf{x}\in S,$$

where for each $i = 1,...,k$
$\bar{z}_{i,mid} = \frac{z_i^{**}+z_i^{nad}}{2}$
$f_i(\mathbf{x})$ = value of i:th objective function at $\mathbf{x}$
z_i^{nad} = approximated nadir objective vector component
z_i^{**} = approximated utopian objective vector component
ρ = some small positive value.

The latter part of (4) is a so-called augmentation term, and it is incorporated to guarantee Pareto optimality of the solutions, and ρ must be strictly positive. A solution gained by this method should be located somewhere in the middle of the Pareto optimal set because the reference point $\bar{z}_{i,mid}$ is located precisely in the middle of the approximated ranges between the upper and lower bounds, i.e., the nadir and ideal objective values. Thus, the solution of (4) is a neutral compromise between conflicting objectives, as its name suggests. It is stated in [145] that neutral solutions like the one defined above might serve as a starting point for interaction with the decision maker. Starting with this solution the decision maker may iteratively balance between different objectives, and emphasize the ones he/she wishes.

Achievement scalarizing function

One approach relying on designer supplied reference point information is the achievement scalarizing function (ACH) [144, 145]. It finds the Pareto optimal solution closest to the reference point $\bar{\mathbf{z}}$, and different Pareto optimal solutions can be obtained by adjusting the reference point accordingly. The problem to be solved is similar to (4) (only the reference point is different), and it is formulated in [97] as:

$$\text{minimize } \max_{i=1,...,k}\left[\frac{f_i(\mathbf{x})-\bar{z}_i}{z_i^{nad}-z_i^{**}}\right]+\rho\sum_{i=1}^{k}\frac{f_i(\mathbf{x})}{z_i^{nad}-z_i^{**}}, \tag{5}$$

$$\text{subject to } \mathbf{x}\in S.$$

STOM

Our second scalarization function exploiting the reference point information comes from the interactive satisficing trade-off method (STOM) [101], and here we present the formulation given in [97]

$$\text{minimize } \max_{i=1,\ldots,k}\left[\frac{f_i(\mathbf{x}) - z_i^{**}}{\bar{z}_i - z_i^{**}}\right] + \rho \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{\bar{z}_i - z_i^{**}}, \tag{6}$$

$$\text{subject to } \mathbf{x} \in S.$$

GUESS

The third scalarization function which uses reference point is related to the one used originally in the GUESS method [18] and slightly modified in [97]. An augmentation term that was not used in the original formulation is included, and we assume that $\bar{z}_i < z_i^{nad}$ for all $i = 1, \ldots, k$

$$\text{minimize } \max_{i=1,\ldots,k}\left[\frac{f_i(\mathbf{x}) - z_i^{nad}}{z_i^{nad} - \bar{z}_i}\right] + \rho \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{z_i^{nad} - \bar{z}_i}, \tag{7}$$

$$\text{subject to } \mathbf{x} \in S.$$

Solutions produced by all the previous scalarization functions are Pareto optimal [97].

To illustrate differences between the ACH, STOM, and GUESS scalarizations, in Figure 3 we show graphically how different Pareto optimal solutions are produced when the same reference point is projected on to the Pareto optimal set. With ACH, the Pareto optimal point is found which is closest to the line spanned from the reference point to the direction between ideal and nadir points. With STOM, the Pareto optimal point closest to the line directed between the ideal and the reference point is found, and with GUESS the direction of line is spanned between the reference and the nadir points.

NIMBUS method

NIMBUS (Nondifferentiable Interactive Multiobjective BUndle-based optimization System) [90, 93, 94, 95, 97] is an interactive multiobjective optimization method designed especially for efficient handling of nonlinear functions. For that reason it is capable of solving complicated real-world problems.

In the NIMBUS method, the interaction phase has been aimed to be comparatively simple and easy to understand for the decision maker. At each iteration the NIMBUS method offers flexible ways to direct the search according to the designer's wishes by means of classification including aspiration levels and upper

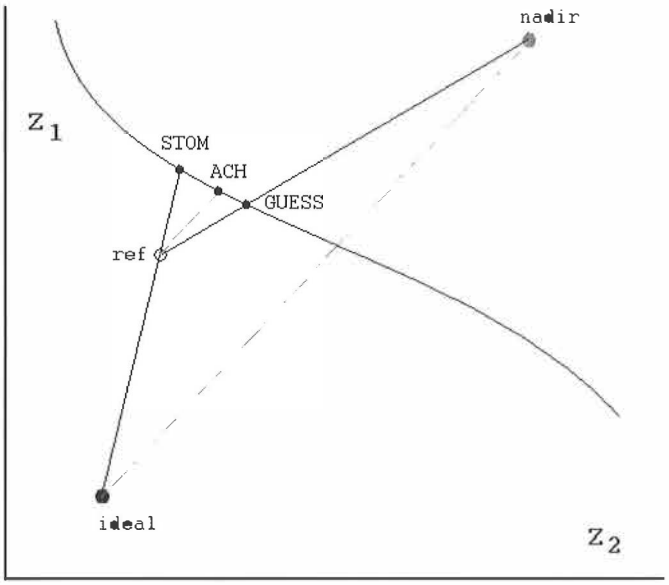


FIGURE 3 Different Pareto optimal solutions produced using ACH, STOM, and GUESS scalarizations and the same reference point.

bounds. The use of classification with aspiration levels avoids using other difficult and artificial concepts for preference information extraction. As the decision maker expresses his/her preferences as desirable objective function values, the preference information has a concrete and readily understandable meaning.

The classification of the objective functions means that the decision maker indicates what kinds of improvements are desirable and what kinds of impairments are tolerable. The basic idea in classification is that the decision maker contemplates the current Pareto optimal objective function values $f_i(\mathbf{x}^c)$ at each iteration of the NIMBUS method and assigns each of the objective functions f_i into one of the following five classes depending on his/her preferences:

1. I^{imp}, function value should be improved as much as possible.
2. I^{asp}, function value should be improved to a certain aspiration level.
3. I^{sat}, function value is satisfactory at the moment.
4. I^{bound}, function value is allowed to impair to a certain upper bound.
5. I^{free}, function value is temporarily allowed to change freely.

Due to the concept of Pareto optimality, a classification is feasible only if at least one of the objective functions is bound to improve and at least one is allowed to impair from their current levels.

After the decision maker has classified the objective functions using the above classes, and specified aspiration levels $\hat{z}_j$, $j \in I^{asp}$ and upper bounds ε_i,

$i \in I^{bound}$ (if required), the original multiobjective optimization problem is transformed into a single objective optimization subproblem, which is then solved. The resulting Pareto optimal solution reflects the classification as well as possible. Then a new iteration of method execution starts. In the synchronous version of the NIMBUS method [97], several scalarizing functions leading to different subproblems may be utilized using the same preference information. In the synchronous version the decision maker must define how many (one to four) different solutions (i.e., scalarizations) he/she wishes to use at each step. The standard NIMBUS scalarization (STD) given in [97] produces Pareto optimal solutions and is formulated as

$$\text{minimize} \max_{\substack{i \in I^{imp} \\ j \in I^{asp}}} \left[\frac{f_i(\mathbf{x}) - z_i^*}{z_i^{nad} - z_i^{**}}, \frac{f_j(\mathbf{x}) - \hat{z}_j}{z_j^{nad} - z_j^{**}} \right] + \rho \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{z_i^{nad} - z_i^{**}} \tag{8}$$

$$\begin{aligned} \text{subject to } & f_i(\mathbf{x}) \leq f_i(\mathbf{x}^c) \text{ for all } i \in I^{imp} \cup I^{asp} \cup I^{sat}, \\ & f_i(\mathbf{x}) \leq \varepsilon_i \text{ for all } i \in I^{bound}, \\ & \mathbf{x} \in S. \end{aligned}$$

Other scalarization functions used in NIMBUS, namely ACH, STOM and GUESS are given in (5), (6) and (7), respectively. Use of several scalarizations synchronously typically results with a set of slightly different Pareto optimal solutions from which the decision maker can choose the best as a starting point for a new classification. The decision maker can also generate an arbitrary number of intermediate solutions between any two Pareto optimal solutions found so far, and use them as a base for a new classification, if desired.

In Figure 4 we present an example screenshot of the IND-NIMBUS software [92, 103] (which implements the NIMBUS method) tackling a problem with three objective functions to be minimized. On the left side of the screen, the Pareto optimal solution to be classified is displayed in the form of bars. The end points of the bars represent the ranges of each objective function in a set of Pareto optimal solutions. The length of the bar represents the current value of the corresponding objective function (the less coloured the area is, the better the value).

Classification can be adjusted either by entering the desirable objective function values to fields $f1$, $f2$ and $f3$, or by clicking the bar with the mouse to indicate a desirable value. Clicking the current value means that it is considered satisfactory and clicking the end point on the left means that the corresponding function should get as good values as possible. Clicking the right end point or leaving the function unclassified means that it can change freely for a while. The right side of the panel contains already found solutions, and any of them can be freely chosen as a starting point for a new classification. When a solution from the right side is selected, it is shaded on the right side, and details of it are displayed on the left side. On the bottom right corner there is an area labelled "Best candidates". This is an area where the designer can drag and drop potentially promising solutions for later inspection or to be used as a starting point for a new classification.

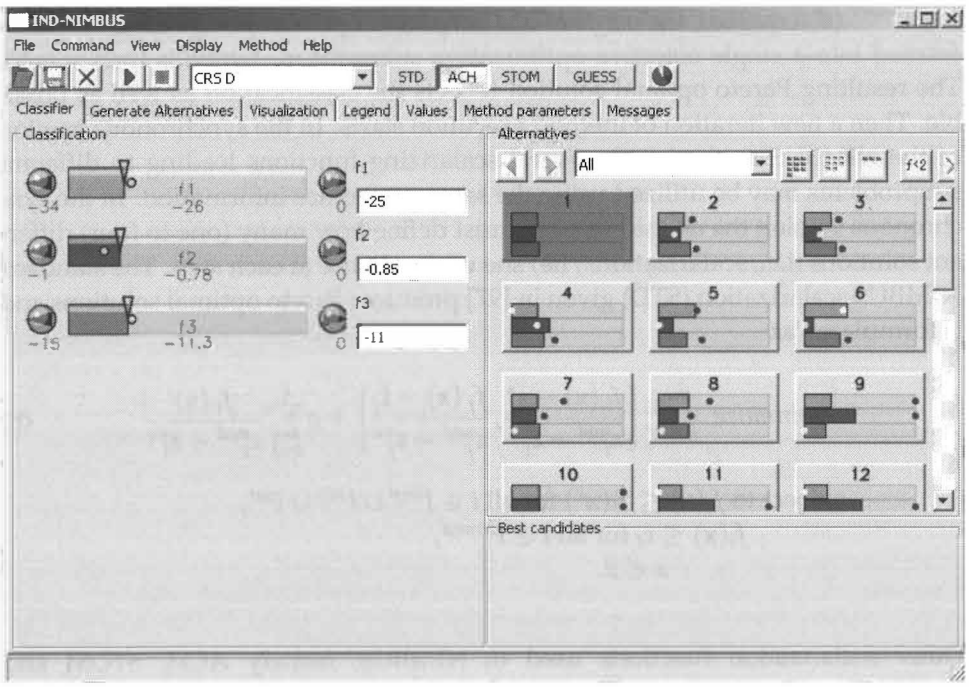


FIGURE 4 Sample screenshot of $IND - NIMBUS^{®}$ software.

In Figure 5 a flowchart of the NIMBUS algorithm is given to illustrate the role and possibilities of the decision maker in the solution process. For further details and more recent advances in the NIMBUS method, see [90] and [97].

2.2.2 Evolutionary multiobjective optimization

With scalarizing function based approaches it may be difficult for the decision maker to express his/her preference information at the beginning of the solution process, without prior knowledge of the problem. Although interactive treatment may help with this regard, it is still possible that the decision maker misses some region of the Pareto optimal set, which could be of even more interest to him/her, than the region that has been discovered. Further, every time the preference information is adjusted, the scalarized problem must be solved once again, which may take time if the problem is computationally costly.

In contrast to scalarization based approach, EMO algorithms strive to produce a discrete approximation of the Pareto optimal set, and afterwards the decision maker can select the most preferred solution as the final solution. Usually the goodness of the approximation set is characterized by two factors. First, the approximation should be well distributed, i.e., it should cover the entire Pareto optimal set with a sufficient density. Second, the approximated points should be as close as possible to the real Pareto optimal set. With regard to the first factor, it is to some extent open to discussion what "well distributed" means. At least two

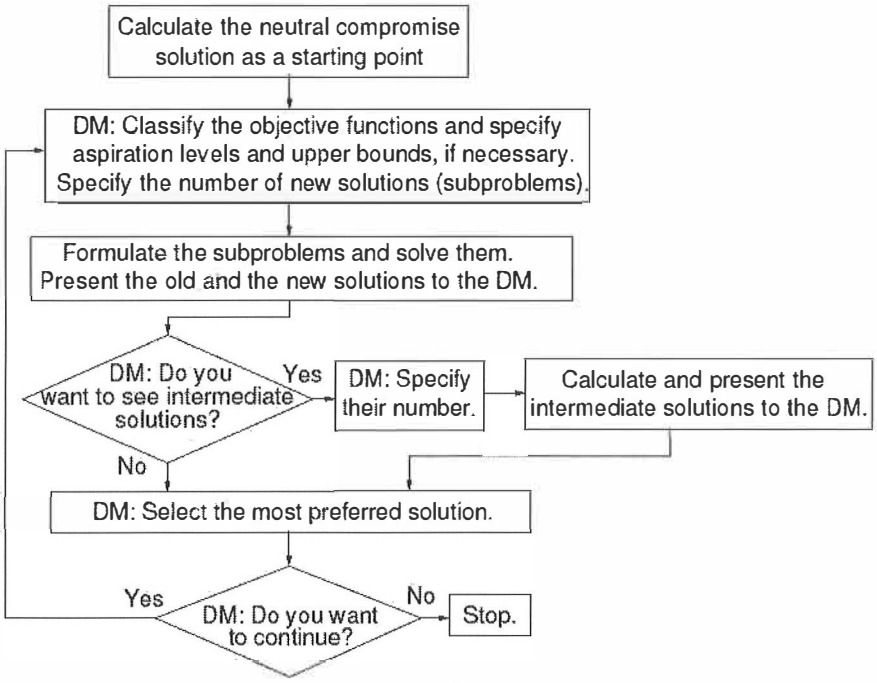


FIGURE 5 A flowchart of the NIMBUS algorithm.

schools of thought exist, the one sees that the points should be spread as evenly as possible throughout the Pareto optimal set, whereas the other sees that more points should be concentrated on the areas where there are significant tradeoff changes between objectives.

To produce a good approximation set, most of the current EMO approaches are based on the concept of dominance, added with some mechanism to maintain a good diversity of the solutions in the objective space. With this approach, the population is usually ranked based on dominance, and naturally non-dominated solutions are considered better, and favored in reproduction. Further, if two solutions with the same rank must be ordered (typically while selecting members for the next generation), it is usually done using some mechanism to select the one which is located in a less crowded region of the objective space. In addition to the current population, also some external archive of good solutions may be maintained and used both for dominance and diversity assessment.

Often referred algorithms in the above mentioned category are for example Pareto Archived Evolution Strategy (PAES) [75], second version of Strength Pareto Evolutionary Algorithm (SPEA2) [153] and Elitist Non-Dominated Sorting GA (NSGA-II) [34]. Some EMO algorithms are referred to as elitist, meaning that they maintain the best found solution during the process. Unfortunately, the concepts of elitism and the best solution are not as straightforward in the case of EMO as in the case of single objective optimization. In fact, at certain stage of the process, the approximation of the Pareto optimal set no longer improves, but

instead the population starts to oscillate because some solution located very near the Pareto optimal set may be replaced by some other non-dominated solution which improves diversity but is at the same time located much farther from the Pareto optimal set. For this reason, algorithms relying on the concept of dominance and diversity preservation mechanism actually never converge [83]. This is one fact that seems not to be widely understood in the EMO community. We have discussed some problematic issues, such as elitism, lack of convergence, diversity maintenance and deterioration of population in the EMO context in [PIV] (see also Subsection 3.2.3). In the same paper we also demonstrated occurrence of oscillation using the ZDT1 test problem [150] and the NSGA-II algorithm.

There are also further problems with dominance based EMO approaches. In [65] it is stated that "optimisers that use Pareto ranking based methods to sort the population will be very effective for small numbers of objectives, but not perform as effectively for many-objectives when compared to optimisers based on non-Pareto ranking methods". In the same study it was found that it is, at least with certain problems, more effective to use many scalarized single objective optimization runs than the single NSGA-II run if there are more than two objectives. The performance tests were conducted using the same amount of objective function evaluations in both approaches, effectively meaning that in the single objective case the number of evaluations was dramatically lower. Decreased performance of the dominance based approach is due to the fact that as the number of objectives increases, a larger portion of the population becomes non-dominated and the selective pressure of evolutionary mechanisms falls rapidly and fails to drive the population towards the Pareto optimal set [65, 107]. As the diversity preservation mechanism is often used as a secondary selection operator, its importance grows as the importance of dominance diminishes, and eventually most of the selective pressure arises from the diversity preservation mechanism, and the quality of the approximation of the Pareto optimal set remains poor.

Presumably due to the above mentioned drawbacks, some of the recent methods have given up using the concepts of dominance and diversity preservation. Probably the most promising of these rely on the concept of *hypervolume* [154], the volume in a hypercube, spanned by the extent of known solutions, that is dominated by the current population, and as such it can give information about both closeness to the Pareto optimal set and diversity of the population at the same time. In [45], a proof is presented that the maximization of hypervolume constitutes the necessary and sufficient condition to produce the maximally diverse set of Pareto optimal solutions for a multiobjective optimization problem. Furthermore, it possesses a desirable property to be used either as a performance metric or to guide the optimization process, as stated in [149]: "whenever one approximation completely dominates another approximation, the hypervolume of the former will be greater than the hypervolume of the latter." Two examples of algorithms employing the hypervolume indicator to guide the search are Indicator-Based Evolutionary Algorithm (IBEA) [152] and S Metric Selection Evolutionary Multiobjective Optimization Algorithm (SMS-EMOA) [39]. Both of these are reported to outperform established algorithms, such as NSGA-II and

SPEA2 in respective papers.

One drawback with hypervolume indicator based algorithms may be the high computational cost due to the computation of the indicator value. For this reason, with inexpensive objective function evaluations, methods of this type are best suited for problems with a small number of objectives and small populations. On the contrary, if objective function evaluations are expensive, cost of computing the indicator value may be considered negligible compared to the enhanced efficiency of the algorithm.

Performance assessment

With EMO algorithms that produce a set of solutions approximating the Pareto optimal set, evaluating the performance of a given algorithm is far from trivial. The same difficulties apply as in the case of a single objective optimization, as discussed in Section 2.1.3. Further, as mentioned earlier, to characterize the goodness of the approximation or solution set, all solutions should be as close as possible to the real Pareto optimal set and the solutions should cover the Pareto optimal set as well as possible, meaning that the distribution of the solutions along the set should be even (no excess gaps), and the extent of solutions should be as high as possible. We refer to these two properties as closeness and diversity, respectively. It is obvious that a good diversity is easier to achieve with a higher number of non-dominated solutions.

Different indicators have been proposed for both closeness (e.g., generational distance) and diversity (e.g., spacing, spread and maximum spread) [32]. If the results of several algorithms are compared using two different indicators, this may easily lead to a situation where value of one indicator is better with some algorithm, and value of another is worse. In this case, there is no way of judging which one of them is actually better.

Recently, a hypervolume indicator [154] discussed earlier has gained popularity as a performance measure. It is useful, as it condenses information about both closeness and diversity into a single value, which can be readily utilized to compare the quality of two or more approximated Pareto optimal sets. In [143] it is claimed that the fastest known algorithm for calculating the hypervolume exactly is the HSO (Hypervolume by Slicing Objectives) algorithm, presented in the same paper. One way to estimate the hypervolume is to use the Monte Carlo based hypervolume computation. In this case the hypercube spanned by the nadir and ideal points is filled with some high number of random points, and the number of those which are dominated by the current approximation is computed. We used implementation [21] of this approach in [PIV].

Another metric giving information about closeness and diversity in a single value is inverted generational distance, IGD [138]. A normal generational distance GD [136] measures how far the solutions of population P are on the average from the nearest real Pareto optimal solutions in set $P*$. To calculate GD, for every solution in P, the closest point in $P*$ is located. As to IGD, for every

solution in $P*$, the closest point in P is located. As a result, if a part of the Pareto optimal set is missing or very poorly approximated by P, the value of IGD grows. Obviously, to reliably compute either GD or IGD, a sufficiently dense set $P*$ of real Pareto optimal points is needed. Thus, these metrics are usable only with (test) problems where the exact Pareto optimal set is known. For a more thorough discussion about different performance metrics, we refer to [32, 151, 156].

In the EMO field there exist some test problem suites for performance assessment. Probably the two most often referred are the Zitzler-Deb-Thiele (ZDT) problems [150] containing six bi-objective test problems with varying number of design variables, and the Deb-Thiele-Laumanns-Zitzler (DTLZ) [36] test problems which are scalable to any number of objectives. Both of these suites contain synthetic problems, whose exact solutions are known. In addition to ZDT and DTLZ test suites, also the Walking Fish Group (WFG) test suite has been proposed in [63, 64] to increase variety and difficulty level of previous suites.

In [PIV] we solved few test problems from ZDT and DTLZ test suites, and we used the generational distance GD, the inverted generational distance IGD and the Monte Carlo based hypervolume HV [21] as performance measures.

Visualization and selection of the final solution

It is necessary to emphasize that the multiobjective optimization problem is not solved when a (good) approximation of the Pareto optimal set is produced. After that, the decision maker should be able to select one final solution, i.e., the one which is ultimately to be implemented or manufactured, among all solutions in the approximated Pareto optimal set. This is a trivial issue only in the case of two (or at most three) objectives, when visualization is straightforward, and it is easy to view all possible solutions. With a higher number of objectives it gets more difficult to represent the information of many non-dominated solutions to the decision maker, and it gets harder to explore the solutions.

In [90, 91], different visualization techniques, including methods such as scatter-plot matrices, value paths, bar charts and star coordinates are presented. Anyhow, these methods have been directed and are thus the best in presenting a small number of solutions at the time, as interpretation of these visualizations gets more challenging with the growth in the number of solutions .

Fortunately, in the literature several more or less sophisticated methods (e.g., [28, 42, 99, 134]) are presented to explore the high dimensional approximated Pareto optimal set. Additionally, in Section 3.3 we present a simple approach to explore the Pareto optimal set using a reference point based approach. Also clustering methods may be used to condense information contained in the original approximation of the Pareto optimal set, and make it thus easier to grasp for the decision maker. We used the clustering based approach in [PIII], and in the literature there exist also similar approaches in [129, 130, 131].

Further, there are also other, non-clustering based approaches, e.g., [71, 81, 88, 137], that strive to to reduce the cognitive burden of the decision maker in selecting the final solution by condensing the information contained in the ap-

proximation by different means.

In this chapter we have discussed basic concepts and approaches related to the simulation based black-box optimization. In the following chapter we discuss in more detail some challenges that solving real-life engineering problem poses to optimization systems, and we also discuss some possible responses to these challenges.

3 CHALLENGES AND POSSIBLE RESPONSES

In this chapter, we describe the main contribution of this thesis by discussing certain challenges posed by simulation based optimization problems, and by proposing respective responses to these challenges. Later, in Chapter 5, we present the chronological steps of our work in a concise manner, and point out how working first with a particular engine design problem led logically, step by step, to improve first interactive methods in general, and then to the development of both single and multiobjective optimization algorithms, as well as a clustering based method to select the final solution from the approximated Pareto optimal set.

As discussed in the previous chapters, solving real life simulation based black-box optimization problems is not a trivial task. Problems of this type may need to be solved using global, multiobjective and efficient (in terms of objective function evaluations) approaches to tackle difficulties caused by several local optima, several conflicting objectives, and computational cost of objective function evaluations, respectively. As there exists a large number of possible approaches and algorithms (different optimization algorithms, scalarizations, EMO approaches, visualizations, etc.) that can be utilized throughout the solution process, it may require some judgement to select different tools needed in different phases of the solution process. Further, simulation based black-box optimization problems may cause some challenges to methods originally developed for analytic or otherwise easier problems.

In the following, we discuss the structure of a simulation based optimization system and describe the process of creating it, and along the way we point out some challenges that may complicate either the construction of the optimization system or the solution process. Further on, we focus on the two issues we consider as the most challenging, namely difficulties caused by computational complexity of the objective functions, and difficulties of choosing the final solution among a large set of Pareto optimal or non-dominated solutions.

In general, as discussed in [PI], a simulation based optimization system consists of two main parts, namely the optimization algorithm and the part which computes values of the objective function(s), in this case, the simulator. Usu-

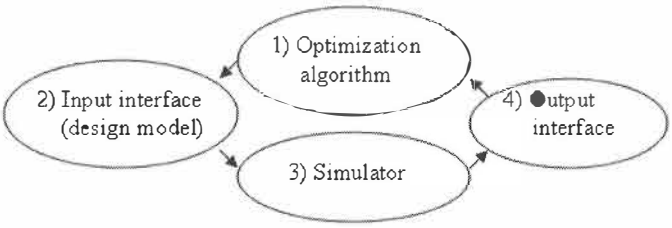


FIGURE 6 Overview of the modules of a heterogenous optimization system.

ally, the optimization algorithm cannot directly access the simulator, thus some interface from the *optimizer* (implementing some optimization algorithm) to the simulator, and vice versa, is needed. Now the general optimization system can be represented by four separate modules as in Figure 6: Optimizer (1), Input interface for the simulator (2), Simulator (3) and Output interface (4) from the simulator to the optimizer.

As the optimization system consists of several modules, it may be very heterogenous and modules may be implemented using different tools and languages, and they can even reside on separate physical computers. For seamless operation of the system, interfaces 2 and 4 in Figure 6 must be implemented with rigour to allow reliable exchange of the information between the modules. In [PI], we addressed the engine design problem, which possessed several typical characteristics of simulation based optimization problems: the problem was computationally expensive (one simulation, i.e., objective function evaluation, took several minutes), had three objectives, and was of global nature. Further, the simulator was an external stand-alone software thus being a black-box system for the optimization algorithm. To solve the problem, in [PI] we constructed a heterogenous optimization system, and all the necessary interfaces for it.

In the input interface to the simulator, six design variables were first converted into a Bezier curve defining the shape of the exhaust pipe. Then the resulting pipe shape was discretized to nine sections, and the dimensions of these were parsed into an input file of the simulator as a list of lengths and start and end diameters for each separate section. With a proper input file, the simulator run was executed as an external stand-alone program, producing an output file, which was parsed to evaluate objective function values. The output file of the simulator contained all the necessary information such as power, torque, bmep and fuel consumption for all rpm steps that were used during the simulation run. For further details about interfacing, see, [1] and [PI].

As a result of [PI], we enhanced the interactive IND-NIMBUS software in a way that with some modifications to the new interfaces it virtually enables the use of any optimization algorithm and simulator combination. For its representative characteristics, we utilized the same problem also in papers [PII] and [PIII].

First, when some simulation based optimization problem is considered, the problem must be modelled in an appropriate form so that optimization tools can

be applied, i.e., we must be able to change the design, control the process, etc., by varying design variable values, and then we can check the resulting objective function value reflecting the goodness of the current design. It may be useful to point out the existence of two different models, one being the *model of the phenomenon* (which is usually implemented as the simulator), and another one, the so-called *design model*, which is a model for mapping design variable values for proper input configurations (i.e., input interface) for the simulator, see item 2 in Figure 6. For example, in our engine design problem discussed in [PI], [PII] and [PIII], the phenomenon modelled by the simulator was the behavior of a two-stroke internal combustion engine, and our aim was to affect the power output of the engine by altering the shape of the exhaust pipe, which was modelled using a Bezier model (see [1] and [PI]) having six design variables.

In the design modelling phase there are already two often conflicting requirements: the design model should have as few design variables as possible to reduce the size of the search space, and at the same time it should be flexible enough to be able to represent all possible and necessary design configurations. For example, in [1] we developed four different models to represent the shape of the exhaust pipe of the two-stroke engine. Models were varying in their flexibility to cover different shapes and in their number of design variables.

It is necessary to emphasize that by means of optimization it is not possible to cure the weaknesses of either the design model or the simulator. For example, if the design model is too rigid, incomplete or contains unnecessarily high number of design variables, or if the simulator is very inaccurate, even the best of optimization algorithms cannot overcome these difficulties and produce good solutions. For this reason, it is central that design models and simulators are implemented with rigour, and that they are by some means verified to possess a sufficient accuracy.

After the simulator is chosen or otherwise obtained, and the design model of the phenomenon to be optimized is created, it is necessary to select a proper optimization algorithm, based on the characteristics of the problem and the design model. These issues are discussed in Section 3.1.

Often the computational complexity in evaluating the objective function(s) may hinder the solution process in different ways. Obviously, the use of efficient algorithms is always beneficial, and other possible ways to reduce the computational burden are welcome where simulation based problems are concerned. We discuss these issues in Section 3.2.

Finally, especially with approaches producing an approximation of the Pareto optimal set as a finite point set (as is the case with EMO approaches), it may not be straightforward to select one of the solutions as the final one. This is discussed in Section 3.3. To summarize, in the following sections we discuss the above mentioned challenges, and point out some possible improvements we have realized during this work.

3.1 Problem of algorithm selection

As we have seen in the previous chapters, there are different types of optimization problems with many characteristics, e.g., problems of global and multiobjective nature. These can be further divided into different subclasses based on certain properties of the problem. With this perspective, it seems plausible that there exists no single optimization algorithm which is superior for all types of problems. For this reason, whenever there is available some a priori information about the problem, that should be exploited.

For example, if we know that the problem is convex, it is not reasonable to solve it using genetic or other evolutionary algorithms, because some efficient local optimization algorithm would solve it significantly faster. Also, if gradient information is available, that should be put to use.

In the case of simulation based optimization, we may usually assume that the problem is nonconvex. This assumption can be verified, for example, by taking few different starting points, and executing local optimization runs from each of them. If the results are clearly different, the algorithm has found separate local optima, and the problem requires global optimization methods.

With simulation based optimization problems, it may be very difficult to obtain information about special characteristics of the particular problem to be solved. This is probably one reason of the popularity of different heuristics and evolutionary algorithms; they are known to perform reasonably well with a wide variety of problems, at least if some effort to tune the parameters (e.g., CR and F in case of DE) of the algorithm is acceptable. The drawback often associated with these methods is a rather high number of objective function evaluations required, which may not be acceptable if the evaluations are costly.

The cost of objective function evaluation may often be one important criterion in the selection of an optimization algorithm. Inexpensive problems, where objective function evaluation takes virtually no time at all, can usually be solved using known heuristics or evolutionary algorithms, for example, differential evolution, genetic algorithms, ant colony optimization, controlled random search, etc. [53, 104, 124]. More expensive problems, where one objective function evaluation takes something from a few seconds to some minutes, can be probably best solved using more intelligent approaches, for example surrogate assisted evolutionary algorithms (e.g., [20, 41, 100, 111]). And finally, very expensive problems, where one evaluation may take several hours or even days, benefit most of more sophisticated (and also more complex) surrogate approaches, such as EGO, where all sampled information is exploited to its full extent. We have touched this topic in [PIV].

If similar types of problems are to be solved regularly, it may be reasonable to do some testing with different algorithms, as well as with different parameter settings for a certain algorithm. For example, the performance of the DE may be altered significantly by adjusting its parameters. E.g., in [1] we made a comparison of several global optimization algorithms in solving an engine design

problem. Based on the results of this comparison, in further research related to the same problem we could use best performing algorithms.

In the case of multiobjective optimization problems, we may encounter even more difficulties. One basic choice to make is whether to use a scalarization based approach (interactively) or some approximation method. If the approximation method is some EMO algorithm, it may not work efficiently if the number of objectives is high. This is especially the case with dominance based EMO's [65]. On the other hand, computational overhead of the indicator based EMO's may prove to be inacceptable in some cases. Further, in case of scalarization based approaches, as discussed in Subsection 2.2.1, different scalarizing functions tend to produce different solutions, and also the selection of a single objective solver is not straightforward, as discussed above. As in the case of single objective optimization, also with multiobjective optimization, one key aspect in choosing the solution approach may be the cost of objective function evaluations. These issues are discussed in the following section.

3.2 Computational complexity

As discussed earlier, computational complexity often hinders the solution process. Although this problem is central in single objective optimization, it is probably even more so in the context of multiobjective optimization. Both of these cases will be discussed in the following subsections.

3.2.1 Single objective optimization

Some simulation based optimization problems are inherently single objective, and in addition, often multiobjective problems are converted into single objective ones by scalarizing them.

Because simulation based objective function evaluations may often be costly, as efficient methods as possible should be used to solve single objective problems. Sometimes with very efficient algorithms, computational overhead of the optimization algorithm itself may prove to be prohibitive compared to the cost of objective function evaluations. This is possible, e.g., with sophisticated surrogate based algorithms such as EGO (e.g., fitting the surrogate model is expensive), and we noticed this kind behavior in the comparison made in [1].

In [PI] and [PII] we solved a multiobjective engine design problem using the interactive NIMBUS method and we noticed that solving the scalarized subproblems using the CRS2 or the DE algorithm took several hours or over night, which was quite tedious for the decision maker. For this reason, in [PV] we tackled the problem of improving the efficiency of a population based single objective evolutionary algorithm by filtering inefficient trial points away. As discussed earlier, the basic idea of meta-modelling approaches seems reasonable; all evaluated points are utilized in judging where the next sample point should be located. On

the other hand, model management may be a problematic issue, as may be fitting of the surrogate model, and further, implementation of algorithms of this type may be difficult. Bearing in mind pros and cons of meta-modelling approaches, we strived for some improvement.

We had noticed earlier in [119] and [PIV] that the point generation mechanism of often employed evolutionary algorithms, DE in this case, is not working efficiently. Instead, DE generates trial points in the regions of the search space where improvements in the objective function value are not probable (see, [119] and [PIV]). To filter out these points, in [PV] we used an idea of a surrogate incorporating information contained in all previously sampled points, but instead of a common and possibly costly approach of kriging or artificial neural networks, etc., we used a simple method based on the nearest neighbor interpolation to estimate possible improvements in the objective function value. More specifically, for each parent in the population one or more trial points are generated, the one with the best predicted value is selected, and if the best predicted value is not better than the parent's objective function value, the trial point is filtered away, i.e., excluded from consideration, and no expensive objective function evaluation is made for it.

The rationale to generate more than one trial point for each parent is to extract more information about the search space using the inexpensive surrogate, and thus to make the search more effective. In our approach, instead of explicitly optimizing the surrogate, we simply select the trial point with the best predicted objective function value. By varying the number of generated trial points, the user may balance between efficiency of the search and the cost of the objective function evaluation; with expensive objective function evaluations it is reasonable to explore the surrogate more extensively by selecting a higher number of trial points to be generated.

In the prediction of the trial point value, we do not merely interpolate objective function values between already known points, but instead incorporate also some information about the uncertainty of the prediction, based on the objective function value variations in the current population. In principle, the uncertainty of the prediction increases as the distance from the predicted point to the nearest known point increases, and if the variation in objective function values in the current population is high, also uncertainty is high. Using predicted objective function values, the balance in the proposed algorithm between local and global search is realized in a similar fashion as in the EGO algorithm. In this way, the trial point in a large unsearched area with relatively bad neighboring objective function values may get better predicted value (due to higher uncertainty in the prediction) than the trial point located in a more crowded area with better neighboring objective function values.

The proposed filtering approach in [PV] does not require any model management procedure, as all evaluated points are always kept in memory, and there is no possibly time consuming surrogate fitting involved. Further, implementation of the algorithm is very straightforward. The performance of the proposed approach was compared to several other variants of the DE using well-known

test problems from the literature, namely Rosenbrock, Michalewicz, Rastrigin, Griewangk, Ackley and Levy test functions. These test problems were selected because they pose different difficulties for the optimization algorithms. Each variant of the algorithm was utilized in 100 independent runs using 2, 5 and 10 design variable versions of the functions, and allowing a maximum budget of 500, 1000 and 2000 objective function evaluations, respectively. This budget was chosen because we are interested in solving expensive problems, where the number of objective function evaluations must be kept in a reasonable level.

In the light of the above described setup, the proposed approach produced notable savings in the number of required objective function evaluations, and the computational overhead of this approach remains negligible compared to the cost of evaluating expensive objective functions. Based on the tests, the performance of the proposed approach seems to be somewhere between traditional EA's and real surrogate algorithms, and thus it is best suited for problems with a mediocre cost for objective function evaluations (taking something between a couple of seconds and a few minutes). Further, the approach of generating more than one trial point for each parent seemed promising, as the use of a modest number of four trial points produced notable convergence speed-up when compared to the use of one trial point only. Anyhow, it remains for further study to show how the performance of the proposed algorithm is related to increased number of trial points in general.

A similar approach to the one proposed in [PV] could be also incorporated with the approach proposed in [PIV] (discussed below in Subsection 3.2.3), hopefully further enhancing the performance of the latter. Objective function values for each of the objectives could be estimated in a similar fashion as in [PV]. The already known objective function values (for all objectives) could be stored and maintained in a single matrix along with the respective decision variable values. In this way, while locating the nearest neighbor for some new trial point, the values for all the objectives at the nearest point would be found with the same effort as in the case of a single objective. Thus, the procedure would scale up with the number of objectives well.

With the predicted objective function values it could be judged whether the trial point would dominate already existing solutions or not, and thus one could decide if it is reasonable to evaluate it or not using the expensive objective functions. It is open to further research whether this combined approach would work or not. One possible problem lies in the estimation of the measure of irregularity L, which is in [PV] computed from the current population. In the single objective case, the population is contracting during the optimization run, and also the value of L is decreasing. This would not necessarily be the case in the multiobjective approach, where the population is expected to be as diverse as possible. In this case, different means to compute L would be required, for example, using some number of nearest neighbors of the trial point, etc.

3.2.2 Scalarization based methods

Often scalarization based methods are most naturally used in a recurrent fashion, i.e., interactively. Some preference information is defined and used to form a scalarized subproblem, a solution to the subproblem is produced, and after this, the decision maker may refine the preference information, and the loop is started once again. As a consequence, the decision maker learns about how the problem behaves and may finally find a solution that pleases him/her the most.

As discussed earlier, if the problem is to be treated in a global fashion, the subproblems involved must be solved using global optimization algorithms. With costly problems, this is obviously rather time consuming, and it is obvious that the interactive nature of the process suffers, that is, the decision maker has to wait too long for the new solutions to be generated before he/she can define new preferences. This behavior was evident in [PI], where scalarized problems were solved using the CRS2 algorithm.

In [PII], we tackled the aforementioned hindrance by increasing accuracy as the interactive NIMBUS solution process progresses instead of replacing the single objective optimization algorithm with a more efficient one. Our rationale behind this approach was that at the beginning of an interactive optimization process, the decision maker is many times also at the beginning of a learning curve: he/she is learning how different objectives are interrelated, what the trade-offs between the objectives are and, also, what kind of objective function values can be reached in general. It seemed safe to assume, that in the learning phase the accuracy of the solution can be sacrificed to some extent in order to save a significant number of objective function evaluations.

After the learning phase, the decision maker has a better understanding of the problem behavior, and he/she can decide what the most satisfactory objective function values for the final solution are. In our approach, we demonstrated that the computational cost can be affected by adjusting the required accuracy of the solutions of the single objective optimization algorithm (by using a predetermined threshold for objective function evaluations in each iteration of the NIMBUS method) during the interactive solution process.

Based on the assumptions above, at the beginning of the optimization process, quite a coarse accuracy of solutions may be used, and only a rather small amount of objective function evaluations is needed to get a general understanding about the problem. As the decision maker learns more about the problem and its behavior, and consequently feels more confident about it, he/she may approach the final solution with an ever increasing accuracy and with a higher number of objective function evaluations. As the final step of the optimization procedure, the final solution can be calculated using the budget that assures the decision maker of a good enough solution quality.

As deciding the proper budget for objective function evaluations in advance is not trivial (although the decision maker has probably gained some feeling about the budget needed in previous iterations of the NIMBUS method) we proposed a measure called *maximum difference percentage* (MDP) in [PII] to assess the

quality of each solution produced during a single objective run (used to solve the scalarized subproblem of the NIMBUS method) with regard to a given classification.

As the name suggests, the maximum difference percentage is the maximum difference between the components of the current solution and the given reference point (with desirable objective values) as percentages for each of the objectives, and formally it is calculated (in the case of minimization) as

$$MDP = \max_{i=1,\ldots,k} \left[\frac{f_i(\mathbf{x}) - \bar{z}_i}{\bar{z}_i}\right], \quad z_i \neq 0. \tag{9}$$

This formulation bears some resemblance with the structure of the achievement scalarizing function presented in Subsection 2.2.1.

While the single objective optimization run progresses, the values of MDP are continuously plotted, and the decision maker can extract different types of information about the status of the current optimization run. The decision maker can see how far the solution is from the given reference point measured as percentage, whether the values are improving in general or if the process has already stagnated, and how much there is variation among adjacent solutions produced by the optimizer. Using this information, and also information absorbed in previous iterations, the decision maker can decide when a sufficient number of objective function evaluations has been reached. Naturally, if online monitoring of the process is not possible, some predetermined budget or stopping conditions based on the MDP may be constructed. Further discussion of these topics is given in [PII].

With the proposed approach that uses increasing accuracy, we could reduce the number of objective function evaluations by some 60-70% compared to the normal NIMBUS approach without deteriorating the quality of the overall solution process. These results are reported in [PII].

Another natural way to save in objective function evaluations with the interactive approach would be to utilize all information obtained in previously made objective function evaluations. Every time a subproblem is solved, lots of information about the behavior of the objective functions is produced. Thus, during the optimization process, an increasing amount of data is produced (e.g., objective function values with regard to design variable values). This data could be used in solving the following subproblems, for example, by computing scalarizing function values for all evaluated points with a new preference information (very cheap operation), and then selecting a proper subset of these to be used as an initial population in a single objective optimization algorithm that solves the current subproblem. In this way, the interaction between the decision maker and the optimization system would be more fluent, and the use of objective function evaluations more sparing. Further, all evaluated data could be used to predict values of unknown points, as can be done with the EGO algorithm or in the algorithm proposed in [PV]. Intelligent implementation and possible integration to the NIMBUS method of both of the aforementioned ideas is open to further research.

3.2.3 EMO approaches for multiobjective optimization

In contrast to scalarization based approaches, in EMO approaches, an approximation of the Pareto optimal set is created without any user intervention, which makes the basic idea of such approaches rather appealing; the decision maker is involved in the solution process only after the most time consuming computation is finished. Anyhow, if creating the approximation takes a large number of objective function evaluations, even this offline waiting period may be considered prohibitive. For this reason, efficiency is one of the key issues also with EMO approaches.

In [PIV], where our aim was to develop a more efficient EMO algorithm, we pointed out some difficulties with widely used dominance based EMO approaches. . We summoned up some issues such as convergence problems, deterioration of population, and difficulties in choosing the proper population size. Also diversity maintenance seemed to be problematic. We discuss these issues in the following paragraphs.

As discussed earlier in Subsection 2.2.2, convergence problems [58, 83, 115, 116, 117, 118] of widely used EMO algorithms stem from the concept of dominance, in conjunction with the diversity preservation mechanisms used. If there is an excess of non-dominated solutions to fit into the population, some of them must be pruned using a diversity preservation mechanism. In this case, some non-dominated solution located very near the Pareto optimal set may be replaced by some other non-dominated solution which improves diversity but is at the same time located much farther from the Pareto optimal set. This behavior leads to oscillation in the solution quality, and prevents convergence to the Pareto optimal set.

By deterioration of population we mean that, in the history of all evaluated points during the optimization run, there exist solutions that dominate solutions in the current population. This behavior is closely related to the aforementioned convergence problems. When the population deteriorates, we can say that some number of objective function evaluations has been wasted, and the population does not reflect the best possible solutions encountered during the optimization process.

Choosing the proper population size is not straightforward, because too few points cannot represent the characteristics of the Pareto optimal set properly. On the other hand, too many points may hinder the performance, because a large portion of the population may consist of dominated solutions, especially in the beginning of the optimization process when there are not enough non-dominated solutions found to fill the population. The population size is also related to the convergence problems, and deterioration. Obviously, the larger the population, the closer to the Pareto optimal set the algorithm may converge before the population starts to oscillate. This behavior was demonstrated in [PIV].

Diversity maintenance may also be a problematic issue. For example, the crowding distance of NSGA-II [34] (used also in other algorithms, e.g., [113, 114]), actually works only in two dimensions, and may fail also in this case [78]. In [79],

a viable way to implement the pruning needed in the diversity maintenance is presented. A relatively high computational cost may be considered as a drawback of this approach, but this in turn is probably not very significant if objective functions are costly.

In the proposed algorithm in [PIV], we utilize the point generation mechanism of DE (as it is known to perform reasonably well among its peers), but the focal point of our approach is that we give up the idea of a fixed population size, and instead keep all the non-dominated solutions in the population. In this way, all problems discussed above can be overcome, at least to some extent. The convergence to the Pareto optimal set is gained because the population cannot oscillate. There is no need to select the population size, and because of that in the beginning of the process the algorithm performs efficiently because the population is not filled to some predetermined size with bad quality dominated solutions. Also the number of solutions in the population is increasing over time, thus having a better capability to capture the characteristics of the Pareto optimal set. The population of the proposed algorithm cannot deteriorate, because it always contains all non-dominated solutions. Further, no explicit diversity preservation mechanism is needed. This may seem counter-intuitive, but it is essential to realize that neither NSGA-II nor DE-based EMO's (see, for example, [80, 147]) strive to actively generate evenly spread solutions around the Pareto optimal set, rather they just select solutions located in less crowded regions to following populations. Thus, if all solutions are kept, also the diversity is maintained at least as well as, for example, in NSGA-II.

As discussed in Subsection 2.2.2, diversity is often used as a secondary selection operator, and in this case the selective pressure of evolutionary mechanisms fails to drive the population towards the Pareto optimal set when there is a higher number of objective functions involved. We assume that the performance of our approach does not deteriorate with an increased number of objectives in a similar fashion as in traditional approaches, because we do not utilize any diversity preservation mechanism but instead rely merely on the dominance of the solutions. However, no empirical tests have yet been conducted to verify this assumption.

Because the size of the population is increasing all the time in our approach, it is obvious that not every point in that population can serve in turn as a target point, as in DE. For this reason, all four points involved in generating the next trial point are selected at random. Non-dominated solutions may be identified either after every new trial point is evaluated, or alternatively after some predetermined number of trial points is evaluated. Obviously, with very expensive objective functions it is reasonable to identify non-dominated solutions as often as possible, thus allowing the use of the most up-to-date information.

With the proposed algorithm all data about the behavior of objective functions gained during the optimization run is put in use, in contrast to several EMO approaches which exploit only information contained in the current population, and possibly in some rather small additional external archive. Intuitively our approach feels advantageous, and numerical results presented in [PIV] comparing

the performance of NSGA-II and the proposed algorithm also support this conception, as the proposed algorithm clearly outperformed NSGA-II. The NSGA-II algorithm was selected as a counterpart to the performance test, because it is widely utilized in several different fields, and in that respect it can be considered as *de facto* algorithm today. In performance comparison we used three bi-objective test problems (ZDT1, ZDT2, ZDT4) and three tri-objective test problems (DTLZ2, DTLZ5, DTLZ7) in 30 independent runs with a budget of 8000 objective function evaluations for each run. The choice of the stopping criterion was motivated by limitations of real engineering problems where the solution typically must be obtained within some specific budget, and in this sense we were more interested in how the algorithms behave with quite low numbers of evaluations.

It is open to discussion whether our approach is appropriate or not with problems with very inexpensive objective function evaluations, as the number of all nondominated solutions may grow very high and eventually lead to an increased computational overhead of the algorithm itself. In that case, it could be more efficient to use some fast running approach with a bounded population size. On the other hand, as discussed earlier, at some point the traditional approach is likely to start to oscillate, and after that it can no longer improve the solutions. Thus, to ultimately judge which algorithm to apply it is necessary to ponder different aspects of the solution process, i.e., quality of the approximation set, wall clock time used, number of required objective function evaluations, etc.

In the single objective field, meta-modelling has been utilized for a long time to reduce the number of objective function evaluations needed. Similar approaches may be applied also in the multiobjective case, for example, by simply using separate surrogates for each of the objectives (which may be somewhat costly) or by representing the multiobjective problem with one surrogate (which may be difficult). Intuitively, meta-modelling based multiobjective optimization approaches seem plausible, and it seems also that there is an ascending trend to improve the efficiency of multiobjective algorithms with a meta-modelling scheme [40, 74, 76, 85, 89, 140].

3.3 Selecting the most preferred Pareto optimal solution

With EMO algorithms, it may be necessary to use relatively large population sizes to be able to capture and represent the behavior of the Pareto optimal set. With the algorithm proposed in [PIV], the population size is allowed to increase on purpose. When the approximation of the Pareto optimal set contains a large number of points it may become cognitively very challenging for the decision maker to select a certain solution as the final one. As stated earlier, in the literature several methods (e.g., [28, 42, 99, 134]) have been presented to explore the high dimensional Pareto optimal set.

In [42] and [99] (and implementation of it [28]) the polyhedral approxima-

tion of the Pareto optimal set is generated using a previously obtained finite set of solutions. The user can then navigate through the approximated Pareto optimal set and get a grasp on the problem behavior. These two methods are differing in the way how the navigation is implemented. In [134], a closed polygon within a star-shaped scale arrangement visualizes Pareto optimal solutions as a so-called spider web chart. The decision maker can move through the solutions by pulling at a vertex of the navigation polygon, at a so-called grip point, towards smaller or larger values (inwards or outwards). As a result, the polygon moves to a neighboring polygon, representing another Pareto optimal solution.

In [PIII], we take a different approach to many Pareto optimal or non-dominated solutions, and use advanced clustering algorithms to reveal the essential characteristics of the Pareto optimal set. This is done by clustering [43, 72] the finite point set approximating the Pareto optimal set, and then using only a handful of solutions closest to the cluster prototypes (the most representative points in each cluster) as representatives of the Pareto optimal set. In this way, the decision maker is involved in the solution process only after most of the time consuming computation is finished. Because of the small number of representative solutions, it is easy for the decision maker to get grasp of the problem behavior on a general level and direct his/her interest into some specific cluster, wherein solutions can be studied in more detail.

In our approach, we use robust clustering methods [5] to avoid sensitivity to deviating or erroneous data and to get a global solution for the clustering problem. After clustering, the cluster prototypes are projected on the Pareto optimal set (using the scalarizing function (5)), to get information about the quality of the current approximation, i.e., how far the prototypes are from the Pareto optimal set. All clustering data is displayed to the decision maker, as discussed in the following. A two dimensional plot (produced using principal component analysis) of cluster prototypes and solutions belonging to each cluster is displayed to the decision maker, who can get an understanding of how solutions are distributed to clusters. Further, for each cluster, all solutions belonging to that cluster are displayed as value paths [90, 91], with respect to both design variables and objective values. Design variable plots help the decision maker to judge whether the solutions are robust or not. If there is only a small variation in some design variable value, it suggests that the solution is sensitive to changes in that particular value; if the value is changed, the solution belongs to another cluster. The objective value plot shows to the decision maker how similar or how different solutions are within one cluster.

Often multiobjective problems are studied merely in the objective space. However, values in the design space may contain some useful information. For this reason, in our approach it is possible to further cluster solutions within some specified cluster, this time with respect to design variable values. This may help to identify different design families, leading to same kind of end results. Obviously, some design variable value combination may be more appealing to the decision maker considering that the end result is almost the same. For example, in the case of the exhaust pipe design for the two stroke engine, if two pipes, one

thicker and one thinner, produce almost exactly the same end result, it may be more convenient to fit the thinner pipe to the frame of the motorcycle.

In [33, 35], analysis of interaction between design and objective spaces has been taken even further. In that approach, Pareto optimal solutions are analyzed to investigate if there exist some common principles among all or many of the solutions. If some relationship between design variables and objective function values is found, such information can be used in a general level to decide how some design can be done in an optimal manner, probably even without resorting to solving a completely new optimization problem again.

The clustering based method proposed in [PIII] bears some resemblance to approaches presented in [129, 130, 131]. Our method differs from these primarily in a way that we use robust clustering algorithms, we study the problem behavior also in the design space, and we present the information to the decision maker using various intuitive plots.

In addition to methods described above, there are also other, non-clustering based approaches, e.g., [71, 81, 88, 137], that strive to reduce the cognitive burden of the decision maker in selecting the final solution by condensing the information contained in the approximation by different means. In this thesis we did not consider these methods, because they often seem to require at least some prior understanding of the problem: for example, the decision maker should be able to rank the objectives according to their importance, or define thresholds for significant tradeoffs, etc.

To complement other methods capable of exploring the approximation of the Pareto optimal set, we now present one additional, rather simple approach, which works on a previously created finite approximation set P, iteratively with user defined reference points. In this way, the decision maker can explore the whole solution set, even in higher dimensions. The proposed approach works as follows:

1. Compute the ideal (best seen values in P for all objectives) and nadir (worst seen values in P for all objectives) vectors.

2. Compute the mean value of all objectives (objective-wise), and use this vector initially as the reference point p_{ref} giving a neutral starting solution.

3. For a given p_{ref}, select four similar points in set P. The first of these is the point with the minimal Euclidean distance between points in P and p_{ref}. The other three are located along the projections from p_{ref} to P, and they are selected using three different scalarization functions, namely satisficing trade-off method (STOM), achievement scalarizing function, and GUESS (see Subsection 2.2.1). For each scalarization, the point which produces a minimum scalarized function value is selected. With this procedure it is possible to produce up to four different solutions. However, there is no guarantee that all four are different.

4. The objective function values of the four selected points are displayed to the decision maker, along with the Euclidean distance information to the given p_{ref}. From this information the decision maker can deduce what the projected solutions are like, and how far they are from p_{ref}, and decide whether he/she wants to further refine current p_{ref} or not. If the decision maker is satisfied, the process is stopped. Otherwise, the decision maker defines a new p_{ref} according to his/her wishes, and the process continues from Step 3.

As can be seen, the whole procedure bears close resemblance to interactive multiobjective optimization methods, for example such as NIMBUS, and also the method in [134], where a set of previously produced solutions is navigated. The major difference to the NIMBUS method is that after the reference point is defined, in the system proposed here, no optimization is done (as the solution set is already available). Instead, best points with regard to a given reference point are selected. With computationally costly problems this is a major benefit, as all time consuming computation is done in advance. Compared to a method in [134], in our approach four different solutions "near" the given p_{ref} are displayed, which may give additional information about the variety in solutions to the decision maker.

Further, the decision maker is supported with information about distances between p_{ref} and four selected points to get a grasp on how close to some already known solution the given p_{ref} actually is. This may help the decision maker to judge how probable it is to find a solution similar to p_{ref}.

4 AUTHOR'S CONTRIBUTION

In the first paper [PI], the problem setting of simulation based optimization is described in a general level, and in particular a computationally expensive triobjective problem of optimizing the performance of a two-stroke internal combustion engine is solved with the interactive NIMBUS method. During this research, the IND-NIMBUS software was enhanced with new input and output interfaces in a way that it enables the use of virtually any simulator and optimization algorithm combination. The author implemented all the connections between different parts of a very heterogenous optimization system consisting of the optimization algorithm, the simulator, and wrappers to turn design variables to simulator input files, and also for deriving objective function values from the simulator output files. Further, the author had developed four different models to represent the exhaust pipe shape in [1], and used one of them with only a modest number of design variables in this study. In the optimization model, also the performance of the whole vehicle was taken into account (by incorporating the effects of gearbox ratios), instead of only engine characteristics. Thus, the engine design problem with a somewhat more practical value than in some previous studies was solved in a multiobjective manner. The paper was written mainly by the author. Professor Kaisa Miettinen brought in her expertise in multiobjective optimization, as well as providing some perspectives and finishing touches.

The second paper [PII] is based on the author's observations about difficulties in solving computationally expensive problems in an interactive fashion in [PI]. The engine design problem was used as an example also in this paper, along with the NIMBUS method. The author developed an idea of using coarse accuracy (and less objective function evaluations) to solve NIMBUS subproblems in the beginning of the interactive solution process when the decision maker is only in the beginning of the learning process about the problem at hand. Further in the process, as the decision maker gains understanding and starts approaching the final solution, the accuracy is improved. Also some tools were proposed for judging when a sufficient number of objective function evaluations has been reached. The above mentioned ideas were matured with Professor Miettinen, and the paper was written mainly by the author.

After using scalarization based methods in [PI] and [PII], the author did some experimenting with sampling procedures and EMO approaches, which did not require recurrent involvement of the decision maker during the process. However, it became clear that the selection of the final solution among the large set of non-dominated solutions is not necessarily an easy task. For this reason, in [PIII] the author developed an idea that essential characteristics of the Pareto optimal set could be captured using clustering methods, and thus one could reduce the cognitive burden of the decision maker in the selection of the final solution. The engine design problem was used as an example also in this paper. In this work, Dr. Sami Äyrämö brought in his expertise about advanced clustering algorithms [5], along with required implementations. The paper was written mainly by the author, excluding the parts concerning clustering, which were exclusively written by Dr. Äyrämö. Professor Miettinen contributed to structuring the ideas and material, and provided her expertise in multiobjective optimization.

Contribution in [PIV] is based on the author's ideas about how the efficiency of EMO approaches could be improved. The author had evidenced that several EMO approaches have detrimental characteristics, such as lack of convergence, deterioration of the population, difficulty to select a proper population size, and partly due to these, poor efficiency. To overcome these deficiencies, the author developed, implemented, and performance tested an algorithm which essentially utilizes an unrestricted population size. The paper was written mainly by the author, while Professor Miettinen brought in some perspectives and finishing touches.

Finally, [PV] is based on the author's observations on the behavior of evolutionary algorithms - mainly on how their trial point generation mechanisms work. The author noted that a large number of trial points are generated in regions of the search space where improvement of the objective function value is not probable. To improve this behavior, the author developed an idea of filtering out inefficient points, by using and simplifying some ideas borrowed from meta-modelling schemes. To be more specific, more than one trial point may be generated, and only the best one is evaluated using the real expensive objective function. The implementation of the algorithm and all the performance testing were done by the author. The paper was written by the author, while Professor Miettinen brought in some perspectives during proof reading.

5 CONCLUSIONS AND FUTURE WORK

In this thesis, we have discussed special features and challenges of simulation based black-box optimization. The characteristics of these types of problems suggest that we need to use global, multiobjective and efficient (in terms of objective function evaluations) approaches to tackle the difficulties caused by several local optima, several conflicting objectives, and computational cost of objective function evaluations, respectively.

In this work, our main emphasis has been on dealing with two issues: problems due to the computational complexity of objective functions, and difficulties of choosing the final solution among a large set of Pareto optimal or non-dominated solutions. As a result of the research, we have developed a heterogeneous optimization system for simulation based problems, an approach to reduce the computational burden of the interactive method, a clustering based method to select the final solution from the approximated Pareto optimal set, and two new optimization algorithms (one for global single objective optimization, and another for multiobjective optimization).

In the beginning of the research, we considered the problem of optimizing the performance of a two-stroke internal combustion engine by optimizing the shape of the exhaust pipe of the engine in [PI]. This problem is computationally demanding (one simulator run, i.e., objective function evaluation, takes several minutes), it has three objectives, is of global nature, and as such it was a good example of multiobjective simulation based engineering problem. To solve this problem, we used the IND-NIMBUS software implementing the interactive NIMBUS method, and developed some new functionality to it to allow aggregation of a heterogenous optimization system. In this work, we noticed that its computational complexity hinders the interactive nature of the solution process, as the decision maker had to wait quite long time before he could continue the solution process.

To improve the efficiency of the overall interactive process, in [PII] we continued with the same problem as in [PI], again using the IND-NIMBUS software. This time our aim was to reduce the computational burden, and to make the overall optimization process more fluent to the decision maker. Our basic idea

was that in the beginning of the interactive solution process, the decision maker is in the beginning of the learning curve about the problem, and thus it is not necessary to solve scalarized subproblems very accurately. Further in the process, when the decision maker has learned what is possible to gain, and what is not, the scalarized subproblems of NIMBUS are solved with a higher accuracy to obtain the final solution. By this approach we could save some 60-70% of the objective function evaluations compared to the previous approach presented in [PI], without compromising the quality of the final solution. It is worth pointing out that the ideas presented in [PII] can be used with other interactive methods as well.

After dealing with scalarization based approaches, we continued in the direction of approximation methods, and to be more accurate, methods such as sampling procedures or EMO approaches producing an approximation of the Pareto optimal set as a finite point set. In this approach, the decision maker is involved in the optimization process only after the most time consuming computation is finished, and tedious waiting periods are avoided. Anyhow, we noticed that it might be difficult for the decision maker to get a grasp of the behavior of the Pareto optimal set, and select one solution as the final one. To overcome this difficulty, in [PIII] we proposed an approach where essential characteristics of the Pareto optimal set are condensed by use of advanced clustering algorithms, and only handful of the most representative points of the Pareto optimal set are presented to the decision maker. In this way, the decision maker may concentrate his/her interest in the most interesting part of the Pareto optimal set conveniently.

In [PIII], we had noticed that the approach of approximating methods seems appealing, but computational cost often associated with EMO approaches seemed disadvantageous. For this reason, in [PIV] we delved into the basic structures of EMO approaches, and noticed some deficiencies in their functioning, such as lack of convergence, ignorance of some good solutions, and difficulty in selecting a proper population size. With our approach, essentially utilizing a population of a varying size, we could overcome several deficiencies, and also improve the efficiency over that of NSGA-II, which is often regarded as one of the state-of-the-art EMO approaches.

In the process of developing our new EMO approach, we had to carefully examine also the functioning of single objective evolutionary algorithms as they utilize similar trial point generation mechanisms. We noticed that the point generation mechanism of DE is not working as efficiently as it could; it seemed to produce trial points in regions of the search space where gains in objective function values seemed highly unlikely. To overcome this behavior, in [PV] we proposed an algorithm where trial points are rejected or accepted based on their predicted objective function value. In the prediction, we used some ideas borrowed from meta-modelling approaches, but to avoid computational overhead associated for example with kriging, we used a simple prediction based on the nearest neighbor interpolation. With this approach, we could gain notable savings compared to some efficient versions of DE.

Although we have proposed some improvements to make simulation based

optimization processes more fluent, there still remain several issues for further study. In the interactive approaches, where the scalarized subproblem must be solved several times, and usually in a global fashion, all objective function evaluations made during the whole process should be exploited to their full extent. In this way, information produced in solving previous subproblems, could be employed in solving further subproblems. It is open for further study how this could be done most efficiently, but obviously the efficiency of the approach presented in [PII] could be further improved, as more information about the behavior of the objective functions is gained during the solution process.

Our EMO approach presented in [PIV] could benefit from a filtering approach similar to that used in [PV], which itself should be tested more extensively to find out how the performance of the proposed algorithm is related to the number of trial points that are generated for each parent. Further, it would be useful to be able to provide guideline or even implement self-adaptivity in the algorithm itself to balance between the cost of the objective function evaluation, and the number of trial points used.

On a general level, we feel that meta-modelling based approaches are very powerful, and will probably become the de-facto algorithms of the future. For this reason, we want to develop our simple nearest neighbor interpolation based meta-modelling scheme further, and we consider the low computational overhead of it advantageous. Further, in the case of multiobjective optimization, we feel that meta-models could be used also within interactive schemes in addition to all evaluated points, thus reducing the need for expensive original objective function evaluations. With regard to EMO methods we feel that approaches based on the concept of dominance and some diversity measure of individual points are on the way out, and giving way to different indicator- and set-based approaches (as, e.g., in [155]), and we also want to move our research efforts to that direction. To conclude our future plans, we state that every objective function evaluation must count, and based by our current knowledge it seems that this is best achieved by exploiting meta-modelling schemes and indicator based EMO's in various combinations.

YHTEENVETO (FINNISH SUMMARY)

Tämän väitöskirjan, "Lähestymistapoja monitavoitteisen, globaalin ja simulaatiopohjaisen optimoinnin haasteisiin", motivaationa ovat olleet käytännön sovelluksissa kohdattavien simulaatiopohjaisten optimointiongelmien hankaluudet. Usein järjestelmän, laitteen, tai prosessin toimintaa voidaan kuvata simulaattorilla, mutta simulaattori itsessään ei kerro kuinka järjestelmän toimintaa voitaisiin tehostaa eli optimoida. Simulaattorissa, kuten oikeassa elämässäkin, järjestelmän toimintaan voidaan vaikuttaa tiettyjen suunnittelumuuttujien arvojen valinnalla. Järjestelmän optimoimiseksi on tarpeen löytää optimaaliset arvot suunnittelumuuttujille, mutta tämä on yrityksen ja erehdyksen menetelmällä inhimillisesti katsottuna erittäin hidas, kuluttava sekä virheherkkä prosessi, eikä lopputuloksen hyvyyttä voida taata. Edellämainittujen hankaluuksien vuoksi onkin järkevämpää hakea suunnittelumuuttujien arvot optimaalisiksi käyttäen tietokoneella ajettavia optimointimenetelmiä.

Usein käytännön sovelluksissa kohdattavat optimointiongelmat ovat luonteeltaan haastavia. Yksittäinen simulaatioajo, joka kertoo kuinka hyvin tietty suunnittelumuuttujien arvoasetus toimii, saattaa kestää minuutteja, tunteja, tai jopa vuorokausia. Näin muodoin on selvää, että optimointiprosessiin kuluvien simulaatioajojen määrä tulisi pitää mahdollisimman pienenä. Usein ei riitä myöskään se, että järjestelmää optimoitaisiin suhteessa yhteen ainoaan tavoitteeseen (yksitavoiteoptimointi), vaan erilaisia tavoitteita saattaa olla useita (monitavoiteoptimointi), ja tyypillisesti nämä tavoitteet ovat ristiriitaisia keskenään, kuten esimerkiksi lopputuotteen hinta ja laatu. Tässä tapauksessa ratkaisut muodostavat ns. Pareto-optimaalisen joukon, jonka kaikki ratkaisut ovat matemaattisesti tarkasteltuna yhtä hyviä, ja siksi tarvitaankin inhimillinen päätöksentekijä valitsemaan ratkaisuista paras.

Tässä työssä käsitellään erilaisia tapoja ratkaista sekä yksi- että monitavoitteisia simulaatiopohjaisia optimointiongelmia tehokkaasti, jolloin aikaavieviä simulaatioajoja tarvitaan mahdollisimman vähän. Tämän työn puitteissa on kehitetty tehokkaat populaatiopohjaiset algoritmit sekä yksi- että monitavoitteiseen optimointiin, laskennan tarkkuuden säätämiseen perustuva tapa interaktiivisen monitavoiteoptimoinnin menetelmien tehostamiseksi, sekä tiedonlouhinta- ja klusterointimenetelmiin perustuva tapa inhimillisen päätöksentekijän kognitiivisen kuorman vähentämiseksi valittaessa lopullista ratkaisua moniulotteisesta Pareto-optimaalisesta joukosta. Kehitettyjen menetelmien suorituskyky näyttää sekä synteettisten testiongelmien että käytännön ongelmien (mm. polttomoottorin tehontuoton optimointi) ratkaisemisen valossa lupaavalta.

ERRATA

In publications [PIV] and [PV] we have used DE's control parameter CR for crossover probability in a different manner than for example in [110], where a higher values of CR actually mean a smaller probability for crossover (in crossover the element to the trial vector is taken from the target vector instead of the mutated vector). In publications [PIV] and [PIV], the higher value of CR means also a higher probability to take the element to the trial vector from the target vector instead of the mutated vector.

REFERENCES

[1] T. Aittokoski (2007): On Optimization of Simulation Based Design. Licentiate Thesis. Jyväskylä Licentiate Thesis in Computing 8. University of Jyväskylä.

[2] M. M. Ali, C. Khompatraporn and Z. B. Zabinsky (2005): A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems. Journal of Global Optimization 31, 635-672.

[3] M. M. Ali and C. Storey (1994): Modified Controlled Random Search Algorithms. International Journal of Computer Mathematics 54, 229-235.

[4] M. M. Ali, C. Storey and A. Törn (1997): Application of some Stochastic Global Optimization Algorithms to Practical Problems. Journal of Optimization Theory and Applications 95, 545-563.

[5] S. Äyrämö (2006): Knowledge Mining Using Robust Clustering. Doctoral Thesis. Jyväskylä Studies in Computing 63. University of Jyväskylä.

[6] J. Banks (1998): Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice. John Wiley & Sons, New York.

[7] M. Bartholomew-Biggs, S. Brown, B. Christianson and L. Dixon (2000): Automatic Differentiation of Algorithms. Journal of Computational and Applied Mathematics 124, 171-190.

[8] J.- F. M. Barthelemy and R. T. Haftka (1993): Approximation Concepts for Optimum Structural Design - A Review. Structural Optimization 5, 129-144.

[9] T. Bartz-Beielstein (2006): Experimental Research in Evolutionary Computation: The New Experimentalism. Springer-Verlag, Berlin.

[10] M. S. Bazaraa, H. D. Sherali and C.M. Shetty (1993): Nonlinear Programming: Theory and Algorithms, 2nd edition. John Wiley and Sons, New York.

[11] H. -G. Beyer and H. -P. Schwefel (2002): Evolution Strategies: A Comprehensive Introduction. Natural Computing 1(1), 3-52.

[12] P. K. Bergey and C. Ragsdale (2005): Modified Differential Evolution: A Greedy Random Strategy for Genetic Recombination. Omega 33, 255-265.

[13] L. T. Biegler (1989): Chemical Process Simulation. Chemical Engineering Progress 85(10), 50-61.

[14] T. Bäck (1996): Evolutionary Algorithms in Theory and Practice. Oxford University Press, Oxford.

[15] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. C. Resende and W. R. Stewart, Jr. (1995): Designing and Reporting on Computational Experiments with Heuristic Methods. Journal of Heuristics 1(1), 9-32.

[16] M. Björkman and K. Holmström (2001): Global Optimization of Costly Nonconvex Functions Using Radial Basis Functions. Optimization and Engineering 1, 373-397.

[17] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer (2006): Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Transactions on Evolutionary Computation, 10(6), 646-657.

[18] J. T. Buchanan (1997): A Naïve Approach for Solving MCDM Problems: the GUESS Method. Journal of the Operational Research Society 48, 202-206.

[19] M. D. Buhmann (2003): Radial Basis Functions. Cambridge University Press, Cambridge.

[20] L. Bull (1999): On Model-based Evolutionary Computation. Soft Computing 3, 76-82.

[21] Y. Cao (2008): Matlab Central File Exchange: Hypervolume Indicator. `http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=19651&objectType=file`

[22] V. Chankong and Y. Y. Haimes (1983): Multiobjective Decision Making Theory and Methodology. Elsevier Science Publishing, New York.

[23] A. Charnes and W. W. Cooper (1977): Goal Programming and Multiple Objective Optimization; Part 1. European Journal of Operational Research 1(1), 39-54.

[24] Z. Chen (2005): Finite Element Methods and Their Applications. Springer-Verlag, Berlin.

[25] C. A. Coello Coello and G. B. Lamont, Eds. (2004): Applications of Multi-Objective Evolutionary Algorithms. World Scientific Publishing Company, Singapore.

[26] C. A. Coello Coello, G. B. Lamont and D. A. Van Veldhuizen (2007): Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd Edition. Springer-Verlag, Berlin.

[27] C. Cortes and V. Vapnik (1995): Support Vector Networks. Machine Learning 20, 273-297.

[28] D. Craft (2008): Matlab Central File Exchange: Pareto Surface Navigator. `http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=13875&objectType=file`

[29] N. Cressie (1990): The Origins of Kriging. Mathematical Geology 22, 197-202.

[30] H. Crowder, R. S. Dembo and J. M. Mulvey (1979): On Reporting Computational Experiments with Mathematical Software. ACM Transactions on Mathematical Software 5(2), 193-203.

[31] I. Das and J. E. Dennis (1997): A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems. Structural Optimization 14(1), 63-69.

[32] K. Deb (2001): Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, New York.

[33] K. Deb (2003): Unveiling Innovative Design Principles by Means of Multiple Conflicting Objectives. Engineering Optimization 35(5),445-470.

[34] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan (2002): A Fast and Elitis Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions in Evolutionary Computation 6(2), 182-197.

[35] K. Deb and A. Srinivasan (2005): Innovization: Innovative Design Principles Through Optimization. KanGAL Report Number 2005007.

[36] K. Deb, L. Thiele, M. Laumanns and E. Zitzler (2002): Scalable Multiobjective Optimization Test Problems. In Proceedings of the 2002 Congress on Evolutionary Computation CEC2002, 825-830.

[37] J. E. Dennis and V. Torczon (1996): Managing Approximation Models in Optimization. In N. Alexandrov and M. Y. Hussaini, eds., Multidisciplinary Design Optimization: State of the Art, 330-347.

[38] L. Dixon and G. Szego (1978): The Global Optimization Problem: An Introduction. In Towards Global Optimization 2, 1-15. Amsterdam, Holland.

[39] M. Emmerich, N. Beume and B. Naujoks (2005): An EMO Algorithm using the Hypervolume Measure as Selection Criterion. In Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization, 62-76.

[40] M. Emmerich, K. Giannakoglou and B. Naujoks (2006): Single- and Multiobjective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels. IEEE Transactions on Evolutionary Computation 10(4), 421-439.

[41] M. Emmerich, A. Giotis, M. Özdenir, T. Bäck and K. Giannakoglou (2002): Metamodel-assisted Evolution Strategies. In Parallel Problem Solving from Nature, number 2439 in Lecture Notes in Computer Science, 371-380. Springer-Verlag, Berlin.

[42] P. Eskelinen, K. Miettinen, K. Klamroth and J. Hakanen (2008): Pareto Navigator for Interactive Nonlinear Multiobjective Optimization. OR Spectrum, to appear.

[43] B. Everitt, S. Landau and M. Leese (2001): Cluster Analysis, 4th Edition. Oxford University Press, Oxford.

[44] P. C. Fishburn (1974): Lexicographic Orders, Utilities and Decision Rules: A Survey. Management Science 20(11), 1442-1471.

[45] M. Fleischer (2002): The Measure of Pareto Optima - Applications to Multi-Objective Metaheuristics. Lecture Notes in Computer Science 2632, 519-533. Springer-Verlag, Berlin.

[46] C. A. Floudas and P. M. Pardalos (1991): A Collection of Test Problems for Constrained Global Optimization Algorithms. Springer-Verlag, Berlin.

[47] C. A. Floudas, P. M. Pardalos, C. S. Adjiman, W. R. Esposito, J. L. Klepeis, Z. H. Gumus, C. A. Meyer and C. A. Schweiger (1999): Handbook of Test Problems in Local and Global Optimization. Kluwer Academic Publishers, Boston.

[48] S. Gass and T. Saaty (1955): The Computational Algorithm for the Parametric Objective Function. Naval Research Logistics Quarterly 2, 39-45.

[49] K. C. Giannakoglou (2002): Design of Optimal Aerodynamic Shapes using Stochastic Optimization Methods and Computational Intelligence. Progress in Aerospace Sciences 38, 43-76.

[50] K. C. Giannakoglou, M. K. Karakasis and I. C. Kampolis (2006): Evolutionary Algorithms with Surrogate Modeling for Computationally Expensive Optimization Problems. In Proceedings of ERCOFTAC 2006 Design Optimization International Conference, April 5-7 2006, Gran Canaria, Spain.

[51] A. P. Giotis, M. Emmerich, B. Naujoks, K. C. Giannakoglo and T. Bäck (2002): Low-cost Stochastic Optimization for Engineering Applications. In Proceedings of International Conference on Evolutionary Methods for Design, Optimization and Control, Barcelona, Spain.

[52] P. E. Gill, W. Murray, M. H. Wright (1981): Practical Optimization. Academic Press, London.

[53] F. Glover and G. A. Kochenberger (2003): Handbook of Metaheuristics. Kluwer Academic Publishers, Boston.

[54] T. J. Gogg and J. R. A. Mott (1996): Improve Quality & Productivity with Simulation, 3rd Edition. JMI Consulting Group, Palos Verdes.

[55] D. E. Goldberg (1989): Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, New York.

[56] A. Gosavi (2003): Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning. Kluwer Academic Publishers, Boston.

[57] H.-M. Gutmann (2001): A Radial Basis Function Method for Global Optimization. Journal of Global Optimization 19, 201-227.

[58] T. Hanne (1999): On the Convergence of Multiobjective Evolutionary Algorithms. European Journal of Operational Research 117(3), 553-564.

[59] S. Haykin (1998): Neural Networks: A Comprehensive Foundation, 2nd Ed. Prentice Hall, New Jersey.

[60] A. Hedar (2008): Test Functions for Unconstrained Global Optimization. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm

[61] K. E. Hillstrom (1977): A Simulation Test Approach to the Evaluation of Nonlinear Optimization Algorithms. ACM Transactions on Mathematical Software 3(4), 305-315.

[62] R. Horst and P. M. Pardalos, Eds. (1995): Handbook of Global Optimization. Kluwer Academic Publishers, Boston.

[63] S. Huband, L. Barone, L. While and P. Hingston (2005): A Scalable Multiobjective Test Problem Toolkit. In Proceedings of Evolutionary Multi-Criterion Optimization: 3rd International Conference, 280-294.

[64] S. Huband, P. Hingston, L. Barone and L. While (2006): A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. IEEE Transactions on Evolutionary Computation 10(5), 477-506.

[65] E. J. Hughes (2005): Evolutionary Many-Objective Optimisation: Many Once or One Many? In Proceedings of the 2005 Congress on Evolutionary Computation (CEC-205), 222-227.

[66] C.-L. Hwang and A. S. M. Masud (1979): Multiple Objective Decision Making - Methods and Applications. Springer-Verlag, Berlin.

[67] J. Jahn (2004): Vector Optimization: Theory, Applications, and Extensions. Springer-Verlag, Berlin.

[68] Y. Jin (2005): A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. Soft Computing 9(1), 3-12.

[69] Y. Jin, M. Olhofer and B. Sendhoff (2001): Managing Approximate Models in Evolutionary Aerodynamic Design Optimization. In Proceedings of IEEE Congress on Evolutionary Computation 1, 592-599.

[70] D. R. Jones, M. Schonlau and W. J. Welch (1998): Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization 13(4), 455-492.

[71] E. M. Kasprzak and K. E. Lewis (2000): An Approach to Facilitate Decision Tradeoffs in Pareto Solution Sets. Journal of Engineering Valuation and Cost Analysis 3(1), 173-187.

[72] L. Kaufman and P. J. Rousseeuw (2005): Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, New York.

[73] R. L. Keeney and H. Raiffa (1976): Decisions with Multiple Objectives: Preferences and Value Tradeoffs. John Wiley & Sons, New York.

[74] J. Knowles (2006): ParEGO: A Hybrid Algorithm with On-line Landscape Approximation for Expensive Multiobjective Optimization Problems. IEEE Transactions on Evolutionary Computation 10(1), 50-66.

[75] J. Knowles and D. Corne (2000): Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. Evolutionary Computation 8, 149-172.

[76] J. Knowles and H. Nakayama (2008): Meta-modeling in Multi-objective Optimization. In Multi-objective Optimization - Interactive and Evolutionary Approaches. Springer-Verlag, Berlin. Forthcoming.

[77] P. Korhonen, H. Moskowitz and J. Wallenius (1992): Multiple Criteria Decision Support - A Review. European Journal of Operational Research 63(3), 361-375.

[78] S. Kukkonen and K. Deb (2006): Improved Pruning of Non-Dominated Solutions Based on Crowding Distance for Bi-Objective Problems. In Proceedings of 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada July 16-21.

[79] S. Kukkonen and K. Deb (2006): A Fast and Effective Method for Pruning of Non-dominated Solutions in Many-Objective Problems. Parallel Problem Solving from Nature - PPSN IX, 553-562. Springer-Verlag, Berlin.

[80] S. Kukkonen and J. Lampinen (2005): GDE3: the Third Evolution Step of Generalized Differential Evolution. In Proceedings of IEEE Congress on Evolutionary Computation, 443-450, Edinburgh, Scotland.

[81] S. Kulturel-Konak, D. W. Coit and F. Baheranwala (2005): Pruned Pareto-optimal Sets for the System Redundancy Allocation Problem Based on Multiple Prioritized Objectives. Rutgers University ISE working paper 05-009.

[82] H. J. Kushner (1964): A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. Journal of Basic Engineering 86, 97-106.

[83] M. Laumans, L. Thiele, K. Deb and E. Zitzler (2002): Combining Convergence and Diversity in Evolutionary Multi-Objective Optimization. Evolutionary Computation 10(3), 263-282.

[84] J. van Leersum (1998): A Numerical Model of a High Performance Two-stroke Engine. Applied Numerical Mathematics 27, 83-108.

[85] D. Lim, Y. Jin, Y. S. Ong and B. Sendhoff (2008): Generalizing Surrogate-assisted Evolutionary Computation. IEEE Transactions on Evolutionary Computation, in press.

[86] G.- R. Liu and S. S. Quek (2003): The Finite Element Method: A Practical Course. Butterworth-Heinemann, Oxford.

[87] R. T. Marler and J. S. Arora (2004): Survey of Multi-objective Optimization Methods for Engineering. Structural and Multidisciplinary Optimization 26, 369-395.

[88] C. A. Mattson, A. A. Mullur and A. Messac (2004): Smart Pareto Filter: Obtaining a Minimal Representation of Multiobjective Design Space. Engineering Optimization 36(6), 721-740.

[89] A. Messac and A. A. Mullur (2008): A Computationally Efficient Metamodeling Approach for Expensive Multiobjective Optimization. Optimization and Engineering 9(1), 37-67.

[90] K. Miettinen (1999): Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston.

[91] K. Miettinen (2003): Graphical Illustration of Pareto Optimal Solutions. In Multi-Objective Programming and Goal Programming: Theory and Applications, T. Tanino, T. Tanaka, M. Inuiguchi (Eds.), 197-202. Springer-Verlag, Berlin.

[92] K. Miettinen (2006): IND-NIMBUS for Demanding Interactive Multiobjective Optimization. In Multiple Criteria Decision Making '05, Ed. by T. Trzaskalik, The Karol Adamiecki University of Economics in Katowice, Katowice, 137-150, 2006.

[93] K. Miettinen and M. M. Mäkelä (1995): Interactive Bundle-Based Method for Non-differentiable Multiobjective Optimization: NIMBUS. Optimization 34(3), 231-246.

[94] K. Miettinen and M. M. Mäkelä (1999): Comparative Evaluation of Some Interactive Reference Point-based Methods for Multi-Objective Optimisation. Journal of the Operational Research Society 50, 949-959.

[95] K. Miettinen and M. M. Mäkelä (2000): Interactive Multiobjective Optimization System WWW-NIMBUS on the Internet. Computers & Operations Research 27, 709-723.

[96] K. Miettinen and M. M. Mäkelä (2002): On Scalarizing Functions in Multiobjective Optimization. OR Spectrum 24, 193-213.

[97] K. Miettinen and M. M. Mäkelä (2006): Synchronous Approach in Interactive Multiobjective Optimization. European Journal of Operational Research 170, 909-922.

[98] J. Mockus (1994): Application of Bayesian Approach to Numerical Methods of Global and Stochastic Optimization. Journal of Global Optimization 4, 347-365.

[99] M. Monz, K. H. Küfer, T. R. Bortfeld and C. Thieke (2008): Pareto Navigation - Algorithmic Formulation of Interactive Multi-criteria IMRT Planning. Physics in Medicine and Biology 53, 985-998.

[100] H. Nakayama, M. Arakawa and R. Sasaki (2002): Simulation-Based Optimization Using Computational Intelligence. Optimization and Engineering 3, 201-214.

[101] H. Nakayama and Y. Sawaragi (1984): Satisficing Trade-Off Method for Multiobjective Programming. In M. Grauer and A. P. Wierzbicki, Eds. Interactive Decision Analysis, 113-122. Springer-Verlag, Berlin.

[102] J. A. Nelder and R. Mead (1965): A Simplex Method for Function Minimization. Computer Journal 7, 308-313.

[103] V. Ojalehto, K. Miettinen and M. M. Mäkelä (2007): Interactive Software for Multiobjective Optimization: IND-NIMBUS. WSEAS Transactions on Computers 6(1), 87-94.

[104] G. C. Onwubolu and B. V. Babu (Eds.) (2004): New Optimization Techniques in Engineering. Springer-Verlag, Berlin.

[105] P. M. Pardalos and H. E. Romeijn, Eds. (2002): Handbook of Global Optimization Volume 2. Springer-Verlag, Berlin.

[106] J. D. Pinter (1996): Global Optimization in Action - Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications. Kluwer Academic Publishers, Boston.

[107] R. C. Purshouse and P. J. Fleming (2003): Evolutionary Many-objective Optimisation: An Exploratory Analysis. In Proceedings of The 2003 Congress on Evolutionary Computation (CEC 2003) 3, 2066-2073, Canberra, Australia, 8-12 December 2003.

[108] W. L. Price (1977): A Controlled Random Search Procedure for Global Optimization. Computer Journal 20(4), 367-370.

[109] W. L. Price (1983): Global Optimization by Controlled Random Search. Journal of Optimization Theory and Applications 40, 333-348.

[110] K. V. Price, R. M. Storn and J. A. Lampinen (2005): Differential Evolution - A Practical Approach to Global Optimization. Springer-Verlag, Berlin.

[111] A. Ratle (1998): Accelerating the Convergence of Evolutionary Algorithms by Fitness Landscape Approximation. In A. Eiben, T. Bäck, M. Schoenauer and H.-P. Schwefel, eds., Parallel Problem Solving from Nature, volume V, 87-96.

[112] A. Ratle (1999): Optimal Sampling Strategies for Learning a Fitness Model. In Proceedings of 1999 Congress on Evolutionary Computation 3, 2078-2085, Washington D.C., July 1999.

[113] C. R. Raquel and P. C. Naval Jr. (2005): An Effective Use of Crowding Distance in Multiobjective Particle Swarm Optimization. In Proceedings of the Genetic and Evolutionary Computation (GECCO 2005), Washington DC, USA, 2005, 257-264.

[114] T. Robic and B. Filipic (2005): DEMO: Differential Evolution for Multiobjective Optimization. In Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005), Guanajuato, Mexico, 2005, 520-533.

[115] G. Rudolph (1996): Convergence of Evolutionary Algorithms in General Search Spaces. In Proceedings of the Third IEEE Conference on Evolutionary Computation, 50-54.

[116] G. Rudolph (1998): Evolutionary Search for Minimal Elements in Partially Ordered Finite Sets. In Proceedings of the 7th Annual Conference on Evolutionary Programming, 345-353. Springer-Verlag, Berlin.

[117] G. Rudolph (1999): Evolutionary Search under Partially Ordered Sets. Technical Report CI-67/99. University of Dortmund.

[118] G. Rudolph and A. Agapie (2000): Convergence Properties of Some Multiobjective Evolutionary Algorithms. In Proceedings of IEEE Congress on Evolutionary Computation, 1010-1016.

[119] S. Ruuska and T. Aittokoski (2008): The Effect of Trial Point Generation Schemes on the Efficiency of Population-Based Global Optimization Algorithms. In Proceedings of International Conference on Engineering Optimization, Rio de Janeiro, Brazil, 2008.

[120] S. Ruzika and M. M. Wiecek (2005): Survey Paper - Approximation Methods in Multiobjective Programming. Journal of Optimization Theory and Applications 126(3), 473-501.

[121] J. Sacks, W. Welch, T. Mitchell and H. Wynn (1989): Design and Analysis of Computer Experiments (with Discussion). Statistical Science 4, 409-435.

[122] R. Salomon (1996): Reevaluating Genetic Algorithm Performance Under Coordinate Rotation of Benchmark Functions. BioSystems, 39(3), 263-278.

[123] M. J. Sasena (2002): Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations. Doctoral dissertation. University of Michigan.

[124] P. Siarry and Z. Michalewicz, Eds. (2007): Advances in Metaheuristics for Hard Optimization. Springer-Verlag, Berlin.

[125] B. D. Sivazlian and L. E. Stanfel (1975): Optimization Techniques in Operations Research. Prentice-Hall, New Jersey.

[126] R. E. Steuer (1986): Multiple Criteria Optimization: Theory, Computation, and Application. John Wiley & Sons, New York.

[127] R. Storn (2008): Differential Evolution Homepage.
`http://www.icsi.berkeley.edu/~storn/code.html`

[128] R. Storn and K. Price (1997): Differential evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11, 341-359.

[129] H. A. Taboada and D. W. Coit (2005): Post-Pareto Optimality Analysis to Efficiently Identify Promising Solutions for Multi-Objective Problems. Rutgers University ISE working paper 05-015.

[130] H. A. Taboada and D. W. Coit (2006): Data Mining Techniques to Facilitate the Analysis of the Pareto-Optimal Set for Multiple Objective Problems. Rutgers University ISE working paper 06-001.

[131] H. A. Taboada and D. W. Coit (2007): Data Clustering of Solutions for Multiple Objective System Reliability Optimization Problems. Quality Technology and Quantitative Management 4(2), 191-210.

[132] A. Törn, M. M. Ali and S. Viitanen (1999): Stochastic Global Optimization : Problem Classes and Solution Techniques. Journal of Global Optimization 14(4), 437-447.

[133] A. Törn and A. Zilinskas (1989): Global Optimization. Springer-Verlag, Berlin.

[134] H. L. Trinkaus and T. Hanne (2005): knowCube: A visual and Interactive Support for Multicriteria Decision Making. Computers & Operations Research 32, 1289-1309.

[135] V. N. Vapnik (1998) Statistical Learning Theory. John Wiley & Sons, New York.

[136] D. V. Veldhuizen (1999): Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations, Ph. D. Thesis, Dayton, OH: Air Force Institute of Technology, Technical Reports No. AFIT/DS/ENG/99-01.

[137] V. Venkat, S. H. Jacobson and J. A. Stori (2004): A Post-Optimality Analysis Algorithm for Multi-Objective Optimization. Computational Optimization and Applications 28, 357-372.

[138] M. A. Villalobos-Arias, G. T. Pulido and C. A. Coello Coello (2005): A Proposal to Use Stripes to Maintain Diversity in a Multi-objective Particle Swarm Optimizer. In Proceedings of Swarm Intelligence Symposium SIS2005, 22-29.

[139] J. B. Vosa, A. Rizzib, D. Darracqc and E. H. Hirschel (2002): Navier-Stokes Solvers in European Aircraft Design. Progress in Aerospace Sciences 38, 601-697.

[140] I. Voutchkov and A. J. Keane (2006): Multiobjective Optimization Using Surrogates. In Proceedings of the 7th International Conference on Adaptive Computing in Design and Manufacture, 167-175.

[141] T. Weise (2008): Global Optimization Algorithms - Theory and Application. eBook, `www.it-weise.de/projects/book.pdf`.

[142] W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell and M. D. Morris (1992): Screening, Predicting, and Computer Experiments. Technometrics 34, 15-25.

[143] L. While, P. Hingston, L. Barone and S. Huband (2006): A Faster Algorithm for Calculating Hypervolume. IEEE Transactions on Evolutionary Computation 10(1), 29-38.

[144] A. P. Wierzbicki (1982): A Mathematical Basis for Satisficing Decision Making. Mathematical Modelling 3(25), 391-405.

[145] A. P. Wierzbicki (1999): Reference Point Approaches. In Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory, and Applications. T. Gal, T. J. Stewart and T. Hanne (Eds.). Kluwer Academic Publishers, Boston.

[146] D. H. Wolpert and W. G. Macready (1997): No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation, 1, 67-82.

[147] D. Zaharie (2003): Multi-objective Optimization with Adaptive Pareto Differential Evolution. In Proceedings of Symposium on Intelligent Systems and Applications (SIA 2003), Iasi, Romania.

[148] A. Zilinskas (1985): Axiomatic Characterization of a Global Optimization Algorithm and Investigation of its Search Strategy. Operations Research Letters 4(1), 35-39.

[149] E. Zitzler, D. Brockhoff and L. Thiele (2007): The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. In EMO 2007, LNCS 4403, 862-876. Springer-Verlag, Berlin.

[150] E. Zitzler, K. Deb and L. Thiele (2000): Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173-195.

[151] E. Zitzler, J. Knowles and L. Thiele (2008): Quality Assessment of Pareto Set Approximations. In Multiobjective Optimization - Interactive and Evolutionary Approaches, J. Branke, K Deb, K. Miettinen and R. Slowinski (Eds.), LNCS 5252, 373-404. Springer-Verlag, Berlin.

[152] E. Zitzler and S. Künzli (2004): Indicator-based Selection in Multiobjective Search. In Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, 832-842.

[153] E. Zitzler, M. Laumanns and L. Thiele (2001): SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Swiss Federal Institute of Technology, technical report TIK-Report 103.

[154] E. Zitzler and L. Thiele (1998): Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In Conference on Parallel Problem Solving from Nature (PPSN V), 292-301, Amsterdam.

[155] E. Zitzler, L. Thiele and J. Bader (2008): On Set-Based Multiobjective Optimization. Swiss Federal Institute of Technology, technical report TIK-Report 300.

[156] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca and V. G. da Fonseca (2003): Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation 7(2), 117-132.

ORIGINAL PAPERS

PI

COST EFFECTIVE SIMULATION-BASED MULTIOBJECTIVE OPTIMIZATION

by

Timo Aittokoski and Kaisa Miettinen 2008

Engineering Optimization 40(7), 593-612

https://doi.org/10.1080/03052150801914429

PII

DECREASING COMPUTATIONAL COST OF SIMULATION BASED INTERACTIVE MULTIOBJECTIVE OPTIMIZATION WITH ADJUSTABLE SOLUTION ACCURACY

by

Timo Aittokoski and Kaisa Miettinen 2008

Reports of the Department of Mathematical Information Technology, Series B.
Scientific Computing, No. B 19/2008, University of Jyväskylä

http://urn.fi/URN:ISBN:978-951-39-9035-0

PIII

CLUSTERING AIDED APPROACH FOR DECISION MAKING IN COMPUTATIONALLY EXPENSIVE MULTIOBJECTIVE OPTIMIZATION

by

Timo Aittokoski, Sami Äyrämö and Kaisa Miettinen

Optimization Methods and Software, to appear

https://doi.org/10.1080/10556780802525331

PIV

EFFICIENT EVOLUTIONARY METHOD TO APPROXIMATE THE PARETO OPTIMAL SET IN MULTIOBJECTIVE OPTIMIZATION

by

Timo Aittokoski and Kaisa Miettinen 2008

Proceedings of International Conference on Engineering Optimization EngOpt 2008, Rio de Janeiro, Brazil, June 1-5

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.576.9001

PV

EFFICIENT EVOLUTIONARY OPTIMIZATION ALGORITHM: FILTERED DIFFERENTIAL EVOLUTION

by

Timo Aittokoski 2008

Reports of the Department of Mathematical Information Technology, Series B.
Scientific Computing, No. B 20/2008, University of Jyväskylä

http://urn.fi/URN:ISBN:978-951-39-9036-7