Multi-objective Robust Optimization Algorithms for Improving Energy Consumption and Thermal Comfort of Buildings

M.Sc. Thesis
by
Robert Marijt

Faculty of Computer Science at the University of Leiden, July, 2009

Supervisors

dr.rer.nat. M.T.M. Emmerich, University Of Leiden, Faculty of Computer Science

dr. C.J. Hopfe, Technical University of Eindhoven, Faculty of Architecture, Building & Planning; Unit Building Physics and Systems

prof.dr.ir J. Hensen, Technical University of Eindhoven, Faculty of Architecture, Building & Planning; Unit Building Physics and Systems

Acknowledgements

This research was greatly supported by Vabi Software BV.

Contents

List of Figures

List of Tables

Abstract

Energy efficiency and thermal comfort are of concern in building design. Due to the fact that one third of national total annual energy consumption is consumed in buildings, it is estimated that substantial energy savings can be achieved through careful planning for energy efficiency. Building performance simulation (BPS) is a powerful tool to predict and analyze the dynamic behavior of performance indicators such as energy consumption and comfort among others. Previous work has shown that the use of BPS is mostly limited to code compliance checking in the detailed design. Also it has been shown that single optimization does support the detailed design stage. In this thesis we want to go a step further. An attempt will be made to interface a robust multi-objective optimization algorithm with a BPS tool. Subjects as uncertainties in building simulation variables and metamodels to support optimization will be topics of discussion. This thesis presents results and experiences on extending an existing BPS tool with a capability for robust multi-objective metamodel supported optimization. The focus of this work is on algorithmic design and results for the optimization of energy consumption and thermal comfort in office buildings.

Summary

Multi-objective Robust Optimization Algorithms for Improving Energy Consumption and Thermal Comfort of Building

Energy efficiency and thermal comfort are of concern in building design. Due to the fact that one third of national total annual energy consumption is consumed in buildings, it is estimated that substantial energy savings can be achieved through careful planning for energy efficiency.
Building performance simulation (BPS) is a powerful tool to predict and analyze the dynamic behavior of performance indicators such as energy consumption and comfort among others. Previous work has shown that the use of BPS is mostly limited to code compliance checking in the detailed design. Also it has been shown that single optimization does support the detailed design stage. In this thesis we want to go a step further. An attempt will be made to interface a robust multi-objective optimization algorithm with a BPS tool. A description is given for the optimization and uncertainty variables. The objectives are explained and additional information is stated regarding the building on which the case is based on.
Different aspects of multi-objective optimization, like the Pareto front and the dominance relation between solutions will be discussed. Also from a number of well-known evolutionary multi-objective algorithms (EMOA) a description is given. It is substantiated why the S-metric selection EMOA (SMS-EMOA) is chosen to be the base for the robust algorithm.
To deal with uncertainty a SMS-EMOA is adjusted to support robust optimization. Uncertainty may exist in different parts of a system where the objective function is based on. If robust optimization is applied, this can result in possible shifts of parts of the Pareto front or the complete Pareto front.
Because of the computationally time consuming objective function a metamodel is applied to partially replace objective function calls with estimated interpolated values from the metamodel. It is explained how an important parameter in the model is estimated to get the model calibrated. Locally and globally metamodel supported optimizations are compared. A metamodel called Kriging is explained and compared to radial basis functions networks to show the similarity between the two techniques.
The algorithm and the parameters involved are described. Different outlines show pseudocode of parts of the algorithm. A description is given of the development of the algorithm, especially about the estimation of the parameters in the metamodel.
The conclusion of this thesis is that it is possible to perform a robust optimization with a robust algorithm supported by a metamodel. Furthermore, different figures and statistics support this conclusion.

1 Introduction

For the analysis and prediction of the dynamic behavior of building performance indicators such as energy consumption and thermal comfort, building performance simulation (BPS) is a powerful tool. Previous work has shown that the use of BPS is mostly limited to code compliance checking in the detailed design [Hop09]. The use of multi-disciplinary optimization [PH02] in building design is still in its early stages [HSHB06]. Disciplinary specific optimization activities are known by Michalek et al. [MCP02] in architectural design and for instance by Wright et al. [WZAB04] in mechanical engineering. In Emmerich et al. [EHM$^+$08] an introduction can be found of a spectrum of optimization and design space exploration techniques that can be used by the engineer to find the optimal solution for a building regarding the performance with respect to the often conflicting objectives. This article mainly focuses on multi-objective robust optimization of a two-objective problem.

Simple straightforward optimization in BPS is an interesting task, but does not fully represent building characteristics as they appear in reality. Almost all variables that participate in the optimization have a degree of uncertainty in it, caused by material characteristics and external and internal condition changes. These uncertainties may cause a shift of the global or local Pareto front in a multi-objective optimization.

In this thesis an algorithm will be built that optimizes two objectives and deals with uncertainties in variables. The different aspects of the algorithm and the techniques used will be explained and problems that show up will be tackled. A description will be given of the evolution of the algorithm and its performance.

The next chapter contains a problem statement and additional information regarding the problem. The chapters following describe different techniques used in de algorithm. Hereafter two chapters give an outline of the algorithm and discuss the results. In the final chapter a conclusion and some remarks on promising directions of future research will be given.

2 Problem description and hypothesis

2.1 Building 'Het Bouwhuis'

The building is based on a building located in Zoetermeer in the Netherlands called "Het bouwhuis". It is the headquarter of Bouwend Nederland, the Dutch organisation of construction companies. The building is a perfect case study, because it combines flexibility and function. It is an office building with eleven floors in a T-shaped plan. There are two levels of underground parking, flexible office concepts, conference facilities and a restaurant with roof garden among others. There is a conventional heating system and mechanical cooling; the building is conditioned by an all air conditioning system with constant air volume (CAV) consisting of an air handling unit, supply and return fans, ducts and control units. Heating is provided by electricity driven radiators inside the room and an electric heater in the air-handling unit (AHU). The system is regulated on the air temperature; during the office hours (8-20h, 5 days per week) and on standby the rest of the time (0-24h, 7 days per week). The AHU keeps the supply air temperature at 20°C when the incoming outside air temperature is 16°C up to 28°C when the outside air temperature goes up to 40°C. The ventilation system provides fresh air with a supply fan (1000 m^3/h) and exhausts the air by an exhaust fan (1000 m^3/h). Air change rate is 0.5 per hour. There is no night cooling. One zone is used in the optimization, as it is sufficient to show some preliminary results. The shaded area in figure 2.1.1 represents the zone.

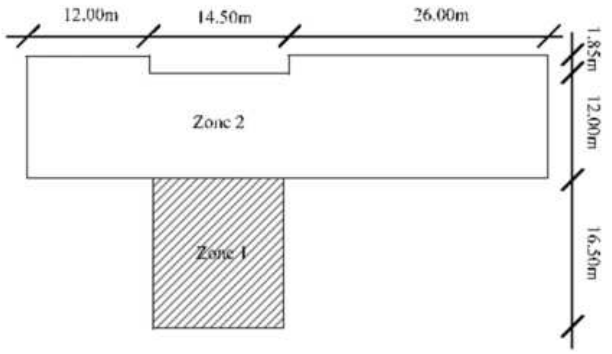


Figure 2.1.1: Building "Het bouwhuis".

2.2 Building performance simulation tool

Building performance simulation uses computer-based models to assess building performance aspects such as energy consumption and thermal comfort. VA114 forms part of the uniform environment. The uniform environment is a software tool box that allows shifting model files between several tools for different types of analysis, including heat loss and heat gain calculation. It is a simulation tool that is well-known and widely used in The Netherlands. This tool is developed by Vabi Software BV and dedicated to the later phases of the design process. VA114 is an engine dedicated to assess annual heating and cooling demand and the thermal behavior of a building. Thermal behavior is measured with a count of underheating and overheating hours,

which is discussed further in the objective description. The objective function is embedded within the BPS tool and can be seen as a black box. For extra information about VA114 and the objective function the reader is referred to [VAB09].

2.3 Optimization variables

The variables mentioned have been studied by Hopfe in an uncertainty and sensitivity analysis [Hop09], [Jia07] and [HHPW07]. The thesis comes back to this in the result section. The variable *geometry* extends the room at one sight. The room varies between 160-240 m^2. The variable *height* defines the height of the window. The unit for this variable is meters. The variables *persons*, *equipment* and *lighting* represent internal heat gains. The ranges are respectively 6-25 Wm^{-2}, 6-35 Wm^{-2} and 6-20 Wm^{-2}. Examples of internal heat gains are the number of people who work in a room, the equipment used (such as computers) and the radiation of light.

2.4 Uncertainty variables

Uncertainty analysis (UA) specifies the uncertainty in model prediction due to the imprecise knowledge of input variables. Uncertainties do arise from many different sources and can be divided into three groups caused by different parameters: physical, design, and scenario uncertainties. The uncertainty variables are selected out of a total of seventy-seven variables. In the result section is explained on which basis these variables have been chosen. Physical uncertainty variables are *Twall3*, *Cfloor4* and *Croof2*, *infiltration* is a scenario uncertainty and *glasswindow* is a design uncertainty.

Twall3 stands for Thickness wall layer 3. The unit is meters. *Cfloor4* stands for conductivity floor layer 4. The unit is WmK^{-1}. *Croof2* stands for conductivity roof layer 2. The unit is WmK^{-1}. *Infiltration* stands for air exchange rate per hour in the building. The unit is m^3h^{-1}. These four variables have gaussian distributed uncertainties. Their distributions can be found in Table 2.4.1. *Glasswindow* stands for single or double glass and can be set to the values zero or one respectively.

Variable	Mean	Standard deviation
infiltration	0.5	0.17
Cfloor4	0.025	0.00875
Croof2	0.5	0.25
Twall3	0.2	0.02

Table 2.4.1: Uncertainty variables with gaussian distributions.

2.5 Objectives description

The information regarding the objectives is purely to inform the reader about the background of the problem. For the optimization tool itself this information is not visible. All technical details about the calculations and applied models are hidden by the VA114 tool and therefore the objective function can be seen as a black box.

The first objective is thermal comfort. To analyze thermal comfort a Dutch criterion, called GTO criterion, published by the Rijksgebouwendienst in 1991 [ISSO 2004]

is applied. The weighted overheating or underheating hours (Dutch: Weeguren or GTO) is based on the Fanger Model. In this criterion the extent in which a predicted mean vote (PMV) of +0.5 is exceeded is expressed by a factor *WF* (weegfactor). The predicted mean vote is a modeled average perception of the indoor climate in buildings valid for a large group of people and is with that a tool for predicting the level of comfort of people in buildings.

As long as the number of overheating hours stays below 150h per year the conditions are in range. The same holds for the underheating hours. Objective function values *WOH+* and *WOH-* stand respectively for weighted overheating and weighted underheating. The unit for both values is *h*.

The second objective is about the energy consumption in the building. A distinct is made between annual cooling and annual heating. The unit for both these values is kWh.

To create a two-dimensional problem of the four objective function values *WOH+* is combined with *WOH-* and annual cooling is combined with annual heating. The first objective is composed of the sum of *WOH+* and *WOH-* and represents the thermal comfort in a building. The second objective is composed of the sum of the absolute value of annual cooling and annual heating and represent the energy consumption in a building.

2.6 Problem statement and hypothesis

The objective in this thesis is to build a robust algorithm which finds a series of solutions for a two-objective minimization problem within a defined search space and within time constraints and where the objective function is a building performance simulation tool which acts as a black box.

The following line describes the problem statement: *Find an algorithm that solves a two-objective robust optimization problem where the objective function is based on the building performance simulation tool:* The Hypothesis regarding the problem statement sounds: *In this thesis it hypothesized that a multi-objective evolutionary algorithm with metamodel support will solve this type of problems.*

The simulation tool has a runtime of approximately fourteen seconds. The robust optimization method chosen in this thesis demands a great number of calls to the objective function. Each round of the Monte Carlo algorithm for measuring the solution quality and robustness for a single solution will produce two hundred and one calls to the objective function. One call to the objective function takes approximately fourteen seconds. A total of four hundred rounds will be executed per optimization. The required time to produce the results of one optimization:

$$O_s = 201 * 400 * 14 = 1125600 \,\text{seconds} \tag{2.6.1}$$

$$O_d = \frac{\frac{1125600}{3600}}{24} \approx 13 \,\text{days} \tag{2.6.2}$$

Equations 2.6.1 and 2.6.2 show the runtime of a single optimization in seconds O_s and days O_d. It is obvious that the optimization is a time consuming task. To obtain

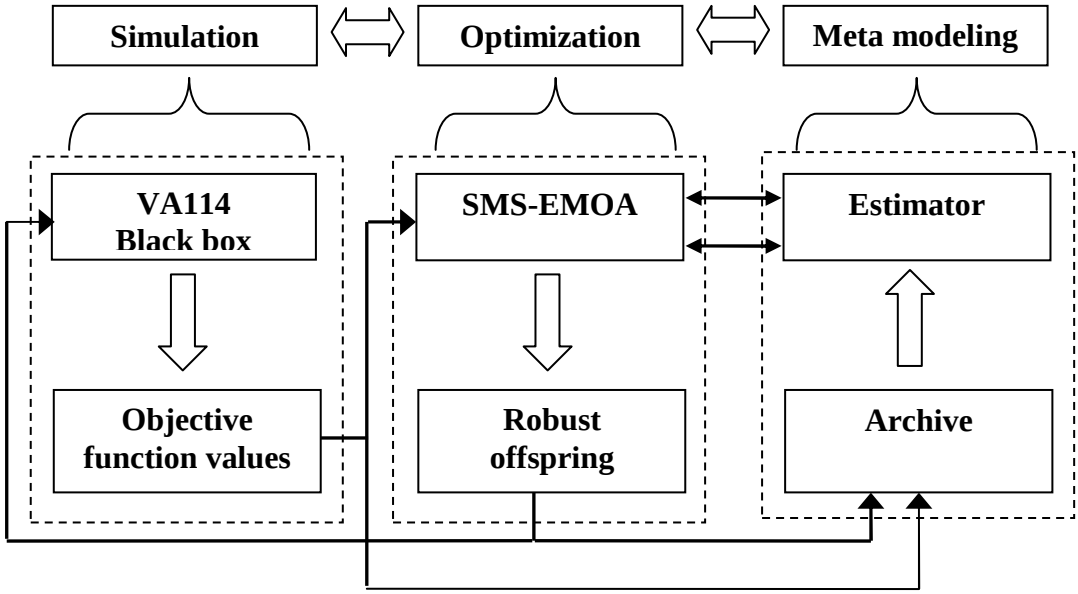


Figure 2.6.2: System overview.

reliable results the optimization is executed multiple times. Taken into account the thirteen days for one optimization, a logical next step would be an attempt to decrease the runtime.

There is more than one solution to solve this problem. If the money budget as well as the time budget is unlimited one can buy enough computers and do the optimizations simultaneously with parallel computing. The budget for the experiments in this thesis were limited to one or two computers and one night of runtime. Therefore a solution must be found in another direction. A model that approximates the expensive objective function can take care of the majority of the calls to this function. Figure 2.6.2 presents a global overview on how the system will work.

3 Single objective optimization

In an earlier research different single objective optimization techniques are examined. The results of this research can be found in [EHM+08]. The tool that was used in [EHM+08] is not able to deal with variable uncertainty which makes it necessary to switch to the earlier mentioned tool VA114. Besides, the test building has also been changed. This makes it difficult to compare the single objective optimization results with the new obtained multi-objective optimization results.

A naïve approach to solve the multi-objective problem would be to split up the problem in multiple single objective problems. In most of these attempts the different objectives get weight factors and are incorporated in a single weighted objective function. A disadvantage of this method is that the weights have to be set a priori which influences the optimization and the outcoming solutions beforehand. If objectives are in conflict with each other, this method can lead to a meaningless objective function. Another big disadvantage is that in case of concave Pareto fronts only extremal solutions optimizing one of the objectives can be obtained. However, it would be interesting to compare these type of algorithms against the multi-objective ones, but this would be beyond the scope of this thesis.

4 Multi-objective optimization

4.1 Introduction

In multi-objective optimization there are two or more objectives which need to be optimized. When one objective is improving and the other objective reacts in the same way by also showing improvement the problem is fairly simple. But in general this is not the case. For instance if a factory needs to produce, with a cost as low as possible, products of a quality as high as possible. These two goals are clearly the opposites of each other. The theoretically optimal solution seems to be high quality without any costs at all. This is of course impossible if one runs a legal business. What we like to obtain from the optimization is a series of solutions where the two boundary solutions are low quality, low production cost and high quality, high production cost. The other solutions will be between these two boundary solutions with none of them being a solution that can be improved in both objectives at the same time still. With this information trade-offs can be discovered between the objectives and compromise solutions can be obtained. The risk in this particular example may be the fact that cheap products not only end up to be from a low quality but also indirect affect the environment in a disadvantageous way. Therefore environment-friendliness inevitably is a third objective in this problem. Which solution eventually is the best depends on the demands and standards of the company that owns the factory and on legislation that is applicable to that business. To determine the progress of an optimization one or more objective functions are created from the design variables. The design variables describe all the matters that have effect on the optimization process. Mostly these variables operate in certain ranges. These constraint make sure that only feasible solutions will be found in the optimization. For the factory mentioned above it can be the case that all the machines are switched off during the night. This limits the variable operation hours.

A multi-objective optimization problem can be formulated as

$$f_{1,\ldots,n} : \mathbb{S} \rightarrow \mathbb{R} \tag{4.1.1}$$

defined on some search space $\mathbb{S}$ where $f_{1,\ldots,n}$ is minimized or maximized.

In order to solve a multi-objective optimization problem, one needs a mechanism that is able to find and keep all solutions that are equally optimal for the multi-objective problem. A mechanism that works well on this kind of problems is called Pareto optimization. In Pareto optimization vectors of objective function values are compared by a preference relation on the objective function vectors. Given a problem with multiple objectives to be minimized[1] and no constraints ($n_f > 1, n_g = 0$) the preference relation can be defined for arbitrary objective function vectors $\mathbf{y} \in \mathbb{R}^{n_f}$ and $\mathbf{y}' \in \mathbb{R}^{n_f}$

[1] Maximization problems can be transformed into minimization problems by inverting the sign of objective functions.

$$\begin{aligned} &\mathbf{y} \prec_p \mathbf{y}'(\mathbf{y} \text{ dominates } \mathbf{y}') :\Leftrightarrow \\ &\forall i \in \{1,\ldots,n_f\} : y_i \leq y'_i \wedge \\ &\exists i \in \{1,\ldots,n_f\} : y_i < y'_i \end{aligned} \tag{4.1.2}$$

This equation can easily be extended to a constraint problem where $n_g > 0$. Let $\mathbf{y} \in \mathbb{R}^{n_f+g_f}$ and $\mathbf{y}' \in \mathbb{R}^{n_f+g_f}$ be two arbitrary solution vectors. Their first n_f positions denote objective functions values to be minimized and the last n_g constraint values. Let $\mathbf{y}_f := (y_1,\ldots,y_{n_f})^T$, $\mathbf{y}_g = (y_{n_f+1},\ldots,y_{n_f+n_g})^T$, $\mathbf{y}'_f := (y'_1,\ldots,y'_{n_f})^T$, $\mathbf{y}'_g = (y'_{n_f+1},\ldots,y'_{n_f+n_g})^T$. Then

$$\begin{aligned} \mathbf{y} \prec_p \mathbf{y}' :\Leftrightarrow \\ \mathbf{y}_g \leq 0 \wedge \mathbf{y}'_g \leq 0 \wedge \mathbf{y}_f \prec_p \mathbf{y}'_f \vee \\ \mathbf{y}_g \leq 0 \wedge \mathbf{y}'_g > 0 \vee \\ \mathbf{y}_g > 0 \wedge \mathbf{y}'_g > 0 \wedge \delta(\mathbf{y}_g) < \delta(\mathbf{y}'_g) \end{aligned} \tag{4.1.3}$$

In Pareto optimization, the following definitions are useful: A solution is non-dominated by a set of solutions, iff no solution in this set dominates the solution. In this case the solution belongs to the efficient set $\mathbb{S}_e$.

$$x \preceq \mathbb{S}_e :\Leftrightarrow \nexists x' \in \mathbb{S}_e : x' \prec x \tag{4.1.4}$$

The Pareto front is defined as the set

$$\text{PF} = (f_1(\mathbf{x}),\ldots,f_m(\mathbf{x}))^\Gamma | \mathbf{x} \in \mathbb{S}_e) \tag{4.1.5}$$

Two solutions are incomparable, iff neither $x \prec x'$ nor $x' \prec x$.

Figure 4.1.1 shows an example of the ranking of a set $S = \{x_1, ..., x_6\}$ of solutions by means of the Pareto preference relation. Solutions x_1, x_2, and x_3 are non-dominated. Solution x_4 is dominated by x_1, x_5 is dominated by x_1, and x_2, solution x_6 is dominated by the complete non-dominated set x_1, x_2, and x_3. This makes solutions x_1, x_2, and x_3 of the same dominance rank and mutually incomparable. Furthermore points x_4, x_5, and x_6 form together also a set of the same dominance rank. The first dominance rank is for the non-dominated solutions. If all non-dominated solutions are removed from the set, the solutions that are non-dominated in the remaining set are of rank two, etc.

It would be an ideal situation if the complete Pareto front is found. This is a computationally expensive task and, if the Pareto front is a continuous curve, it can be even impossible. In this case there are in general infinitely many Pareto optimal solutions which makes it infeasible to find them all. Most of the time it suffices to find a limited number of the Pareto-optimal solutions, which include the extremal solutions and solutions in parts of the solution space where good compromise solutions can be found. Especially the ones located near knee-points. Knee-points are locations where small changes in the design variables cause major changes in one of the objective function values while the remaining objectives experience minor changes.

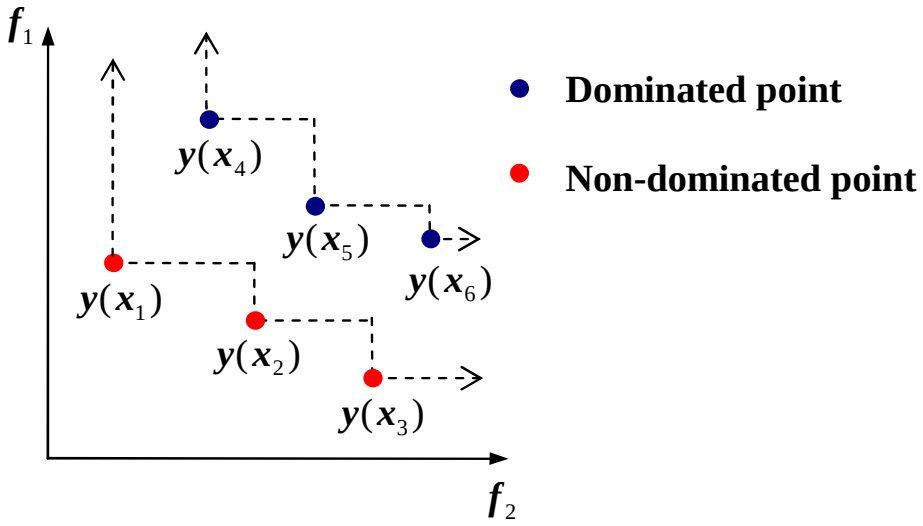


Figure 4.1.1: Pareto dominance.

Evolutionary multi-objective algorithms (EMOA) are well represented in the domain of multi-objective algorithms, because of their robustness and flexible design. Well-known EMOA are PESA, PAES, SPEA, SPEA2, SPEA2+, NSGA, NSGA-II and SMS-MEOA. A brief description for these algorithms is given below.

In a Pareto envelope based selection algorithm (PESA) selection is only based on an archive which stores the current non-dominated set. A density measure allows to sample the archive members differently according to the degree of crowding. Newly generated offspring is checked for replacement in the archive. Only offspring members that are not dominated replace dominated members in the archive, the remaining members are dismissed. This method never allows solutions in the archive to be dominated by other solutions in the archive.

The Pareto archived evolution strategy (PAES) [KC99] is a simple algorithm with starts with an initial randomly generated parent solution. To start, first mutate the parent and generate one offspring. If the offspring, dominates the parent, it replaces the parent for the next round and is added to the archive. If the offspring is dominated by the parent, it is discarded. If the parent and offspring are incomparable, the archive of previously non-dominated individuals is used to compare with. If the offspring dominates one of the archive members, it will be added to the archive and becomes the new parent. Any dominated solutions are removed from the archive. If the offspring does not dominate any member of the archive, both parent and offspring are checked for their crowded distances to the solutions in the archive. If the offspring is situated in a least crowded region of the search space compared to the crowded region of the parent, it is accepted as a parent and added to the archive. Repeat this cycle from start until a stop condition is reached. The archive maintains population diversity along the Pareto front.

The strength Pareto evolutionary algorithm (SPEA) [ZLT01] starts with an empty archive and an initial population. The following steps are repeated in each round: All non-dominated solutions are copied to the archive and at the same time any duplicates or dominated solutions are removed from the archive. Whenever the size of the archive exceeds a predefined limit further deletion takes place based on a clustering technique which preserves the properties of a non-dominated front. Hereafter fitness values are

assigned to the archive solutions and the population members. Each solution i in the archive is assigned a strength or fitness value $S(i) = F(i) = \frac{\sum_{j=1}^{n}(x(i) \preceq x(j) == \text{true})}{n+1}$ where n is the population size and $x(j)$ is a population member. The fitness $F(j)$ of a solution in the population is $\sum_{i=1}^{k} S(i)$ where $x(i) \preceq x(j)$ where k is the archive size and $x(i)$ is an archive member. The selection process follows a binary tournament selection, where each archive member has a higher chance to be selected than any population member. Finally, the old population is replaced by the resulting offspring population after recombination and mutation.

There are a couple of issues with SPEA. With one member in the archive the algorithm behaves like a random search algorithm. Despite of clustering information SPEA may lose extremal solution. SPEA2 takes care of these problems.

SPEA2's improvements [ZLT01] with respect to SPEA are the following: an improved fitness assignment scheme which takes into account for each individual how many individuals it dominates and it is dominated by. Moreover a nearest neighbour density estimation technique is used which allows better guidance of the search process, and finally a new archive truncation method guarantees the preservation of boundary solution.

SPEA2+ adds three important things to SPEA2: first of all neighbourhood crossover increases the rate of crossing over individuals that are located close to each other in the objective space, secondly mating selection is added that reflects all quality solutions in the archive and third, the use of two archives is supported to maintain diverse solution in the objective space as well as in the design space.

The non-dominated sorting genetic algorithm (NSGA) [DPAM02] implemented by Srinivas and Deb works like NSGA II that is described below, but with the shortcomings that after the proposal became clear. A major disadvantage is the estimation of the sharing parameter, which seems to be a difficult task. Other criticism is about the computational complexity of non-dominated sorting and the lack of elitism. Elitism is the ability to preserve certain quality solutions for the next generations.

The criticism of NSGA led to the development of non-dominated sorting genetic algorithm II (NSGA-II). The non-dominated sorting is improved: Elitist preservation is added and parameterless niching is included to preserve diversity in the population.

In NSGA-II the current population is first divided in into different fronts according to the concept of Pareto dominance. Objects in the first non-dominated front are ranked first, in the second non-dominated front the second highest rank until all fronts are assigned a rank. See page 9 about Pareto dominance. Within a rank a crowding measure is used to define an order between the individuals. The sum of the distances to the closest two neighbour points along each objective is used to define an order for these points. Extremal solutions that have no neighbouring solutions in at least one of the coordinate directions are always preferred to non-extremal solutions. After ranking all solutions, selection can be carried out in the usual way. Typically $(\mu + \mu)$ selection is used in NSGA-II. In this way the archive maintains a constant size, while for PEAS mentioned earlier the size can be flexible.

Though NSGAII and SPEA2 are widely used algorithms, SMS-EMOA[2] has shown in several test problems it performs better. In [EBN05] is demonstrated that SMS-EMOA in the summarized result outperforms SPEA2 and NSGA-II with the conver-

[2]SMS-EMOA stands for S-Metric Selection-EMOA

gence and S measures on the ZDT-suite. What contributes to the better performance is that SMS-EMOA covers knee point regions in a better way than NSGAII does. Despite of the often stated conjecture that S-metric based selection favors convex parts of Pareto fronts, empirical results do not provide evidence for this alleged disadvantage. [EBN05] show that SMS-EMOA is able to find a concave Pareto front with nearly optimal hypervolume for the ZDT2 problem. Recent theoretical results on optimal distributions of points when the S-metric is maximized comply with this and disapprove the conjecture. The shape of a Pareto front is not necessarily known beforehand. The shape can be convex, concave or linear. In figure 4.1.2 the three possible shapes have been approximated for a two-objective minimization problem with the SMS-EMOA. The selected EMOA for the algorithm in this thesis is therefore SMS-EMOA.

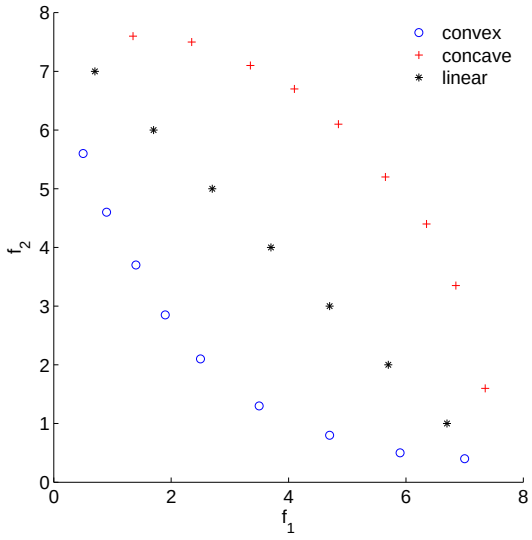


Figure 4.1.2: Different Pareto front shapes.

4.2 Hypervolume S-metric

A disadvantage of NSGA-II and other EMOA is the inability to measure improvement during the optimization. For the SMS-EMOA there is a straightforward way for measuring progress. S-metric selection is based on the hypervolume measure. This mechanism can be used to measure the quality of a Pareto front approximation as well as a criterion for comparing solutions of the same dominance rank.

Figure 4.2.3 illustrates the hypervolume measure for a bi-objective problem. Vector y_{max} is chosen in such a way that its elements are larger than every possible value of y_1 and y_2. To control the importance of the objectives one can variate the value of y_{max} that is assigned to each objective. The size of the dominated space measures the progress of the optimization. Furthermore, is the space that is dominated by a point also a measure for the contribution of this point to the current Pareto front. The hypervolume measure was originally proposed by Zitzler and Thiele [ZT98]. In two dimensions the hypervolume measure $S(M)$ is the area of the union of rectangles r_i defined by a set of a non-dominated solution m_i in A and the reference point y_{max} that is dominated by all solutions in A.

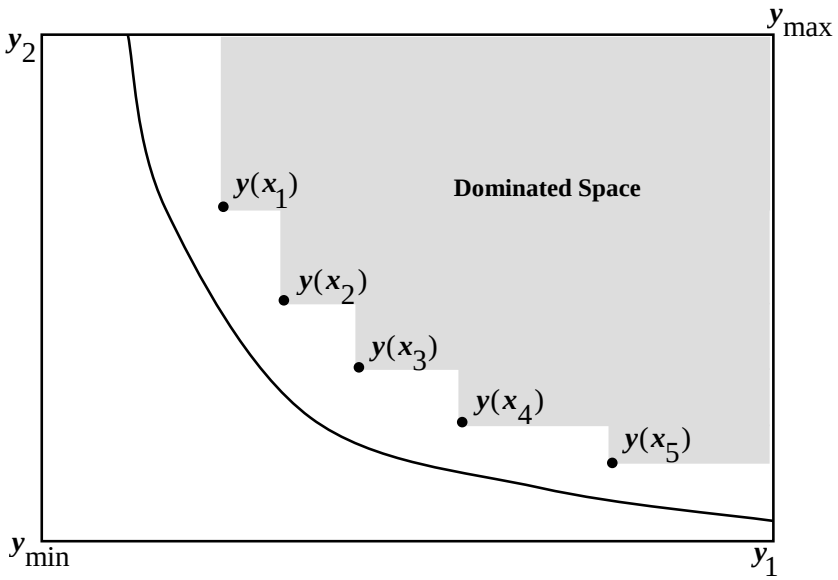


Figure 4.2.3: Two-dimensional hypervolume example.

$$S(M) := \{\bigcup_i r_i | m_i \in M\} = \bigcup_{m \in A} \{y | m \prec y \prec y_{max}\} \tag{4.2.6}$$

4.3 S-metric selection - evolutionary multi-objective algorithm

The SMS-EMOA is an algorithm that fully utilizes the non-dominating sorting procedure in combination with the hypervolume measure. As well as for measuring the quality of a Pareto front as for individual solution replacement the same process is executed. The population size is kept constant and dominated solutions are allowed in the population. For a series of populations the following equation always holds $S(P_{t+1}) \leq S(P_t)$. Depending on the choice of the reference point it can affect the contribution of the extremal solutions to the hypervolume measure. There is some criticism to this fact. To circumvent this problem an infinite reference point is chosen. The result of this change is that $S(M)$ becomes an incomparable value for any set of solutions M. However, the increase in hypervolume $\triangle_S = S(M \cup \{\mathbf{x}\}) - S(M)$ can take a finite value for non-extremal solutions. For extremal solutions $\triangle_S$ always equals infinity. Therefore it has been suggested in [EBN05] to keep extremal solutions for the next generation. As in the case of two objectives the number of extremal solutions is at most two, this causes no problems in terms of memory size. It would be interesting to compare the behavior of the crowding distance that is used NSGA-II to the hypervolume measure that is used in SMS-EMOA regarding single solutions around a knee-point in convex areas of the Pareto front. This has been done in figure 4.3.4. The hypervolume measures of $y(x_4)$ and $y(x_5)$ show that solution x_4 is preferred above x_5. This seems to be the right way of assessing the two points, indeed solution x_4 is a good compromise point for the two objectives and is interesting for further exploration. Solutions x_5 has less chance to be improved and a small advance for objective f_2 results in large decay of f_1. On the other hand, NSGA-II assesses the two solutions otherwise and prefers x_5. This is a point in favor of SMS-EMOA.

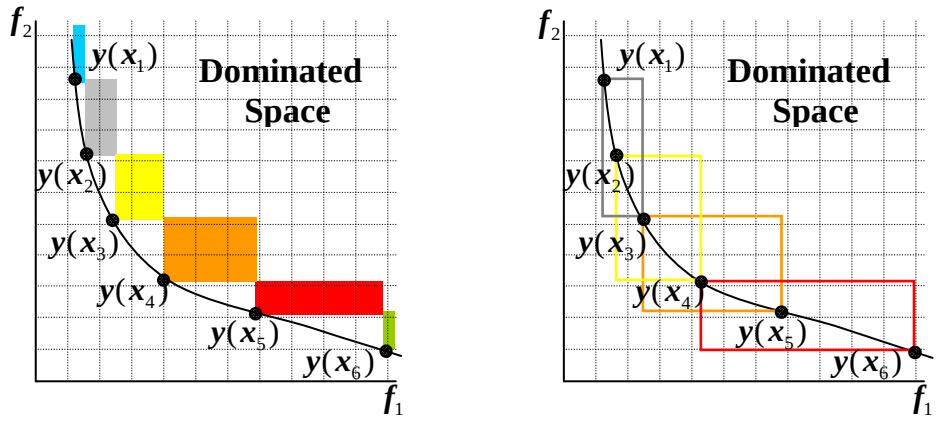


Figure 4.3.4: Ranking differences of SMS-EMOA (left) and NSGA-II (right).

An outline of a basic SMS-EMOA is given in Algorithms 1 and 2. First a parent population is initialized. Next an offspring is created from a randomly selected parent by variation operators, mutation and recombination. Select μ solutions out of the $\mu + 1$ solutions for the new population. To replace the worst solution find the worst dominating front R_w by using non dominating sorting. Next, find the solution with the lowest hypervolume in the worst front R_w and remove it from the population. Repeat this round until the stop criterium is reached.

Algorithm 1 SMS-EMOA

1: $P_0 \leftarrow$ **initialize**() { initialize random parent population with μ individuals}
2: $t \leftarrow 0$
3: **repeat**
4: $\quad x_{t+1} \leftarrow$ **generate**(P_t) {Generate one offspring by variation operator}
5: $\quad P_{t+1} \leftarrow$ **replace**$_{\triangle S}(P_t \cup \{x_{t+1}\})$ {Select a maximum of μ individuals for the new parent population}
6: $\quad t \leftarrow t + 1$
7: **until** stop criterium is reached

Algorithm 2 replace$_{\triangle S}(Q)$

1: $\{R_1, \ldots, R_l\} \leftarrow$ **non-dominated-sort**(Q)
2: **for all** $\mathbf{x} \in R_l$ **do**
3: $\quad \triangle_S(\mathbf{x}, R_l) \leftarrow S(R_l) - S(R_l \backslash \{\mathbf{x}\})$
4: **end for**
5: $\mathbf{x} \leftarrow \arg\min_{\mathbf{x} \in R_l}[\triangle_S(\mathbf{x}, R_l)]$
6: $Q' \leftarrow Q \backslash \{\mathbf{x}\}$

In chapter seven a more elaborate description is given for the different aspects of the created algorithm.

5 Robustness and optimization

5.1 Introduction

Many real-world optimization problems are subject to uncertainties and noise. These uncertainties and noise are caused by manufacturing errors, measurement errors and external factors, e.g. unpredictable weather changes. The uncertainties emerge in different parts of the optimization process. This makes it necessary to make a distinction between the different uncertainties. An overview of robust optimization methods can be found in [BS07].

5.2 Uncertainty involved in different parts of the system

An objective function with no uncertainty involved is given by $f(\mathbf{x}, \mathbf{a})$ were $\mathbf{x}$ is a design variable and a contains the environmental conditions.

There are four categories of uncertainties. A description follows for each of them.

To the first category belong uncertainties that occur because of changing environmental and operating conditions. Examples are temperature changes, pressure changes, humidity changes and changing material properties like expanding and shrinking of material caused by the temperature. The uncertainties are modeled by vector α. Vector $\mathbf{x}$ is a design object with n variables $x_1, ..., x_n$. These two vectors form the input for objective function $f(\mathbf{x}, \alpha)$.

$$\mathbf{f} = \mathbf{f}(\mathbf{x}, \alpha) \tag{5.2.1}$$

The second type of uncertainties δ arise in the design variables. In real world problems these uncertainties are mainly caused by approximative realizations of the design variables. Approximations are sometimes inevitably because of machinery or sensors which operate under a certain degree of accuracy. Although it may be possible to minimize the uncertainties by using high precision equipment, the cost for these equipment may be much higher than the budget allows.

$$\mathbf{f} = \mathbf{f}(\mathbf{x} + \delta, a) \tag{5.2.2}$$

The use of models instead of precise data and the existence of measurement errors in the calculation of the system output leads to uncertainties in the objective function values. Design variable $\mathbf{x}$ and environmental variable α are also generated by a model and therefore subject to uncertainty. The actually output $\overline{f}$ is a (random) function of f

$$\overline{\mathbf{f}} = \overline{\mathbf{f}}[\mathbf{f}(\mathbf{x} + \delta, \alpha)] \tag{5.2.3}$$

The fourth type of uncertainties that appear in the constraints of design variables are called feasibility uncertainties.

For the four categories of uncertainties described above, three types of quantifications are possible. Namely a deterministic, probabilistic and/or possibilistic quantification.

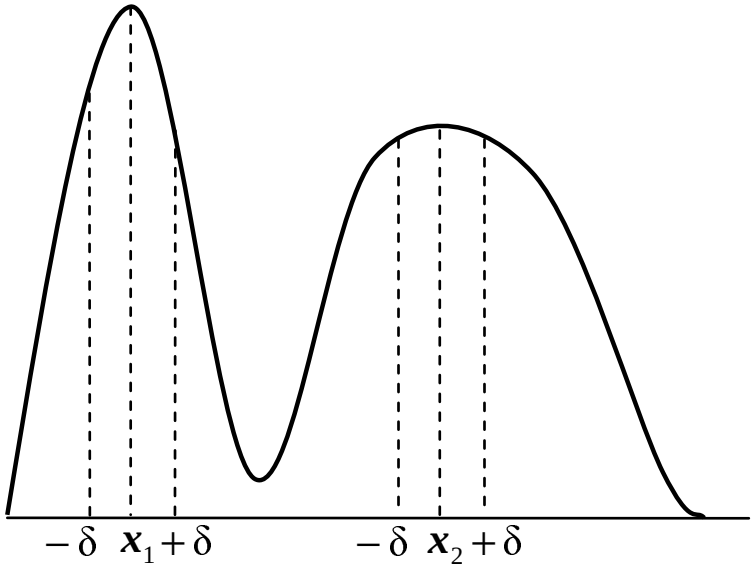


Figure 5.3.1: Quality-stability trade off in robust optimization.

The deterministic type defines parameter domains in which the parameters can vary. For the probabilistic type a probability density functions can be used and the possibilistic type uses fuzzy measures to define the plausibility of a certain event.

In the case study of this thesis the uncertainties are attributed to the environmental group of uncertainty variables. Four uncertainty variables make use of a probabilistic quantification and one variable belongs to the deterministic group.

5.3 Effects of uncertainty on the Pareto front

Although the manufacturing errors and measurements errors may be within an acceptable range, they can have a great influence on the characteristics of the building. Some minor deviations in the input variables of a system to be optimized may result in great deviations in the objective function values. The goal of robust optimization is not only to optimize the objectives, but also to take care of deviations of objective function values caused by small or large changes or fluctuations in the input variables. For multi-objective optimization this means that instead of looking for the global non-robust Pareto front one is looking for the global robust Pareto front.

Figure 5.3.1 illustrates an example of a simple one dimensional optimization problem for a continuous function with one design variable, where the consideration of robustness leads to a different global optimum if a worst case scenario is executed or performance fluctuation is fatal. If one is searching for the global optimum x_1 is the preferred solution above x_2, which forms a local optimum. However if the design variables x_1 and x_2 are liable to a deterministic quantified uncertainty δ this should have an effect on the function value of x_1 much more than the function value of x_2. Variable x_2 has a nearly constant performance with respect to all possible variations around this variable. Variations around x_1 on the other hand result in relatively higher performance drops and in the worst case the performance drops below the worst objective function value possible for x_2. If guaranteed performance is required one should choose for object x_2. If instability of the objective function is not a problem one can choose for object x_1.

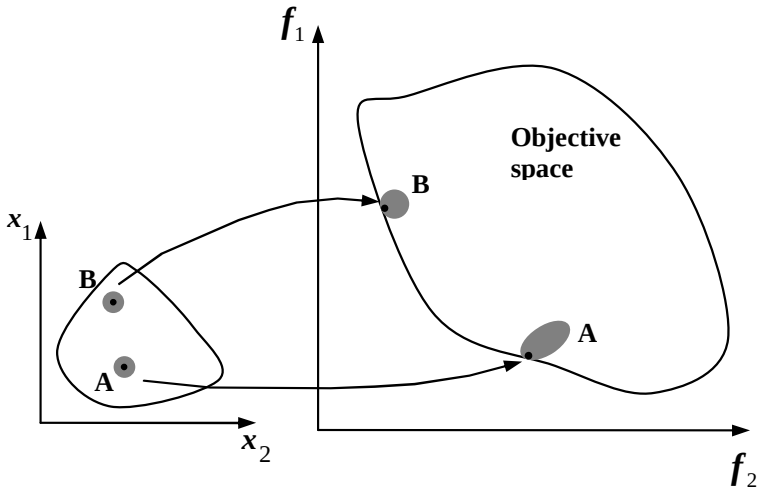


Figure 5.3.2: Quality-stability trade off in robust two-dimensional optimization.

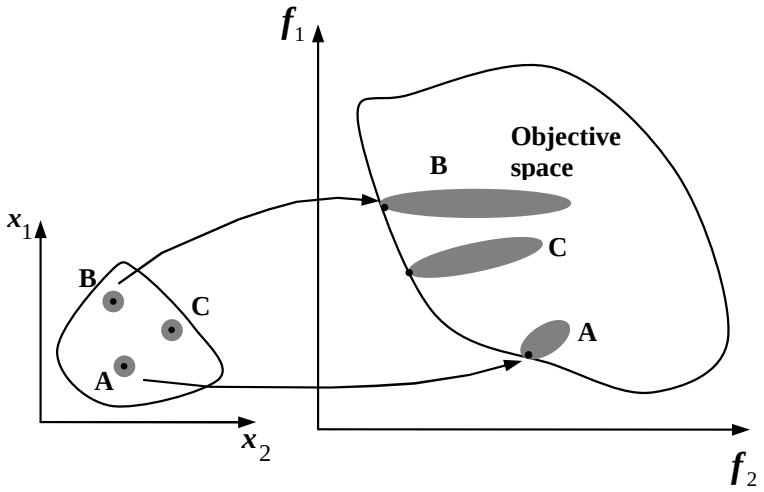


Figure 5.3.3: Pareto front shift.

To understand the idea of a (partially) shifting Pareto front in robust optimization an example of a two-dimensional minimization problem is given in figure 5.3.2. Now we have two design variables x_1 and x_2 and two objectives f_1 and f_2. The grey areas around the points in the $x_1 - x_2$ diagram represents the uncertainty of the design variables. Labels A and B connect the two objects with their corresponding objective function values. In this case B has less decline regarding the average performance than A has. If one thinks of an imaginary Pareto front through the points (black dots) in the $f_1 - f_2$ diagram and thinks of an other one though the worst case points, one can see a shift in the direction of the upper right corner of the robust Pareto front. However, the domination of the two objects does not change in this case. The objective function values of the uncertainty sets A and B are incomparable in the robust Pareto front.

It can be the case that a non-dominated object in the Pareto front is dominated by an other object in the robust Pareto front. In this case this object is not of much interest anymore and is possibly replaced by a non-dominated solution. Figure 5.3.3 demonstrates the case. There is a considerable chance that either B or C gets dominated in the robust Pareto front.

Two different main methods can be used to achieve robustness in your solutions [DG06]. The first method replaces the objective function with a mean effective function.

$$\text{Minimize } f^{eff}(\mathbf{x}) = \frac{1}{|\mathcal{B}_\delta(\mathbf{x})|} \int_{\mathbf{y} \in \mathcal{B}_\delta(\mathbf{x})} f(\mathbf{y}) d\mathbf{y} \text{ and } \mathbf{x} \in \mathbb{S} \tag{5.3.4}$$

Instead of choosing a best solution for the next generations an average solution is chosen. For this method it is required to have access to the objective function.

The second method calculates a normalized difference between f and the perturbed function value f^p. Operator $\|.\|$ can be any suitable norm. Parameter η is a threshold, which controls the degree of robustness.

$$\text{Minimize } f(\mathbf{x}) \text{ subject to } \frac{\|f^p(\mathbf{x}) - f(\mathbf{x})\|}{\|f(\mathbf{x})\|} \leq \eta \tag{5.3.5}$$

In this thesis a variant of the first method is proposed. Around a solution $\mathbf{x}$ two hundred and one perturbations are generated. The objective function value is obtained for each perturbation and the worst solution is selected. It depends on the choice of the algorithm which solution is actual the worst. See section 4.3 where SMS-EMOA and NSGA-II assess two objective function vectors differently. The objective function is a black box and therefore not adjustable as suggested in expression 5.3.4.

6 Metamodels

6.1 Introduction

When an algorithm requires a large number of objective function calls to a computationally expensive function, this function can be partially replaced with a model used to compute the objective function faster, but approximated. For the problem in this thesis, where Monte Carlo sampling is used to allow robust optimization and the objective function is computationally time expensive it is preferable and even necessary to partially replace evaluations with metamodels to achieve an acceptable runtime for the optimizations. Metamodels[3] are an important class of surrogate models. They are based on the data of previous evaluations with the original model, which in most cases they interpolate.

The model must be capable of spatial interpolation and is able to deal with multi-objective problems. A type of metamodel that is capable of these demands is a gaussian random field model (GRFM) and will be discussed in the next section.

6.2 Gaussian random field models

The basics of a gaussian random field model. Assume we have n evaluated search points

$$\mathbf{X} := [x_1, ..., x_n] \in \mathbb{S} \tag{6.2.1}$$

where $\mathbb{S}$ is the search space. Furthermore we have corresponding objective function values

$$\mathbf{y} := [y_1, ..., y_n] \tag{6.2.2}$$

with

$$y_1 = y(x_1), ..., y_n = y(x_n) \tag{6.2.3}$$

that have been calculated by the original function. Information about the degree of differentiability of the function is not required. However we assume that the function is continuous.

Given a new point $x' \in \mathbb{S}$, the aim is to create a model, capable of approximating the value of $y(x')$. If $x' \in \mathbf{X}$ is a precalculated value $y(x')$ will be returned.

Figure 6.2.1 shows a visual example of the prediction of y for an unknown point x'. In the figure there are three precalculated points which are used as training points to predict the intermediate unknown points. The bold line is the predicted response $\hat{y}(x)$, $x \in \mathbb{R}$. The two dashed lines which meet at the training points and then diverge from the training points define the confidence area of the response. The size of the confidence area is determined by adding or subtracting an estimated local standard deviation $\hat{s}(x')$. If $\hat{y}(x')$ equals one of the known values at the location of the training

[3]Metamodels are named this way as they are models of models

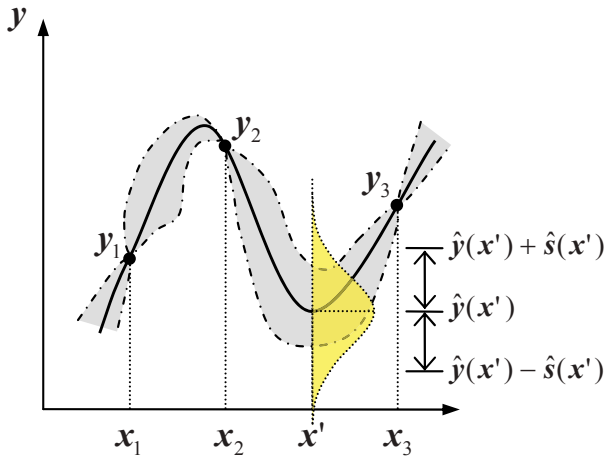


Figure 6.2.1: Gaussian Random Field Model in 1-D.

patterns, then $\hat{s}(x') = 0$. If this is not the case $\hat{s}(x')$ grows relatively to the distance of the unknown point to the surrounding training points. The divergence speed is determined by a correlation parameter θ, that will be discussed later in this thesis.

It is plausible to assume that an increasing number of training points in the neighbourhood of the unknown point results in a better qualitative prediction. To validate this assumption a graph with a comparison of neighbourhood sizes is given in the result section.

A GRFM predicts the outcome of gaussian random field $\mathcal{F}_{\mathbf{x}}$, $\mathbf{x} \in \mathbb{R}^d$. A gaussian random field[4] is a mapping that assigns an one-dimensional gaussian distributed random variable $\mathcal{F}(\mathbf{x})$ with constant mean $\beta := E(\mathcal{F}_{\mathbf{x}})$ and variance $s^2 = Var(\mathcal{F}_{\mathbf{x}})$ to each point $\mathbf{x} \in \mathbb{R}^d$ and it quantifies the probability of density $Pr(\mathcal{F}_{\mathbf{x}} = y)$ for the unknown precise output. If $Pr(\mathcal{F}_{\mathbf{x}} = y)$ takes a high value, the GRFM predicts that y is more likely the precise result. Once the gaussian random field has been sampled, the sample path $f(x)$ forms a non-random function of x. This deterministic function is called a sample path or a realization.

In GRFM it is assumed that the correlation between the errors of the estimated values is related to the distance between the corresponding points. Such a spatial correlation can be expressed by a correlation function.

The following isotropic gaussian correlation is often used.

$$c(\theta) = \exp(-\theta \cdot |x - x'|) \tag{6.2.4}$$

6.3 Kriging

This section describes the basics of Kriging. Kriging is a combination of GRFM and regression models. It is commonly used interpolation technique which takes surrounding points to predict an unknown point in the search space.

[4]For $d = 1$ and sometimes also for $d > 1$ a gaussian random field is referred to as a gaussian process.

$$\mathcal{F}_{\mathbf{x}} = \sum_{i=1}^{n_r} \beta_i \cdot r_i(\mathbf{x}) + \mathcal{R}_{\mathbf{x}} \tag{6.3.5}$$

The part before the plus sign is called the global trend, and after the sign local deviation.

There are three Kriging variants available. With simple Kriging no trend is assumed, i.e. $\mathcal{F}_{\mathbf{x}} = \mathcal{R}_{\mathbf{x}}$. In ordinary Kriging a constant trend is assumed, $\mathcal{F}_{\mathbf{x}} = \beta + \mathcal{R}_{\mathbf{x}}$ and in universal Kriging a linear trend is assumed like in expression 6.3.5. In this thesis ordinary Kriging is used to start with.

To make a Kriging model work θ, s^2 and β have to be estimated. These parameters will be estimated in a calibration phase. After the calibration is finished, the parameters of the GRF are fixed. At this point predictions can be computed by the model for every input vector. For a point x' the model is now able to return a prediction $\hat{y}(x')$ and an error $\hat{s}(x')$.

Parameter θ is estimated by the maximum likelihood heuristic [SWMW00], β and s^2 are estimated through a sample by a generalized least squares method. The likelihood of a sample $X, \mathbf{y}$ is expressed by the joint distribution of the probability density function of $\mathcal{F}_i, i = 1, \ldots, n$ and looks as follows:

$$\begin{aligned} &\mathrm{PDF}(\mathcal{F}_{x_1} = y_1 \wedge \cdots \wedge \mathcal{F}_{x_m} = y_m) = \\ &\frac{1}{(2\pi)^{m/2} \cdot (\hat{s})^{m/2} \cdot \sqrt{\det(C)}} \exp\left[-\frac{(\mathbf{y} - 1\hat{\beta})^T \cdot C^{-1} \cdot (\mathbf{y} - 1\hat{\beta})}{2\hat{s}} \right] \end{aligned} \tag{6.3.6}$$

with correlation matrix C obtained from the correlation function $c(x, x')$ of that field

$$\mathbf{C} = \begin{bmatrix} c_\theta(\mathbf{x}_1, \mathbf{x}_1) & \cdots & c_\theta(\mathbf{x}_1, \mathbf{x}_m) \\ \vdots & \ddots & \vdots \\ c_\theta(\mathbf{x}_m, \mathbf{x}_1) & \cdots & c_\theta(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix}, \mathbf{1} = \begin{bmatrix} \mathbf{1} \\ \vdots \\ \mathbf{1} \end{bmatrix} \tag{6.3.7}$$

using the generalized least squares estimates of β and s^2

$$\hat{\beta} = \frac{\mathbf{1}^T \cdot C^{-1} \cdot \mathbf{y}}{\mathbf{1}^T \cdot C^{-1} \cdot \mathbf{1}} \tag{6.3.8}$$

$$\hat{s} = \frac{(\mathbf{y} - \mathbf{1} \cdot \hat{\beta})^T \mathbf{C}^{-}1(\mathbf{y} - \mathbf{1} \cdot \hat{\beta})}{m} \tag{6.3.9}$$

If $\hat{s}$ in expression 6.3.6 is replaced by expression 6.3.9 parts of the fraction in the exponent term are crossed out.

$$\frac{1}{(2\pi)^{m/2} \cdot (\hat{s})^{m/2} \cdot \sqrt{\det(C)}} \exp\left[-\frac{\cancel{(\mathbf{y} - 1\hat{\beta})^T \cdot C^{-1} \cdot (\mathbf{y} - 1\hat{\beta})}}{2 \cdot \cancel{\frac{(\mathbf{y} - \mathbf{1} \cdot \hat{\beta})^T \mathbf{C}^{-}1(\mathbf{y} - \mathbf{1} \cdot \hat{\beta})}{m}}} \right] \tag{6.3.10}$$

What follows is the next expression which is to be maximized:

$$\frac{1}{(2\pi)^{m/2} \cdot (\hat{s})^{m/2} \cdot \sqrt{\det(C)}} \exp(-\frac{m}{2}) \rightarrow \max \tag{6.3.11}$$

This is equal to the minimization of the reciprocal term:

$$(2\pi)^{m/2} \cdot (\hat{s})^{m/2} \cdot \sqrt{\det(C)} \cdot \exp(-\frac{m}{2}) \rightarrow \min \tag{6.3.12}$$

After logarithmization and elimination of constant values the following expression remains to be minimized:

$$m \log \hat{s}(\theta) + \log \det \mathbf{C}(\theta) \tag{6.3.13}$$

After estimation of the parameters the calibration phase has finished. For every $x \in \mathbb{S}$ the mean is calculated as follows:

$$\hat{y}(\mathbf{x}) = \beta + (\mathbf{y} - \mathbf{1}\beta)^T \mathbf{C}^{-1} \mathbf{c}(\mathbf{x}) \tag{6.3.14}$$

where

$$\mathbf{c}(\mathbf{x}) = [c_\theta(\mathbf{x}, \mathbf{x}_1), \ldots, c_\theta(\mathbf{x}, \mathbf{x}_m)]^T \tag{6.3.15}$$

Above expressions can be rewritten in the form of a linear predictor

$$\hat{y}(\mathbf{x}) = \beta + \sum_{i=1}^{m} \lambda^{(i)} \cdot c(\mathbf{x}, \mathbf{x}_i) \tag{6.3.16}$$

with

$$[\lambda^{(1)}, \ldots, \lambda^{(m)}] = (\mathbf{y} - \mathbf{1}\beta) \cdot \mathbf{C}^{-1} \tag{6.3.17}$$

The local variance of a variable is not used in this thesis and therefore not stated. More information about the variance is given in [Emm04] .

6.4 Radial basis functions networks

Artificial neural networks (ANN) use a biological analogy for the way networks operate, in order to process information, just like evolutionary algorithms use the theory of biological evolution as an analogy in optimizations. The idea of ANN is based on the structures found in the cerebral cortex in the human brain. The cerebral cortex contains billions of neurons (processing units) that will or will not fire (let through, exchange signals) depending on the connections they have with other neurons. A neuron will fire if a certain electrochemical threshold is reached.

Although the representation is the same as in biology, ANN neurons have a far more simple representation than their biological counterparts. ANN use interconnected processing units to form a processing data system. The threshold of these units is a certain value which can be reached by the sum of weights of incoming units. If an unit

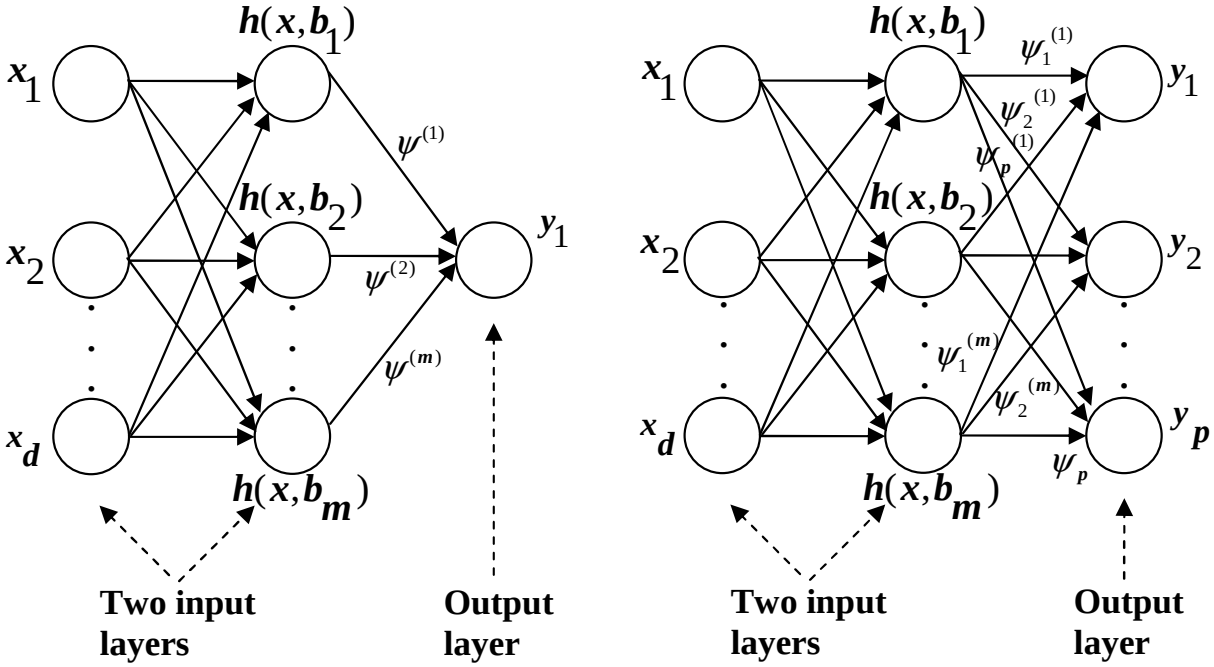


Figure 6.4.2: Radial Basis Function Network.

fires a so-called activation function produces the output of unit. Such a connected network can be trained to learn data (pictures, numbers). When the network has reached a stable state, it can for example act on incomplete data input and estimate the unknown values.

Radial basis functions networks (RBFN) are the most common ANN used for the approximation of functions and especially for interpolation. RBFN consist of three layers which are all connected. All signals flow from the inputs through the units and reach the outputs. No signals flow from the hidden layer to the input layer and no signals flow from the output units to the hidden layer or input layer. Under these conditions the network is called a feedforward network. A visualization is given in figure 6.4.2. The network performs a nonlinear mapping from the d inputs to the m hidden units followed by a linear mapping from the hidden units to the p outputs. In the left figure $p = 1$.

Giannakoglou [Gia02] introduced a method which uses a RBF network as a function interpolater. It will be shown that these networks are actually use the same equations as in simple Kriging.

Let $x^{(1)}, \ldots, x^{(m)}$ be the points precalculated by the objective function and $y^{(1)} = y(x^{(1)}), \ldots, y^{(m)} = y(x^{(m)})$. Then define for each precalculated point $x^{(i)}$ a RBF center.

$$\mathbf{b}^{(i)} := \mathbf{x}^{(i)}, i = 1, \ldots, m \tag{6.4.18}$$

Let $|.| : \mathbb{R}^d \to \mathbb{R}_0^+$ denote a norm on $\mathbb{R}$ and $r : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ a positive definite function on $\mathbb{R}_0^+$, then the activation function of the hidden layer is defined as follows:

$$h(\mathbf{x}, \mathbf{b}^{(i)}) := r(|\mathbf{x} - \mathbf{b}^{(i)}|), i = 1, \ldots, m \tag{6.4.19}$$

The activation function based on r is called a radial basis function, because it depends

on the distance to the RBF center.

$$\hat{y}(h^{(1)}, \cdots, h^{(m)}) = \sum_{i=1}^{m} \psi^{(i)} h(\mathbf{x}, \mathbf{b}^{(i)}) \tag{6.4.20}$$

The function from the output values of the hidden layer to the output value of the RBFN is defined as a linear expression with weights ψ that have to be adapted during the training phase. If the network is given a known input vector $x^{(j)}$ the sum of functions must be $y^{(j)}$. This gives the general expression:

$$\sum_{i=1}^{m} \psi^{(i)} h(\mathbf{x^{(j)}}, \mathbf{x}^{(i)}) \stackrel{!}{=} y^{(j)}, j = 1, \ldots, n \tag{6.4.21}$$

Rewritten in matrix form it results in:

$$\underbrace{\begin{bmatrix} h(\mathbf{x}^{(1)}, \mathbf{b}^{(1)} & \cdots & h(\mathbf{x}^{(1)}, \mathbf{b}^{(m)} \\ \vdots & \ddots & \vdots \\ h(\mathbf{x}^{(m)}, \mathbf{b}^{(1)} & \cdots & h(\mathbf{x}^{(m)}, \mathbf{b}^{(m)} \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} \psi^{(1)} \\ \vdots \\ \psi^{(n)} \end{bmatrix}}_{\psi} \stackrel{!}{=} \underbrace{\begin{bmatrix} y^{(i)} \\ \vdots \\ y^{(m)} \end{bmatrix}}_{\mathbf{y}} \tag{6.4.22}$$

Because of the fact that the RBF centers $b^{(i)}, i = 1, \ldots, m$ are equivalent to the input points $x^{(i)}, i = 1, \ldots, m$ and the symmetry of the distance measure, if follows that matrix $\mathbf{H}$ is a symmetrical matrix.

Under the requirements there are no equal points in the database and the RBF is positive definite, the weights are given by the solution of this system, i.e.

$$\psi = \mathbf{H}^{-1}\mathbf{y} \tag{6.4.23}$$

At this moment a link to simple Kriging comes into mind. If the correlation function c in the matrix of equation 6.3.7 in the GRFM is replaced by the activations functions h in the RBFN, it becomes clear that $\mathbf{H}\hat{=}\mathbf{C}$ and $\psi\hat{=}\lambda$. See 6.3.17. The special case with simple Kriging where β is given a priori is thus equivalent to the prediction with RBFN explained here.

The possible advantage of ANN with respect to GRFM is the possibility to have multiple output units at the same time which allow to train all outputs in one training phase. This is not researched during the thesis and will not be discussed further.

7 Description of the algorithm

7.1 VA114 interface

The interface to the VA114 simulation tool consists of eight Matlab files. Two Matlab files are for template support. Five Matlab files are used to write data to the VA114 input files, which prepares the VA114 system for the next simulation round. One Matlab file reads the output file of VA114 and return the objective function values. See figure 7.1.1.

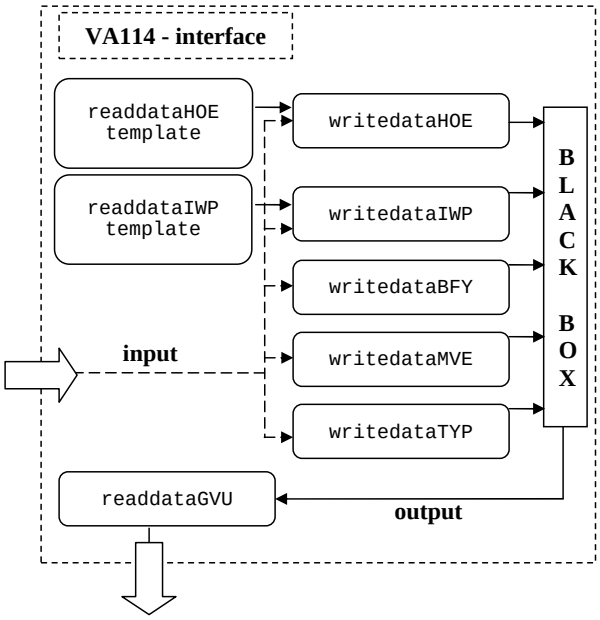


Figure 7.1.1: Matlab interface files for VA114.

7.2 Archive characteristics

To support a metamodel it is required to prepare an archive of precalculated points before the algorithm starts to run. The samples in the archive are preferred to cover the search space as good as possible. Sampling types are random sampling, latin hypercube sampling (LHS) and orthogonal sampling. See figure 7.2.2 for visual two-dimensional representations of these sampling methods. Random sampling does not guarantee any coverage of the search space at all, but information about the number of samples is not required to be available before starting. With latin hypercube sampling the number of points P must be known beforehand. The range of each variable in the design space is divided in M equally probable intervals and in each interval a sample point is placed. If this is carried out for each variable eventually all points lie on a virtual grid and it guarantees that each point is not engaging an other point on the grid. Orthogonal sampling is LHS with an added restriction. Also the search space is divided in subspaces of equal size. Besides the LHS distribution each subspace must be sampled with the same density. Orthogonal sampling is therefore more difficult to implement. In this thesis latin hypercube sampling is chosen from the three methods. Attempts are made to create LHS distributions with an improved total Euclidean distance. As it did not result in noticeable improvements in the metamodel approximations, the detailed description of this approach is left out.

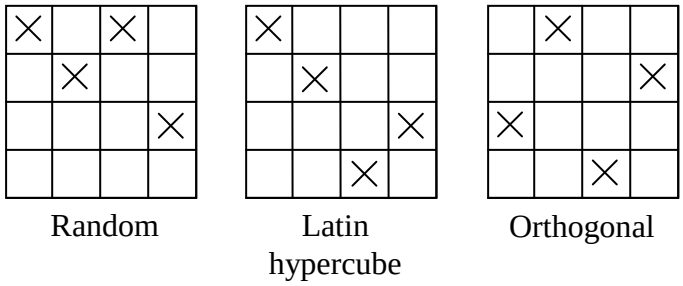


Figure 7.2.2: Sampling types.

Some of the variables with uncertainties involved are unbounded because of the nature of their uncertainty distributions. The uncertainties all have gaussian distributions. The lower boundary is set by subtracting four times the standard deviation from the mean and the upper boundary by adding four times the standard deviation. In this way at least 99.9% of the search space is covered.

New archive points which are calculated by the objective function, are not allowed to look similar to existing archive points, as this can disturb the calibration phase of the metamodel. They do not affect the global metamodel, because this model is only initialized once before the algorithm starts to run. The local metamodel on the other hand can use the new archive points as possible neighbours to an unknown solution that is estimated next.

7.3 Algorithm

The metamodel supported robust $(\mu + 1)$ SMS-EMOA has to find robust solutions by means of a worst case scenario.

There are a number of parameters to be adjusted. One can adjust the size of the population μ. Whenever the size is to small areas with possibly good solutions are not represented by the population. If the size to big convergence can slow down.

The mutation strength controls the mutation stepsize. A high mutation strength means more exploration of the search. This can result in good convergence speed at the beginning and may avoid local minima. A problem may be slow convergence and lack of precision to a non-global local minimum. A low mutation strength on the other hand may steadily improve the population during the complete optimization, but steps in the first local minimum it encounters.

Simple mutation strength adaptation cuts the mutation strength in half after a fixed number of unsuccessful rounds where the population did not improve. The number of unsuccessful rounds can be set.

The neighbourhood size defines how may neighbours will be selected to estimate an unknown point. Theoretically the size can vary from one to the size of the archive.

The algorithm works as follows. First a parent population P_t is initialized. Each parent is perturbed with a Monte Carlo sampling of two hundred and one samples. For each perturbed point the objective function value is estimated with a metamodel. Non-dominated sorting is executed to partition the fronts in increasing order. The worst front is the front that is dominated by all remaining fronts. For each point in the worst front the hypervolume contribution of this point is calculated. For the

sample with the smallest hypervolume contribution the precise value is computed by the objective function. This value is saved in the parent population P_t and in the archive. An outline is given in Algorithm 3.

Algorithm 3 initializeParentPop(A)

1: $P_w \leftarrow$ **initialize**(μ) {Initialize randomly μ parents}
2: **for all** $p \in P$ **do**
3: $Y_e \leftarrow \emptyset$ {Array of estimated objective function values}
4: $Perturbations \leftarrow$ **createPerturbations()** {Create perturbations (Monte Carlo Sampling)}
5: **for all** $pert \in Perturbations$ **do**
6: $Y_e \leftarrow Y_e \cup$ **metamodel**(p, $pert$) {Perturb parent p with perturbation $pert$ and calculate the corresponding objective function value with a metamodel. Put the result in Y_e}
7: **end for**
8: $index_w \leftarrow$ **getWorstIndex**(Y_e) {Find in Y_e the worst objective function value and return the index}
9: $P_w \leftarrow P_w \cup$ **createParent**(p, $pert[index_w]$)
10: **end for**
11: $(P_w, A) \leftarrow$ **calculatePreciseValuesAndUpdateArchive**(P_w)
12: **return** P_w

After the initialization of the parent population the algorithm will run for a predetermined number of rounds. In each round a parent is randomly chosen from the population and mutated by the variation operator mutation to create an offspring. The offspring is perturbed with two hundred Monte Carlo samples and the worst sample o_w is found in the same way as in the parent population initialization. To select μ best solutions from the population extended with the new offspring $P_t \cup o_w$, the replacement operator in Algorithm 2 of chapter four is used. If the offspring indeed improves the parent population, its estimated value is replaced by a precise value calculated by the objective function. This new calculated point is added to the archive. An outline is given in Algorithm 4.

In this particular case the objective function is time expensive. When solving a robust optimization problem this can be an issue because of the high number of objective function calls that is used. A solution to this issue is the integration of a metamodel into the algorithm. A metamodel simulates the objective function and partially replaces calls to the objective function. The metamodel approximates the original objective function with a limited precision.

At this point a local metamodel is assumed. Initially the parameters θ, β and s^2 were set by a simple evolution strategy (ES) algorithm. The ES algorithm maximizes the likelihood equation 6.3.13 on page 23 that is based on the maximum likelihood heuristic. The maximum number of rounds is set to one thousand. The setup of θ in this way does not give the performance aimed at. The optimization of the likelihood function by itself is too time expensive. In this state of the algorithm a single optimization would take fifteen days. This even exceeds the required time of an optimization without metamodel support. The delay is mainly caused by the matrix inversion and determinant computations needed for maximizing the likelihood term. Even if the number of rounds is brought back to two hundred and fifty rounds there will be still left an optimization of three to four days. The expected gain of time

Algorithm 4 Robust SMS-EMOA supported by a metamodel

1: $A_0 \leftarrow$ **initializeArchive()** {Initialize archive to support metamodel}
2: $P_0 \leftarrow$ **initializeParentPop**(A_0) {Initialize parent population}
3: $t \leftarrow 0$
4: **repeat**
5: $Y_e \leftarrow \emptyset$ {Array of estimated objective function values}
6: $x_{t+1} \leftarrow$ **generate**(P_t) {Generate one offspring by variation operator}
7: $Perturbations \leftarrow$ **createPerturbations()** {Create perturbations (Monte Carlo Sampling)}
8: **for all** $pert \in Perturbations$ **do**
9: $Y_e \leftarrow Y_e \cup$ **metamodel**(A_t, x_{t+1}, $pert$) {Perturb offspring x_{t+1} with perturbation $pert$ and calculate the corresponding objective function value with a metamodel model. Put the result in Y_e}
10: **end for**
11: $index_w \leftarrow$ **getWorstIndex**(Y_e) {Find in Y_e the worst objective function value and return the index}
12: $o_w \leftarrow$ **createOffspring**(x_{t+1}, $pert[index_w]$) {Create worst offspring from x_{t+1} and $pert[index_w]$}
13: $(o_w, A) \leftarrow$ **calculatePreciseValueAndUpdateArchive**(o_w)
14: $P_{t+1} \leftarrow$ **replace**$_{\triangle S}$($P_t \cup \{o_w\}$) {Select a maximum of μ individuals for the new parent population}
15: $t \leftarrow t + 1$
16: **until** stop criterium reached

by introducing a metamodel into the system is consumed by the ES maximizing the likelihood term. An outline of the likelihood maximization can be found in Algorithm 6 on page 30. The likelihood maximization is actually a minimization of expression 6.3.13 on page 23.

An alternative for setting the parameters is the cross validation tool. With cross validation one point is left out from the sample set and is estimated by the remaining points. This is repeated for every point in the sample set. Let $\mathbf{y}_i$ be the precise objective fucntion value and $\mathbf{y}'_i$ be the estimated value then $\sqrt{\sum_{i=1}^{n}(|\mathbf{y} - \mathbf{y}'|^2)}$ forms the total error, that must be minimized. This method is from a computationally point of view even worse than the former one. Again the matrix operation, here only the inversion is the limiting factor. For each element in the neighbourhood a matrix inversion of a matrix of size neighbourhood minus one is executed. Although a computationally unfeasible method, the predicted θ is a better estimation than the θ produced by the likelihood function. In literature cross validation is mentioned as a more robust error function and the likelihood function to be more qualitative one. In this case the opposite is true. Cross validation performs better on the estimation of θ.

Instead of maximizing the likelihood function or minimizing the cross validation error with an ES, another possibility is to predefine a series $(\theta_1, \theta_2 \dots)$ which start for example at 10E-09 and increase with stepsize ten on a logarithmic scale to one hundred thousand. For each θ_i the likelihood function or cross validation error is calculated and the best θ_i is chosen. Again the likelihood function does not come up with the best possible θ.

Again for cross validation the same answer applies as stated before. Running cross validation slows down the optimization too much and is therefore rejected.

So far the Kriging metamodel is not able to produce reliably quality predictions with an automatically determined parameter θ. Therefore θ is set manually. An disadvantage of manually set parameters is obviously the lack of a possibility to let the model adjust parameter θ during the optimization. If the metamodel gets unstable and produces nonsense output the optimization must be halted and the parameters have to be adjusted before starting a new run. But after practical experience it seems that a metamodel is stable with a constant θ without losing quality in the predictions. A Kriging model with a constant θ will perform computations that are formally equivalent to computations with the RBFN introduced in section 6.4.

For a metamodel that is globally applied, the cross validation can be used to set θ. In this case the estimation of θ will use a fixed amount of time during the initialization of the optimization.

Algorithm 5 calibratingMetamodel(X, Y, I) Kriging - set parameters θ, β and s^2

1: $D \leftarrow$ **calculateDistanceMatrix**(X, I) {Calculate distance matrix D with neighbourhood set X and importance factor I}
2: $\theta \leftarrow$ **ES**(θ) {Maximize likelihood function **maximizeLikelihood** with simple ES with variable θ and return found θ or manually set θ}
3: $C \leftarrow \exp(-\theta \cdot D)$ { Create Matrix C with values between zero and one}
4: $invC \leftarrow C^{-1}$ {Put the inverse of C in $invC$}
5: $\beta \leftarrow$ **calculateBeta**(Y, $invC$, m) {Calculate least squares estimate of β}
6: $s^2 \leftarrow$ **calculateS**(Y, β, $invC$, m) {Calculate least squares estimate of s^2}
7: **return** $\beta, s^2, \theta, C, invC$

Algorithm 6 maximizeLikelihood(θ, D, Y, m)

1: $C \leftarrow \exp(-\theta \cdot D)$ { Create Correlation C with values between zero and one}
2: $invC \leftarrow C^{-1}$ {Put the inverse of C in $invC$}
3: $\beta \leftarrow$ **calculateBeta**(Y, $invC$, m) {Calculate least squares estimate of β}
4: $s^2 \leftarrow$ **calculateS**(Y, β, $invC$, m) {Calculate least squares estimate of s^2}
5: $e \leftarrow m \cdot \log(s) + \log(\det(C))$ {Calculate energy}
6: **return** e

Algorithm 7 minimizeCrossValidationError(θ, D, Y, m, I)

$e \leftarrow 0$ {initialize error to zero}
for all $d_i \in D$ **do**
  $E_{d_i} \leftarrow (D - d_i)$ {E_{d_i} is set D with d excluded}
  $C \leftarrow \exp(-\theta \cdot E_{d_i})$ { Create Correlation C with values between zero and one}
  $invC \leftarrow C^{-1}$ {Put the inverse of C in $invC$}
  $\beta \leftarrow$ **calculateBeta**(Y, $invC$, m) {Calculate least squares estimate of β}
  $s^2 \leftarrow$ **calculateS**(Y, β, $invC$, m) {Calculate least squares estimate of s^2}
  $\lambda \leftarrow$ **calculateLambda**(Y, β, $invC$) {Calculate λ vector}
  $y' \leftarrow$ **estimateFunctionValue**(d_i, β,s^2, $invC$, λ, $invC$, D, I)
  $e \leftarrow e + (y_i - y')^2$
end for
return $\sqrt{e}$

8 Results

8.1 Introduction

The results in this section are a combination of the outcome of variable uncertainty and sensitivity analysis and graphical and statistical representations of test function, metamodeling and VA114 results. Furthermore it shows the development process of the algorithm with the results of the importance factor and the local and global metamodel optimizations.

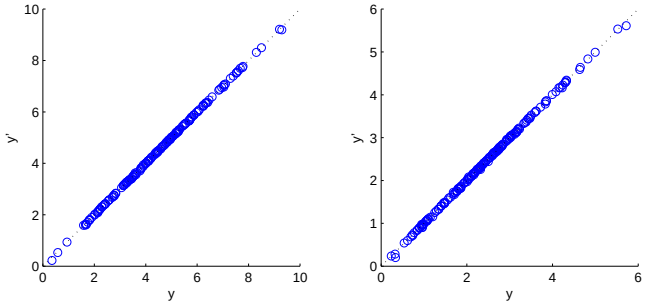
8.2 Uncertainty and sensitivity analysis

The optimization and uncertainty variables enumerated in chapter one are based on uncertainty and sensitivity analysis of seventy-seven variables performed by Hopfe in [Jia07] and [HHPW07]. The selected variables are important in the robust optimization and produce the most effect on the objective function values. An interesting measure of variable importance is given by Partial Correlation Coefficients (PCC). It is based on the concept of correlation and partial correlation. In particular PCC provides a measure of variable importance that tend to exclude the effect of other variables. Figure 8.2.1 shows the result of the PCC for the weighted overheating hours.

8.3 Superspheres2D

The bi-objective superspheres function is a well known test function in the field of optimization. In this thesis the superspheres2D test function is used to validate the functionality of a metamodel in combination with SMS-EMOA. At the moment of writing this theses there are no optimal robust Pareto fronts known from the superspheres2D function. Therefore the validation is done by making a comparison between a summary attainment surface plot obtained from nine robust optimizations and a summary attainment surface plot obtained from nine robust optimizations with a metamodel involved. Since the execution of the superspheres2D function is rather quickly the metamodel slows down the optimization. However, the goal here is to

Metamodel results from test samples - objective y_1 (left) and objective y_2 (right)



Metamodel results from optimization - objective y_1 (left) and objective y_2 (right)

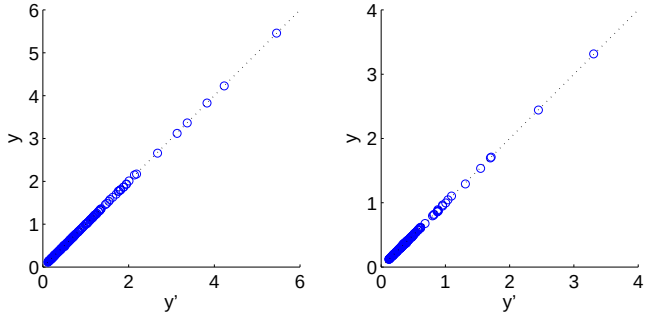


Table 8.3.1: Superspheres2D y-y'diagrams for test and optimization runs.

validate a metamodel assisting a robust optimization and not to improve the speed of the algorithm or the objective function.

8.3.1 Metamodel setup results

The metamodel starts with an archive of two hundred precalculated solutions. A set of two hundred samples is used to manually set the metamodel parameter θ and validate the functioning of the model. The upper two figures in table 8.3.1 show the $y - y'$ diagrams of the two hundred samples for both objectives. The y stands for the precise objective function value and the y' stands for the approximated objective function value. The points follow nicely the diagonal of the diagram, which means that the error is minimal. The two lower figures in table 8.3.1 show the $y-y'$ diagrams of the points collected during an optimization. These points also follow the diagonal.

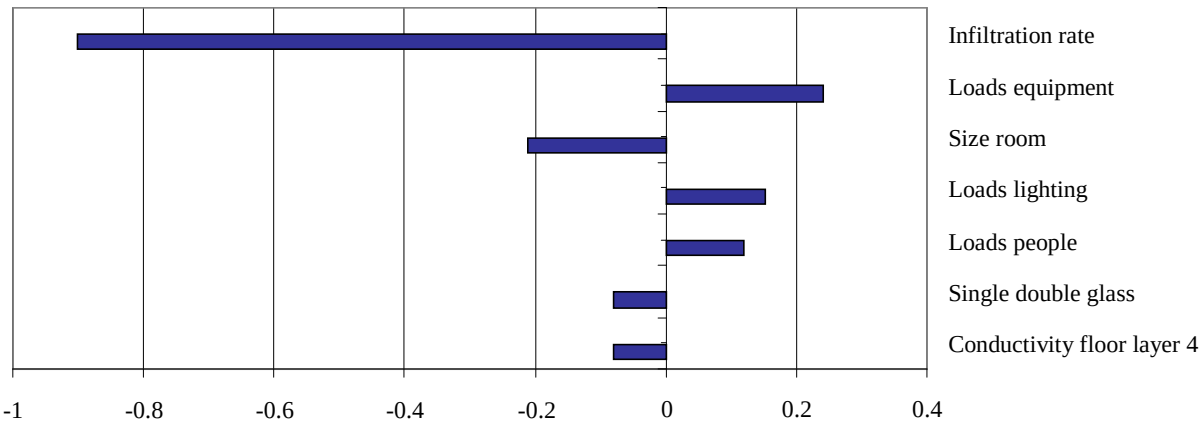
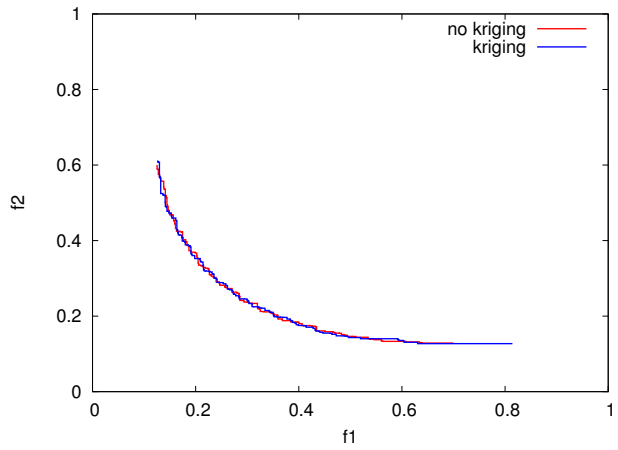


Figure 8.2.1: PCC of weighted overheating hours.

Median SAS plots for the superspheres2D function.



Hypervolume measures of the Pareto front during an optimization.

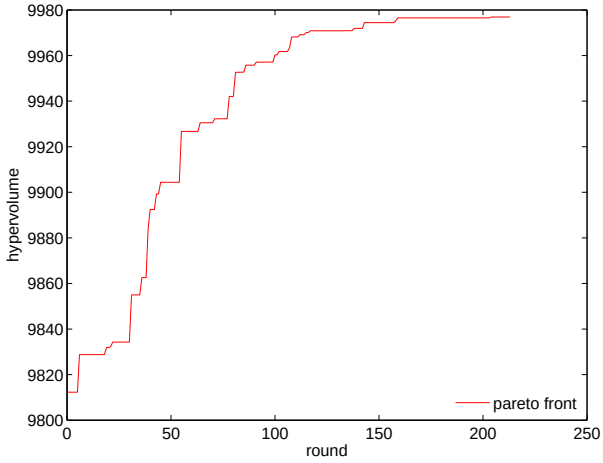


Table 8.3.2: Superspheres2D results: SAS plot (left) and hypervolume measures during optimization (right).

	mean	median	worst	best
no metamodel	9974	9974	9972	9976
metamodel	9975	9974	9973	9977

Table 8.3.3: Hypervolume measures of final Pareto front.

The relative error for objective one lies between 8E-06% and 37% with a median of 0.2% and a mean of 0.01%. For objective two the relative error lies between 5E-04% and 40% with a median of 0.4% and a mean of 0.01%. Both the minimum and maximum errors can be regarded as statistical outliers. All other percentages show a stable performance.

8.3.2 Robust optimization results for superspheres2D

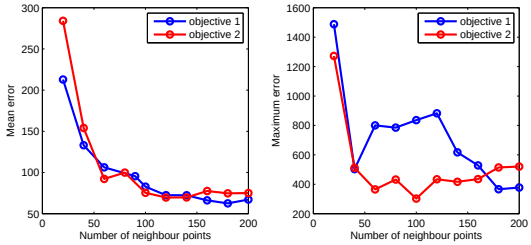
To level out above and below average runs, a total of nine runs are executed. From the nine runs a median summary attainment surface (SAS) plot is calculated. The figures in 8.3.2 give a clear view how the metamodel performs. In the left figure both SAS plots, from the optimization with and without metamodel support, are found and they look equivalent. The metamodel supported run even performs slightly better than the normal run, but that is a minimal observation, which can be the other way around in a next run. The hypervolume measures of the robust optimization of the superspheres2D problem with metamodel support can be found in table 8.3.3.

8.4 VA114 results

8.4.1 Metamodel setup

Initially a local metamodel model was chosen that is supported by an archive of pre-calculated points. For an unknown point a neighbourhood set is created from existing points that are close to the unknown point. After selection of the neighbourhood set,

A comparison of different neighbourhood sizes for a local metamodel



A comparison of different neighbourhood sizes for a global metamodel

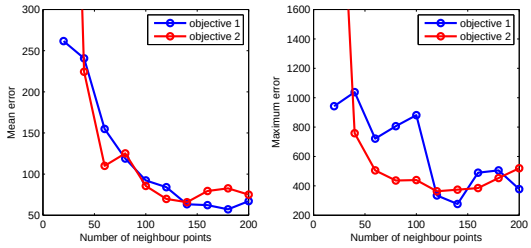


Table 8.4.4: Metamodel errors compared with different neighbourhood size. Left figures show average errors, the right figure shows maximum errors.

the metamodel is calibrated with the set. In this manner the metamodel calibration is repeated for each new point that needs to be estimated, which considerably increases the runtime.

If on the other hand a global metamodel is chosen, calibration is executed only once in a preprocessing phase before the algorithm actually running. There is a possibility to update the model once at the moment a new point is added to the archive under the assumption that a higher amount of neighbouring points increases the quality of the model. Another option is to update the model after a number of predetermined calculated points are added to the archive. In the results of the optimizations supported by a global metamodel the size of the archive is kept constant. This means that there is no recalibration of the metamodel when new objective function values are calculated.

Table 8.4.4 shows the results of test runs for an increasing neighbourhood size with regard to the effect on the average error and maximum error. In the upper figure are presented the results for a local metamodel. The lower figure shows the results for a global metamodel. The figures at the left side show the average error and at the right side the maximum error.

For neighbourhood sizes up to hundred neighbours the local metamodel has better average error measures than the global metamodel. Between one hundred and one hundred and fifty neighbours, the average errors of the local and global metamodel are similar. In the global metamodel the average error of the two objectives diverges stronger for neighbourhood sizes above one hundred and fifty neighbours.

The maximum error in the local metamodel for objective one strongly decreases after one hundred and twenty neighbours, but an increased neighbourhood size also drastically increase the runtime. For the global metamodel a neighbourhood size of one hundred twenty neighbours looks very attractive; The maximum error for both objectives is low compared to the maximum error of one hundred and sixty neighbours and more.

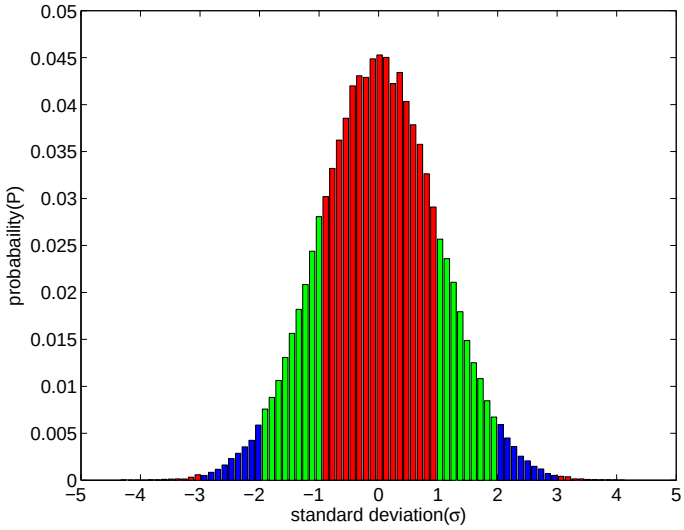


Figure 8.4.2: Random sampling of a gaussian distributed variable.

8.4.2 Importance factors

Initially the design dimension were equally weighted in the correlation matrix of the metamodel. The results of two hundred approximated samples are given in the upper two $y - y'$ diagram in table 8.4.5, where the left figure contains the data for objective one and the right figure for objective two. Objective two already follows nicely the diagonal and shows no exceptional error. Objective one on the other hand is less stable and shows a cloud of points. The cloud is however following the diagonal in a very coarse manner.

The importance factor is a simple weight factor for each dimension in the search space. It is calculated as follows: for n_f objectives and d design variables, take for each variable the minimum and maximum value in their range, while the remaining variables are attributed their mean values. For the variables with gaussian distributed uncertainties the ranges are defined by their mean μ values plus/minus three times the standard deviation σ. This covers at least 99% of the gaussian distribution. Figure 8.4.2 shows a random sampling of a gaussian distributed variable. The probability $P(x)$ that a sample x lies in the range of $[-3\sigma, 3\sigma]$ is approximately 99.7%.

Now, calculate for the created input vectors the objective function values and put these in $Y_1, \ldots Y_{n_f}$, where $Y_i = y_i, \ldots, y_d$ and

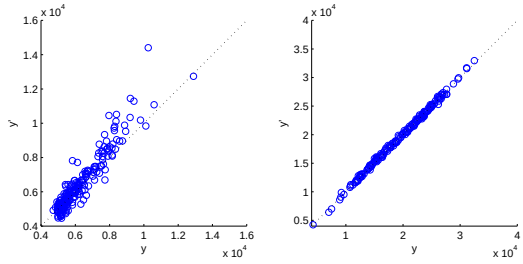
$$\begin{aligned} y_i = f(\vec{x}_{mean} + \vec{e_i}\frac{x_{i_{max}} - x_{i_{min}}}{2}, \vec{\alpha}_{mean} + 3\sigma_i\vec{e_i}) \\ - f(\vec{x}_{mean} - \vec{e_i}\frac{x_{i_{max}} - x_{i_{min}}}{2}, \vec{\alpha}_{mean} - 3\sigma_i\vec{e_i}) \end{aligned} \tag{8.4.1}$$

where $e_i \in \mathbb{R}^d$ is the i-th unit vector, and $\vec{\alpha}$ is the mean vector for the environmental variables (see table 2.4.1).

The importance factor c_j for each design dimension is

$$c_j^{(k)} = \frac{Y_k^{(j)}}{\sum_{i=1}^{n_f} Y_k} \text{ where } j = 1, \ldots, d \text{ and } k = 1, \ldots, n_f \tag{8.4.2}$$

Metamodel results of sample set without importance factor



Metamodel results of sample set with importance factor activated

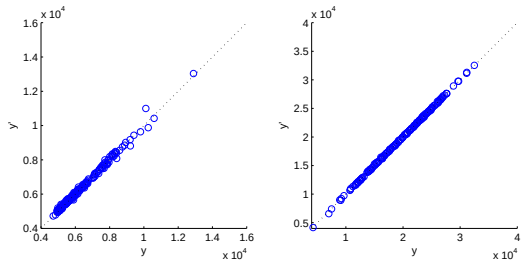


Table 8.4.5: Metamodel results regarding the importance factor.

After introduction of the importance factor in the metamodel correlation matrix, the test results show much more stable approximations of objective one and approximations of objective two are improved a little. The results can be found in the lower two diagrams of table 8.4.5

8.5 Robust optimization results for VA114

Below are all optimization results regarding VA114. A division is made between optimizations from a local metamodel supported algorithm and optimizations from a global metamodel supported algorithm. The main difference between these approaches is the frequency of calls to the metamodel and the update of the metamodel with new points in the archive.

To give an impression of the performance of the robust optimization the following is carried out. From the initial population of a run with twenty parents a point is taken out. Around these point fifty perturbations are randomly created and calculated by the objective function. The differences between the objective function values of the point and the objective function values of the perturbations is summed and divided by the number of perturbations. This results in an average deviation of the fifty calculated perturbations around the original point. This is repeated for all points in the populations.

The same method is applied to all points in the final population after four hundred rounds of optimization starting with the initial population. The results of the average deviations are found in table 8.5.8. The left part of the table shows the average deviations for the initial population, the right side of the table contains the average deviations for the final population. For objective one the solutions in the initial population show a better average robustness than the solutions in the final populations, in other words, the robustness has been declined. It is objective two that profits from the robust optimization. The initial population has an average deviation of approximately 9135 WmK^{-1}. This value is improved to 5313 WmK^{-1} in the final population.

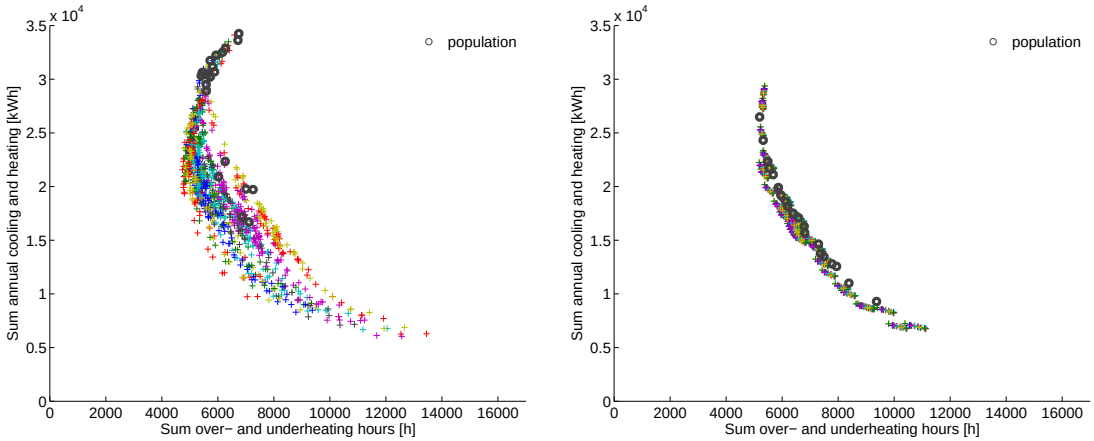


Table 8.5.6: Scatter plot of the initial population with its perturbations (left) and the final population and its perturbations (right) for a worst case scenario.

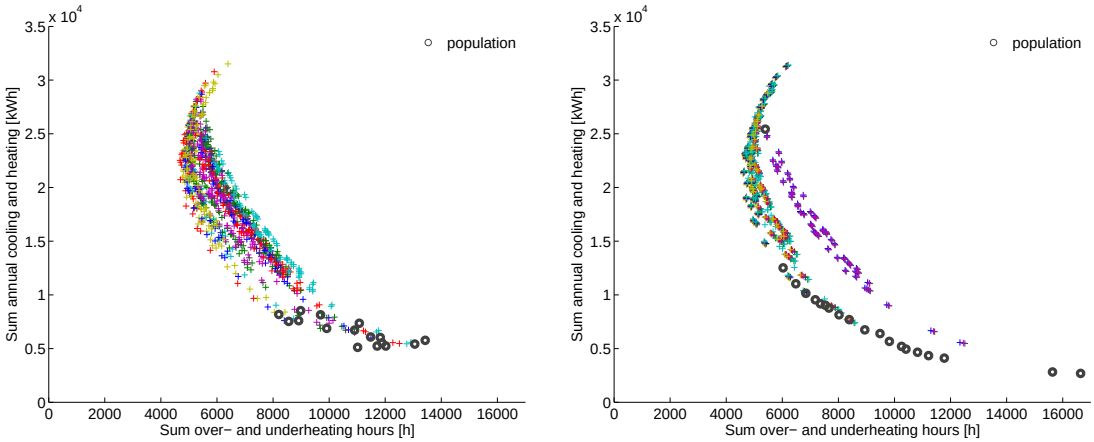


Table 8.5.7: Scatter plot of the initial population with its perturbations (left) and the final population with its perturbations (right) for a best case scenario.

Because the problem is two-dimensional it is the combination of the two objectives that is supposed to be robust. This is better explained with figures that show the objective functions values of the initial and final population together with the objective function values of the perturbations. The left figure in table 8.5.6 involves the initial population and perturbations. The figure clearly shows the instability caused by the uncertainty in the design variables. It is not always the case that perturbed objection function values are worse compared to their original unperturbed objective function values, but it is impossible to guarantee the output values in a small range of the objective space. The right figure in table 8.5.6 shows the final population and their perturbations. This is certainly much more robuster than it was in the initial situation.

In the robust optimization in this thesis a worst case scenario is chosen, what means that the worst solution out of two hundred and one perturbations is compared with the current parent population. In table 8.5.7 is demonstrated how a best scenario case performs in a robust optimization. Again, the initial population and final population are given. The right figure clearly shows that after four hundred optimization rounds the Pareto front is better than in the worst case, but with much more instability involved, caused by the uncertainty in the design variables.

Round 0				Round 400			
obj 1	obj 2	obj 1	obj 2	obj 1	obj 2	obj 1	obj 2
560.84	10125.76	1024.68	3706.50	2670.56	7860.58	1598.62	5533.64
687.58	10391.32	1164.34	11039.58	1279.66	6187.92	867.04	3802.62
1482.88	12356.08	1215.34	4753.06	1368.02	5138.06	1956.68	6706.02
836.48	10444.48	1094.50	5100.72	1177.64	4242.94	873.46	3972.90
1257.26	5828.44	669.88	10944.28	979.18	3795.80	913.86	4485.06
666.12	12387.30	1058.62	10580.40	849.24	4086.56	1499.30	7904.88
760.98	12049.24	1348.74	11143.90	1128.34	5571.32	1492.46	9646.36
658.98	11175.20	605.38	9720.30	1520.82	5461.76	1084.40	5242.50
733.92	10642.38	1084.68	3704.44	1049.30	4195.58	864.56	3896.84
1468.62	5128.00	810.48	11483.22	931.50	3747.48	1257.54	4788.80
Average: obj1: 959.515 obj2: 9135.23				Average: obj1: 1268.109 obj2: 5313.381			

Table 8.5.8: Robustness comparison of an initial population and a final population of an optimization.

8.5.1 Local metamodels results

The setup of the experiments is as follows. The population size is set to ten, fifteen and twenty parents. Each experiment is repeated with mutation strength adaptation on and off. The mutation strength is initially set to 0.05. Each experiment is repeated nine times to obtain a median summary attainment surface plot. The neighbourhood size is one hundred and twenty neighbours.

Table 8.5.9 contains the results of the robust optimizations supported by a local metamodel. The two figures in the top of the table and the one in the middle to the left show the summary attainment surface (SAS) plots for respectively the parent populations of size ten, fifteen and twenty, with mutation strength adaptation on and off. If adaptation is on, the SAS plots show for all three populations a stronger extremal solution at one side of the graph. From Table 8.5.10 can be noticed that these stronger extremal solutions do not lead to higher hypervolume measures.

The different parent populations that have mutation strength adaptation on show all improved best runs compared to the runs that have mutation strength adaptation off. The median, however, is worse for each population size when mutation strength adaptation is on. The adaptation of the mutation strength can result in finding a better optimum for one of the nine runs, but does not guarantee this for all runs.

The last two figures in table 8.5.9 show that for both adaptation on and off an increasing parent population leads to an increasing hypervolume measure for the mean, median, and best cases.

8.5.2 Global metamodel results

The setup of the experiments is as follows. The population size is set to ten, fifteen and twenty parents. Each experiment is repeated with mutation strength adaptation on and off. The mutation strength is initially set to 0.05. Each experiment is repeated nine times to obtain a median summary attainment surface plot. The runs are repeated with a neighbourhood size of one hundred and twenty neighbours and

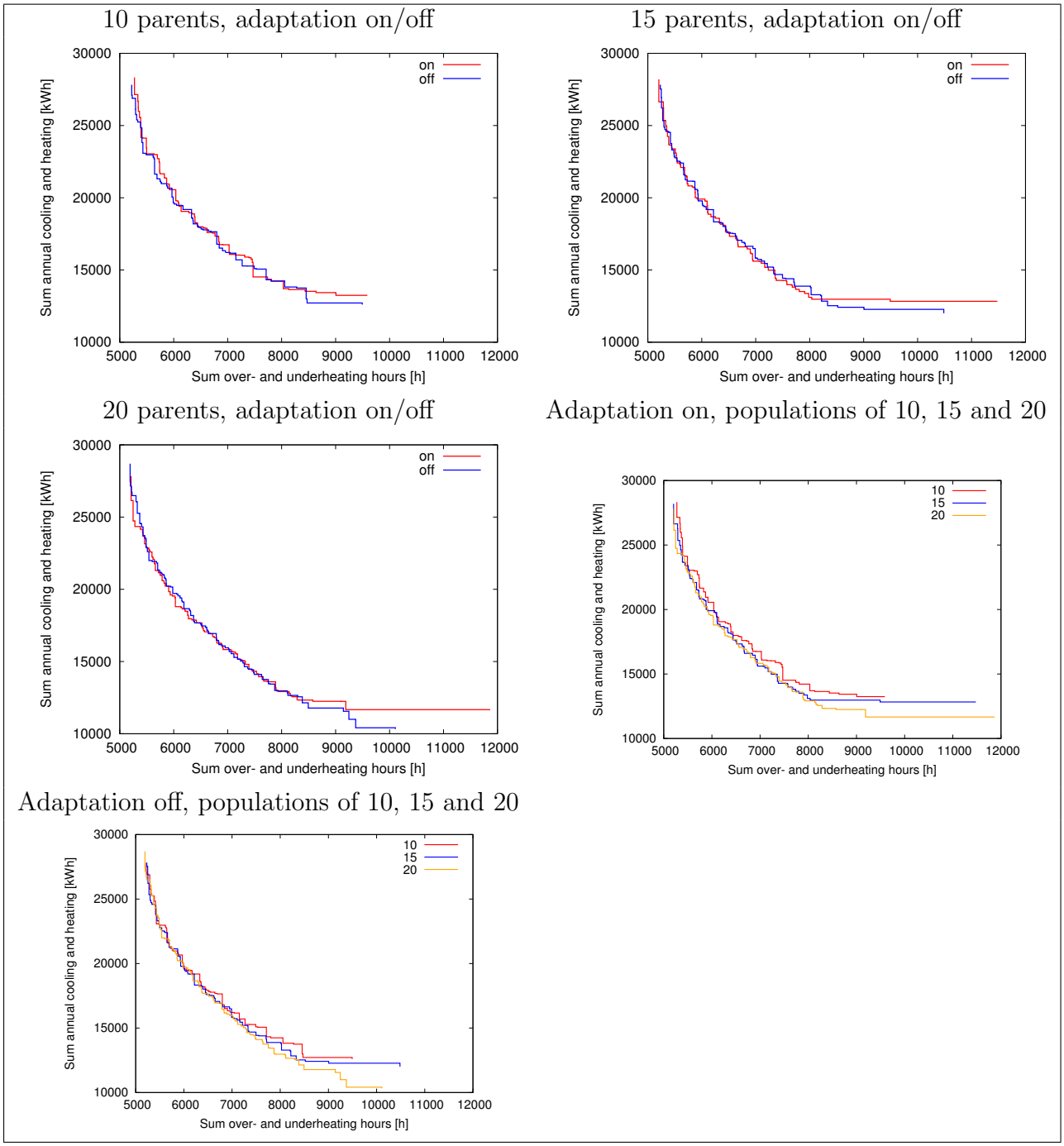


Table 8.5.9: Summary attainment surface plots of the optimizations supported with a local metamodel with a neighbourhood size of 120.

run	parents	adaptation	mean	median	worst	best
run 1	10	no	8274513365	8280420688	8077367940	8431328567
run 2	15	no	8369735883	8318621402	8173352253	8569027800
run 3	20	no	8456456772	8478629931	8152189845	8698856757
run 4	10	yes	8241348177	8195346348	8133439409	8473882869
run 5	15	yes	8331754803	8249109739	8159335318	8625906322
run 6	20	yes	8435536355	8348664406	8245511242	8847065079

Table 8.5.10: Hypervolume measures for optimizations with a local metamodel with a neighbourhood size of 120.

two hundred neighbours.

Tables 8.5.11 and 8.5.12 contain the results of the robust optimizations supported by a global metamodel with a neighbourhood size of one hundred and twenty neighbours and tables 8.5.13 and 8.5.14 contain the results for a neighbourhood size of two hundred neighbours. In contrary to the local metamodel the increasing population size does not automatically lead to an increased hypervolume measure for the median and best statistics. Also in this case the adaptation of the mutation strength does not result in an advantage for the optimization. However, the adaptation works a little bit better than in the local metamodel. In general all outcomes are in the same ranges.

If a neighbourhood size of two hundred neighbours is chosen the performance of the optimization drops significantly. This may be caused by the fact that a part of the two hundred neighbours do not contribute to the metamodel estimation. These neighbours are too far away to affect the estimation in advantageous way.

If we look at the case of one hundred and twenty neighbours, the results are quite good compared to the local metamodel results. Besides, the global metamodel is less time consuming.

8.5.3 Pure global metamodel results

The setup of the experiments is as follows. The population size is set to ten, fifteen and twenty parents. Each experiment is repeated with mutation strength adaptation on and off. The mutation strength is initially set to 0.05. Each experiment is repeated nine times to obtain a median summary attainment surface plot. The runs are repeated with a neighbourhood size of one hundred and twenty neighbours and two hundred neighbours. The objective function is called only μ times after the optimization is finished. The precise values are not calculated by the objective function for the improved solutions found during the optimization. This saves extra runtime compared to the global metamodel supported optimization.

For a neighbourhood of one hundred and twenty neighbours the pure global metamodel optimization is performing very well. The results are comparable to the results of the local metamodel supported optimization. In tables 8.5.15 and 8.5.16 the results can be found.

A neighbourhood size of two hundred neighbours shows the same performance drop that is present in the global metamodel supported optimization with the same neighbourhood size. The results can be found in 8.5.17 and 8.5.18.

Although the (pure) global metamodel optimizations run faster than the local metamodel optimizations, the highest values for the mean, median, worst and best are all found in the local metamodel supported statistics. If time is not an issue and the best solutions are required to be found, local metamodel supported optimization is the best option, otherwise global metamodel supported optimization satisfies.

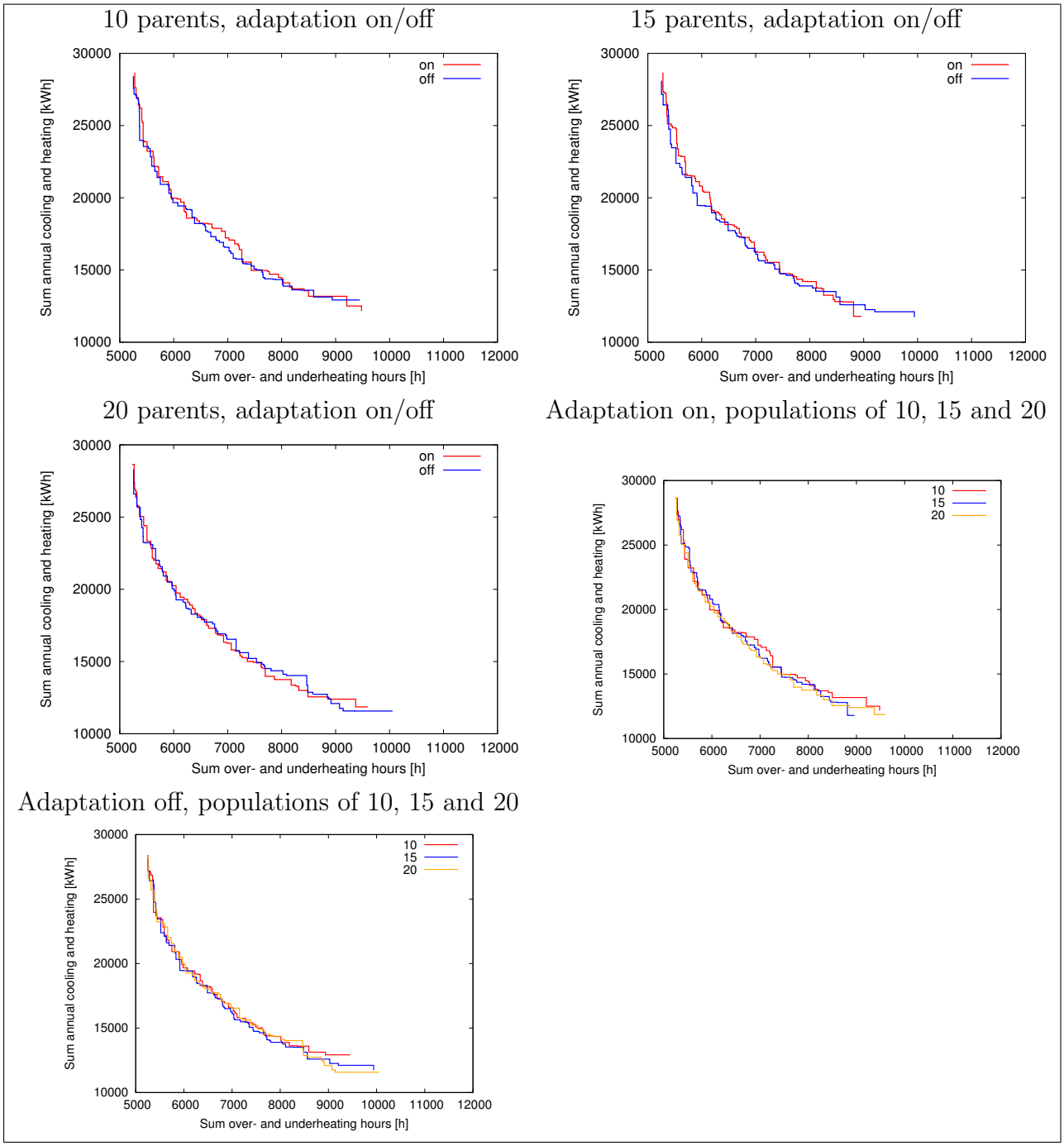


Table 8.5.11: Summary attainment surface plots of the optimizations supported with a global metamodel with a neighbourhood size of 120.

run	parents	adaptation	mean	median	worst	best
run 1	10	no	8267283643	8234988511	8128113139	8527693978
run 2	15	no	8346278895	8356574404	8184536829	8560724197
run 3	20	no	8355317780	8362835910	8171392173	8487982969
run 4	10	yes	8268245999	8297779350	8119494063	8419465921
run 5	15	yes	8294989588	8339822532	8065659995	8411623504
run 6	20	yes	8324085196	8332673023	8162834867	8426057544

Table 8.5.12: Hypervolume measures for optimizations with a global metamodel and a neighbourhood size of 120.

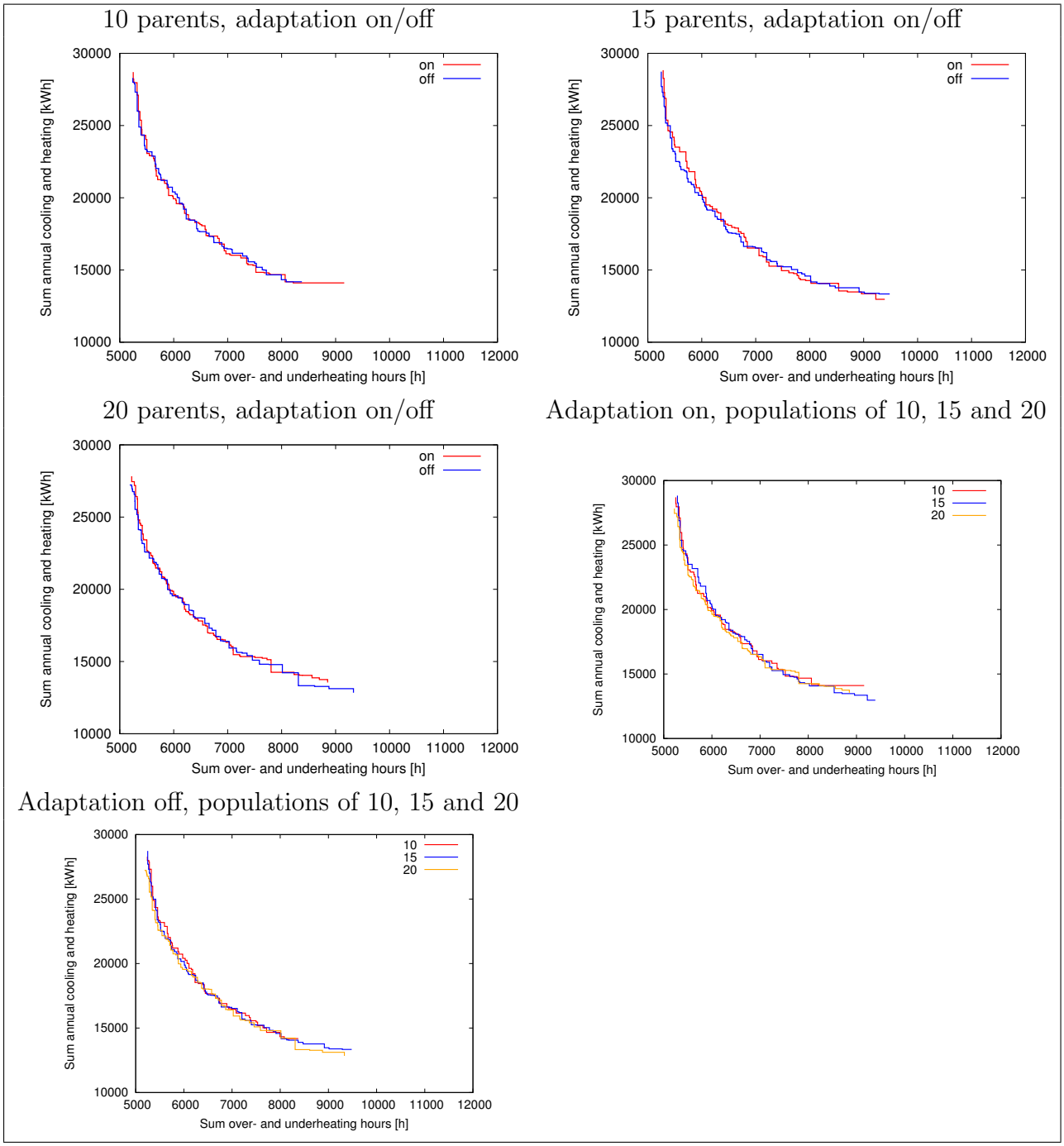


Table 8.5.13: Summary attainment surface plots of the optimizations supported with a global metamodel with a neighbourhood size of 200.

run	parents	adaptation	mean	median	worst	best
run 1	10	no	8150330943	8117725647	8043274858	8310061420
run 2	15	no	8198463645	8198081779	8078832413	8302510977
run 3	20	no	8220953268	8243823349	7993938752	8351264678
run 4	10	yes	8155087516	8120635086	8002805790	8429516989
run 5	15	yes	8219794205	8225853963	8114729396	8341335509
run 6	20	yes	8178969461	8179590114	8098361236	8283853156

Table 8.5.14: Hypervolume measures for optimizations with a global metamodel and a neighbourhood size of 200.

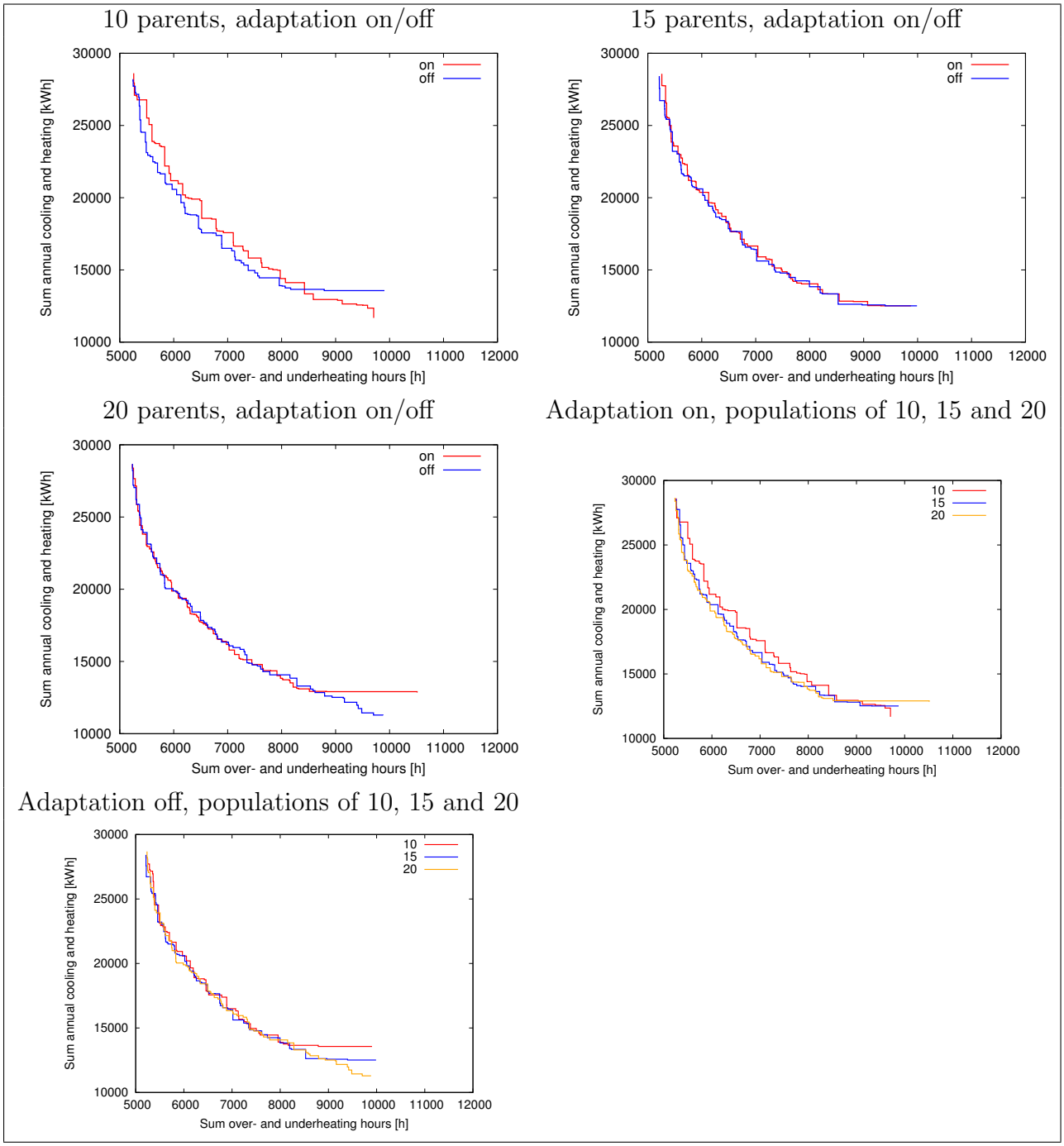


Table 8.5.15: Summary attainment surface plots of the optimizations supported with a pure global metamodel with a neighbourhood size of 120 and μ calls to the objective function.

run	parents	adaptation	mean	median	worst	best
run 1	10	no	8222476857	8180687385	8093928156	8487061208
run 2	15	no	8277409446	8271901922	8051140824	8460429972
run 3	20	no	8352302898	8401087950	8211450915	8458035144
run 4	10	yes	8303217390	8337438412	8197624973	8415298946
run 5	15	yes	8274044267	8265246924	8036161236	8468027215
run 6	20	yes	8316819073	8237597956	8144567116	8514178093

Table 8.5.16: Hypervolume measures for optimizations with a pure global metamodel with a neighbourhood size of 120 and μ calls to the objective function.

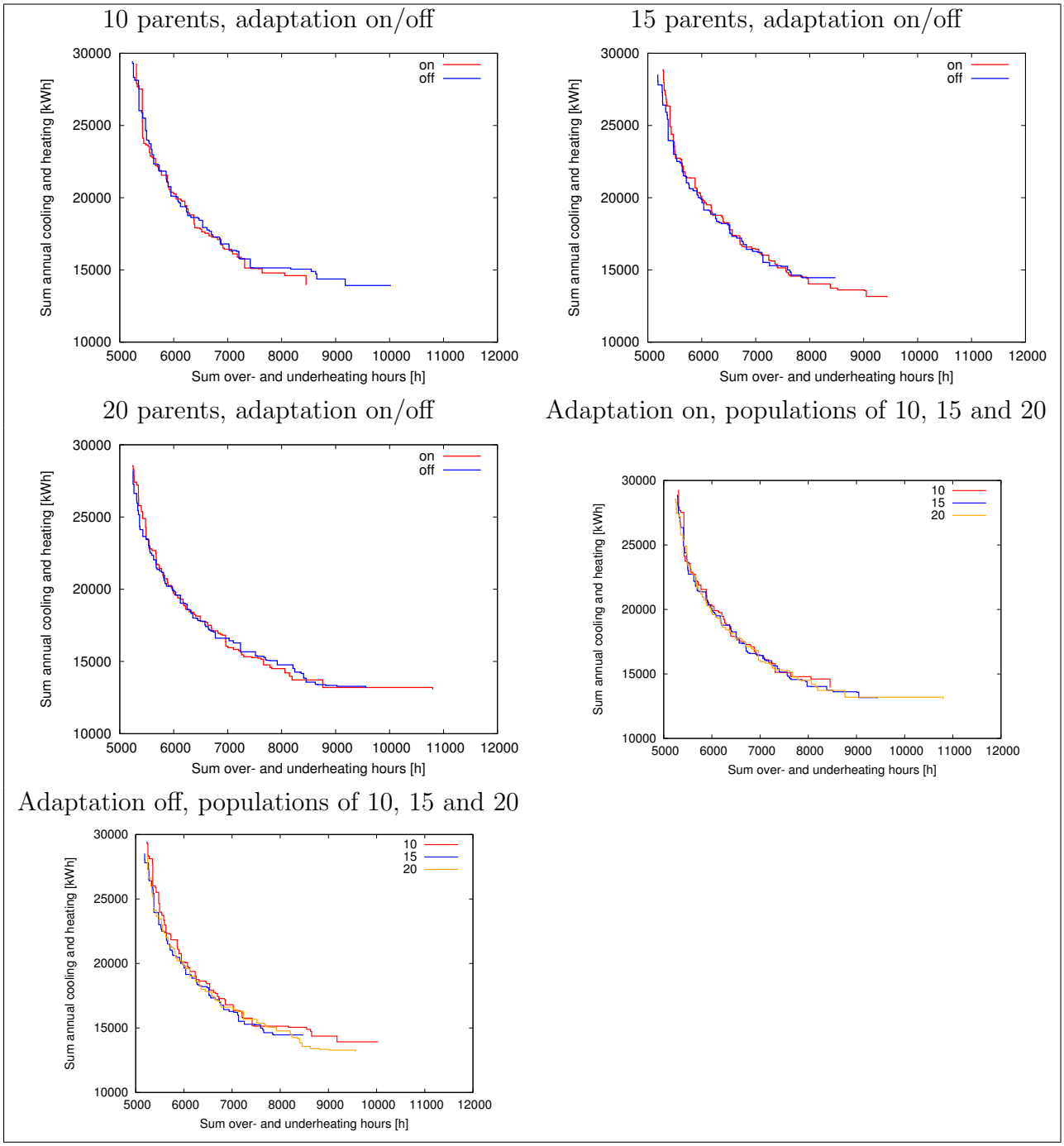


Table 8.5.17: Summary attainment surface plots of the optimizations supported with a pure global metamodel with a neighbourhood size of 200 and μ calls to the objective function.

run	parents	adaptation	mean	median	worst	best
run 1	10	no	8130750432	8153252617	7899730660	8416082756
run 2	15	no	8123799180	8113353784	8056374855	8298518214
run 3	20	no	8231560585	8222782825	8056236004	8420255441
run 4	10	yes	8116492084	8123819191	8037014491	8234980443
run 5	15	yes	8218458400	8210634985	8094757530	8334026605
run 6	20	yes	8237891802	8225151741	8022393544	8562180108

Table 8.5.18: Hypervolume measures for optimizations with a pure global metamodel with a neighbourhood size of 200 and μ calls to the objective function.

9 Conclusion and outlook

The hypothesis in section 2.6 was correct. It is possible to support a multi-objective algorithm with a metamodel and to optimize the problem to a robust Pareto front.

Estimating the parameters of the metamodel with a maximum likelihood function does not result in the best possible θ and slows down the optimization. With cross validation it is possible to estimate the metamodel parameters, but it is too time consuming. A manually set parameter θ worked out best. The importance factor makes a clear difference for the quality of the estimation of one of the objective function values.

A local metamodel supported algorithm is working within the time constraint of one night of runtime. A (pure) global metamodel performs almost like the local metamodel and saves even more time. Kriging is a suitable metamodel for the problem in this thesis. A disadvantage of Kriging is the limitation in the number of design variables (1-20) at which the metamodel still does quality estimations. This problem can be circumvented by reducing the number of parameters with the help of uncertainty and sensitivity analysis to identify the most sensitive variables.

A comparison to another multi-objective optimization tool, like NSGA-II, was not carried out. That is the reason why it is not possible to decide whether the choice for SMS-EMOA was the best choice for this moment. Future work can be done regarding parameter settings and neighbourhood sizes. A local search method to fine tune the convergence to a local or global optimum is another task for the future.

An idea is to compare a metamodel Assisted EA to an EA, that uses parallel computing. With the increasing number of cores in processors these days this is not that expensive anymore.

It would be of interest to see how the outcome of this thesis can be applied in practice. And backwards, how practical experience can lead to improvements of robust optimizations algorithms for building design.

A Multi-objective test problem

The superspheres2D problem is described as

$$\begin{aligned} f_1(\mathbf{x}) &= \sum_{i=1}^{d} (x_i)^2 \\ f_2(\mathbf{x}) &= \sum_{i=1}^{d} (x_i - 1)^2 \end{aligned} \tag{A.0.1}$$

The problem has a convex Pareto front with extrema $y_1 = 0$, $y_2 = 1$ and $y_1 = 1$, $y_2 = 0$. The known Pareto front is $f_2 = \sqrt{(1 - f_1)^2}$.

References

[BS07] H.-G. Beyer and B. Sendhoff. Robust optimization - A comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33-34):3190–3218, July 2007.

[DG06] K. Deb and H. Gupta. Introducing robustness in multi-objective optimization. *Evolutionary Computation*, Vol. 14(4):463–494, December 2006.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation (IEEE-TEC)*, 6(2):182–197, April 2002.

[EBN05] M.T.M. Emmerich, N. Beume, and B. Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In C. Coello Coello et al., editor, *Proc. Evolutionary Multi-Criterion Optimization: Third Int'l Conference (EMO 2005)*, volume 3410/2005, pages 62–76. Springer, Berlin, March 2005.

[ED07] M.T.M. Emmerich and A.H. Deutz. Test problems based on lamé superspheres. In *Evolutionary Multiobjective Optimization 2007 (EMO2007)*, volume 4403/2007, pages 922–936. Springer, May 2007.

[EHM+08] M.T.M. Emmerich, C.J. Hopfe, R. Marijt, J.L.M. Hensen, C. Struck, and P. Stoelinga. Evaluating optimization methodologies for future integration in building performance tools. In Ian Parmee, editor, *Proceedings of the 8th Int. Conf. on Adaptive Computing in Design and Manufacture (ACDM), 29 April - 1 May, Bristol*, April 2008.

[Emm04] M.T.M. Emmerich. *Single- and Multi-objective Evolutionary Design Optimization assisted by Gaussian Random Field Metamodels.* PhD thesis, University of Dortmund, Department of Computer Science, 2004.

[Gia02] K.C. Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences*, 38(1):43–76, January 2002.

[HHPW07] C.J. Hopfe, J.L.M. Hensen, W. Plokker, and A.J.T.M. Wijsman. Model uncertainty and sensitivity analysis for thermal comfort prediction. In *Proceedings of the 12th Symp for Building Physics, 19-31 March, Technische Universitat Dresden.*, pages 103–112, 2007.

[Hop09] C.J. Hopfe. *Uncertainty and sensitivity analysis in building performance simulation for decision support and design optimization.* PhD thesis, Univertity of Eindhoven, Department of Building Engineering, 2009.

[HSHB06] C.J. Hopfe, C. Struck, J. Hensen, and M. Böhms. Adapting engineering design approaches to building design- potential benefits. In *proceedings of* 6^{th} *Int. Postgraduate Research Conf. in the Built and Human Environment, 6-7 April, Technische Universiteit Delft, BuHu, University of Salford*, pages 369–378, 2006.

[Jia07] Yi Jiang, editor. *Uncertainty and sensitivity analysis for detailed design support*, 2007.

[KC99] J. Knowles and D. Corne. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In P.J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 98–105. IEEE Press, June-September 1999.

[MCP02] J.J. Michalek, R. Choudhary, and P.Y. Papalambros. Architectural layout design optimization. *Engineering Optimization*, 34(5):461–484(24), January 2002.

[PH02] I.C. Parmee and P. Hajela. *Optimization in Industry.* The Chartered Institute Of Purchasing And Supply, 2002.

[SWMW00] J. Sacks, W. J. Welch, W. J. Mitchell, and H.-P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 2000.

[VAB09] VABI. Handleiding, 2009. http://www.vabi.nl/downloads/handleidingen/.

[WZAB04] J. Wright, Y. Zhang, P.P. Angelov, and R.A. Buswell. Building system design synthesis and optimization. In *Final Report to ASHRAE on Research Project 1049-RP*, 2004.

[ZLT01] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK) Department of Electrical Engineering Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

[ZT98] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In A. E. Eiben et al, editor, *Parallel Problem Solving from Nature - PPSN V, Amsterdam*, pages 292–301, Berlin, 1998. Springer.