Efficient approximation of black-box functions and Pareto sets

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit van Tilburg, op gezag van de rector magnificus, prof.dr. Ph. Eijlander, in het openbaar te verdedigen ten overstaan van een door het college voor promoties aangewezen commissie in de aula van de Universiteit op vrijdag 20 november 2009 om 14.15 uur door

Gijs Rennen

geboren op 17 juli 1982 te IJsselstein.

PROMOTORES: prof.dr. ir. D. den Hertog
prof.dr. ir. E.R. van Dam

THOMAS STIELTJES INSTITUTE
FOR MATHEMATICS

Science can purify religion from error and superstition;
religion can purify science from idolatry and false absolutes.
Each can draw the other into a wider world,
a world in which both can flourish.

(John Paul II, *Letter to the Reverend George V. Coyne, S.J.*)

Acknowledgements

> The hardest arithmetic to master is that which enables us to count our blessings.
>
> (Eric Hoffer, *Reflections on the Human Condition*)

When I look back on the three years of my Ph.D. career, I consider myself to be truly blessed. Not only blessed with interesting and fruitful research of which this thesis is the final result, but also blessed in many other respects. Most of all, I feel blessed with the guidance, help, and company of many people who made it such a formative, valuable, and wonderful time. Therefore, I would like to use this opportunity to express my gratitude to all of them.

I would like to begin by thanking Dick den Hertog and Edwin van Dam for their support as my supervisors. After being my supervisor for both my Bachelor and Master thesis, Dick offered me the possibility and encouraged me to continue my research as an M.Phil and Ph.D. student. I am still very glad that I accepted this offer and that Edwin agreed to become my second supervisor. In terms of my research subjects, Dick and Edwin gave me a lot of freedom, which is reflected by the variety of subjects discussed in this thesis. I am grateful for the pleasant and stimulating manner in which we could discuss my progress at our meetings. Their guidance, help, encouragement, and enthusiasm have been indispensable factors in the successful progress and completion of my Ph.D. career.

Besides Dick and Edwin, several other people have also directly contributed to the research in this thesis. My first paper is based on the results of my internship at the Center for Quantitative Methods (CQM) where I was supervised by Peter Stehouwer and Erwin Stinstra. I am still thankful for their support and for the opportunity to write a paper on the obtained results together with Erwin Stinstra and Geert Teeuwen. Secondly, I owe many thanks to Bart Husslage, who finished his Ph.D. career at Tilburg University one year after I started. His research on space-filling (nested) Latin hypercube designs provided a solid basis for much of my research on this topic. Furthermore, I thank him for the pleasant cooperation on several papers included in this thesis. Thirdly, I would like to thank Aswin Hoffman, clinical physicist at the Department of Radiation

Oncology at Radboud University Nijmegen Medical Centre. Aswin introduced me to the very interesting and relevant subject of multi-objective IMRT optimization. I'm very thankful for his help in learning to understand the background, terminology and characteristics of this problem. Furthermore, I want to thank him for arranging several occasions where I could meet people involved in both the practical and research side of IMRT optimization.

When doing research, it is also important to have others researchers critically and expertly evaluate one's work. Therefore, I am very thankful that David Craft, Annie Cuyt, Jack Kleijnen, Dolf Talman, and Vassili Toropov are willing to join Dick and Edwin in my thesis committee. Their expertise on the different topics covered in this thesis resulted in valuable comments and suggestions. I would also like to thank Dolf for critically proofreading two of the papers included in this thesis.

Just as a plant grows better in good soil, so does research 'grow' better in a stimulating and pleasant working environment. Therefore, I want to thank all my colleagues in the Department of Econometrics and Operations Research, who created just such an environment. I will never forget the many puns at the lunch table, the cryptic crosswords during the coffee breaks, and the ingenious door-switching joke. Especially, I would like to thank the 'sub-department' consisting of Cristian, Edwin, Elleke, Gerwald, John, Lisanne, Marieke, Mark, Marloes, Romeo, Ruud, and Salima. Also after working hours, they have been great company during game, sinterklaas and bungee-soccer evenings.

Besides the contacts with my colleagues, I am also very grateful for many other people whom I have met during my Ph.D. years. Especially, I would like to thank all the great people that I've learned to know (better) via Student Alpha, Connect, Ichthus and the Evangelical Baptist Church in Tilburg. Meeting Christians with different backgrounds, personalities, and nationalities has taught me many lessons, which are more valuable to me than the results in this thesis.

However, the persons who have taught me the most valuable lessons in live are my parents, Jan and Merence. I feel truly blessed with the values they taught me and the example they have been and still are. I also would like to thank Jan, Merence, and Els very much for their support in both my working and personal life during my Ph.D. years. Furthermore, I would like to thank Els and Kees for agreeing to be paranymphs at my PhD defence.

Finally and most of all, I want to give thanks to Him from whom all blessings flow. He who created heaven and earth. He who knows me completely and still loves me. I want to thank Him for the talents He gave me and the opportunities to use them. For bringing many wonderful people on my path. And for blessing me in more ways than can be counted by any arithmetic.

Contents

Chapter 1

Introduction

> The most exciting phrase to hear in science, the one that heralds new discoveries, is not "Eureka!" ("I've found it!") but "That's funny..."
>
> (Isaac Asimov)

In recent decades, computers have been able to solve increasingly complex problems. Despite the increased computational power, certain problems still take a considerable amount of time to solve. Optimization problems involving complex deterministic simulation models or dealing with multiple objectives are two classes of problems that can be very time-consuming to solve. For these problems, determining optimal or good solutions in a reasonable amount of time is a complicated task. Therefore, the central topic of this thesis is to develop and improve methods for dealing efficiently with these time-consuming optimization problems. Although the techniques to deal with these two classes of optimization problems are different, they do share one important characteristic. In both cases, the used approach involves determining an approximation of a set or function based on data points that are time-consuming to calculate.

In the case of complex simulation models, we often approximate the black-box function describing the relation between the input- and output-variables of the simulation model. A black-box function is defined as a mathematical function of which no explicit description is known, but which can be evaluated to obtain output-values for feasible input-values. We approximate this black-box function by a so-called metamodel. This metamodel does have an explicit description and can be used to optimize the output or to provide insight into the input-output relation. To build this metamodel, we need a set of data points that contain the value of the output variable for a certain setting of the input-variables. Because obtaining each data point requires an evaluation of the complex simulation model, these data points are time-consuming to calculate.

When dealing with multi-objective optimization problems, we can try to approximate the unknown Pareto set. This set contains all Pareto optimal solutions, i.e., solutions for which it is not possible to improve one objective without deteriorating another. Selecting a solution that does not satisfy this definition is sub-optimal as we can improve one or more objectives at no costs for the other objectives. Therefore, determining the Pareto set is often an important part of solving a multi-objective optimization problem. As calculating a Pareto optimal solution requires formulating and solving a time-consuming optimization problem, determining the complete Pareto set is generally infeasible. Instead an approximation is determined that can be used by the decision maker to select a desirable solution.

As the calculation of a data point is time-consuming in both problems, we want to determine an accurate metamodel or approximation of the Pareto set using as few points as possible. In this thesis, our main focus is, therefore, on efficiently selecting which simulations to run or optimizations to perform. By selecting these simulations and optimizations in an efficient way, we thus aim to reduce the number of points necessary to obtain an accurate solution.

In Sections 1.1 and 1.2, we treat the problems of simulation-based optimization and multi-objective optimization in more detail. Methods for building a metamodel or an approximation of the Pareto set are discussed in Section 1.3. Particularly, we describe the metamodelling approach described in Den Hertog and Stehouwer (2002) and the basic steps of sandwich algorithms used for approximating convex Pareto sets. Furthermore, we also discuss the similarities and differences between these two approaches. Section 1.4 gives a summary of the contributions of this thesis. Finally, an overview of the papers included in this thesis is provided in Section 1.5.

1.1 Simulation-based optimization

With the advance of computer technology and simulation techniques, deterministic computer simulations have found many applications in the recent decades. Applications described in literature include supply chain optimization (Kleijnen (2005), Zheng et al. (2008)), medical radiation (Campos (2006)), ecological simulation (Kleijnen et al. (1992)), dynamical systems (Fang et al. (2000)), and electrical, chemical, automotive and aerospace engineering (see, e.g., Simpson et al. (2001), Chen et al. (2006), Oden et al. (2006), and Kleijnen (2008)). Many of these applications concern the design of products, systems, and processes. Designers use the simulation models to optimize certain quantifiable characteristics of the design. In the case of product design, physical experiments were used in the past for these optimizations. One could think, for instance, of the crash-tests performed with prototypes of cars to optimize their road safety. However, building differ-

ent prototypes and performing the tests is a time-consuming and expensive process. The increased complexity of new products and the reduced time-to-market further raised the necessity for an alternative approach. Therefore, physical prototyping is nowadays often replaced by virtual prototyping; i.e., computer simulations are used instead of physical tests to determine certain properties of a design. This means, for example, that the consequences of a car crash are no longer measured by crashing real cars, but by simulating the forces exerted on a car during a crash using computer models. Although these simulations are faster than physical experiments, they can nevertheless require several minutes or even hours to evaluate. An increase in computational power could reduce these calculation times. However as the simulation models also tend to become more complex, simulation models with large calculation times are likely to continue to exist.

An often used method to deal with time-consuming simulation models is to approximate the input-output relation of the simulation model by a so-called metamodel; see, e.g., Montgomery (2009), Sacks et al. (1989a), (1989b), Koehler and Owen (1996), Myers (1999), Jones et al. (1998), Booker et al. (1999), Den Hertog and Stehouwer (2002), Santner et al. (2003), and Kleijnen (2008). These metamodels are explicit functions and can be evaluated instantaneously. In literature, metamodels are also referred to as compact models, surrogate models, response surface models, and emulators.

1.2 Multi-objective optimization

Many optimization problems, including simulation-based problems, do not involve a single objective but involve multiple objectives. These multi-objective optimization problems (MOPs) occur in various fields such as supply chain management, medical decision making, and design engineering (Stewart et al. (2008)). For a comprehensive overview, we refer to White (1990), which contains a list of 500 papers describing different applications in various fields.

An example of a multi-objective optimization problem discussed in this thesis is the IMRT optimization problem. This medical decision problem deals with determining a good radiation plan for treating a tumor. In general, the objectives involved in this problem can be divided into two groups. Firstly, we have objectives that aim at maximizing the probability of eradicating the tumor. These objectives often focus on delivering a certain prescribed radiation dose to the tumor. The second group of objectives deals with minimizing the risk of damage to healthy tissue. As radiation needs to pass through other tissue before reaching the tumor, it is unavoidable that tissue surrounding the tumor also receives a certain radiation dose. The objectives especially aim at limiting the dose delivered to healthy tissue that is very sensitive to radiation. As the first group of objectives often favors more radiation and the second less radiation, it is impossible

to determine a radiation plan that optimizes all objectives. Instead, a good trade-off between the different objectives needs to be determined. As the definition of a good trade-off also depends on the preference of the physician and the patient, it is difficult to formulate a general definition.

An often used approach to determine a suitable trade-off is to convert the multi-objective optimization problem into a single objective problem. This can, for instance, be done by optimizing a weighted sum of the objectives or by optimizing one objective while putting bounds on the others. A drawback of these approaches is that it requires determining weights or bounds before the decision maker has any information on the possible solutions. By choosing the weights or bounds, the decision maker thus has to formulate a certain preference before knowing anything about the possible trade-offs among the different objectives. Therefore, several optimizations and interactions of the decision maker are often required before a satisfying solution is found. Another drawback is that the decision maker obtains little insight into the trade-offs between the different objectives. Information on these trade-offs can be valuable as it may influence the choice of the decision maker. If, for example, a small deterioration in one objective allows for a large improvement of another objective, the decision maker may wish to select a solution with a different trade-off between these objectives.

An approach that overcomes these problems is to approximate the Pareto set. The Pareto set contains all Pareto optimal solutions, e.g., solutions for which improvement in one objective is not possible without deterioration of another. By evaluating the Pareto set, decision makers can thus obtain insight into the possible trade-offs among the objectives. This insight can help the decision maker to select a solution that better reflects his preferences. The fact that the decision maker no longer has to formulate his preference a-priori by setting weights or bounds is a clear advantage of using the Pareto set. To determine Pareto optimal points, also single-objective optimization problems have to be formulated and solved which can be rather time-consuming. Combined with a large number of solutions in the Pareto set, it is generally impossible to determine the complete Pareto set. Instead an approximation of the Pareto set is determined based on a limited set of solutions. Determining this approximation is thus an important step in solving many multi-objective optimization problems.

1.3 Approximation methods

1.3.1 Metamodeling approach for black-box functions

In this section, we describe the global steps of the metamodeling approach; see Den Hertog and Stehouwer (2002). This approach determines a metamodel which can be used to analyse or optimize the simulation model and the simulated (real) system. The

four different steps of the metamodeling approach are: problem definition, design of computer experiments, metamodeling, and analysis and optimization. Next, we shortly explain these steps and mention the problems relevant for this thesis. A more detailed description can be found in Stinstra (2006).

Step 1: Problem specification

The first step consists of formulating the characteristics of the approximation problem. Firstly, we identify and define the input and output variables. At this stage, it is often not certain which input variables have an important effect on the output variable. The set of input variables may thus contain irrelevant variables.

Secondly, we determine the design space, i.e., the region in the input space for which we want to approximate the simulation model. The design space can be limited by lower and upper bounds on the individual input variables but also by constraints based on combinations of input variables. These constraints can, for instance, be physical constraints that cause certain combinations of input variables to be infeasible. Expert knowledge can also reduce the design space when it is already know that certain input combinations do not result in good solutions. Constraints on the output variables can also be formulated. However, as the output value is only known after running the simulation model, we can only check these constraints afterwards and cannot use them to limit the design space a priori.

Thirdly, we need to set the simulation budget which determines how many simulations can be performed to approximate the simulation model. This budget can be formulated in terms of the number of simulation runs or the amount of simulation time. Lastly, if the metamodel is used for optimization, the objective function and constraints can be specified in this step of the metamodelling approach.

Step 2: Design of computer experiments

To obtain information about the simulation model, we must perform simulations for different combinations of input values. A set of several combinations of input values is called a design; each combination is called a design point. When each simulation run is time-consuming, the number of design points we can evaluate is limited. This implies that it becomes important how we select these design points. When functions are subject to stochastic noise, design of experiments deals with this question, see, e.g., Kleijnen (2008). However, these designs of experiments are not suitable for determinist simulation models. Some causes are the following (see Stehouwer and Den Hertog (1999)):

- The presence of stochastic noise in traditional experiments can result in different output values when the same design point is evaluated multiple times. Therefore, design points are often evaluated more than once in traditional design of

experiments. For deterministic black-box functions, this is unnecessary as multiple evaluations of the same point always results in the same output value.

- In traditional design of experiments, the design points are often selected at or near the border of the design space. The use of different metamodeling techniques for deterministic black-boxes may result in a different preference for the positioning of the design points. For example, when fitting a Kriging metamodel, is it better to spread the design points evenly over the entire design space.

For approximating deterministic simulation models, designs of computer experiments are developed to serve as good designs. Several types of designs are developed including (space-filling) Latin hypercube designs, orthogonal arrays and uniform designs. Which design type is most suitable for a particular application depends, among others, on the type of metamodeling technique. Therefore, we must select both the metamodeling technique and the design in this step. In several chapters of this thesis, we focus on determining designs of computer experiments having several desirable properties. More information on designs of computer experiments can be found in Sacks et al. (1989a) (1989b), Myers (1999), Simpson et al. (2001), Santner et al. (2003), Bursztyn and Steinberg (2006), Fang and Sudjianto (2006), Husslage (2006), and Forrester et al. (2008).

Step 3: Metamodeling

Using the input and output values of the evaluated design points, we now have to fit a metamodel that accurately describes the input-output relation of the simulation model. There exist many different types of metamodels. Polynomials, neural networks, rational functions, radial basis functions, symbolic regression and Kriging models are just a number of possible approximation methods. In this thesis, we use the last three methods. For more information on different types of metamodels, see Santner et al. (2003), Fang and Sudjianto (2006), and Forrester et al. (2008).

After fitting a metamodel, the model must be validated using techniques such as cross-validation, see Kleijnen and Sargent (2000). If the metamodel is evaluated to be invalid, we should fit a different metamodel or evaluate additional design points.

Step 4: Analysis and optimization

When we have found a valid metamodel, we can use it for a number of purposes. Firstly, we can us it to gain insight into the relation between the input and output variables of the simulation model. This insight can be used to validate the simulation model; see Kleijen (1999). Secondly, the metamodel can be used to optimize some function of the output value of the simulation model.

In this thesis, we focus on so-called one-shot metamodeling approaches in which each of the above steps is performed once. However, there are also sequential approaches in which the selection of design points is based on metamodels fit to previously evaluated design points. This means that Steps 2 and 3 are performed multiple times. In Section 1.3.3, we motivate why we chose to focus our research on one-shot approaches instead of sequential approaches.

1.3.2 Sandwich algorithms for Pareto sets

Many different methods have been developed for approximating Pareto sets. Certain methods can be used only for problems with two objectives. Methods and techniques that can handle more than two objectives include:

- ϵ-Constraint method (Haimes et al. (1971))
- Genetic algorithms (Fonseca and Fleming (1995))
- Normal-boundary intersection (Das (1999a))
- Normal constraint method (Messac et al. (2003), Messac and Mattson (2004))
- Physical programming (Messac and Mattson (2002))
- Sandwich algorithms (Solanki et al. (1993), Klamroth et al. (2002), Craft et al. (2006) and Shao and Ehrgott (2008))
- Simulated annealing (Czyżak and Jaszkiewicz (1998))
- Weighting method (Zadeh (1963))

General overviews and discussions of multi-objective optimization methods are given in the books of Hwang and Masud (1979), Steuer (1986), Miettinen (1999), Ehrgott (2005), and Branke et al. (2008), and the survey papers of Ruzika and Wiecek (2003), Marler and Arora (2004), and Ehrgott and Wiecek (2005).

We focus our research on the approximation of multi-dimensional convex Pareto sets using sandwich algorithms. By multi-dimensional, we mean that the problem has more than two objectives. Several factors make the approximation of these Pareto sets much more complicated than bi-objective Pareto sets. A Pareto set is called convex if the union of the Pareto set and all points dominated by the Pareto set forms a convex set. Our research into convex Pareto sets is motivated by the IMRT optimization problem where most objectives are convex or can be made convex. However, convex Pareto sets also occur in other areas, as the geometric programming example in Chapter 7 shows.

Sandwich algorithms approximate a convex Pareto set by determining an inner and outer approximation between which the Pareto set is sandwiched. Using these two approximations, an upper bound on the approximation error of the Pareto set can be determined. In most sandwich algorithms, this upper bound is used to select which part of the approximation should be improved in each step of the algorithm. This approach thus aims at efficiently selecting the optimizations by using the upper bound. Furthermore, the upper bound provides the decision maker with a quality guarantee on the accuracy of the approximation.

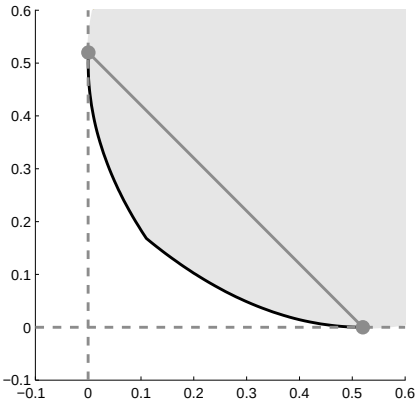


Figure 1.1: Convex Pareto set with initial inner and outer approximation.

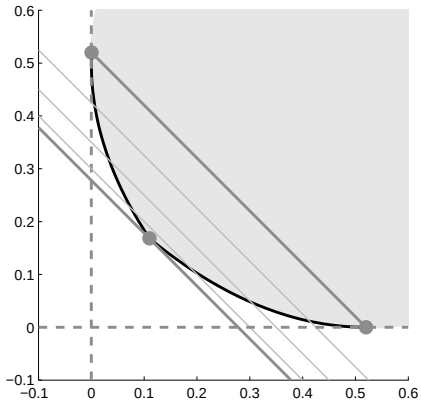


Figure 1.2: Solving optimization to determine new Pareto optimal point.

Although there are differences, most sandwich algorithms use the same basic steps. To explain these steps, we use the example in Figure 1.1. Although we consider multi-dimensional problems in this thesis, we use a bi-objective example to simplify the explanations. The shaded area in Figure 1.1 represents the set of all feasible objective vectors and the black line represents the Pareto set.

Step 1: Determining initial approximations

The first step of each sandwich algorithm is to obtain an initial inner and outer approximation. These initial approximations are often based on the solutions found by minimizing each objective separately. This gives the two Pareto optimal solutions represented by the two dots in Figure 1.1. As we approximate a convex Pareto set, the gray line connecting these two solutions forms an inner approximation of the Pareto set. The initial outer approximation is shown by the two dashed lines. We know that there are only solutions above and to the right of these two bounds, because the two initial solutions were obtained by minimizing the individual objectives.

Step 2: Formulating and solving a single-objective optimization problem

In the second step, we determine a new Pareto optimal point by formulating and solving a single-objective optimization problem. In our example, this problem is a weighted sum problem, which consists of optimizing a weighted sum of the two objectives. The weights are chosen such that the iso-objective lines are parallel to the inner approximation. In Figure 1.2, we plotted a number of these iso-objective lines and the optimal point. The figure shows that by using these weights, we find a Pareto optimal point furthest away from the current inner approximation.

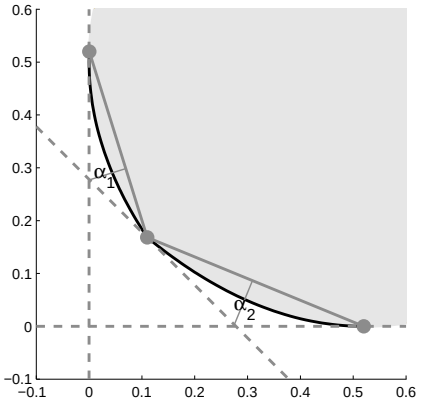


Figure 1.3: Calculation of upper bound on approximation error to select the next optimization.

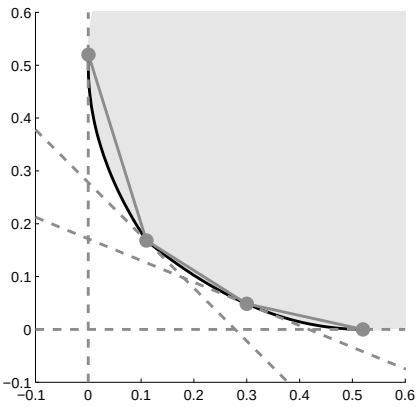


Figure 1.4: Inner and outer approximation after two iterations.

Step 3: Updating the inner and outer approximation

The inner approximation is updated by adding the Pareto optimal point found in Step 2. As shown in Figure 1.3, the inner approximation now consists of two line pieces, which are referred to as facets. The outer approximation is updated by adding the iso-objective line at the new Pareto optimal point. Because this is the optimal iso-objective line, we know that there are no feasible points below this line.

Step 4: Calculating an upper bound for the approximation error

The fourth step consists of calculating an upper bound for the approximation error. We choose to calculate this bound by determining the maximal distance between each facet and the outer approximation. The lines showing these maximal distances for each facet are indicated by α_1 and α_2 in Figure 1.3.

Step 5: Stop or select facet for further improvement

If the approximation is accurate enough or some other stopping criterion is met, the sandwich algorithm is terminated at this step. Otherwise, we select the facet with the largest maximal distance α and return to Step 2. The weights of the weighted sum problem in Step 2 are then chosen such that the iso-objective lines are parallel to the selected facet. In this example, we would select the right facet because $\alpha_2 > \alpha_1$. After performing Steps 2 and 3 for this facet, we obtain the approximation plotted in Figure 1.4.

Determining an approximation of the Pareto set is only one step in the complete solution process of a multi-objective optimization problem. When we have found a good approximation of the Pareto set, a decision maker still has to choose a solution from this set. When only two or three objectives are considered, the approximation of the Pareto set can still be visualized using a plot of the objective-vectors. However, when the multi-objective optimization problem has more than three objectives, other methods have to be used. Several different methods to deal with this problem see are discussed in Das (1999b), Monz (2006), Craft et al. (2007), Eskelinen et al. (2007), and Branke et al. (2008).

1.3.3 Similarities and differences between approximating black-box functions and Pareto sets

As we mentioned before, the problems of approximating a black-box function and a Pareto set are similar in certain respects. An important similarity is the time-consuming nature of the calculations needed to determine data points on which the approximations can be based. Therefore, efficient selection of the simulations to run or optimizations to perform is important in both approximation problems. This similarity generated our interest in the problem of approximating Pareto sets as we wanted to see whether approaches from design of computer experiments could also be used for the selection of optimizations. However, when we started to study this problem, we discovered that a different approach would be more suitable for the efficient approximation of convex Pareto sets.

An important difference between the used methods is that we use a so-called one-shot approach for black-box functions and a sequential approach for convex Pareto sets. The designs of computer experiments considered in this thesis are one-shot because the selection is made once before the evaluations are performed and is thus not influenced by the outcomes of these evaluations. The sandwich algorithms, on the other hand, are sequential because outcomes of previous optimizations are used to select new optimizations. Although we use a one-shot approach for black-box functions and a sequential approach for Pareto sets, one-shot and sequential approaches exist for both approximation prob-

lems. For example, the normal constraint method (Messac et al. (2003), Messac and Mattson (2004)) selects a fixed set of optimization problems after minimizing the individual objectives to determine their relevant ranges. An example of a sequential design of computer experiments can be found in Jin et al. (2002) where the maximum entropy and the integrated mean squared error criterion are used to select new evaluations. Other examples of sequential designs of computer experiments can be found in Kleijnen and Van Beers (2004) and Van Beers and Kleijnen (2008).

The decision to focus on one-shot approaches for the design of computer experiments is motivated by two main reasons. Firstly, one-shot designs are useful as an initial design on which to base the first metamodel in a sequential approach. As the quality of this initial metamodel can influence the number of subsequent evaluations needed to obtain an accurate metamodel, it is important to select a good initial design. The design points must be well positioned in the design space and not be too few. Therefore, if the simulation budget is very limited, a good one-shot design may give better results than a sequential design with a too small initial design. Secondly, when using a one-shot design, we can easily perform parallel computations and deal with multiple outputs. When we can perform m simulation runs in parallel, we can simply start with evaluating m design points from the one-shot design and select a new unevaluated design point when one of the simulation runs is finished. Furthermore, the design points do not depend on the output of the black-box function, so the number of outputs has no influence on the design. Although some sequential approaches can also deal with parallel computations and multiple outputs, these approaches are generally less straightforward.

For the approximation of convex Pareto sets, we use sequential sandwich algorithms. The two main reasons for using a sequential approach are the following. Firstly, because of the convexity of the Pareto set, it is relatively easy to determine inner and outer approximations of the Pareto set. These approximations enable us to determine upper bounds on the approximation error, which can be used to identify which parts of the approximations can potentially improve the most. A sequential approach can use this valuable information to select the optimizations, whereas a one-shot approach cannot. Therefore, a one-shot approach may result in unnecessary optimizations that determine data points in parts where the approximation is already accurate enough. Secondly, determining data points in the Pareto set requires formulating and solving a single-objective optimization problem. Most approaches use one formulation in which certain parameters are varied to determine different data points. For example, the weighted sum method optimizes a weighted sum of the objective functions for different sets of weights. Even though we know that the Pareto set is convex, determining an efficient set of parameter values is generally very difficult without additional information on the exact shape of the Pareto set. For the weighted sum method, Das and Dennis (1997) have

shown that using an uniform distribution of weight vectors does not generally give an uniform distribution of points from a Pareto set. The normal constraint method of Messac and Mattson (2004), on the other hand, does give a relatively even distribution of points on the Pareto set. However, this method often results in more optimizations and data points than strictly necessary; e.g., relatively flat regions of a convex Pareto set can often be adequately approximated by the convex hull of a small number of data points. As the Normal Constraint method and other one-shot approaches have no information on these flat areas, they often generate unnecessary data points. These problems illustrate that a one-shot approach is generally not capable of selecting an efficient set of optimizations.

Besides the one-shot and sequential approach, another difference is that the dimensionality of the data points in Pareto sets is generally lower than that of data points of black-box functions. The dimensionality of the first is determined by the number of objectives and of the second by the number of input and output variables. The number of objectives is generally fewer than ten as it is very difficult for a decision maker to determine a good trade-off between many different objectives; see Miller (1956) and Saaty and Ozdemir (2003). Although an approximation of the Pareto set can support the decision maker, trading of more than ten objectives is not useful in practice. The number of input variables of a black-box function on the other hand can be much larger than ten, especially in complex simulation models.

1.4 Contribution

1.4.1 Maximin Latin hypercube designs

The contributions of this thesis can be divided into four topics. The first topic concerns several aspects of maximin Latin hypercube designs (LHDs). Maximin LHDs form a class of designs of computer experiments and have two important properties: non-collapsingness and space-fillingness. By non-collapsingness, we mean that for every input variable it holds that all design points have a different value. By choosing the design points in this way, we avoid the situation that design points that coincide in one or several input variables collapse when the input variable in which they differ turn out to have no (important) influence on the output value. Especially, when each evaluation of the black-box function is time-consuming, it is important to avoid this situation. Space-fillingness means that we select the design points such that they cover all parts of the design-space uniformly. The maximin criteria, which aims at maximizing the minimal distance between any pair of design points, is one criterion to enhance space-fillingness. The maximin criterion exists in several variants depending on the chosen distance measure, e.g., the ℓ^2, ℓ^1 and ℓ^∞ norm.

In Chapter 2, we consider several methods for finding maximin LHDs. Firstly, we construct maximin LHDs by limiting the search to so-called periodic designs. Secondly, we use the Enhanced Stochastic Evolutionary (ESE) algorithm of Jin et al. (2005) to find approximate maximin LHDs where the term 'approximate' indicates that optimality of the maximin objective is not guaranteed. The (approximate) maximin LHDs are determined for up to 10 input variables and up to 300 design points. The Euclidean distance measure ℓ^2 is used for the maximin criterion. Besides ℓ^2-maximin, we also use the Audze Eglais measure (Audze and Eglais (1977)) to find space-filling LHDs.

Because determining maximin LHDs becomes more difficult as the number of input variables and design points increase, we introduce upper bounds on the ℓ^2-, ℓ^1-, and ℓ^∞-maximin criteria in Chapter 3. By comparing these bounds to the separation distance—the minimal distance between any pair of points—of an approximate maximin LHD, we can assess the quality of these designs. For all three maximin criteria, we determine bounds for different numbers of input variables and design points. To determine these bounds, we use several techniques and problem-formulations including Mixed Integer Programming, the Traveling Salesman Problem and the Graph Covering Problem.

In certain situations, we need a special type of designs consisting of two separate designs, one being a subset of the other. These nested designs can be used to deal with training and test sets, models with different levels of accuracy, linking parameters, and sequential evaluations. As non-collapsingness and space-fillingness are also important for nested designs, we discuss the construction of nested maximin LHDs in Chapter 4. A difficulty when constructing these designs is that depending on the number of design points in each design, the LHD-structure can not always be satisfied for both designs. For these cases, we introduce three different grid-structures that aim at maintaining the LHD-structure as much as possible. For each of these structures, we construct nested approximate maximin designs and discuss how to determine which grid to use for a specific application. Nested maximin designs with two input variables are determined using a branch-and-bound algorithm. To determine nested approximate maximin designs with more input variables, four different variants of the ESE algorithm of Jin et al. (2005) are introduced and compared.

1.4.2 Subset selection from large non-uniform datasets

Similar to design of computer experiments, the second topic of this thesis also deals with determining a dataset for fitting a metamodel. The main difference is that now we already have a dataset at our disposal containing the input and output values of a large number of deterministic simulations, experiments, or function evaluations. When building a metamodel, the general intuition is that using more data always results in a better model. However, when the dataset is large and non-uniformly distributed over

the design space, we show that this need not always be true. In Chapter 5, we show for the Kriging method–which is frequently used for fitting metamodels—that using such a dataset can cause several problems. By using a uniform subset instead of the complete dataset, we aim to reduce these problems. Some aspects that can be improved by using a uniform subset are reducing the time necessary to fit the model, avoiding numerical inaccuracies, and improving the robustness with respect to errors in the output data. We describe and compare several new and current methods for selecting a uniform subset. These methods are tested and compared on several artificial datasets and one real life dataset.

1.4.3 Complexity control in symbolic regression

Chapter 6 deals with a different method for fitting a metamodel: symbolic regression. This technique has the advantage of being very flexible as only few restrictions are imposed on the structure of the metamodel. This can result in more accurate and better interpretable metamodels, but can also lead to "overfitting". A model is called overfitted if it also tries to explain the noise in the training data and thus becomes a less accurate description of the general behavior of the model underlying the training data. To reduce the risk of overfitting, we introduce a measure to quantify the complexity of a function. This measure is based on the idea that the complexity of a function is correlated with the minimal degree of a polynomial necessary to approximate this function with a certain accuracy. Using Pareto simulated annealing, we determine metamodels that give a good balance between complexity and accuracy in the training data.

1.4.4 Enhancement of sandwich algorithms for approximating convex Pareto sets

The first three topics all involve the approximation of a black-box function. Chapter 7, on the other hand, deals with approximating a Pareto set. More specifically, we consider the enhancement of sandwich algorithms for approximating convex Pareto sets with two or more objectives. These sets generally result from multi-objective optimization problems (MOPs) with convex objective functions and constraints. Sandwich algorithms are used to approximate these Pareto sets as they can provide a quality guarantee on the accuracy of the approximation. We introduce three enhancements for existing sandwich algorithms. Firstly, we introduce dummy points that can help us in better selecting which single-objective optimization to run in each step of the sandwich algorithm. Secondly, we define a quality measure that provides easily interpretable quality guarantees. We also describe how this measure can easily be calculated using dummy points. Thirdly, transformations are introduced that can improve the approximations and extend the ap-

plication of sandwich algorithm to certain non-convex MOPs. To test the effect of these enhancements, we compare several existing sandwich algorithms with our new algorithm that incorporates the above enhancements. A comparison using four test cases shows that our new algorithm is generally more efficient; i.e., it requires less time-consuming optimization to reach the same level of (guaranteed) accuracy.

1.5 Overview of research papers

This thesis contains the following six research papers:

Chapter 2 Husslage, B.G.M., G. Rennen, E.R. van Dam, and D. den Hertog. Space-filling Latin hypercube designs for computer experiments, *Optimization and Engineering*. Submitted.

Chapter 3 Dam, E.R. van, G. Rennen, and B.G.M. Husslage (2009). Bounds for maximin Latin hypercube designs, *Operations Research*, **57**.

Chapter 4 Rennen, G., B.G.M. Husslage, E.R. van Dam, and D. den Hertog (2009). Nested maximin Latin hypercube designs, *Structural and Multidisciplinary Optimization*. To appear.

Chapter 5 Rennen, G. (2009). Subset selection from large datasets for Kriging modeling, *Structural and Multidisciplinary Optimization*, **38(6)**, 545–569.

Chapter 6 Stinstra, E.D., G. Rennen, and G.J.A. Teeuwen (2008). Metamodeling by symbolic regression and Pareto simulated annealing, *Structural and Multidisciplinary Optimization*, **35(4)**, 315–326.

Chapter 7 Rennen, G., E.R. van Dam, and D. den Hertog. Enhancement of sandwich algorithms for approximating multi-dimensional convex Pareto sets, *INFORMS Journal on Computing*. Submitted.

The thesis is divided into four parts that correspond to the four contributions described in Section 1.4.

Part I

Maximin Latin hypercube designs

Chapter 2

Space-filling Latin hypercube designs for computer experiments

> "Space," it says, "is big. Really big. You just won't believe how vastly hugely mind-bogglingly big it is. I mean you may think it's a long way down the road to the chemist, but that's just peanuts to space."
>
> (Douglas Adams, *The Hitchhiker's Guide to the Galaxy*)

2.1 Introduction

A k-dimensional Latin hypercube design (LHD) of n points, is a set of n points $x_i = (x_{i1}, x_{i2}, \ldots, x_{ik}) \in \{0, \ldots, n-1\}^k$ such that for each dimension j all x_{ij} are distinct. A LHD is called maximin when the separation distance $\min_{i \neq j} d(x_i, x_j)$ is maximal among all LHDs of given size n, where d is a certain distance measure. In this chapter, we concentrate on the Euclidean (or ℓ^2) distance measure, i.e.,

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^{k} (x_{il} - x_{jl})^2}, \tag{2.1}$$

because this measure is often chosen in practice.

Besides maximin LHDs, we also treat Audze-Eglais LHDs. These LHDs minimize the following objective:

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{d(x_i, x_j)^2}, \tag{2.2}$$

where $d(x_i, x_j)$ is again the Euclidean distance between points x_i and x_j. By minimizing this objective, we can also obtain LHDs with uniformly distributed points (Bates et al. (2004)).

For both classes of LHDs, we aim to construct a database of the best designs known in literature. We do this by generating new designs and comparing them with existing designs. These new designs are often approximate maximin or Audze-Eglais designs in the sense that optimality of the objective is not guaranteed. The reason is that optimization over the total set of LHDs can be very time-consuming for larger values of k and n. Therefore, to find good designs, optimization is often done over a certain class of LHDs or heuristics are used, which do not guarantee optimality. A good example of the first case are the periodic LHDs described in this chapter. Examples of the second case are simulated annealing used by Morris and Mitchell (1995), the permutation genetic algorithm of Bates et al. (2004) and the Enhanced Stochastic Evolutionary (ESE) algorithm of Jin et al. (2005).

The designs that are best according to the comparisons in this chapter can be downloaded for free from the website `http://www.spacefillingdesigns.nl`. As far as we know this is the first extensive online catalogue of maximin and Audze-Eglais LHDs, although there are several catalogues for classical design of experiments, see, e.g., the WebDOE$^{\text{TM}}$ website of Crary (2008). Crary et al. (2000) developed I-OPT$^{\text{TM}}$ to generate designs with minimal integrated mean squared error (IMSE). They found that IMSE-optimal designs can have proximate design points, which they call "twin points"; see also Crary (2002). For more websites and software for various types of designs, we refer to pages 54 and 130 of Kleijnen (2008).

Our main motivation for investigating this topic is that maximin and Audze-Eglais Latin hypercube designs are very useful in computer simulation. One important area where computer simulation is much used is engineering. Engineers are confronted with the task of designing products and processes. Since physical experimentation is often expensive and difficult, computer models are frequently used for simulating physical characteristics. Engineers often need to optimize the product or process design; i.e., they need to find the best settings for a number of design parameters that influence the critical quality characteristics of the product or process. A computer simulation run is often time-consuming and there is a large number of possible input combinations. For these reasons, engineers construct metamodels that model the quality characteristics as explicit functions of the design parameters. Such a metamodel—also called a (global) approximation model or surrogate model—is obtained by simulating a number of design points. Well-known metamodel types are polynomial and Kriging models. Since a metamodel evaluation is much faster than a simulation run, in practice the metamodel is used instead of the simulation model, to gain insight into the characteristics of the product or process and to optimize it. A review of metamodeling applications in structural optimization can be found in Barthelemy and Haftka (1993), and in multidisciplinary design optimization in Sobieszczanski-Sobieski and Haftka (1997).

As observed by many researchers, there is an important distinction between designs for computer experiments and designs for the more traditional response surface methods. Physical experiments exhibit random errors and computer experiments are often deterministic (Simpson et al. (2004)). This distinction is crucial and much research is therefore aimed at obtaining efficient designs for deterministic computer experiments.

As several authors recognize, designs for computer experiments should at least satisfy the following two criteria (see Johnson et al. (1990) and Morris and Mitchell (1995)). First of all, the design should be *space-filling* in some sense. When no details on the functional behavior of the response is available, it is important to be able to obtain information for the entire design space. Therefore, design points should be "evenly spread" over the entire region. One of the measures often used to obtain such space-filling designs is the maximin measure. The Audze-Eglais measure is another measure used for this purpose. Note that in other fields of research, space-filling designs are referred to as *low discrepancy* designs.

Secondly, the design should be *non-collapsing*. When one of the design parameters has (almost) no influence on the function value, two design points that differ only in this parameter will "collapse", i.e., they can be considered as the same point that is evaluated twice. For deterministic simulation models this is not desirable. Therefore, two design points should not share any coordinate values when it is not known a priori which dimensions are important. To obtain non-collapsing designs, the Latin hypercube structure is often enforced. It can be shown that if the function of interest is independent of one or more of the k parameters then, after removal of the irrelevant parameters, the projection of the LHD onto the reduced design space retains good spatial properties; see Koehler and Owen (1996). Maximin LHDs are frequently used in practical applications, see, e.g., the examples given in Driessen et al. (2002), Den Hertog and Stehouwer (2002), Alam et al. (2004), and Rikards and Auzins (2004).

Only a few authors consider the construction of maximin LHDs. For example, Morris and Mitchell (1995) use simulated annealing to find approximate maximin LHDs for up to five dimensions and up to 12 design points, and a few larger values, with respect to the ℓ^1- and ℓ^2-distance measure. Van Dam et al. (2007) derive general formulas for two-dimensional maximin LHDs when the distance measure is ℓ^∞ or ℓ^1, while for the ℓ^2-distance measure (approximate) maximin LHDs up to 1000 design points are obtained by using a branch-and-bound algorithm and constructing (adapted) periodic designs. Ye et al. (2000) propose an exchange algorithm for finding approximate maximin symmetric LHDs. The symmetry property is used as a compromise between computing effort and design optimality. Jin et al. (2005) describe an enhanced stochastic evolutionary (ESE) algorithm for finding approximate maximin LHDs. They also apply their method for other space-filling criteria. Cioppa and Lucas (2007) consider a combination of the maximin

ℓ^2-distance and modified ℓ^2-discrepancy. Furthermore, they limit their search to nearly orthogonal LHDs; i.e., LHDs for which the maximum pairwise correlation between column of the matrix $X = [x_{ij}]$ is below 0.03 and the condition number of $X^\top X$ is below 1.13. Lastly, the Statistics Toolbox of Matlab also contains a function `lhsdesign` to generate approximate maximin LHDs. This function randomly generates a number of LHDs and picks the one with the largest separation distance. Although this method is very fast, other methods generally result in much better space-filling LHDs. To asses the quality of approximate maximin LHDs, Van Dam et al. (2009b) generate upper bounds on the separation distance for certain classes of maximin LHDs. By comparing the separation distances of LHDs to these bounds, we can get an indication of their quality.

There is much more literature related to maximin designs that are not restricted to LHDs. These maximin designs are certainly space-filling, but not necessarily non-collapsing. Firstly, the problem of finding the maximal common radius of n circles that can be packed into a square is equivalent to the maximin design problem in two dimensions. Melissen (1997) gives a comprehensive overview of historical developments and state-of-the-art research in this field. For the ℓ^2-distance measure in the two-dimensional case, optimal solutions are known for $n \leq 30$ and $n = 36$, see, e.g., Kirchner and Wengerodt (1987), Peikert et al. (1991), Nurmela and Östergård (1999), and Markót and Csendes (2005). Furthermore, many good approximating solutions have been found for $n \geq 31$; see the Packomania website of Specht (2008). Baer (1992) solved the maximum ℓ^∞-circle packing problem in a k-dimensional unit cube. The ℓ^1-circle packing problem in a square has been solved for many values of n; see Fejes Tóth (1971) and Florian (1989).

Secondly, the maximin design problem has been studied in location theory. In this area of research, the problem is usually referred to as the *max-min facility dispersion problem* (see Erkut (1990)). Facilities are placed such that the minimal distance to any other facility is maximal. Again, the resulting solution is certainly space-filling, but not necessarily non-collapsing. A few publications consider maximin designs in higher dimensions, e.g., Trosset (1999), Locatelli and Raber (2002), Stinstra et al. (2003), and Dimnaku et al. (2005). These publications describe nonlinear programming heuristics to find approximate maximin designs.

Audze-Eglais LHDs are also constructed by only a few authors. The criterion was first introduced by Audze and Eglais (1977) and is based on the analogy of minimizing forces between charged particles. Bates et al. (2004) formulate the problem of finding Audze-Eglais LHDs and use a permutation genetic algorithm to generate them. Liefvendahl and Stocki (2006) compare maximin and Audze-Eglais LHDs and recommend the Audze-Eglais criterion over the maximin criterion. Examples of practical applications of Audze-Eglais LHDs can be found in Rikards et al. (2001), Bulik et al. (2004), Stocki (2005), and Hino et al. (2006).

There are several other measures proposed in the literature besides maximin and Audze-Eglais, e.g., maximum entropy, minimax, IMSE, and discrepancy. For a good overview, we refer to Koehler and Owen (1996). In statistical environments, Latin hypercube sampling (LHS) is often used. In such an approach, points on the grid are sampled without replacement, thereby deriving a random permutation for each dimension; see McKay et al. (1979). Giunta et al. (2003) give an overview of pseudo- and quasi-Monte Carlo sampling, LHS, orthogonal array sampling, and Hammersley sequence sampling. They notice that the basic LHS technique can lead to designs with poor space-filling properties. Extensions of the basic LHS technique are therefore necessary to obtain better designs, but these are unfortunately not standard yet in all software packages. Bates et al. (1996) obtain designs for computer experiments by exploring so-called lattice points and using results from number theory.

Several papers combine space-filling criteria with the Latin hypercube structure. Jin et al. (2005) describe an enhanced stochastic evolutionary algorithm for finding maximum entropy and uniform designs. Van Dam (2008) derives interesting results for two-dimensional minimax LHDs.

In the literature different designs for computer experiments have been compared and the overall conclusion tends to be that the maximum entropy and distance-based criteria, such as maximin and Audze-Eglais, often perform best; see, e.g., Simpson et al. (2001), Santner et al. (2003), and Bursztyn and Steinberg (2006).

This chapter is organized as follows. Section 2.2 describes how periodic designs can be used to obtain good approximate maximin and Audze-Eglais LHDs. In Section 2.3, we shortly describe some heuristics found in literature that are used for the same purpose. The ESE algorithm of Jin et al. (2005) described in this section and periodic designs are used to generate new approximate maximin and Audze-Eglais LHDs. Computational results for up to ten dimensions and for up to 300 design points, as well as a comparison of the new and existing results, are provided in Section 2.4. Finally, Section 2.5 contains conclusions.

2.2 Periodic designs

Van Dam et al. (2007) show that two-dimensional maximin Latin hypercube designs often have a nice, periodic structure. By constructing (adapted) periodic designs, many maximin and otherwise good LHDs are found for up to 1000 points. Therefore, extending this idea to higher dimensions seems natural.

Let a k-dimensional Latin hypercube design of n points be represented by the sequences $y_1, \ldots, y_k$, with every y_i a permutation of the set $\{0, \ldots, n-1\}$. As in the two-dimensional case, a design is constructed by fixing the first dimension, without loss

of generality, to the sequence $y_1 = (0, \ldots, n-1)$ and assigning (adapted) periodic sequences to all other dimensions. Two types of periodic sequences are considered. The first one is the sequence $(v_0, \ldots, v_{n-1})$, where

$$v_i = (i+1)p \bmod (n+1) - 1, \text{ for } i = 0, \ldots, n-1 \tag{2.3}$$

and p is the period of the sequence, which is chosen such that $n+1$ and p have no common divisor, i.e., $\gcd(n+1, p) = 1$, resulting in a permutation of the set $\{0, \ldots, n-1\}$.

Note that the periodic designs obtained in this way resemble *lattices*; see, e.g., Bates et al. (1996). The main difference is that lattices are infinite sets of points, which may collapse, so constructing a (finite) Latin hypercube design requires a proper subset of non-collapsing lattice points. For given n, the structure of the lattice will, however, not always lead to a Latin hypercube design with a sufficient number of points. This is in contrast to periodic designs, for which the modulo operator insures that for every combination of periods p_j, with $\gcd(n+1, p_j) = 1$, $j = 2, \ldots, k$, a feasible Latin hypercube design is obtained.

The second type of sequence is the more general sequence $(w_0, \ldots, w_{n-1})$, where

$$w_i = (s + ip) \bmod n \text{ for } i = 0, \ldots, n-1. \tag{2.4}$$

Note that we changed the modulus compared to equation (2.3). In this case, all starting points $s = 0, \ldots, p$ and all periods $p = 1, \ldots, \lfloor \frac{n}{2} \rfloor$ will be considered. Note, however, that the resulting sequence w may no longer be one-to-one; i.e., some values may occur more than once, and, hence, the resulting design may no longer be an LHD. Now, let $r > 0$ be the smallest value for which $w_r = w_0$; it then follows that $r = \frac{n}{\gcd(n,p)}$. When $r < n$, a way to construct a one-to-one sequence of length n is by shifting parts of the sequence by, say, q, and repeating this when necessary. To formulate this procedure more explicitly, for the updated sequence w it now holds that

$$w_i = (s + ip + jq) \bmod n, \text{ for } i = jr, \ldots, (j+1)r - 1, \text{ and } j = 0, \ldots, \gcd(n, p) - 1. \tag{2.5}$$

Let m represent the modulus and, hence, the type of sequence used, i.e., $m = n + 1$ corresponds to the first type and $m = n$ to the second. For given n, we now have to set the parameters (p, q, s, m) for every sequence $y_2, \ldots, y_k$.

To find the best settings for the parameters, it would be best to test all values. However, when the dimension and the number of points increase, the number of possibilities increases rapidly. Hence, computing all possibilities gets very time-consuming or even impossible. Therefore, three classes of parameter settings (named A, B, and C) are distinguished. The largest one, class A, consists of checking the following parameter values: $p = 1, \ldots, \lfloor \frac{n}{2} \rfloor$, $q = 1 - p, \ldots, p - 1$, $s = 0, \ldots, p$, and $m \in \{n, n+1\}$.

Dimension k	Class A	Class B	Class C
3	$2 \leq n \leq 70$	$71 \leq n \leq 100$	–
4	$2 \leq n \leq 25$	$26 \leq n \leq 100$	–
5	–	$2 \leq n \leq 80$	$81 \leq n \leq 100$
6	–	$2 \leq n \leq 35$	$36 \leq n \leq 100$
7	–	–	$2 \leq n \leq 100$

Table 2.1: Classes of periodic sequences used to generating maximin designs with number of points n and dimension k.

Testing in three and four dimensions indicated that almost all adapted periodic maximin designs are based on a shift of $1 - p$, -1, or 1 (as was the case for two dimensions; see Van Dam et al. (2007)). Furthermore, most maximin designs are found to have a starting point equal to either $p - 1$ or p. Class B is therefore set up to be a subset of class A with the aforementioned restrictions on the parameters q and s. Finally, for the dimensions 5 to 7 the number of possibilities has to be reduced even further, leading to parameter class C, which (based on some more test results) restricts class B to the values $q = 1$ and $s = p$, leaving the other parameters unchanged. Table 2.1 shows the different classes used in the computations of the approximate maximin LHDs for each dimension. For the approximate Audze-Eglais LHDs, we only used class C.

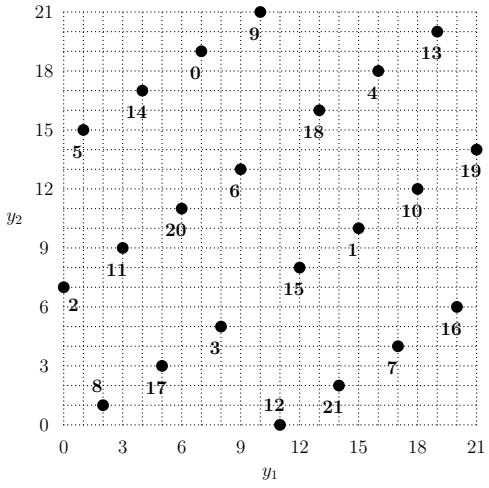


Figure 2.1: Two-dimensional projection of the three-dimensional LHD (y_1, y_2, y_3) of 22 points.

As an example, consider a three-dimensional adapted periodic LHD of 22 points. For the maximin criterion, a best parameter setting (class A) is found to be $(p_2, q_2, s_2, m_2) = (8, -7, 7, 22)$ and $(p_3, q_3, s_3, m_3) = (3, 0, 2, 23)$ and, hence, the corresponding maximin

LHD, with separation distance 69, is defined by the sequences

$$\begin{aligned} y_1 &= (0,\ 1\ ,2,\ 3\ ,\ 4\ ,\ 5\ ,\ 6\ ,\ 7\ ,8,\ 9\ ,10,11,12,13,14,15,16,17,18,19,20,21\ ), \\ y_2 &= (7,15,1,\ 9\ ,17,\ 3\ ,11,19,5,13,21,\ 0\ ,\ 8\ ,16,\ 2\ ,10,18,\ 4\ ,12,20,\ 6\ ,14\ ), \\ y_3 &= (2,\ 5\ ,8,11,14,17,20,\ 0\ ,3,\ 6\ ,\ 9\ ,12,15,18,21,\ 1\ ,\ 4\ ,\ 7\ ,10,13,16,19\ ). \end{aligned} \quad (2.6)$$

Thus, y_3 is a periodic sequence, with $m = n + 1$, and y_2 is an adapted periodic sequence, with $m = n$ and $q_2 = -7$. Note that to obtain a one-to-one sequence, the second part of y_2, i.e., $(0, 8, \ldots, 14)$, is formed by shifting the first part of y_2, i.e., $(7, 15, \ldots, 21)$, by -7. The periods and shift are clearly visible in the two-dimensional projection of the LHD in Figure 2.1. In this figure the y_3-values are depicted at the design points.

Like in the two-dimensional case, it may happen that for a given n the corresponding maximin LHD has a separation distance that is smaller than the distance of a design of $n - 1$ points. For these n, however, better designs can usually be derived by adding an extra "corner point" to the LHD of $n - 1$ points. In this way, a monotone nondecreasing sequence of separation distances was found for all dimensions; see Table 2.6 in Appendix 2.A.

2.3 Other methods

2.3.1 Enhanced stochastic evolutionary algorithm

Besides restricting ourselves to a certain class of LHDs, we can also generate good maximin or Audze-Eglais LHDs using heuristics. The ESE algorithm of Jin et al. (2005) is one of the methods developed for this purpose, and is used in this chapter to generate new approximate maximin and Audze-Eglais LHDs.

This method starts with an initial design and tries to find better designs by iteratively changing the current design. To determine if a new design is accepted, a threshold-based acceptance criterion is used. This criterion is controlled in the outer loop of the algorithm. In the inner loop of the algorithm new designs are explored.

The inner loop explores the design space as follows. At each iteration, the algorithm creates a fixed number of new designs by exchanging two randomly chosen elements. The new design with the largest separation distance or with the smallest Audze-Eglais objective value is then compared to the current design with a threshold criterion. The criterion is such that better designs are always accepted and worse designs can also be accepted with a certain probability. If the new design is accepted, it replaces the current design. This proces is repeated for a user defined number of iterations.

The outer loop controls the threshold value. After the inner loop is completed, the outer loop determines how much improvement is made in the inner loop. If the amount of improvement is above a certain level, the algorithm starts an improvement process in which it tries to rapidly find a local optimum. It does this by lowering the threshold value

and thus accepting fewer deteriorations in the inner loop. If too little improvement is made, an exploration process is started that is intended to escape from a local optimum. The threshold value is first rapidly increased to move away from a local optimum and later slowly decreased to find better designs after moving away. The final design of the algorithm is the best design found during all iterations of the inner loop.

For a more detailed description of the algorithm, we refer to the original paper of Jin et al. (2005). To find maximin and Audze-Eglais LHDs, we implemented the ESE algorithm in Matlab. The parameters of the algorithm were set to the values suggested in Jin et al. (2005). The only adjustment we made to the original algorithm is in the choice of stopping criterion. Instead of stopping after a fixed number of runs of the outer loop, our criterion is to stop when in the last 1000 runs of the outer loop no improvement is made.

2.3.2 Simulated Annealing

Another heuristic used to find maximin LHDs is simulated annealing. Morris and Mitchell (1995) were the first to apply simulated annealing for this purpose. The simulated annealing method tries to find good designs by iteratively changing a random starting design. A key characteristic of simulated annealing is that not only improvements are accepted but also changes that result in worse designs can be accepted. This enables simulated annealing to escape from local optima.

Besides Morris and Mitchell (1995), also Husslage (2006) uses simulated annealing for finding maximin LHDs. One of the main differences between the two methods is the objective function. Husslage (2006) directly uses the separation distance of a design, whereas Morris and Mitchell (1995) use a surrogate measure ϕ_p. The latter measure also takes into account the number of pairs of points with a certain distance between them. By including this information, it is easier to decide which design is best if they have the same separation distance. This surrogate measure is also used by other authors, e.g., Jin et al. (2005) and Palmer and Tsui (2001).

2.3.3 Permutation Genetic Algorithm

To obtain Audze-Eglais LHDs, Bates et al. (2004) use a permutation genetic algorithm. The genetic algorithm uses a population of 10 designs and creates new generations of designs by applying different crossover methods. Results of the algorithm are reported for eight different combinations of n and k. In Section 2.4, we make a comparison between these results and our designs obtained with periodic designs and ESE.

2.4 Computational results

Using (adapted) periodic designs and the ESE algorithm, we obtain approximate maximin and Audze-Eglais LHDs for the cases described in Table 2.2. All computations have been performed on PCs with a 2.8 GHz Pentium D processor. For the cases with $n > 100$, a limit of 6 hours was imposed on the calculation time.

Dimension k	3	4	5	6	7	8	9	10
Maximin PD	300	300	100	100	100			
Maximin ESE	300	300	100	100	100	100	100	100
Audze-Eglais PD	100	100	100					
Audze-Eglais ESE	100	100	100	100	100	100	100	100

Table 2.2: Largest values of n for which LHDs were generated using periodic designs (PD) and the ESE algorithm.

Table 2.6 in Appendix 2.A shows the squared ℓ^2 separation distance of the (approximate) maximin LHDs that were obtained by applying periodic designs and the ESE algorithm. From this table it can be seen that (adapted) periodic designs work particularly well for larger values of n. For dimension 2 to 4, there is a noticeable break-even point in the table, i.e., a point (or, better, an interval) where the preference shifts from the designs found by ESE to (adapted) periodic designs. Furthermore, these break-even points seem to increase with the dimension of the design and it is to be expected that break-even points for k-dimensional designs with $k \geq 5$ will occur for larger values of n, i.e., $n > 250$. This behavior could be explained by the "border effect", i.e., the irregularity of designs that is caused by the borders of the design space. Clearly, the number of "borders" of the k-dimensional box region increases exponentially, with respect to k. However, due to the Latin hypercube structure the number of design points that are located on or near these borders is limited. This, in turn, leads to very irregular optimal Latin hypercube designs when the number of design points is small with respect to the number of borders (which again depends on k). Hence, the nice, periodic structure that is sought for by our periodic heuristic works well only when the number of design points is relatively large compared to the dimension. Van Dam et al. (2007) have already shown the presence of this particular behavior in two-dimensional maximin Latin hypercube designs, i.e., their optimal designs can all be represented by periodic designs. The results of Table 2.6 suggest that this behavior also occurs in higher dimensions. ESE, however, does not depend on an underlying structure, so it can therefore often find better designs. Since all six- and seven-dimensional (adapted) periodic designs with 3 to 100 points are dominated by the designs found by ESE, the former are not computed for larger dimensions.

n	3 dim		4 dim		5 dim		6 dim		7 dim		8 dim		9 dim	
	M&M	ESE	M&M	ESE	M&M	ESE	M&M	ESE	M&M	ESE	M&M	ESE	M&M	ESE
3	**6**	**6**	**7**	**7**	**8**	**8**								
4	**6**	**6**	**12**	**12**	**14**	**14**								
5	**11**	**11**	**15**	**15**	**24**	**24**								
6	**14**	**14**	**22**	**22**	**32**	**32**	**40**	**40**						
7	**17**	**17**	**28**	**28**	**40**	**40**			**61**	**61**				
8	**21**	**21**	**42**	**42**	**50**	**50**					**91**	89		
9	**22**	**22**	**42**	**42**	**61**	**61**							**126**	**126**
10	**27**	**27**	**50**	47	**82**	**82**								
11	29	**30**	**55**	**55**	**80**	**80**								
12	**36**	**36**	**63**	**63**	**91**	**91**	**139**	136						
13														
14									**219**	215				

Table 2.3: Squared ℓ^2 separation distance of designs found by Morris and Mitchell (1995) and the ESE algorithm.

With the ESE algorithm, we can match the results of Morris and Mitchell (1995) for most combination of k and n. Only for the cases (k, n) equal to $(4, 10)$, $(6, 12)$, $(7, 14)$, and $(8, 8)$, we obtain slightly worse designs. Morris and Mitchell (1995), however, have already observed that designs that satisfy $n = k$ or $n = 2k$ exhibit special symmetric properties; they refer to them as *foldover designs*. For the case $k = 3, n = 11$, we obtained an improved (and optimal) design. Furthermore, using a branch-and-bound algorithm, the three-dimensional designs of up to 14 points have been verified to be optimal (Van Dam et al. (2009b)).

When comparing the ESE results with the simulated annealing results in Husslage (2006), we again see that ESE gives better or equally good results for most combination of k and n. For only nine combinations, the results of the SA algorithm are better. However, especially for larger values of n, the ESE algorithm finds designs with considerably higher separation distances.

$k \times n$	2×17	3×17	4×17	5×17	6×17	7×17	8×33	9×33	10×33
NOLHD	17	21	68	103	131	140	525	720	745
ESE	**18**	**54**	**103**	**159**	**214**	**227**	**1037**	**1244**	**1436**

Table 2.4: Squared ℓ^2 separation distance of the nearly orthogonal LHDs found by Cioppa and Lucas (2007) and the LHDs found by the ESE algorithm.

Table 2.4 contains a comparison with the results of Cioppa and Lucas (2007). For almost all cases, the separation distance of the nearly orthogonal LHD is considerably lower than the separation distance of the LHD found by ESE. Besides the use of a different search algorithm, this difference is probably also caused by two other factors; Cioppa and Lucas (2007) optimize a combination of the maximin ℓ^2-distance and the modified ℓ^2-discrepancy, and they consider only nearly orthogonal LHDs.

$k \times n$	2×5	2×10	2×120	3×5	3×10	3×120	5×50	5×120
PermGA	**1.2982**	**2.0662**	5.5174	**0.7267**	1.0242	1.9613	0.7270	0.7930
ESE	**1.2982**	**2.0662**	**5.4941**	**0.7267**	**1.0199**	**1.9328**	**0.7195**	**0.7840**

Table 2.5: Audze-Eglais values of designs found by Bates et al. (2004) and the ESE algorithm.

The results obtained for the Audze-Eglais measure are given in Table 2.7 in Appendix 2.A. We can easily see that for almost all cases the results of the ESE algorithm are better than the (adapted) periodic designs. It is likely that by running ESE for some more starting solutions, better or equally good designs can be found for all cases. The ESE algorithm thus outperforms the periodic designs for the Audze-Eglais measure. In Table 2.5, we compare the results with those found by Bates and see that the ESE algorithm gives better or equally good results. This shows that the ESE algorithm is quite successful in finding LHDs with a good Audze-Eglais value.

2.5 Conclusions

This chapter discusses existing and new results for maximin and Audze-Eglais Latin hypercube designs. Such designs can play an important role in the area of computer simulation. The new results are obtained using two heuristics. The first heuristic is based on the observation that many optimal LHDs—and two-dimensional LHDs in particular—exhibit a periodic structure. By considering periodic and adapted periodic designs, approximate maximin LHDs for up to seven dimensions and for up to 300 design points are constructed. The second heuristic uses the ESE algorithm of Jin et al. (2005) to find approximate maximin LHDs for up to ten dimensions. These new results are compared to existing results obtained with simulated annealing and permutation genetic algorithms. In most cases, the ESE algorithm resulted in the best maximin and Audze-Eglais LHDs. However, when the number of design points is large with respect to the dimension of the design, the periodic designs tend to be better. Appendix 2.A contains all the obtained squared ℓ^2 separation distances and Audze-Eglais values. All corresponding designs can be downloaded from the website `http://www.spacefillingdesigns.nl`.

2.A Tables of numerical results

n	3 dim		4 dim		5 dim		6 dim		7 dim		8 dim	9 dim	10 dim
	ESE	Per	ESE	Per	ESE	Per	ESE	Per	ESE	Per	ESE	ESE	ESE
2	**3**	**3**	**4**	**4**	**5**	**5**	**6**	**6**	**7**	**7**	**8**	**9**	**10**
3	**6**	3	**7**	4	**8**	5	**12**	6	**13**	7	**14**	**18**	**19**
4	**6**	**6**	**12**	**12**	**14**	11	**20**	15	**21**	16	**26**	**28**	**33**
5	**11**	6	**15**	12	**24**	11	**27**	15	**32**	16	**40**	**43**	**50**
6	**14**	**14**	**22**	16	**32**	23	**40**	28	**47**	29	**53**	**61**	**68**
7	**17**	14	**28**	16	**40**	23	**52**	28	**61**	31	**70**	**80**	**89**
8	**21**	**21**	**42**	25	**50**	32	**63**	42	**79**	46	**90**	**101**	**114**
9	**22**	21	**42**	25	**61**	39	**75**	45	**92**	47	**112**	**126**	**142**
10	**27**	21	**47**	36	**82**	55	**91**	62	**109**	68	**131**	**154**	**171**
11	**30**	24	**55**	39	**80**	55	**108**	62	**129**	69	**152**	**178**	**206**
12	**36**	30	**63**	46	**91**	62	**136**	91	**152**	95	**177**	**204**	**235**
13	**41**	35	**70**	51	**103**	64	**138**	91	**178**	95	**205**	**235**	**268**
14	**42**	35	**77**	70	**114**	86	**154**	104	**215**	119	**236**	**268**	**305**
15	**45**	42	**87**	71	**129**	88	**171**	111	**220**	129	**273**	**309**	**347**
16	**50**	42	**93**	85	**151**	101	**190**	130	**241**	155	**317**	**352**	**393**
17	**53**	42	**99**	85	**158**	113	**208**	131	**266**	161	**332**	**396**	**442**
18	**56**	50	**108**	94	**170**	123	**231**	155	**291**	186	**361**	**451**	**496**
19	**59**	57	**119**	94	**184**	136	**256**	169	**323**	195	**390**	**469**	**554**
20	**65**	57	**130**	106	**206**	139	**279**	210	**349**	226	**425**	**506**	**625**
21	**68**	65	**145**	116	**223**	165	**302**	210	**380**	236	**463**	**548**	**650**
22	**72**	69	**150**	117	**235**	174	**325**	223	**418**	270	**501**	**595**	**691**
23	**75**	72	**159**	130	**250**	178	**348**	236	**448**	273	**542**	**640**	**747**
24	**81**	76	**170**	138	**266**	201	**374**	258	**481**	308	**585**	**690**	**800**
25	86	**91**	**178**	156	**285**	205	**400**	286	**520**	350	**626**	**739**	**857**
26	86	**91**	**188**	156	**302**	226	**426**	296	**548**	365	**664**	**791**	**910**
27	90	**91**	**198**	157	**310**	238	**447**	310	**585**	382	**712**	**840**	**976**
28	**94**	**94**	**210**	174	**331**	258	**479**	339	**620**	406	**766**	**898**	**1041**
29	**101**	94	**221**	174	**349**	269	**507**	346	**654**	417	**817**	**956**	**1100**
30	**105**	**105**	**233**	194	**367**	310	**531**	390	**691**	458	**849**	**1019**	**1173**
31	**110**	107	**244**	212	**405**	310	**563**	390	**728**	482	**900**	**1104**	**1241**
32	110	**114**	**253**	212	**413**	341	**587**	419	**778**	518	**966**	**1139**	**1318**
33	**117**	114	**264**	215	**426**	341	**622**	430	**814**	537	**1010**	**1201**	**1396**
34	125	**133**	**273**	230	**445**	358	**648**	470	**851**	561	**1072**	**1270**	**1478**
35	126	**133**	**286**	234	**467**	366	**683**	495	**914**	586	**1113**	**1326**	**1555**
36	131	**133**	**297**	250	**486**	400	**719**	518	**939**	636	**1181**	**1405**	**1647**
37	138	**152**	**309**	266	**520**	408	**744**	528	**976**	668	**1236**	**1477**	**1721**
38	142	**152**	**321**	283	**541**	415	**788**	561	**1028**	709	**1286**	**1534**	**1790**
39	146	**152**	**330**	283	**566**	439	**816**	561	**1084**	726	**1344**	**1609**	**1870**
40	152	**155**	**342**	291	**575**	492	**876**	632	**1122**	786	**1416**	**1675**	**1946**
41	158	**162**	**355**	293	**596**	492	**882**	632	**1156**	802	**1496**	**1765**	**2058**
42	161	**168**	**367**	319	**626**	496	**907**	670	**1209**	903	**1526**	**1843**	**2149**
43	**171**	168	**383**	323	**666**	520	**947**	670	**1256**	903	**1597**	**1905**	**2224**
44	179	**186**	**396**	331	**680**	548	**992**	696	**1336**	903	**1653**	**1994**	**2319**
45	182	**186**	**407**	347	**698**	565	**996**	737	**1366**	926	**1723**	**2079**	**2415**
46	186	**189**	**421**	366	**723**	592	**1064**	797	**1408**	985	**1794**	**2155**	**2507**
47	**189**	**189**	**438**	378	**754**	611	**1088**	797	**1459**	985	**1847**	**2244**	**2600**
48	**201**	189	**450**	413	**763**	632	**1119**	857	**1531**	1054	**1924**	**2336**	**2732**
49	**203**	196	**464**	415	**803**	634	**1167**	893	**1592**	1074	**1989**	**2397**	**2828**
50	206	**213**	**478**	415	**830**	663	**1203**	893	**1639**	1113	**2041**	**2492**	**2893**
51	206	**213**	**490**	421	**850**	692	**1230**	917	**1662**	1161	**2132**	**2566**	**3006**
52	**217**	213	**504**	455	**883**	709	**1274**	1003	**1734**	1231	**2203**	**2686**	**3134**
53	**219**	216	**515**	455	**894**	716	**1340**	1003	**1808**	1241	**2234**	**2713**	**3261**
54	209	**233**	**534**	477	**932**	760	**1359**	1019	**1856**	1288	**2356**	**2805**	**3339**
55	230	**243**	**546**	483	**956**	760	**1421**	1082	**1896**	1325	**2429**	**2935**	**3452**
56	230	**243**	**558**	515	**982**	784	**1431**	1104	**2003**	1358	**2444**	**3021**	**3551**
57	249	**261**	**574**	515	**1007**	846	**1488**	1136	**2024**	1479	**2554**	**3119**	**3651**
58	245	**261**	**594**	539	**1035**	846	**1554**	1166	**2043**	1479	**2650**	**3187**	**3795**
59	254	**266**	**609**	544	**1063**	849	**1564**	1223	**2136**	1509	**2733**	**3297**	**3889**
60	261	**273**	**618**	568	**1094**	904	**1631**	1242	**2232**	1577	**2796**	**3420**	**4090**
61	266	**274**	**630**	620	**1128**	904	**1667**	1258	**2266**	1615	**2868**	**3525**	**4158**
62	269	**283**	**657**	620	**1150**	934	**1715**	1306	**2345**	1680	**2977**	**3636**	**4313**
63	281	**297**	**670**	620	**1178**	967	**1781**	1380	**2376**	1680	**3056**	**3690**	**4355**
64	278	**297**	**684**	625	**1206**	985	**1804**	1430	**2452**	1769	**3097**	**3820**	**4514**
65	290	**314**	**694**	630	**1216**	997	**1868**	1430	**2492**	1786	**3219**	**3932**	**4581**
66	299	**314**	**718**	666	**1261**	1050	**1874**	1476	**2543**	1857	**3279**	**4004**	**4769**
67	294	**314**	**735**	666	**1299**	1072	**1954**	1482	**2638**	1868	**3399**	**4081**	**4942**
68	306	**314**	**746**	685	**1330**	1087	**1983**	1538	**2693**	1940	**3453**	**4212**	**4995**
69	306	**324**	**765**	698	**1351**	1112	**2028**	1588	**2746**	1965	**3520**	**4317**	**5127**
70	314	**325**	**779**	716	**1378**	1150	**2094**	1633	**2838**	2130	**3588**	**4464**	**5276**

Table 2.6: Squared ℓ^2 separation distance found using periodic designs (PD) and the ESE algorithm (ESE). (*Table continued on next page*).

n	3 dim		4 dim		5 dim		6 dim		7 dim		8 dim	9 dim	10 dim
	ESE	Per	ESE	Per	ESE	Per	ESE	Per	ESE	Per	ESE	ESE	ESE
71	314	**325**	**793**	716	**1413**	1150	**2141**	1644	**2871**	2130	**3749**	**4548**	**5437**
72	314	**341**	**810**	750	**1430**	1203	**2136**	1768	**2960**	2177	**3810**	**4666**	**5556**
73	329	**350**	**834**	759	**1462**	1229	**2197**	1768	**3042**	2206	**3932**	**4776**	**5661**
74	341	**350**	**842**	767	**1512**	1229	**2291**	1774	**3120**	2244	**3941**	**4915**	**5817**
75	341	**350**	**867**	771	**1530**	1274	**2303**	1862	**3157**	2295	**4073**	**5006**	**5937**
76	341	**363**	**882**	813	**1569**	1300	**2387**	1935	**3218**	2375	**4178**	**5179**	**6111**
77	341	**363**	**894**	823	**1591**	1308	**2433**	1947	**3323**	2403	**4266**	**5222**	**6272**
78	371	**387**	**910**	844	**1621**	1382	**2479**	2014	**3387**	2505	**4390**	**5385**	**6384**
79	374	**387**	**927**	848	**1639**	1382	**2498**	2037	**3474**	2525	**4465**	**5535**	**6466**
80	374	**403**	**949**	873	**1691**	1395	**2554**	2037	**3550**	2590	**4565**	**5577**	**6653**
81	381	**406**	**963**	916	**1730**	1406	**2648**	2064	**3619**	2642	**4679**	**5748**	**6780**
82	374	**406**	**989**	938	**1742**	1475	**2680**	2141	**3669**	2753	**4719**	**5859**	**6935**
83	374	**417**	**1002**	940	**1762**	1501	**2696**	2141	**3723**	2767	**4848**	**5976**	**7094**
84	406	**426**	**1021**	967	**1818**	1534	**2790**	2229	**3870**	2838	**4920**	**6119**	**7256**
85	413	**426**	**1043**	967	**1866**	1552	**2819**	2232	**3919**	2874	**5032**	**6212**	**7357**
86	413	**428**	**1053**	967	**1882**	1573	**2875**	2375	**3958**	3103	**5164**	**6346**	**7532**
87	413	**428**	**1073**	976	**1934**	1598	**2913**	2375	**4095**	3103	**5225**	**6469**	**7639**
88	434	**437**	**1086**	1050	**1954**	1685	**2975**	2398	**4166**	3183	**5340**	**6660**	**7877**
89	426	**443**	**1102**	1050	**1990**	1690	**3067**	2400	**4176**	3183	**5450**	**6750**	**7950**
90	446	**481**	**1134**	1060	**2027**	1710	**3104**	2516	**4308**	3190	**5576**	**6901**	**8128**
91	434	**481**	**1134**	1089	**2031**	1748	**3143**	2516	**4379**	3234	**5626**	**6950**	**8330**
92	446	**481**	**1149**	1089	**2100**	1805	**3216**	2599	**4428**	3277	**5758**	**7067**	**8442**
93	446	**481**	**1171**	1098	**2130**	1813	**3283**	2604	**4512**	3361	**5832**	**7342**	**8601**
94	470	**481**	**1199**	1124	**2169**	1881	**3348**	2747	**4581**	3474	**6007**	**7436**	**8774**
95	**482**	481	**1219**	1135	**2206**	1901	**3335**	2747	**4703**	3531	**6064**	**7469**	**8877**
96	486	**509**	1250	**1261**	**2227**	1965	**3451**	2769	**4808**	3639	**6222**	**7645**	**9146**
97	474	**515**	1258	**1261**	**2299**	1965	**3514**	2817	**4848**	3639	**6304**	**7781**	**9379**
98	485	**531**	**1283**	1261	**2299**	1965	**3560**	2850	**4936**	3690	**6376**	**7896**	**9381**
99	489	**531**	**1298**	1261	**2338**	2009	**3628**	2878	**4999**	3731	**6448**	**8023**	**9617**
100	494	**554**	**1305**	1261	**2401**	2053	**3648**	3000	**5040**	3903	**6617**	**8228**	**9835**
105	521	**563**	**1395**	1329									
110	566	**626**	**1510**	1414									
115	594	**650**	**1591**	1499									
120	629	**702**	**1708**	1603									
125	629	**713**	**1798**	1750									
130	693	**766**	**1906**	1872									
135	729	**780**	**1995**	1909									
140	758	**845**	**2103**	2089									
145	779	**894**	2185	**2225**									
150	825	**934**	**2310**	2278									
155	842	**986**	2365	**2367**									
160	854	**1002**	2486	**2548**									
165	904	**1041**	2582	**2648**									
170	914	**1121**	2659	**2869**									
175	965	**1132**	2771	**2902**									
180	1011	**1208**	2897	**3077**									
185	1026	**1224**	2970	**3267**									
190	1061	**1298**	3094	**3325**									
195	1086	**1350**	3210	**3492**									
200	1106	**1371**	3257	**3596**									
205	1166	**1425**	3273	**3708**									
210	1196	**1473**	3377	**3767**									
215	1229	**1538**	3476	**3983**									
220	1259	**1544**	3543	**4159**									
225	1293	**1611**	3661	**4292**									
230	1329	**1646**	3703	**4326**									
235	1305	**1706**	3815	**4532**									
240	1350	**1806**	3893	**5061**									
245	1397	**1891**	3986	**5061**									
250	1412	**1901**	3990	**5075**									
255	1417	**1923**	4100	**5122**									
260	1445	**1971**	4164	**5236**									
265	1449	**2021**	4182	**5519**									
270	1464	**2144**	4361	**5656**									
275	1478	**2150**	4487	**5746**									
280	1493	**2184**	4388	**6023**									
285	1501	**2209**	4607	**6094**									
290	1476	**2269**	4722	**6380**									
295	1526	**2354**	4726	**6590**									
300	1542	**2409**	4898	**6604**									

Table 2.6: Squared ℓ^2 separation distance found using periodic designs (PD) and the ESE algorithm (ESE). (*Continued*).

n	2 dim		3 dim		4 dim		5 dim		6 dim	7 dim	8 dim	9 dim	10 dim
	ESE	Per	ESE	Per	ESE	Per	ESE	Per	ESE	ESE	ESE	ESE	ESE
2	**0.500**	**0.500**	**0.333**	**0.333**	**0.250**	**0.250**	**0.200**	**0.200**	**0.167**	**0.143**	**0.125**	**0.111**	**0.100**
3	**0.900**	**0.900**	**0.611**	**0.611**	**0.386**	0.450	**0.321**	0.362	**0.250**	**0.230**	**0.193**	**0.200**	**0.151**
4	**1.000**	**1.000**	**0.642**	**0.642**	**0.454**	0.489	**0.367**	0.382	**0.300**	**0.260**	**0.225**	**0.201**	**0.180**
5	**1.298**	1.390	**0.727**	0.891	**0.509**	0.658	**0.401**	0.527	**0.336**	**0.287**	**0.250**	**0.222**	**0.200**
6	**1.521**	**1.521**	**0.794**	0.800	**0.561**	0.594	**0.431**	0.476	**0.358**	**0.307**	**0.268**	**0.238**	**0.215**
7	**1.598**	**1.598**	**0.867**	0.975	**0.599**	0.694	**0.464**	0.532	**0.376**	**0.322**	**0.282**	**0.250**	**0.225**
8	**1.804**	1.879	**0.921**	0.960	**0.619**	0.696	**0.488**	0.538	**0.398**	**0.334**	**0.292**	**0.260**	**0.234**
9	**1.935**	**1.935**	**0.971**	1.052	**0.660**	0.742	**0.504**	0.567	**0.414**	**0.349**	**0.301**	**0.267**	**0.240**
10	**2.066**	**2.066**	**1.020**	1.085	**0.686**	0.744	**0.515**	0.556	**0.425**	**0.360**	**0.311**	**0.273**	**0.246**
11	**2.196**	2.279	**1.069**	1.137	**0.709**	0.785	**0.536**	0.612	**0.434**	**0.369**	**0.319**	**0.281**	**0.250**
12	**2.273**	**2.273**	**1.095**	1.163	**0.724**	0.785	**0.551**	0.589	**0.441**	**0.375**	**0.326**	**0.287**	**0.256**
13	**2.401**	2.487	**1.128**	1.191	**0.746**	0.825	**0.563**	0.632	**0.453**	**0.381**	**0.331**	**0.292**	**0.261**
14	**2.476**	**2.476**	**1.167**	1.252	**0.762**	0.829	**0.575**	0.635	**0.462**	**0.385**	**0.335**	**0.296**	**0.265**
15	**2.578**	2.643	**1.194**	1.255	**0.775**	0.818	**0.583**	0.636	**0.470**	**0.393**	**0.339**	**0.299**	**0.268**
16	**2.666**	2.683	**1.221**	1.290	**0.791**	0.848	**0.589**	0.642	**0.477**	**0.398**	**0.341**	**0.302**	**0.271**
17	**2.721**	**2.721**	**1.246**	1.340	**0.805**	0.866	**0.600**	0.656	**0.483**	**0.404**	**0.347**	**0.305**	**0.273**
18	**2.819**	2.848	**1.271**	1.337	**0.816**	0.875	**0.609**	0.655	**0.488**	**0.408**	**0.350**	**0.307**	**0.275**
19	**2.890**	2.984	**1.292**	1.374	**0.827**	0.895	**0.615**	0.667	**0.492**	**0.413**	**0.354**	**0.310**	**0.277**
20	**2.959**	2.962	**1.318**	1.394	**0.835**	0.907	**0.620**	0.681	**0.496**	**0.416**	**0.358**	**0.313**	**0.278**
21	**3.025**	3.033	**1.339**	1.408	**0.847**	0.914	**0.625**	0.671	**0.501**	**0.419**	**0.361**	**0.316**	**0.281**
22	**3.070**	**3.070**	**1.357**	1.426	**0.856**	0.922	**0.632**	0.687	**0.505**	**0.422**	**0.363**	**0.318**	**0.283**
23	**3.138**	3.159	**1.377**	1.454	**0.868**	0.925	**0.638**	0.693	**0.510**	**0.425**	**0.366**	**0.321**	**0.285**
24	**3.197**	3.201	**1.396**	1.458	**0.875**	0.931	**0.644**	0.677	**0.513**	**0.427**	**0.368**	**0.323**	**0.287**
25	**3.254**	3.293	**1.412**	1.485	**0.884**	0.940	**0.648**	0.701	**0.516**	**0.430**	**0.370**	**0.324**	**0.289**
26	**3.309**	3.332	**1.428**	1.480	**0.891**	0.947	**0.653**	0.707	**0.518**	**0.432**	**0.372**	**0.326**	**0.290**
27	**3.360**	3.383	**1.442**	1.499	**0.898**	0.957	**0.657**	0.708	**0.521**	**0.435**	**0.373**	**0.328**	**0.292**
28	**3.405**	3.420	**1.454**	1.503	**0.906**	0.961	**0.660**	0.712	**0.524**	**0.437**	**0.375**	**0.329**	**0.293**
29	**3.458**	3.539	**1.468**	1.543	**0.912**	0.978	**0.664**	0.716	**0.527**	**0.439**	**0.376**	**0.330**	**0.294**
30	**3.505**	3.515	**1.481**	1.528	**0.919**	0.974	**0.667**	0.716	**0.530**	**0.441**	**0.378**	**0.331**	**0.295**
31	**3.543**	3.550	**1.493**	1.563	**0.925**	0.976	**0.671**	0.719	**0.533**	**0.443**	**0.380**	**0.333**	**0.296**
32	**3.589**	3.623	**1.505**	1.562	**0.931**	0.996	**0.674**	0.729	**0.535**	**0.444**	**0.381**	**0.334**	**0.297**
33	**3.636**	3.642	**1.517**	1.588	**0.935**	0.990	**0.678**	0.732	**0.537**	**0.446**	**0.383**	**0.335**	**0.298**
34	**3.676**	3.713	**1.528**	1.565	**0.941**	1.005	**0.682**	0.735	**0.540**	**0.447**	**0.384**	**0.336**	**0.299**
35	**3.716**	3.786	**1.539**	1.601	**0.946**	1.003	**0.685**	0.731	**0.542**	**0.449**	**0.385**	**0.337**	**0.300**
36	**3.758**	3.774	**1.549**	1.600	**0.950**	1.005	**0.688**	0.734	**0.543**	**0.450**	**0.386**	**0.338**	**0.301**
37	**3.794**	3.819	**1.558**	1.599	**0.956**	1.019	**0.691**	0.736	**0.545**	**0.452**	**0.387**	**0.339**	**0.301**
38	**3.828**	**3.828**	**1.568**	1.623	**0.959**	1.023	**0.694**	0.746	**0.547**	**0.453**	**0.388**	**0.340**	**0.302**
39	**3.868**	3.879	**1.578**	1.646	**0.965**	1.025	**0.696**	0.742	**0.548**	**0.455**	**0.389**	**0.341**	**0.303**
40	**3.906**	3.918	**1.587**	1.636	**0.968**	1.019	**0.699**	0.742	**0.550**	**0.456**	**0.390**	**0.341**	**0.303**
41	**3.939**	4.009	**1.596**	1.639	**0.971**	1.033	**0.701**	0.751	**0.551**	**0.457**	**0.391**	**0.342**	**0.304**
42	**3.974**	3.974	**1.604**	1.658	**0.975**	1.031	**0.703**	0.742	**0.552**	**0.458**	**0.392**	**0.343**	**0.305**
43	**4.007**	4.045	**1.612**	1.675	**0.979**	1.042	**0.705**	0.752	**0.554**	**0.460**	**0.393**	**0.344**	**0.306**
44	**4.029**	**4.029**	**1.621**	1.670	**0.983**	1.040	**0.708**	0.754	**0.555**	**0.461**	**0.394**	**0.344**	**0.306**
45	**4.063**	4.074	**1.628**	1.678	**0.986**	1.044	**0.710**	0.752	**0.557**	**0.462**	**0.394**	**0.345**	**0.307**
46	**4.096**	4.115	**1.636**	1.693	**0.990**	1.044	**0.712**	0.753	**0.559**	**0.463**	**0.395**	**0.346**	**0.307**
47	**4.130**	4.179	**1.643**	1.695	**0.993**	1.055	**0.714**	0.761	**0.560**	**0.464**	**0.396**	**0.346**	**0.308**
48	**4.160**	4.206	**1.650**	1.699	**0.997**	1.052	**0.716**	0.759	**0.561**	**0.464**	**0.397**	**0.347**	**0.308**
49	**4.187**	**4.187**	**1.657**	1.711	**1.001**	1.059	**0.718**	0.762	**0.563**	**0.465**	**0.398**	**0.347**	**0.309**
50	**4.216**	4.254	**1.665**	1.713	**1.004**	1.058	**0.720**	0.765	**0.564**	**0.466**	**0.398**	**0.348**	**0.309**
51	**4.246**	4.280	**1.671**	1.729	**1.007**	1.063	**0.721**	0.768	**0.566**	**0.467**	**0.399**	**0.348**	**0.310**
52	**4.273**	4.277	**1.678**	1.730	**1.010**	1.062	**0.723**	0.765	**0.567**	**0.468**	**0.400**	**0.349**	**0.310**
53	**4.302**	4.343	**1.685**	1.734	**1.013**	1.072	**0.725**	0.771	**0.568**	**0.468**	**0.400**	**0.349**	**0.310**
54	**4.331**	4.341	**1.690**	1.739	**1.016**	1.070	**0.726**	0.769	**0.569**	**0.469**	**0.401**	**0.350**	**0.311**
55	**4.355**	4.413	**1.697**	1.755	**1.018**	1.073	**0.728**	0.773	**0.570**	**0.470**	**0.401**	**0.350**	**0.311**
56	**4.382**	4.404	**1.703**	1.756	**1.022**	1.071	**0.729**	0.772	**0.571**	**0.470**	**0.402**	**0.351**	**0.312**
57	**4.404**	4.427	**1.708**	1.760	**1.024**	1.079	**0.731**	0.776	**0.572**	**0.471**	**0.403**	**0.351**	**0.312**
58	**4.431**	4.437	**1.714**	1.763	**1.027**	1.076	**0.732**	0.776	**0.573**	**0.472**	**0.403**	**0.352**	**0.312**
59	**4.458**	4.498	**1.719**	1.777	**1.030**	1.087	**0.734**	0.780	**0.574**	**0.473**	**0.404**	**0.352**	**0.313**
60	**4.482**	4.490	**1.725**	1.772	**1.032**	1.079	**0.735**	0.777	**0.575**	**0.473**	**0.404**	**0.353**	**0.313**
61	**4.499**	4.530	**1.731**	1.778	**1.034**	1.087	**0.736**	0.776	**0.576**	**0.474**	**0.405**	**0.353**	**0.314**
62	**4.526**	4.576	**1.736**	1.786	**1.036**	1.087	**0.738**	0.781	**0.576**	**0.475**	**0.405**	**0.354**	**0.314**
63	**4.556**	4.576	**1.742**	1.789	**1.039**	1.094	**0.739**	0.783	**0.577**	**0.475**	**0.406**	**0.354**	**0.314**
64	**4.573**	4.590	**1.746**	1.794	**1.041**	1.087	**0.740**	0.784	**0.578**	**0.476**	**0.406**	**0.354**	**0.315**
65	**4.595**	4.599	**1.751**	1.802	**1.043**	1.095	**0.742**	0.786	**0.579**	**0.477**	**0.407**	**0.355**	**0.315**
66	**4.619**	4.635	**1.757**	1.804	**1.045**	1.093	**0.742**	0.785	**0.580**	**0.477**	**0.407**	**0.355**	**0.315**
67	**4.636**	4.642	**1.761**	1.812	**1.047**	1.100	**0.744**	0.787	**0.581**	**0.478**	**0.407**	**0.355**	**0.315**
68	**4.661**	4.681	**1.766**	1.819	**1.049**	1.096	**0.745**	0.790	**0.581**	**0.478**	**0.407**	**0.356**	**0.316**
69	**4.683**	4.713	**1.771**	1.818	**1.052**	1.104	**0.746**	0.791	**0.582**	**0.479**	**0.408**	**0.356**	**0.316**
70	**4.703**	4.710	**1.775**	1.818	**1.053**	1.100	**0.747**	0.790	**0.583**	**0.480**	**0.408**	**0.356**	**0.316**

Table 2.7: Audze-Eglais values found using periodic designs (PD) and the ESE algorithm (ESE). (*Table continued on next page*).

n	2 dim		3 dim		4 dim		5 dim		6 dim	7 dim	8 dim	9 dim	10 dim
	ESE	Per	ESE	Per	ESE	Per	ESE	Per	ESE	ESE	ESE	ESE	ESE
71	**4.727**	4.742	**1.780**	1.831	**1.055**	1.108	**0.747**	0.795	**0.584**	**0.480**	**0.409**	**0.357**	**0.317**
72	**4.743**	4.746	**1.784**	1.829	**1.057**	1.103	**0.749**	0.791	**0.584**	**0.481**	**0.409**	**0.357**	**0.317**
73	**4.763**	4.781	**1.789**	1.836	**1.059**	1.106	**0.749**	0.794	**0.585**	**0.481**	**0.409**	**0.357**	**0.317**
74	**4.781**	4.820	**1.793**	1.842	**1.061**	1.112	**0.751**	0.795	**0.586**	**0.482**	**0.410**	**0.358**	**0.317**
75	**4.803**	4.817	**1.796**	1.847	**1.063**	1.112	**0.752**	0.797	**0.586**	**0.482**	**0.410**	**0.358**	**0.318**
76	**4.823**	4.828	**1.801**	1.850	**1.064**	1.110	**0.753**	0.796	**0.587**	**0.483**	**0.410**	**0.358**	**0.318**
77	**4.838**	4.853	**1.805**	1.851	**1.066**	1.112	**0.755**	0.799	**0.587**	**0.483**	**0.411**	**0.359**	**0.318**
78	**4.863**	4.883	**1.809**	1.847	**1.068**	1.114	**0.755**	0.795	**0.588**	**0.484**	**0.411**	**0.359**	**0.318**
79	**4.882**	4.934	**1.812**	1.863	**1.070**	1.119	**0.756**	0.801	**0.589**	**0.484**	**0.411**	**0.359**	**0.318**
80	**4.895**	4.922	**1.816**	1.864	**1.071**	1.117	**0.757**	0.799	**0.589**	**0.484**	**0.412**	**0.359**	**0.319**
81	**4.920**	4.942	**1.820**	1.869	**1.072**	1.120	**0.758**	0.802	**0.590**	**0.485**	**0.412**	**0.360**	**0.319**
82	**4.936**	4.944	**1.824**	1.862	**1.074**	1.120	**0.759**	0.801	**0.590**	**0.485**	**0.413**	**0.360**	**0.319**
83	**4.949**	4.949	**1.827**	1.879	**1.076**	1.126	**0.760**	0.805	**0.591**	**0.486**	**0.413**	**0.360**	**0.319**
84	**4.968**	4.992	**1.831**	1.876	**1.077**	1.122	**0.761**	0.802	**0.591**	**0.486**	**0.413**	**0.360**	**0.320**
85	**4.985**	5.014	**1.834**	1.879	**1.079**	1.124	**0.761**	0.804	**0.592**	**0.486**	**0.414**	**0.360**	**0.320**
86	**5.003**	5.014	**1.838**	1.882	**1.081**	1.125	**0.762**	0.804	**0.592**	**0.487**	**0.414**	**0.361**	**0.320**
87	**5.019**	5.060	**1.842**	1.891	**1.082**	1.130	**0.763**	0.808	**0.593**	**0.487**	**0.414**	**0.361**	**0.320**
88	**5.034**	5.047	**1.845**	1.885	**1.083**	1.130	**0.764**	0.805	**0.594**	**0.487**	**0.414**	**0.361**	**0.320**
89	**5.056**	5.096	**1.848**	1.895	**1.085**	1.133	**0.765**	0.810	**0.594**	**0.488**	**0.415**	**0.361**	**0.321**
90	5.070	**5.063**	**1.852**	1.885	**1.086**	1.131	**0.766**	0.807	**0.594**	**0.488**	**0.415**	**0.361**	**0.321**
91	**5.086**	5.113	**1.854**	1.890	**1.088**	1.134	**0.766**	0.809	**0.595**	**0.489**	**0.415**	**0.362**	**0.321**
92	**5.104**	5.114	**1.858**	1.902	**1.089**	1.135	**0.767**	0.809	**0.595**	**0.489**	**0.416**	**0.362**	**0.321**
93	**5.119**	5.122	**1.861**	1.903	**1.090**	1.136	**0.768**	0.810	**0.596**	**0.489**	**0.416**	**0.362**	**0.321**
94	**5.130**	5.143	**1.864**	1.900	**1.092**	1.138	**0.769**	0.810	**0.596**	**0.490**	**0.416**	**0.362**	**0.321**
95	**5.151**	5.177	**1.867**	1.909	**1.093**	1.138	**0.769**	0.813	**0.597**	**0.490**	**0.416**	**0.362**	**0.322**
96	**5.163**	5.183	**1.870**	1.910	**1.094**	1.139	**0.770**	0.811	**0.597**	**0.490**	**0.417**	**0.363**	**0.322**
97	**5.177**	5.179	**1.872**	1.915	**1.096**	1.138	**0.771**	0.814	**0.598**	**0.490**	**0.417**	**0.363**	**0.322**
98	**5.198**	5.223	**1.876**	1.915	**1.097**	1.142	**0.771**	0.812	**0.598**	**0.491**	**0.417**	**0.363**	**0.322**
99	**5.211**	5.244	**1.879**	1.923	**1.098**	1.143	**0.772**	0.815	**0.599**	**0.491**	**0.417**	**0.363**	**0.322**
100	5.223	**5.221**	**1.882**	1.921	**1.099**	1.143	**0.773**	0.812	**0.599**	**0.491**	**0.418**	**0.363**	**0.322**

Table 2.7: Audze-Eglais values found using periodic designs (PD) and the ESE algorithm (ESE). (*Continued*).

Chapter 3

Bounds for maximin Latin hypercube designs

> Genius may have its limitations, but stupidity is not thus handicapped.
>
> (Elbert Hubbard)

3.1 Introduction

Latin hypercube designs form a class of designs that are often used for finding approximations of the input-output behavior of deterministic computer simulation models on a box-constrained domain. This type of simulation model is used in engineering, logistics, and finance to analyze and optimize the design of products or processes (see Driessen (2006) and Stinstra (2006)). The reason for approximating these models is that a computer simulation run is usually rather time-consuming to perform. This makes the model impractical when it comes to obtaining insight in the underlying process or in optimizing its parameters. A common approach to overcome this problem is to determine a metamodel that approximates the relation between the input and output parameters of the computer simulation model. Such a metamodel is based on the information obtained from a limited number of simulation runs; see, e.g., Montgomery (2009), Sacks et al. (1989a), Sacks et al. (1989b), Jones et al. (1998), Myers (1999), Booker et al. (1999), and Den Hertog and Stehouwer (2002). The quality of the metamodel depends, among others, on the choice of the simulation runs. The inputs of a simulation run can be represented as a vector containing the values of all input parameters or factors. When the simulation model has k input parameters, the simulation runs can therefore be represented by points in the k-dimensional input space. A set of these design points is called a *design* and we denote the number of design points by n. As designs can be scaled to

any box-constrained domain, the designs in this chapter are without loss of generality constructed on a hypercube.

As is recognized by several authors, a design should at least satisfy the following two criteria (see Johnson et al. (1990) and Morris and Mitchell (1995)). Firstly, the design should be *space-filling*. This means that the whole design space should be well-represented by the design points. To achieve this characteristic, we consider the maximin criterion, which states that the points should be chosen such that the minimal distance between any two points is maximal. This minimal distance is called the *separation distance* of the design. The maximin criterion is defined for different distance measures. In this chapter, we use the ℓ^∞, ℓ^1, and ℓ^2-measures. We remark that Johnson et al. (1990) explicitly link the distance measure to the covariance function in a spatial process model. Thus, ℓ^1 and ℓ^2 are related to the Ornstein-Uhlenbeck and Gaussian covariance functions. The ℓ^∞-measure does not relate to an accepted covariance function, nevertheless, it is of theoretical value. Other space-filling designs are minimax, integrated mean squared error, and maximum entropy designs. A good survey of these designs can be found in the book of Santner et al. (2003). Secondly, the design should be *non-collapsing*. When a parameter has (almost) no influence on the output, then two design points that differ only in this parameter can be considered as the same point. As each point is time-consuming to evaluate, this situation should be avoided. Therefore, non-collapsingness requires that for each parameter the values in all design points should be distinct.

Latin hypercube designs (LHDs) are a particular class of non-collapsing designs. For LHDs on the $[0, n-1]^k$ hypercube, the values of the input parameters are chosen from the set $\{0, 1, \ldots, n-1\}$ and for each input parameter each value in this set is chosen exactly once. More formally, we can describe a k-dimensional LHD of n design points as a set of n points $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ik})$ with $\{x_{ij} | i = 1, 2, \ldots, n\} = \{0, 1, \ldots, n-1\}$ for all j.

Finding maximin LHDs can be time-consuming for larger values of k and n. Therefore, most results in this field concern approximate maximin LHDs. With "approximate maximin", we mean that it is not guaranteed that the separation distance is maximal. For some cases, however, we do have proof of maximality and thus know "optimal" maximin LHDs. For the distance measures ℓ^∞ and ℓ^1 for instance, Van Dam et al. (2007) derive general formulas for two-dimensional maximin LHDs. Furthermore, they obtain two-dimensional ℓ^2-maximin LHDs for $n \leq 70$ by using a branch-and-bound algorithm.

For approximate maximin LHDs more results are available. Van Dam et al. (2007) construct two-dimensional approximate ℓ^2-maximin LHDs for up to 1000 points by optimizing a periodic structure. Husslage et al. (2006) extend these results to more dimensions. Morris and Mitchell (1995) use a simulated annealing approach to obtain approximate ℓ^1- and ℓ^2-maximin LHDs for up to five dimensions and up to 12 points

and a few larger values. Jin et al. (2005) describe an enhanced stochastic evolutionary algorithm for finding space-filling LHDs. The maximin distance criterion is one of the criteria that they consider. Ye et al. (2000) use an exchange algorithm to obtain approximate maximin symmetric LHDs; they impose the symmetry property to reduce the computational effort.

In this chapter, we construct bounds for the separation distance of certain classes of maximin LHDs. These bounds are useful for assessing the quality of approximate maximin LHDs by comparing their separation distances with the corresponding upper bounds. Until now only upper bounds are known for the separation distance of certain classes of unrestricted maximin designs (by unrestricted design we mean any set of n points in the $[0, n-1]^k$ hypercube, i.e., there need not be a *Latin* hypercube structure). Oler (1961), for instance, gives an upper bound for two-dimensional unrestricted ℓ^2-maximin designs. Furthermore, Baer (1992) gives an upper bound for the separation distance of unrestricted ℓ^∞-maximin designs. The separation distance of maximin LHDs also satisfies these unrestricted bounds. By using some of the special properties of LHDs, we are able to find new and tighter bounds for maximin LHDs. Table 3.1 gives an overview of the classes of maximin LHDs treated in each section of this chapter. For these classes, different methods are used to determine the upper bounds. Within the methods, a variety of combinatorial optimization techniques are employed; namely, Mixed Integer Programming, the Traveling Salesman Problem, and the Graph Covering Problem. Besides these bounds, also a construction method is described for generating LHDs that meet Baer's bound for the ℓ^∞-distance measure for certain values of n.

	ℓ^2	ℓ^∞	ℓ^1
$k = 2$	Section 3.2.2		
$k = 3$		Section 3.3.3	
k large relative to n	Section 3.2.1	Section 3.3.1	Section 3.4
$n \approx m^k$ for $k, m \in \mathbb{N}$		Section 3.3.2	

Table 3.1: Overview of the classes of maximin LHDs treated in this chapter.

We realize that the results obtained for k large relative to n may not be of direct practical use. Still, it is important to have these theoretical results that indicate the boundaries for the separation distance. The other results in this chapter do concern LHDs where the number of points n is larger than k, which therefore are of more practical use.

This chapter is organized as follows. In Section 3.2, two methods are described that give bounds for ℓ^2-maximin LHDs. The first method is based on the average squared ℓ^2-distance, which is useful when k is large relative to n. The second method gives a bound for two-dimensional LHDs by partitioning the hypercube into smaller parts. Section 3.3 describes bounds for ℓ^∞-maximin LHDs. By reformulating the problem as an

edge covering problem in graphs, a bound is obtained for k-dimensional maximin LHDs. Furthermore, a method is described to construct LHDs meeting Baer's bound. Also a specific bound is given for three-dimensional maximin LHDs. The bound is found by projecting the three-dimensional hypercube onto two dimensions, and then partitioning it into strips. Section 3.4 gives a bound for ℓ^1-maximin LHDs that is based on the average ℓ^1-distance. This method is similar to the first method for the ℓ^2-distance. Finally, Section 3.5 gives some final remarks and conclusions.

3.2 Upper bounds for the ℓ^2-distance

3.2.1 Bounding by the average

We obtain a bound for the separation distance of an LHD from the fact that the minimal squared distance is at most equal to the average squared distance between points of an LHD.

Proposition 3.1. *Let D be an LHD of n points in k dimensions. Then the separation ℓ^2-distance d satisfies*

$$d^2 \leq \left\lfloor \frac{n(n+1)k}{6} \right\rfloor.$$

Proof. Let $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, with $\mathbf{x}_i = (x_{i1}, \ldots, x_{ik})$. The average squared distance among the points of D is

$$\frac{1}{\binom{n}{2}} \sum_{i>j} \sum_{h} (x_{ih} - x_{jh})^2 = \frac{1}{\binom{n}{2}} \sum_{h} \sum_{i>j} (x_{ih} - x_{jh})^2 = \frac{1}{\binom{n}{2}} \sum_{h} \sum_{i'>j'} (i' - j')^2 = \frac{n(n+1)k}{6}. \quad (3.1)$$

Since the squared separation distance is integer and at most equal to the average squared distance, rounding (3.1) finishes the proof. □

For fixed k, the separation distance of n points in a k-dimensional cube of side $n-1$ is at most order $n^{\frac{k-1}{k}}$ (this can be seen by comparing the total volume of n pairwise disjoint balls of diameter d to the total volume of the cube). It follows that the bound in Proposition 3.1 is not of the right order to be tight if k is fixed and n grows.

Note that if an LHD would have a separation distance close to the bound in Proposition 3.1, then the separation and average distances are about the same, i.e., all points are at approximately the same distance from each other. Supported by the fact that the maximal number of equidistant points in a k-dimensional space is $k+1$, we are led to believe that the bound in Proposition 3.1 can be close to tight only if k is large relative to n. Note that when n is fixed the upper bound is linear in k (ignoring the rounding). The following lemma also provides a lower bound that is linear in k, which shows that the bound is of the right order if n is fixed and k grows.

Lemma 3.1. *Let $d_{\max}(n,k)$ be the maximin ℓ^2-distance of an LHD of n points in k dimensions. Then $d_{\max}(n,k_1+k_2)^2 \geq d_{\max}(n,k_1)^2 + d_{\max}(n,k_2)^2$.*

Proof. Let $D_1 = \{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$ and $D_2 = \{\mathbf{y}_1,\ldots,\mathbf{y}_n\}$ be maximin LHDs in dimensions k_1 and k_2, respectively. Let $\mathbf{z}_i$ be the concatenation of $\mathbf{x}_i$ and $\mathbf{y}_i$, for $i = 1,\ldots,n$, then one obtains an LHD $D = \{\mathbf{z}_1,\ldots,\mathbf{z}_n\}$ of n points in dimension k_1+k_2 with squared separation distance at least $d_{\max}(n,k_1)^2 + d_{\max}(n,k_2)^2$. □

To show the strength of the bound in Proposition 3.1, we determine the maximin distance for LHDs of at most 5 points in any dimension. For this purpose, we first formulate the maximin problem as an integer programming problem.

Let $D = \{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$, with $\mathbf{x}_i = (x_{i1},\ldots,x_{ik})$, be an LHD. For each $j = 1,\ldots,k$ the map π sending i to $x_{ij}+1$ is a permutation of $\{1,2,\ldots,n\}$. Thus the maximin distance is the solution of the following problem:

$$\begin{array}{ll} \max & d \\ \text{s.t.} & \sum_{\pi\in S_n} k_\pi(\pi(i)-\pi(j))^2 \geq d^2, \quad \forall\ i>j \\ & \sum_{\pi\in S_n} k_\pi = k \\ & k_\pi \in \mathbb{N}_0, \quad \forall\ \pi \in S_n \end{array} \tag{3.2}$$

where S_n is the set of permutations of $\{1,2,\ldots,n\}$. Note that for any j, replacing x_{ij} by $n-1-x_{ij}$ for all i does not change the separation distance of the design. Thus we may restrict the set S_n to its first half when ordered lexicographically. This reduces the number of variables in the program to $n!/2$. Note also that we may assume that $k_{\pi^*} \geq 1$ for an arbitrary permutation π^*, because we may reorder the points of the design as we wish.

Consider now the cases $n = 3, 4$, and 5 (for $n = 2$ the bound is trivially attained).

Proposition 3.2. *For $n = 3$, the maximin ℓ^2-distance satisfies $d_{\max}(3,k)^2 = k + 3\lfloor \frac{k}{3} \rfloor$.*

Proof. The stated result follows from solving the above integer programming problem (3.2) by hand (the number of variables is 3). □

Dimension k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$d_{\max}(4,k)^2$	1	5	6	12	14	20	21	26	28	33	35	40	41	46	48	53	55	60	62
Upper bound	3	6	10	13	16	20	23	26	30	33	36	40	43	46	50	53	56	60	63

Table 3.2: Squared maximin ℓ^2-distance for LHDs of 4 points.

For $n = 4$ we have that $d^2 \leq \lfloor \frac{10k}{3} \rfloor$. We obtain Table 3.2 by solving the integer program (3.2) for $k \leq 19$ using the CPLEX-solver in AIMMS (Bisschop and Entriken 1993).

Proposition 3.3. *For $n = 4$, the maximin ℓ^2-distance satisfies $d_{\max}(4,k)^2 = \lfloor \frac{10k}{3} \rfloor - 1$ if $k \equiv 1$ or 5 (mod 6), $d_{\max}(4,k)^2 = \frac{10k}{3} - 2$ if $k \equiv 3$ (mod 6), and $d_{\max}(4,k)^2 = \lfloor \frac{10k}{3} \rfloor$ if k is even, except for the cases $k \leq 5$, $k = 7$, $k = 13$; see Table 3.2.*

Proof. By recursively applying Lemma 3.1 (always with $k_2 = 6$, and starting with $k_1 = 6, 8$, and 10) one obtains maximin LHDs for all even dimensions at least 6 meeting the upper bound.

For the odd dimensions the upper bound $\lfloor \frac{10k}{3} \rfloor$ cannot be attained. Even worse, for odd k divisible by 3, $d^2 = \frac{10k}{3} - 1$ cannot be attained. Suppose on the contrary that one of the above values is attained; then the minimal squared distance is at least $\frac{10k-3}{3}$. Fix the point that has the smallest average squared distance to the remaining points. Then this average squared distance equals $\frac{10k-e}{3}$, where e equals 0, 1, 2, or 3. Now let k_0 be the number of coordinates where the fixed point is 0 or 3, and let $k_1 = k - k_0$ be the number of coordinates where it is 1 or 2. It follows that the average squared distance of this point to the other points equals $\frac{1^2+2^2+3^2}{3}k_0 + \frac{(-1)^2+1^2+2^2}{3}k_1 = \frac{14}{3}k_0 + 2k_1 = \frac{10k-e}{3}$. It now follows that $k_0 = \frac{k}{2} - \frac{e}{8}$, and hence k should be even (and $e = 0$). Thus, if the upper bound is attained, then k cannot be odd, and for odd k divisible by 3, the gap with the upper bound is at least 2.

Now by recursively applying Lemma 3.1 (always with $k_2 = 6$, and starting with $k_1 = 9, 11$, and 19) one obtains maximin LHDs for all odd $k \geq 21$. □

For $n = 5$ we have that $d^2 \leq 5k$. We obtain Table 3.3 by solving the integer program (3.2) for $k \leq 14$ using the CPLEX-solver in AIMMS (Bisschop and Entriken 1993).

Dimension k	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$d_{\max}(5,k)^2$	1	5	11	15	24	27	32	40	43	50	54	60	64	70
Upper bound	5	10	15	20	25	30	35	40	45	50	55	60	65	70

Table 3.3: Squared maximin ℓ^2-distance for LHDs of 5 points.

Proposition 3.4. *For $n = 5$, the maximin ℓ^2-distance satisfies $d_{\max}(5,k)^2 = 5k - 1$ if k is odd, and $d_{\max}(5,k)^2 = 5k$ if k is even, except for the cases $k \leq 4$, $k = 6$, $k = 7$, $k = 9$. For these exceptions, see Table 3.3.*

Proof. We claim that the bound $5k$ can only be attained for even k. Indeed, if this bound is attained, then all points of the design are at equal distance. Fix a point, let k_0 be the number of coordinates where this point is 0 or 4, let k_1 be the number of coordinates where it is 1 or 3, and let k_2 be the number of coordinates where it is 2. It follows that the average squared distance of this point to the other points equals $\frac{30}{4}k_0 + \frac{15}{4}k_1 + \frac{10}{4}k_2 = 5k$. Since $k_0 + k_1 + k_2 = k$, it follows that $3k_1 + 4k_2 = 2k$, and hence k_1 is always even. We claim that this implies that the distance between any two points must be even, and hence that k must be even. To prove the claim, consider two points, and let k_{ee} and k_{oo} be the number of coordinates where both points are even and odd, respectively. Also, let k_{eo} and k_{oe} be the number of coordinates where the first point is even and the second one is odd, and the other way around, respectively. From the above it follows that both

$k_{oe} + k_{oo}$ and $k_{eo} + k_{oo}$ are even (k_1 is even), and hence $k_{oe} + k_{eo}$ is even. But then the distance between the two points is even, which proves the claim. Thus we may conclude that the bound $d^2 \leq 5k$ cannot be attained for odd k.

Besides the maximin designs obtained from integer programming, we obtain maximin designs in even dimensions by recursively applying Lemma 3.1 with k_1 and k_2 both even and at least 8. Then maximin LHDs for other odd dimensions are obtained by applying Lemma 3.1 with $k_1 = 5$ and k_2 even and at least 8. □

Also for $n = 6$ we computed the integer programming problem (3.2) for some small values of k; see Table 3.4. For some values of k, we only obtain a lower bound.

Dimension k	1	2	3	4	5	6	7	8	9	10	11	12	13
$d_{\max}(6,k)^2$	1	5	14	22	32	40	≥ 47	≥ 53	≥ 61	≥ 67	≥ 74	≥ 82	≥ 89
Upper bound	7	14	21	28	35	42	49	56	63	70	77	84	91

Table 3.4: Squared maximin ℓ^2-distance for LHDs of 6 points.

Note that Husslage (2006) have found better designs for $k = 8$ and 10 using simulated annealing. More specifically, $d_{\max}(6,8)^2 \geq 54$ and $d_{\max}(6,10)^2 \geq 68$.

3.2.2 Bounding by non-overlapping circles in two dimensions

In this section, we first introduce a new method for determining bounds on the ℓ^2-maximin distance for two-dimensional LHDs. Next, we compare the new bounds with an existing bound for unrestricted designs and with (approximate) maximin LHDs in Van Dam et al. (2007).

Methods to determine upper bounds

To find a bound for two-dimensional LHDs, we first look at the more general class of unrestricted designs. An upper bound for the ℓ^2-maximin distance of unrestricted designs, derived with Oler's theorem (Oler 1961), is:

$$d \leq 1 + \sqrt{1 + (n-1)\frac{2}{\sqrt{3}}}.$$

For LHDs, the value of d^2 is always the sum of two squared integers. We can use this property to define a slightly tighter upper bound. The Oler bound for LHDs is obtained by rounding down

$$\left(1 + \sqrt{1 + (n-1)\frac{2}{\sqrt{3}}}\right)^2$$

to the nearest integer that can be written as the sum of two squared integers.

To determine a bound more tailored to the special characteristics of two-dimensional ℓ^2-maximin LHDs, we use the following properties. A two-dimensional LHD of n points can be represented by a sequence y that is a permutation of the set $\{0, 1, \ldots, n-1\}$. The points of the LHD are then given by $\{(x, y_x)|x = 0, \ldots, n-1\}$. We can depict a two-dimensional ℓ^2-maximin LHD with separation distance d by n non-overlapping circles with diameter d and their centers given by $\{(x, y_x)|x = 0, \ldots, n-1\}$. We call circles consecutive if they have consecutive x-values.

The general idea for the new bound is the following. First, determine for each d how much distance along the y-axis is at least needed to place $\lceil d \rceil$ consecutive non-overlapping circles with diameter d on the $\{0, 1, \ldots, \lceil d \rceil - 1\} \times \mathbb{N}_+$-grid. With this information, we can determine a lower bound for the distance along the y-axis necessary to place n non-collapsing points with separation distance d. The second step is to determine d_n that denotes the minimal d for which this distance is larger than $n-1$. By taking the largest sum of two squares that is strictly smaller than d_n^2, we have found an upper bound on the squared separation distance of two-dimensional ℓ^2-maximin LHDs of n points. In the remainder of this section, we describe these two steps in more detail.

For the first step, fix $x \in \{0, 1, \ldots, n - \lceil d \rceil\}$ and consider a subset of $\lceil d \rceil$ circles with consecutive x-values $x, \ldots, x + \lceil d \rceil - 1$ with y-values $y_x, \ldots, y_{x+\lceil d \rceil - 1}$. The distance along the x-axis between any of these circles is less than d. This implies that the y-value of any circle in this set influences the y-value of any other circle in the set, due to the non-overlapping criterion.

The first step is thus to determine the minimal distance along the y-axis necessary to place $\lceil d \rceil$ consecutive non-overlapping circles with diameter d. This minimal distance is independent of the fixed value x and equal to $Y(d)$ in the following problem:

$$\begin{aligned} Y(d) = \quad & \min \quad (\max\{y_1, \ldots, y_{\lceil d \rceil}\} - \min\{y_1, \ldots, y_{\lceil d \rceil}\}) \\ & \text{s.t.} \quad \|(k, y_k) - (l, y_l)\| \geq d \quad \forall\, k, l \in \{1, \ldots, \lceil d \rceil\}, k \neq l \\ & \quad\; y \in \mathbb{N}_+^{\lceil d \rceil}, \end{aligned} \tag{3.3}$$

where $y_1, \ldots, y_{\lceil d \rceil}$ represent the y-values of $\lceil d \rceil$ consecutive circles.

For every $k, l \in \{1, \ldots, \lceil d \rceil\}$, $k \neq l$, we can calculate the minimal required difference between y_k and y_l. When we take, without loss of generality, $y_k \leq y_l$, applying Pythagoras' theorem gives:

$$y_l - y_k \geq \left\lceil \sqrt{d^2 - (l-k)^2} \right\rceil \geq 1.$$

In this result, we can round up because y_l and y_k must be integer. Furthermore, the last inequality holds because $(l-k)^2 < d^2$. This inequality implies that the points in the set are also non-collapsing. Thus, adding non-collapsingness constraints will not influence the value of $Y(d)$.

A drawback of solving problem (3.3) is that it is very time-consuming for larger values of d. Therefore, instead of solving problem (3.3), we propose to solve the following problem:

$$\begin{array}{rll} \widetilde{Y}(d) = & \min & y_{\sigma(\lceil d \rceil)} - y_{\sigma(1)} \\ & \text{s.t.} & \left\| (\sigma(i+1), y_{\sigma(i+1)}) - (\sigma(i), y_{\sigma(i)}) \right\| \geq d \quad \forall\, i \in \{1, \ldots, \lceil d \rceil - 1\} \\ & & y_{\sigma(1)} < y_{\sigma(2)} < \ldots < y_{\sigma(\lceil d \rceil)} \\ & & \sigma \in S_{\lceil d \rceil} \\ & & y \in \mathbb{N}_+^{\lceil d \rceil}, \end{array} \tag{3.4}$$

where $S_{\lceil d \rceil}$ is the set of all permutations of $\{1, \ldots, \lceil d \rceil\}$. For $\widetilde{Y}(d)$ the following holds.

Lemma 3.2. *For any d, $\widetilde{Y}(d) \leq Y(d)$.*

Proof. The difference between problems (3.3) and (3.4) is only in the constraints. Problem (3.4) only requires non-overlappingness of circles with consecutive y-values, whereas problem (3.3) requires that all circles are non-overlapping. As the constraints of problem (3.4) are thus a subset of the constraints of problem (3.3), $\widetilde{Y}(d)$ is at most $Y(d)$. □

Take for example $d = \sqrt{65}$. By total enumeration, we find that $Y(\sqrt{65}) = 49$ and $\widetilde{Y}(\sqrt{65}) = 46$. Figures 3.1 and 3.2 show two settings for y that attain these values. As can be seen, the solution to problem (3.3) gives a solution where all circles are non-overlapping. Problem (3.4), on the other hand, results in overlapping circles with non-consecutive y-values.

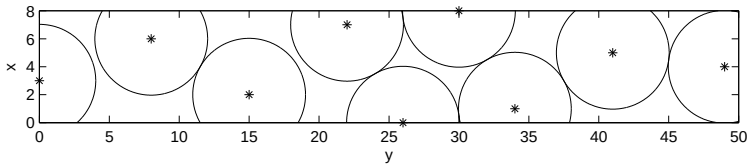


Figure 3.1: Setting for y that attains $Y(\sqrt{65}) = 49$.

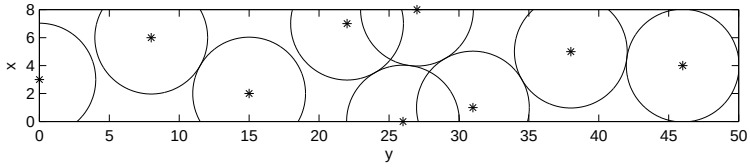


Figure 3.2: Setting for y that attains $\widetilde{Y}(\sqrt{65}) = 46$.

The following lemma shows that problem (3.4) can be reformulated.

Lemma 3.3. *$\widetilde{Y}(d)$ in problem (3.4) is equal to:*

$$\widetilde{Y}(d) = \min_{\sigma} \sum_{i=1}^{\lceil d \rceil - 1} \left\lceil \sqrt{d^2 - (\sigma(i+1) - \sigma(i))^2}. \right\rceil \tag{3.5}$$

Proof. Clearly, $y_{\sigma(\lceil d \rceil)} - y_{\sigma(1)} = \sum_{i=1}^{\lceil d \rceil - 1} (y_{\sigma(i+1)} - y_{\sigma(i)})$. For a given permutation σ, we must choose y such that $y_{\sigma(i+1)} - y_{\sigma(i)}$ is minimized for each $i \in \{1, \ldots, \lceil d \rceil - 1\}$ and satisfies the constraints. This way, we also minimize the sum of these terms. Applying Pythagoras' theorem gives that the minimal difference between $y_{\sigma(i+1)}$ and $y_{\sigma(i)}$ that satisfies the constraints is:

$$\left\lceil \sqrt{d^2 - (\sigma(i+1) - \sigma(i))^2} \right\rceil.$$

We can round up because $y_{\sigma(i+1)}$ and $y_{\sigma(i)}$ must be integer. Using this result, we can rewrite $\widetilde{Y}(d)$ as stated in (3.5). □

Determining $Y(d)$ and $\widetilde{Y}(d)$ can be done in a number of ways:

- Total enumeration. This can be done within reasonable time for $d \leq 10$. For larger values of d, computation time becomes very large, because the number of permutations is $\lceil d \rceil !$. We can use this method to determine both $Y(d)$ and $\widetilde{Y}(d)$.

- Mixed Integer Program (MIP). We can rewrite problem (3.3) as a MIP as follows:

$$\begin{array}{rlll} Y(d) = & \min y_{max} & & \\ & \text{s.t. } y_{max} \geq y_i & & \forall\ i \in \{1, \ldots, \lceil d \rceil\} \\ & (y_l - y_k) + M x_{kl} \geq \left\lceil \sqrt{d^2 - (l-k)^2} \right\rceil & & \forall\ k, l \in \{1, \ldots, \lceil d \rceil\}, k \neq l \\ & (y_k - y_l) + M(1 - x_{kl}) \geq \left\lceil \sqrt{d^2 - (l-k)^2} \right\rceil & & \forall\ k, l \in \{1, \ldots, \lceil d \rceil\}, k \neq l \\ & y \in \mathbb{R}_+^{\lceil d \rceil}, x_{kl} \in \{0, 1\} & & \forall\ k, l \in \{1, \ldots, \lceil d \rceil\}, \end{array}$$

  with $M = 2\lceil d \rceil$. Note that we do not have to require $y \in \mathbb{N}_+^{\lceil d \rceil}$, as the constraints enforce this property. We can also rewrite problem (3.4) as a MIP problem, but we omit this as the next method is more suitable.

- Traveling Salesman Problem (TSP). It is possible to rewrite problem (3.5) as a TSP problem. Take a complete graph $K_{\lceil d \rceil + 1}$ and label the vertices $0, 1, \ldots, \lceil d \rceil$. Define the weights of the edges as follows:

$$\begin{array}{rcll} w_{0,i} & = & 0 & \forall\ i = 1, \ldots, \lceil d \rceil \\ w_{i,j} & = & \left\lceil \sqrt{d^2 - (j-i)^2} \right\rceil & \forall\ i, j = 1, \ldots, \lceil d \rceil. \end{array}$$

  A shortest tour in this graph now corresponds to a permutation that minimizes problem (3.5).

We can thus determine a lower bound for the minimal distance along the y-axis, necessary to place $\lceil d \rceil$ consecutive non-overlapping circles with diameter d. Step 2 is now to use

$Y(d)$ or $\widetilde{Y}(d)$ to find an upper bound for the maximin distance of a two-dimensional LHD. When we base our upper bound on $Y(d)$, we define d_n as follows:

$$\begin{aligned} d_n = \quad & \min \quad d \\ & \text{s.t.} \quad Y(d) + \left\lfloor \tfrac{n}{\lceil d \rceil} \right\rfloor - 1 > n - 1 \\ & \qquad d^2 \in \mathbb{N}_+. \end{aligned} \tag{3.6}$$

Proposition 3.5. *Let d_n^{*2} be the largest sum of two squares that is strictly smaller than d_n^2. Then the value d_n^* is an upper bound for the separation distance of a two-dimensional LHD of n points.*

Proof. First determine for a given number of points n whether an LHD with ℓ^2-distance d can possibly exist. For given n and d, we do this as follows. The maximal number of mutually disjoint subsets of $\lceil d \rceil$ consecutive circles is given by $\left\lfloor \frac{n}{\lceil d \rceil} \right\rfloor$. For each subset, $Y(d)$ is a lower bound for the distance along the y-axis necessary to place the points in the subset. Due to the non-collapsingness criterion, the y-value of the point in a subset with the smallest y-value must be different for each subset. Therefore, we need at least $Y(d) + \left\lfloor \frac{n}{\lceil d \rceil} \right\rfloor - 1$ distance along the y-axis to construct an LHD of n points with ℓ^2-distance d. By solving problem (3.6), we find the minimal d for which this minimal required distance is larger than $n-1$, i.e., for which $Y(d)+\left\lfloor \frac{n}{\lceil d \rceil} \right\rfloor - 1 > n-1$ holds. We thus know that d_n is the minimal d for which our method shows that no LHD with maximin distance d_n exists. By taking d_n^{*2} equal to the largest sum of two squares that is strictly smaller than d_n^2, we have an upper bound for the maximin distance of a two-dimensional LHD. □

We can also determine an upper bound by replacing $Y(d)$ in problem (3.6) with $\widetilde{Y}(d)$ or any other lower bound on $Y(d)$. By doing so, we find an upper bound that is at most as good as the bound based on $Y(d)$.

As we expect that the separation distance of two-dimensional maximin LHDs is non-decreasing in n, we would like the upper bound to have this same property. The following lemma shows that the upper bound d_n^* indeed has this property.

Lemma 3.4. *The upper bound d_n^* is non-decreasing in n.*

Proof. To show that the bound is non-decreasing in n, we use the fact that d_n is the smallest d for which $Y(d) + \left\lfloor \frac{n}{\lceil d \rceil} \right\rfloor - 1 > n - 1$ holds. As $d_n^* < d_n$, we thus know that:

$$Y(d_n^*) + \left\lfloor \frac{n}{\lceil d_n^* \rceil} \right\rfloor - 1 \leq n - 1,$$

which implies that:

$$Y(d_n^*) + \left\lfloor \frac{n+1}{\lceil d_n^* \rceil} \right\rfloor - 1 \leq Y(d_n^*) + \left\lfloor \frac{n}{\lceil d_n^* \rceil} \right\rfloor \leq n.$$

This means that d_n^* does not satisfy the constraint of problem (3.5) for $n+1$. Therefore, $d_{n+1} > d_n^*$. Recall that d_{n+1}^{*2} is obtained by rounding down $d_{n+1}^2$to the largest sum of two squares that is strictly smaller than d_{n+1}^2. As d_n^{*2} is a sum of two squares that is strictly smaller than d_{n+1}^2, we can conclude that $d_{n+1}^* \geq d_n^*$. Hence, the upper bound d_n^* based on $Y(d)$ is non-decreasing in n. □

The same holds for the upper bound based on $\widetilde{Y}(d)$ or any other lower bound on $Y(d)$.

Numerical results

We use the MIP formulation in (4) to determine $Y(d)$ for $d^2 = 2, \ldots, 144$. To solve the MIP, we implemented it in AIMMS (Bisschop and Entriken 1993) and used the CPLEX 9.1 solver (CPLEX 2005). All calculations were done on a PC with a 2.40 GHz Pentium IV processor. Small values of d^2 required less than a second to solve, but the largest d^2 required two days.

The TSP formulation was used to determine $\widetilde{Y}(d)$ for $d^2 = 2, \ldots, 665$. The TSP problem is symmetric, which enabled us to use the algorithm described by Volgenant and Jonker (1982). Their exact algorithm is based on the 1-tree relaxation in a branch-and-bound algorithm. We used the implementation provided and described in Volgenant (1990). Most cases were solved in less than a minute, but a few of the larger cases required several hours to solve.

With the values obtained for $Y(d)$ and $\widetilde{Y}(d)$, we determined upper bounds for $n = 2, \ldots, 114$ and $n = 2, \ldots, 529$, respectively. All bounds can be found in the appendix. For $n = 2, \ldots, 70$, Van Dam et al. (2007) determined optimal maximin designs using branch-and-bound techniques. In Table 3.5 a comparison is made between the upper bounds and the d- and d^2-values of these optimal maximin LHDs. One might wonder if it is fair to use the Oler bound in this comparison as it is a bound for unrestricted designs. However, as it is the only comparable known upper bound, it is not possible to make a better comparison.

	Average % above optimal d	Average % above optimal d^2	Number of tight cases
Oler bound	22.24	50.26	0
Bound based on $Y(d)$	5.77	12.04	12
Bound based on $\widetilde{Y}(d)$	6.44	14.47	12

Table 3.5: Comparison between bounds and optimal maximin LHDs for $n = 2, \ldots, 70$, given in Van Dam et al. (2007).

The table shows that the new bounds are a considerable improvement when compared with the Oler bound for smaller values of n. By definition, the bound based on $Y(d)$ is

always at least as good as the bound based on $\widetilde{Y}(d)$. For $n = 2, \ldots, 70$, the bound based on $Y(d)$ is tighter for 13 values of n.

As we may not have optimal maximin LHDs for $n > 70$, we compare the upper bounds with the approximate maximin LHDs in Van Dam et al. (2007). Table 3.6 shows that the new bounds are still better than the Oler bound, but the differences are smaller.

	Average % above best known d	Average % above best known d^2
Oler bound	18.49	41.16
Bound based on $Y(d)$	5.89	12.26
Bound based on $\widetilde{Y}(d)$	6.51	13.58

Table 3.6: Comparison between bounds and approximate maximin LHDs for $n = 2, \ldots, 114$, given in Van Dam et al. (2007).

For $n = 115, \ldots, 529$, we can compare only the Oler bound and the bound based on $\widetilde{Y}(d)$. In Table 3.7 the comparison is made for different intervals of n. We see that the Oler bound becomes relatively better as n increases.

Size n	$[2, 100]$	$[101, 200]$	$[201, 300]$	$[301, 400]$	$[401, 500]$	$[501, 529]$
Oler bound	22.40	10.45	7.75	6.46	5.74	5.66
Bound based on $\widetilde{Y}(d)$	6.65	6.31	5.85	5.72	5.70	5.95

Table 3.7: Average percentage that the bounds are above the separation distance d of the approximate maximin LHDs given in Van Dam et al. (2007).

The bound based on $\widetilde{Y}(d)$ is at least as good as the Oler bound for $n = 2, \ldots, 410$. For $n = 411, \ldots, 415$, sometimes one bound is better and sometimes the other. For values of $n \geq 416$, the Oler bound is at least as good as the $\widetilde{Y}(d)$ based bound. This has two causes. Firstly, the LHD becomes more similar to an unrestricted design as n increases. Since the original Oler bound is intended for unrestricted designs, it is to be expected that the Oler bound becomes better as n increases. Secondly, the definition of $\widetilde{Y}(d)$ allows certain circles to overlap. When d becomes larger, this will occur more and more frequently. The bound based on $\widetilde{Y}(d)$ is thus weaker for large n. However, in practice LHDs are used for relatively small values of n (several dozens) which makes this drawback less relevant.

3.3 Upper bounds for the ℓ^∞-distance

3.3.1 Bounding by graph covering

For the ℓ^∞-distance we obtain a bound for the separation distance of an LHD as follows.

Proposition 3.6. *Let D be an LHD of n points and dimension k. Then the separation ℓ^∞-distance d satisfies*

$$k(n-d)(n-d+1) \geq n(n-1).$$

Proof. In each coordinate $(n-d)(n-d+1)/2$ pairs of points have distance at least d. Since each pair of points must be separated by a distance at least d in at least one of the coordinates, it follows that $k(n-d)(n-d+1)/2 \geq n(n-1)/2$. □

Similar to the bound for the ℓ^2-case in Section 3.2.1, this bound does not seem to be of the right order if k is fixed. For example, if $k = 2$, the inequality in the proposition is satisfied if $n \geq 4d$. However, the maximin distance satisfies $d = \lfloor \sqrt{n} \rfloor$; see Van Dam et al. (2007).

Rather than finding the maximin distance given n and k, it seems more convenient to find the smallest $k = k_{\min}(n, d)$ for which an LHD of size n and dimension k with separation distance d exists. The proof of Proposition 3.6 suggests to formulate the problem as a graph covering problem. Consider the complete graph on n vertices (representing the points of the design). Each edge of this graph (representing a pair of points) must be covered by one of k subgraphs of a particular form. For each coordinate this graph has as edges those pairs of points that are at distance at least d in this coordinate. These subgraphs are all isomorphic copies of the graph that can be described as follows: the vertices are the points $0, 1, \ldots, n-1$, and two points are adjacent if their absolute difference is at least d. Thus the problem can now be reformulated as to find the minimal number of copies of a graph $G(n, d)$ that cover all edges of the complete graph K_n. Such graph covering problems are not studied frequently. However, graph partitioning problems—where the complete graph must be partitioned into (the right number of) copies of a given graph—are studied; see Heinrich (1996). Of course, if such a partitioning exists, then it is a minimal covering.

The graphs $G(n, d)$ that are of interest to us depend only on the difference between n and d, so it makes sense to fix this difference. To start off easily, let $d = n - 1$ (which is extremal). The graph $G(n, d)$ now consists of a single edge (and some isolated vertices that we may discard), and it is clear that we can cover (partition) the edges of the complete graph K_n by $\binom{n}{2}$ copies. Thus the above bound is tight, and we have the following proposition.

Proposition 3.7. *For $d = n - 1$, the smallest $k = k_{\min}(n, d)$ for which an LHD of size n in k dimensions with separation ℓ^∞-distance d exists, satisfies $k_{\min}(n, n-1) = \binom{n}{2}$.*

For $d = n - 2$, we have the following.

Proposition 3.8. *For $d = n - 2$, the smallest $k = k_{\min}(n, d)$ for which an LHD of size n in k dimensions with separation ℓ^∞-distance d exists, satisfies $k_{\min}(n, n-2) = \lceil \frac{n(n-1)}{6} \rceil$.*

Proof. Let $d = n - 2$, then the graph $G(n, d)$ is a path P_4 of 4 vertices (again we may discard the isolated vertices). Bermond and Sotteau (1976) showed that if $n(n-1)$ is a multiple of 6, then K_n can be partitioned into copies of P_4. It is straightforward to extend their result to minimal coverings, i.e., K_n can be covered by $k_{\min}(n, n-2) = \lceil \frac{n(n-1)}{6} \rceil$ copies of P_4. Thus for LHDs of size n and separation distance $d = n-2$ we need precisely this many dimensions. □

For $d = n - 3$ we were able to show the following. We omit the proof, which is similar (but more technical) to the proof of the case $d = n - 2$.

Proposition 3.9. *For $d = n-3$, the smallest $k = k_{\min}(n, d)$ for which an LHD of size n in k dimensions with separation ℓ^∞-distance d exists, satisfies $k_{\min}(n, n-3) = \lceil \frac{n(n-1)}{12} \rceil$.*

For smaller d the situation becomes more complicated. For large n we have the following: if $e = n-d$ is fixed, then by a result of Wilson (1976) there is a function $N(e)$ such that a partition of K_n into copies of $G(n, n-e)$ exists if $n(n-1)$ is a multiple of $(n-d)(n-d+1)$ and $n > N(e)$. Thus in those cases $k_{\min}(n, d = n-e) = \frac{n(n-1)}{e(e+1)}$.

For $n \leq 10$ and all d it is possible to construct LHDs with dimension k meeting the lower bound $\lceil \frac{n(n-1)}{(n-d)(n-d+1)} \rceil$, except in the cases $n = 8, d = 3$ and $n = 10, d = 5$. For the first exception, $k = 2$ cannot be attained, since in two dimensions we have $n \geq d^2$; see Van Dam et al. (2007). For the second exception, $k = 3$ cannot be attained, as we prove by a complete search by computer (see also Section 3.5.1).

3.3.2 Attaining Baer's bound

Baer (1992) showed that the maximin ℓ^∞-distance d for unrestricted designs of n points on $[0, n-1]^k$ equals $\frac{n-1}{\lfloor (n-1)^{1/k} \rfloor}$. For $n = m^k + 1$, this maximin distance equals m^{k-1}. In this section, we give a construction of maximin LHDs of $n = m^k$ points with separation distance m^{k-1}, and show that n cannot be smaller to achieve this separation distance. First, we need the following lemma.

Lemma 3.5. *For the ℓ^∞-distance, the maximin distance d for LHDs of n points in k dimensions is a non-decreasing function of n.*

Proof. Consider an LHD D of size n and separation distance d. Let the point of D with first coordinate $n - d$ have remaining coordinates $x_2, x_3, \ldots, x_k$. Now construct D' from D by increasing by one all coordinates that are at least $x_2, x_3, \ldots, x_k$, respectively, and adding the point $(n, x_2, x_3, \ldots, x_k)$. Then D' is an LHD of size $n + 1$ with the same separation distance d as D, which proves the lemma. □

From this lemma and the above observation it follows that an LHD of $n = m^k$ points in k dimensions has separation distance at most m^{k-1}. We now give a construction of an LHD attaining that upper bound.

Construction 3.1. *Let $m \geq 2$ and $k \geq 1$ be integers, and let $n = m^k$. For $\mathbf{a} = (a_1, a_2, \ldots, a_k) \in \{0, 1, \ldots, m-1\}^k$ and $j = 1 \ldots, k$, let $\mathbf{x}(\mathbf{a}) = (x_1, x_2, \ldots, x_k)$, where*

$$x_j = \sum_{i=k-j}^{k-1} a_{i+1-k+j} m^i + m^{k-j} - 1 - \sum_{i=0}^{k-j-1} a_{k-i} m^i,$$

and let D be the design $D = \{\mathbf{x}(\mathbf{a}) \mid \mathbf{a} \in \{0, 1, \ldots, m-1\}^k\}$.

Examples of this construction are given in Tables 3.8 and 3.9.

a_1	0	1	0	1	0	1	0	1
a_2	0	0	1	1	0	0	1	1
a_3	0	0	0	0	1	1	1	1
x_1	3	7	1	5	2	6	0	4
x_2	1	3	5	7	0	2	4	6
x_3	0	1	2	3	4	5	6	7

Table 3.8: Construction 3.1 for $m = 2$ and $k = 3$.

a_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
a_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
a_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
a_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_1	7	15	3	11	5	13	1	9	6	14	2	10	4	12	0	8
x_2	3	7	11	15	1	5	9	13	2	6	10	14	0	4	8	12
x_3	1	3	5	7	9	11	13	15	0	2	4	6	8	10	12	14
x_4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Table 3.9: Construction 3.1 for $m = 2$ and $k = 4$.

Proposition 3.10. *The design D from Construction 3.1 is an LHD of $n = m^k$ points in k dimensions with maximin ℓ^∞-distance $d = m^{k-1}$.*

Proof. One can check that for each j, the map sending $\mathbf{a}$ to x_j is a one-to-one map from $\{0, 1, \ldots, m-1\}^k$ to $\{0, 1, \ldots, n-1\}$. Thus D is an LHD on n points. Next, observe the recursive structure of the construction. For fixed m, each point $\mathbf{x}'(a_1, a_2, \ldots, a_{k-1}) = (x'_1, x'_2, \ldots, x'_{k-1})$ of the LHD in dimension $k-1$, and each value a_k determines a point $\mathbf{x}(a_1, a_2, \ldots, a_k) = (x_1, x_2, \ldots, x_k)$ in the LHD in dimension k, where $x_j = m(x'_j + 1) - a_k - 1$, for $j = 1, \ldots, k-1$, and $x_k = \sum_{i=0}^{k-1} a_{i+1} m^i$. For an example, see the constructed designs in Tables 3.8 and 3.9. We use this recursion now to prove by induction on k that the separation distance is $d = m^{k-1}$. Of course this is trivial for $k = 1$, the basis for induction. Now suppose that the statement is true for $k-1$. Let $\mathbf{a}$ and $\mathbf{b} \in \{0, 1, \ldots, m-1\}^k$, and consider the corresponding design points $\mathbf{x}(\mathbf{a}) = (x_1, x_2, \ldots, x_k)$ and $\mathbf{y}(\mathbf{b}) = (y_1, y_2, \ldots, y_k)$, respectively. If x_k and y_k differ by at least d, then we are done, hence we may assume that they differ by at most $d-1$. Then it follows that a_k and b_k differ by at most one. Since the points $\mathbf{x}' = (\frac{1}{m}(x_j + a_k + 1) - 1)_{j=1,\ldots,k-1}$ and $\mathbf{y}' = (\frac{1}{m}(y_j + b_k + 1) - 1)_{j=1,\ldots,k-1}$ are in the LHD of dimension $k-1$ (as explained above),

which by assumption has separation distance m^{k-2}, it follows that if $a_k = b_k$, then $\mathbf{x}$ and $\mathbf{y}$ are at distance at least $m \cdot m^{k-2} = d$. Moreover, if a_k and b_k differ by one, say (without loss of generality) that $a_k = b_k + 1$, then the points $\mathbf{x}'$ and $\mathbf{y}'$ are at distance at least m^{k-2}. If this distance is at least $m^{k-2} + 1$, then $m(\mathbf{x}' + 1)$ and $m(\mathbf{y}' + 1)$ have distance at least $d + m$, hence $\mathbf{x}$ and $\mathbf{y}$ are at distance at least $d + m - 1 \geq d$. If the aforementioned distance between $\mathbf{x}'$ and $\mathbf{y}'$ is however exactly m^{k-2}, then $(a_1, a_2, \ldots, a_{k-1})$ and $(b_1, b_2, \ldots, b_{k-1})$ must differ (by one) in exactly one coordinate, say the t-th one. If $a_t = b_t - 1$, then $y_t - x_t = m^{k-1} - b_k + a_k = d + 1$; otherwise $a_t = b_t + 1$, and then $x_k = y_k + m^{k-1} + m^{t-1} > d$, and so in any case $\mathbf{x}$ and $\mathbf{y}$ have distance at least d. The statement now follows by induction. □

In fact, we can slightly generalize the above result.

Proposition 3.11. *Let $m \geq 2$, $k \geq 2$, and $t \leq m$ be nonnegative integers, and let $n = m^k + t$. Then the maximin distance d for LHDs of n points in k dimensions satisfies $d = m^{k-1}$.*

Proof. It follows from Lemma 3.5 and Proposition 3.10 that the maximin distance is at least as stated. From Baer's upper bound $\frac{n-1}{\lfloor (n-1)^{1/k} \rfloor}$ (rounded down) it follows that it is at most as stated. □

Note also that if D is an LHD with separation distance d, and we remove an arbitrary point $(x_1, x_2, \ldots, x_k)$ from D, and from the remaining points in D we decrease by one all coordinates that are larger than $x_1, x_2, \ldots, x_k$, respectively, then we obtain an LHD of size $n-1$ with separation distance at least $d-1$. Thus the maximin distance cannot increase by more than one as n increases by one. We now show that the above construction is extremal in the sense that we cannot decrease n and still achieve the same maximin distance.

Proposition 3.12. *Let $m \geq 2$ and $k \geq 2$ be integers, and let $n = m^k - 1$. Then the maximin ℓ^∞-distance d for LHDs of n points in k dimensions satisfies $d = m^{k-1} - 1$.*

Proof. By the above observation and Proposition 3.10 it suffices to prove that an LHD on n points cannot have separation distance $d = m^{k-1}$. Suppose on the contrary that we have such an LHD. Partition the set $I = \{0, 1, \ldots, n-1\}$ into m parts: $I_i = \{id, id + 1, \ldots, id + d - 1\}, i = 0, \ldots, m-2$ (each of cardinality d), and $I_{m-1} = \{(m-1)d, (m-1)d + 1, \ldots, (m-1)d + d - 2\}$ (of cardinality $d-1$). Accordingly, partition the set I^k into m^k parts $I_{i_1} \times I_{i_2} \times \cdots \times I_{i_k}$. In each of these parts the points are at mutual distance at most $d-1$, hence each part contains at most one design point. Suppose now that the part $I_{i_1} \times I_{i_2} \times \cdots \times I_{i_k}$ does not contain a design point. Since $I_{i_1} \times I \times \cdots \times I$ then contains $|I_{i_1}|$ points on one hand, and at most $m^{k-1} - 1 = d - 1$ on the other hand, this

implies that $i_1 = m-1$. Similarly it follows that $i_2 = \cdots = i_k = m-1$, hence all parts except $I_{m-1} \times I_{m-1} \times \cdots \times I_{m-1}$ contain precisely one point.

Now consider a slightly different partition of I, i.e., into parts $J_i = I_i$, for $i = 0, \ldots, m-3$, $J_{m-2} = I_{m-2} \backslash \{(m-1)d-1\}$, and $J_{m-1} = \{(m-1)d-1\} \cup I_{m-1}$. By considering the partition of I^k into parts $J_{i_1} \times I_{i_2} \times \cdots \times I_{i_k}$, it follows that $J_{m-1} \times I_{m-1} \times \cdots \times I_{m-1}$ contains precisely one design point. Similarly $I_{m-1} \times J_{m-1} \times \cdots \times I_{m-1}$ contains precisely one design point. Since $I_{m-1} \times I_{m-1} \times \cdots \times I_{m-1}$ does not contain a design point, these two points must be distinct. However, both are contained in $J_{m-1} \times J_{m-1} \times I_{m-1} \times \cdots \times I_{m-1}$, which contradicts the fact that also this part can contain at most one design point. □

3.3.3 Bounding by projection and partitioning in three dimensions

Consider a three-dimensional LHD of n points (x_i, y_i, z_i), $i = 1, \ldots, n$, with ℓ^∞-distance d. Now, project all design points for which $z_i \leq d-1$ onto the (x, y)-plane. Since the z-values of all these design points differ less than d, the differences of the x- or y-values should at least be d for all points, i.e., the projected points form a two-dimensional design with separation distance d. The same holds for any other "layer" within the three-dimensional LHD for which the z-values of the design points differ less than d. By taking the appropriate layer and further analyzing the projected design we obtain the following proposition.

Proposition 3.13. *For integers $n \geq 3$ and $d \geq 2$, let $N(n, d)$ be given by*

$$N(n,d) = \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \left(\left\lfloor \frac{n - \lfloor \frac{n}{d} \rfloor - i + 1}{d} \right\rfloor + 1 \right) + \min \left\{ n - d\lfloor \frac{n}{d} \rfloor, \left\lfloor \frac{n - 2\lfloor \frac{n}{d} \rfloor}{d} \right\rfloor + 1 \right\}. \quad (3.7)$$

The maximal d such that $d \leq N(n, d)$ is an upper bound for the ℓ^∞-maximin distance $d_{\max}$ for a three-dimensional LHD of n points.

Proof. Consider the mutually disjoint "layers" $I_i = \{id, id+1, \ldots, id+d-1\}, i = 0, \ldots, \lfloor \frac{n}{d} \rfloor - 1$, of z-values of the LHD. Among these $\lfloor \frac{n}{d} \rfloor$ layers there must be at least one for which the corresponding projected design (as described above) has all its x-values at most $n - \lfloor \frac{n}{d} \rfloor$ (since all x-values are distinct). The projection of this layer will be onto the $(n - \lfloor \frac{n}{d} \rfloor) \times (n-1)$-grid; see Figure 3.3.

In Figure 3.3, one can identify $\lfloor \frac{n}{d} \rfloor$ mutually disjoint strips of size $(n - \lfloor \frac{n}{d} \rfloor) \times (d-1)$. Furthermore, since the differences in y-values within each strip are less than d, the x-values have to differ at least d, and, hence, the first strip contains at most $\lfloor \frac{n - \lfloor \frac{n}{d} \rfloor}{d} \rfloor + 1$ points. Moreover, since all x-values are distinct, the second strip contains at most $\lfloor \frac{n - \lfloor \frac{n}{d} \rfloor - 1}{d} \rfloor + 1$ points, the third strip at most $\lfloor \frac{n - \lfloor \frac{n}{d} \rfloor - 2}{d} \rfloor + 1$ points, et cetera.

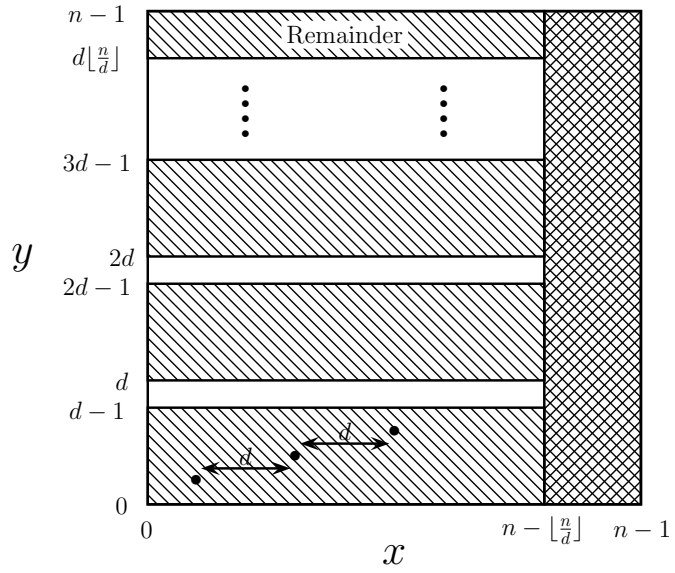


Figure 3.3: Strips within the projection onto the (x, y)-plane.

When n is not divisible by d, the remaining "partial" strip contains at most $\lfloor\frac{n-2\lfloor\frac{n}{d}\rfloor}{d}\rfloor+1$ points, but also at most $n-d\lfloor\frac{n}{d}\rfloor$ points. Note that in case n is divisible by d (and there is no remaining strip), the latter term is equal to 0 and the former term is non-negative (since $d \geq 2$). Thus, $N(n, d)$ is an upper bound for the number of points in the projected design, and the result follows. □

For values of n up to 165, the corresponding upper bounds are provided in Table 3.10. For many values of n the bound is better than Baer's bound. The bound also confirms Proposition 3.12 for $k = 3$ and $m \leq 5$.

Size $n \leq$	3	5	10	13	15	18	21	30	34	38	41	45	49	53	68
Bound	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Size $n \leq$	73	78	83	87	92	97	102	107	130	136	142	148	154	159	165
Bound	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table 3.10: Upper bound for ℓ^∞-maximin distance $d_{\max}$ for several n.

3.4 Upper bounds for the ℓ^1-distance

In this section, we apply the ideas of Section 3.2.1 concerning the ℓ^2-distance to the ℓ^1-distance. Bounding by the average gives the following bound.

Proposition 3.14. *Let D be an LHD of n points in k dimensions. Then the separation ℓ^1-distance d satisfies*

$$d \leq \left\lfloor \frac{(n+1)k}{3} \right\rfloor.$$

Proof. Let $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, with $\mathbf{x}_i = (x_{i1}, \ldots, x_{ik})$. The average distance among the points of D is:

$$\frac{1}{\binom{n}{2}} \sum_{i>j} \sum_{h} |x_{ih} - x_{jh}| = \frac{1}{\binom{n}{2}} \sum_{h} \sum_{i>j} |x_{ih} - x_{jh}| = \frac{1}{\binom{n}{2}} \sum_{h} \sum_{i'>j'} (i' - j') = k(n+1)/3,$$

and rounding finishes the proof. □

Similar remarks as in the ℓ^2-case apply here. More evidence that the bound is not of the right order to be tight if k is fixed is given by the case $k = 2$, where the maximin distance is known to be $\lfloor\sqrt{2n+2}\rfloor$; see Van Dam et al. (2007). The analogue of Lemma 3.1 is the following.

Lemma 3.6. *Let $d_{\max}(n,k)$ be the maximin ℓ^1-distance of an LHD of n points in k dimensions. Then $d_{\max}(n, k_1 + k_2) \geq d_{\max}(n, k_1) + d_{\max}(n, k_2)$.*

We can write the maximin distance d as the solution of the following integer programming problem:

$$\begin{array}{ll} \max & d \\ \text{s.t.} & \sum_{\pi \in S_n} k_\pi |\pi(i) - \pi(j)| \geq d, \quad \forall\ i > j \\ & \sum_{\pi \in S_n} k_\pi = k \\ & k_\pi \in \mathbb{N}_0, \quad \forall\ \pi \in S_n, \end{array} \tag{3.8}$$

where S_n is the set of permutations of $\{1, 2, \ldots, n\}$. As before, we may restrict the set S_n to its first half when ordered lexicographically, and we may assume that $k_{\pi^*} \geq 1$ for an arbitrary permutation π^*.

We again consider the cases $n = 3, 4$, and 5 to show the strength of the bound in Proposition 3.14.

Proposition 3.15. *For $n = 3$, the maximin ℓ^1-distance satisfies $d_{\max}(3,k) = \lfloor \frac{4k}{3} \rfloor$.*

Proof. The stated result follows from solving the above integer programming problem (3.8) by hand (the number of variables is 3). Alternatively, it also follows by using the upper bound and recursively applying Lemma 3.6 starting from $d_{\max}(3,1) = 1$, $d_{\max}(3,2) = 2$ (both trivial), and $d_{\max}(3,3) = 4$. The latter is attained by the design $\{(0,1,2), (1,2,0), (2,0,1)\}$. □

Proposition 3.16. *For $n = 4$, the maximin ℓ^1-distance satisfies $d_{\max}(4,k) = \lfloor \frac{5k}{3} \rfloor - 1$ if $k \equiv 3 \pmod 6$, and $d_{\max}(4,k) = \lfloor \frac{5k}{3} \rfloor$ otherwise.*

Proof. First, we show that the upper bound $\lfloor \frac{5k}{3} \rfloor$ cannot be attained if $k \equiv 3 \pmod 6$. Suppose that k is a multiple of 3, and that an LHD with separation distance $d = 5k/3$ exists. This implies that all points in the design are at equal distance. Fix one point, and let k_0 be the number of coordinates where this point is 0 or 3, and let $k_1 = k - k_0$ be the number of coordinates where it is 1 or 2. It follows that the average distance of this point to the other points equals $2k_0 + \frac{4}{3}k_1 = \frac{5}{3}k$. It now follows that $k_1 = k/2$, hence k should be even. Thus, for $k \equiv 3 \pmod 6$ the bound cannot be attained.

By solving the integer programming problem (3.8) for $k \leq 6$ by computer and using Lemma 3.6 (with $k_2 = 6$), we then find that the upper bound $\lfloor \frac{5k}{3} \rfloor$ is attained for all k except for $k \equiv 3 \pmod 6$, and that for these exceptions the maximin distance is one less. □

Proposition 3.17. *For $n = 5$, the maximin ℓ^1-distance satisfies $d_{\max}(5, k) = 2k - 1$ if $k \leq 4$ or $k = 7$, and $d_{\max}(5, k) = 2k$ otherwise.*

Proof. We first show that the bound $2k$ cannot be attained for $k \leq 4$ and $k = 7$. If the bound is attained, then all points of the design are at equal distance. Fix a point, let k_0 be the number of coordinates where this point is 0 or 4, let k_1 be the number of coordinates where it is 1 or 3, and let k_2 be the number of coordinates where it is 2. It follows that the average distance of this point to the other points equals $\frac{10}{4}k_0 + \frac{7}{4}k_1 + \frac{6}{4}k_2 = 2k$. Since $k_0 + k_1 + k_2 = k$, it follows that $k_1 + \frac{4}{3}k_2 = \frac{2}{3}k$. For $2 \leq k \leq 4$ and $k = 7$, there is a unique nonnegative integer solution (k_0, k_1, k_2) to these equations, and so each point has the same number k_2 of coordinates where this point is 2. This implies that the total number of coordinates where a 2 occurs equals $5k_2$ on one hand, and k on the other hand. This gives a contradiction in these cases.

By solving the integer programming problem (3.8) for $k \leq 9$ by computer, and using Lemma 3.6 (with $k_2 = 5$ or 6), we then find that the upper bound $2k$ is attained for all k except for $k \leq 4$ and $k = 7$, and that for these exceptions the maximin distance is one less than the given upper bound. □

Also for $n = 6$ and $n = 7$, we computed the integer programming problem (3.8) for $k \leq 20$; see Tables 3.11 and 3.12. Note that for some values of k only a lower bound was obtained. The tables show that the upper bound is again attained for many values of k. In particular it follows that for $n = 6$, the upper bound $\lfloor \frac{7k}{3} \rfloor$ is attained for all $k \equiv 0, 1, 2, 5 \pmod 6$, except for $k = 1, 2$ and possibly for $k = 7$. For $n = 7$, the upper bound $\lfloor \frac{8k}{3} \rfloor$ is attained for all $k \equiv 0, 1 \pmod 3$, except for $k = 1$ and 3. Similar to Section 3.2.1, we thus obtained a bound that is tight for n fixed and k increasing.

Dimension k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$d_{\max}(6,k)$	1	3	6	8	11	14	≥ 15	18	≥ 20	≥ 22	25	28	30	32	≥ 34	≥ 36	39	42	44	46
Upper bound	2	4	7	9	11	14	16	18	21	23	25	28	30	32	35	37	39	42	44	46

Table 3.11: Maximin ℓ^1-distance for LHDs of 6 points.

Dimension k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$d_{\max}(7,k)$	1	4	6	10	12	16	18	≥ 20	24	26	≥ 28	32	34	≥ 36	40	42	≥ 44	48	50	≥ 52
Upper bound	2	5	8	10	13	16	18	21	24	26	29	32	34	37	40	42	45	48	50	53

Table 3.12: Maximin ℓ^1-distance for LHDs of 7 points.

3.5 Final remarks and conclusions

3.5.1 Final remarks

By a branch-and-bound algorithm we were able to find maximin LHDs in three dimensions for small n and the three distance measures ℓ^2, ℓ^1, and ℓ^∞. The maximin distances are given in Table 3.13.

Size n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Squared maximin ℓ^2-distance	3	6	6	11	14	17	21	22	27	30	36	41	42	48		
Maximin ℓ^1-distance	3	4	4	5	6	6	7	8	8	8	9	10	10	11	11	
Maximin ℓ^∞-distance	1	2	2	2	3	3	4	4	4	4	5	5	5	6	6	6

Table 3.13: Maximin distances for LHDs in three dimensions.

The corresponding maximin designs and all other (approximate) maximin LHDs that appeared in this chapter can be downloaded from `http://www.spacefillingdesigns.nl`.

In two dimensions, the ℓ^∞-maximin distance is equal to $\lfloor n^{\frac{1}{2}} \rfloor$; see Van Dam et al. (2007). The results in three dimensions suggest that the corresponding ℓ^∞-maximin distance equals $\lfloor n^{\frac{2}{3}} \rfloor$. A natural extension would be that the ℓ^∞-maximin distance in k dimensions equals $d = \lfloor n^{\frac{k-1}{k}} \rfloor$. However, this is not the case in general, because for the case $n = 17$ and $k = 23$ the optimal distance is smaller than $\lfloor 17^{\frac{22}{23}} \rfloor = 15$ according to Proposition 3.6. The expression for d may, however, still provide an upper bound for the maximin distance.

Another interesting point is that we conjecture—but were unable to prove—that the analogue of Lemma 3.5 holds for the ℓ^2- and ℓ^1-distance measures, i.e., that also for these distance measures the maximin distance is non-decreasing in n.

3.5.2 Conclusions

We have obtained bounds for the separation distance of LHDs for several distance measures. These bounds are useful to assess the quality of approximate maximin LHDs by comparing their separation distances with the corresponding upper bounds. For the ℓ^2-

and ℓ^1-distances we obtain bounds by considering the average distance. These bounds are almost tight when the dimension k is relatively large. For the ℓ^2-distance in two dimensions we obtain a method that produces a bound that is better than Oler's bound if the number of points of the LHD is at most 400. For the ℓ^∞-distance we obtain a bound by looking at it as a graph covering problem. Besides this bound we construct maximin LHDs attaining Baer's bound for infinitely many values of n (the number of points) in all dimensions. Finally, we present a method for obtaining a bound for three-dimensional LHDs that is better than Baer's bound for many values of n.

3.A Bounds on two-dimensional ℓ^2-maximin LHDs

n	Oler	$d_n^{*2}(Y(d))$	$d_n^{*2}(\widetilde{Y}(d))$	d^2
2	5	2	2	2*
3	5	2	2	2*
4	8	5	5	5*
5	10	5	5	5*
6	10	5	5	5*
7	13	8	8	8*
8	13	8	8	8*
9	17	10	10	10*
10	18	13	13	10*
11	20	13	13	10*
12	20	13	13	13*
13	20	13	13	13*
14	25	17	17	17*
15	26	17	17	17*
16	26	18	18	17*
17	29	20	20	18*
18	29	20	20	18*
19	32	25	25	18*
20	32	25	25	18*
21	34	25	25	20*
22	34	26	26	25*
23	37	29	29	26*
24	37	29	29	26*
25	40	29	29	26*
26	41	29	29	26*
27	41	32	32	26*
28	41	34	34	29*
29	45	34	34	29*
30	45	34	34	29*
31	45	34	37	32*
32	45	37	40	32*
33	50	40	40	34*
34	52	41	41	37*
35	53	41	41	37*
36	53	41	41	37*
37	53	45	45	37*
38	53	45	45	41*
39	58	45	45	41*
40	58	45	50	41*
41	61	45	52	41*
42	61	50	52	41*
43	61	52	52	41*
44	65	52	52	50*
45	65	52	53	50*
46	68	53	53	50*
47	68	58	58	50*
48	68	58	58	50*
49	72	58	58	50*
50	73	61	61	52*
51	74	61	61	52*
52	74	61	65	58*
53	74	61	65	58*
54	74	61	65	58*
55	80	65	65	58*
56	80	65	68	58*
57	82	68	68	58*
58	82	68	73	61*
59	85	73	73	61*
60	85	73	73	65*
61	85	74	74	65*
62	89	74	74	65*
63	90	74	74	65*
64	90	74	74	65*
65	90	80	80	68*
66	90	80	80	68*
67	90	80	82	74*
68	97	80	85	74*
69	98	85	85	74*
70	98	85	85	74*
71	100	85	85	74
72	101	85	89	74
73	101	85	89	74
74	104	89	89	74
75	106	89	90	80
76	106	90	90	85
77	106	97	97	85
78	109	97	97	85
79	109	97	97	85
80	109	97	97	85
81	113	100	100	85
82	113	100	101	85
83	116	100	104	90
84	117	100	104	90
85	117	100	106	90
86	117	104	106	97
87	117	106	106	97
88	122	106	106	97
89	122	106	109	97
90	125	109	109	98
91	125	109	109	98
92	125	113	113	98
93	128	113	116	100
94	130	116	116	100
95	130	117	117	100
96	130	117	117	101
97	130	117	117	101
98	130	117	122	101
99	136	117	125	101
100	137	117	125	109
101	137	117	125	109
102	137	125	125	113
103	137	125	125	113
104	137	125	130	117
105	137	128	130	117
106	145	130	130	117
107	146	130	130	117
108	146	130	130	117
109	149	130	136	117
110	149	130	136	117
111	149	136	136	128
112	149	136	137	128
113	153	137	137	128
114	153	137	137	128
120	162	148	128	
130	173	160	145	
140	185	173	149	
150	200	185	170	
160	212	202	178	
170	225	208	185	
180	234	225	202	
190	245	234	208	
200	261	250	218	
210	274	261	241	
220	281	274	245	
230	298	290	250	
240	306	298	269	
250	320	314	277	
260	333	325	292	
270	346	338	305	
280	360	349	320	
290	370	365	320	
300	377	373	338	
310	394	388	346	
320	405	401	356	
330	416	410	370	
340	433	425	386	
350	445	442	401	
360	457	450	409	
370	468	464	410	
380	481	477	425	
390	493	490	442	
400	505	505	450	
410	514	514	461	
420	522	530	466	
430	541	544	485	
440	549	549	490	
450	565	565	509	
460	578	580	509	
470	586	592	533	
480	601	601	545	
490	613	617	549	
500	626	629	565	
510	637	641	578	
520	650	656	586	
529	661	661	586	

Table 3.14: Oler bound, bounds based on $Y(d)$ and $\widetilde{Y}(d)$, and d^2 of the best known LHD. When an optimal maximin LHD is known, the corresponding d^2 is marked with *.

Chapter 4

Nested maximin Latin hypercube designs

Mathematic puns are the first sine of madness.

(Johann Von Haupkoph)

4.1 Introduction

Latin hypercube designs are very useful in the approximation of black-box functions. By definition, black-box functions have no explicit description, but can be evaluated to obtain output values for specific input values. As evaluations of a black-box function often involve time-consuming computer simulations, we would like to construct an approximating model (or metamodel) based on evaluations in a (small) number of points; see, e.g., Montgomery (2009), Sacks et al. (1989a), (1989b), Myers (1999), Jones et al. (1998), Booker et al. (1999), Den Hertog and Stehouwer (2002), Santner et al. (2003), Queipo et al. (2005), Wang and Shan (2007), and Kleijnen (2008). A review of metamodeling applications in structural optimization can be found in Barthelemy and Haftka (1993), and in multidisciplinary design optimization in Sobieszczanski-Sobieski and Haftka (1997) and Simpson et al. (2008).

We use the term *design* to denote the set of evaluation points. As observed by many researchers, there is an important distinction between designs for computer experiments and designs for the more traditional response surface methods. Physical experiments exhibit random errors whereas computer experiments are often deterministic (see, e.g., Simpson et al. (2004), Forrester et al. (2006), and Forrester et al. (2008)). Therefore, designs for experiments often evaluate certain points multiple times. For designs for computer experiments, replication is redundant because the same input always results in the same output. This distinction is crucial, so one of the main aims in the field of design

of computer experiments (DoCE) is therefore to obtain efficient designs for computer experiments.

As is recognized by several authors, a design for computer experiments should at least satisfy the following two criteria (see Johnson et al. (1990) and Morris and Mitchell (1995)). First of all, the design should be *space-filling* in some sense. When no details on the functional behavior of the response parameters are available, it is important to be able to obtain information for the entire design space. Therefore, design points should be "evenly spread" over the entire region. Secondly, the design should be *non-collapsing*. When one of the design parameters has (almost) no influence on the black-box function value, two design points that differ only in this parameter will "collapse", i.e., they can be considered as the same point that is evaluated twice. As evaluation of the deterministic black-box function is often time-consuming, this is not a desirable situation. Therefore, two design points should not share any coordinate values when it is not known a priori which parameters are important. Moreover, we would like the projections of the points onto the axes to be separated as much as possible. When we consider a black-box function on a box-constrained domain, this can be accomplished by using Latin hypercube designs. A Latin hypercube design (LHD) of n points in k dimensions can be defined as an $n \times k$ matrix, were each column is a permutation of the set $\{0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\}$. The rows $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$, $i = 1, \ldots, n$, then define the n design points. Because the columns are permutations of the above set, for all of the k coordinates it holds that no two design points have the same value.

To obtain space-filling designs, the evaluation points are chosen in such a way that the separation distance (i.e., the minimal distance among any pair of points) is maximized, leading to so-called maximin designs. Other space-filling designs, like minimax, integrated mean squared error (IMSE), Audze-Eglais, discrepancy and maximum entropy designs, are also used in the literature. For a good survey of these designs see the book of Santner et al. (2003). Goel et al. (2008) argue that it would be better to use several criteria when selecting a design. However, Santner et al. (2003) show that maximin Latin hypercube designs—generally speaking—yield good approximations.

Maximin Latin hypercube designs were first constructed by Morris and Mitchell (1995) using simulated annealing. Ye et al. (2000) considered only the class of symmetric approximate maximin LHDs to reduce the computing effort. Jin et al. (2005) introduce the enhanced stochastic evolutionary (ESE) algorithm for finding various space-filling designs, including approximate maximin LHDs. Husslage et al. (2008) use the ESE algorithm to construct approximate maximin LHDs for up to 10 dimensions and up to 300 design points. Furthermore, they also construct approximate maximin LHDs by optimizing the maximin criterion over all LHDs having a certain periodic structure. This approach is an extension of the method used by Van Dam et al. (2007) to ob-

tain two-dimensional approximate maximin LHDs. In that paper, two-dimensional maximin LHDs are also found using a branch-and-bound algorithm. Finally, Grosso et al. (2009) use Iterated Local Search heuristics to find good approximate maximin LHDs for up to 10 dimensions. The best designs found in these papers are published on-line at `http://www.spacefillingdesigns.nl`. This website also contains the upper bounds on the separation distance for certain classes of maximin LHDs found by Van Dam et al. (2009b). These upper bounds can be used to asses the quality of approximate maximin LHDs.

In real-life, there are situations where we need a special type of designs called *nested designs*. This type of design consists of two separate designs, with the requirement that one design is a subset of the other design. Van Dam et al. (2009a) show how to construct one-dimensional nested maximin designs; the current chapter focuses on two- and higher-dimensional designs[1]. Four main reasons for nesting maximin designs are: *validation*, *models with different levels of accuracy*, *linking parameters*, and *sequential evaluations*.

To start with the first reason, consider the problem of fitting and validating a particular metamodel. In practice, the following approach is often used. First, a metamodel is fitted to the responses obtained when evaluating the design points in the training set. Then, a new set of design points—called the test set—is evaluated and the responses are compared with the response values predicted by the metamodel. If the differences between the predicted and the actual response values are small, the metamodel is considered to be valid; see also Cherkassky and Mulier (1998) for a more detailed description of the use of training and test sets. Because a metamodel should be a global approximation model, i.e., it should be valid for the entire feasible region, both the training set and the test set should cover the entire region. Moreover, the design points in the test set should not lie too close to the design points in the training set, i.e., the total set of design points should be space-filling. This can be accomplished by nesting two designs that are optimized with respect to, for example, the maximin criterion. The design points that are in both designs then form the training set and the points that are only in the large design make up the test set.

The second reason applies when an output variable of a process, product, or system is modeled by two black-box functions with different accuracy levels. These black-box functions could, for instance, be simulation models with different levels of detail. As a more accurate model is in general also more time-consuming, we can perform fewer evaluations of the high-accuracy model than of the low-accuracy model in the same amount of time. Instead of choosing to use either the high or low-accuracy model, we can choose to use both. We can then evaluate the high-accuracy model at all points in

[1]This chapter is a revision and extension of Husslage et al. (2005).

the small design and the low-accuracy model at all points in the large design. By using a nested design, high and low-accuracy evaluations are thus performed at all points in the small design. Multi-fidelity methods can combine the results from both models to obtain a metamodel that is better than a metamodel obtained by using only one of the two models and the same amount of time. More information on multi-fidelity methods can be found in Cressie (1993), Kennedy and O'Hagan (2000), Qian et al. (2006), and Forrester et al. (2007).

A third reason for using nested designs concerns linking parameters. Consider a product that consists of two components, each represented by a separate black-box function. To obtain an approximating model describing the behavior of the complete product, we need function evaluations of each black-box function. When one black-box function is more time-consuming to evaluate than the other, it could be better to perform different numbers of function evaluations of each black-box function. Moreover, in practice it may occur that these functions have input parameters in common; such parameters are called *linking parameters*, see Husslage et al. (2003). Evaluating the linking parameters at the same setting in both functions leads to an evaluation of the product. Not only do product evaluations provide a better understanding of the product, they are also very useful in the product optimization process. Another reason for using the same settings for (linking) parameters is due to physical restrictions on the black-box functions. Setting the parameters for computer experiments can be a time-consuming job in practice, because characteristics, such as shape and structure, have to be redefined for every new experiment. Therefore, it is preferable to use the same settings as much as possible. By constructing nested designs, we can determine the settings for linking parameters.

Nested designs are also useful when dealing with sequential evaluations. In practice it is common that after evaluating an initial set of points, extra evaluations are needed. As an example, suppose we construct an approximating model for some black-box function based on n_1 function evaluations. However, after validating the obtained model, it turns out that an extra set of function evaluations is needed to build a better model. We then face the problem of constructing a design on a total of n_2 points given the initial design on n_1 points with $n_2 > n_1$. To anticipate the possibility of extra evaluations, one can construct the two designs (on n_1 and n_2 points) at once, by constructing a nested design. An alternative method to deal with this situation would be sequential sampling. As this is beyond the scope of this chapter, we refer to Jones et al. (1998), Jin et al. (2002), and Kleijnen and Van Beers (2004) for more information on sequential sampling.

Above, we described why both Latin hypercube designs and nested designs are important. In this chapter, we construct nested maximin Latin hypercube designs in k dimensions with $k \geq 2$. Section 4.2 gives a more detailed formulation of this problem. When nesting two designs, it is not always possible to satisfy the LHD-structure for

both designs. Therefore, we introduce in Section 4.3 three different grid-structures that approximate the LHD-structure as good as possible. In Section 4.4, we present a branch-and-bound method for determining two-dimensional nested maximin designs and discuss two-dimensional Pareto optimal nested designs. For higher dimensions, determining the nested LHD that maximizes d becomes too time consuming. In Section 4.5, we therefore introduce a heuristic that also aims to maximize d but does not guarantee to find the optimal d. In Section 4.6, numerical results obtained with different variants of this heuristic are presented and compared. Furthermore, we discuss how to select a grid-structure and design based on these results. Finally, Section 4.7 contains conclusions and suggestions for further research.

4.2 Problem formulation

In this chapter, we focus on the problem of nesting two designs, X_1 and X_2, with $X_1 \subset X_2$, $X_j = \{x_i = (x_{i1}, x_{i2}, \ldots, x_{im}) \mid i \in I_j\}$, and $|I_j| = n_j$, $j = 1, 2$. Thus, the index set $I_1 \subset I_2 = \{1, 2, \ldots, n_2\}$ defines which design points x_i are part of both designs. The nested design is defined by the combination of X_1 and X_2. All design parameters are scaled such that they take values in the interval $[0, 1]$.

The first condition that we impose on the nested designs is that X_1 and X_2 must both be non-collapsing. This can be accomplished by using the LHD-structure. The main property of a regular LHD is that all points when projected onto one of the axes are equidistantly distributed. To form an LHD, the points in X_1 must thus be projected onto the set $\{0, \frac{1}{n_1-1}, \frac{2}{n_1-1}, \ldots, 1\}$ and the points in X_2 onto $\{0, \frac{1}{n_2-1}, \frac{2}{n_2-1}, \ldots, 1\}$. In order for X_1 and X_2 to both form a Latin hypercube design, the first set must be a subset of the second. However, this only holds when $n_2 - 1$ is a multiple of $n_1 - 1$ or, stated differently, when

$$c_2 := \frac{n_2 - 1}{n_1 - 1}$$

is integer. In all other cases, we have to compromise on the LHD-structure of one or both designs. As there are different ways of doing this, we propose three different grid-structures in Section 4.3: *nested n_1-grids*, *nested n_2-grids*, and *grids with nested maximin axes*. All of these grid-structures are constructed to compromise as little as possible on the LHD-structure. When c_2 is integer, the different grid-structures coincide and are such that both X_1 and X_2 are LHDs.

Secondly, we aim to determine the design points x_i and the set I_1 such that both designs are as much as possible space-filling given the chosen grid-structure. To optimize the space-fillingness, we choose to use the maximin distance criterion. As the distances between the points in X_1 will naturally be greater than the distances between the points in

X_2, scaling of these distances is necessary to enable a fair comparison with the maximin distance criterion. Therefore, we define d_j as the minimal scaled distance between all points in the design X_j:

$$d_j := \min_{\substack{l,m \in I_j \\ l \neq m}} \frac{d(x_l, x_m)}{s_j}, \; j = 1, 2, \tag{4.1}$$

where $d(\cdot, \cdot)$ is the Euclidean distance measure and s_1 and s_2 are scaling factors for the Euclidean distances in X_1 and X_2, respectively. Because one-dimensional designs of n points have distance $1/(n-1)$ and the minimum distance of n points in an k-dimensional hypercube is at most of the order $1/\sqrt[k]{n-1}$, it seems natural to use scaling factors

$$s_j := 1/\sqrt[k]{n_j - 1}, \; j = 1, 2$$

in (4.1) for k-dimensional designs. As we use the maximin distance criterion, we have to maximize the minimal scaled distance between any pair of points in X_1 and X_2. Therefore, what remains is to maximize the minimal distance

$$d = \min\{d_1, d_2\}$$

over all $I_1 \subset I_2$, with $|I_1| = n_1$, and $x_i \in [0, 1]^2$.

We are aware that the above formulation is just one way of combining the two separation distances into one objective and that other scaling factors or formulations are also possible. Using different scaling factors is no problem as all methods in this chapter can also be used for other values of s_1 and s_2. In Section 4.7, we discuss some other alternative objectives. Dealing with maximizing d_1 and d_2 as a bi-objective optimization problem is another possibility. For two-dimensional nested designs with small n_1 and n_2, we use this approach in Section 4.4.2. However, to limit the scope of this chapter, our main focus is on the above maximin objective.

By limiting the choice of design points to certain grids to obtain non-collapsingness, we generally obtain less space-filling designs. However, as a comparison of two-dimensional non-nested designs in Van Dam et al. (2007) shows, the loss in space-fillingness by imposing the LHD-structure is quite small. Furthermore, the non-collapsingness achieved by the LHD-structure is important when dealing with deterministic computer experiments. Especially when black-box function evaluations are expensive, using a separate screening design to determine the significant design parameters is often not an option. Consequently, we assume that the benefit of non-collapsingness justifies a limited loss in space-fillingness. By determining space-filling nested LHDs, we aim to limit this loss as much as possible.

4.3 Grid-structures for nested Latin hypercube designs

As mentioned in the previous section, X_1 and X_2 can only both form a Latin hypercube design if $c_2 := \frac{n_2-1}{n_1-1}$ is integer. When n_1 and n_2 do not satisfy this condition, we have to use a different structure that compromises on the LHD-structure of one or both designs. In this section, we introduce three different grid-structures that represent different compromises. A discussion on how to decide which grid-structure is most suitable for a particular situation is provided in Section 4.6.

To illustrate the different structures, examples are provided for the two-dimensional case of $n_1 = 6$ and $n_2 = 13$ points. In Figures 4.1 and 4.2 also the individual maximin Latin hypercube designs of $n_1 = 6$ and $n_2 = 13$ points are depicted to enable comparison with the non-nested case. Note that Figures 4.1 is not a subset of Figure 4.2. Furthermore, the circles illustrate the maximin distance because when we draw circles with the design points as their center, the maximin distance is equal to largest diameter such that the circles are non-overlapping. Moreover, it shows where the separation distance is attained.

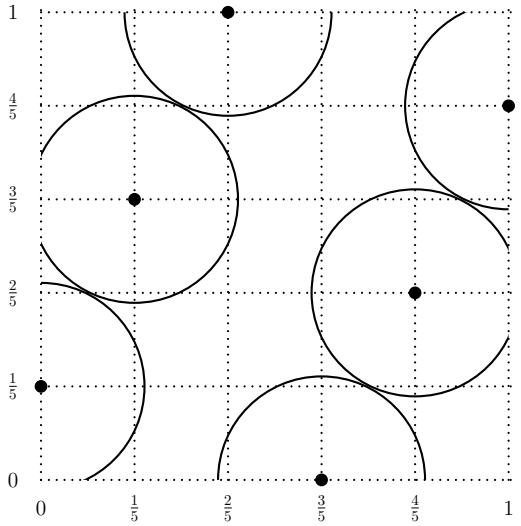


Figure 4.1: A maximin Latin hypercube design of 6 points; $d_1 = 1.0000$.

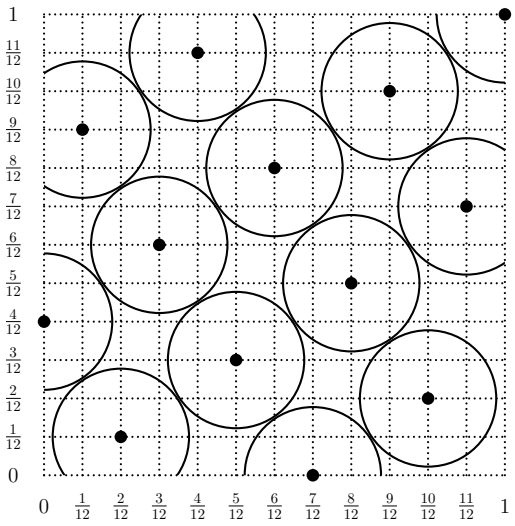


Figure 4.2: A maximin Latin hypercube design of 13 points; $d_2 = 1.0408$.

4.3.1 Nested n_2-grid

Before we explain the nested n_2-grid, let us first introduce the term *X_j-coordinates*. With X_j-coordinates we denote the levels obtained when projecting the design points of design X_j onto one of the axes (or dimensions), for $j = 1, 2$. For X_j to be an LHD, the X_j-coordinates must thus be equidistantly distributed for every dimension.

To construct a nested design where X_2 is an LHD, we have to choose all design points on the n_2-grid, with grid points $\{0, \frac{1}{n_2-1}, \frac{2}{n_2-1}, \ldots, 1\}^k$. Remember that we selected the LHD-structure because of the non-collapsingness with respect to the projections of the design points onto the axes. For the design X_2, the non-collapsingness is guaranteed by the equidistant distribution of the X_2-coordinates. To obtain a non-collapsing design $X_1 \subset X_2$, we also want to select the X_1-coordinates equidistantly distributed. If this is not possible, we try to obtain a space-filling distribution of the X_1-coordinates. Hence, what remains is to add restrictions that lead to the desired distribution of the X_1-coordinates.

To start, consider the case where $c_2 = \frac{n_2-1}{n_1-1} \in \mathbb{N}$. In this case, a non-collapsing design X_1 is obtained by limiting the choice of design points (of X_1) to the set of equidistantly distributed X_1-coordinates $\{0, \frac{1}{n_1-1}, \frac{2}{n_1-1}, \ldots, 1\}^k$. See, for example, the two-dimensional nested maximin Latin hypercube design of $n_1 = 16$ and $n_2 = 31$ points (with $c_2 = 2$) depicted in Figure 4.3. As all grid-structures coincide when c_2 is integer, this design is also a nested maximin design for the other two grid-structures. Therefore, we refer to it as a nested maximin LHD instead of a nested maximin n_2-LHD.

For the case $c_2 \notin \mathbb{N}$, the situation is more complicated. Because we are bound to the n_2-grid, and $n_1 - 1$ is no longer a divisor of $n_2 - 1$, it is not possible to have the X_1-coordinates equidistantly distributed. From the one-dimensional case, however, we know that for equidistantly distributed X_2-coordinates (as is the case with the n_2-grid) it is optimal to have either $\lfloor c_2 \rfloor - 1$ or $\lceil c_2 \rceil - 1$ X_2-coordinates between succeeding X_1-coordinates; see Van Dam et al. (2009a). Therefore, we require the X_1-coordinates to be separated by either $\lfloor c_2 \rfloor \frac{1}{n_2-1}$ or $\lceil c_2 \rceil \frac{1}{n_2-1}$.

Note that this restriction still leaves multiple grids possible for design X_1 when $c_2 \notin \mathbb{N}$. Figure 4.4 shows an example of a nested maximin design on a nested n_2-grid of $n_1 = 6$ and $n_2 = 13$ points, with $d = d_2 = 0.9129$ and $d_1 = 1.0035$. In this and following figures, the design points of X_1 are represented by solid dots, the open dots represent the extra design points needed to complete design X_2, hence, the solid and open dots together form the design points of X_2. The diameters of the dotted and solid circles are equal to the unscaled distance $d_1 * s_1$ and $d_2 * s_2$, respectively. They thus illustrated the separation distances of the designs X_1 and X_2.

For the nested n_2-grid, a suitable method to determine nested LHDs would seem to take an existing LHD of n_2-points for X_2 and select a subset of n_1 points for X_1. Forrester et al. (2007), for instance, use an exchange algorithm to implement this approach for multi-fidelity modeling. Although this method is quite attractive because of its simplicity, it does not generally yield a nested LHD satisfying all the restrictions of the nested n_2-grid. We illustrate this with the example in Figure 4.5. The figure shows a maximin Latin hypercube design for $n = 15$ obtained in Van Dam et al. (2007). Let us assume we want to construct a nested LHD with $n_1 = 8$ and $n_2 = 15$. Because in this case $c_2 = 2$,

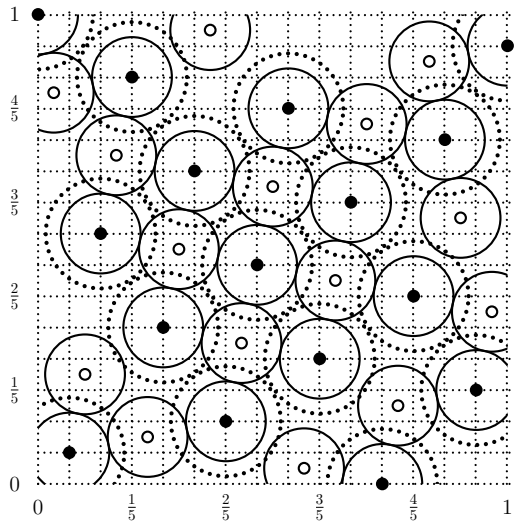


Figure 4.3: A nested maximin Latin hypercube design of $n_1 = 16$ and $n_2 = 31$ points; $d = d_1 = d_2 = 0.9309$.

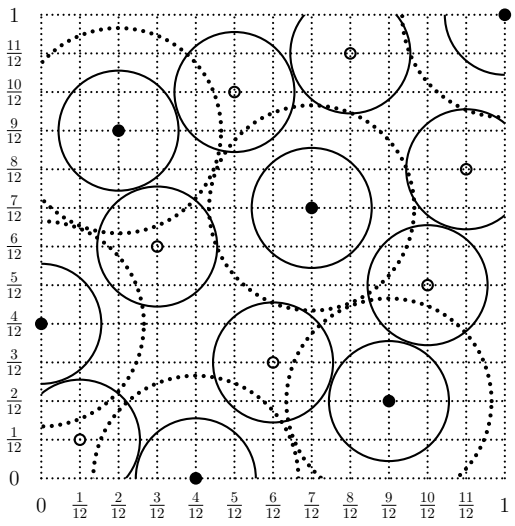


Figure 4.4: A nested maximin n_2-Latin hypercube design of $n_1 = 6$ and $n_2 = 13$ points; $d = d_2 = 0.9129$ and $d_1 = 1.0035$.

the nested n_2-grid is unique and both the X_1- and X_2-coordinates must be equidistantly distributed for both dimensions. The solid dots represent X_1 when we satisfy this latter restriction for the dimension on the horizontal axis. We can easily see that the distribution of the X_1-coordinates on the other axis is certainly not equidistant or space-filling. This problem also occurs for many other Latin hypercube designs and is even more likely to occur when the number of dimensions increases. Therefore, we do not use this method to construct nested LHDs, but use the methods described in Sections 4.4.1 and 4.5.1.

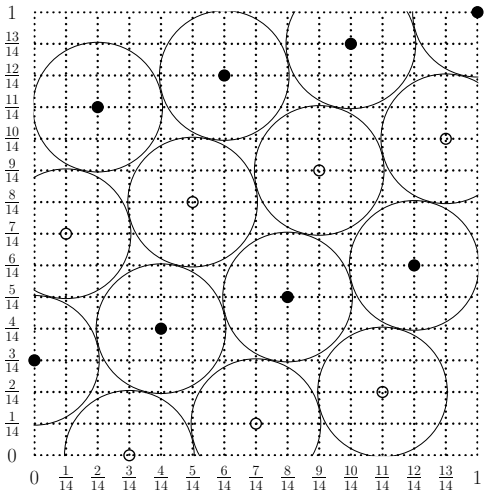


Figure 4.5: Example of the problem occurring when taking X_1 equal to a subset of an existing LHD.

4.3.2 Nested n_1-grid

When we want X_1 to be an LHD instead of X_2, we can use the nested n_1-grid. The design X_1 is then obtained by choosing n_1 design points on the n_1-grid, with grid points $\{0, \frac{1}{n_1-1}, \frac{2}{n_1-1}, \ldots, 1\}^k$. The additional X_2-coordinates are placed equidistantly between the X_1-coordinates. Similar to the nested n_2-grid, the (interiors of the) intervals formed by consecutive X_1-coordinates are again required to contain either $\lfloor c_2 \rfloor - 1$ or $\lceil c_2 \rceil - 1$ X_2-coordinates. Hence, consecutive X_2-coordinates are separated by either $\frac{1}{\lfloor c_2 \rfloor}\frac{1}{n_1-1}$ or $\frac{1}{\lceil c_2 \rceil}\frac{1}{n_1-1}$. Again, this leaves multiple grids possible when $c_2 \notin \mathbb{N}$. See Figure 4.6 for an example of a nested maximin design on a nested n_1-grid of $n_1 = 6$ and $n_2 = 13$ points, with $d = d_2 = 0.9522$ and $d_1 = 1.0000$.

4.3.3 Grid with nested maximin axes

The use of the Latin hypercube structure in the construction of a nested maximin design implies a preference of one design over the other. Design X_1 is assumed to be more important than design X_2 when a nested n_1-grid is used; design X_2 is preferred over design X_1 in case of a nested n_2-grid. If both sets are assumed to be of equal importance we would like to treat them equally. To deal with this problem, the X_1- and X_2-coordinates could be restricted to take only values at the levels of a (known) one-dimensional nested maximin design of n_1 and n_2 points; see Van Dam et al. (2009a). The design points of X_1 and X_2 could then be chosen from the grid points obtained in this way. Note that in this case the projections of the design points onto the axes are always optimally space-filling with respect to the maximin distance criterion. Furthermore, note that a one-dimensional maximin design, with $c_2 \notin \mathbb{N}$, is (again) not unique, so there are multiple grids possible. Figure 4.7 depicts an example of a nested maximin design of $n_1 = 6$ and $n_2 = 13$ points on a grid with nested maximin axes, with $d = d_1 = 0.9589$ and $d_2 = 0.9805$.

4.4 Two-dimensional nested designs

4.4.1 Branch-and-bound algorithm

To obtain two-dimensional nested maximin LHDs, we use an extension of the branch-and-bound algorithm of Van Dam et al. (2007). This extended branch-and-bound algorithm works as follows. Given n_1, n_2 and the grid-structure, we first determine all possible nested grids and calculate the possible distances that can occur for X_1 and X_2. These distances form a discrete set that can be efficiently searched and optimized. To determine whether a nested LHD exists with X_1 and X_2 having separation distances at least d_1 and d_2, respectively, a branch-and-bound search is performed for each possible nested grid. This branch-and-bound method is similar to the one used for the usual LHDs described

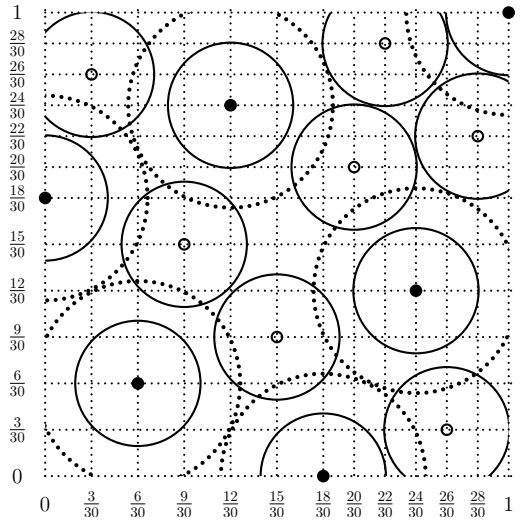


Figure 4.6: A nested maximin n_1-Latin hypercube design of $n_1 = 6$ and $n_2 = 13$ points; $d = d_2 = 0.9522$ and $d_1 = 1.0000$.

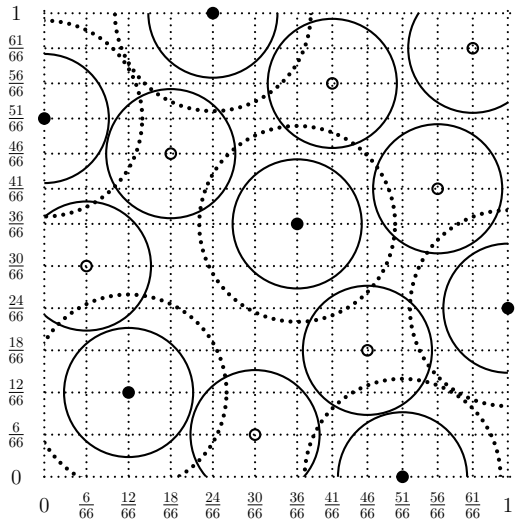


Figure 4.7: A nested maximin design of $n_1 = 6$ and $n_2 = 13$ points on a grid with nested maximin axes; $d = d_1 = 0.9589$ and $d_2 = 0.9805$.

in Van Dam et al. (2007); however, in general the nested grid-structures do not allow for the refinements given there. In the search tree, a node at level t corresponds to a partial nested design consisting of t design points $(x_1, \ldots, x_t)$, where the first n_1 points are in X_1 and the points are, furthermore, ordered by ascending first coordinate. Nodes in the tree are pruned when they correspond to partial nested designs that are collapsing or that have separation distances smaller than d_1 or d_2.

Using the extended branch-and-bound algorithm, we obtained results for n_2 up to 15 for all three grid-structures. For the cases where $c_2 \in \mathbb{N}$, the algorithm is refined so that nested maximin LHDs could be obtained for n_2 up to 32. The maximin distances of these designs can be found in Tables 4.6 and 4.7 in Appendix 4.A. The corresponding designs can be found on the website `http://www.spacefillingdesigns.nl`. In Section 4.6, the results are compared and discussed.

4.4.2 Pareto nested designs

Besides nested designs that maximize the objective function $d = \min\{d_1, d_2\}$, there are also some other interesting nested designs, namely *Pareto nested designs*. We call a combination of distances (d_1, d_2) Pareto optimal (or Pareto) if it is not possible to improve one of the distances, without deteriorating the other distance. A Pareto nested design is a nested design of which the distances (d_1, d_2) form a Pareto combination. For $c_2 \in \mathbb{N}$ and $n_2 \leq 32$, we have found all the Pareto combinations using a slightly adjusted version of the branch-and-bound algorithm. Furthermore, the original branch-and-bound algorithm

already ensures that the distances (d_1, d_2) of all nested maximin designs provided in Table 4.6 (in Appendix 4.A) are Pareto optimal.

n_1	n_2	Pareto combinations (d_1, d_2)
4	10	(0.8165, 0.9428), (1.2910, 0.7454)
4	16	(1.2910, 0.9309), (0.8165, 1.0646)
6	16	(1.0000, 0.7303), (0.6325, 1.0646)
9	17	(1.1180, 0.7906), (0.7906, 1.0607)
4	19	(1.2910, 0.9718), (0.8165, 1.0000)
7	19	(1.1547, 0.9718), (0.5774, 1.0000)
10	19	(1.0541, 0.7454), (0.7454, 1.0000)
11	21	(1.0000, 0.7071), (0.7071, 1.0000)
8	22	(1.0690, 0.8997), (0.5345, 0.9258)
12	23	(0.8528, 1.0871), (0.9535, 0.6742)
4	25	(1.2910, 1.0206), (0.8165, 1.0408)
5	25	(1.1180, 0.9129), (0.7071, 1.0408)
7	25	(1.1547, 0.8660), (0.5774, 1.0408)
9	25	(1.0000, 1.0408), (1.1180, 0.9129)
14	27	(1.0000, 1.0000), (1.1435, 0.8321)
4	28	(1.2910, 0.9623), (0.8165, 0.9813)
10	28	(0.9428, 0.9813), (1.0541, 0.8607)
5	29	(1.1180, 0.9636), (0.7071, 1.0177)
8	29	(1.0690, 0.9449), (0.8452, 0.9636)
15	29	(0.9636, 0.9636), (1.1019, 0.8018)
7	31	(1.1547, 0.9129), (0.5774, 0.9309)
16	31	(0.9309, 0.9309), (1.0646, 0.7746), (0.7303, 1.0328)

Table 4.1: All two-dimensional pairs (n_1, n_2) with more than one Pareto combination; $c_2 \in \mathbb{N}$, $n_2 \leq 32$.

Table 4.1 provides all Pareto combinations (d_1, d_2) corresponding to the pairs (n_1, n_2), with $c_2 \in \mathbb{N}$ and $n_2 \leq 32$, for which there exist more than one such combination. In this table, the first entry corresponds to the optimal maximin combination (d_1, d_2), followed by the other Pareto combination(s). Note that in case of $n_1 = 11$ and $n_2 = 21$ points there exist two different Pareto combinations, both with a maximin distance equal to $d = 0.7071$. For the (n_1, n_2) pairs $(9, 17)$ and $(10, 19)$, the objective values of the Pareto nested designs are also equal (0.7906 and 0.7454, respectively); however, the individual maximal distances of the second Pareto combination are smaller than the maximal distances of the (optimal) first combination ($1.0607 < 1.1180$ and $1.0000 < 1.0541$, respectively). The Pareto nested designs can also be found on the website `http://www.spacefillingdesigns.nl`.

4.5 Higher-dimensional nested designs

4.5.1 Enhanced stochastic evolutionary algorithm

For dimensions higher than two and for larger values of n_1 and n_2, the above branch-and-bound algorithm becomes too time-consuming for determing nested LHDs that maximize d. In these cases, we can use heuristics to find nested approximate maximin LHDs,

where "approximate" indicates that optimality is not guaranteed. One possible heuristic is the ESE algorithm of Jin et al. (2005). Using this algorithm, Husslage et al. (2008) obtain good results for approximate maximin LHDs. Furthermore, the algorithm is used in Viana et al. (2007) to generate space-filling LHDs. Although this algorithm was originally designed for non-nested designs, with some changes it is also applicable to nested designs. Before we look at these changes, we first give a short description of the original ESE algorithm; this description is based on Husslage et al. (2008).

The algorithm starts with an initial design and tries to find better designs by iteratively changing the current design. To determine if a new design can be accepted, a threshold-based acceptance criterion is used. This criterion is controlled in the outer loop of the algorithm. In the inner loop of the algorithm, new designs are explored.

The inner loop explores the design space as follows. At each iteration, first a dimension $m \in \{1, \ldots, k\}$ is selected. The algorithm then creates a fixed number of new designs by exchanging the m^{th} coordinate value of two randomly chosen points of the current design. The new design with the largest separation distance is then compared with the current design using a threshold criterion. The criterion ensures that better designs are always accepted and that worse designs can be accepted with a certain probability depending on the threshold value. If the new design is accepted, it replaces the current design. This proces is repeated until a certain stopping criterion is met.

The outer loop controls the threshold value. After the inner loop is completed, the outer loop determines how much improvement is made in the inner loop. If the improvement is above a certain level, the algorithm starts an improvement process in which it tries to rapidly find a local optimum. It does this by lowering the threshold value and thus accepting fewer deteriorations in the inner loop. If too little improvement is made, an exploration process is started that is intended to escape from a local optimum. The threshold value is first increased rapidly to move away from a local optimum and later slowly decreased to find better designs after moving away. The final design given by the algorithm is the best design found during all iterations of the inner loop.

To use the ESE algorithm for nested designs, the step that needs to be changed most is the generation of new designs. When one point is selected from X_1 and the other from $X_2 \backslash X_1$, exchanging the m^{th} coordinate value can distort the nested grid-structure. Figures 4.8 and 4.9 give an example where this distortion indeed occurs. The design in Figure 4.9 is obtained by exchanging one coordinate value of two points in the lower left part of Figure 4.8. As we require in each dimension that the first and last point should be in X_1, the new design is not a valid nested LHD. We could try to repair this by changing the assignment of the points to the sets X_1 and X_2. However, there exists no assignment such that the invalid nested design in Figure 4.9 becomes a valid nested LHD on a nested n_2-grid.

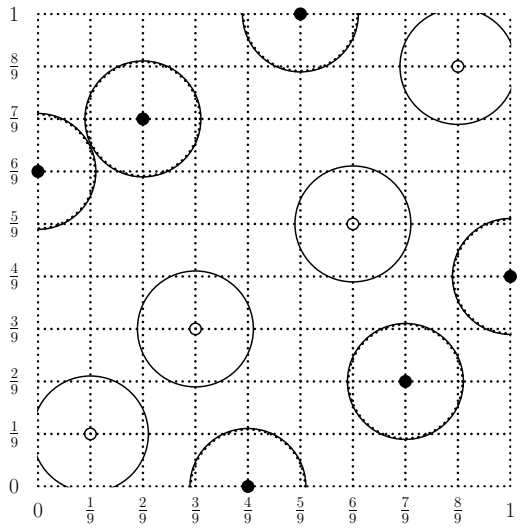


Figure 4.8: A nested Latin hypercube design of $n_1 = 6$ and $n_2 = 10$ points; $d = d_1 = 0.3086$ and $d_2 = 0.5556$.

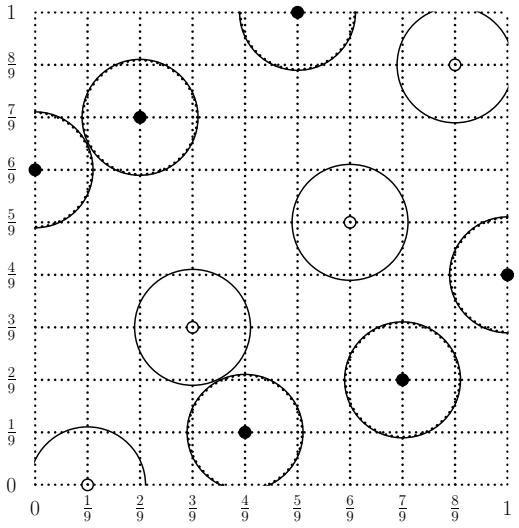


Figure 4.9: A nested design obtained by exchanging one coordinate value between a point in X_1 and one point in $X_2 \backslash X_1$.

Besides solving the above problem, a new method for generating designs can also take into account that we use a different objective function. As we consider the minimum of d_1 and d_2, we can, for instance, use different methods depending on which of the two values is smallest. In the next section, we discuss some different methods of generating new designs that take the above two aspects into account.

4.5.2 Generating new designs

The main problem that needs to be addressed by the new methods for generating new designs is the distortion of the chosen grid-structure. Fortunately, we can quite easily solve this problem in the following way. Instead of randomly choosing two points from the complete set of points, we choose two points from either X_1 or $X_2 \backslash X_1$. By exchanging coordinate values between two points within the same set, the chosen grid-structure is always maintained. Using this method, we do need to decide how to choose one of the two sets. Random selection is one option, but as we aim to maximize $\min\{d_1, d_2\}$, we could also base our choice on whether d_1 or d_2 is smallest. When d_1 is smallest, selecting two points from $X_2 \backslash X_1$ is not very useful as their positions do not directly influence the value of d_1. The value of d_2, on the other hand, does depend on the positions of all points and therefore both sets are relevant when d_2 is smallest.

We can also take into account that the grids are not unique when c_2 is non-integer. For instance, when $n_1 = 6$ and $n_2 = 13$, it can be verified that there are 21 different two-dimensional nested n_2-grids, after accounting for reflection and rotational symmetry.

In such cases, the choice of a specific grid can affect the maximal attainable value of d. Therefore, we consider different methods of selecting the grid of the initial design. Furthermore, we also look at methods that can change a grid without distorting it.

Based on the above observations, we develop the following four methods for generating new designs. The first method, which we call POINTRAND, starts with randomly selecting a point from X_2. Depending on whether this point is in X_1 or not, we select a second point from X_1 or $X_2 \backslash X_1$ respectively. This simple method is probably closest to the original ESE algorithm. However, it does not take into account the values of d_1 and d_2 and is not able to change the grid. To determine the effect of the first aspect, we develop a second method called POINTDMIN. When d_2 is smallest, the method works in the same way as POINTRAND. However when d_1 is smallest, we only choose points from X_1 because they are the only ones affecting d_1.

The third and fourth methods, GROUPRAND and GROUPDMIN, are able to change the grid. Remember that for all types of grids, there must be either $\lfloor c_2 \rfloor - 1$ or $\lceil c_2 \rceil - 1$ X_2-coordinates between every pair of consecutive X_1-coordinates. By deciding between which pairs we place $\lfloor c_2 \rfloor - 1$ points and between which pairs we place $\lceil c_2 \rceil - 1$ points, we fix a grid. To change a grid without it becoming invalid, we must thus change the assignment of $\lfloor c_2 \rfloor - 1$ and $\lceil c_2 \rceil - 1$ X_2-points to the pairs of consecutive X_1-coordinates. This principle leads to the following definitions of the two GROUP methods. The methods GROUPRAND and GROUPDMIN start with selecting a first point in the same way as in POINTRAND and POINTDMIN, respectively. After this first step, both methods continue in the same way. If a point in X_1 is selected, we simply exchange two points in X_1. Otherwise, we decide with equal probability to either exchange the selected point with another point in $X_2 \backslash X_1$ or to perform a group-exchange. A group-exchange is performed by first selecting two pairs of consecutive X_1-points, i.e., X_1-points that have consecutive X_1-coordinates in the m^{th} dimension. All X_2-points between a pair of consecutive X_1-points are now referred to as a group. Note that when $\lfloor c_2 \rfloor = 1$, a group can be empty. To generate a new design, we now switch the two groups. As both groups contain $\lfloor c_2 \rfloor - 1$ or $\lceil c_2 \rceil - 1$ points, this switch results in a valid nested design. When the number of points in the groups differ, the exchange of the groups also changes the grid. Depending on the type of grid, the group exchange not only affects the position of the points in the group but possibly other points too. Which and how other points are affected differs per type of grid, but is fairly straightforward to determine.

To illustrate the different methods, we again use the design in Figure 4.8. To simplify notation, we use the terms DMIN, RAND, POINT or GROUP to refer to any of the two methods whose name contain these words, e.g., POINTDMIN and GROUPDMIN are both DMIN methods. As $d_1 < d_2$, a DMIN method would exchange a coordinate value of two points in X_1. In the ESE algorithm, a fixed number of designs is generated

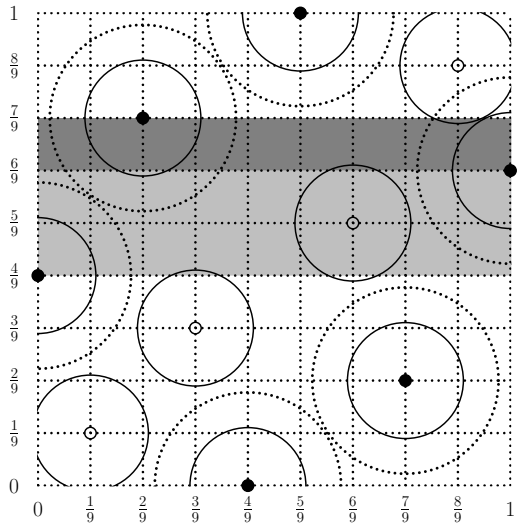


Figure 4.10: A nested Latin hypercube design of $n_1 = 6$ and $n_2 = 10$ points; $d = d_2 = 0.7454$ and $d_1 = 0.8958$.

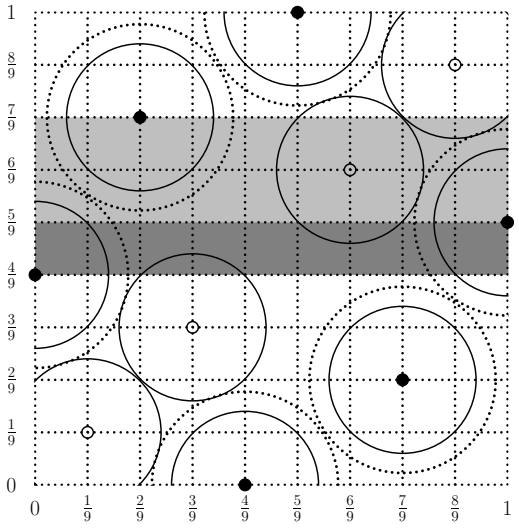


Figure 4.11: A nested maximin Latin hypercube design of $n_1 = 6$ and $n_2 = 10$ points; $d = d_1 = 0.8958$ and $d_2 = 0.9428$.

and the best is selected for comparison with the current design. We could, for instance, obtain the design in Figure 4.10 by the exchange of one coordinate value between the points with coordinates $(0, \frac{6}{9})$ and $(1, \frac{4}{9})$ in the current design of Figure 4.8. Looking at the d-values, we see that this design is an improvement and is thus selected by the ESE algorithm to become the new current design. Let us next consider a GROUP method and take m equal to 2, i.e., the dimension on the vertical axis. The two differently shaded areas in Figure 4.10 now form two possible groups. Notice that the top group is empty as there are no X_2-points between the X_1-points. In Figure 4.11, we see the result of exchanging the two groups. Because the groups are of different size, the grid has now changed. Again the design has improved and in this case the design is even optimal.

4.6 Numerical results

To compare the four different variants of the ESE algorithm, we generated three- and four-dimensional nested designs with $n_1 = 5, 10, \ldots, 50$ and $n_2 = n_1 + 5, n_1 + 10, \ldots, 60$ for each of the three grid-structures. We thus consider 65 different pairs (n_1, n_2) for each dimension and grid-structure. As the grid is not unique when $c_2 \notin \mathbb{N}$, the grid we select might affect the space-fillingness of the final design. For each combination of n_1, n_2, dimension, and grid-structure, we therefore ran the ESE algorithm ten times with a different grid and initial design. These computations have been performed on PCs with a 2.8 GHz Pentium D processor and the variants were implemented in Matlab R2007a. Per variant, grid and dimension, it took as much as 2 to 4 days to perform the ten runs

of the ESE algorithm for all 65 pairs. These computation times may seem quite large, but fortunately they are only one-time costs. When a good design is found and stored, it can be used many times in various applications without any additional computational cost. Therefore, we made all nested designs generated for this chapter available on the website `http://www.spacefillingdesigns.nl`.

For these ten runs, we tried two types of initial grids and designs: random and diagonal. For the first type, we randomly select a grid and design that satisfy the restrictions of the chosen grid-structure. The second type starts with a diagonal design, where each design point has the same value for all coordinates. However, the results did not indicate a significant effect of the chosen type on the space-fillingness of the final design. Also the calculation times of the ESE algorithm did not significantly differ. Therefore, we do not make a distinction between these two types in the remainder of this chapter.

	RAND		DMIN	
Three-dimensional	GROUP	POINT	GROUP	POINT
Nested n_1-grid	60	37	12	17
Nested n_2-grid	63	34	17	17
Grid with nested maximin axes	55	23	6	17
Four-dimensional				
Nested n_1-grid	66	26	2	11
Nested n_2-grid	57	29	6	15
Grid with nested maximin axes	49	31	9	11

Table 4.2: Percentage of the (n_1, n_2)-pairs for which a certain variant of the ESE algorithm finds a best design.

Using the best results of the ten runs of the ESE algorithm, we determine for each (n_1, n_2)-pair which method(s) obtained a best design. In Tables 4.8 and 4.9 of Appendix 4.A, the separation distances of the best designs are given for each of the three grid-structures. Table 4.2 contains the percentage of the 65 (n_1, n_2)-pairs for which a certain method performs best. Note that the sum of the percentages per row is larger than 100%, because—for some cases—a best design is found by multiple variants of the ESE algorithm. For the same reason, we cannot take the sum of two columns to determine the combined performance of two methods. When we study the results, we see that the two RAND-methods find the best design for most cases. One explanation for the relative poor performance of the DMIN-methods could be that the number of neighbor designs is smaller. By "neighbor" designs we mean all designs that can be obtained by making one possible change to the current design. When $d_1 < d_2$, a DMIN-method produces fewer neighbor designs than a RAND-method, because the DMIN-methods allow

fewer changes. This can make it more difficult for a DMIN-method to escape from a local minimum, which could result in a lower performance. Of the two RAND-methods, GROUPRAND performs the best for most cases. This indicates that the possibility of changing the grid indeed improves the performance of the ESE algorithm. Based on these results, we decided to use both RAND-methods to obtain nested approximate maximin designs for five up to ten dimensions. For dimension ten, calculating all 65 pairs took approximately 8 days per grid and variant.

	$l_1(n_1, n_2)$		$l_2(n_1, n_2)$	
Two-dimensional	Average	Range	Average	Range
Nested n_1-grid	4.13	[0.00,36.75]	11.12	[-5.41,52.56]
Nested n_2-grid	3.10	[-33.33,36.75]	8.92	[0.00,36.75]
Grid with nested maximin axes	3.83	[-14.29,36.75]	10.08	[-11.61,45.79]
Three-dimensional				
Nested n_1-grid	11.00	[0.00,17.60]	7.23	[-1.07,15.81]
Nested n_2-grid	10.95	[1.14,19.60]	7.90	[3.03,12.92]
Grid with nested maximin axes	11.38	[0.19,17.63]	8.02	[2.88,13.69]
Four-dimensional				
Nested n_1-grid	8.90	[0.00,17.54]	4.01	[-2.82,10.78]
Nested n_2-grid	9.56	[-7.04,16.82]	5.63	[2.11,11.21]
Grid with nested maximin axes	10.10	[-1.37,17.22]	5.82	[2.53,11.93]

Table 4.3: Average and range of percentage loss $l_j(n_1, n_2)$ caused by using nested (approximate) maximin designs instead of (approximate) maximin LHDs.

Furthermore, we are interested in the loss of space-fillingness caused by using nested instead of non-nested designs. We therefore compare the d_1- and d_2-values of the best nested design to the scaled separation distances of the (approximate) maximin LHDs of the same size. For a pair (n_1, n_2), we denote the first distances by $d_1(n_1, n_2)$ and $d_2(n_1, n_2)$ and the latter distances by $d(n_1)$ and $d(n_2)$. For $d(n_1)$ and $d(n_2)$, we use the best known (approximate) maximin LHDs available on `http://www.spacefillingdesigns.nl` (December 2008). We now define the percentage loss in separation distance as

$$l_j(n_1, n_2) := \left(d_j(n_1, n_2) - d(n_j)\right)/d(n_j),\ j = 1, 2.$$

Table 4.3 displays the averages and the ranges of these percentage losses over all evaluated (n_1, n_2)-pairs. Note that for two dimensions, we evaluated different pairs than for the other dimensions. When we consider the two-dimensional results, we see that the n_2-grid on average gives the best space-fillingness for both designs X_1 and X_2. For the

higher dimensions, the averages and ranges are closer, but the nested n_1-grid performs slightly better on both d_1 and d_2. These results are a bit surprising as we expected the nested n_2-grid to perform better on d_2 and the nested n_1-grid to perform better on d_1. Another observation, which might be surprising at first sight, is that some ranges also contain negative values. This means that for some (n_1, n_2)-pairs, the d_1- or d_2-distance is better than the distance of the corresponding (approximate) maximin LHD. Our main explanation is that the designs X_1 or X_2 do not always have to satisfy the LHD-structure. In some cases, this enables X_1 or X_2 to attain a larger separation distance than the (approximate) maximin LHD.

Although the above results give some indication of the performance of the grids in general, the results do not tell us which grid gives the highest d, d_1 and d_2 values for a specific pair (n_1, n_2). In Table 4.4, we present the percentages of pairs (n_1, n_2), with $c_2 \notin \mathbb{N}$, for which a grid type performs best on a particular distance. We do not consider the pairs with $c_2 \in \mathbb{N}$ because for these pairs all grids are equal.

	Percentage best designs		
Two-dimensional	d	d_1	d_2
Nested n_1-grid	17	36	16
Nested n_2-grid	67	56	72
Grid with nested maximin axes	16	11	13
Three-dimensional			
Nested n_1-grid	52	45	53
Nested n_2-grid	19	37	21
Grid with nested maximin axes	29	18	26
Four-dimensional			
Nested n_1-grid	69	71	66
Nested n_2-grid	24	18	26
Grid with nested maximin axes	8	11	10

Table 4.4: Percentage of the (n_1, n_2)-pairs, with $c_2 \notin \mathbb{N}$, for which a particular grid type performs best on d, d_1, or d_2.

Not surprisingly, the grids with the lowest average loss in Table 4.3 also have the highest percentage of pairs for which they perform best. However, there is still a considerable percentage of pairs where one of the other two grids perform better. It thus depends on the particular pair (n_1, n_2) which grid to choose based on the separation distances. Furthermore, in many practical situations, the values of n_1 and n_2 are not fixed which leaves some freedom to change these values. In those situations, we can thus also consider nested designs where n_1 and n_2 are slightly lower or higher. Let us, for example, consider the two-dimensional designs with $n_1 = 5$ and $n_2 = 10$. In Table 4.5, we compare the losses

of these designs to the losses of the designs with $n_1 = 6$ and $n_2 = 10$. The comparison shows that all losses either reduce or stay the same. Choosing n_1 equal to 6 instead of 5 thus seems to be a better choice in terms of space-fillingness.

Two-dimensional	$l_1(5,10)$	$l_2(5,10)$	$l_1(6,10)$	$l_2(6,10)$
Nested n_1-grid	36.75	16.15	0.00	10.00
Nested n_2-grid	28.34	10.56	10.42	10.56
Grid with nested maximin axes	31.20	12.28	2.02	8.17

Table 4.5: Example of reduction in percentage loss $l_j(n_1, n_2)$ by choosing different value for n_1.

The choice for a specific grid or (n_1, n_2)-pair not only depends on the space-fillingness. When it is not known a priori which design parameters are important, the non-collapsingness criterion should also be considered. The projections of the design point onto the axes should preferably be space-filling, which is accomplished by choosing a grid with nested maximin axes.

Furthermore, the reason why a nested design is used may also affect the choice for a particular grid. For example, a nested n_1-grid or a grid with the highest d_1 could be preferable for sequential evaluations, because it is known with certainty that the first set of design points is evaluated, whereas the evaluation of an extra set of design points may depend on the previously evaluated set. However, in the same setting, a nested n_2-grid is preferred when the final set of design points (i.e., X_2) is required to be a Latin hypercube design, as is often the case in practice. When dealing with linking design parameters, the choice for a specific grid mostly depends on the question which of the two designs—X_1 or X_2—is considered to be the most important one and should, thus, have an LHD-structure or have the largest separation distance. A grid with nested maximin axes should be used when there is no explicit preference for either one of the designs. When constructing a training set and a test set, design X_1 forms the training set and is the most important of the two designs. This is because the prediction accuracy of a metamodel is, among others, affected by the choice of the evaluation points in the training set. A space-filling distribution of these points over the feasible region is desirable and, hence, the grid for which the design points of X_1 have the largest separation distance is preferred. When combining high and low-accuracy models, it is hard to say which of the two designs is more important. The X_2 design is important because it is used to fit the initial model, but the X_1 design is also important as it is used to evaluate the accurate model whose results must improve the initial model.

From this discussion it follows that the notion of the "best" nested grid-design depends on the application at hand and the user's preferences. Fortunately, when $c_2 \in \mathbb{N}$, the comparison of the various nested grid-designs is superfluous. In this case, we do not have

to differentiate between different grid-structures, because they all yield the same nested maximin design (and maximin distance).

4.7 Conclusions and further research

4.7.1 Conclusions

A nested design consists of two separate designs, one being a subset of the other. Using these nested designs instead of traditional designs is useful when dealing with training and test sets, models with different levels of accuracy, linking parameters, or sequential evaluations, because nested designs can capture the dependencies between the two black-box functions or evaluation stages (with respect to the design parameters). This chapter focuses on constructing nested (approximate) maximin Latin hypercube designs. The maximin criterion is used to find space-filling nested designs, i.e., designs with the design points spread over the entire design space. By choosing the design points on a grid, we ensure non-collapsingness, i.e., no two design points will have the same coordinate values. We distinguish between three types of grids: a nested n_1-grid, a nested n_2-grid, and a grid with nested maximin axes. Which grid to use mainly depends on the application under consideration and the user's preferences. For two-dimensional designs, a branch-and-bound algorithm gives nested maximin designs for all grids and for values of n_2 up to 15. In the special case where $n_1 - 1$ is a divisor of $n_2 - 1$, we provide maximin distances for n_2 up to 32.

For dimensions higher than two, we introduced four variants of the ESE algorithm. Using a comparison of three- and four-dimensional designs, we determined that the POINTRAND- and GROUPRAND-methods obtained the best results. Therefore, these methods are also used to obtain designs for dimensions five up to ten and for up to 100 design points. Note that both variants of the ESE algorithms can also be used for higher dimensions and larger numbers of points. When the number of points or dimensions increases, changing the design and calculating the new value for d will become more time-consuming. We could solve or reduce this problem by performing fewer iterations of the inner loop of the ESE algorithm, although this might reduce the quality of the final nested design.

Besides comparing the different variants, we also studied the loss in space-fillingness by using nested designs instead of non-nested designs. The results show that the nested n_2-grid in general gives the smallest losses in two dimensions and the nested n_1-grid does so in higher dimensions. We also show that we can reduce this loss by choosing slightly different values for n_1 and n_2.

4.7.2 Further research

We remark that the objective $d = \min\{d_1, d_2\}$ used in this chapter is only one way of combining the separation distances of X_1 and X_2. As mentioned in the introduction, alternative scaling factors and formulations are possible. Taking the weighted sum of both objectives instead of the minimum would be a possible alternative objective. When using this objective, the branch-and-bound and ESE algorithms can still be used with little or no adjustments. Note however, that the DMIN-methods are not a very logical choice for this alternative objective as these methods explicitly consider the minimum of d_1 and d_2. Dealing with maximizing d_1 and d_2 as a bi-objective optimization problem is another possibility. In that case, different Pareto optimal nested designs could be found by using the weighted sum objective with various scaling factors.

Furthermore, we could also change the number of designs we want to nest. For instance in multi-fidelity modeling, we could come across models with more than two levels of accuracy. In these situations, we can use a nested design consisting of more than two designs. Note that for one-dimensional designs, Van Dam et al. (2009a) already considered optimizing the maximin criterion for these nested designs. To generate these nested designs for higher dimensions, we could extend the methods described in this chapter. The three main challenges would then be the following. Firstly, we must find grid-structures such that each design satisfies the LHD-structure as much as possible. This will become more difficult when we want to nest more designs. To solve this problem, we could initially consider only nested designs for which all designs can satisfy the LHD-structure. When nesting three designs $X_1 \subset X_2 \subset X_3$ with n_1, n_2 and n_3 design points, this would be possible if both $\frac{n_2-1}{n_1-1}$ and $\frac{n_3-1}{n_2-1}$ are integer. Secondly, we should decide on a criterion or method to achieve good space-fillingness for all the designs. Although we also need to make this decision for two sets, the decision will become more difficult when more sets have to be nested. Thirdly, when using the ESE algorithm, the methods for generating new designs should not distort the grid-structure. Depending on the grid-structure, we should determine whether the methods presented in this chapter are still applicable or if they should be adjusted. When a suitable grid and method is determined and a single objective is used for space-fillingness, the ESE-method in this chapter can be used to generate nested designs with more than two designs.

4.A Maximin and separation distances

Table 4.6 provides the maximin distances for nested maximin Latin hypercube designs in two dimensions with $c_2 \in \mathbb{N}$, and for $n_2 \leq 32$. For n_2 up to 15, and $c_2 \notin \mathbb{N}$, Table 4.7 provides the maximin distances for the two-dimensional nested maximin designs for all three grid-structures. Tables 4.8 and 4.9 provide the separation distances of three- and four-dimensional nested approximate maximin designs with $n_1 = 5, 10, \ldots, 50$ and $n_2 = n_1 + 5, n_1 + 10, \ldots, 60$ for all three grid-structures. Besides these distances, all tables also contain the scaled separation distances $d(n_1)$ and $d(n_2)$ of the approximate maximin LHDs available on `http://www.spacefillingdesigns.nl` (December 2008). The nested approximate maximin designs for dimensions five up to ten can also be found on this website.

n_1	n_2	$d(n_1)$	$d(n_2)$	d	d_1	d_2
2	3	1.4142	1.0000	1.0000	1.4142	1.0000
2	4	1.4142	1.2910	0.8165	1.4142	0.8165
2	5	1.4142	1.1180	0.7071	1.4142	0.7071
3	5	1.0000	1.1180	0.7071	1.0000	0.7071
2	6	1.4142	1.0000	1.0000	1.4142	1.0000
2	7	1.4142	1.1547	0.9129	1.4142	0.9129
3	7	1.0000	1.1547	0.9129	1.0000	0.9129
4	7	1.2910	1.1547	0.8165	0.8165	1.1547
2	8	1.4142	1.0690	0.8452	1.4142	0.8452
2	9	1.4142	1.1180	1.0000	1.4142	1.0000
3	9	1.0000	1.1180	1.0000	1.0000	1.0000
5	9	1.1180	1.1180	1.1180	1.1180	1.1180
2	10	1.4142	1.0541	0.9428	1.4142	0.9428
4	10	1.2910	1.0541	0.8165	0.8165	0.9428
2	11	1.4142	1.0000	1.0000	1.4142	1.0000
3	11	1.0000	1.0000	1.0000	1.0000	1.0000
6	11	1.0000	1.0000	1.0000	1.0000	1.0000
2	12	1.4142	1.0871	0.9535	1.4142	0.9535
2	13	1.4142	1.0408	0.9129	1.4142	0.9129
3	13	1.0000	1.0408	0.9129	1.0000	0.9129
4	13	1.2910	1.0408	0.9129	1.2910	0.9129
5	13	1.1180	1.0408	0.8165	1.1180	0.8165
7	13	1.1547	1.0408	0.9129	1.1547	0.9129
2	14	1.4142	1.1435	1.0000	1.4142	1.0000
2	15	1.4142	1.1019	0.9636	1.4142	0.9636
3	15	1.0000	1.1019	0.8452	1.0000	0.8452
8	15	1.0690	1.1019	0.8452	1.0690	0.8452
2	16	1.4142	1.0646	1.0646	1.4142	1.0646
4	16	1.2910	1.0646	0.9309	1.2910	0.9309
6	16	1.0000	1.0646	0.7303	1.0000	0.7303
2	17	1.4142	1.0607	1.0308	1.4142	1.0308
3	17	1.0000	1.0607	1.0000	1.0000	1.0308
5	17	1.1180	1.0607	0.9014	1.1180	0.9014
9	17	1.1180	1.0607	0.7906	1.1180	0.7906
2	18	1.4142	1.0290	1.0000	1.4142	1.0000
2	19	1.4142	1.0000	1.0000	1.4142	1.0000
3	19	1.0000	1.0000	1.0000	1.0000	1.0000
4	19	1.2910	1.0000	0.9718	1.2910	0.9718
7	19	1.1547	1.0000	0.9718	1.1547	0.9718
10	19	1.0541	1.0000	0.7454	1.0541	0.7454
2	20	1.4142	0.9733	0.9733	1.4142	0.9733
2	21	1.4142	1.0000	0.9487	1.4142	0.9487
3	21	1.0000	1.0000	0.9487	1.0000	0.9487
5	21	1.1180	1.0000	0.9487	1.1180	0.9487
6	21	1.0000	1.0000	0.9220	1.0000	0.9220
11	21	1.0000	1.0000	0.7071	1.0000	0.7071
2	22	1.4142	1.0911	0.9258	1.4142	0.9258
4	22	1.2910	1.0911	0.9258	1.2910	0.9258
8	22	1.0690	1.0911	0.8997	1.0690	0.8997
2	23	1.4142	1.0871	0.9535	1.4142	0.9535
3	23	1.0000	1.0871	0.9535	1.0000	0.9535
12	23	1.0871	1.0871	0.8528	0.8528	1.0871
2	24	1.4142	1.0632	1.0426	1.4142	1.0426
2	25	1.4142	1.0408	1.0408	1.4142	1.0408
3	25	1.0000	1.0408	1.0000	1.0000	1.0408
4	25	1.2910	1.0408	1.0206	1.2910	1.0206
5	25	1.1180	1.0408	0.9129	1.1180	0.9129
7	25	1.1547	1.0408	0.8660	1.1547	0.8660
9	25	1.1180	1.0408	1.0000	1.0000	1.0408
13	25	1.0408	1.0408	1.0408	1.0408	1.0408
2	26	1.4142	1.0198	1.0198	1.4142	1.0198
6	26	1.0000	1.0198	1.0000	1.0000	1.0000
2	27	1.4142	1.0000	1.0000	1.4142	1.0000
3	27	1.0000	1.0000	1.0000	1.0000	1.0000
14	27	1.1435	1.0000	1.0000	1.0000	1.0000
2	28	1.4142	1.0364	0.9813	1.4142	0.9813
4	28	1.2910	1.0364	0.9623	1.2910	0.9623
10	28	1.0541	1.0364	0.9428	0.9428	0.9813
2	29	1.4142	1.0177	0.9636	1.4142	0.9636
3	29	1.0000	1.0177	0.9636	1.0000	0.9636
5	29	1.1180	1.0177	0.9636	1.1180	0.9636
8	29	1.0690	1.0177	0.9449	1.0690	0.9449
15	29	1.1019	1.0177	0.9636	0.9636	0.9636
2	30	1.4142	1.0000	1.0000	1.4142	1.0000
2	31	1.4142	1.0328	0.9832	1.4142	0.9832
3	31	1.0000	1.0328	0.9832	1.0000	0.9832
4	31	1.2910	1.0328	0.9309	1.2910	0.9309
6	31	1.0000	1.0328	0.9309	1.0000	0.9309
7	31	1.1547	1.0328	0.9129	1.1547	0.9129
11	31	1.0000	1.0328	0.9309	1.0000	0.9309
16	31	1.0646	1.0328	0.9309	0.9309	0.9309
2	32	1.4142	1.0160	0.9672	1.4142	0.9672

Table 4.6: Maximin distances for two-dimensional nested maximin Latin hypercube designs; $c_2 \in \mathbb{N}$.

				Nested n_1-grid			Nested n_2-grid			Grid with nested axes		
n_1	n_2	$d(n_1)$	$d(n_2)$	d	d_1	d_2	d	d_1	d_2	d	d_1	d_2
3	4	1.0000	1.2910	0.6124	1.0000	0.6124	0.8165	1.3333	0.8165	0.6999	1.1429	0.6999
4	5	1.2910	1.1180	1.0541	1.2910	1.0541	1.1180	1.3693	1.1180	1.0880	1.3325	1.0880
3	6	1.0000	1.0000	0.9317	1.0000	0.9317	1.0000	1.2000	1.0000	0.9091	1.0909	0.9091
4	6	1.2910	1.0000	0.8165	0.8165	1.0541	0.9798	0.9798	1.0000	0.8645	0.8645	1.1161
5	6	1.1180	1.0000	0.8839	1.1180	0.8839	0.8944	0.8944	1.0000	0.9575	0.9722	0.9575
5	7	1.1180	1.1547	0.9682	1.1180	0.9682	0.9428	0.9428	1.1547	0.9897	1.0302	0.9897
6	7	1.0000	1.1547	1.0000	1.0000	1.0392	1.0541	1.0541	1.1547	1.1161	1.1161	1.1207
3	8	1.0000	1.0690	0.9354	1.0000	0.9354	1.0690	1.1429	1.0690	0.9978	1.0667	0.9978
4	8	1.2910	1.0690	0.7916	0.8165	0.7916	0.8452	1.3325	0.8452	0.7990	1.3152	0.7990
5	8	1.1180	1.0690	1.0458	1.1180	1.0458	0.8452	1.0302	0.8452	0.9990	1.0968	0.9990
6	8	1.0000	1.0690	0.8367	1.0000	0.8367	0.9035	0.9035	1.0690	0.9126	0.9383	0.9126
7	8	1.1547	1.0690	0.9129	0.9129	0.9860	0.9897	0.9897	1.0690	1.0319	1.0319	1.0349
4	9	1.2910	1.1180	0.8889	1.2910	0.8889	1.0000	1.2624	1.0000	0.9231	1.2814	0.9231
6	9	1.0000	1.1180	0.8944	1.0000	0.8944	1.0000	1.0078	1.0000	0.9575	0.9575	0.9722
7	9	1.1547	1.1180	0.9129	0.9129	1.0000	0.9682	0.9682	1.1180	0.9422	0.9422	1.0000
8	9	1.0690	1.1180	0.8571	1.0690	0.8571	1.0458	1.0458	1.1180	0.9990	0.9990	1.0679
3	10	1.0000	1.0541	0.8485	1.0000	0.8485	0.9428	1.1111	0.9428	0.8932	1.0526	0.8932
5	10	1.1180	1.0541	0.7071	0.7071	0.8839	0.8012	0.8012	0.9428	0.7692	0.7692	0.9247
6	10	1.0000	1.0541	0.9487	1.0000	0.9487	0.8958	0.8958	0.9428	0.9680	0.9798	0.9680
7	10	1.1547	1.0541	0.7906	1.1547	0.7906	0.9428	0.9813	0.9428	0.8883	0.8883	0.9035
8	10	1.0690	1.0541	0.8452	0.8452	0.9583	0.9296	0.9296	1.0541	0.9097	0.9226	0.9097
9	10	1.1180	1.0541	0.9561	1.0000	0.9561	0.9938	0.9938	1.0541	0.9513	0.9513	1.0090
4	11	1.2910	1.0000	0.8784	1.2910	0.8784	0.8944	1.1619	0.8944	0.8863	1.2103	0.8863
5	11	1.1180	1.0000	0.7454	1.1180	0.7454	0.8944	1.0770	0.8944	0.8131	1.0985	0.8131
7	11	1.1547	1.0000	0.8333	1.1547	0.8333	0.8944	1.0392	0.8944	0.8824	0.9666	0.8824
8	11	1.0690	1.0000	0.8452	0.8452	1.0102	0.9539	0.9539	1.0000	0.8965	0.8965	1.0715
9	11	1.1180	1.0000	1.0000	1.0000	1.0078	0.8944	1.0198	0.8944	0.9722	0.9722	1.0009
10	11	1.0541	1.0000	0.9428	0.9428	0.9938	0.9487	0.9487	1.0000	0.9680	0.9680	0.9798
3	12	1.0000	1.0871	0.8740	1.0000	0.8740	0.9535	1.0041	0.9535	0.9120	1.0009	0.9120
4	12	1.2910	1.0871	0.9965	1.2910	0.9965	1.0871	1.2695	1.0871	1.0250	1.2838	1.0250
5	12	1.1180	1.0871	0.7817	1.1180	0.7817	0.8528	1.0602	0.8528	0.7984	1.1039	0.7984
6	12	1.0000	1.0871	0.9446	1.0000	0.9446	0.9535	1.0947	0.9535	0.9511	1.0699	0.9511
7	12	1.1547	1.0871	0.8740	1.1547	0.8740	0.9535	0.9959	0.9535	0.8863	1.1222	0.8863
8	12	1.0690	1.0871	0.8452	0.8452	1.0051	0.9535	1.0205	0.9535	0.8518	0.8518	1.0307
9	12	1.1180	1.0871	1.0000	1.0000	1.0570	0.9271	0.9271	1.0871	0.9552	0.9552	0.9998
10	12	1.0541	1.0871	0.9428	0.9428	1.0423	0.9833	0.9833	1.0871	0.9664	0.9664	0.9862
11	12	1.0000	1.0871	1.0000	1.0000	1.0488	1.0365	1.0365	1.0871	1.0102	1.0102	1.0595
6	13	1.0000	1.0408	0.9522	1.0000	0.9522	0.9129	1.0035	0.9129	0.9589	0.9589	0.9805
8	13	1.0690	1.0408	0.8452	0.8452	1.0498	0.9129	0.9860	0.9129	0.8158	1.0380	0.8158
9	13	1.1180	1.0408	0.9186	1.0000	0.9186	0.9129	1.0000	0.9129	0.8748	1.0000	0.8748
10	13	1.0541	1.0408	0.9428	0.9428	0.9813	0.9129	1.0607	0.9129	0.9404	0.9583	0.9404
11	13	1.0000	1.0408	0.8944	0.8944	0.9798	0.9501	0.9501	1.0408	0.9301	0.9301	0.9476
12	13	1.0871	1.0408	0.9535	0.9535	0.9959	0.9965	0.9965	1.0408	0.9720	0.9720	1.0153
3	14	1.0000	1.1435	0.9286	1.0000	0.9286	0.8771	1.0769	0.8771	0.9082	0.9630	0.9082
4	14	1.2910	1.1435	0.9329	1.2910	0.9329	0.8771	1.3122	0.8771	0.8971	1.2280	0.8971
5	14	1.1180	1.1435	0.8498	1.1180	0.8498	0.7845	1.1717	0.7845	0.8035	1.1557	0.8035
6	14	1.0000	1.1435	0.8750	1.0000	0.8750	0.8771	1.0879	0.8771	0.8755	0.9722	0.8755
7	14	1.1547	1.1435	0.9234	1.1547	0.9234	0.8771	1.0659	0.8771	0.8863	1.0851	0.8863
8	14	1.0690	1.1435	0.8144	1.0690	0.8144	0.8771	1.0377	0.8771	0.8228	1.0801	0.8228
9	14	1.1180	1.1435	0.7906	0.7906	1.0078	0.8971	0.8971	1.1435	0.8210	0.8210	1.0284
10	14	1.0541	1.1435	0.9428	0.9428	1.0214	0.9515	0.9515	1.1435	0.9600	0.9600	1.0790
11	14	1.0000	1.1435	0.9192	1.0000	0.9192	1.0030	1.0030	1.1435	0.9774	0.9774	1.1144
12	14	1.0871	1.1435	0.9535	0.9535	1.0365	1.0519	1.0519	1.1435	1.0244	1.0244	1.0592
13	14	1.0408	1.1435	1.0408	1.0408	1.0623	1.0987	1.0987	1.1435	1.0716	1.0716	1.1154
4	15	1.2910	1.1019	0.8994	1.2910	0.8994	0.8748	0.8748	1.1019	0.9010	1.2852	0.9010
5	15	1.1180	1.1019	0.8432	1.1180	0.8432	0.9636	1.1518	0.9636	0.8994	1.1333	0.8994
6	15	1.0000	1.1019	0.9080	1.0000	0.9080	0.8452	0.9313	0.8452	0.8960	0.9731	0.8960
7	15	1.1547	1.1019	0.9582	1.1547	0.9582	1.1019	1.2372	1.1019	1.0456	1.2049	1.0456
9	15	1.1180	1.1019	0.7906	0.7906	0.9922	0.8452	1.0880	0.8452	0.7967	0.7967	1.0242
10	15	1.0541	1.1019	0.9428	0.9428	1.0599	0.9091	0.9091	0.9636	0.9299	0.9299	1.0758
11	15	1.0000	1.1019	0.9539	1.0000	0.9539	0.9583	0.9583	0.9636	0.9272	0.9272	1.0295
12	15	1.0871	1.1019	0.8672	1.0871	0.8672	0.9768	0.9768	1.1019	0.9675	0.9675	1.0147
13	15	1.0408	1.1019	0.9129	0.9129	0.9860	1.0202	1.0202	1.1019	0.9923	0.9923	1.0718
14	15	1.1435	1.1019	1.0000	1.0000	1.0176	1.0619	1.0619	1.1019	1.0368	1.0368	1.0759

Table 4.7: Maximin distances for two-dimensional nested designs; $c_2 \notin \mathbb{N}$.

				Nested n_1-grid			Nested n_2-grid			Grid with nested axes		
n_1	n_2	$d(n_1)$	$d(n_2)$	d	d_1	d_2	d	d_1	d_2	d	d_1	d_2
5	10	1.3162	1.2009	1.1400	1.1906	1.1400	1.0583	1.0583	1.0591	1.0990	1.0990	1.1664
5	15	1.3162	1.1927	1.0827	1.3162	1.0827	1.1023	1.2524	1.1023	1.0661	1.2929	1.0661
5	20	1.3162	1.1410	1.0506	1.1906	1.0506	1.0510	1.1991	1.0510	1.0888	1.3087	1.0888
5	25	1.3162	1.1465	1.0546	1.1906	1.0546	1.0546	1.1906	1.0546	1.0546	1.1906	1.0546
5	30	1.3162	1.1061	1.0582	1.1906	1.0582	1.0647	1.2708	1.0647	1.0672	1.1876	1.0672
5	35	1.3162	1.1030	1.0605	1.1906	1.0605	1.0695	1.2148	1.0695	1.0530	1.1868	1.0530
5	40	1.3162	1.1033	1.0623	1.1906	1.0623	1.0507	1.1811	1.0507	1.0587	1.1847	1.0587
5	45	1.3162	1.0943	1.0553	1.1906	1.0553	1.0553	1.1906	1.0553	1.0553	1.1906	1.0553
5	50	1.3162	1.0899	1.0517	1.1906	1.0517	1.0508	1.2095	1.0508	1.0537	1.1726	1.0537
5	55	1.3162	1.0911	1.0510	1.1906	1.0510	1.0476	1.1857	1.0476	1.0471	1.1930	1.0471
5	60	1.3162	1.0902	1.0431	1.1906	1.0431	1.0454	1.2020	1.0454	1.0492	1.3137	1.0492
10	15	1.2009	1.1927	1.0285	1.0841	1.0285	1.0612	1.1119	1.0612	1.0396	1.0396	1.0559
10	20	1.2009	1.1410	0.9895	1.0074	0.9895	1.0030	1.0034	1.0030	1.0114	1.0118	1.0114
10	25	1.2009	1.1465	1.0036	1.0074	1.0036	1.0056	1.0218	1.0056	0.9895	0.9932	0.9895
10	30	1.2009	1.1061	1.0074	1.0074	1.0118	1.0051	1.0566	1.0051	1.0052	1.0052	1.0092
10	35	1.2009	1.1030	1.0221	1.0591	1.0221	1.0438	1.0684	1.0438	1.0182	1.0200	1.0182
10	40	1.2009	1.1033	1.0074	1.0074	1.0231	1.0289	1.0411	1.0289	1.0272	1.0531	1.0272
10	45	1.2009	1.0943	1.0108	1.0591	1.0108	1.0216	1.0216	1.0275	1.0224	1.0550	1.0224
10	50	1.2009	1.0899	1.0201	1.0591	1.0201	1.0402	1.0748	1.0402	1.0183	1.0183	1.0230
10	55	1.2009	1.0911	1.0119	1.0591	1.0119	1.0119	1.0591	1.0119	1.0119	1.0591	1.0119
10	60	1.2009	1.0902	1.0129	1.0591	1.0129	1.0265	1.0303	1.0265	1.0345	1.0512	1.0345
15	20	1.1927	1.1410	1.0612	1.0612	1.0908	1.0320	1.0383	1.0320	1.0431	1.0432	1.0431
15	25	1.1927	1.1465	0.9889	0.9889	1.0250	0.9984	1.0092	0.9984	1.0146	1.0146	1.0255
15	30	1.1927	1.1061	0.9889	0.9889	0.9996	0.9834	0.9834	0.9938	0.9824	0.9824	0.9994
15	35	1.1927	1.1030	0.9889	0.9889	0.9945	0.9975	0.9975	0.9993	0.9822	0.9843	0.9822
15	40	1.1927	1.1033	0.9889	0.9889	0.9889	0.9990	1.0117	0.9990	0.9942	0.9957	0.9942
15	45	1.1927	1.0943	0.9889	0.9889	0.9981	0.9957	1.0130	0.9957	0.9914	0.9914	0.9970
15	50	1.1927	1.0899	1.0038	1.0038	1.0041	0.9991	1.0140	0.9991	0.9990	1.0390	0.9990
15	55	1.1927	1.0911	1.0034	1.0038	1.0034	1.0046	1.0275	1.0046	1.0176	1.0185	1.0176
15	60	1.1927	1.0902	1.0283	1.0329	1.0283	1.0158	1.0221	1.0158	1.0111	1.0111	1.0128
20	25	1.1410	1.1465	1.0600	1.0603	1.0600	1.0311	1.0311	1.0409	1.0240	1.0240	1.0290
20	30	1.1410	1.1061	1.0224	1.0224	1.0322	1.0051	1.0080	1.0051	1.0097	1.0097	1.0135
20	35	1.1410	1.1030	0.9758	1.0030	0.9758	0.9856	1.0051	0.9856	0.9858	0.9858	0.9868
20	40	1.1410	1.1033	0.9570	0.9931	0.9570	0.9676	0.9676	0.9722	0.9711	0.9788	0.9711
20	45	1.1410	1.0943	0.9831	0.9831	0.9915	0.9854	0.9854	0.9892	0.9750	0.9750	0.9750
20	50	1.1410	1.0899	0.9909	0.9931	0.9909	0.9851	0.9938	0.9851	0.9854	0.9854	0.9872
20	55	1.1410	1.0911	0.9908	1.0030	0.9908	0.9849	0.9895	0.9849	0.9888	0.9888	0.9924
20	60	1.1410	1.0902	0.9831	0.9831	0.9897	0.9897	1.0011	0.9897	1.0013	1.0013	1.0117
25	30	1.1465	1.1061	1.0546	1.0546	1.1068	1.0289	1.0289	1.0647	1.0367	1.0396	1.0367
25	35	1.1465	1.1030	1.0339	1.0339	1.0521	1.0038	1.0038	1.0129	1.0081	1.0117	1.0081
25	40	1.1465	1.1033	0.9984	0.9984	1.0066	0.9952	1.0033	0.9952	0.9980	0.9994	0.9980
25	45	1.1465	1.0943	0.9618	0.9911	0.9618	0.9827	0.9834	0.9827	0.9877	0.9878	0.9877
25	50	1.1465	1.0899	0.9491	0.9764	0.9491	0.9737	0.9797	0.9737	0.9778	0.9792	0.9778
25	55	1.1465	1.0911	0.9744	0.9764	0.9744	0.9704	0.9704	0.9749	0.9725	0.9728	0.9725
25	60	1.1465	1.0902	0.9764	0.9764	0.9796	0.9720	0.9827	0.9720	0.9671	0.9698	0.9671
30	35	1.1061	1.1030	1.0647	1.0647	1.0759	1.0342	1.0342	1.0350	1.0267	1.0273	1.0267
30	40	1.1061	1.1033	1.0271	1.0271	1.0508	1.0119	1.0119	1.0252	1.0043	1.0043	1.0247
30	45	1.1061	1.0943	0.9995	0.9995	1.0094	0.9989	1.0022	0.9989	0.9897	0.9897	0.9930
30	50	1.1061	1.0899	0.9825	0.9825	0.9936	0.9823	0.9894	0.9823	0.9810	0.9839	0.9810
30	55	1.1061	1.0911	0.9357	0.9357	0.9466	0.9799	0.9805	0.9799	0.9823	0.9847	0.9823
30	60	1.1061	1.0902	0.9113	0.9113	0.9179	0.9658	0.9658	0.9720	0.9773	0.9834	0.9773
35	40	1.1030	1.1033	1.0653	1.0653	1.1151	1.0441	1.0441	1.0650	1.0322	1.0341	1.0322
35	45	1.1030	1.0943	1.0306	1.0306	1.0525	1.0122	1.0122	1.0212	1.0062	1.0062	1.0113
35	50	1.1030	1.0899	1.0129	1.0129	1.0139	1.0027	1.0027	1.0075	0.9980	0.9985	0.9980
35	55	1.1030	1.0911	0.9764	0.9764	0.9990	0.9840	0.9840	0.9849	0.9897	0.9897	0.9923
35	60	1.1030	1.0902	0.9764	0.9764	0.9766	0.9838	0.9838	0.9919	0.9833	0.9833	0.9841
40	45	1.1033	1.0943	1.0614	1.0614	1.1050	1.0483	1.0483	1.0553	1.0392	1.0392	1.0412
40	50	1.1033	1.0899	1.0362	1.0362	1.0647	1.0242	1.0242	1.0267	1.0073	1.0087	1.0073
40	55	1.1033	1.0911	1.0066	1.0066	1.0291	1.0046	1.0068	1.0046	0.9976	0.9976	1.0014
40	60	1.1033	1.0902	0.9761	0.9761	0.9919	0.9897	0.9956	0.9897	0.9899	0.9899	0.9907
45	50	1.0943	1.0899	1.0584	1.0584	1.0804	1.0416	1.0416	1.0508	1.0300	1.0300	1.0300
45	55	1.0943	1.0911	1.0492	1.0492	1.0686	1.0119	1.0170	1.0119	1.0116	1.0116	1.0164
45	60	1.0943	1.0902	1.0181	1.0181	1.0194	0.9977	0.9977	1.0007	0.9956	0.9983	0.9956
50	55	1.0899	1.0911	1.0402	1.0402	1.0744	1.0277	1.0277	1.0499	1.0321	1.0330	1.0321
50	60	1.0899	1.0902	1.0267	1.0267	1.0563	1.0172	1.0172	1.0265	1.0061	1.0068	1.0061

Table 4.8: Scaled separation distances for three-dimensional nested approximate maximin designs.

				Nested n_1-grid			Nested n_2-grid			Grid with nested axes		
n_1	n_2	$d(n_1)$	$d(n_2)$	d	d_1	d_2	d	d_1	d_2	d	d_1	d_2
5	10	1.3693	1.3608	1.2141	1.2748	1.2141	1.2472	1.3240	1.2472	1.2201	1.3279	1.2201
5	15	1.3693	1.3035	1.2069	1.2247	1.2069	1.2203	1.3325	1.2203	1.2001	1.3880	1.2001
5	20	1.3693	1.2862	1.1683	1.2247	1.1683	1.1784	1.1839	1.1784	1.1888	1.2094	1.1888
5	25	1.3693	1.2407	1.1738	1.3693	1.1738	1.1738	1.3693	1.1738	1.1738	1.3693	1.1738
5	30	1.3693	1.2241	1.1689	1.3693	1.1689	1.1706	1.2250	1.1706	1.1612	1.2926	1.1612
5	35	1.3693	1.2074	1.1735	1.2247	1.1735	1.1735	1.3323	1.1735	1.1588	1.2132	1.1588
5	40	1.3693	1.1902	1.1558	1.2247	1.1558	1.1640	1.2146	1.1640	1.1577	1.2222	1.1577
5	45	1.3693	1.1881	1.1560	1.2748	1.1560	1.1560	1.2748	1.1560	1.1560	1.2748	1.1560
5	50	1.3693	1.1830	1.1459	1.2247	1.1459	1.1492	1.3715	1.1492	1.1428	1.3652	1.1428
5	55	1.3693	1.1773	1.1490	1.2247	1.1490	1.1502	1.2642	1.1502	1.1475	1.2695	1.1475
5	60	1.3693	1.1734	1.1463	1.3693	1.1463	1.1487	1.2699	1.1487	1.1378	1.2282	1.1378
10	15	1.3608	1.3035	1.2347	1.2472	1.2347	1.1966	1.1995	1.1966	1.1871	1.1878	1.1871
10	20	1.3608	1.2862	1.1599	1.1706	1.1599	1.1419	1.1710	1.1419	1.1265	1.1265	1.1327
10	25	1.3608	1.2407	1.1564	1.1706	1.1564	1.1319	1.1319	1.1333	1.1365	1.1511	1.1365
10	30	1.3608	1.2241	1.1410	1.1706	1.1410	1.1373	1.1473	1.1373	1.1362	1.1564	1.1362
10	35	1.3608	1.2074	1.1482	1.1706	1.1482	1.1496	1.1672	1.1496	1.1362	1.1362	1.1374
10	40	1.3608	1.1902	1.1359	1.1386	1.1359	1.1427	1.1700	1.1427	1.1389	1.1462	1.1389
10	45	1.3608	1.1881	1.1364	1.1547	1.1364	1.1410	1.1497	1.1410	1.1371	1.1392	1.1371
10	50	1.3608	1.1830	1.1547	1.1547	1.1599	1.1454	1.1606	1.1454	1.1382	1.1425	1.1382
10	55	1.3608	1.1773	1.1425	1.1547	1.1425	1.1425	1.1547	1.1425	1.1425	1.1547	1.1425
10	60	1.3608	1.1734	1.1222	1.1222	1.1268	1.1391	1.1647	1.1391	1.1298	1.2066	1.1298
15	20	1.3035	1.2862	1.2124	1.2124	1.2207	1.1741	1.1741	1.1886	1.1751	1.1751	1.1852
15	25	1.3035	1.2407	1.1510	1.1560	1.1510	1.1341	1.1341	1.1407	1.1451	1.1456	1.1451
15	30	1.3035	1.2241	1.1126	1.1139	1.1126	1.1059	1.1101	1.1059	1.1020	1.1150	1.1020
15	35	1.3035	1.2074	1.1394	1.1394	1.1535	1.1139	1.1163	1.1139	1.1178	1.1178	1.1197
15	40	1.3035	1.1902	1.1135	1.1225	1.1135	1.1117	1.1267	1.1117	1.1128	1.1128	1.1225
15	45	1.3035	1.1881	1.1207	1.1225	1.1207	1.1060	1.1303	1.1060	1.1074	1.1074	1.1105
15	50	1.3035	1.1830	1.1239	1.1309	1.1239	1.1184	1.1221	1.1184	1.1103	1.1157	1.1103
15	55	1.3035	1.1773	1.1218	1.1309	1.1218	1.1112	1.1219	1.1112	1.1223	1.1234	1.1223
15	60	1.3035	1.1734	1.1139	1.1139	1.1155	1.1156	1.1166	1.1156	1.1036	1.1036	1.1091
20	25	1.2862	1.2407	1.1987	1.1987	1.2162	1.1671	1.1671	1.1702	1.1747	1.1747	1.1765
20	30	1.2862	1.2241	1.1732	1.1732	1.1904	1.1401	1.1429	1.1401	1.1386	1.1386	1.1418
20	35	1.2862	1.2074	1.1152	1.1313	1.1152	1.1155	1.1155	1.1162	1.1336	1.1336	1.1368
20	40	1.2862	1.1902	1.0988	1.0988	1.1083	1.0912	1.0945	1.0912	1.0970	1.1097	1.0970
20	45	1.2862	1.1881	1.1249	1.1260	1.1249	1.1013	1.1108	1.1013	1.0952	1.0952	1.1015
20	50	1.2862	1.1830	1.1098	1.1098	1.1145	1.0971	1.0971	1.1000	1.0917	1.0917	1.0945
20	55	1.2862	1.1773	1.1062	1.1098	1.1062	1.1010	1.1010	1.1078	1.0931	1.0931	1.0963
20	60	1.2862	1.1734	1.0988	1.0988	1.1024	1.1123	1.1123	1.1156	1.0933	1.0933	1.0947
25	30	1.2407	1.2241	1.2024	1.2024	1.2192	1.1600	1.1600	1.1651	1.1670	1.1670	1.1685
25	35	1.2407	1.2074	1.1629	1.1629	1.1690	1.1297	1.1313	1.1297	1.1431	1.1431	1.1474
25	40	1.2407	1.1902	1.1407	1.1407	1.1548	1.1227	1.1251	1.1227	1.1193	1.1193	1.1204
25	45	1.2407	1.1881	1.0996	1.1143	1.0996	1.0951	1.0975	1.0951	1.1057	1.1057	1.1108
25	50	1.2407	1.1830	1.0969	1.0990	1.0969	1.0946	1.0963	1.0946	1.0817	1.0817	1.0827
25	55	1.2407	1.1773	1.1182	1.1182	1.1204	1.1021	1.1127	1.1021	1.0875	1.0898	1.0875
25	60	1.2407	1.1734	1.1182	1.1182	1.1193	1.0916	1.0957	1.0916	1.0869	1.0869	1.0895
30	35	1.2241	1.2074	1.1950	1.1950	1.1980	1.1522	1.1522	1.1605	1.1509	1.1513	1.1509
30	40	1.2241	1.1902	1.1596	1.1596	1.1784	1.1258	1.1258	1.1264	1.1255	1.1266	1.1255
30	45	1.2241	1.1881	1.1401	1.1401	1.1477	1.1076	1.1076	1.1091	1.1198	1.1198	1.1204
30	50	1.2241	1.1830	1.1061	1.1146	1.1061	1.1105	1.1147	1.1105	1.1123	1.1123	1.1130
30	55	1.2241	1.1773	1.0861	1.0972	1.0861	1.0987	1.0998	1.0987	1.0935	1.0956	1.0935
30	60	1.2241	1.1734	1.0943	1.0943	1.1022	1.0822	1.0822	1.0835	1.0824	1.0824	1.0831
35	40	1.2074	1.1902	1.1799	1.1799	1.2144	1.1567	1.1567	1.1622	1.1488	1.1493	1.1488
35	45	1.2074	1.1881	1.1670	1.1670	1.1905	1.1365	1.1367	1.1365	1.1266	1.1266	1.1340
35	50	1.2074	1.1830	1.1496	1.1496	1.1568	1.1184	1.1259	1.1184	1.1078	1.1078	1.1123
35	55	1.2074	1.1773	1.1341	1.1341	1.1360	1.1062	1.1062	1.1067	1.1097	1.1097	1.1100
35	60	1.2074	1.1734	1.1071	1.1071	1.1147	1.0952	1.0952	1.0976	1.0892	1.0892	1.0939
40	45	1.1902	1.1881	1.1780	1.1780	1.2141	1.1556	1.1556	1.1574	1.1450	1.1457	1.1450
40	50	1.1902	1.1830	1.1728	1.1728	1.1848	1.1243	1.1243	1.1262	1.1319	1.1329	1.1319
40	55	1.1902	1.1773	1.1485	1.1498	1.1485	1.1146	1.1174	1.1146	1.1146	1.1146	1.1166
40	60	1.1902	1.1734	1.1354	1.1354	1.1437	1.0986	1.0988	1.0986	1.1064	1.1064	1.1075
45	50	1.1881	1.1830	1.1736	1.1736	1.2056	1.1407	1.1407	1.1454	1.1454	1.1454	1.1471
45	55	1.1881	1.1773	1.1780	1.1780	1.2026	1.1225	1.1236	1.1225	1.1225	1.1225	1.1244
45	60	1.1881	1.1734	1.1455	1.1455	1.1494	1.1121	1.1121	1.1126	1.1112	1.1112	1.1152
50	55	1.1830	1.1773	1.1718	1.1718	1.1981	1.1449	1.1449	1.1480	1.1443	1.1443	1.1445
50	60	1.1830	1.1734	1.1606	1.1606	1.2065	1.1254	1.1256	1.1254	1.1152	1.1152	1.1163

Table 4.9: Scaled separation distances for four-dimensional nested approximate maximin designs.

Part II

Subsets of large non-uniform datasets

Chapter 5

Subset selection from large datasets for Kriging modeling

> Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.
>
> (Antoine de Saint-Exupery)

5.1 Introduction

5.1.1 Motivation

Kriging is an interpolation technique that finds its roots in geostatistics. The method is named after Krige, a South-African mining engineer, and is based on his work at the Witwatersrand reef complex (Krige 1951). In the 1960s, the French mathematician Matheron formalized Krige's method (Matheron 1963). Besides geostatistics, Kriging has also found numerous applications in other fields. Sacks et al. (1989b) applied Kriging in the field of deterministic simulation for the design and analysis of computer experiments. Since then, many others followed; see Jones et al. (1998), Jones (2001), Koehler and Owen (1996), Santner et al. (2003), Stehouwer and Den Hertog (1999), and Kleijnen (2009). A basic description of Kriging can be found in Appendix 5.E.

When building a Kriging model, the general intuition is that using more data always results in a better model. Therefore, a large dataset is regarded as a good starting point for building a model. However, when the dataset is already given, there are situations where using only a subset has certain advantages. Especially when the large given dataset is non-uniformly distributed over the whole design space, problems can occur. In this chapter, we analyze and test methods to select subsets in order to reduce these problems.

Large non-uniform datasets can occur in several situations. The first situation occurs when there is a set of "legacy" data (Srivastava et al. 2004). Legacy datasets contain results of experiments, simulations or measurements performed in the past. These results are stored for future use to avoid having to generate them again. This is especially useful if there are many results or if the expenses to obtain these results are high. As this data is not generated especially for making a global model, it is often not uniformly distributed over the whole design space. Another cause of non-uniformity in legacy datasets can be that they contain measurements of a system that cannot be fully controlled.

A second situation occurs when the data is the result of a sequential optimization method. These methods often generate more data points near potential optima than in other regions (Booker et al. 1999). If we want to use this data to fit a global metamodel, we should account for clusters of points.

Thirdly, non-uniform data occurs if models are "coupled" (Husslage et al. 2003). We call two models coupled if the output of the first model is input for the second. If we want to construct a metamodel of both models, we can use a space-filling design of experiments for the first model. It could be argued that we could also do this for the second model. In some cases, however, it is better to use the output of the first model. Although the input of the first model is space-filling, its output is often not. When we want to construct a metamodel of the second model, we thus have to use a non-uniform dataset.

Altogether, these are a number of situations where we can come across large non-uniform datasets. Using these sets directly to build our model can impose problems that can often be solved by using a uniform subset. We call a subset uniform if the input data of the data points in the subset are "evenly spread" over the entire design space. Whether or not using a uniform subset is better, depends not only on the dataset but also on the chosen modeling method.

Important reasons for using a subset instead of the complete dataset are the following:

- **Creating training and validation set**
  Validation is the most common reason for using only part of the data as training data. Splitting a dataset into a training and a validation set is a well-known validation method and is often done randomly (Cherkassky and Mulier 1998, Golbraikh and Tropsha 2002). In the field of design of computer experiments (DoCE), however, a consensus has been reached that designs used for deterministic computer experiments should be space-filling (Simpson et al. 2001). A design is called space-filling if it fills the whole design space. For a given number of design points, the design space is best filled if the design points are evenly spread over the whole design space. Therefore, it would be a good idea to also take a uniform subset when selecting a training set from an existing dataset (Golbraikh et al. 2003).

- **Time savings**
  A second reason could be the reduction in time necessary to fit the Kriging model. This is certainly an important issue as its time-consumption is generally regarded as one of the main drawbacks of the Kriging method (Jin et al. 2001). With current implementations and computing power, it is sometimes even impossible to fit a Kriging model using all available data. For instance, when using the Matlab toolbox DACE provided by Lophaven et al. (2002) on a PC with a 2.4 GHz Pentium 4 processor, we encountered problems for datasets containing 3000 points or more.

  Especially the inversion of the correlation matrix that is part of Kriging can impose problems as this requires much time and memory capacity. Kriging, however, is not the only method that can benefit from using less data. Genetic programming could also significantly benefit as it requires a lot of models to be fitted to a training set (Koza 1992, Banzhaf et al. 1998). Often this forms a large part of the total computation time and is thus worthwhile to reduce.

  Besides fitting a Kriging model, the prediction of new points also requires less time because using less training data results in simpler models. This is mainly because the number of terms in the Kriging model depends on the size of the training set. The same holds for Lagrange interpolation. Especially for on-line monitoring and optimization, finding a fast and simple model is important (Kordon 2006).

- **Avoiding numerical inaccuracies**
  A common property of large datasets is that they are non-uniform, which implies that they may contain points that lie very close together. This property can make the corresponding correlation matrix ill-conditioned (Davis and Morris 1997, Booker et al. 1999). Solving a linear system with an ill-conditioned matrix can cause major numerical inaccuracies. The optimization of the Kriging parameters requires solving a linear system and can thus be inaccurate when data points lie close together. Removing points from the dataset can avoid an ill-conditioned correlation matrix and can, therefore, improve the numerical accuracy of the Kriging model.

- **Improving robustness**
  Robustness of the Kriging model with respect to errors in the output data can be negatively influenced when data points lie close together. Siem and Den Hertog (2007) show that points that are close together can get assigned relatively large Kriging weights. Due to these large weights, errors in the output values at these points can have a large influence on the output value of the Kriging model at other points. Removing some of the points may result in lower weights for the remaining points and thus a smaller effect of errors. Therefore, we can sometimes improve this kind of robustness by using only a subset instead of the whole dataset.

These different motivations for subset selection require us to look at different performance criteria. In Section 5.4.2, we describe the criteria used in this chapter to measure the effects of subset selection on the four aspects: accuracy, CPU-time, susceptibility to numerical inaccuracies, and robustness with respect to errors in the output data.

For most of these aspects, it is important that the selected subset is uniform. Selecting a uniform set of points also occurs in other problems like Design of Computer Experiments (DoCE) and the dispersion problem. Although these are different problems, we can use some ideas from the fields of DoCE and dispersion problems in determining our subset. To see why and how we can use these ideas, we look at the similarities and differences between the problems in the following two sections.

5.1.2 Design of Computer Experiments

The main reason for using ideas from DoCE is that DoCE has the same aim as subset selection. In both problems the aim is to select a training set that will produce a model that most accurately approximates the function or process underlying the data. In practice, however, we do not have an explicit description of the underlying function or process, which makes it impossible to directly optimize this objective. Instead, we optimize certain properties of the training set that generally improve the quality of the resulting model. Of these properties, the most frequently used is space-fillingness (Simpson et al. 2001).

Another similarity is that in both problems we often have a restriction on the number of points we can use. In DoCE the restriction is caused by the large computation time per design point. The reasons for limiting the number of points through subset selection were given in Section 5.1.1.

Besides similarities, there are also differences between the two problems. The first difference is the set of points we can choose from. In DoCE, we can select all points in the design space. Sometimes we choose to restrict ourselves to points with a certain structure or property, such as a Latin hypercube (McKay et al. 1979, Stein 1987) or an orthogonal array (Owen 1992, Tang 1993), but even then we are free to select a structure or property. In subset selection, we can only choose from the points in the original dataset. This implies that the subset can cover the design space only as well as the original dataset does. It could be argued that sometimes additional points can be evaluated to improve for instance uniformity. Nevertheless, we do not take this option into account, as it is beyond the scope of this chapter.

The second difference is that the output values of all possible points are known in the case of subset selection. This in contrast to DoCE where we have to select our points without knowing their output values. Even in sequential DoCE, only the output values of previously selected and evaluated points are known. Using the output information in

subset selection will most likely result in training sets that give more accurate models.

These differences show that we cannot directly use results from DoCE. In the paper by Srivastava et al. (2004), however, a method is described that uses a space-filling DoCE to determine a uniform subset. The main idea is to take a randomized orthogonal array and to select for each point of this orthogonal array, the data point closest to it. This idea can also be applied to other types of space-filling DoCEs. However, to limit the number of methods compared in this chapter, we consider only orthogonal arrays. In Section 5.3.1, the method of Srivastava et al. (2004) is described in more detail.

5.1.3 Dispersion problems

The dispersion problem can be described as follows. Given n possible locations, locate $m < n$ facilities such that some function of the distances between the facilities is maximized (Ravi et al. 1994). Two commonly used functions are the minimum and the sum of the distances (Erkut and Neuman 1989). The first case is often referred to as MAXMIN and the second as MAXSUM. Both versions of the dispersion problem are strongly NP-hard. This was independently proven by Erkut (1990) and Ghosh (1996) for the MAXMIN problem and by Hansen and Moon (1994) and Kuo et al. (1993) for the MAXSUM problem. MAXMIN is also used as a measure of space-fillingness in DoCE. When we see locations as points, we can thus reformulate the problem as follows. Given n possible points, select a uniform subset of m points.

If we focus only on uniformity when selecting our subset and use MAXSUM or MAXMIN to measure the uniformity then subset selection is the same as the MAXSUM or MAXMIN dispersion problem. Even so, this does not mean that we can directly apply all facility location algorithms and heuristics to the problem of subset selection. The two main reasons are the following.

Firstly, we have to consider the dimensionality of the design space. In dispersion problems, the locations are often points in two- or three-dimensional space. Subset selection on the other hand applies to datasets of any dimension. In practice, the dimension is often considerably larger than 3. This means that we have to determine whether an algorithm for the dispersion problem extends to higher dimensions, before we can use it in subset selection.

Secondly, many algorithms give only a solution in a reasonable amount of time for sets containing relatively few (a couple of dozen or a few hundred) points (Agca et al. 2000, Pisinger 2006). Subset selection on the other hand may be necessary when there are thousands of points. Ideas that give no computational problems for dispersion problems could require too much time or memory capacity for subset selection.

Keeping these two aspects in mind, we can apply some methods originally developed for the dispersion problem directly to subset selection. The greedy MAXMIN method

(Ravi et al. 1991) is one such method. It starts by selecting for the subset the two points furthest away from each other. Then iteratively, the point furthest away from the already selected points is added to the subset. Although this method is quite simple, Ravi et al. (1991) have shown that if the triangle inequality holds, the heuristic gives an approximation ratio of 2; i.e., the MAXMIN distance of the resulting subset is at most twice the MAXMIN distance of the optimal subset. Furthermore, due to its simplicity the method is quite fast and thus suitable for large datasets. Therefore, we also use this method for uniform subset selection and describe it in more detail in Section 5.3.3.

5.1.4 Overview

The structure of the rest of this chapter is as follows. In Section 5.2, we show through a simple example that taking a subset can indeed improve the aspects mentioned in this introduction. New and current methods to select a subset are described in Section 5.3. The main difference between the new and current methods is that the new methods also use the output data. To determine whether this method results in subsets that produce beter Kriging models, we compare the methods in Section 5.4. Furthermore, we compare our Kriging models with radial basis function (RBF) models fitted to the complete dataset. The advantage of RBF models is that they have shown good fits for both stochastic and deterministic functions (Powell 1987) and that they require significantly less time to be fitted than Kriging models (Jin et al. 2001). RBF models can therefore be fitted to datasets for which Kriging models would be too time-consuming. In Section 5.4, we also compare the performance, in terms of CPU-time and accuracy, of Kriging models fitted to a subset and RBF models fitted to the complete dataset. A basic description of RBF models can be found in Appendix 5.F. Finally, Section 5.5 contains conclusions and suggestions for further research.

5.2 Example

To show that selecting a subset can really improve the aspects mentioned in Section 5.1.1, we introduce the following simple artificial example. We try to approximate the six-hump camel-back function (Dixon and Szegö 1978):

$$f(x) = 4x_1^2 - 2.1x_2^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4,$$

with $x_1 \in [-2, 2]$ and $x_2 \in [-1, 1]$. As our dataset, we take a maximin LHD of 20 points (Van Dam et al. 2007) with four additional points close to an existing point as depicted in Figure 5.1. By adding these four points, we create a cluster of points in the dataset. As we mentioned in Section 5.1.1, this can cause several problems.

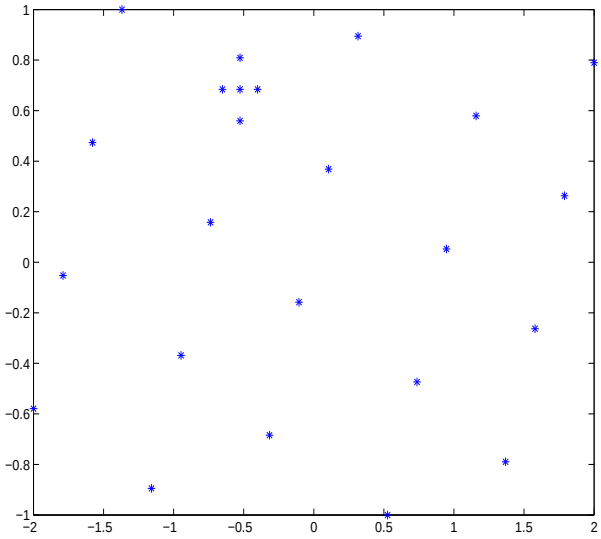


Figure 5.1: Maximin LHD of 20 points with 4 additional points.

We test the effect of selecting a uniform subset by fitting Kriging models to the datasets with and without the four additional points. To measure the effects of taking a subset, we use the performance measures described in Section 5.4.2. The results for both Kriging models are given in Table 5.1. For all measures it holds that a lower value is better.

	Without four additional points	With four additional points
RMSE	0.48	0.51
Maximum Error	2.03	2.46
Condition Number	76	1491564
Average Robustness	0.97	8.36
Max. Robustness	1.57	100.66

Table 5.1: Performance of Kriging models fitted to the datasets with and without the four additional points.

Both the Root Mean Squared Error (RMSE) and the Maximum Error show that the accuracy of the Kriging model without the four points is better than that of the model with the additional four points. The latter Kriging model focusses more on fitting accurately in the region of these additional points and, as a result, is more accurate in this subregion. However, this additional accuracy is at the expense of accuracy in other regions, which deteriorates the overall accuracy. When optimization is the goal of the Kriging metamodel and the cluster is close the optimum, this behavior is often not a problem. However, when we are still exploring the design space for potential optima or if we are interested in a globally accurate metamodel, this behavior may be less desirable.

When we compare the condition numbers, we see a very large difference. This shows that the additional four points make the resulting Kriging model much more susceptible to numerical inaccuracies. Finally, the larger maximal and average robustness values indicate that this model is also less robust with respect to errors in the output data.

This simple example thus shows that removing some points can improve the quality of the resulting Kriging model. Determining which points to remove is quite easy in this case. In practice when the number of points is much larger, this becomes less straightforward. Therefore, in the next section, we introduce some current and new methods to select points from a dataset.

5.3 Subset selection methods

5.3.1 Orthogonal Array Selection

Srivastava et al. (2004) discuss the problem of selecting 500 or fewer points from a dataset containing 2490 points in 25-dimensional space. The selected points are used to create a Kriging model and the remaining points are used to check the accuracy of this model. Results are compared for different numbers of selected points and compared with results for quadratic response-surface models.

To select the points, first a randomized orthogonal array is constructed. Then for each point of the orthogonal array a "nearest neighbor" is determined, i.e., the data point closest to the orthogonal array point. All "nearest neighbors" together form the set of selected points. It is possible that this set contains fewer points than the orthogonal array because one point in the dataset can be the "nearest neighbor" of several points of the orthogonal array. As we do not know in advance how often this happens, we cannot set the subset size exactly. However, we do know that the size of the orthogonal array is an upper bound for the subset size.

The idea of selecting "nearest neighbors" to points of an orthogonal array can be extended to other types of space-filling DoCEs. For instance, we could use maximin LHDs instead of orthogonal arrays. In order to limit the number of methods compared in this chapter, we do not use Maximin LHD Selection as a separate method. We do, however, use it to generate starting solutions for the Sequential Selection methods described in Section 5.3.5. The (approximate) maximin LHDs we use are obtained through the ESE algorithm of Jin et al. (2005).

Another reason for using Orthogonal Array Selection is that it enables us to compare our results with those found by Srivastava et al. (2004). However, it is unclear which method they exactly used to construct the randomized orthogonal arrays. We have decided to use the method described on page 131 in Hedayat et al. (1999) to construct 61- and 113-dimensional orthogonal arrays of respectively 250 and 686 points. Using

a uniform distribution, we randomly select 6 or 25 dimensions to obtain randomized orthogonal arrays that are suitable for our test problems. As different orthogonal arrays may result in different subsets, the choice of orthogonal array could affect the quality of the resulting subset. For each dataset, we therefore determine ten subsets using ten different orthogonal arrays. We thus hope to determine whether the choice of orthogonal arrays indeed affects the quality of the subset and the resulting model.

Notice, that this method requires us to find a suitable orthogonal array. This could be a problem as an orthogonal array of the desired number of points and dimensions might not be known. However, due to the flexibility in the desired number of points of the orthogonal array, this problem can be easily resolved in most cases.

5.3.2 Fast Exchange Algorithm

The Fast Exchange Algorithm was introduced by Lam et al. (2002) to select a subset from a very large database containing characteristics of molecules. The goal of the algorithm is to select a subset that covers the numerical space described by the database. However, as this space is often high-dimensional, nearly all these databases are sparse. This makes it impossible to densely cover the whole space with a reasonably sized subset of points. The algorithm therefore focusses on selecting the subset such that it is space-filling in low-dimensional projections of the space.

Globally, the algorithm works as follows. First, sets of bins are constructed for all 1-D, 2-D and 3-D projections. These bins form a partition of the projected design spaces and their sizes depend partly on the distribution of the data. The aim is now to select a subset such that each bin contains approximately the same number of points. To measure how close a certain subset is to this aim, the uniform cell coverage criterion is used. This criterion has the advantage that we can easily calculate the effect of adding or removing a particular point from the subset on the criterion value.

The Fast Exchange Algorithm now tries to find the best subset by exchanging points that are inside and outside the subset. This is done in two steps. In the first step, the best point to add to the current subset is determined. The resulting subset thus has a size of $n+1$. In the second step, the best point to remove from this subset is determined. The main difference with the basic exchange algorithm is in these two steps. In the basic exchange algorithm, all possible points outside the subset are tried in the first step, and all points inside the subset are tried in the second step. As a result, the basic exchange algorithm loops through all points in the dataset, before an exchange is made. The Fast Exchange Algorithm, on the other hand, uses a distribution of the improvements to select the exchange. At the beginning of the algorithm, one hundred additions and removals of points are tried to estimate the two distributions of the improvements in Steps 1 and 2. During the algorithm when more additions and removals are tried, this estimated

distribution is updated. A point is added or removed if it belongs to the upper tail of this distribution. This often implies that we loop through much fewer points before an exchange is made.

5.3.3 Greedy MAXMIN Selection

The greedy MAXMIN method comes from the field of dispersion problems. The k-dimensional data points are therefore seen as points in k-dimensional space. As the name indicates, it seeks to maximize the minimal Euclidean distance between any two points in the chosen subset. It does this in essentially the same way as the "furthest point outside the neighborhood" heuristic described in Steuer (1986).

Let us denote the total dataset by N and the Euclidean distance between points i and j by $d_{i,j}$. We can then describe this simple heuristic as follows (Ravi et al. 1991):

1. Take $S = \emptyset$.
2. Let (i, j) be such that $d_{i,j}$ is maximal.
3. Add i and j to the set S.
4. Find a point $i \in N \backslash S$ such that $\min_{j \in S} d_{i,j}$ is maximum among the points in $N \backslash S$.
5. Add point i to S.
6. Repeat Steps 4 and 5 until the set S contains the desired number of points.

In Step 4, we thus determine the point farthest away from the already chosen points. Although the heuristic is quite simple, Ravi et al. (1991) have shown that if the triangle inequality holds, the heuristic gives an approximation ratio of 2. Furthermore, they have proven that, unless P=NP, no polynomial-time relative approximation algorithm can provide a better performance.

Notice, that if the domains of the input variables are of different magnitudes, it is better to normalize the domains in order to obtain a uniform subset. For all datasets in this chapter, we therefore normalize the domains before applying this method.

5.3.4 Greedy DELETION Algorithm

Besides MAXMIN Selection, we use another simple greedy algorithm. This method constructs a subset by iteratively removing one point of the pair of points with the smallest Euclidean distance between them. To decide which of the two points should be removed, we look for each point at the distance to its second closest point. The point for which this distance is smallest is removed from the dataset.

Using the same notation as for MAXMIN Selection, we can describe the DELETION Algorithm as follows:

1. Take $S = N$.
2. Let (i, j) be such that $d_{i,j}$ is minimal.
3. Determine $c_i = \min_{k \in S/\{i,j\}} d_{i,k}$ and $c_j = \min_{k \in S/\{i,j\}} d_{j,k}$.
4. Remove the point with the lowest c value from the set S.
5. Repeat Steps 2, 3, and 4 until the set S contains the desired number of points.

As this method constructs a subset through the removal of points from the total dataset, the method requires fewer iterations for larger subsets—unlike, for instance, MAXMIN and Sequential Selection, which build a subset by adding points to an initially empty set. Furthermore, for this method it is also useful to normalize the domains of the variables in order to obtain a uniform subset.

5.3.5 Sequential Selection

In the four methods discussed in the previous subsections, the output values of the points are not used in the selection process. However, as the selected points are used to determine a model of the output, it seems a good idea to explicitly use the output information in selecting the training points. As we previously mentioned, this is an important difference with DoCE where generating the output values is expensive. In our situation, we work with a given dataset with known output values and can thus use the output values at no additional computational costs.

We use the output values in the following way. First, we apply Maximin LHD Selection (see Section 5.3.1) using only the *input* values to determine an initial training set. Then we fit a Kriging model to the training set and calculate the prediction error at the non-selected points. The idea is now to add non-selected points with a large error to the training set. However, if the Kriging model is inaccurate in a certain region, all points in that region have a large error. Consequently, simply selecting, for instance, n_1 points with the highest error might result in adding points that are clustered together in one or a few regions.

To reduce this problem, we use two methods. The first method is to add one point at a time. This means that the point with the largest error is added to the training set and then the Kriging model is refitted. This method completely solves the above problem, but is unfortunately rather time-consuming as we fit a new Kriging model after every added point. Therefore, we also develop a second method. This method determines the $n_2 > n_1$ worst points and then uses the greedy MAXMIN heuristic described in Section 5.3.3 to

select a uniform subset of n_1 points. These n_1 points are then added to the training set. In this chapter, we use $n_1 = 10$ and $n_2 = 40$ to test this method. We have not tested the influence of the choice of n_1 and n_2 on the quality of the resulting subset. Determining this possible influence and a suitable method for selecting these values remain topics for further research. To determine whether the above problem really occurs and results in worse models, we also test the method where we simply add the $n_1 = 10$ worst points.

For each method, the selection and addition steps are repeated until the desired number of points are selected. Because we use Maximin LHD Selection to determine the initial training set, the initial training set might not always be equal to the size of the LHD. We take this into account when determining the number of selection and addition steps. As we add 10 points per step in the second method, it is not always possible to determine this such that we exactly get the desired number of points. In these cases, we select the largest attainable number of points smaller than the desired number of points.

As these methods are sequential methods, we might decide to use another stopping criterion to determine how often we repeat the steps. An example would be to stop when the maximum error in the non-selected points is below a certain predefined value. In this chapter, however, we choose to predefine the number of selected points for easier comparison between the different methods.

Finally, we want to make a remark on the implementation of this method. As mentioned, the method requires that a new Kriging model is fitted in every iteration. As this is the most time consuming part of the method, we try to speed up the optimization of the Kriging parameters. We manage to do this by using the fact that subsets in consecutive iterations are quite similar because each set is obtained by adding points to the previous set. This makes it quite plausible that the optimal parameter settings of the Kriging models fitted to both subsets are also similar. The optimal parameter setting of the Kriging model in the previous iteration therefore seems a good starting solution for fitting the Kriging model in the current iteration. Some tests show that using the optimal parameters of the previous iteration instead of a fixed starting solution, can indeed reduce the time needed to select a subset by 15 percent. Therefore, we use this implementation for the Sequential Selection method in the remainder of the chapter.

5.4 Computational results

5.4.1 Subset selection methods

We compare the following eight methods:

- Random Selection (RS).
- Orthogonal Array Selection (OAS).

- Fast Exchange Algorithm (FEX).
- Greedy MAXMIN Selection (MAXMIN).
- Greedy DELETION Algorithm (DELETION).
- Sequential Selection adding one point at a time (SS1).
- Sequential Selection adding ten worst points at a time (SS10).
- Sequential Selection adding ten uniform points from the 40 worst points at a time (SS1040).

The first method randomly selects the required number of points without taking into account any of their properties. The performance of this method is used as a reference for the performance of the other methods.

To test the effect of the training set size on the resulting Kriging model, we select subsets of 250, 350, and 500 points from the datasets that are described in Section 5.4.3. As the SS1 method is rather time-consuming, we generate only subsets of 250 and 350 points using this method. Furthermore for Orthogonal Array Selection, we cannot determine the exact subset size in advance. By using orthogonal arrays of 250 and 686 points, we get subsets of at most these amounts of points. As mentioned in Section 5.3.1, we generate subsets using ten different orthogonal arrays to test the effect of the choice of orthogonal array on the resulting subset. For each performance measure, we therefore report the average, minimum, and maximum over these ten subsets.

5.4.2 Performance measures

The different reasons for selecting a subset require that we use several performance measures to determine how good a certain subset is. We describe these performance measures and our motivation for choosing them.

RMSE and Maximum Error

Our first reason is the selection of a subset that results in an accurate model. We thus need to measure the accuracy of the resulting Kriging model. Common measures are the Average Error, Root Mean Squared Error (RMSE), and Maximum Error. We do not measure these on the training data because the Kriging model interpolates through the training data, so these measures are always zero. Instead, we need a validation set, which could either be the remaining dataset or a separately generated set.

In the case of a real-life dataset, we have only the first option. If the original dataset is non-uniform, the remaining dataset might not be particularly suited to measure the

overall accuracy of the model as the accuracy in densely populated regions weighs heavier than accuracy in sparsely populated regions. This problem could be reduced by taking the Weighted Average Error or Weighted RMSE with weights determined such that errors in points in sparse regions get more weight than errors in points in dense regions. However, it is unclear how exactly the weights should be determined to achieve this effect. Therefore, we use the usual RMSE in this chapter, although we are aware of its deficiency. Finding a method for determining the weights remains a worthwhile subject for further research. Besides the RMSE, we will also use the Maximum Error to compare the accuracy of different Kriging models.

For artificial datasets, the best option is to separately generate a uniform validation set. This way we avoid any problems caused by densely versus sparsely populated regions. In this chapter, a uniform grid is therefore used as the validation set.

Time for model fitting

To determine the reduction in time necessary to fit the Kriging model, we simply use the amount of required CPU time. For fitting the Kriging model, we used the Matlab toolbox DACE provided by Lophaven et al. (2002). All calculations were performed on a PC with a 2.4 GHz Pentium 4 processor; the CPU times are reported in minutes.

Time for subset selection

Besides fitting the Kriging model, the time necessary for constructing the subset is also measured. This is needed to determine whether the time gained for model fitting is not outweighed by the additional time needed to select a good subset. Note that we coded all subset selection methods in Matlab. By using the same program for each method, we aim to make a fair comparison between the different methods. The results are again reported in minutes.

Condition number

We also want to avoid ill-conditioned correlation matrices by selecting a subset. To determine if a correlation matrix is ill-conditioned, we use its condition number. The condition number κ of a square matrix C is defined as (Golub and Van Loan 1996):

$$\kappa(C) = \frac{\sigma_{\max}(C)}{\sigma_{\min}(C)},$$

where $\sigma_{\max}(C)$ and $\sigma_{\min}(C)$ are the largest and smallest singular values of C. A nonnegative scalar σ is a singular value of C is there exist two nonzero vectors u and v such that $Cv = \sigma u$ and $C^\top u = \sigma v$. The condition number is a measure of the worst case loss in precision when solving a linear system. A matrix with a large condition number is called

ill-conditioned and is thus susceptible to numerical inaccuracies. An orthogonal matrix has κ equal to 1.

Average and maximum robustness

Finally, we want to measure the influence of errors in the output data on the resulting Kriging model. Depending on the kind of data we are dealing with, these errors can have different causes. If, for instance, the data is obtained using a simulation model, the error in the output can result from model errors or numerical errors. If a Kriging model is robust, a small inaccuracy in the output data does not cause the Kriging model to deviate much from the Kriging model fitted to the correct data.

To measure the robustness of a Kriging model with respect to errors in the output data, Siem and Den Hertog (2007) suggest to use $||c(x)||$ as a measure of robustness at the point x, where $c(x)$ is the vector of Kriging weights and $||.||$ is the Euclidean norm. (See Appendix 5.E for a basic description of the Kriging method.) The general idea behind this measure is the following. Let us assume that we have a training dataset $x^1, \ldots, x^n$ for which the true output values are $y^1, \ldots, y^n$, but the measured output values are $y^1+\varepsilon^1, \ldots, y^n+\varepsilon^n$. An upper bound for the deviation from the true Kriging model at point x is then approximated by $||\varepsilon||||c(x)||$ where $\varepsilon = [\varepsilon^1 \ldots \varepsilon^n]^\top$. The measure $||c(x)||$ thus gives an indication of the factor by which errors in the output values of the dataset are multiplied when calculating the output value of the Kriging model at point x.

This robustness-criterion determines the robustness only at a certain point x. To measure the overall robustness, we use the maximal and average value of the robustness values at a set of points. As we do not need to know the true output values at these points, we select them on a rectangular grid—even in the case of a real-life dataset.

5.4.3 Datasets

To test the different selection methods, we use two types of datasets: artificial datasets with known underlying function and real-life datasets for which the underlying function is unknown. For real-life datasets, it is clearly more difficult to judge the accuracy of the resulting model, especially as these datasets are often unstructured and non-uniform. This also holds for the real-life dataset used in this chapter. The dataset was originally used in the design of the High Speed Civil Transport (HSCT) aircraft and contains 2490 points in 25-dimensional space (Padula et al. 1996). As this dataset is used by Srivastava et al. (2004), we use it to make a comparison with their results. Before using the dataset, we first remove the duplicate points, which leaves us with a dataset of 2487 unique points.

Artificial datasets are generally constructed by drawing points from the design space and calculating their value for a known function. In this chapter, we use the six-variable

Hartman-6 function (Dixon and Szegö 1978) which is defined as follows:

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} \alpha_{ij}(x_j - p_{ij})^2\right)$$

with $x_j \in [0,1]$, $j = 1,\ldots,6$, and where the parameters are given in Table 5.2. The

i	α_{ij}, $j=1,\ldots,6$						c_i	p_{ij}, $j=1,\ldots,6$					
1	10	3	17	3.5	1.7	8	1	.1312	.1696	.5569	.0124	.8283	.5886
2	.05	10	17	0.7	8	14	1.2	.2329	.4135	.8307	.3736	.1004	.9991
3	3	3.5	1.7	10	17	8	3	.2348	.1451	.3522	.2883	.3047	.6650
4	17	8	.05	10	0.1	14	3.2	.4047	.8828	.8732	.5743	.1091	.0381

Table 5.2: Parameter values of the six-variable Hartman-6 function.

output values of the Hartman-6 function range between approximately -3.23 and 0. The Hartman-6 function is chosen because it is a widely used test problem with a relatively large number of dimensions (Wang et al. 2001, Jin et al. 2002). As most real-life datasets are high-dimensional, we prefer this test problem over other widely used but lower-dimensional test problems.

To sample the points from the design space, the following methods are used:

1. Sample the points from a uniform distribution on the ranges of the variables x_j, $j = 1,\ldots,6$.

2. Divide the design space into m^k equally sized cells, where k is the dimension. Determine m^k numbers that sum up to the total size of the dataset. Randomly assign these numbers to the cells. For each cell use a uniform distribution to sample the assigned number of points within the cell. By adjusting the distribution of the number of points, we can vary the uniformity of the dataset.

We refer to datasets constructed with the first method as uniform datasets. We use the second method to construct non-uniform datasets of 2000, 5000, and 10000 points. For all sizes, m is set equal to 3 which means that we have 729 cells. The distributions of the number of points per cell are given in Table 5.3.

For 2000 points, it is still possible to fit a Kriging model and a RBF model to the complete dataset. Therefore, we use these datasets to test if taking a subset is really better than using the complete set. For the datasets of 5000 points, we were no longer able to fit a Kriging model to the complete dataset with the DACE toolbox. A RBF model, however, could still be fitted to the complete dataset. Hence, we use these datasets to test how a Kriging model fitted to a subset compares to a RBF model fitted to the complete set. When using 10000 points, we could no longer fit a RBF model to the

Number of cells	30	30	74	238	431	595
Points per cell for dataset of 2000 points	20	10		1	2	
Points per cell for dataset of 5000 points	50	25	5			4
Points per cell for dataset of 10000 points	100	50	10			8

Table 5.3: Distributions of the number of points per cell used to construct non-uniform datasets of 2000, 5000, and 10000 points.

complete dataset. We thus use these datasets to compare the results of the different subset selection methods for Kriging only.

For each size and type of artificial dataset, we randomly generated 50 datasets. In the comparisons, we use the average, minimum, and maximum performance over these datasets for all performance measures.

5.4.4 Results for artificial datasets of 2000 points

For datasets of 2000 points, it is still possible to fit a Kriging and a RBF model to the complete dataset. By taking a subset, we aim to improve the different performance measures. In Table 5.4 in Appendix 5.A, we give detailed results of the performance measures.

When looking at the RMSE and the Maximum Error, the Kriging model fitted to the complete dataset is the most accurate model for both uniform and non-uniform datasets. Unfortunately, it requires by far the most time to fit. Fitting a RBF model is much faster than Kriging and also faster than most subset selection methods if we take into account the times necessary to select the subsets. However, some of the Kriging models fitted to a subset are more accurate than the RBF model. For subsets of 200 points, the RMSE of most Kriging models is still worse, but SS1, SS10, and SS1040 produce almost equally good or even slightly better Kriging models. For 350, points SS1, SS10, and SS1040 result in Kriging models with a much lower RMSE than the RBF model. When using 500 points, these methods result in Kriging models with even lower RMSE values, as expected. For non-uniform datasets, the best subsets obtained with OAS also result in Kriging models with a lower RMSE. The Maximum Error shows even more cases where Kriging models fitted to subsets are better than the RBF model.

When we consider the subset selection times, two methods that require hardly any time are RAND and OAS. Nevertheless, using RAND is not a very good choice, as the resulting Kriging models are rather inaccurate. The OAS method results in considerably more accurate Kriging models. Subset methods that result in even more accurate Kriging models are SS1, SS10, and SS1040. When we compare these methods for each subset

size separately, SS1 results in the lowest RMSE and SS1040 in the lowest Maximum Error. However, when we take into account the time to select the subset, SS1 is the least attractive method. Selecting a subset of 350 points requires even more time than fitting a Kriging model to the complete dataset. The methods SS10 and SS1040 are also relatively time-consuming compared with RAND, MAXMIN, FEX, and OAS, but they are more accurate and still have a considerable time reduction compared with Kriging fitted to the complete dataset.

Considering the condition numbers, SS1, SS1040, MAXMIN, DELETION, and OAS perform very well. The good performance of the latter two methods might be explained by the fact that they only focus on optimizing the uniformity of the input data which influences the condition number of the correlation matrix. In contrast, it is remarkable that the FEX method, which has the same focus, performs considerably worse. This seems to indicate that FEX does not succeed as well in selecting a uniform subset as SS1, SS1040, MAXMIN, DELETION, and OAS do. Furthermore, we see that SS1040 gives better result than SS10, as expected. Taking 10 uniformly selected points from the 40 worst instead of just taking the 10 worst points, thus seems to result in a more uniform subset. Most important, however, is that all subset selection methods result in a correlation matrix with a considerably better condition number than that of the correlation matrix of the complete dataset. Therefore, taking a subset clearly reduces the chance of numerical inaccuracies.

For robustness, taking a subset also considerably improves the performance measure. When we look at the maximum robustness, then SS1, SS1040, MAXIMIN, DELETION, and OAS perform best. For the average robustness SS10 also performs well.

Besides these performance measures, we present the actual sizes of the subsets, because for OAS the number of points in the subset is not known exactly in advance. The results show that for 250 points, the number of selected points is quite close to the size of the orthogonal array. For the orthogonal arrays of 686 points, the difference is much larger. As we mentioned in Section 5.3.5, SS10 and SS1040 may result in slightly different subset sizes as Maximin LHD Selection is used for the initial training set. Even so, the results show that this does not occur for the tested datasets.

When we compare uniform and non-uniform datasets, we see almost the same results for most performance measures. The ranking of the different methods is generally the same. The main differences are between the condition number and robustness of the Kriging models fitted on the complete dataset. On these aspects, the Kriging models fitted on the uniform datasets perform better than the Kriging models fitted on the non-uniform datasets. For the Kriging models fitted on the subsets, the differences are much smaller. This seems to indicate that the difference in uniformity between the subsets is smaller than the difference in uniformity between the two types of complete sets.

The subset size is also a factor that influences the performance measures. When we compare the performance measures, we see two opposite effects. The RMSE and Maximum Error improve when larger subsets are used, whereas the other performance measures worsen. To determine the 'best' subset size, we thus have to make a trade-off between accuracy and robustness, computer time, and numerical accuracy. Which trade-off is best depends on the importance of these aspects and the estimated accuracy of the dataset.

Finally, we examine the effect of the choice of an orthogonal array on the results of the OAS method. Therefore, we compare the mean, minimum, and maximum of the performance measures over ten subsets obtained with ten different orthogonal arrays. The differences in these results show that the choice of an orthogonal array has a clear effect on the quality of the resulting subset and Kriging model. When using the OAS method, it is therefore better not to use a single random orthogonal array. It would be better to carefully choose a suitable orthogonal array or to create multiple subsets using different (random) orthogonal arrays, and then select the best subset. Which option works best and how exactly we should implement them, is not immediately clear and are thus interesting subjects for further research.

5.4.5 Results for artificial datasets of 5000 and 10000 points

For datasets of 5000 points, we could no longer fit a Kriging model to the complete dataset. A RBF model, on the other hand, could still be fitted. As most results are quite similar to the results of the datasets containing 2000 points, we focus on the differences. Detailed results can be found in Table 5.5 in Appendix 5.B.

When comparing the RMSE, we see that now only SS10 and SS1040 with a subset of 500 points result in Kriging models with RMSE values approximately to the RBF model. The Maximum Error, however, is considerably lower for all subsets of SS1, SS10, and SS1040 and for the large subsets obtained with OAS. Combining these two measures, the Kriging models based on subsets of 500 points obtained with SS10 and SS1040 seem to be most accurate. Unfortunately, the time required to select these subsets and to fit the model is considerably more than for the RBF model. We thus have a trade-off between accuracy and the required time. For the condition number and robustness, we see the same result as for the datasets of 2000 points. Considering the subset sizes, the only difference is that the subsets obtained with OAS have become larger.

The above results also apply to the datasets of 10000 points. The only difference is that for 10000 points, it was not possible to fit a RBF model. A comparison with a model fitted to the complete dataset could therefore no longer be made. All the results can be found in Table 5.6 in Appendix 5.C.

5.4.6 Results for HSCT dataset of 2487 points

To make a comparison with the results of Srivastava et al. (2004) for the HSCT dataset, we have to use some different performance measures. Srivastava et al. (2004) measure the accuracy through the Root Mean Squared Percentage Error (RMSPE), the Average Percentage Error and the Maximum Percentage Error. We therefore report these accuracy measures for the different subsets instead of the Maximum Error and the RMSE.

Another difference is in the subset sizes as Srivastava et al. (2004) find subsets of 126, 283, and 372 points. To make a fair comparison, we compare our subsets of 250 points with Srivastava's subset of 283 points and our subsets of 350 points with Srivastava's subset of 372 points. The subsets of 500 points are thus used only for the comparison between the different methods in this chapter.

When comparing the results of Srivastava with our results, one of the surprising results is the difference in subset sizes when using the OAS method. Srivastava reports that using random orthogonal arrays of 250 and 686 points results in subsets of 126 and 283 points, respectively. However, when we use orthogonal arrays of these sizes, we obtain subsets of approximately 225 and 500 points. We tried to determine the cause of this large difference, but were not able to find a satisfying explanation. Use of different orthogonal arrays could be one explanation, but it seems unlikely that this alone causes such a large difference.

Examining the other performance measures for the subsets of 250 and 283 points, we see that the RMSPE and the Average Percentage Error obtained by Srivastava are equal to the results obtained with RAND. This is surprising as RAND is a very naive method. Furthermore, SS1, SS10, and SS1040 perform better on these two performance measures. For the Maximum Percentage Error, SS1, SS10, SS1040, MAXMIN, and OAS all give better results. Although OAS is also applied by Srivastava, these results are difficult to compare because of the difference in the resulting subset sizes.

For the subsets of 350 an 372 points, Srivastava obtained the best results for the RMSPE and Average Percentage Error. Of the other methods, SS1 and SS1040 also perform well on these measures. The Maximum Percentage Error is lowest for the SS1 method, followed by Srivastava's results and SS1040. Notice, that SS10 scores rather poorly on the Maximum Percentage Error and—to a lesser extend—on the RMSPE. This could indicate that indeed a cluster of points is selected, which negatively affects the overall accuracy.

As the other performance measures are not calculated by Srivastava, we can only compare them over the different methods we implemented ourselves. For the condition number and average robustness, SS1, SS1040, MAXMIN, DELETION, and OAS give good results. The high values for the condition number and the average robustness measure for the SS10 method are again a sign of possible clustering.

The subset selection times are again lowest for RAND and OAS, followed by MAXMIN and DELETION. The time needed to fit the model is lowest for SS1, SS10, and SS1040. This is the result of using the optimal parameters settings of previous iterations of the algorithm. If we would not have used this information and instead used a standard starting vector, then the time to fit the model would be comparable to those of the RAND method. All in all, the results are similar to our results for the three artificial datasets. Although we used some different performance measures for this dataset, the ranking of the different methods for most aspects is the same. This increases our confidence that results obtained by the artificial datsets can also be obtained for real life datasets.

5.5 Conclusions and further research

In this chapter, we show that fitting a Kriging model to a smaller but more uniform dataset can result in better Kriging models. Especially for large non-uniform datasets using a uniform subset can have several advantages; we can reduce the time necessary to fit the model, avoid numerical inaccuracies and improve the robustness with respect to errors in the output data.

To select a uniform subset, we consider several new and current methods. All these methods are tested on artificial subsets of three different sizes and two levels of uniformity. Furthermore, tests are performed on the HSCT dataset, which was also used by Srivastava et al. (2004). The tests show that by using uniform subsets, we can indeed find accurate Kriging models in less time. Furthermore, these Kriging models are more robust and less susceptible to numerical inaccuracies. When comparing the different methods for finding subsets, there is no overall winner. SS1040 generally performs well on accuracy, robustness, and numerical accuracy. Subsets obtained with SS1040 even result in Kriging models that are more accurate than RBF models fitted on the complete dataset. Compared with the other methods, SS1040, however, is relatively time consuming, but still considerably faster than fitting a Kriging model to the complete dataset. The OAS method is much faster, but has lower accuracy, robustness, and numerical accuracy. Deciding which method is best for a practical application, thus depends on how the different aspects are valued. The comparison made in this chapter can be used to help the user to make a good choice.

Further research could aim at finding new methods for selecting a subset. These new methods could, for instance, try to find better subsets by using a different objective to obtain a uniform subset or by taking into account other properties of the dataset. Another option would be to develop methods that dynamically determine a suitable subset size. Furthermore, it would be interesting to determine whether the results found in this chapter also apply to other modeling methods than Kriging.

5.A Results for artificial datasets of 2000 points

	Uniform Datasets									Non-Uniform Datasets								
	Average over datasets			Maximum over datasets			Minimum over datasets			Average over datasets			Maximum over datasets			Minimum over datasets		
	RMSE			RMSE			RMSE			RMSE			RMSE			RMSE		
Kriging	0.06			0.08			0.06			0.07			0.08			0.06		
RBF	0.10			0.11			0.10			0.11			0.12			0.10		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	0.17	0.15	0.13	0.21	0.18	0.16	0.14	0.13	0.11	0.18	0.15	0.13	0.22	0.18	0.17	0.15	0.13	0.11
SS1	0.10	0.08		0.11	0.10		0.09	0.08		0.10	0.09		0.11	0.10		0.09	0.08	
SS10	0.11	0.09	0.08	0.12	0.10	0.08	0.10	0.08	0.07	0.11	0.09	0.08	0.15	0.10	0.09	0.09	0.08	0.07
SS1040	0.10	0.09	0.08	0.19	0.10	0.08	0.09	0.08	0.07	0.10	0.09	0.08	0.12	0.10	0.09	0.09	0.08	0.07
MAXMIN	0.16	0.14	0.11	0.19	0.19	0.14	0.14	0.12	0.10	0.17	0.14	0.12	0.20	0.17	0.14	0.14	0.12	0.10
DELETION	0.16	0.14	0.12	0.20	0.16	0.14	0.14	0.12	0.10	0.17	0.14	0.12	0.20	0.18	0.14	0.15	0.13	0.10
FEX	0.17	0.15	0.12	0.21	0.19	0.15	0.14	0.13	0.11	0.17	0.15	0.13	0.20	0.17	0.14	0.15	0.13	0.11
OAS mean	0.17		0.11	0.19		0.12	0.17		0.11	0.17		0.11	0.18		0.12	0.16		0.11
OAS min	0.15		0.10	0.17		0.11	0.14		0.10	0.15		0.10	0.17		0.11	0.14		0.09
OAS max	0.20		0.13	0.24		0.14	0.18		0.11	0.19		0.13	0.21		0.14	0.17		0.12
	Maximum Error			Maximum Error			Maximum Error			Maximum Error			Maximum Error			Maximum Error		
Kriging	0.76			1.05			0.50			0.85			1.29			0.54		
RBF	1.31			1.58			0.96			1.36			1.68			1.09		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	1.62	1.41	1.28	2.05	1.98	1.74	1.15	0.85	0.86	1.65	1.54	1.42	2.02	1.96	1.81	1.12	0.99	0.84
SS1	0.96	0.94		1.48	1.43		0.56	0.53		1.04	1.02		1.53	1.47		0.71	0.67	
SS10	0.96	0.92	0.88	1.39	1.37	1.34	0.60	0.52	0.49	1.03	1.00	0.97	1.43	1.43	1.42	0.67	0.61	0.65
SS1040	0.94	0.90	0.87	1.30	1.35	1.34	0.63	0.54	0.48	1.00	0.98	0.96	1.53	1.45	1.41	0.71	0.67	0.60
MAXMIN	1.54	1.40	1.20	1.95	1.71	1.64	1.18	0.92	0.92	1.53	1.37	1.21	2.27	1.82	1.65	1.13	0.99	0.81
DELETION	1.53	1.34	1.22	1.97	1.77	1.56	1.11	0.93	0.83	1.55	1.41	1.22	2.00	1.89	1.70	1.02	0.98	0.82
FEX	1.56	1.44	1.28	2.05	1.87	1.74	1.12	0.88	0.85	1.57	1.46	1.34	1.99	1.85	1.82	1.03	0.99	0.86
OAS mean	1.60		1.18	1.72		1.40	1.39		1.01	1.59		1.25	1.74		1.47	1.47		1.02
OAS min	1.28		0.93	1.51		1.23	0.98		0.71	1.30		1.00	1.59		1.30	0.99		0.78
OAS max	1.92		1.45	2.22		1.72	1.67		1.20	1.88		1.50	2.09		1.85	1.75		1.29
	Time Subset Selection			Time Subset Selection			Time Subset Selection			Time Subset Selection			Time Subset Selection			Time Subset Selection		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SS1	9.56	24.15		10.18	24.91		9.39	23.67		9.52	24.05		10.12	24.89		9.35	23.63	
SS10	0.92	2.33	6.82	0.95	2.38	7.38	0.90	2.28	6.69	0.92	2.32	6.81	0.96	2.39	7.40	0.90	2.26	6.67
SS1040	0.92	2.32	6.79	0.95	2.77	7.24	0.86	2.26	6.64	0.92	2.32	6.80	0.95	2.37	7.39	0.89	2.25	6.67
MAXMIN	0.02	0.02	0.02	0.02	0.02	0.03	0.01	0.01	0.01	0.02	0.02	0.02	0.02	0.02	0.03	0.01	0.01	0.01
DELETION	0.07	0.07	0.06	0.08	0.07	0.07	0.06	0.06	0.06	0.07	0.07	0.07	0.08	0.08	0.07	0.06	0.06	0.06
FEX	0.21	0.28	0.33	0.37	0.47	0.70	0.08	0.09	0.11	0.24	0.30	0.39	0.35	0.42	0.57	0.15	0.17	0.20
OAS mean	0.00		0.01	0.00		0.01	0.00		0.01	0.00		0.01	0.00		0.01	0.00		0.01
OAS min	0.00		0.01	0.00		0.01	0.00		0.01	0.00		0.01	0.00		0.01	0.00		0.01
OAS max	0.00		0.01	0.01		0.01	0.00		0.01	0.00		0.01	0.00		0.01	0.00		0.01
	Time Model Fitting			Time Model Fitting			Time Model Fitting			Time Model Fitting			Time Model Fitting			Time Model Fitting		
Kriging	15.09			15.81			14.87			15.02			15.40			14.87		
RBF	0.25			0.26			0.25			0.25			0.26			0.25		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	0.10	0.21	0.51	0.11	0.23	0.53	0.09	0.20	0.48	0.10	0.21	0.51	0.11	0.23	0.53	0.10	0.20	0.48
SS1	0.08	0.16		0.09	0.17		0.07	0.14		0.08	0.16		0.15	0.17		0.08	0.15	
SS10	0.08	0.16	0.40	0.09	0.19	0.44	0.07	0.14	0.38	0.08	0.16	0.40	0.10	0.19	0.44	0.07	0.14	0.37
SS1040	0.08	0.16	0.40	0.09	0.17	0.44	0.07	0.15	0.37	0.08	0.16	0.41	0.10	0.17	0.71	0.07	0.14	0.36
MAXMIN	0.05	0.10	0.22	0.05	0.11	0.23	0.05	0.09	0.21	0.05	0.10	0.22	0.05	0.11	0.24	0.05	0.09	0.21
DELETION	0.05	0.10	0.22	0.05	0.10	0.23	0.05	0.09	0.21	0.05	0.10	0.22	0.05	0.11	0.23	0.04	0.09	0.21
FEX	0.10	0.21	0.51	0.11	0.25	0.53	0.10	0.21	0.48	0.10	0.21	0.51	0.11	0.23	0.54	0.09	0.20	0.48
OAS mean	0.08		0.62	0.08		0.70	0.08		0.59	0.08		0.59	0.08		0.62	0.08		0.57
OAS min	0.07		0.50	0.08		0.60	0.06		0.47	0.07		0.47	0.07		0.53	0.06		0.42
OAS max	0.08		0.71	0.09		0.85	0.08		0.66	0.08		0.68	0.10		0.72	0.08		0.64

Table 5.4: Average, minimum, and maximum of all performance measures over 50 artificial datasets of 2000 points. (*Table continued on next page*).

	Uniform Datasets							Non-Uniform Datasets						
	Average over datasets			Maximum over datasets			Minimum over datasets	Average over datasets			Maximum over datasets			Minimum over datasets
	Condition number			Condition number			Condition number	Condition number			Condition number			Condition number
Kriging	237859			935201			66964	420661			1409590			95175
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	1826	5367	8592	19835	47339	47927	355	495	651	1986	4194	11183	9879	17801	38230	314	1110	2384
SS1	288	579		679	1024		137	333		349	811		983	2539		188	403	
SS10	625	1119	1999	1869	3292	4792	259	387	778	748	1461	3012	2596	8534	16395	285	587	1390
SS1040	266	493	1100	1361	1014	1956	157	314	728	286	633	1645	667	1275	3744	155	326	977
MAXMIN	150	355	904	731	2062	3750	31	102	188	145	332	925	475	916	3062	28	100	237
DELETION	187	430	1167	457	1418	4019	42	101	172	235	405	998	656	1023	3514	68	112	214
FEX	919	2267	4898	2390	12665	14522	280	242	1199	922	2634	5159	3717	8739	27249	218	448	892
OAS mean	473		3991	1090		6902	273		2524	505		4222	958		9199	228		2072
OAS min	148		1351	271		2531	74		514	172		1419	299		2526	79		586
OAS max	1232		9668	4787		28866	434		4320	1275		9628	3344		28256	342		3972
	Maximum Robustness			Maximum Robustness			Maximum Robustness	Maximum Robustness			Maximum Robustness			Maximum Robustness
Kriging	45.73			107.39			22.90	73.79			259.69			35.74
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	7.27	10.12	12.60	21.34	29.26	56.40	3.53	4.03	4.01	9.98	12.43	18.97	24.40	26.62	42.52	3.91	4.40	7.23
SS1	1.75	1.95		2.33	2.54		1.41	1.50		1.85	2.14		2.76	3.44		1.45	1.72	
SS10	2.38	2.61	2.74	4.52	4.77	4.63	1.53	1.95	2.02	2.48	2.90	3.29	4.15	10.50	11.57	1.72	2.02	2.24
SS1040	1.72	1.93	2.20	2.49	2.67	2.86	1.38	1.65	1.68	1.85	2.08	2.50	2.76	3.29	3.52	1.39	1.65	1.96
MAXMIN	1.35	1.45	1.64	1.78	1.89	2.22	1.10	1.21	1.30	1.35	1.44	1.67	1.66	1.83	2.24	1.10	1.21	1.30
DELETION	1.43	1.56	1.76	1.84	2.00	2.34	1.12	1.25	1.37	1.51	1.58	1.74	1.96	1.98	2.28	1.25	1.24	1.38
FEX	5.25	6.47	8.59	13.72	13.19	16.36	2.51	3.24	4.53	5.33	8.19	9.66	14.61	30.67	24.34	2.86	4.19	3.81
OAS mean	2.18		3.13	2.59		3.68	1.92		2.64	2.25		3.26	2.85		3.98	1.91		2.71
OAS min	1.64		2.33	2.02		2.87	1.41		1.82	1.68		2.40	2.06		2.91	1.43		1.97
OAS max	2.96		4.37	4.01		7.11	2.23		2.98	3.21		4.74	5.38		8.11	2.25		2.99
	Average Robustness			Average Robustness			Average Robustness	Average Robustness			Average Robustness			Average Robustness
Kriging	6.33			10.62			4.52	8.48			11.92			6.08
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	1.11	1.57	1.94	2.29	5.01	3.97	0.60	0.73	0.78	1.19	1.55	2.19	2.33	3.38	4.32	0.55	0.81	1.19
SS1	0.67	0.76		0.74	0.81		0.61	0.70		0.69	0.78		0.74	0.83		0.65	0.74	
SS10	0.74	0.81	0.93	0.82	0.89	1.00	0.64	0.73	0.87	0.76	0.85	0.97	0.83	0.90	1.02	0.69	0.79	0.91
SS1040	0.71	0.79	0.90	0.84	0.86	0.94	0.64	0.71	0.83	0.73	0.80	0.92	0.81	0.85	0.97	0.68	0.75	0.89
MAXMIN	0.82	0.89	1.01	1.04	1.11	1.25	0.66	0.76	0.81	0.81	0.89	1.00	0.99	1.03	1.21	0.63	0.75	0.82
DELETION	0.84	0.93	1.05	1.00	1.12	1.28	0.64	0.73	0.78	0.87	0.92	1.03	1.05	1.09	1.28	0.70	0.77	0.82
FEX	0.98	1.28	1.70	1.47	2.94	2.53	0.60	0.56	1.00	0.98	1.39	1.65	1.79	2.35	3.98	0.52	0.57	0.79
OAS mean	0.94		1.27	1.05		1.38	0.88		1.17	0.94		1.27	1.03		1.44	0.83		1.14
OAS min	0.79		1.08	0.88		1.21	0.67		0.93	0.79		1.08	0.87		1.22	0.63		0.96
OAS max	1.12		1.48	1.49		1.78	0.98		1.31	1.11		1.47	1.34		1.69	0.94		1.29
	Real Subset Size			Real Subset Size			Real Subset Size	Real Subset Size			Real Subset Size			Real Subset Size
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
SS1	250	350		250	350		250	350		250	350		250	350		250	350	
SS10	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
SS1040	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
MAXMIN	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
DELETION	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
FEX	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
OAS mean	237		578	239		583	235		571	236		565	238		572	233		560
OAS min	227		533	233		552	219		517	225		518	231		542	217		494
OAS max	244		610	248		624	241		596	243		601	248		616	239		587

Table 5.4: Average, minimum, and maximum of all performance measures over 50 artificial datasets of 2000 points. (*Continued*).

5.B Results for artificial datasets of 5000 points

Uniform Datasets																	
	Average over datasets			Maximum over datasets			Minimum over datasets			Average over datasets			Maximum over datasets			Minimum over datasets		
	RMSE			RMSE			RMSE			RMSE			RMSE			RMSE		
RBF	0.07			0.08			0.07			0.08			0.08			0.07		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	0.17	0.14	0.12	0.19	0.17	0.15	0.14	0.12	0.11	0.18	0.15	0.13	0.22	0.18	0.16	0.15	0.13	0.11
SS1	0.10	0.08		0.11	0.08		0.09	0.07		0.10	0.08		0.11	0.09		0.09	0.07	
SS10	0.11	0.09	0.07	0.14	0.09	0.07	0.10	0.08	0.06	0.11	0.09	0.07	0.13	0.09	0.07	0.10	0.08	0.07
SS1040	0.10	0.08	0.07	0.19	0.09	0.07	0.09	0.08	0.06	0.10	0.08	0.07	0.11	0.09	0.07	0.09	0.08	0.06
MAXMIN	0.17	0.14	0.11	0.23	0.19	0.14	0.14	0.12	0.10	0.17	0.14	0.11	0.21	0.17	0.13	0.14	0.12	0.10
DELETION	0.17	0.14	0.12	0.19	0.17	0.15	0.14	0.12	0.10	0.17	0.14	0.12	0.20	0.18	0.15	0.14	0.12	0.10
FEX	0.17	0.15	0.12	0.20	0.18	0.15	0.15	0.13	0.10	0.17	0.15	0.13	0.23	0.18	0.15	0.14	0.13	0.11
OAS mean	0.17		0.11	0.18		0.11	0.16		0.10	0.17		0.11	0.18		0.11	0.16		0.10
OAS min	0.15		0.10	0.16		0.11	0.14		0.09	0.15		0.10	0.17		0.11	0.14		0.09
OAS max	0.19		0.12	0.22		0.14	0.17		0.11	0.19		0.12	0.21		0.14	0.17		0.11
	Maximum Error			Maximum Error			Maximum Error			Maximum Error			Maximum Error			Maximum Error		
RBF	1.06			1.30			0.79			1.09			1.33			0.86		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	1.59	1.45	1.33	1.96	1.90	1.79	1.14	1.05	0.95	1.63	1.48	1.36	2.12	1.97	1.92	1.07	1.01	0.98
SS1	0.76	0.74		1.04	1.09		0.51	0.52		0.81	0.77		1.18	1.13		0.51	0.48	
SS10	0.77	0.72	0.69	1.02	1.01	0.97	0.55	0.48	0.50	0.80	0.76	0.73	1.14	1.10	1.07	0.47	0.51	0.45
SS1040	0.74	0.70	0.68	1.31	0.97	0.90	0.53	0.50	0.45	0.78	0.75	0.73	1.21	1.13	1.04	0.54	0.47	0.49
MAXMIN	1.58	1.39	1.15	2.29	2.08	1.39	1.09	1.11	0.84	1.47	1.28	1.10	1.90	1.72	1.42	1.11	1.00	0.77
DELETION	1.56	1.41	1.17	2.21	2.19	1.84	1.14	0.87	0.84	1.59	1.40	1.19	2.13	1.88	1.75	1.15	1.06	0.81
FEX	1.61	1.50	1.29	1.95	2.01	1.76	1.23	0.92	0.74	1.59	1.47	1.33	1.92	1.93	1.70	1.07	1.00	0.96
OAS mean	1.59		1.15	1.75		1.36	1.48		0.97	1.58		1.14	1.68		1.36	1.42		1.01
OAS min	1.30		0.88	1.51		1.07	1.07		0.68	1.25		0.87	1.43		1.11	0.92		0.74
OAS max	1.92		1.46	2.24		1.78	1.70		1.24	1.87		1.45	2.19		1.70	1.60		1.21
	Time Subset Selection			Time Subset Selection			Time Subset Selection			Time Subset Selection			Time Subset Selection			Time Subset Selection		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SS1	14.37	30.66		14.78	31.38		14.10	29.91		14.32	31.01		14.74	31.68		13.93	30.60	
SS10	1.44	3.04	7.36	1.54	3.24	7.57	1.35	2.89	7.06	1.43	3.05	7.23	1.50	3.16	7.52	1.37	2.97	7.00
SS1040	1.39	2.97	7.18	1.56	3.14	7.36	1.34	2.85	6.98	1.43	3.05	7.35	1.57	3.26	7.74	0.88	1.87	4.60
MAXMIN	0.19	0.19	0.20	0.25	0.26	0.27	0.09	0.10	0.11	0.18	0.18	0.19	0.24	0.25	0.26	0.09	0.09	0.10
DELETION	0.73	0.72	0.70	0.94	0.93	0.91	0.51	0.50	0.48	0.72	0.71	0.69	0.89	0.88	0.87	0.39	0.38	0.37
FEX	0.45	0.52	0.72	0.62	0.72	1.11	0.30	0.38	0.44	0.57	0.77	1.06	0.72	1.03	1.47	0.38	0.53	0.77
OAS mean	0.01		0.03	0.01		0.03	0.01		0.03	0.01		0.03	0.01		0.03	0.01		0.03
OAS min	0.01		0.03	0.01		0.03	0.01		0.02	0.01		0.03	0.01		0.03	0.01		0.03
OAS max	0.01		0.03	0.01		0.04	0.01		0.03	0.01		0.03	0.01		0.04	0.01		0.03
	Time Model Fitting			Time Model Fitting			Time Model Fitting			Time Model Fitting			Time Model Fitting			Time Model Fitting		
RBF	1.86			2.14			1.79			1.84			2.08			1.77		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	0.10	0.21	0.51	0.11	0.22	0.53	0.09	0.19	0.49	0.10	0.21	0.51	0.11	0.22	0.53	0.09	0.20	0.48
SS1	0.06	0.13		0.08	0.14		0.05	0.12		0.06	0.13		0.07	0.15		0.06	0.08	
SS10	0.06	0.13	0.28	0.07	0.15	0.34	0.05	0.11	0.26	0.06	0.13	0.28	0.07	0.15	0.35	0.06	0.11	0.25
SS1040	0.06	0.12	0.27	0.06	0.14	0.29	0.05	0.11	0.26	0.06	0.13	0.29	0.09	0.16	0.35	0.04	0.08	0.19
MAXMIN	0.07	0.16	0.35	0.09	0.18	0.40	0.07	0.14	0.32	0.08	0.16	0.34	0.10	0.19	0.40	0.05	0.10	0.23
DELETION	0.08	0.16	0.35	0.10	0.20	0.40	0.06	0.14	0.32	0.08	0.16	0.34	0.10	0.19	0.40	0.05	0.10	0.22
FEX	0.10	0.21	0.51	0.11	0.23	0.54	0.10	0.20	0.49	0.10	0.21	0.52	0.11	0.22	0.54	0.10	0.20	0.50
OAS mean	0.09		0.92	0.10		0.93	0.09		0.90	0.09		0.89	0.10		0.91	0.09		0.87
OAS min	0.09		0.82	0.09		0.87	0.08		0.77	0.09		0.79	0.09		0.83	0.08		0.75
OAS max	0.10		1.00	0.10		1.05	0.10		0.96	0.10		0.97	0.10		1.02	0.09		0.92

Table 5.5: Average, minimum, and maximum of all performance measures over 50 artificial datasets of 5000 points. (*Table continued on next page*).

	Uniform Datasets						Non-Uniform Datasets					
	Average over datasets			Maximum over datasets			Minimum over datasets			Average over datasets			Maximum over datasets			Minimum over datasets		
	Condition number			Condition number			Condition number			Condition number			Condition number			Condition number		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	1516	3242	5873	7140	25241	26979	217	718	1724	2028	4115	10185	10351	11534	28277	343	783	1584
SS1	245	485		1023	1337		115	231		270	574		514	1275		127	285	
SS10	791	1117	1787	2338	3027	4562	214	436	826	1618	2251	3549	13515	15838	21532	274	583	978
SS1040	219	446	841	1048	2099	2102	129	260	543	238	445	1116	631	900	2906	148	293	601
MAXMIN	129	283	602	422	877	1676	31	55	186	126	267	725	489	1114	3277	32	67	190
DELETION	259	417	917	2732	2473	2730	36	96	242	170	361	951	447	1071	5724	50	106	309
FEX	792	2161	4006	3857	15572	12351	129	380	582	1476	2292	5400	11319	10979	20195	187	239	812
OAS mean	462		4156	1689		6510	269		2497	497		4590	848		13719	266		2599
OAS min	144		1588	265		2714	59		637	163		1566	284		3007	71		710
OAS max	1194		9764	7357		23807	488		3804	1326		13113	3973		93526	566		4254
	Maximum Robustness			Maximum Robustness			Maximum Robustness			Maximum Robustness			Maximum Robustness			Maximum Robustness		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	5.96	8.39	10.54	15.50	21.88	34.12	2.03	2.13	2.97	8.51	11.28	15.67	30.62	35.52	61.64	2.33	3.03	3.20
SS1	1.64	1.78		2.52	2.42		1.31	1.50		1.70	1.93		2.23	2.78		1.32	1.58	
SS10	2.63	2.71	2.80	5.07	5.15	5.09	1.77	1.76	1.97	3.15	3.36	3.63	9.76	10.48	10.41	1.64	2.02	2.25
SS1040	1.71	1.85	2.06	3.46	2.74	2.83	1.33	1.51	1.65	1.69	1.91	2.27	2.26	2.95	3.47	1.41	1.55	1.76
MAXMIN	1.33	1.41	1.52	1.60	1.74	1.92	1.07	1.19	1.28	1.32	1.42	1.57	1.61	1.85	2.02	1.09	1.23	1.30
DELETION	1.45	1.54	1.72	2.26	2.32	2.31	1.13	1.26	1.36	1.45	1.53	1.72	1.94	1.95	2.60	1.16	1.24	1.50
FEX	3.81	4.95	6.75	7.82	15.33	18.55	1.55	2.13	2.10	5.47	5.77	7.59	20.37	11.10	15.56	1.97	2.16	2.25
OAS mean	2.15		3.14	2.78		3.58	1.84		2.75	2.22		3.21	2.80		3.97	1.92		2.79
OAS min	1.62		2.39	2.05		2.65	1.32		1.71	1.65		2.41	1.99		2.82	1.34		2.00
OAS max	3.15		4.48	6.98		7.80	2.28		3.35	3.17		4.55	4.74		7.40	2.22		3.40
	Average Robustness			Average Robustness			Average Robustness			Average Robustness			Average Robustness			Average Robustness		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	1.11	1.35	1.66	2.35	2.63	3.35	0.51	0.91	1.09	1.22	1.56	1.98	2.37	2.50	3.46	0.69	0.87	1.09
SS1	0.63	0.70		0.68	0.73		0.59	0.66		0.64	0.71		0.68	0.74		0.60	0.68	
SS10	0.74	0.79	0.88	0.89	0.86	0.94	0.67	0.72	0.83	0.75	0.82	0.91	0.94	0.91	1.03	0.67	0.74	0.83
SS1040	0.67	0.72	0.80	0.71	0.87	0.86	0.58	0.68	0.77	0.67	0.73	0.82	0.77	0.78	0.87	0.64	0.68	0.78
MAXMIN	0.81	0.87	0.95	0.94	1.02	1.08	0.67	0.71	0.83	0.80	0.87	0.97	0.95	1.09	1.19	0.67	0.73	0.83
DELETION	0.84	0.92	1.03	1.24	1.23	1.24	0.66	0.76	0.87	0.83	0.91	1.03	0.99	1.11	1.34	0.67	0.76	0.90
FEX	0.91	1.19	1.48	1.46	2.93	2.33	0.54	0.71	0.87	1.12	1.23	1.64	3.22	2.02	2.81	0.52	0.55	0.90
OAS mean	0.93		1.28	1.04		1.37	0.88		1.21	0.94		1.29	1.02		1.41	0.88		1.20
OAS min	0.78		1.12	0.89		1.26	0.62		0.95	0.80		1.11	0.88		1.25	0.67		1.01
OAS max	1.10		1.46	1.41		1.76	0.96		1.30	1.11		1.51	1.36		1.75	0.97		1.30
	Real Subset Size			Real Subset Size			Real Subset Size			Real Subset Size			Real Subset Size			Real Subset Size		
Subset size	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
RAND	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
SS1	250	350		250	350		250	350		250	350		250	350		250	350	
SS10	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
SS1040	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
MAXMIN	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
DELETION	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
FEX	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
OAS mean	245		642	247		646	244		640	244		634	246		639	243		628
OAS min	240		614	243		627	236		600	238		604	243		617	233		587
OAS max	249		664	250		672	247		653	249		657	250		669	247		646

Table 5.5: Average, minimum, and maximum of all performance measures over 50 artificial datasets of 5000 points. (*Continued*).

5.C Results for artificial datasets of 10000 points

Uniform Datasets

	Average over datasets RMSE			Maximum over datasets RMSE			Minimum over datasets RMSE		
Subset size	250	350	500	250	350	500	250	350	500
RAND	0.17	0.15	0.13	0.21	0.18	0.15	0.15	0.13	0.11
SS1	0.10	0.08		0.11	0.08		0.09	0.07	
SS10	0.12	0.09	0.07	0.14	0.10	0.07	0.10	0.08	0.06
SS1040	0.10	0.08	0.06	0.21	0.10	0.07	0.09	0.07	0.06
MAXMIN	0.17	0.14	0.11	0.20	0.17	0.15	0.14	0.12	0.10
DELETION	0.17	0.14	0.12	0.20	0.17	0.15	0.14	0.12	0.10
FEX	0.17	0.14	0.13	0.20	0.16	0.14	0.14	0.13	0.11
OAS mean	0.17		0.10	0.18		0.11	0.16		0.10
OAS min	0.15		0.09	0.17		0.10	0.14		0.09
OAS max	0.19		0.12	0.21		0.13	0.18		0.11

	Maximum Error			Maximum Error			Maximum Error		
Subset size	250	350	500	250	350	500	250	350	500
RAND	1.56	1.48	1.35	2.12	2.01	1.95	1.16	0.94	0.83
SS1	0.59	0.54		0.90	0.88		0.45	0.38	
SS10	0.64	0.56	0.50	0.87	0.84	0.73	0.49	0.37	0.36
SS1040	0.63	0.54	0.50	1.48	0.78	0.71	0.45	0.38	0.38
MAXMIN	1.61	1.39	1.11	2.37	1.92	1.79	1.12	1.06	0.78
DELETION	1.59	1.42	1.23	2.04	1.84	1.60	1.12	0.88	0.82
FEX	1.60	1.40	1.32	1.99	1.87	1.86	1.12	1.01	0.91
OAS mean	1.60		1.11	1.75		1.25	1.45		0.96
OAS min	1.29		0.85	1.54		1.08	1.10		0.73
OAS max	1.93		1.42	2.33		1.70	1.71		1.15

	Time Subset Selection			Time Subset Selection			Time Subset Selection		
Subset size	250	350	500	250	350	500	250	350	500
RAND	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SS1	24.17	52.39		27.94	55.15		23.59	50.23	
SS10	2.39	5.17	11.67	2.74	5.51	12.33	2.30	4.87	11.21
SS1040	2.36	5.10	11.57	2.73	5.53	12.01	2.29	4.83	11.18
MAXMIN	0.78	0.79	0.82	0.87	0.89	0.92	0.55	0.57	0.59
DELETION	3.55	3.53	3.49	3.68	3.65	3.60	3.08	3.06	3.02
FEX	0.83	0.95	1.18	1.16	1.40	1.92	0.59	0.74	0.93
OAS mean	0.02		0.05	0.02		0.05	0.02		0.05
OAS min	0.02		0.05	0.02		0.05	0.01		0.04
OAS max	0.02		0.05	0.03		0.07	0.02		0.05

	Time Model Fitting			Time Model Fitting			Time Model Fitting		
Subset size	250	350	500	250	350	500	250	350	500
RAND	0.10	0.20	0.50	0.10	0.22	0.52	0.09	0.19	0.48
SS1	0.08	0.17		0.09	0.18		0.07	0.15	
SS10	0.08	0.17	0.40	0.09	0.19	0.46	0.07	0.15	0.37
SS1040	0.08	0.17	0.41	0.09	0.18	0.45	0.07	0.14	0.37
MAXMIN	0.10	0.21	0.50	0.11	0.22	0.52	0.09	0.19	0.46
DELETION	0.10	0.21	0.50	0.11	0.23	0.53	0.09	0.19	0.48
FEX	0.10	0.21	0.50	0.10	0.26	0.55	0.09	0.20	0.48
OAS mean	0.09		0.98	0.10		1.00	0.09		0.96
OAS min	0.09		0.92	0.09		0.96	0.08		0.85
OAS max	0.10		1.03	0.11		1.09	0.10		0.99

Non-Uniform Datasets

	Average over datasets RMSE			Maximum over datasets RMSE			Minimum over datasets RMSE		
Subset size	250	350	500	250	350	500	250	350	500
RAND	0.17	0.15	0.13	0.22	0.20	0.16	0.14	0.13	0.10
SS1	0.10	0.08		0.21	0.09		0.09	0.07	
SS10	0.11	0.09	0.07	0.13	0.10	0.07	0.09	0.08	0.06
SS1040	0.10	0.08	0.06	0.11	0.09	0.07	0.09	0.07	0.06
MAXMIN	0.17	0.14	0.11	0.21	0.18	0.14	0.14	0.12	0.10
DELETION	0.17	0.14	0.12	0.20	0.17	0.14	0.15	0.12	0.10
FEX	0.17	0.14	0.13	0.22	0.19	0.15	0.15	0.13	0.11
OAS mean	0.17		0.10	0.18		0.11	0.16		0.10
OAS min	0.15		0.09	0.16		0.10	0.14		0.09
OAS max	0.19		0.12	0.21		0.13	0.17		0.11

	Maximum Error			Maximum Error			Maximum Error		
Subset size	250	350	500	250	350	500	250	350	500
RAND	1.55	1.52	1.32	1.92	2.17	1.78	1.06	0.95	0.89
SS1	0.69	0.62		1.46	0.84		0.45	0.44	
SS10	0.69	0.62	0.58	0.97	0.98	0.94	0.51	0.41	0.37
SS1040	0.65	0.62	0.57	0.99	0.99	0.85	0.49	0.43	0.38
MAXMIN	1.58	1.34	1.11	1.95	1.84	1.50	1.10	0.76	0.76
DELETION	1.58	1.39	1.10	2.03	1.99	1.47	1.21	1.01	0.69
FEX	1.64	1.45	1.38	1.92	1.86	1.74	1.25	1.00	0.87
OAS mean	1.58		1.13	1.75		1.24	1.40		1.00
OAS min	1.27		0.85	1.48		1.10	1.00		0.65
OAS max	1.89		1.42	2.15		1.72	1.63		1.25

	Time Subset Selection			Time Subset Selection			Time Subset Selection		
Subset size	250	350	500	250	350	500	250	350	500
RAND	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SS1	24.21	52.33		25.68	55.15		22.94	49.67	
SS10	2.38	5.16	11.69	2.78	5.76	12.30	2.32	4.89	11.24
SS1040	2.37	5.15	11.60	2.73	5.66	12.26	2.31	4.82	11.19
MAXMIN	0.79	0.81	0.84	1.16	1.18	1.20	0.66	0.67	0.70
DELETION	3.62	3.59	3.55	4.02	3.99	3.96	3.27	3.24	3.20
FEX	1.10	1.41	1.93	1.36	1.77	2.40	0.86	0.97	1.61
OAS mean	0.02		0.05	0.02		0.05	0.02		0.05
OAS min	0.02		0.05	0.02		0.05	0.02		0.04
OAS max	0.02		0.05	0.03		0.06	0.02		0.05

	Time Model Fitting			Time Model Fitting			Time Model Fitting		
Subset size	250	350	500	250	350	500	250	350	500
RAND	0.10	0.20	0.50	0.11	0.22	0.52	0.09	0.19	0.48
SS1	0.08	0.17		0.09	0.18		0.08	0.16	
SS10	0.08	0.17	0.40	0.09	0.19	0.50	0.07	0.15	0.36
SS1040	0.08	0.17	0.41	0.09	0.19	0.46	0.07	0.15	0.37
MAXMIN	0.10	0.21	0.50	0.11	0.22	0.87	0.09	0.19	0.48
DELETION	0.10	0.21	0.50	0.11	0.23	0.54	0.09	0.20	0.47
FEX	0.10	0.21	0.50	0.11	0.22	0.54	0.09	0.20	0.48
OAS mean	0.09		0.98	0.10		0.97	0.09		0.94
OAS min	0.09		0.92	0.09		0.93	0.08		0.84
OAS max	0.10		1.03	0.11		1.05	0.10		0.97

Table 5.6: Average, minimum, and maximum of all performance measures over 50 artificial datasets of 10000 points. (*Table continued on next page*).

	Uniform Datasets							Non-Uniform Datasets						
	Average over datasets			Maximum over datasets		Minimum over datasets		Average over datasets			Maximum over datasets		Minimum over datasets	
	Condition number			Condition number		Condition number		Condition number			Condition number		Condition number	
Subset size	250	350	500	250	350	250	350	250	350	500	250	350	500	250	350	500
RAND	1553	2824	6239	6462	12486	31343	343	473	1179	2083	5611	11945	8990	61456	52456	459	1165	2194
SS1	217	432		612	925		115	261		341	590		4518	4283		134	244	
SS10	1153	1546	2509	4013	3682	9362	293	596	1095	1605	2139	3153	6111	9442	11228	574	831	1045
SS1040	214	463	762	568	5698	6318	116	206	476	212	423	849	314	798	1883	132	268	445
MAXMIN	109	221	685	302	855	2031	31	78	167	121	273	607	283	940	2142	32	72	150
DELETION	183	356	813	693	1378	2289	36	131	231	186	359	961	638	1624	4593	56	59	248
FEX	1012	1886	5069	5271	5548	23544	234	533	732	1209	2937	4149	8944	14051	18497	226	469	1230
OAS mean	448		4296	891		6921	236		2990	447		4487	909		8439	257		2831
OAS min	141		1435	232		2653	55		614	145		1581	322		2517	83		599
OAS max	1226		10107	5348		33058	408		5288	1172		11156	3812		44761	435		4509
	Maximum Robustness			Maximum Robustness		Maximum Robustness		Maximum Robustness			Maximum Robustness		Maximum Robustness			Maximum Robustness		
Subset size	250	350	500	250	350	250	350	250	350	500	250	350	500	250	350	500	250	350	500
RAND	7.39	9.09	11.91	22.77	22.48	24.26	3.66	4.25	4.12	9.88	14.76	19.66	30.46	37.12	56.86	4.41	6.41	8.52
SS1	1.64	1.77		2.54	2.56		1.31	1.50		1.70	1.93		3.87	4.41		1.35	1.48	
SS10	3.05	3.24	3.41	7.21	6.96	6.60	1.91	1.99	2.10	3.39	3.37	3.50	8.03	5.55	5.95	2.00	2.36	2.49
SS1040	1.67	1.79	1.90	2.28	3.43	3.50	1.33	1.50	1.61	1.67	1.85	1.99	2.11	2.60	2.90	1.35	1.52	1.59
MAXMIN	1.35	1.36	1.53	1.77	1.62	1.92	1.08	1.19	1.30	1.32	1.40	1.50	1.68	1.71	2.01	1.06	1.16	1.23
DELETION	1.41	1.53	1.68	1.83	1.80	1.97	1.12	1.27	1.36	1.45	1.53	1.71	2.05	1.98	2.45	1.15	1.26	1.36
FEX	5.23	5.65	8.26	17.93	10.25	19.53	2.69	3.58	3.49	5.31	7.48	8.71	12.84	14.01	20.42	2.29	3.50	4.93
OAS mean	2.15		3.19	2.75		3.79	1.80		2.77	2.14		3.25	2.70		4.41	1.78		2.89
OAS min	1.61		2.39	1.91		2.77	1.37		1.97	1.64		2.39	1.88		2.86	1.45		1.84
OAS max	3.14		4.53	6.02		8.86	2.01		3.42	3.06		4.72	8.58		10.42	2.08		3.26
	Average Robustness			Average Robustness		Average Robustness		Average Robustness			Average Robustness		Average Robustness			Average Robustness		
Subset size	250	350	500	250	350	250	350	250	350	500	250	350	500	250	350	500	250	350	500
RAND	1.13	1.37	1.76	2.61	2.23	3.01	0.69	0.60	1.03	1.25	1.72	2.24	2.38	5.04	4.15	0.60	1.06	1.11
SS1	0.60	0.66		0.65	0.69		0.57	0.64		0.61	0.67		0.75	0.90		0.57	0.64	
SS10	0.76	0.81	0.88	1.01	0.94	1.01	0.67	0.72	0.82	0.78	0.83	0.91	0.87	0.90	0.99	0.70	0.76	0.86
SS1040	0.64	0.69	0.76	0.73	0.84	0.83	0.50	0.65	0.72	0.64	0.69	0.77	0.68	0.74	0.80	0.60	0.65	0.73
MAXMIN	0.80	0.86	0.96	0.92	1.00	1.10	0.68	0.77	0.84	0.80	0.88	0.95	0.91	1.02	1.13	0.67	0.74	0.82
DELETION	0.84	0.91	1.01	1.04	1.06	1.17	0.63	0.78	0.86	0.84	0.89	1.02	1.04	1.13	1.30	0.69	0.68	0.85
FEX	1.02	1.25	1.67	2.05	2.07	3.49	0.59	0.82	0.90	1.06	1.42	1.63	2.45	3.04	3.04	0.56	0.72	1.01
OAS mean	0.92		1.29	0.99		1.35	0.84		1.23	0.93		1.29	1.00		1.36	0.86		1.20
OAS min	0.78		1.11	0.87		1.23	0.67		1.00	0.79		1.11	0.87		1.25	0.71		0.95
OAS max	1.08		1.46	1.31		1.68	0.98		1.34	1.09		1.49	1.25		1.84	0.90		1.32
	Real Subset Size			Real Subset Size		Real Subset Size		Real Subset Size			Real Subset Size		Real Subset Size			Real Subset Size		
Subset size	250	350	500	250	350	250	350	250	350	500	250	350	500	250	350	500	250	350	500
RAND	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
SS1	250	350		250	350		250	350		250	350		250	350		250	350	
SS10	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
SS1040	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
MAXMIN	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
DELETION	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
FEX	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500	250	350	500
OAS mean	248		666	249		669	247		663	247		660	248		663	246		656
OAS min	245		650	247		660	242		636	244		640	246		652	241		629
OAS max	250		678	250		683	249		670	250		675	250		680	248		668

Table 5.6: Average, minimum, and maximum of all performance measures over 50 artificial datasets of 10000 points. (*Continued*).

5.D Results for HSCT dataset of 2487 points

	Results Srivastava	
Subset size	283	372
RMSPE	1.00	0.24
Avg. Perc. Err.	0.59	0.17
Max. Perc. Err.	6.12	2.02

	HSCT Dataset					
	RMSPE			Condition number		
Subset size	250	350	500	250	350	500
RAND	1.00	0.93	0.86	712731778	341942761	3072209097
OAS mean	1.30		1.11	112944986		1047914507
OAS min	1.05		0.86	2213		3344
OAS max	1.50		1.38	447474975		2377979099
FEX	1.72	1.73	1.73	2035776984	2328684961	1600933318
MAXMIN	1.31	1.14	1.22	68976	69945	55685
DELETION	1.70	1.83	1.46	29772	22407	69130
SS1	0.66	0.65		69777	197848	
SS10	0.60	1.28	0.68	1568232546	7678597938	9457605211
SS1040	0.58	0.44	0.36	162191	183002	108242
	Average Percentage Error			Average Robustness		
Subset size	250	350	500	250	350	500
RAND	0.59	0.52	0.52	43.08	15.11	27.80
OAS mean	0.85		0.69	19.06		63.38
OAS min	0.73		0.57	0.87		0.75
OAS max	0.95		0.81	62.81		117.07
FEX	1.19	1.17	1.21	177.70	182.17	142.40
MAXMIN	0.92	0.81	0.91	0.84	0.85	0.77
DELETION	1.22	1.33	1.08	1.61	1.66	1.06
SS1	0.53	0.51		1.50	1.35	
SS10	0.46	0.76	0.53	126.76	85.95	106.68
SS1040	0.44	0.35	0.27	1.17	1.03	0.91
	Maximum Percentage Error			Real Subset Size		
Subset size	250	350	500	250	350	500
RAND	9.37	8.96	7.65	250	350	500
OAS mean	5.77		5.67	226		505
OAS min	4.75		4.97	224		494
OAS max	7.11		6.82	230		515
FEX	8.59	9.19	8.45	250	350	500
MAXMIN	5.60	5.44	5.32	250	350	500
DELETION	7.55	7.76	6.51	250	350	500
SS1	2.02	2.34		250	350	
SS10	2.37	13.73	2.13	250	350	500
SS1040	2.36	1.76	2.50	250	350	500
	Time Subset Selection			Time Model Fitting		
Subset size	250	350	500	250	350	500
RAND	0.00	0.00	0.00	0.41	0.84	1.92
OAS mean	0.01		0.04	0.42		2.40
OAS min	0.01		0.04	0.23		2.23
OAS max	0.01		0.05	0.78		2.55
FEX	7.29	7.81	7.97	0.41	0.85	1.93
MAXMIN	0.12	0.12	0.13	0.38	0.56	1.21
DELETION	0.47	0.45	0.42	0.34	0.72	1.63
SS1	21.30	57.15		0.27	0.45	
SS10	2.64	6.08	16.72	0.23	0.43	0.97
SS1040	2.31	5.74	17.12	0.23	0.41	0.98

Table 5.7: Results for HSCT-dataset of 2487 points

5.E Kriging model

As we focus on Kriging, we summarize some theory according to Sacks et al. (1989b). In Kriging, the output data $y(x)$ is treated as a realization of a random function $Y(x)$. This random function is divided into a regression part and a stochastic part:

$$Y(x) = \sum_{j=0}^{k} \beta_j f_j(x) + Z(x),$$

where $k+1$ is the number of regression functions including $f_0(x) = 1$. In this chapter for the regression part, we use the linear functions $f_j(x) = x_j$ for $j = 1, \ldots, k$ where k is the number of dimensions. The stochastic part $Z(x)$ is assumed to be a gaussian stationary process with zero mean and a covariance between $Z(x)$ and $Z(w)$ of the form

$$V(w, x) = \sigma^2 R(w, x),$$

where σ^2 is the constant process variance and $R(w, x)$ is the correlation between $Z(x)$ and $Z(w)$. To fit the Kriging model, we use a dataset with input data $X = [x^1, \ldots, x^n]$ and corresponding output data $y_X = [y(x^1), \ldots, y(x^n)]$. In Kriging, the vector y_X is assumed to be a realization of the stochastic vector $[Y(x^1), \ldots, Y(x^n)]$.

To predict the output value at a new point x, Kriging uses the Best Linear Unbiased Predictor (BLUP). This means that $\hat{y}(x)$, the predicted output value at point x, is given by

$$\hat{y}(x) = c^\top(x) y_X,$$

where the Kriging weights $c(x)$ are determined such that they minimize

$$MSE(\hat{y}(x)) = E(c^\top(x) Y_X - Y(x))^2 \tag{5.1}$$

under the unbiasedness constraint

$$E(c^\top(x) Y_X) = E(Y(x)). \tag{5.2}$$

Let us now introduce the following notation:

$$\begin{aligned}
f(x) &= [f_0(x), f_1(x), \ldots, f_k(x)] = [1, x_1, \ldots, x_d]^\top \\
F &= \begin{bmatrix} f^\top(x^1) \\ \vdots \\ f^\top(x^n) \end{bmatrix} \\
r(x) &= [R(x, x^1), \ldots, R(x, x^n)]^\top \\
R &= \begin{bmatrix} r^\top(x^1) \\ \vdots \\ r^\top(x^n) \end{bmatrix} = \begin{bmatrix} R(x^1, x^1) & R(x^1, x^2) & \ldots & R(x^1, x^n) \\ R(x^2, x^1) & R(x^2, x^2) & \ldots & R(x^2, x^n) \\ \vdots & \vdots & \ddots & \vdots \\ R(x^n, x^1) & R(x^n, x^2) & \ldots & R(x^n, x^n) \end{bmatrix}.
\end{aligned}$$

The matrix R is thus the correlation matrix containing the correlations between $Z(x^i)$ and $Z(x^j)$ for all $i, j \in \{1, \ldots, n\}$, so R is positive semi-definite and symmetric with ones on the diagonal.

Classical Kriging assumes that the Kriging weights $c(x)$ are independent of the output data. Therefore, we can rewrite the MSE in (5.1) as (Santner et al. 2003)

$$MSE(\hat{y}(x)) = \sigma^2(1 + c^\top(x)Rc(x) - 2c^\top(x)r(x)) \tag{5.3}$$

and the constraint in (5.2) as

$$F^\top c(x) = f(x).$$

Using Lagrange multipliers $\lambda(x)$, the MSE in (5.3) can be minimized by solving the following system of equations:

$$\begin{bmatrix} 0 & F^\top \\ F & R \end{bmatrix} \begin{bmatrix} \lambda(x) \\ c(x) \end{bmatrix} = \begin{bmatrix} f(x) \\ r(x) \end{bmatrix}.$$

Solving this system gives the following expressions for $\lambda(x)$ and $c(x)$:

$$\begin{aligned} \lambda(x) &= (F^\top R^{-1} F)^{-1}(F^\top R^{-1} r(x) - f(x)) \\ c(x) &= R^{-1}(r(x) - F\lambda(x)). \end{aligned}$$

We see that determining the prediction of the output at point x requires solving a linear system containing the $n \times n$ matrix R. If the size of the dataset, n, becomes very large this can be rather time consuming and we can thus save time by using less data.

Thus far, we have not specified the form of the correlation function $R(x, w)$. For the correlation function, there are a number of alternatives. We choose to use the Gaussian correlation function:

$$R^\theta(w, x) = \prod_{j=1}^{d} \exp\left(-\theta_j |w_j - x_j|^2\right)$$

as this is the most frequently used correlation function for Kriging (Jin et al. 2001).

We now still have to determine β, σ, and θ such that the Kriging model interpolates the training data. For this, we use the Maximum Likelihood Estimator (MLE). For β, this gives the generalized least-squares estimate:

$$\hat{\beta} = (F^\top R^{-1} F)^{-1} F^\top R^{-1} y_X.$$

The MLE of σ^2 is given by:

$$\hat{\sigma}^2 = \frac{1}{n}(y_X - F\hat{\beta})^\top R^{-1}(y_X - F\hat{\beta}).$$

To determine the MLE of θ, we must solve the following minimization problem (Sacks et al. 1989b):

$$\min_{\theta} |R|^{1/n} \hat{\sigma}^2.$$

Unfortunately, we do not have an analytic expression for the $\hat{\theta}$ that solves this problem. We thus need some numerical optimization procedure to determine $\hat{\theta}$. As mentioned in Section 5.4, we use the DACE toolbox of Lophaven et al. (2002). Notice that this minimization problem also contains R^{-1} as it is part of the expression for $\hat{\theta}$. The numerical optimization procedure has to determine R for multiple values of θ in order to find the solution to the minimization problem. This can become very time-consuming if R is very large. As R is an $n \times n$ matrix, the size of the training set n directly affects the time-consumption of this step.

5.F Radial basis functions

Besides Kriging models, we can also use radial basis function (RBF) models to construct a model. RBF models have been developed by Hardy (1971) to interpolate a training set of multi-dimensional data. To approximate the output value at a point x, RBF uses the distances between this point and the training points. Each distance value is used as the input of a radially symmetric function. A linear combination of the output values of this function forms the approximation of the output value in x. The simple RBF model used in this chapter to approximate y is

$$\hat{y} = \sum_i \beta_i ||x - x_i||,$$

where $||x - x_i||$ denotes the Euclidean distance between the training point x_i and the approximation point x. When we fill in all training points (x_i, y_i) for $i = 1, \ldots, n$, we obtain a set of linear equations. By solving this system, we can determine the coefficients β_i.

The advantage of RBF models is that they have shown good fits to both stochastic and deterministic functions (Powell 1987) and that fitting them requires much less time than Kriging models (Jin et al. 2001). Therefore, they are more suitable for larger datasets. However, by combining subset selection and Kriging, we aim to make Kriging equally, or even more, suitable for large datasets. We therefore compare Kriging models fitted on a subset with an RBF fitted on the complete dataset. As some performance measures cannot be calculated for radial basis functions, we only compare the accuracy and the time required to fit the model. For the Kriging models, this time also includes the time necessary to select the subset.

Part III

Complexity control in symbolic regression

Chapter 6

Metamodeling by symbolic regression and Pareto simulated annealing

> Essentially, all models are wrong,
> but some are useful.
>
> (George E.P. Box)

6.1 Introduction

In many scientific areas, it is important to relate output of a system to its input. Getting insight into the sensitivities of outputs with respect to inputs or finding the best input values with respect to the outputs may require a large number of system evaluations. In many applications, the number of evaluations that can be used to do this is limited. Therefore, metamodels (also called approximating models, compact models or response-surface models) are used to approximate the behavior of the system.

A typical example of such a process can be found in virtual prototyping tools, such as finite element analysis (FEA) software. These tools are used to predict physical properties of a product. Such simulations often require much computation time. A simulation run that takes hours is no exception, so the number of experiments may be very limited. Parameters that are input to the simulation software are referred to as design parameters. Parameters that quantify the simulated physical behavior are called response parameters. In simulation-based optimization, values for the design parameters are searched, such that the response parameters are in some sense optimal; see, e.g., Barthelemy and Haftka (1993), Alexandrov et al. (1998), Jones et al. (1998), and Simpson et al. (2004).

In literature, many types of metamodels are used. The most obvious choice is the first-degree or second-degree polynomial model (see Montgomery (2009)). More complicated

model structures include rational functions (Cuyt and Verdonk (1992)), Kriging models (Sacks et al. (1989b)), support vector regression machines (Vapnik et al. (1997)), neural network structures like RBFs (Powell (1987)), and symbolic regression (Koza (1992)). Comparative studies on metamodel types can be found in Jin et al. (2001, 2003).

Symbolic regression searches for the best metamodel by systematically altering operators in a set of explicit formulas. These formulas are represented using a tree structure. The best tree structure is found through a combinatoric optimization technique, An important advantage of symbolic regression is its flexibility. A metamodel that is found by symbolic regression is not restricted to a certain class of functions. Symbolic regression models can be used for extrapolation, because they capture the underlying physics, which is an advantage in comparison to, e.g., Kriging models. Furthermore, symbolic regression models are usually better interpretable, which makes it easier to extract new knowledge from symbolic regression models. To the best of our knowledge, symbolic regression models are always used in combination with genetic programming (e.g., see Koza (1992), and, for an example in engineering, Gambling et al. (2001)). This chapter introduces a symbolic regression approach that is not based on genetic programming, but on simulated annealing (see Aarts and Korst (1989)). This optimization technique usually requires fewer iterations than genetic algorithms to converge to an optimal solution. The algorithm is extended with a number of new concepts. Complexity control is used to ensure interpretability of the resulting model. Pareto simulated annealing is used to find not only one best model, but a range of models on a best fit/complexity trade-off curve (see also Smits and Kotanchek (2004)). The best-fit metamodel may not be the metamodel that eventually will be chosen, because it may be less intuitive than metamodels that have a worse fit. Further, linear regression (see Montgomery (2009)) is used to fit coefficients in the formulas. Finding the best coefficient values is often recognized as a difficult problem in symbolic regression. A binary tree data structure is used for fast neighborhood exploration. The resulting approximating model is compared with Kriging models and genetic-programming-based symbolic regression through a number of typical cases from simulation-based optimization.

The remainder of this chapter is organized as follows. In Section 6.2, we describe the basic approach. Section 6.3 describes a number of extensions to the basic algorithm. In Section 6.4, we test our approach on several cases and compare the results with other metamodel types. In Section 6.5, we draw conclusions.

6.2 Symbolic regression approach

Symbolic regression is a technique for finding the best model in a very large class of candidates. The candidates are explicit symbolic formulas. In this section, we first

describe the set of all possible approximating models. After that, we describe how to find the best approximating model. The approach assumes that all simulation data are already generated.

6.2.1 Model structure

The set of functions in which we search for the best metamodel consists of all functions that can be described as a linear combination of transformation functions. Each transformation function defines a transformation of the original parameters into a transformed parameter, using operators such as addition, subtraction, multiplication, division, exponents, sines, and cosines. Thus, an approximating model can be written as:

$$f(x_1, \ldots, x_n) = \sum_i a_i g_i(x_1, \ldots, x_n), \tag{6.1}$$

where x_i is the i^{th} input parameter and a_i is the multiplier of transformation function $g_i(x)$. Suppose for example, we have a problem with three inputs, x_1, x_2, and x_3, with the following approximating model:

$$f(x_1, x_2, x_3) = 0.341 + 1.231 \sin\left(x_1 + \frac{x_2}{x_3}\right) - 2.114 \exp(x_3). \tag{6.2}$$

Once we know the transformation functions, the coefficients a_i can be calculated using linear regression. The problem remains to find suitable transformation functions g_i that lead to a model that fits the data well and is not too complex.

Operator	Formula	Operator	Formula
Sum	R+L	RootB	$\sqrt{R}$
DiffA	L-R	LogA	$log(L)$
DiffB	R-L	LogB	$log(R)$
Mult	R*L	ExpA	e^L
DivA	L/R	ExpB	e^R
DivB	R/L	SinA	$\sin(L)$
PowerA	L^R	SinB	$\sin(R)$
PowerB	R^L	Left	L
RootA	$\sqrt{L}$	Right	R

Table 6.1: The operators that are used in the transformation functions. L refers to the value of the left subtree, and R refers to the value of the right subtree.

Each transformation function can be represented by a binary tree, consisting of nodes that represent (binary or unary) operators, input parameters, or constants. The operators that we can select for a transformation tree are listed in Table 6.1. A root node can only be a constant or an input parameter; other nodes can only be operators. The second transformation function of example 6.2 is depicted in Figure 6.1.

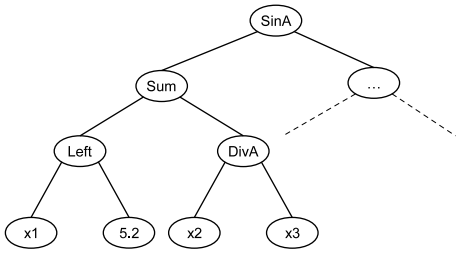


Figure 6.1: Example of the transformation function $\sin\left(x_1 + \frac{x_2}{x_3}\right)$.

To make reference to a node in the parse tree easier, we assign an index to each node. The assignment is done from left to right as depicted in Figure 6.2. This numbering has the advantage that we can distinguish between function and terminal nodes simply by looking at the index. All function nodes have an odd numbered index, and all terminal nodes have an even numbered index. Since terminal nodes have a distinct application in our algorithm (they can contain only variables or constants), this property speeds up the algorithm.

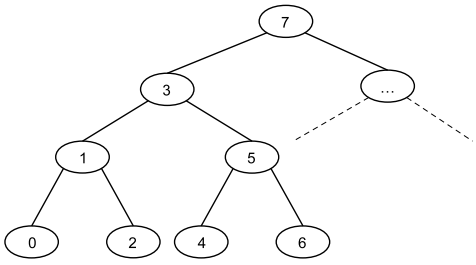


Figure 6.2: The data structure of the model tree.

6.2.2 Finding the best transformation functions

In this section, we describe how to find good transformation functions that make up the metamodel. First, we describe the basic algorithm. Then, we zoom into details on data representation, the quality aspects of the model that we want to optimize, and their calculation. The simulated-annealing-based algorithm is formulated in Algorithm 6.1. For a general description of simulated annealing, we refer to Aarts and Korst (1989).

In the following, we describe each of these steps:

0. **Initialization**

   The transformation function i is initialized as depicted in Figure 6.3. The abbreviations "r.o." and "r.v." mean random operator and random variable, respectively. The number of transformation trees and their depth are user-defined and fixed during the algorithm. Not all trees need to have the same depth.

Algorithm 6.1 The Symbolic Regression algorithm based on Simulated Annealing.

```
0: Initialize: select a good initial set of transformation functions
   Repeat
1:   Adapt annealing temperature
2:   Select a transformation function
3:   Change the selected transformation function
4:   Check the integrity of the model
     If the integrity-check is OK then
5:     Calculate the coefficients of the model
6:     Evaluate the quality of the approximating model
7:     Accept the change if the model is better, or accept with probability
         based on annealing temperature when the model is worse
       If the model is changed
8:       Simplify the selected transformation function
       Endif
     Endif
9: Until stopping criterion is reached
```

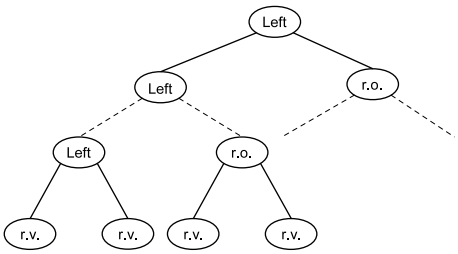


Figure 6.3: The initialized transformation function.

1. **Adapting the annealing temperature**
   We describe the changing of the annealing temperature in Step 7.

2. **Selecting a transformation function**
   A transformation function is randomly selected for change.

3. **Changing a transformation function**
   A transformation function is changed by randomly selecting a position in the function tree, i.e., by drawing a random integer between 0 and $2^{T+1}-1$ where T is equal to the tree depth. If the selected integer is odd, its contents are set to a randomly chosen operator. If the selected integer is even, its contents are set to a random input parameter or a constant. In case a constant is set, the value of this constant is chosen randomly. An example of such a change is the following. Suppose that at some point during the algorithm, a transformation function has the form depicted

in Figure 6.1. This tree evaluates to

$$f(x_1, x_2, x_3) = \sin\left(x_1 + \frac{x_2}{x_3}\right). \tag{6.3}$$

After changing the node containing the operator DIVA to ROOTA, the transformation function becomes

$$f(x_1, x_2, x_3) = \sin\left(x_1 + \sqrt{x_2}\right). \tag{6.4}$$

4. **Integrity checking**

   Since the transformation functions are changed randomly, it is possible that a proposed transformation cannot be evaluated in one or more points of the domain for which the model is used. Examples are division by zero and square roots of a negative number. To avoid this, we restrict the algorithm to those transformation functions that can be evaluated on the entire domain. The validity on a domain can be calculated using interval arithmetic (see Keijzer (2003)).

   The basic idea is the following: Given the domain of the input parameters, we can easily calculate a (sometimes conservative) domain for each node in the function tree. Using these node domains, we can easily verify whether a function is valid on the domain; for example, a partly negative domain in combination with a square root leads to an invalid function.

Operator	Lower bound	Upper bound	Invalid if
Sum	$lb_1 + lb_2$	$ub_1 + ub_2$	
DiffA	$lb_1 - ub_2$	$ub_1 - lb_2$	
DiffB	$lb_2 - ub_1$	$ub_2 - lb_1$	
Mult	$\min\{lb_1lb_2, lb_1ub_2, ub_1lb_2, ub_1ub_2\}$	$\max\{lb_1lb_2, lb_1ub_2, ub_1lb_2, ub_1ub_2\}$	
DivA	$\min\{lb_1/lb_2, lb_1/ub_2, ub_1/lb_2, ub_1/ub_2\}$	$\max\{lb_1/lb_2, lb_1/ub_2, ub_1/lb_2, ub_1/ub_2\}$	$0 \in [lb_2, ub_2]$
DivB	$\min\{lb_1/lb_2, lb_1/ub_2, ub_1/lb_2, ub_1/ub_2\}$	$\max\{lb_1/lb_2, lb_1/ub_2, ub_1/lb_2, ub_1/ub_2\}$	$0 \in [lb_1, ub_1]$
PowerA	$min\left\{lb_1^{lb_2}, lb_1^{ub_2}, ub_1^{lb_2}, ub_1^{ub_2}, 0\right\}$ if $0 \in [lb_1, ub_1]$	$max\left\{lb_1^{lb_2}, lb_1^{ub_2}, ub_1^{lb_2}, ub_1^{ub_2}, 0\right\}$ if $0 \in [lb_1, ub_1]$	$lb_1 < 0$
	$min\left\{lb_1^{lb_2}, lb_1^{ub_2}, ub_1^{lb_2}, ub_1^{ub_2}\right\}$ otherwise	$max\left\{lb_1^{lb_2}, lb_1^{ub_2}, ub_1^{lb_2}, ub_1^{ub_2}\right\}$ otherwise	
PowerB	$min\left\{lb_2^{lb_1}, lb_2^{ub_1}, ub_2^{lb_1}, ub_2^{ub_1}, 0\right\}$ if $0 \in [lb_2, ub_2]$	$max\left\{lb_2^{lb_1}, lb_2^{ub_1}, ub_2^{lb_1}, ub_2^{ub_1}, 0\right\}$ if $0 \in [lb_2, ub_2]$	$lb_2 < 0$
	$min\left\{lb_2^{lb_1}, lb_2^{ub_1}, ub_2^{lb_1}, ub_2^{ub_1}\right\}$ otherwise	$max\left\{lb_2^{lb_1}, lb_2^{ub_1}, ub_2^{lb_1}, ub_2^{ub_1}\right\}$ otherwise	
RootA	$\sqrt{lb_1}$	$\sqrt{ub_1}$	$lb_1 < 0$
RootB	$\sqrt{lb_2}$	$\sqrt{ub_2}$	$lb_2 < 0$
LogA	$\log(lb_1)$	$\log(ub_1)$	$lb_1 < 0$
LogB	$\log(lb_2)$	$\log(ub_2)$	$lb_2 < 0$
ExpA	$\exp(lb_1)$	$\exp(ub_1)$	
ExpB	$\exp(lb_2)$	$\exp(ub_2)$	
SinA	-1	1	
SinB	-1	1	
Left	lb_1	ub_1	
Right	lb_2	ub_2	

Table 6.2: Rules for interval arithmetic.

The interval arithmetic rules are denoted in Table 6.2. The columns called “lower bound” and “upper bound” describe the formulas needed to calculate the domain

of the function represented by a (sub) tree based on the domain of the left and the right subtree. The "invalid if" column describes when a (sub) tree is considered to be possibly invalid.

As we mentioned, this approach may be quite conservative: Valid functions may be rejected on the basis of the above rules. For example, consider Figure 6.4. Suppose that variable x_1 has a range of $[-1, 1]$. Using the rules, the domain of the subtree starting at "mult" is estimated at $[-1, 1]$. Therefore, this tree will be considered invalid because the square root could be taken of a negative number. However, the real range of the subtree starting at "mult" should be $[0, 1]$, leading to a valid tree. This is not a big problem though, because we simplify the function and check again if the function is considered invalid.

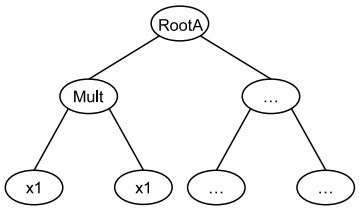


Figure 6.4: Example of a model tree for which the interval arithmetic rules are too strict.

5. **Calculating the coefficients**
   After the transformation functions are determined, calculating the coefficients is a linear regression problem. Note that only the linear constants by which the transformation functions are multiplied can be calculated; the remaining constants are entered randomly during a transformation function change.

6. **Evaluating the quality of a metamodel**
   The quality of the changed metamodel is evaluated on the basis of two criteria. First, the metamodel should fit well to the data in the training set. For this criterion, we calculate the root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y^i - f(x^i)\right)^2}, \tag{6.5}$$

   in which y^i represents the output of simulation i, x^i represents the input of the model of simulation i, m the number of simulations, and $f(x)$ the approximating model. Since trying to find a perfect RMSE does not prevent overfitting, we use the leave-one-out cross-validation measure, which is defined as

$$CV - RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y^i - f_{-i}(x^i)\right)^2}, \tag{6.6}$$

where f_{-i} denotes the metamodel obtained by fitting all simulation data except simulation i. Therefore, a fit is made m times based on $m-1$ simulations, where the remaining simulation is used only for model validation. Note that the CV-RMSE can be quickly evaluated using the projection matrix (this method uses only one calculation of a solution to a system of linear equations, whereas m solutions to a linear system of equations are needed in the straightforward method for calculating formula (6.6):

$$CV - RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{y^i - f(x^i)}{1 - H_{i,i}}\right)^2}, \tag{6.7}$$

where

$$H = X(X^\top X)^{-1}X^\top. \tag{6.8}$$

Note that for the calculation of the CV-RMSE, we refit only the linear coefficients and not the model structure.

Further, the metamodel should be interpretable, i.e., not too complex. A measure that estimates this quality is described in Section 6.3.2. For now, we consider only the objective function as a user-specified linear combination of the RMSE and the CV-RMSE. In Section 6.3.3, we describe how the approach handles multiple objectives.

7. **Acceptance of a change**
   If the change in the transformation function results in an improvement of the objective function, the change is always accepted. To avoid getting stuck in a local minimum, we sometimes have to accept a deterioration of the objective. The greater the deterioration, the smaller is the probability of acceptance. The probability of acceptance is given by

$$P = e^{-|\Delta objective|/c}, \tag{6.9}$$

   where $\Delta objective$ is the change in the quality of the metamodel compared with the previous iteration, and parameter c is the annealing temperature. In simulated annealing, this parameter gradually decreases in each iteration, which decreases the probability of accepting big deteriorations of the objective. The general idea is that at the start of the search, we would like to have a broad look at all parts of the solution space. Thus, we have to accept relatively large deteriorations. The closer we get to the end of the search, the smaller the chance that a large deterioration will eventually lead to an improvement of the objective; hence, the smaller the probability should be that large deteriorations are accepted.

A difficulty using simulated annealing is determining an initial value for the annealing temperature c, because this choice implies that we have to quantify a "large deterioration". This is particularly difficult in symbolic regression because this means that we need to be able to tell beforehand how well the quality of a function may become. Therefore, we introduce the new concept of reannealing: the annealing is started with a temperature that is a percentage of the first objective value. After a number of iterations, it is checked how often a change is accepted or rejected. If too many changes are accepted, the starting temperature was too high, so we start the annealing again with half the temperature. If on the other hand too few changes were accepted, the temperature was too low and we start with double the temperature. We continue this process until a suitable temperature is found.

8. **Simplification**
   Obviously, random changes in the transformation functions may lead to functions that can be simplified. Symbolic simplification requires much computation time though. Therefore, only some simple rules are checked after each iteration:

   (a) If a function node evaluates to a constant, the subtree starting at that node is replaced by a constant term. The value of the constant term is equal to the constant output of the function node.

   (b) If the left and right arguments of a function node are equal, we can make replacements for certain functions. For example:

   - $f(x) + f(x)$ is replaced by $2f(x)$.
   - $f(x) - f(x)$ is replaced by 0.
   - $f(x)/f(x)$ is replaced by 1.
   - $f(x)f(x)$ is replaced by $f^2(x)$.

   These simplifications are carried out only if the changed solution is accepted by the simulated annealing algorithm. The simplification rules are applied as long as changes are made during the previous application of the simplification rules. Whether this simplification step has a positive effect on the search varies very much with the application. The effect may be negative as simplifications make the transformation functions smaller, causing a change to be relatively large on average. Future research should show whether this step is beneficial for the search.

9. **Stopping criterion**
   If either the maximum number of iterations is reached or the approximation is evaluated as good enough, then the algorithm stops. The maximum number of iterations is typically $200,000$.

6.3 Extensions to the basic algorithm

6.3.1 Reasons for extension

As we mentioned in the introduction, metamodels are often used when getting data through the actual model is too time consuming. Fitting functions on these data sets implies the risk of finding a function that fits well only on this particular data set but does not describe the general behavior of the system. This problem is called overfitting. Using the CV-RMSE instead of only the RMSE to measure the quality of the metamodel reduces the risk of overfitting, but it does not eliminate it.

Furthermore, to improve interpretability of the metamodel, it is usually wise to limit the depth of the function trees. However, this has two drawbacks. Firstly, the function-tree depth does not always represent the complexity of a function; e.g., a sine operator is often considered more difficult to interpret than a plus-operator although they both require one function node. Secondly, the upper bound on the tree depth has to be set by the user, but is difficult to set beforehand.

The extensions we describe in the next subsections are meant to reduce overfitting and improve interpretability. First, we introduce a complexity measure that aims to measure interpretability and penalize possible overfitting. Second, instead of putting an upper bound on the complexity value we add minimization of complexity as a second objective. To deal with these two objectives, we use an algorithm based on Pareto simulated annealing.

6.3.2 Complexity measure

The basic idea of the complexity measure is that the complexity of a model is measured by the minimal degree of the polynomial necessary to approximate the model with a precision ϵ in a set of points S. The idea is that overfitted metamodels often have high oscillation. This makes them difficult to approximate, and results in a polynomial of a high degree to obtain the required precision.

To determine the necessary complexity of the approximating polynomial, we use the function-tree representation. We calculate the complexity in every node from the terminal nodes to the root node. We use calculation rules for the binary and nested operators. For the unary operators, we calculate the complexity by successively approximating the function through a polynomial of increasing degree. This method is based on Garishina and Vladislavleva (2004). However, we use different calculation rules and a different method for approximation of unary operators. The main difference is the method for approximation of unary operators. In the measure of Garishina and Vladislavleva, a polynomial with increasing degree is fitted in a fixed number of points until the approximation of the polynomial to the unary operator in these points is accurate enough. In

our approach, we apply polynomial interpolation through an increasing number of training points until the accuracy in a certain test set is high enough. Other differences are that we use some different calculation rules and take the number of nodes in a function explicitly into account to make the measure more discriminative.

To determine the complexity of the terminal and binary function nodes, we use a set of calculation rules. The complexity of a constant node is zero, and the complexity of a variable node is one. The rules for binary and nested operators are derived from relations between polynomial interpolations and are listed in Table 6.3.

Operator	Complexity rule
$f(x) + g(y)$	$\max\{compl(f(x)), compl(g(y))\}$
$f(x) - g(y)$	$\max\{compl(f(x)), compl(g(y))\}$
$f(x) * g(y)$	$compl(f(x)) + compl(g(y))$
$f(x)/g(y)$	$compl(f(x)) + compl(1/g(y))$ (special case of $f(x) * g(y)$)
$f(g(y))$	$compl(f(x)) * compl(g(y))$
$f(x)^{const}$	$compl(f(x)) * const$ if $const \geq 0 \wedge const \in \mathbb{N}$ (special case of $f(x) * g(y)$)
	$compl(f(x)) * compl(y^{const})$ otherwise, where y is a variable with the same range as $f(x)$
$const^{f(x)}$	$compl(f(x)) * compl(const^{y})$ where y is a variable with the same range as $f(x)$
$f(x)^{g(y)}$	Use the relation $f(x)^{g(y)} = \exp(g(y) * \log(f(x)))$ in combination with the previous rules

Table 6.3: Complexity rules for binary and nested operators.

Let us consider the formula $h(x) = x^6 + 6x$. This formula can be written as $f(x) + g(x)$ with $f(x) = x^6$; and $g(x) = 6x$. With the sixth rule, we can find that $f(x)$ has complexity 6 and $g(x)$ has complexity 1. The first rule now tells us that $h(x)$ has complexity 6.

The complexity rules serve as an approximation of the complexity defined by the minimal degree of the polynomial necessary to approximate the model. To explain these calculation rules, let us first define $P_S(f(x))$ as a polynomial that interpolates $f(x)$ in a set of points S. The function $P_S(f(x)) + P_T(g(y))$ then forms a polynomial interpolation of $f(x) + g(y)$ in the set of points $S \times T$ because the sum of two polynomials is again a polynomial. The degree of this polynomial is at most the maximum of the degrees of $P_S(f(x))$ and $P_T(g(y))$. This explains the calculation rule for addition.

The calculation rule for nested functions follows from replacing every x in $P_S(f(x))$ by $P_T(g(y))$. This gives a polynomial with a degree equal to the product of the degrees of $P_S(f(x))$ and $P_T(g(y))$. The other rules are obtained in a similar way and are therefore not discussed.

The complexity of a unary function node is determined in the following way. First, we determine the minimal size of the training set such that the unique polynomial interpolation through the data in the training set approximates the unary operator well enough. The polynomial interpolation is considered only on the range of the input argument of the unary operator. This range is determined by the interval arithmetic (see Section 6.2.2, step 3). The approximation is considered good enough if the maximum approximation error on a test set is below a user-defined threshold ϵ. The test set consists of a number

of data points located between the interpolation points.

The algorithm to determine the complexity of a unary function node is described in Algorithm 6.2.

Algorithm 6.2 Method to determine the complexity of a unary operator.

```
Initialize: Approximate the range of the input values with interval arithmetic
            Set k = 1
Repeat
  Increase k by 1
  Create a training set consisting of k Chebyshev points
  Find the interpolating polynomial of degree at most k-1
  Create a test set
Until the maximum error on the test set is below ϵ

Complexity of the unary operator is k-1
```

The reason for sampling Chebyshev points instead of equidistant points for the training set is avoiding Runge's phenomena, which states that for some functions, the approximation by a polynomial interpolation through an increasing number of equidistant points on a fixed interval gets worse when the number of points increases. However, in Algorithm 6.2, we assume that the approximation does get better as we use more points. Fortunately, this does hold for Chebyshev points because they do not suffer from Runge's phenomena (Mathews and Fink (2004)).

The procedure to determine the complexity of binary operators might seem time consuming. However, when using it in our simulated-annealing-based algorithm, it can be calculated quite efficiently. After changing one node in a function, we have to recalculate only the complexity of the nodes between (and including) the changed node and the root node of the transformation function. This limits the number of times we have to use Algorithm 6.2 per iteration of the simulated annealing algorithm. The computation time can also be limited by setting a maximum to the value of k in Algorithm 6.2.

6.3.3 Pareto simulated annealing

Now that we have determined a complexity measure, we still need a method for finding a function with a desired quality and complexity. The main problem is that functions with better fit generally have a higher complexity. The trade-off decision between fit and complexity is difficult to make before we have any results. We regard maximizing metamodel quality and minimizing complexity as two separate objectives and use a multiobjective combinatorial optimization (MOCO) method to find multiple good solutions.

The use of simulated annealing in the basic algorithm makes it a natural choice to use the multiobjective extension called Pareto simulated annealing (see Czyżak and

Jaszkiewicz (1998)). Pareto simulated annealing does not try to find a solution that is optimal according to one objective, but finds an approximation of the Pareto set. The Pareto set is the set of Pareto optimal solutions. In our situation, Pareto optimal solutions are metamodels for which the fit cannot be improved without deteriorating the complexity and vice versa.

The two main differences between "standard" simulated annealing and Pareto simulated annealing are the method for comparing two solutions and the method for determining the performance difference of two solutions. In Pareto simulated annealing, comparing two solutions $f_1(x)$ and $f_2(x)$ leads to four possible scenarios:

- $f_1(x)$ and $f_2(x)$ are equally good: $f_1(x)$ and $f_2(x)$ are equally good on all objectives.
- $f_1(x)$ dominates $f_2(x)$: $f_1(x)$ is at least equally good as $f_2(x)$ on all objectives and better on at least one.
- $f_1(x)$ is dominated by $f_2(x)$: $f_1(x)$ is at most equally good as $f_2(x)$ on all objectives and worse on at least one.
- $f_1(x)$ and $f_2(x)$ are mutually non-dominating: $f_1(x)$ is worse than $f_2(x)$ on at least one objective and better on at least one other objective.

We choose to always accept $f_2(x)$ if it dominates $f_1(x)$ or if it is equally good as $f_1(x)$. We accept $f_2(x)$ only with a certain probability if it is dominated by $f_1(x)$ or if $f_1(x)$ and $f_2(x)$ are mutually non-dominating.

The acceptance probability depends on the performance difference of the two solutions. To determine this difference, we need to convert the performances on multiple objectives into a single measure. We choose to use the dominance-based performance measure, introduced in Smith et al. (2004), which solves some drawbacks of more traditional measures like the weighted sum. The dominance-based performance measure is defined by:

$$\Delta E(f_1(x), f_2(x)) = \frac{1}{|\widetilde{APF}|}\left(|\widetilde{APF}_{f_1(x)}| - |\widetilde{APF}_{f_2(x)}|\right) \tag{6.10}$$

with:

- APF: set of solutions that approximate the Pareto front.
- $\widetilde{APF}$: $APF \cup \{f_1(x), f_2(x)\}$.
- $\widetilde{APF}_{f(x)}$: set of solutions in $\widetilde{APF}$ that dominate $f(x)$.

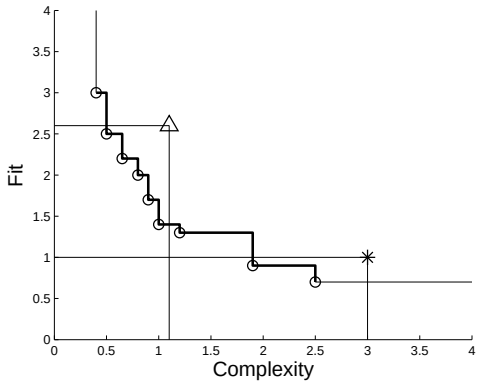


Figure 6.5: Example of the APF and the attainment surface.

An example of an APF is depicted in Figure 6.5 by the circles. When we let the triangle depict $f_1(x)$ and the star depict $f_2(x)$, then $|\widetilde{APF}| = 11$, $|\widetilde{APF}_{f_1(x)}| = 5$ and $|\widetilde{APF}_{f_2(x)}| = 2$. The value of the dominance-based performance measure thus becomes 3/11.

Therefore, the performance of a solution is measured by the percentage of solutions in $\widetilde{APF}$ that it dominates. An advantage of this measure is that we always accepted solutions that are not dominated by a solution in the current APF—which is not always the case with other measures. Furthermore, solutions in sparsely populated areas of the APF generally have a better performance than solutions in densely populated areas. This stimulates the search of a set of solutions that is evenly spread over the APF.

The main drawback of the measure is that performance differences may be relatively high when the APF contains few points. This may result in a coarse acceptance probability distribution, implying that a slightly worse fit or complexity may result in a large reduction in acceptance probability. Smith et al. (2004) present several methods to alleviate this problem. The solution we choose is to create extra points evenly spread on the "attainment surface of the APF" described in Smith et al. (2004), which is the boundary of the area containing all points dominated by the points in the APF. The points in the APF are thus on the attainment surface. In Figure 6.5, the attainment surface is depicted by the black line. The extra points are created on the part of the attainment surface that lies between the solution with the highest complexity and the solution with the lowest complexity. This part of the attainment surface is depicted in Figure 6.5 by the thick part. Half of the extra points are now evenly spread over the horizontal parts and the other half over the vertical parts. An important advantage of this solution is that it maintains both advantages of the dominance-based performance measure.

The Pareto simulated annealing version of Algorithm 1 is described in Algorithm 3. Note that steps 6 and 7 are adapted and step 9 is added. To efficiently store and update the APF, we use two vectors containing the complexities and fits of the functions in the APF, respectively. In these vectors, the functions are ordered according to increasing

complexity. For functions in the APF, this is equivalent to sorting them according to decreasing fit. Through these two vectors, we can easily determine if the current solution dominates or is dominated by solutions in the APF. We change the APF only in two ways. Firstly, if some solutions in the APF are dominated by the current solution, we remove these solutions. Secondly, if no solution in the APF dominates our current solution, the current solution is added to the APF.

Algorithm 6.3 The Pareto simulated annealing version of the symbolic regression algorithm. The differences to Algorithm 6.2 are noted in italics.

```
0: Initialize: find a good initial set of transformation functions
   Repeat
1:   Adapt annealing temperature
2:   Select a transformation function
3:   Change the selected transformation function
4:   Check the integrity of the model
     If the integrity-check is OK then
5:     Calculate the coefficients of the model
6:     Evaluate the quality and complexity of the approximating model
7:     Always accept the change if the model dominates its predecessor or if it
         is equally good.
       Otherwise, accept the change with a probability based on the performance
         difference and the annealing temperature.
       If the model is accepted
8:       Simplify the selected transformation function
9:       Compare the model with the current APF and update the APF if necessary
       Endif
     Endif
10: Until stopping criterion is reached
```

6.4 Numerical comparison to other metamodel types

In this section, we present a comparison between our suggested approach and two other metamodels approaches. First of all, we compare the results of our algorithm to the Kriging model (for details on fitting Kriging models, see Sacks et al. (1989b)). Secondly, we compare the results to symbolic regression based on genetic programming. We use the implementation of Smits and Kotanchek (2004) to compare our results.

Sections 6.2 and 6.3 show that there are many parameters in our algorithm that may influence the numerical results. However, the reannealing concept takes care of the simulated annealing parameters. Most other parameters that we varied turned out to have no important effect on the quality of the model, except the tree depth and the number of trees. We varied these parameters in the first case.

6.4.1 The six-hump-camel-back function

The first case is called the six-hump-camel-back function; see Figure 6.6. This is an explicit formula, which has the advantage that we can accurately assess how the approximations compare with the real function. For the training set, we created a two-dimensional space-filling latin hypercube design (LHD) containing 30 experiments in the range $[-2, 2] \times [-1, 1]$. Using 30 experiments, it should be possible to build an accurate model. A space-filling (Maximin) LHD is often used for the approximation of deterministic simulation models. For details on the construction of such LHDs, we refer to Morris and Mitchell (1995).

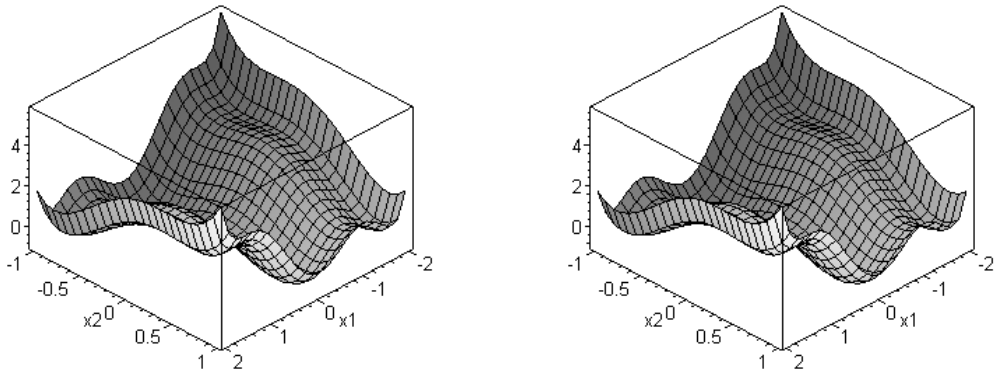


Figure 6.6: The six-hump camel back function (left) and the metamodel (right).

The explicit formula for the six-hump-camel-back function is given by:

$$f(x_1, x_2) = x_1^2 \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) + x_1 x_2 + x_2^2 \left(-4 + 4x_2^2\right). \tag{6.11}$$

The best symbolic regression function that was found is selected by choosing the function with the lowest RMSE with an acceptable complexity from the Pareto front. This selection process depends on the preferences of the user; e.g., if a simple model is requested, then a worse model fit would be accepted. The chosen metamodel is given by

$$\begin{aligned} f(x_1, x_2) \;=\; & c_1 + c_2 \cos(x_1) + c_3 \cos(2x_1) + c_4 (x_1 x_2) + c_5 \cos(x_2) + \\ & c_6 \sin\left(c_7 x_1^2\right) + c_8 \left(x_1^2\right), \end{aligned} \tag{6.12}$$

where c_i is a constant. Note that all constants c_i can be fit using linear regression except c_7, which was randomly entered by our algorithm. Further note that formula (12) has very little resemblance to formula (11). If we would have carried out enough iterations, and the original function would fit in the model trees that we would have used (i.e., enough terms and tree depth), then the algorithm would probably have come up with the original function.

The APF after $200,000$ iterations was calculated in a few minutes on a PC with 2.00 GHz Intel Pentium M processor. The time intensive part of each iteration is the

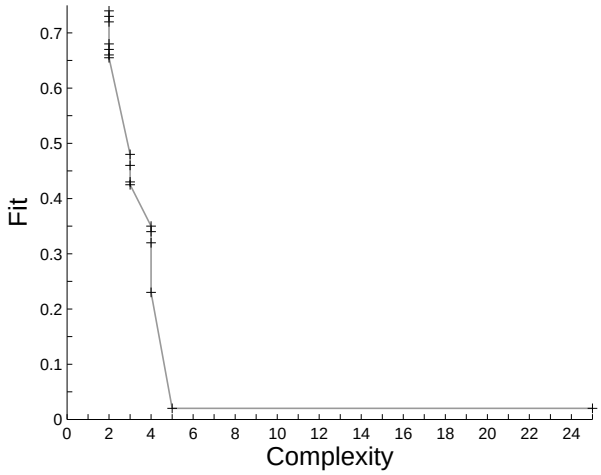


Figure 6.7: A linear interpolation of the points in the APF for the six-hump camel back case after 200,000 iterations, in which the crosses represent the Pareto optimal solutions. Note that this is not the attainment surface of the APF. The fit on the vertical axis is a linear combination of CV-RMSE and RMSE.

calculation of the least squares coefficients and the cross validation statistic. This is an $o(n^3 + m^2)$ operation where n is equal to the number of transformation functions and m is equal to the number of experiments. The APF is depicted in Figure 6.7. To evaluate the actual quality of the metamodel, we created another test set of 30 experiments by extending the original LHD to a new space-filling LHD consisting of 60 experiments. The results are shown in Table 6.4.

Metamodel	Training set RMSE	Test set RMSE
Kriging	0.000	0.112
Symbolic regression model/SA	0.037	0.039
Symbolic regression model/GP	0.114	0.141

Table 6.4: Results for the metamodels on the six-hump-camel-back case.

These results were found with a maximal tree depth of 4. Increasing the tree depth to 6 led to considerably more complex metamodels, but also to better results on the test and training set.

6.4.2 The Kotanchek formula

The second test case is also based on an explicit formula. This example originates from Smits and Kotanchek (2004). The data consists of five input variables $(x_1, \ldots, x_5) \in [0, 4]^5$ of which only the first two (x_1 and x_2) are significant. The output variable (y) consists of two parts. The first part is an explicitly known formula; the second part is noise. The

former formula is given by

$$y(x_1, x_2, x_3, x_4, x_5) = \frac{e^{-(x_2-1)^2}}{1.2 + (x_1 - 2.5)^2}, \tag{6.13}$$

to which we refer as the Kotanchek formula. In Figure 6.8, a plot of the Kotanchek formula is depicted.

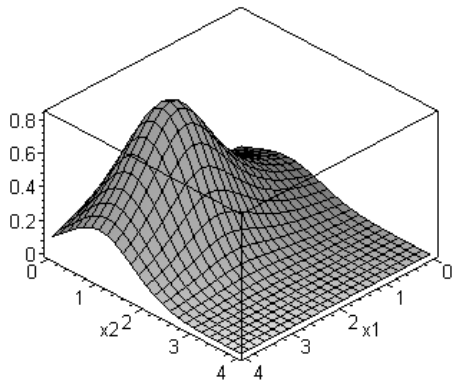


Figure 6.8: The Kotanchek formula (6.13).

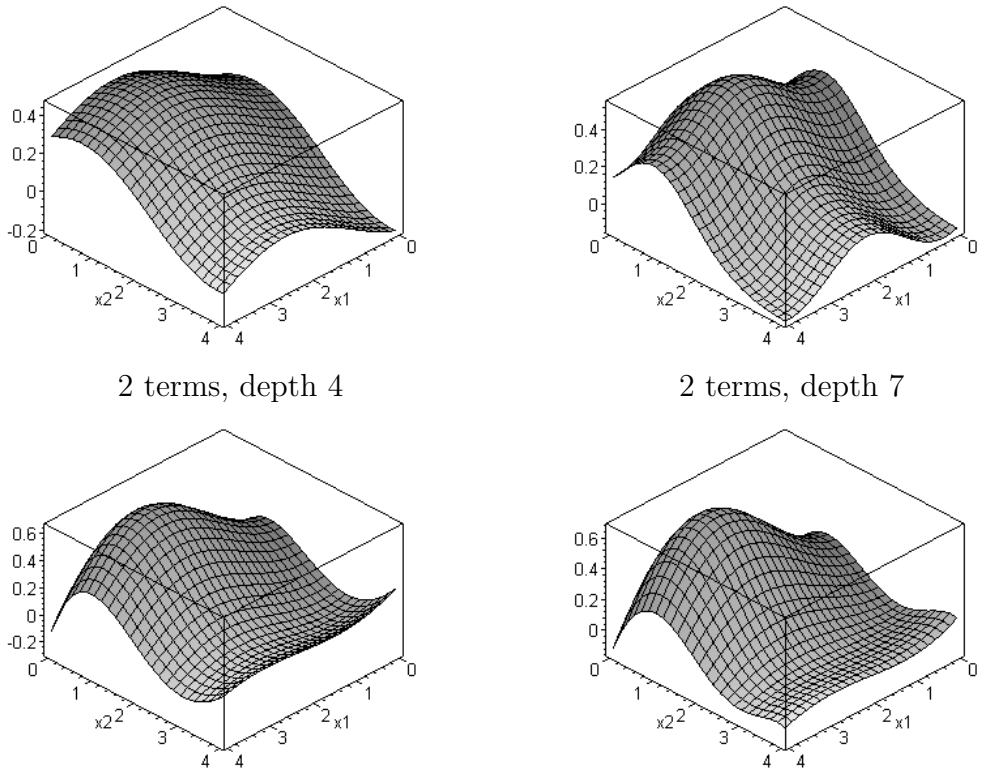
We assume the noise to be uniformly distributed in the interval $[-0.05, 0.05]$. On 100 data points (again a spacefilling LHD), four Pareto fronts were created in $200,000$ iterations using different settings of the algorithm: two or six terms and a tree depth of four or seven. For each of the settings, a function was chosen from the Pareto front by selecting a function with a low RMSE and an acceptable complexity, for which the RMSE cannot be improved much by selecting a function with higher complexity. Then, a test set of 50 data points is generated by extending the LHD.

Metamodel	No. terms	Tree depth	Training set RMSE	Test set RMSE
Symbolic regression model/SA 1	2	4	0.111	0.090
Symbolic regression model/SA 2	2	7	0.104	0.088
Symbolic regression model/SA 3	6	4	0.056	0.052
Symbolic regression model/SA 4	6	7	0.041	0.037
Kriging			0.000	0.039
Symbolic regression model/GP			0.033	0.048

Table 6.5: The training set and test set results for the five symbolic regression models and the Kriging model.

Table 6.5 contains the results for this experiment. The results are compared with the Kriging model. On the test set, the result of the Kriging model is comparable to the best symbolic regression model. It is far less interpretable though, because it consists of 100 terms in which all 5 dimensions are present. To detect the redundant variables and improve the interpretability, we could test the magnitudes of the correlation

parameters of the Kriging model. For the symbolic regression models, the minimization of the complexity measure for a given RMSE should prevent that the resulting models contain redundant variables. When we consider the four symbolic regression models, we see that metamodels 1, 2 and 4 indeed consist of only variables x_1 and x_2. Metamodel 3 consists of variables x_1, x_2 and x_3, so it contains only one of the three redundant variables. The four metamodels are depicted in Figure 6.9. We conclude that in this experiment, even the model with six terms and a tree depth of seven did not overfit the data.



2 terms, depth 4 | 2 terms, depth 7

6 terms, depth 4 | 6 terms, depth 7

Figure 6.9: The four metamodels for the Kotanchek set. The variable x_3 is fixed to 0.

6.5 Conclusions

In this chapter, we have described a simulated-annealing-based approach to symbolic regression. We have elaborated on the algorithm and data structures, and have presented the results based on two cases. We conclude that although it requires some effort to find the best symbolic regression model, the quality of the metamodels that can be found is very promising. A major advantage of symbolic regression compared with Kriging is interpretability. The complexity measure described in this chapter quantifies the interpretability. Although the expressions used by the symbolic regression metamodels

are less complex, the fit results are comparable with or better than the Kriging models. Compared with genetic programming approaches, our method offers a better way of dealing with constants in the model via linear regression. In genetic programming, estimating constants is a major issue. Furthermore, the number of models that need to be evaluated in a simulated annealing method will on average be smaller compared with the tens of thousands of models that need to be evaluated in each generation of Genetic Programming. Usually, many generations are needed to come up with an acceptable model. However, further testing on a wider variety of test functions needs to be performed to validate these claims. The Pareto simulated annealing algorithm produces a list of solutions with different fit/complexity trade-offs. This gives users the flexibility to choose the metamodel that best fits their preferences.

Nevertheless, there are still some open issues. One significant improvement of the algorithm might be found by first applying a number of transformations of the response data. Next, the search procedure can check the metamodel on all transformation without much computational effort, and select not only the best transformation functions for the input parameters, but also the best transformation for the output parameter. Another interesting extension of the algorithm could be fitting rational functions of transformation functions. This would increase the number of parameters in the metamodel that can be efficiently calculated, and thus probably increase the quality of the model. Finally, it would be beneficial to be able to dynamically alter the number of terms and the depth of the trees during the search, so the users would not need to make these decisions themselves.

Part IV

Sandwich algorithms for approximating convex Pareto sets

Chapter 7

Enhancement of sandwich algorithms for approximating multi-dimensional convex Pareto sets

> The Bureau of Incomplete Statistics reports that one out of three.
>
> (Unknown)

7.1 Introduction

In many fields, we come across problems where we want to optimize several conflicting objectives simultaneously. An engineering example is the optimization of ride safety and ride comfort when designing car suspensions (Nguyen (2007)). But also in health care, optimization of multiple objectives can be an issue. When treating cancer tumors, maximizing the probability of eradicating a tumor and minimizing the probability of damaging healthy tissue are two conflicting objectives. These are just two of many possible applications of multi-objective optimization. For a comprehensive overview of the range of applications, we refer to White (1990), which lists 500 papers describing different applications in various fields.

Given the large number of applications, it is not surprising that multi-objective optimization is an active field of research. Many approaches have been developed to deal with optimizing multiple objectives. When there exists no solution that optimizes all objectives simultaneously, the concept of Pareto optimality is often used to find a solution for a multi-objective problem (MOP). A solution is called Pareto optimal if it is not possible to improve an objective without worsening one or more of the other objectives. The set of all Pareto optimal solutions is called the Pareto set or Pareto frontier. Many multi-objective optimization techniques focus on approximating (part of) this Pareto set. Such an approximation can be used by human decision makers to get insight into the

trade-offs among the various objectives and to select a solution that best satisfies their preferences. For a general discussion of multi-objective optimization techniques, we refer to the books of Hwang and Masud (1979), Steuer (1986), Miettinen (1999), Ehrgott (2005), and Branke et al. (2008), and the survey papers of Ruzika and Wiecek (2003), Marler and Arora (2004), and Ehrgott and Wiecek (2005).

In this chapter, we consider the class of multi-dimensional convex MOPs. By multi-dimensional we mean that the MOP has more than two objectives. We make this distinction because many methods apply only to two objectives. Nevertheless, the methods discussed in this chapter can also be applied to bi-objective convex MOPs. An MOP is called convex if all objective functions and the set of all feasible solutions are convex (Miettinen (1999)). Romeijn et al. (2004) have shown that for these problems the Pareto set is also convex. The choice for this class of problems was inspired by the intensity-modulated radiation therapy (IMRT) optimization problem. This problem deals with designing a beam fluence map that delivers enough radiation dose to a cancer tumor to eradicate it, while keeping the dose in other tissue low enough to avoid damage. Different objective functions can be used to formulate this problem as an MOP. Romeijn et al. (2004), Hoffmann et al. (2006), and Siem et al. (2008) have shown that many commonly used objective functions are convex or can be transformed into convex functions without changing the Pareto set. For an overview of papers treating the IMRT problem as an MOP, we refer to Romeijn and Dempsey (2008) and the references therein. Note that although our research was inspired by the IMRT problem, the methods discussed in this chapter apply to all convex MOPs and even to certain non-convex MOPs.

There exists a wide variety of methods for approximating multi-dimensional convex Pareto sets (see, e.g., Ruzika and Wiecek (2003), Marler and Arora (2004), Ehrgott and Wiecek (2005), and Karasakal and Koksalan (2009)). However, we focus on multi-dimensional sandwich algorithms because they have several interesting and useful properties. *Sandwich algorithms* approximate the Pareto set by iteratively improving an inner and outer approximation. As the real Pareto set is sandwiched between the inner and outer approximation, an upper bound on the approximation error can be determined. We consider this property to be a major advantage over other types of methods as the latter generally cannot provide information on the accuracy of the approximation. The availability of this information has two benefits. Firstly, most sandwich algorithms use this information to determine which part of the approximation needs to be improved in each step of the algorithm. By improving the part where the upper bound on the approximation error is largest, sandwich algorithms efficiently improve the approximation. Secondly, the decision maker can use this approximation error to determine if a certain approximation is accurate enough. However, in order for the error to be useful in practice, the error measure should be easy to calculate and interpret.

As with general MOP-algorithms, many sandwich algorithms are developed for the bi-objective cases only and cannot directly be extended to higher dimensions. However, there are several sandwich algorithms that can deal with multi-dimensional MOPs. Solanki et al. (1993), for instance, have extended their bi-objective method to make it suitable for higher dimensions. Although their method is described for multi-objective linear programming problems, it can also be applied to general convex MOPs. Klamroth et al. (2002) also introduced methods for generating outer and inner approximations. The sandwich method of Craft et al. (2006) was introduced for multi-dimensional IMRT problems, but can also be applied to other convex MOPs. Lastly, Shao and Ehrgott (2008) also developed a sandwich algorithm motivated by the IMRT problem. However, their method deals with it as a multi-objective linear programming problem and cannot easily be extended to general convex MOPs.

We extend multi-dimensional sandwich algorithms in three different ways. Firstly, we introduce the new concept of adding dummy points to the inner approximation of a Pareto set. By using these dummy points we can determine accurate inner and outer approximations more efficiently, i.e., using fewer time-consuming optimizations. We illustrate this result by enhancing the method of Solanki et al. (1993) with dummy points and comparing this method with existing sandwich algorithms on a number of test cases. Furthermore, certain points of the inner approximation might be irrelevant as they are dominated by other points of the inner approximation. The detection of the relevant points can also be simplified by the use of dummy points.

Secondly, we introduce an error measure, which determines the quality of an approximation based on the concept of ϵ-dominance. An important benefit of this error measure is that it provides the decision maker with quality guarantees that are easy to interpret. However, the calculation of this error measure is not straightforward. Therefore, we introduce a new calculation method that simplifies the calculations by using dummy points. When calculating an upper bound for this measure using the inner and outer approximations, the method simplifies the calculations to solving a number of relatively simple LP problems. As the measure thus becomes easy to calculate, it can also be used in sandwich algorithms to determine which part of the approximation should be improved in each iteration. In this way, we are likely to obtain an accurate approximation more efficiently. Furthermore, it enables the decision maker to easily evaluate the accuracy of the approximations at each iteration of the algorithm. The combination of easy calculation and easy interpretation thus makes it a very suitable measure for sandwich algorithms.

Thirdly, we show how transforming certain objective functions can improve the results of sandwich algorithms or extend their applicability. Approximations of convex Pareto sets can be improved if we can find a strictly increasing concave transformation function such that the transformed objective is still convex. We prove that both inner and

outer approximations can be improved in this way. The improved accuracy means that even fewer time-consuming optimizations are needed to achieve a certain accuracy. By using transformation functions, we also extend the application of sandwich algorithms to certain non-convex MOPS. We show that if we can find a strictly increasing transformation function which transforms the non-convex objective functions into convex ones, we can use the sandwich algorithms to determine inner and outer approximations for non-convex Pareto sets. We also discuss the calculation of the introduced error measure when using transformations. For bi-objective MOPs, Siem et al. (2008) have already shown similar results by using transformations. Our results thus extend their findings to multi-dimensional MOPs.

To show the effect of these enhancements, we make a numerical comparison using four test cases. The set of cases consists of a three-dimensional strictly convex MOP, a five-dimensional linear MOP, a four-dimensional IMRT case, and a three-dimensional geometric programming case. The results of these cases show that we indeed need substantially fewer optimizations to achieve an accurate approximation when using the enhancements.

This chapter is organized as follows. In Section 7.2, we give a formal definition of the convex MOP and introduce the necessary notation. Section 7.3 contains descriptions of the above mentioned multi-dimensional sandwich algorithms. The concept of dummy points and their application is introduced in Section 7.4. In Section 7.5, we motivate and define the previously mentioned error measure and show how we can easily calculate it by using dummy points. The transformation of objective functions is discussed in Section 7.6. Calculation of the error measure when using transformations is also discussed in this section. In Section 7.7, we describe how the dummy points, error measure, and transformation functions can be applied in several sandwich and non-sandwich algorithms for approximating MOPs. To compare the sandwich algorithms and to show the effect of the above enhancements, Section 7.8 contains a numerical comparison consisting of four test cases. Finally, Section 7.9 finishes with concluding remarks.

7.2 Problem definition and notation

Throughout this chapter, we use the following orderings of vectors. Let $x, y \in \mathbb{R}^n$ with $n \geq 2$. By x_i we denote the i^{th} element of the vector x. To enumerate different vectors, we use superscripts. When ordering two vectors, we use:

- $x < y \Leftrightarrow x_i < y_i$ for all $i = 1, \ldots, n$.
- $x \leq y \Leftrightarrow x_i \leq y_i$ for all $i = 1, \ldots, n$ and $x \neq y$.
- $x \leqq y \Leftrightarrow x_i \leq y_i$ for all $i = 1, \ldots, n$.

The symbols $>$, $\geq$, $\geqq$ are defined accordingly. We furthermore define the set $\mathbb{R}^n_{\leqq} = \{x \in \mathbb{R}^n : x \leqq 0\}$. If $X \subset \mathbb{R}^n$, then we define $X + \mathbb{R}^n_{\leqq} = \{y \mid \exists x \in X : y \leqq x\}$. The sets $\mathbb{R}^n_{\geqq}$, $\mathbb{R}^n_{\leq}$, and $\mathbb{R}^n_{\geq}$ and the sets $X + \mathbb{R}^n_{\geqq}$ $X + \mathbb{R}^n_{\leq}$, and $X + \mathbb{R}^n_{\geq}$ are again defined accordingly.

In this chapter, we consider the following multi-objective optimization problem (MOP):

$$\begin{aligned} & \min_x f(x) = [f_1(x), \ldots, f_k(x)]^\top \\ \text{s.t.} \quad & x \in X \\ & f(x) \leqq z^{ub}, \end{aligned} \tag{7.1}$$

where X is the feasible set, $f : X \to \mathbb{R}^k_{\geqq}$ is the vector of k real valued objective functions and z^{ub} is an upper bound on the objective function values. Note that this definition differs in two ways from the common MOP. Firstly, we assume that all objective functions give values greater than or equal to zero. This is no practical limitation as adding a fixed constant to each objective value does not essentially change the problem or solution set. By simply adding the utopia point, which we define further on, we can ensure that an arbitrary objective function satisfies this condition. Secondly, the problem is usually formulated without the upper bound restriction on the objectives. In practice, however, the decision makers generally have an idea of the maximal value they are willing to accept for the different objectives. Solutions with higher objective values are irrelevant and can thus be avoided using this restriction.

As it is generally impossible to find an $x \in X$ that minimizes all objectives at the same time, our aim is to find a set of so-called Pareto optimal or non-dominated solutions.

Definition 7.1.
An objective vector $f(x)$ for $x \in X$ is (strongly) dominated if there exists an $\widetilde{x} \in X$ such that $f(\widetilde{x}) < f(x)$.
An objective vector $f(x)$ for $x \in X$ is weakly dominated if there exists an $\widetilde{x} \in X$ such that $f(\widetilde{x}) \leq f(x)$.

Definition 7.2.
An objective vector $f(x)$ for $x \in X$ is (strongly) Pareto optimal if there exists no $\widetilde{x} \in X$ such that $f(\widetilde{x}) \leq f(x)$.
An objective vector $f(x)$ for $x \in X$ is weakly Pareto optimal if there exists no $\widetilde{x} \in X$ such that $f(\widetilde{x}) < f(x)$.

Pareto optimality thus implies that it is not possible to improve one objective without deteriorating at least one other objective. If, on the other hand, such an improvement is possible, then $f(x)$ is weakly dominated. When it is possible to improve all objectives simultaneously, then $f(x)$ is strongly dominated. The concept of Pareto optimality is also known under the names efficiency and non-dominance. The different terms are sometimes

used to distinguish between points in the design space and objective space, but—as Ehrgott (2005) points out—there is no consensus on which term should be used for which space. In this chapter, we choose to use efficiency for points in the design space and Pareto optimality for points in the objective space. The set of all feasible efficient solutions is thus a subset of X and is denoted by X_E. The set containing all the Pareto optimal vectors corresponding to the efficient solutions, i.e., $PS := \{f(x) \mid x \in X_E, f(x) \leqq z^{ub}\}$, is called the Pareto set. The set PS is a subset of the set Z of all feasible criterion vectors, which is defined as $Z := \{f(x) \mid x \in X, f(x) \leqq z^{ub}\}$. We denote vectors in the solution space by x and vectors in the objective space by z. Furthermore, we refer to vectors x as solutions and to vectors z as points.

We assume that the decision maker has selected the upper bounds z^{ub} such that all solutions in PS are viable solutions. The decision maker is thus interested in the complete Pareto set. However, as determining the complete Pareto set is generally impossible, we try to approximate it. To determine Pareto optimal points, an often used method is to solve the following weighted sum problem:

$$z^* = \arg\min\{w^\top z \mid z \in Z\} \tag{7.2}$$

where $w \in \mathbb{R}^k_{\geq}$. Advantages of this problem formulation are that it is often easy to implement and that the problem is convex for convex MOP. Furthermore for $w > 0$, solving this problem always produces a Pareto optimal point. If $w_i = 0$ for one or more $i \in \{1, \ldots, k\}$, then the resulting point is also guaranteed to be Pareto optimal if all corresponding objective functions $f_i(x)$ are strongly convex. However, when some of the corresponding objective functions $f_i(x)$ are weakly convex, the resulting point can be weakly Pareto optimal (see, e.g., Miettinen (1999)). To determine a point z^{**} that is guaranteed to be Pareto optimal and that weakly dominates or is equal to z^*, we can solve the following additional optimization problem:

$$z^{**} = \arg\min\{(w')^\top z \mid z \leqq z^*, z \in Z\},$$

where $w' > 0$. However, as we assume that the optimizations require much computation time and the benefit of this additional optimization may be relatively small, we do not perform this step for the test cases in this chapter.

A common approach to approximate a Pareto set is to solve the weighted sum problem for weight vectors w which are evenly spread over the set $\{w \mid w \geq 0, \sum_{i=1}^k w_i = 1\}$. However, Das and Dennis (1997) have shown that this method does not generally give an even spread of points from a Pareto set. We therefore need a more advanced method to efficiently determine a good approximation of the Pareto set.

Besides Pareto points, several auxiliary points in the objective space are often used to approximate the Pareto set. An anchor point z^{Ai} of an MOP is defined as

$$z^{Ai} = \arg\min\{z_i \mid z \in Z\} \text{ for } i = 1, \ldots, k,$$

which implies that one of the objectives is minimized without taking the other objectives into account. Note that anchor points can be found by solving the weighted sum problem with w equal to the i^{th} unit vector. The results concerning Pareto optimality of solutions of the weighted sum problem thus also apply to anchor points.

The utopia point z^U is found by taking the minimal values of all the objectives:

$$z_i^U = z_i^{Ai} \text{ for } i = 1, \ldots, k.$$

The utopia point is thus the best possible point for each objective, but in general it is infeasible. The nadir point is the opposite of the utopia point and obtained by

$$z_i^N = \max\{z_i \mid z \in PS\} \text{ for } i = 1, \ldots, k.$$

Even though the addition of the upper bound z^{ub} to the common MOP formulation can simplify the computations, determining the nadir point for $k > 2$ remains in general hard as we have to optimize over the unknown set PS (see, e.g., Miettinen (1999)). In Schandl et al. (2002), the following generalization of the nadir-point concept is proposed:

$$z_i^{pN} = \max\{z_i^{Aj} \mid j = 1, \ldots, k\} \text{ for } i = 1, \ldots, k.$$

We refer to this point as the pseudo-nadir point.

Furthermore, as we stated in the introduction, we assume that the MOP is convex, i.e., all objective functions $f_i(x)$ and the set X are convex. According to Romeijn et al. (2004), this implies that the set

$$Z + \mathbb{R}^k_{\geqq} = \{z \mid \exists \widetilde{z} \in Z : z \geqq \widetilde{z}\}$$

of dominated points is also convex. In Jin and Sendhoff (2004), the convexity of a Pareto set is formulated as follows.

Definition 7.3. *A set PS is convex if for all $u, v \in PS$ and for all $\lambda \in (0, 1)$, there exists a vector $w \in PS$ such that $\lambda u + (1 - \lambda)v \geq w$.*

The result of Romeijn et al. (2004) implies that also convexity according to the definition of Jin and Sendhoff (2004) is satisfied for convex MOP.

To approximate a convex Pareto set, we often use the convex hull of a set of points in Z. For $Y \subset Z$, the convex hull $\text{conv}\{Y\}$ is defined as the set of all convex combinations of points in Y. All points in $\text{conv}\{Y\}$ that cannot be written as a convex combination of other points in Y, are called extreme points of the convex hull. If Y is a finite set of points, the convex hull of Y can also be described through a finite set of hyperplanes in the objective space. Similar to Solanki et al. (1993), we use the following definitions.

Definition 7.4. *A hyperplane in the objective space is given by* $H(w,b) = \{z \mid w^\top z = b\}$ *with* $w \in \mathbb{R}^k \backslash \{0\}$ *and* $b \in \mathbb{R}$*. The vector* w *is a normal of the hyperplane. If* $\|w\| = 1$*, the vector* w *is a unit normal.*

Definition 7.5. *The set* $HS(w,b) = \{z \mid w^\top z \geq b\}$ *is the half-space given by* $w \in \mathbb{R}^k \backslash \{0\}$ *and* $b \in \mathbb{R}$*. The vector* w *is an inner normal of the half-space. If* $\|w\| = 1$*, the vector* w *is the inner unit normal of the half-space. The vector* $\bar{w} = -w$ *is an outer normal of the half-space.*

In this chapter, the vector w always refers to an inner unit normal, unless explicitly specified otherwise.

Definition 7.6. *If* $V \subset C$ *where* C *is a convex set, then a hyperplane* $H(w,b)$ *supports* C *at* V *if* $V \subset H(w,b)$ *and* $C \subset HS(w,b)$*.*

Definition 7.7. *A set of points* F *is a* m*-face of* C *if* F *has dimensionality* m *and there exists a supporting hyperplane* $H(w,b)$ *that supports* C *at* F *and for which holds that* $H(w,b) \cap C = F$*. If* $C \subset R^k$*, its* $(k-1)$*-faces are facets and its* 0*-faces are the extreme points.*

In this chapter, we mainly consider convex sets C with a finite number of facets. We denote the n facets belonging to a certain convex set C by $F^1, \ldots, F^n$ and the set of extreme points by C^E. When $H(w^i, b^i)$, $i = 1, \ldots, n$, are the supporting hyperplanes at these faces, then every point $z \in C$ must satisfy $(w^i)^\top z \geq b^i$ for $i = 1, \ldots, n$. If z satisfies any of these inequalities with equality, then z is on the corresponding facet. By taking the intersection of the half spaces $HS(w^i, b^i)$, $i = 1, \ldots, n$, defined by the facets, we can describe the convex hull as an intersection of half spaces.

To determine the convex hull, we use the function "convhulln" in Matlab, which uses the Qhull algorithm (Barber et al. (1996)). For convex hulls of dimensions at least five, the output of this function may contain facets with empty areas or volumes. These facets are removed because they are redundant.

In all algorithms used in this chapter, we approximate the Pareto set by a set of faces of $\text{conv}\{Y\}$ with $Y \subset Z$. For this set Y it holds that $\text{conv}\{Y\} = \text{conv}\{Y^E\}$ and that we know the corresponding $x \in X$ for all points $z \in Y^E$. However, we do not know the corresponding $x \in X$ for all other vectors $z \in \text{conv}\{Y\}$. For these vectors, we can use the following method to determine a vector $x^* \in X$ such that $z^* \leqq z$ for $z^* = f(x^*)$. As $z \in \text{conv}\{Y\}$, the vector z can be written as a convex combination of k vectors in Y^E. Let us denote these vectors by $z^1, \ldots, z^k$ and the weights defining the convex combination by $\lambda^1, \ldots, \lambda^k$. If $x^1, \ldots, x^k$ are the (known) vectors such that $z^i = f(x^i)$ for $i = 1, \ldots, k$, then we can define $x^* = \sum_{i=1}^k \lambda^i x^i$. Because of the convexity of X and $f(x)$, it is now easy to prove that $x^* \in X$ and $z^* \leqq z$ for $z^* = f(x^*)$.

7.3 Sandwich algorithms

7.3.1 Inner and outer approximations

In the introduction, we mentioned that we want to find an accurate approximation of the complete relevant part of the Pareto set using as few optimizations as possible. Furthermore, we would also like to be able to give quality guarantees on the accuracy of the approximation. For both aims, sandwich algorithms seem to be very suitable. The main characteristic of sandwich algorithms is that they provide two approximations that sandwich the set PS of Pareto optimal points. In the case of an MOP where all objectives must be minimized, these two approximations can be defined as follows.

Definition 7.8. *A set $IPS \subseteq Z$ is an inner approximation of PS if it satisfies $IPS \subseteq PS + \mathbb{R}^k_{\leqq}$.*

Definition 7.9. *A set $OPS \subseteq Z$ is an outer approximation of PS if it satisfies $PS \subseteq OPS + \mathbb{R}^k_{\leqq}$.*

The inner and outer approximations are sometimes called the upper and lower bounds or approximations. However, to avoid confusion with other upper and lower bounds, we choose to use the terms inner and outer approximations.

As the true Pareto set lies between the inner and outer approximations, we can use them to determine an upper bound on the approximation error. This upper bound can be used to guide the algorithm and to give guarantees. Guiding can be done by generating new points in areas of the Pareto set where the upper bound on the approximation error is relatively large. In this way, we are likely to gain more in terms of accuracy than by generating points in areas where the upper bound is much smaller. By generating new points where the potential for improvement is highest, sandwich algorithms try to accurately approximate a Pareto set through as few optimization runs as possible. Furthermore, the upper bound on the approximation error gives the decision makers the guarantee that by using the approximation, they will never select a solution of which the objective vector z is more than a certain amount worse than an objective vector $\widetilde{z}$ that is an element of the true Pareto set.

In the next three sections, we describe four different sandwich methods.

7.3.2 Algorithm of Solanki et al.

The main steps of the XNISE1 algorithm in Solanki et al. (1993) are:

1. Find all anchor points $z^{A1}, \ldots, z^{Am}$, and all points $z^{Mi} = \arg\max\{z_i \mid z \in PS\}$ for $i = 1, \ldots, k$. Initialize $IPS = \text{conv}\{z^{A1}, \ldots, z^{Am}, z^{M1}, \ldots, z^{Mm}\}$.

2. Initialize $OPS = \{z \mid z_i^{Ai} \leqq z_i \leqq z_i^{Mi}, i = 1 \ldots, k\}$.

3. Calculate the error for each facet (see below).

4. Select the facet F^* with the largest error. If this error is below a certain value, we stop. Otherwise, let $H(w, b)$ be a supporting hyperplane at F^* and go to Step 5.

5. Determine z^* by solving:

$$z^* = \quad \arg\min\{w^\top z \mid z \in Z\}.$$

6. If $w^\top z^* = b$, then no new extreme point is found. Change the error of this facet to zero, and return to Step 4 if this happens. Otherwise, go to Step 7.

7. Update IPS by replacing it with $\text{conv}\{z^*, IPS\}$.

8. Update OPS by adding the inequality $w^\top z \leq b$.

9. Return to Step 3.

In Step 3, the error of the facet is calculated by first solving the following problem:

$$w^\top \bar{z} = \quad \min\{w^\top z \mid z \in OPS\}. \tag{7.3}$$

Notice that (7.3) is a (simple) LP problem because OPS can be described by a set of linear inequalities. The error is calculated as the distance between $\bar{z}$ and the hyperplane defined by the facet. Because w satisfies $\|w\| = 1$, this distance can easily be calculated as $b - w^\top \bar{z}$.

Determining $z^{M1}, \ldots, z^{Mn}$ in Step 1 is not straightforward, because we have no explicit description of PS and there exist no suitable weight vectors to determine these points through the weighted sum method. Moreover, the maximal values of the individual objectives can be attained at one or more of the anchor points. For example, the maximum of objective z_1 can be attained by the anchor points z^{A2} or z^{A3}. Therefore, we choose to change the algorithm by leaving out $z^{M1}, \ldots, z^{Mn}$ in Step 1, and replacing z_i^{Mi} by z_i^{ub} in Step 2.

The set IPS obtained with this algorithm forms an inner approximation of PS. However, not all points of this convex hull are useful as an approximation of PS as some are (weakly) dominated by other points in IPS. Solanki et al. (1993) developed the XNISE2 algorithm to remove all points in IPS that are weakly dominated by other points in IPS. The remaining set is used as the final approximation.

As mentioned by Solanki et al. (1993), taking the weight vector w equal to the normal of a facet can be problematic when the normal contains both positive and negative

elements. In that case, minimizing the weighted sum in Step 5 could result in a non-Pareto optimal point, i.e., a point that is not part of PS. To avoid the resulting point to be too far away from PS, Solanki et al. (1993) put upper and lower bounds on the objective values in their approach. In Sections 7.3.4 and 7.4, we discuss other methods for dealing with this situation.

7.3.3 Algorithm of Klamroth et al.

The algorithm in Klamroth et al. (2002) uses two separate algorithms for generating the inner and outer approximation of the Pareto set. They combine the two algorithms by alternatingly performing one iteration of each algorithm.

The algorithm for the inner approximation also starts with a facet defined by the anchor points. For this facet, the algorithm tries to find the point in the Pareto set furthest away from this facet through the so-called gauge-method. This method determines a new point z^* by solving the following problem:

$$\begin{aligned} \gamma(z^*) \quad &= \quad \max \sum_{i=1}^{m} \lambda_i \\ \text{s.t.} \qquad & z \leqq z^{pN} + \sum_{i=1}^{m} \lambda_i (z^i - z^{pN}) \\ & \lambda_i \geq 0 \text{ for } i = 1, \ldots, m \\ & z \in Z, \end{aligned} \tag{7.4}$$

where the vectors $z^1, \ldots, z^m$ are the extreme points of the facet. The deviation of the new point z^* is subsequently calculated by $|\gamma(z^*) - 1|$. The new point with the largest deviation is added to the inner approximation, and new facets are determined by updating the convex hull. The gauge method is performed for all new facets, and again the point with the largest deviation is added. This last step is repeated until some stopping criterion is met.

This inner approximation method of Klamroth et al. (2002) is based on earlier research of Schandl et al. (2002), who formulate the gauge method slightly differently. In their formulation, the first inequality constraint in problem (7.4) is formulated as an equality constraint. We tested both methods and refer to them as Klamroth$^{\leq}$ and Klamroth$^{=}$, respectively.

Our tests with both methods showed that—in certain situations—the point with the largest deviation can be a point that is already part of the IPS. If this happens, the same point is added in every subsequent step of the algorithm, so the IPS no longer changes. To avoid this behavior, we add a tabu list to the algorithm. If applying the gauge-method for a certain facet results in an already found point, the facet is added to the tabu list and is no longer evaluated in subsequent steps of the algorithm.

The outer approximation method is formulated as follows. First the utopia and pseudo-nadir point are determined. The hypercube defined by the utopia and pseudo-nadir point is taken as the initial outer approximation. The outer approximation is thus a convex polytope. We call the extreme points of this polytope the fundamental vectors and denote them by $v^1, \ldots, v^m$. Next, for every fundamental vector $v^i \neq z^{pN}$, we solve the problem

$$\begin{aligned} \delta^i &= \max \lambda \\ \text{s.t.} \quad & f(x) - (z^{pN} + \lambda(v^i - z^{pN})) \leqq 0 \\ & \lambda \geq 0 \\ & x \in X, \end{aligned} \tag{7.5}$$

and denote the optimal Lagrange multiplier of the first constraint by u^i and the optimal value of $f(x)$ by z^i. The fundamental vector with the minimal value of δ^i is now used to update the outer approximation. If m is the index of this vector, then the outer approximation is thus updated by adding the inequality $(u^m)^\top z \geq (u^m)^\top z^m$. After determining the fundamental vectors of this updated polytope, we solve problem (7.5) for all new vectors and again update the outer approximation using the vector with the largest δ^i. We repeat solving problem (7.5) and updating the outer approximation, until some stopping criterion is satisfied.

7.3.4 Algorithm of Craft et al.

Craft et al. (2006) introduce the PGEN algorithm to approximate a convex Pareto set. The algorithm is similar to the algorithm of Solanki et al. (1993), except for two important characteristics.

The first characteristic is the way the error is calculated. Instead of solving an LP problem, the error is calculated by looking at the hyperplanes of the outer approximation going through the corner points of a facet. If these planes intersect in a point, this point is called the lower distal point. The error is defined as the distance between the facet and its lower distal point.

Secondly, the algorithm differs in the way it deals with facets that have a normal with both positive and negative elements. Whereas Solanki et al. (1993) choose to put upper and lower bounds on the objective values, the PGEN algorithm deals with this problem by using a different weight vector. Instead of using the normal of the facet, the weight vector is determined by taking a linear combination of the weight vectors used to obtain the corner points of the facet.

7.4 Adding dummy points to IPS

7.4.1 Motivation of dummy points

As we mentioned in the previous section, the weighted sum method used in the algorithms of Solanki et al. (1993) and Craft et al. (2006), can give non-Pareto points if the weight vector has both positive and negative components. Unfortunately, facets of the convex hull may have normals with this property. Using these 'undesirable' normals as weight vectors may result in non-Pareto points. Solanki et al. (1993) and Craft et al. (2006) developed different ways of dealing with this problem, as we described in Section 7.3.4. The drawback of the approach of Solanki et al. (1993) is that it may still produce non-Pareto points, although there is a limit on how far the non-Pareto point is from PS. The approach of Craft et al. (2006) does not have this drawback because it always uses a non-negative weight vector. However, tests with this algorithm show that sometimes the same facet with 'undesirable' normal is selected in subsequent iterations. This implies that solving the weighted sum problem for the alternative weight vector does not always produce a point that reduces the error measure used by Craft et al. (2006). As we assume that every optimization is CPU-time consuming, this is not a desirable property.

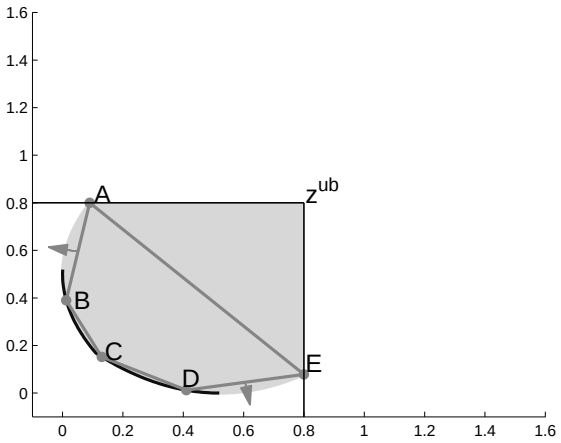


Figure 7.1: Example of facets with 'undesirable' normals.

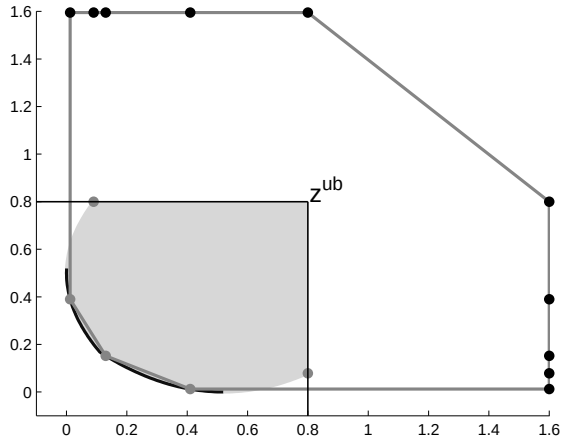


Figure 7.2: Example of avoiding facets with 'undesirable' normals by adding dummy points.

In this section, we introduce a new way of dealing with the problem of 'undesirable' normals. By taking the convex hull of IPS and a set of dummy points, we ensure that all relevant facets have a 'desirable' inner normal with only non-negative elements. Besides solving the problem of obtaining non-Pareto points, this approach has a number of other benefits. In Section 7.4.3, we show that the dummy points help us to determine the IPS points that are not dominated by other IPS points. Furthermore, we introduce an error measure in Section 7.5 that has a number of desirable properties, including the property

that calculating an upper bound for this measure based on IPS and OPS can be done by solving a number of simple LP problems when using dummy points.

To explain the general idea behind these dummy points and their advantages, we use the bi-criteria example in Figure 7.1. Although the problem with 'undesirable' normals does not occur for bi-criteria problems, we use a bi-criteria example to simplify the explanation of the general idea behind the dummy points. In Figure 7.1, the shaded area represents Z and the points A, B, C, D, and E are the current extreme points of IPS. As the arrows indicate, facets AB and DE both have 'undesirable' normals. Using these normals in the weighted sum method means that we search in the direction of the arrows for the point farthest away from the facet. In both cases, this results in a non-Pareto solution. In Figure 7.2, two dummy points are added for each extreme point $z \in IPS^E$. These dummy points are created by replacing one of the two coordinates of z by a large value (an exact definition of the dummy points is given in Section 7.4.2). Once the dummy points are created, the set IPS is replaced by the convex hull of IPS and all its dummy points. All facets containing at least one IPS-point now have a normal with non-negative elements only. All other facets, which contain only dummy points, do not satisfy the upper bound constraint on the objectives, so they are irrelevant.

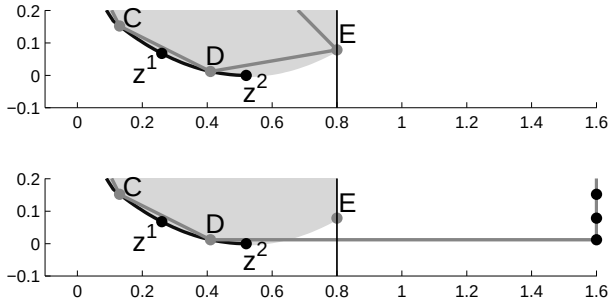


Figure 7.3: Example of improved error calculation with dummy points.

Besides solving the problem of 'undesirable' normals, adding dummy points helps us to determine the approximation error. To explain this property, we use the points z^1 and z^2 in Figure 7.3. When the error in a certain point is calculated by the Euclidean distance to the approximation, point z^1 will be considered more accurately approximated than point z^2. Point D, however, approximates z^2 quite closely as it is only slightly worse in objective 2 and better in objective 1. The distance between z^2 and the facet between D and the corresponding dummy point is therefore a better error measure. Now point z^2 has a slightly better error value than z^1, which seems a much better representation of the approximation accuracy. So, adding dummy points to the approximation also improves the usefulness of the Euclidean distance as an error measure. In Section 7.5, we introduce $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$ to measure the approximation error and show how we can easily calculate this measure using the dummy points.

Finally, the IPS obtained by any of the sandwich algorithms may contain points that are weakly or strongly dominated by other IPS points. As it is not optimal for the decision maker to choose any of these points, we want to detect and remove these points before presenting IPS to the decision maker. As we mentioned before, Solanki et al. (1993) presented the XNISE2 algorithm for this purpose. Using dummy points, however, we can use an easier approach for finding these points. We introduce and discuss this new approach in Section 7.4.3.

7.4.2 Effect of dummy points on inner normals

To prove that dummy points also work in higher dimensions, we first give a formal definition of the dummy points. For a fixed and small $\theta > 0$ and for each extreme point $z \in IPS^E$, the dummy points $d^1(z), \ldots, d^k(z)$ are defined in the following way.

Definition 7.10. *Let $\theta > 0$. The dummy point $d^i(z)$ generated from $z \in Z$ is given by*

$$d^i_j(z) = \begin{cases} z_j & \text{if} \quad j \neq i \\ kz^{ub}_j + \theta & \text{if} \quad j = i \end{cases} \quad \text{for } j = 1, \ldots, k.$$

Definition 7.11. *The set D is defined as*

$$D = \{d^i(z) \mid z \in IPS^E, i = 1, \ldots, k\}.$$

Note that all dummy points are weakly dominated by the points they are generated from. They thus never dominate any of the Pareto optimal points in IPS. Furthermore, in many instances, dummy points can be written as a convex combination of other dummy points. In the example of Section 7.4.1, this is true for six of the ten generated dummy points. The points for which this happens can be easily detected. As we always use D to determine $\text{conv}\{IPS, D\}$, these points can be removed from D, which reduces the computation costs of determining $\text{conv}\{IPS, D\}$. Lastly, because the dummy points are not smaller or equal to z^{ub}, certain facets of $\text{conv}\{IPS, D\}$ may contain no points $\widetilde{z}$ for which $\widetilde{z} \leqq z^{ub}$. As all points $z \in PS$ do satisfy $z \leqq z^{ub}$, these facets are not relevant when approximating PS. In the example in the previous section, we thus have only four relevant facets. To distinguish between relevant and irrelevant facets, we introduce the following formal definitions.

Definition 7.12. *A facet F of conv$\{IPS, D\}$ is a relevant facet if at least one of its extreme points is in IPS. The set of all relevant facets is denoted by RF.*

Definition 7.13. *A hyperplane $H(w, b)$ is a relevant hyperplane if it supports the set conv$\{IPS, D\}$ at a relevant facet. The set of all relevant hyperplanes is denoted by RH.*

As we assume that $z \leqq z^{ub}$ for every $z \in IPS$, all relevant facets contain points that are smaller or equal than z^{ub} and can thus approximate PS. The following lemma shows that all facets of conv$\{IPS, D\}$ not satisfying the above definition are not relevant.

Lemma 7.7. *Consider a facet F of conv$\{IPS, D\}$. If F is not a relevant facet, then there is no point z on F with $z \leqq z^{ub}$.*

Proof. As F is not a relevant facet, all extreme points must be dummy points. Furthermore, any point z on F can be written as $z = \sum_{i=1}^{k} \lambda_i z^i$ with $z^1, \ldots, z^k$ being extreme points of F, $\sum_{i=1}^{k} \lambda_i = 1$ and $\lambda_i \geq 0$ for $i = 1 \ldots, k$. Assume without loss of generality that $\lambda_1 = \max_i \lambda_i$. From $\sum_{i=1}^{k} \lambda_i = 1$ and $\lambda_i \geq 0$ for $i = 1 \ldots, k$, it follows that $\lambda_1 \geq \frac{1}{k}$. As $z^i \geq 0$ for all $i = 1, \ldots, k$, this gives the following inequality:

$$z = \lambda_1 z^1 + \sum_{i=2}^{k} \lambda_i z^i \geq \lambda_1 z^1 \geq \frac{1}{k} z^1. \tag{7.6}$$

As z^1 is a dummy point, $z_j^1 = k z_j^{ub} + \theta$ must hold for one element j. Combining this property with inequality (7.6) gives

$$z_j \geq \frac{1}{k} z_j^1 = \frac{1}{k}(k z_j^{ub} + \theta) > z_j^{ub}.$$

This shows that there is no point on F for which $z \leqq z^{ub}$. □

In Section 7.4.1, we have already mentioned that by using conv$\{IPS, D\}$ instead of IPS, we want to ensure that all relevant facets have an inner normal with only non-negative elements. Lemma 7.8 shows that for the above defined relevant facets and set D this indeed holds.

Lemma 7.8. *Consider conv$\{IPS, D\}$ and $F \in RF$. If $H(w, b) \in RH$ is a supporting hyperplane at F, then $w \geq 0$.*

Proof. Let z be an extreme point of F that is in IPS. Suppose by contradiction that there is an i such that $w_i < 0$. Then for the dummy point $d^i(z)$ it holds that

$$w^\top d^i(z) = w^\top z + w_i(k z_i^{ub} + \theta - z_i) < w^\top z = b.$$

This inequality is a contradiction, because $d^i(z)$ is in conv$\{IPS, D\}$, which has $H(w, b)$ as a supporting hyperplane. □

7.4.3 Determining non-IPS-dominated points of IPS

When we have determined an inner approximation IPS of PS, some points in IPS may be dominated by other points in IPS. For a decision maker it is not optimal to choose

one of these dominated points. Therefore, we want to determine all points in IPS that are not strongly or weakly dominated by other points in IPS. We refer to these points as non-IPS-dominated points and denote the set of all non-IPS-dominated points by IPS^{nId}. The set $\text{conv}\{IPS, D\}$ and the following lemma and proposition can be used to determine IPS^{nId}.

Lemma 7.9. *Consider $z \in conv\{IPS, D\}$ with $z \notin IPS$. Then z is weakly dominated by a point in IPS.*

Proof. Let us denote all points in IPS^E by $z^1, \ldots, z^m$. As $z \in \text{conv}\{IPS, D\}$, we know that z can be written as $z = \sum_i \lambda_i z^i + \sum_{i,j} \mu_{i,j} d^j(z^i)$ with $\sum_i \lambda_i + \sum_{i,j} \mu_{i,j} = 1$, $\lambda_i \geq 0$ and $\mu_{i,j} \geq 0$ for $i = 1, \ldots, m$, $j = 1, \ldots, k$. Furthermore, because $z \notin IPS$, we know that $\mu_{i,j} > 0$ must hold for at least one combination of i and j. By definition, each dummy point $d^j(z^i) \in D$ is weakly dominated by the point $z^i \in IPS^E$ from which it is generated. Define $z^* = \sum_i \lambda_i z^i + \sum_{i,j} \mu_{i,j} z^i$ as the point obtained by replacing each dummy point $d^j(z^i)$ with z^i in the convex combination that formed z. Then this point z^* is an element of IPS and weakly dominates z. □

The above lemma thus shows that we should only consider points $z \in IPS$, even when using $z \in \text{conv}\{IPS, D\}$ as an inner approximation.

Proposition 7.18. *Consider $z^{IPS} \in IPS$. Then z^{IPS} is not strongly dominated by another point in IPS if and only if z^{IPS} is on a relevant facet of conv$\{IPS, D\}$.*

Proof. Assume that z^{IPS} is not strongly dominated by another point in IPS. This implies that there exists no $\widehat{z} \in \text{conv}\{IPS, D\}$ such that $\widehat{z} < z^{IPS}$ and that z^{IPS} should thus be part of a facet F of $\text{conv}\{IPS, D\}$. As $z^{IPS} \in IPS$, we know that $z^{IPS} \leqq z^{ub}$ must hold and—according to Lemma 7.7—facet F should be a relevant facet of $\text{conv}\{IPS, D\}$.

Let us now assume that z^{IPS} is on a relevant facet F of $\text{conv}\{IPS, D\}$ and, by contradiction, that there exists a point $\widetilde{z} \in IPS$ such that $\widetilde{z}$ strongly dominates z^{IPS}. If $H(w, b)$ is a supporting hyperplane at F, then all points $z \in \text{conv}\{IPS, D\}$ must satisfy $w^\top z \geq b$. Furthermore, because of Lemma 7.8, $w^\top z^{IPS} = b$ and $w \geq 0$. So

$$w^\top \widetilde{z} < w^\top z^{IPS} = b.$$

This inequality shows that $\widetilde{z}$ cannot be an element of IPS. By contradiction, we have thus proven that z^{IPS} cannot be strongly dominated by another point in IPS. □

Taking the points on the relevant facets of $\text{conv}\{IPS, D\}$ that are also in IPS, we obtain a set of points in IPS that are not strongly dominated by any other point in IPS. The added dummy points thus make it quite easy to detect and discard the strongly dominated points. However, the set can still contain points that are weakly dominated by a point

in IPS. Consider, for instance, the points $z^1 = (1,0,1)^\top$, $z^2 = (0,1,1)^\top$ and $z^3 = (0.5,0.5,0)^\top$. For an MOP with three objectives, these points could be elements of IPS and be on a facet of $\text{conv}\{IPS, D\}$. However, the point $\widetilde{z} = 0.5z^1 + 0.5z^2 = (0.5,0.5,1)^\top$ then also has these properties, but is weakly dominated by z^3. Therefore, we need a different criterion to discard the weakly dominated points. The following proposition provides such a criterion.

Proposition 7.19. *Consider $z^{IPS} \in IPS$. Let W be the set of inner unit normals of the relevant facets of $\text{conv}\{IPS, D\}$ containing z^{IPS} and let w^+ be the sum of all $w \in W$. Then z^{IPS} is not weakly dominated by a point in $\text{conv}\{IPS, D\}$ if and only if $w^+ > 0$.*

Proof. Assume $w^+ > 0$. This implies that for each dimension $d = 1, \ldots, k$, there exists an inner normal $w^d \in W$ for which it holds $w^d_d > 0$. Let $H(w^d, b^d)$ be the corresponding supporting hyperplane of the facet, so that, by definition, $w^{d\top} z \geq b^d$ for $z \in \text{conv}\{IPS, D\}$ and $w^{d\top} z^{IPS} = b^d$. As w^d is an inner normal of a relevant facet, Lemma 7.8 implies $w^d \geq 0$. If $\widehat{z}$ is a point that weakly dominates z^{IPS}, then there is a dimension d such that $\widehat{z}_d < z^{IPS}_d$. Because $w^d \geq 0$ and $w^d_d > 0$, it follows that $w^{d\top}\widehat{z} < w^{d\top} z^{IPS} = b^d$, so $\widehat{z}$ is not an element of $\text{conv}\{IPS, D\}$. Since $w^+ > 0$, there thus exists no $\widehat{z}$ in $\text{conv}\{IPS, D\}$ that weakly dominates z^{IPS}.

Let us now assume that z^{IPS} is not weakly dominated by a point in $\text{conv}\{IPS, D\}$. This implies that any point $\widetilde{z}$ satisfying $\widetilde{z}_d < z^{IPS}_d$ and $\widetilde{z}_i = z^{IPS}_i$ for $i \neq d$ is not in $\text{conv}\{IPS, D\}$. As this result holds for any such point $\widetilde{z}$, there must exist a supporting hyperplane $H(w^d, b^d)$ of $\text{conv}\{IPS, D\}$ such that $w^{d\top} z^{IPS} = b^d$ and $w^{d\top}\widetilde{z} < b^d$. Because $\widetilde{z}_d - z^{IPS}_d < 0$ and

$$w^d_d(\widetilde{z}_d - z^{IPS}_d) = w^{d\top}(\widetilde{z} - z^{IPS}) < b^d - b^d = 0,$$

it follows that $w^d_d > 0$. The vector w^d must be in W as only relevant facets can contain points $z^{IPS} \leqq z^{ub}$. Furthermore, $w \geq 0$ for all $w \in W$ and $w^d_d > 0$ implies that w^+_d must also be greater than zero. This result holds for every dimension $d = 1, \ldots, k$, so $w^+ > 0$. □

To determine IPS^{nId}, we can check all points on relevant facets of IPS with the criterion in Proposition 7.19. However, as the relevant facets of IPS contain infinitely many points, determining IPS^{nId} requires a more efficient method. The set W used in Proposition 7.19, however, is the same for all points on a face of IPS. This implies that we can check all points on a face simultaneously. If IPS is the convex hull of a finite number of points, the number of faces of IPS is also finite. As this is true for all previously discussed sandwich algorithms, we can use the following algorithm, inspired by XNISE2 in Solanki et al. (1993), to determine IPS^{nId}:

1. Set $d = k$ and $IPS^{nId} = \emptyset$.

2. Denote by P^d the set of all $(d-1)$-faces of $\text{conv}\{IPS, D\}$ having d extreme points in IPS.

3. For each face in P^d, determine if it is a subset of a multi-dimensional face in IPS^{nId}. If so, remove the face from P^d.

4. For each remaining face in P^d, calculate the vector w^+ by taking the sum of the inner unit normals of the facets of which this $(d-1)$-face is a subset. If $w^+ > 0$, add the face to IPS^{nId}.

5. Set $d = d - 1$.

6. If $d \geq 1$, return to Step 2. Otherwise, stop.

To illustrate the above algorithm, we use the example in Figure 7.4. When we take $IPS = \text{conv}\{z^1, z^2, z^3, z^4\}$ and $z^{ub} = [1\ 1\ 1]^\top$, the figure shows all relevant facets of $\text{conv}\{IPS, D\}$. The Pareto optimal points are on the facet with extreme points z^1, z^2, and z^3, and on the edge with extreme points z^1 and z^4. In the algorithm, we look at the 2-, 1- and 0-faces, which in three dimensions correspond to facets, edges, and extreme points, respectively.

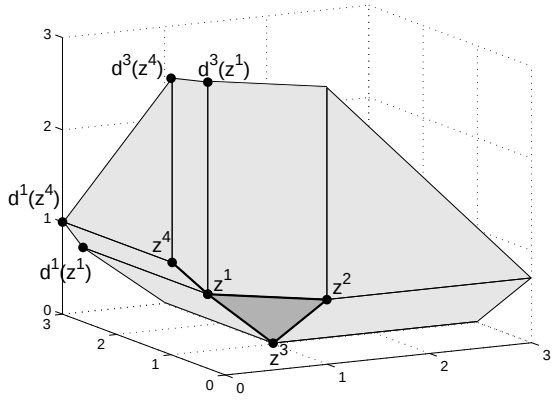


Figure 7.4: Example of determining non-IPS-dominated points in IPS.

We start by looking at all facets in the set P^3, which in this example contains only the facet with extreme points z^1, z^2, and z^3. As the inner normal w of this facet satisfies $w > 0$, we have $w^+ > 0$ and the facet is added to IPS^{nId} in Step 4. Next, we take $d = 2$ and continue with the set P^2 containing all edges of $\text{conv}\{IPS, D\}$ connecting two extreme points of IPS. This implies that $P^2 = \{(z^1, z^2), (z^1, z^3), (z^2, z^3), (z^1, z^4)\}$. As the first three edges are part of the facet which we have already added to IPS^{nId}, they are removed

in Step 3. For the remaining edge (z^1, z^4), we determine vector w^+ by summing the inner normals of the facets defined by $\{z^1, z^4, d^1(z^4), d^1(z^1)\}$ and $\{z^1, z^4, d^3(z^4), d^3(z^1)\}$. The first facet has an inner unit normal w^1 with $w_1^1 = 0$ and $w_2^1, w_3^1 > 0$. For the inner unit normal w^2 of the second facet it holds that $w_3^2 = 0$ and $w_1^2, w_2^2 > 0$. For all points on the edge (z^1, z^4), the vector w^+ is equal to $w^1 + w^2 > 0$. The edge (z^1, z^4) should thus be added to the set IPS^{nId} according to Proposition 7.19, and this is exactly what the algorithm does. Finally, we look at all points in $P^1 = \{z^1, z^2, z^3, z^4\}$. As all these point are already in IPS^{nId}, the set P^1 becomes empty in Step 3 and the algorithm stops.

7.5 Error measure

7.5.1 Motivation and definition of $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$

To assess the quality of outer and inner approximations, we need a measure to quantify the accuracy of the approximation. There exist many different measures and methods that can be used to determine the quality of an approximation. Carlyle et al. (2003) and Zitzler et al. (2003) provide two extensive reviews and analyses of different comparison methods. In most sandwich algorithms, the accuracy is measured by the distance between the inner approximation and the true Pareto set. However, there are differences in the exact definition of this distance and in the way it is calculated.

To decide which definition to use, we first discuss the definition of the accuracy of an approximation. Recall that we assume that the user is interested in the complete Pareto set. Therefore, we could call an approximation accurate if each point of the Pareto set is represented accurately by a point on the approximation. Note that this definition is essentially different from measuring if each point on the approximation accurately represents a point of the Pareto set. Consider, for instance, an approximation that consists of only one point that is an element of the Pareto set PS. According to the second definition, this would be a very accurate approximation as all points of the approximation are in PS. However, using the first definition, the approximation is probably not very accurate as this one point is not likely to be an accurate approximation of all points in PS. Consider, on the other hand, an approximation that consists of all points in PS plus a point that does not accurately represent a point in PS. Now the approximation is very accurate according to the first definition but not according to the second. As these examples illustrate, we could say that the first definition determines if an approximation is not too small, whereas the second definition determines if an approximation is not too large. Ideally, a measure should indicate when an approximation is accurate according to both definitions.

When using these definitions, we still have to specify when a point in PS is accurately represented and when a point of the approximation is an accurate representation of a

point in PS. To decide on these definitions, we also take into account the final user of the approximation. In our opinion, the measure should not only measure the accuracy in an adequate way but it should also be easy to understand by the user. The second aspect implies that the interpretation of the measure should be easy to explain. This is not only important because it helps the user to accept the approximation; the user must also be able to specify a desired accuracy. We assume that most users prefer to specify their desired accuracy in terms of a maximum percentage or absolute amount of allowed inaccuracy per objective.

Taking this assumption into account, ϵ-dominance seems to be a suitable way to measure whether a point in the Pareto set is accurately represented. This measure has two different variants: additive and multiplicative. The additive variant was simultaneously introduced by Evtushenko and Potapov (1987) and Reuter (1990); the multiplicative variant was introduced by Ruhe and Fruhwirth (1990). The additive ϵ-dominance is defined as follows.

Definition 7.14. *Let $\epsilon \in \mathbb{R}^k_{\leqq}$. A point z is ϵ-dominated by a point $\widehat{z}$ if:*

$$\widehat{z} \leqq z + \epsilon.$$

This definition implies that a point is ϵ-dominated if there exists a point that is at most ϵ_i worse in each objective $i = 1, \ldots, k$. For multiplicative ϵ-dominance, $z + \epsilon$ is replaced by $z(1 + \epsilon)$, which implies that there must exist a point that is at most a factor ϵ_i worse in each objective $i = 1, \ldots, k$. So, these variants impose an upper bound on either the absolute error or the relative error per objective. Which variant to use, depends mainly on the kind of guarantee the user prefers. Note, however, that using the multiplicative variant may be difficult if objective values can get close or equal to zero. In this chapter, we therefore use the additive variant.

When all points in PS are ϵ-dominated by points in IPS, we call IPS an ϵ-approximation. If IPS is not an ϵ-approximation, we wish to know how far off it is from an ϵ-approximation. We can measure this by determining the smallest multiple of ϵ for which PS is ϵ-dominated. Therefore, we introduce the following definitions.

Definition 7.15. *For a fixed $\epsilon \in \mathbb{R}^k_{\leqq}$, the error measure $\alpha(z, T)$ is the minimal α for which point z is $\alpha\epsilon$-dominated by a point in the set T. For the set S, the measure $\alpha(S, T)$ is defined as $max_{z \in S}\alpha(z, T)$.*

The value of $\alpha(PS, IPS)$ is thus a measure of how close the approximation IPS is to being an ϵ-approximation of PS. In Section 7.8, we use the value of α to compare different IPSs.

When we do not know PS as is usually the case, in practice, we can still use the inner and outer approximation of PS to determine upper bounds on the above accuracy

measure. If all points in OPS are ϵ-dominated by points in IPS, then all points in PS must also be ϵ-dominated by points in IPS. Therefore, $\alpha(OPS, IPS)$ is an upper bound for $\alpha(PS, IPS)$. Furthermore, note that $\alpha(z, IPS) = \alpha(z, IPS^{nId})$, which implies that IPS^{nId} is an ϵ-approximation of PS if and only if IPS is an ϵ-approximation of PS.

To determine whether a point of the approximation is an accurate approximation of a point in PS, we use a concept similar to ϵ-dominance, called ε-Pareto optimality or ε-efficiency. This concept was introduced by Loridan (1984), and is defined as follows.

Definition 7.16. *Let $\varepsilon \in \mathbb{R}^k_{\geqq}$. A point $z \in Z$ is ε-Pareto optimal if there is no $\widehat{z} \in Z$ such that*

$$\widehat{z} \leq z - \varepsilon.$$

This definition implies that a point is ε-Pareto optimal if there exists no point that is more than ε_i better in each objective ($i = 1, \ldots, k$).

To satisfy both definitions of accuracy discussed at the beginning of this section, we wish to find approximations IPS and OPS such that each point of PS is ϵ-dominated by a point of IPS^{nId} and each point of IPS^{nId} is ε-Pareto optimal. As we have illustrated in the examples at the beginning of this section, an approximation that satisfies the first criterion does not necessarily satisfy the second criterion. However, if $\epsilon = \varepsilon$, we can show that all points of IPS^{nId} are ε-Pareto optimal when IPS^{nId} is an ϵ-approximation. We prove this result in the following proposition.

Proposition 7.20. *Let $\epsilon \in \mathbb{R}^k_{\geqq}$ and let IPS^{nId} be an ϵ-approximation of PS. Then all points $z \in IPS^{nId}$ are ϵ-Pareto optimal.*

Proof. Assume, by contradiction, that there exists a point $\widehat{z} \in PS$ such that $\widehat{z} \leq z - \epsilon$. As IPS^{nId} is an ϵ-approximation of PS, there must exist a $\widetilde{z} \in IPS^{nId}$ such that $\widetilde{z} \leqq \widehat{z} + \epsilon$. Combining these inequalities gives $\widetilde{z} \leq z$. As $\widetilde{z}$ and z are both elements of IPS, this implies that z cannot be an element of IPS^{nId} as it is (weakly) dominated by $\widetilde{z} \in IPS$. This contradiction shows that there exists no point $\widehat{z} \in PS$ such that $\widehat{z} \leq z - \epsilon$ and hence that z is ϵ-Pareto optimal. □

7.5.2 Calculating $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$

In this section, we introduce a method to calculate $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$ for a fixed $\epsilon \in \mathbb{R}^k_{>}$ when $|RH|$ is finite. For all IPSs obtained by the discussed sandwich algorithms, this last condition is satisfied as IPS is the convex hull of a finite number of points and thus has a finite number of facets. One main advantage of the method introduced in this section is that $\alpha(OPS, IPS)$ can be calculated by solving a number of LP problems. These LP problems are the same as the ones used in the algorithm of

Solanki et al. (1993) to determine the error of a facet. As the proofs and reasoning for $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$ are similar, we give them only for the first.

The general idea behind the method is the following. We first prove that for each $z \in PS$ and $H(w,b) \in RH$ it holds that $\alpha(z, IPS) \geq \alpha(z, H(w,b))$. Furthermore, we show that for each $z \in PS$, equality in the above expression holds for at least one hyperplane in RH, which implies that

$$\alpha(z, IPS) = \max_{H(w,b) \in RH} \alpha(z, H(w,b)). \tag{7.7}$$

Instead of solving problem (7.7) for every $z \in PS$, we determine $\max_{z \in PS} \alpha(z, H(w,b))$ for every $H(w,b) \in RH$. Thus, we have to solve $|RH|$ weighted sum problems. Using

$$\begin{aligned} \alpha(PS, IPS) = \max_{z \in PS} \alpha(z, IPS) &= \max_{z \in PS} \max_{H(w,b) \in RH} \alpha(z, H(w,b)) \\ &= \max_{H(w,b) \in RH} \max_{z \in PS} \alpha(z, H(w,b)), \end{aligned} \tag{7.8}$$

we can determine $\alpha(PS, IPS)$ by taking the maximum over all solutions of these weighted sum problems. As we mentioned before, the weighted sum problems become LP problems when calculating $\alpha(OPS, IPS)$.

To prove the various steps in the above algorithm, we first need to prove the following lemma.

Lemma 7.10. *Let $\epsilon \in \mathbb{R}^k_>$ and $H(w,b) \in RH$. For every $z \in PS$, we have that $\alpha(z, H(w,b))$ is equal to the unique $\widehat{\alpha}$ for which the point $z + \widehat{\alpha}\epsilon$ is on the hyperplane $H(w,b)$.*

Proof. The existence and uniqueness of $\widehat{\alpha}$ such that $z + \widehat{\alpha}\epsilon$ is on $H(w,b)$ follows easily from the fact that $w^\top \epsilon \neq 0$ because $w \geq 0$ and $\epsilon > 0$. As z is $\alpha\epsilon$-dominated by $z + \widehat{\alpha}\epsilon \in H(w,b)$, we know that $\alpha(z, H(w,b)) \leq \widehat{\alpha}$. If $\alpha(z, H(w,b)) < \widehat{\alpha}$, there must exist a $\widehat{z} \in H(w,b)$ such that $\widehat{z} < z + \widehat{\alpha}\epsilon$. However, this would imply:

$$b = w^\top \widehat{z} < w^\top (z + \widehat{\alpha}\epsilon) = b.$$

Because of this contradiction, we conclude that $\alpha(z, H(w,b)) = \widehat{\alpha}$. □

Using the results of Lemmas 7.7, 7.8, and 7.10, we can prove the following two lemmas.

Lemma 7.11. *Consider $\epsilon \in \mathbb{R}^k_>$, $z \in PS$ and $H(w,b) \in RH$. Then $\alpha(z, IPS) \geq \alpha(z, H(w,b))$.*

Proof. Lemma 7.8 implies $w \geq 0$. Assume by contradiction that z is $\widehat{\alpha}\epsilon$-dominated by IPS with $\widehat{\alpha} < \alpha(z, H(w,b))$. This implies that there should exist a point $\widehat{z} \in IPS$ such that $\widehat{z} \leqq z + \widehat{\alpha}\epsilon$. For this point $\widehat{z}$ it holds that

$$w^\top \widehat{z} \leqq w^\top z + \widehat{\alpha}(w^\top \epsilon) < w^\top z + \alpha(z, H(w,b))(w^\top \epsilon) = w^\top (z + \alpha(z, H(w,b))\epsilon) = b,$$

where the last equality follows from Lemma 7.10. As $H(w,b)$ is a supporting hyperplane of a relevant facet of $\text{conv}\{IPS, D\}$, for all points in IPS it must hold that $w^\top z \geq b$. This implies that $\widehat{z}$ cannot be an element of IPS. Therefore, z can only be $\alpha\epsilon$-dominated by IPS with $\alpha \geq \alpha(z, H(w,b))$. □

Lemma 7.12. *Consider $\epsilon \in \mathbb{R}^k_>$, $z \in PS$. Then $\alpha(z, IPS) = \alpha(z, H(w,b))$ must hold for at least one $H(w,b) \in RH$.*

Proof. Define $z^\alpha = z + \alpha(z, IPS)\epsilon$ and let $z^{IPS} \in IPS$ be such that $\alpha(z, \{z^{IPS}\}) = \alpha(z, IPS)$. By the definition of $\alpha(z, \cdot)$, we know that $z^{IPS} \leqq z^\alpha$ where the equality holds for at least one coordinate. Furthermore, there cannot exist a point $\widehat{z} \in IPS$ such that $\widehat{z} < z^{IPS}$ or $\widehat{z} < z^\alpha$. Because all points in D are weakly dominated by points in IPS, the same is true for all points $\widehat{z}$ in $\text{conv}\{IPS, D\}$. This implies that z^{IPS} must lie on one or more facets of $\text{conv}\{IPS, D\}$. Let F be one of these facets. Because $z^{IPS} \leqq z^{ub}$ and $z^{IPS} \in F$, F must be a relevant facet of $\text{conv}\{IPS, D\}$ according to Lemma 7.7. Let $H(\widetilde{w}, \widetilde{b}) \in RH$ be a supporting hyperplane of F. For this hyperplane it holds that $z^{IPS} \in F \subset H(\widetilde{w}, \widetilde{b})$. Using this property, we can show the following relation:

$$\alpha(z, H(\widetilde{w}, \widetilde{b})) \leq \alpha(z, \{z^{IPS}\}) = \alpha(z, IPS).$$

The inequality follows from the definition of $\alpha(z, \cdot)$ and the fact that $z^{IPS} \in H(\widetilde{w}, \widetilde{b})$. The equality follows from the definition of z^{IPS}. Combining the above equation with Lemma 7.11 gives that $\alpha(z, IPS) = \alpha(z, H(\widetilde{w}, \widetilde{b}))$ must hold for $H(\widetilde{w}, \widetilde{b}) \in RH$. □

Combining Lemmas 7.11 and 7.12, we obtain equation (7.7), which gives an expression for $\alpha(z, IPS)$. To determine $\alpha(PS, IPS)$, we must take the maximum of $\alpha(z, IPS)$ over all $z \in PS$ by definition. However, determining $\alpha(z, IPS)$ explicitly for every $z \in PS$ is not an option as there are an infinite number of vectors in the set PS. The number of $H(w,b) \in RH$, on the other hand, is finite. Therefore, we decide to calculate $\max_{z \in PS} \alpha(z, H(w,b))$ for every $H(w,b) \in RH$ and use these results to calculate $\alpha(PS, IPS)$. Another difficulty is that an explicit description of the set PS is often not known. However, the set Z can be described using the objectives and constraints of the MOP. The following proposition shows that determining $\max_{z \in PS} \alpha(z, H(w,b))$ can be done by solving a weighted sum problem over the set Z.

Proposition 7.21. *Consider a hyperplane $H(w,b) \in RH$, a vector $\epsilon \in \mathbb{R}^k_>$ and the following two optimization problems:*

$$\begin{aligned} \beta^* = \min_z \quad & w^\top z \\ \text{s.t.} \quad & z \in Z, \end{aligned} \tag{7.9}$$

and

$$\begin{aligned} \alpha^* = \max_{\alpha,z} \quad & \alpha \\ \text{s.t.} \quad & z \in PS \\ & w^\top(z + \alpha\epsilon) = b. \end{aligned} \tag{7.10}$$

The values α^* and β^* are related by $\alpha^* = \frac{b-\beta^*}{w^\top \epsilon}$ and α^* is equal to $\max_{z \in PS} \alpha(z, H(w,b))$.

Proof. Because $H(w,b) \in RH$, $w \geq 0$ according to Lemma 7.8. Using this property, we can easily show that problem (7.9) always has at least one solution $\widehat{z}$ such that $\widehat{z} \in PS$. This implies that replacing the constraint $z \in Z$ by $z \in PS$ gives the same value of β^*. Furthermore, the proof of Lemma 7.10 implies that for every $z \in PS$ there exists a unique α that satisfies $w^\top(z + \alpha\epsilon) = b$. Using these two properties, we can change the first problem into the following problem that gives the same value of β^*:

$$\begin{aligned} \beta^* = \min_{\alpha,z} \quad & w^\top z \\ \text{s.t.} \quad & z \in PS \\ & w^\top(z + \alpha\epsilon) = b. \end{aligned}$$

Because $w^\top \epsilon > 0$, the last constraint can also be written as $\alpha = \frac{b - w^\top z}{w^\top \epsilon}$. This relation implies that a vector $\widehat{z} \in PS$ that maximizes $w^\top z$ also maximizes α. We thus obtain the relation $\alpha^* = \frac{b-\beta^*}{w^\top \epsilon}$ between α^* and β^*.

Lastly, for every combination of z and α that satisfies the two constraints of problem (7.10) it holds that α is equal to $\alpha(z, H(w,b))$, according to Lemma 7.10. Therefore, α^* is equal to $\max_{z \in PS} \alpha(z, H(w,b))$. □

Solving the above problem for every $H(w,b) \in RH$, we can determine the value of $\max_{z \in PS} \alpha(z, H(w,b))$ explicitly for every $H(w,b) \in RH$. Equation (7.8) shows that taking the maximum over all these solutions gives $\alpha(PS, IPS)$.

By replacing Z and PS by OPS in the above equations and lemmas, we derive a similar method for determining $\alpha(OPS, IPS)$. Notice, that problems (7.3) and (7.9) then become the same. Calculating $\alpha(OPS, IPS)$ thus requires no more effort than calculating the error measure of Solanki et al. (1993). However, the advantage of using $\alpha(OPS, IPS)$ is that we can give an interpretation of this measure in terms of ϵ-dominance. Therefore, we expect that this measure is easier to understand by the user than the measures used in the other sandwich algorithms.

7.6 Transformations

7.6.1 Notation

Up to now, we have assumed that all objective functions f_i, $i = 1, \ldots, k$, are convex. In practice, however, there are many multi-objective optimization problems for which one or more objectives are not convex. For bi-criteria problems, Siem et al. (2008) have shown that, under certain conditions, it is possible to transform non-convex objective functions in such a way that an IPS and OPS can be determined for a non-convex set PS. Moreover, they show that if both objective functions are convex, transforming the objective functions may result in better IPS and OPS.

In this section, we show that similar results can be obtained for multi-objective optimization problems. To show these results, we introduce the following notation. The transformation function $h : \mathbb{R}^k \mapsto \mathbb{R}^k$ is defined as $h([z_1, \ldots, z_k]^\top) = [h_1(z_1), \ldots, h_k(z_k)]^\top$. The function h_i thus only transforms the output of the i^{th} objective function f_i; i.e., the vector function h consists of k functions $h_i : \mathbb{R} \mapsto \mathbb{R}$. Similarly, the inverse transformation function $h^{-1} : \mathbb{R}^k \mapsto \mathbb{R}^k$ is a vector function consisting of the inverse functions of $h_1, \ldots, h_k$. When discussing properties of these vector functions, we also consider the properties componentwise; e.g., we call h convex if all h_i, $i = 1, \ldots, k$, are convex. We call the set of vectors $\{h(z) \mid z \in Z\}$ the transformed objective space. To obtain the results mentioned above, we require that the transformation function h is strictly increasing. For the remainder of this chapter, we therefore assume that h is strictly increasing. The corresponding inverse function h^{-1} is then also strictly increasing. Furthermore, to improve IPS and OPS for convex PS, we require that h is strictly increasing and concave. This implies that h^{-1} exists, is strictly increasing, and convex. We mention explicitly when concavity of h is also required.

To distinguish between equivalent concepts in the transformed and original objective spaces, we use a different font to indicate concepts in the transformed space; e.g., $\mathcal{PS} := \{h(f(x)) \mid x \in \mathcal{X}_E, h(f(x)) \leq h(z^{ub})\}$ where $\mathcal{X}_E$ is the set of solutions x for which there exists no $\widetilde{x} \in X$ such that $h(f(\widetilde{x})) \leq h(f(x))$. The dummy points $\mathcal{d}^i(h(z))$ in the transformed space are defined as:

$$\mathcal{d}^i_j(h(z)) = \begin{cases} h_j(z_j) & \text{if} \quad j \neq i \\ kh_j(z^{ub}_j) + \theta & \text{if} \quad j = i \end{cases} \quad \text{for } j = 1, \ldots, k.$$

Furthermore, sets obtained by applying the inverse transformation h^{-1} to points in the transformed space are indicated by -1 as a superscript. This implies that $\text{conv}^{-1}\{\mathcal{IPS}, \mathcal{D}\}$ is defined as $\{z \mid h(z) \in \text{conv}\{\mathcal{IPS}, \mathcal{D}\}\}$. Similarly, applying h^{-1} to all points in $\mathcal{H}(w, b) = \{h(z) \mid w^\top h(z) = b\}$ gives us $\mathcal{H}^{-1}(w, b) = \{z \mid w^\top h(z) = b\}$, which we call an inverted hyperplane. The set $\mathcal{RH}^{-1}$ is defined as $\{\mathcal{H}^{-1}(w, b) \mid \mathcal{H}(w, b) \in \mathcal{RH}\}$ where $\mathcal{RH}$ is a

set containing all relevant hyperplanes of $\text{conv}\{\mathcal{IPS}, \mathcal{D}\}$. Lastly, when $\mathcal{F}$ is a facet of $\text{conv}\{\mathcal{IPS}, \mathcal{D}\}$, we refer to $\mathcal{F}^{-1}$ as an inverted facet of $\text{conv}^{-1}\{\mathcal{IPS}, \mathcal{D}\}$.

7.6.2 Non-convex objectives

To obtain inner and outer approximations of PS in case of non-convex objectives, we use the transformations in the following way. First, we have to find a transformation function $h(z)$ that is strictly increasing in z and for which $h(f(x))$ is convex in x. In order to use results proven in previous sections, we also assume that $h(f(x)) \geq 0$ for all $x \in X$. This assumption is no real limitation as we can always satisfy this condition if the previous two conditions are met. If we have found a transformation function h satisfying these properties, the transformed Pareto set $\mathcal{PS}$ is convex. This implies that the sets $\mathcal{IPS}$ and $\mathcal{OPS}$ can be determined using one of the previously discussed sandwich algorithms. Because the transformation function is strictly increasing, the corresponding sets $\mathcal{IPS}^{-1}$ and $\mathcal{OPS}^{-1}$ are inner and outer approximations of $\mathcal{PS}^{-1}$. Finally, we can show that $PS = \mathcal{PS}^{-1}$, which implies that $\mathcal{IPS}^{-1}$ and $\mathcal{OPS}^{-1}$ are also inner and outer approximations of PS. We prove that the above method indeed produces inner and outer approximations of PS through the following proposition.

Proposition 7.22. *Let $h : \mathbb{R}^k \mapsto \mathbb{R}^k$ be a strictly increasing transformation function. If $\mathcal{IPS}$ and $\mathcal{OPS}$ are inner and outer approximations of $\mathcal{PS}$, then $\mathcal{IPS}^{-1}$ and $\mathcal{OPS}^{-1}$ are inner and outer approximations of PS. Furthermore, $PS = \mathcal{PS}^{-1}$.*

Proof. As h is strictly increasing, h^{-1} exists and is also strictly increasing. By definition, $\mathcal{IPS}$ satisfies $\mathcal{IPS} \subseteq \mathcal{PS} + \mathbb{R}^k_{\leqq}$. So for every $h(z) \in \mathcal{IPS}$ there exists a $h(\widehat{z}) \in \mathcal{PS}$ such that $h(\widehat{z}) \leqq h(z)$. Because h^{-1} is strictly increasing, this condition implies that for every $z \in \mathcal{IPS}^{-1}$, there exists a vector $\widehat{z} \in \mathcal{PS}^{-1}$ such that $\widehat{z} \leqq z$, and so $\mathcal{IPS}^{-1} \subseteq \mathcal{PS}^{-1} + \mathbb{R}^k_{\leqq}$. In the same way, we can also show that $\mathcal{OPS}^{-1}$ satisfies $\mathcal{PS}^{-1} \subseteq \mathcal{OPS}^{-1} + \mathbb{R}^k_{\leqq}$. The sets $\mathcal{IPS}^{-1}$ and $\mathcal{OPS}^{-1}$ are thus inner and outer approximations of $\mathcal{PS}^{-1}$.

To complete the proof, we show that $\mathcal{PS}^{-1} = PS$. Because h is strictly increasing, $\mathcal{X}_E$ is equal to X_E and $\mathcal{PS}$ is equal to $\{h(z) \mid z \in PS\}$. Using this formulation of $\mathcal{PS}$, we conclude that $\mathcal{PS}^{-1} = \{z \mid h(z) \in \mathcal{PS}\}$ is equal to PS. □

The above results show how we can use $\mathcal{IPS}$ and $\mathcal{OPS}$ to find approximations $\mathcal{IPS}^{-1}$ and $\mathcal{OPS}^{-1}$ for PS. Similar to the untransformed case, we still need to determine which points of $\mathcal{IPS}^{-1}$ are non-$\mathcal{IPS}^{-1}$-dominated. As $\mathcal{IPS}^{-1}$ is in general not a convex hull of a finite number of points, we cannot directly use the method introduced in Section 7.4.3. The set $\mathcal{IPS}$, on the other hand, does satisfy this condition. Therefore, we can use the method of Section 7.4.3 to determine the non-$\mathcal{IPS}$-dominated points of $\mathcal{IPS}$. The following lemma shows that this method also gives us the non-$\mathcal{IPS}^{-1}$-dominated points of $\mathcal{IPS}^{-1}$.

Lemma 7.13. *Let $h : \mathbb{R}^k \mapsto \mathbb{R}^k$ be a strictly increasing transformation function. Then $z \in I\!P\!S^{-1}$ is a non-$I\!P\!S^{-1}$-dominated point if and only if the corresponding $h(z) \in I\!P\!S$ is a non-$I\!P\!S$-dominated.*

Proof. This result follows easily from the strict increasingness of h. □

7.6.3 Improving IPS and OPS

To improve IPS and OPS when all objectives are convex, we also have to find a transformation function h that satisfies several properties. Similar to the transformation function in Section 7.6.2, $h(z)$ must be strictly increasing in z and $h(f(x))$ must be convex in x. An additional property is that h must be a concave function. If we can determine a function h satisfying these properties, the following proposition shows how to improve IPS.

Proposition 7.23. *Let $f : \mathbb{R}^k \mapsto \mathbb{R}^k$ be convex and $h : \mathbb{R}^k \mapsto \mathbb{R}^k$ be a strictly increasing and concave transformation function such that $h \circ f$ is convex. Consider an inner approximation $IPS = conv\{z^1, \ldots, z^n\}$ of PS. If $I\!P\!S = conv\{h(z) \mid z \in IPS\}$ and $I\!P\!S^{-1} = \{z \mid h(z) \in I\!P\!S\}$, then $\alpha(z, I\!P\!S^{-1}) \leq \alpha(z, IPS)$ for every $z \in PS$.*

Proof. Because h is strictly increasing, we can easily see that the set $\{h(z) \mid z \in IPS\}$ is an inner approximation of $\mathcal{PS}$. Because $h \circ f$ is convex, $\mathcal{PS}$ must be a convex set. Hence, $I\!P\!S = \text{conv}\{h(z) \mid z \in IPS\}$ is also an inner approximation of $\mathcal{PS}$. According to Proposition 7.22, the corresponding set $I\!P\!S^{-1}$ must be an inner approximation of PS.

For a point $z \in PS$, let $\widehat{z} \in IPS$ be a point that $\alpha\epsilon$-dominates z with $\alpha = \alpha(z, IPS)$. Because $IPS = \text{conv}\{z^1, \ldots, z^n\}$, we can write $\widehat{z}$ as $\sum_{i=1}^n \lambda^i z^i$ with $\sum_{i=1}^n \lambda^i = 1$ and $\lambda^i \geq 0$ for $i = 1, \ldots, n$. As $h(z^i) \in I\!P\!S$ for all $i = 1, \ldots, n$, the point $\sum_{i=1}^n \lambda^i h(z^i)$ is an element of $I\!P\!S$. Subsequently, the point $\widetilde{z} = h^{-1}\left(\sum_{i=1}^n \lambda^i h(z^i)\right)$ is an element of $I\!P\!S^{-1}$. Note that the inverse function h^{-1} exists and is strictly increasing and convex because h is strictly increasing and concave. Using the convexity of h^{-1}, we can show the following:

$$\widetilde{z} = h^{-1}\left(\sum_{i=1}^n \lambda^i h(z^i)\right) \leq \sum_{i=1}^n \lambda^i h^{-1}(h(z^i)) = \sum_{i=1}^n \lambda^i z^i = \widehat{z}.$$

Because $\widetilde{z} \in I\!P\!S^{-1}$ and $\widetilde{z} \leq \widehat{z}$, this proves that $\alpha(z, I\!P\!S^{-1}) \leq \alpha(z, IPS)$. □

Note that when we take an IPS obtained with one of the sandwich algorithms, the set $\text{conv}\{IPS, D\}$ also forms an inner approximation of PS. Therefore, the above lemma also holds when we replace IPS by $\text{conv}\{IPS, D\}$.

To show that OPS can also be improved through this transformation, we first prove the following lemma, which shows how we can find a supporting hyperplane of $\mathcal{PS}$ when we have a supporting hyperplane of PS.

Lemma 7.14. *Let $f : \mathbb{R}^k \mapsto \mathbb{R}^k$ and X be convex and $h : \mathbb{R}^k \mapsto \mathbb{R}^k$ be a strictly increasing and concave transformation function such that $h \circ f$ is convex. Consider a hyperplane $H(w, b)$ that supports PS at $\bar{z} \in Z$. Then hyperplane $\mathcal{H}(\widetilde{w}, \widetilde{b})$ with $\widetilde{w}_i = w_i(h_i^{-1})'(\bar{y}_i)$ for $i = 1, \ldots, k$ and $\widetilde{b} = \widetilde{w}^\top \bar{y}$ supports $\mathcal{PS}$ at $\bar{y} = h(\bar{z})$.*

Proof. Let $p(z)$ be a continuous non-decreasing function such that $PS = \{z \mid p(z) = 0, z \in Z\}$. Note that such a function exists because f and X are convex, which implies that PS is convex, non-decreasing, and connected (see, e.g., Miettinen (1999)). As $PS = \mathcal{PS}^{-1}$ according to Proposition 7.22, this expression for PS implies that $\mathcal{PS} = \{y \mid p(h^{-1}(y)) = 0, y \in Y\}$ where $Y := \{h(z) \mid z \in Z\}$. As $H(w, b)$ supports PS at $\bar{z}$, we know that w is a subgradient of $p(z)$ at $\bar{z}$, i.e., $\frac{\partial p(\bar{z})}{\partial z} = w$. Using the following chain rule for subgradients (see Theorem 10.6 of Rockafellar and Wets (1998)) at $\bar{y} = h(\bar{z})$:

$$\frac{\partial p(h^{-1}(\bar{y}))_i}{\partial y_i} = \frac{\partial p(\bar{z})}{\partial z_i}(h_i^{-1})'(\bar{y}_i) \text{ for } i = 1, \ldots, k,$$

we obtain that $\widetilde{w}$ with $\widetilde{w}_i = w_i(h_i^{-1})'(\bar{y}_i)$ for $i = 1, \ldots, k$ is a subgradient of $\mathcal{PS}$. Therefore, we conclude that $\mathcal{H}(\widetilde{w}, \widetilde{b})$ is a supporting hyperplane of $\mathcal{PS}$ at $\bar{y}$. □

In Proposition 7.24, we show that the back-transformation of the supporting hyperplane of $\mathcal{PS}$ obtained in Lemma 7.14 gives a better OPS.

Proposition 7.24. *Let $f : \mathbb{R}^k \mapsto \mathbb{R}^k$ and X be convex and $h : \mathbb{R}^k \mapsto \mathbb{R}^k$ be a strictly increasing and concave function such that $h \circ f$ is convex. Consider a hyperplane $H(w, b)$ that supports PS at $\bar{z} \in Z$ and the corresponding hyperplane $\mathcal{H}(\widetilde{w}, \widetilde{b})$ that supports $\mathcal{PS}$ at $\bar{y} = h(\bar{z})$, with $\widetilde{w}_i = w_i(h_i^{-1})'(\bar{y}_i)$ for $i = 1, \ldots, k$ and $\widetilde{b} = \widetilde{w}^\top \bar{y}$. Then $\mathcal{H}^{-1}(\widetilde{w}, \widetilde{b})$ is a tighter outer approximation of PS than $H(w, b)$; i.e., $\mathcal{H}^{-1}(\widetilde{w}, \widetilde{b})$ is an outer approximation of PS and $H(w, b)$ is an outer approximation of $\mathcal{H}^{-1}(\widetilde{w}, \widetilde{b})$.*

Proof. First we show that $\mathcal{H}^{-1}(\widetilde{w}, \widetilde{b})$ is an outer approximation of PS. Since $\mathcal{H}(\widetilde{w}, \widetilde{b})$ supports $\mathcal{PS}$ at $\bar{y}$, it follows that

$$\forall y^1 \in \mathcal{PS}, \exists y^2 \in \mathcal{H}(\widetilde{w}, \widetilde{b}) \quad : \quad y^2 \leqq y^1.$$

Because h^{-1} is strictly increasing and $PS = \mathcal{PS}^{-1}$, this inequality is equivalent to

$$\forall z^1 \in PS, \exists z^2 \in \mathcal{H}^{-1}(\widetilde{w}, \widetilde{b}) \quad : \quad z^2 \leqq z^1.$$

Hence $\mathcal{H}^{-1}(\widetilde{w}, \widetilde{b})$ is an outer approximation of PS.

Now we prove that $H(w, b)$ is an outer approximation of $\mathcal{H}^{-1}(\widetilde{w}, \widetilde{b})$. Let $p(z)$ be a non-decreasing function such that $PS = \{z \mid p(z) = 0, z \in Z\}$. This implies that $\mathcal{PS} = \{y \mid p(h^{-1}(y)) = 0, y \in Y\}$ where $Y := \{h(z) \mid z \in Z\}$. The supporting hyperplane $\mathcal{H}(\widetilde{w}, \widetilde{b})$ at $\bar{y} = h(\bar{z})$ of Lemma 7.14 can now be written as

$$\mathcal{H}(\widetilde{w}, \widetilde{b}) = \left\{ y \middle| \sum_i w_i(h_i^{-1})'(\bar{y}_i)[y_i - \bar{y}_i] = 0 \right\}, \tag{7.11}$$

where $w = \frac{\partial p(\bar{z})}{\partial z}$ is a subgradient of $p(z)$ at $\bar{z}$. Transforming $\mathcal{H}(\widetilde{w}, \widetilde{b})$ back, we get $\mathcal{H}^{-1}(\widetilde{w}, \widetilde{b}) = \{z \mid p_2(z) = 0\}$ where

$$p_2(z) = \sum_i w_i (h_i^{-1})'(\bar{y}_i)[h_i(z_i) - h_i(\bar{z}_i)].$$

Moreover, we have $H(w, b) = \{z \mid p_1(z) = 0\}$, where

$$p_1(z) = \sum_i w_i [z_i - \bar{z}_i].$$

For the i^{th} term of $p_2(z)$, we have

$$(h_i^{-1})'(\bar{y}_i)[h_i(z_i) - h_i(\bar{z}_i)] \leq (h_i^{-1})'(\bar{y}_i)[h_i(\bar{z}_i) + h_i'(\bar{z}_i)(z_i - \bar{z}_i) - h_i(\bar{z}_i)] = z_i - \bar{z}_i,$$

where the inequality holds because h_i is concave and the equality holds because $(h_i^{-1})' = \frac{1}{h_i'}$ and h_i^{-1} is strictly increasing. Since $p(z)$ is non-decreasing and thus $\frac{\partial p(\bar{z})}{\partial z_i} \geq 0$, we have $p_2(z) \leq p_1(z)$. We thus have $p_1(z) \geq 0, \forall z \in \mathcal{H}^{-1}(\widetilde{w}, \widetilde{b})$, which means that $H(w, b)$ is an outer approximation of $\mathcal{H}^{-1}(\widetilde{w}, \widetilde{b})$. □

7.6.4 Calculating $\alpha(PS, I\mathcal{PS}^{-1})$ and $\alpha(O\mathcal{PS}^{-1}, I\mathcal{PS}^{-1})$

The general idea behind the method for calculating $\alpha(PS, I\mathcal{PS}^{-1})$ and $\alpha(O\mathcal{PS}^{-1}, I\mathcal{PS}^{-1})$ is the same as for $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$ described in Section 7.5.2. The main difference is that in all steps we must replace $H(w, b)$ by $\mathcal{H}^{-1}(w, b)$. This has the drawback that $\max_{z \in O\mathcal{PS}^{-1}} \alpha(z, \mathcal{H}^{-1}(w, b))$ can no longer be calculated as an LP problem. Instead, we have to solve a non-convex problem. To prove that all steps made in the algorithm still hold, we first prove the following lemma, which is similar to Lemma 7.10.

Lemma 7.15. *Let $\epsilon \in \mathbb{R}^k_>$, $h : \mathbb{R}^k \mapsto \mathbb{R}^k$ be a strictly increasing and continuous transformation function and $\mathcal{H}^{-1}(w, b) \in \mathcal{RH}^{-1}$. For every $z \in PS$, $\alpha(z, \mathcal{H}^{-1}(w, b))$ is equal to the unique $\widehat{\alpha}$ for which the point $z + \widehat{\alpha}\epsilon$ is on $\mathcal{H}^{-1}(w, b)$.*

Proof. Let $\widehat{z}$ be an arbitrary point on $\mathcal{H}^{-1}(w, b)$. As $\epsilon \in \mathbb{R}^k_>$, we can always determine an α such that $z + \alpha\epsilon > \widehat{z}$. Because h is strictly increasing and $w \geq 0$, $w^\top h(z + \alpha\epsilon) > w^\top h(\widehat{z}) = b$. Similarly, we can prove that there must exist an α such that $w^\top h(z + \alpha\epsilon) < b$. As h is a strictly increasing continuous function, there must exist a unique $\widehat{\alpha}$ such that $w^\top h(z + \widehat{\alpha}\epsilon) = b$. As z is $\alpha\epsilon$-dominated by $z + \widehat{\alpha}\epsilon \in \mathcal{H}^{-1}(w, b)$, we know that $\alpha(z, \mathcal{H}^{-1}(w, b)) \leq \widehat{\alpha}$. If $\alpha(z, \mathcal{H}^{-1}(w, b)) < \widehat{\alpha}$, there must exist a $\widetilde{z} \in \mathcal{H}^{-1}(w, b)$ such that $\widetilde{z} < z + \widehat{\alpha}\epsilon$. However, this would imply

$$b = w^\top h(\widetilde{z}) < w^\top h(z + \widehat{\alpha}\epsilon) = b.$$

Because of this contradiction, we conclude that $\alpha(z, \mathcal{H}^{-1}(w, b)) = \widehat{\alpha}$. □

Using the results of Lemma 7.7, 7.8 and 7.15, we prove the following two lemmas.

Lemma 7.16. *Let $\epsilon \in \mathbb{R}^k_>$ and $h : \mathbb{R}^k \mapsto \mathbb{R}^k$ be a strictly increasing transformation function. Consider $z \in PS$ and $\mathcal{H}^{-1}(w,b) \in \mathcal{RH}^{-1}$. Then $\alpha(z, \mathcal{IPS}^{-1}) \geq \alpha(z, \mathcal{H}^{-1}(w,b))$.*

Proof. Lemma 7.8 implies $w \geq 0$. Assume by contradiction that z is $\widehat{\alpha}\epsilon$-dominated by $\mathcal{IPS}^{-1}$ with $\widehat{\alpha} < \alpha(z, \mathcal{H}^{-1}(w,b))$. This implies that there should exist a point $\widetilde{z} \in \mathcal{IPS}^{-1}$ such that $\widetilde{z} \leqq z + \widehat{\alpha}\epsilon$. For this point $\widetilde{z}$ it holds that

$$w^\top h(\widetilde{z}) \leqq w^\top h(z + \widehat{\alpha}\epsilon) < w^\top h(z + \alpha(z, \mathcal{H}^{-1}(w,b))\epsilon) = b,$$

where the last equality follows from Lemma 7.15. As $\mathcal{H}^{-1}(w,b) \in \mathcal{RH}^{-1}$, for all points z in $\mathcal{IPS}^{-1}$ it must hold that $w^\top h(z) \geq b$. This implies that $\widetilde{z}$ cannot be an element of $\mathcal{IPS}^{-1}$. Therefore, z can be $\alpha\epsilon$-dominated only by $\mathcal{IPS}^{-1}$ with $\alpha \geq \alpha(z, \mathcal{H}^{-1}(w,b))$. □

Lemma 7.17. *Consider $\epsilon \in \mathbb{R}^k_>$, a strictly increasing transformation function $h : \mathbb{R}^k \mapsto \mathbb{R}^k$, and $z \in PS$. Then $\alpha(z, \mathcal{IPS}^{-1}) = \alpha(z, \mathcal{H}^{-1}(w,b))$ must hold for at least one $\mathcal{H}^{-1}(w,b) \in \mathcal{RH}^{-1}$.*

Proof. Define $z^\alpha = z + \alpha(z, \mathcal{IPS}^{-1})\epsilon$ and let $z^{\mathcal{IPS}^{-1}} \in \mathcal{IPS}^{-1}$ be such that $\alpha(z, \{z^{\mathcal{IPS}^{-1}}\}) = \alpha(z, \mathcal{IPS}^{-1})$. By the definition of $\alpha(z,\cdot)$, we know that $z^{\mathcal{IPS}^{-1}} \leqq z^\alpha$ with equality for at least one coordinate. Furthermore, there cannot exist a point $\widehat{z} \in \mathcal{IPS}^{-1}$ such that $\widehat{z} < z^{\mathcal{IPS}^{-1}}$ or $\widehat{z} < z^\alpha$. Because all points in $\mathcal{D}$ are weakly dominated by points in $\mathcal{IPS}$ and $h^{-1}(z)$ is strictly increasing, the same is true for all points $\widehat{z}$ in $\text{conv}^{-1}\{\mathcal{IPS}, \mathcal{D}\}$. This implies that $z^{\mathcal{IPS}^{-1}}$ must lie on one or more inverted facets of $\text{conv}^{-1}\{\mathcal{IPS}, \mathcal{D}\}$. Let $\mathcal{F}^{-1}$ be one of these inverted facets and $\mathcal{F}$ be the associated facet in the transformed space. Because $h(z^{\mathcal{IPS}^{-1}}) \leqq h(z^{ub})$ and $h(z^{\mathcal{IPS}^{-1}}) \in \mathcal{F}$, $\mathcal{F}$ must be a relevant facet of $\text{conv}\{\mathcal{IPS}, \mathcal{D}\}$ according to Lemma 7.7. Consequently, $\mathcal{F}^{-1}$ is a relevant inverted facet, which has a supporting inverted hyperplane $\mathcal{H}^{-1}(\widetilde{w}, \widetilde{b}) \in \mathcal{RH}^{-1}$. For this inverted hyperplane it holds that $z^{\mathcal{IPS}^{-1}} \in \mathcal{F}^{-1} \subset \mathcal{H}^{-1}(\widetilde{w}, \widetilde{b})$. Using this property, we derive the following relation:

$$\alpha(z, \mathcal{H}^{-1}(\widetilde{w}, \widetilde{b})) \leq \alpha(z, \{z^{\mathcal{IPS}^{-1}}\}) = \alpha(z, \mathcal{IPS}^{-1}).$$

The inequality follows from the definition of $\alpha(z,\cdot)$ and the fact that $z^{\mathcal{IPS}^{-1}} \in \mathcal{H}^{-1}(\widetilde{w}, \widetilde{b})$. The equality follows from the definition of $z^{\mathcal{IPS}^{-1}}$. Combining the above equation with Lemma 7.16 gives that $\alpha(z, \mathcal{IPS}^{-1}) = \alpha(z, \mathcal{H}^{-1}(\widetilde{w}, \widetilde{b}))$ must hold for this $\mathcal{H}^{-1}(\widetilde{w}, \widetilde{b}) \in \mathcal{RH}^{-1}$. □

By combining Lemmas 7.16 and 7.17, we obtain the following equation:

$$\alpha(z, \mathcal{IPS}^{-1}) = \max_{\mathcal{H}^{-1}(w,b) \in \mathcal{RH}^{-1}} \alpha(z, \mathcal{H}^{-1}(w,b)), \tag{7.12}$$

which gives an expression for $\alpha(z, I\!P\!S^{-1})$. By definition, to determine $\alpha(PS, I\!P\!S^{-1})$, we must take the maximum of $\alpha(z, I\!P\!S^{-1})$ over all $z \in PS$. However determining $\alpha(z, I\!P\!S^{-1})$ explicitly for every $z \in PS$ is not an option as there are an infinite number of vectors in the set PS. The number of $\mathcal{H}^{-1}(w,b) \in \mathcal{RH}^{-1}$, on the other hand, is finite. Therefore, we decide to calculate $\max_{z\in PS} \alpha(z, \mathcal{H}^{-1}(w,b))$ for every $\mathcal{H}^{-1}(w,b) \in \mathcal{RH}^{-1}$ and use these results to calculate $\alpha(PS, I\!P\!S^{-1})$. The following lemma gives a method for determining $\max_{z\in PS} \alpha(z, \mathcal{H}^{-1}(w,b))$ without knowing PS explicitly.

Lemma 7.18. *Consider a hyperplane $\mathcal{H}^{-1}(w,b) \in \mathcal{RH}^{-1}$, a strictly increasing transformation function $h : \mathbb{R}^k \mapsto \mathbb{R}^k$, and $\epsilon \in \mathbb{R}^k_>$. Then $\max_{z\in PS} \alpha(z, \mathcal{H}^{-1}(w,b))$ can be found by solving the following optimization problem:*

$$\begin{array}{ll} \max_{\alpha,z} & \alpha \\ \text{s.t.} & z \in Z \\ & w^\top h(z + \alpha\epsilon) \leq b. \end{array} \tag{7.13}$$

Proof. Let us define $g(z,\alpha) = w^\top h(z+\alpha\epsilon)$. As $w \geq 0$, $\epsilon > 0$, and h is strictly increasing, g is a strictly increasing function in α. Because we maximize α, this implies that we can replace the last inequality constraint by the equality constraint $w^\top h(z + \alpha\epsilon) = b$.

We now show that there always exists a solution $\widehat{z} \in PS$ for this problem. Let us denote an optimal z and α of problem 7.13 by z^* and α^*. We can easily show that there cannot exist an $z \in Z$ such that $z < z^*$ as this would imply that α^* is not optimal. Therefore, either z^* is in PS or z^* is weakly dominated by another point $\widehat{z} \in PS$. In the first case, we can simply take $\widehat{z} = z^*$. In the second case, because $w \geq 0$ and h is strictly increasing, we can show the following:

$$w^\top h(\widehat{z} + \alpha^*\epsilon) \leq w^\top h(z^* + \alpha^*\epsilon) = b.$$

When $\widehat{\alpha}$ satisfies $w^\top h(\widehat{z}+\widehat{\alpha}\epsilon) = b$, then $\widehat{\alpha} \geq \alpha^*$ because $g(z,\alpha)$ is strictly increasing in α. However, $\widehat{\alpha} > \alpha^*$ cannot hold because α^* is optimal. We can thus conclude that $\widehat{\alpha} = \alpha^*$ and that $\widehat{z} \in PS$ is also an optimal solution. The constraint $z \in Z$ can thus be replaced by $z \in PS$. We know from Lemma 7.15 that for every $z \in PS$, $\alpha(z, \mathcal{H}^{-1}(w,b))$ is equal to the unique α that satisfies the equality constraint $w^\top h(z + \alpha\epsilon) = b$. This proves that we can determine $\max_{z\in PS} \alpha(z, \mathcal{H}^{-1}(w,b))$ by solving the above problem. □

Solving problem (7.13) for every the value of $\mathcal{H}(w,b) \in \mathcal{RH}$, we can determine $\max_{z\in PS} \alpha(z, \mathcal{H}(w,b))$ explicitly for every $\mathcal{H}(w,b) \in \mathcal{RH}$. Combining the above results, we can use the following equation to calculate $\alpha(PS, I\!P\!S^{-1})$:

$$\begin{aligned} \alpha(PS, I\!P\!S^{-1}) = \max_{z\in PS} \alpha(z, I\!P\!S^{-1}) &= \max_{z\in PS} \max_{\mathcal{H}(w,b)\in\mathcal{RH}} \alpha(z, \mathcal{H}(w,b)) \\ &= \max_{\mathcal{H}(w,b)\in\mathcal{RH}} \max_{z\in PS} \alpha(z, \mathcal{H}(w,b)). \end{aligned}$$

7.7 Application of enhancements

7.7.1 Application of dummy points

In previous sections, we have seen that adding dummy points to IPS has several benefits. These benefits are not just for one specific sandwich algorithm but apply to various sandwich algorithms. Some benefits even apply to general MOP methods for approximating a convex set PS. To illustrate this, we describe various possible applications of dummy points.

Adding dummy points to IPS to solve the problem of 'undesirable' normals can be useful for any MOP algorithm that uses normals of facets as weights to determines Pareto points. The algorithms of Solanki et al. (1993) and Craft et al. (2006) are only two examples of such algorithms. By eliminating the relevant facets with 'undesirable' normals, no special rules have to be constructed to deal with these facets.

The dummy points can be used to determine IPS^{nId} when IPS is a convex hull of a finite number of points. Besides this condition on the form of IPS, no other conditions have to be satisfied in order to apply the dummy points for this purpose. After adding the dummy points to IPS, we can simply use the algorithm described in Section 7.4.3 to determine all non-IPS-dominated points.

7.7.2 Application of error measure

Calculating $\alpha(PS, IPS)$ by using $\text{conv}\{IPS, D\}$ can be done for any IPS that is the convex hull of a finite number of points. The calculation of $\alpha(PS, IPS)$ is mainly useful if we want to know the true approximation error of IPS. In real life applications, however, the method in Section 7.5 for calculating $\alpha(PS, IPS)$ is generally too time consuming as it requires solving a large number of weighted sum problems. Furthermore, the outcomes of these weighted sum problems could be used to improve IPS and OPS. After this improvement, the calculated value of $\alpha(PS, IPS)$ is probably not the real accuracy of the new IPS but only an upper bound.

The calculation of $\alpha(OPS, IPS)$ does not have the above mentioned problems. As the weighted sum problems become LP problems, calculating $\alpha(OPS, IPS)$ can be done much faster. Another difference is that the outcomes of these optimizations cannot be used to improve IPS. Because the measure $\alpha(OPS, IPS)$ obviously requires an IPS and OPS, we can calculate it only for sandwich algorithms. Furthermore, we can use the measure also within a sandwich algorithm to decide which facet to use for a next optimization. In Section 7.7.4, we describe how this can be done in the algorithm of Solanki et al. (1993).

7.7.3 Application of transformations

The transformations described in Section 7.6 can be divided into two classes. Firstly, we can use transformations to determine OPS and IPS if we have non-convex objectives. In this case, we must determine for each non-convex objective f_i a transformation function h_i that is strictly increasing and for which $h_i \circ f_i$ is convex. When this is possible, we can use any of the sandwich algorithms to determine $\mathcal{IPS}$ and $\mathcal{OPS}$, which approximate $\mathcal{PS}$. These approximations can subsequently be used to determine $\mathcal{IPS}^{-1}$ and $\mathcal{OPS}^{-1}$ as approximations of PS.

Secondly, we can transform objectives to obtain tighter OPS and IPS for a convex PS. To do this, we must find a strictly increasing and concave function h such that $h \circ f$ is convex. We can use this transformation to improve the final IPS and OPS obtained with any of the sandwich algorithms. Furthermore, instead of using $\alpha(OPS, IPS)$ to select a next facet, we can use $\alpha(\mathcal{OPS}^{-1}, \mathcal{IPS}^{-1})$. However, a drawback of $\alpha(\mathcal{OPS}^{-1}, \mathcal{IPS}^{-1})$ is that it is more time-consuming to calculate than $\alpha(OPS, IPS)$. We thus have to decide whether the additional calculation time is justified by a possibly better choice of facets.

Objectives to which we can apply the above transformations can, among others, be found in geometric programming and IMRT optimization. An example in geometric programming is treated in Section 7.8.5. For transformations of IMRT objectives, we refer to the paper of Hoffmann et al. (2008).

7.7.4 Enhanced version of algorithm of Solanki et al.

All enhancements described above are aimed at determining an accurate approximation of a convex (or non-convex) PS more efficiently. To test whether these enhancements can indeed improve efficiency, we enhance the algorithm of Solanki et al. (1993) by adding dummy points and changing the error measure. We do not include the use of transformations as this is not possible for all (convex) MOPs. More specifically, the enhanced version of the algorithm of Solanki et al. (1993) differs from the original version in the following ways.

In every step of the algorithm, IPS is replaced by $\text{conv}\{IPS, D\}$. As shown in Section 7.5.2, this enables us to easily calculate $\alpha(OPS, IPS)$ by solving problem (7.3) for all relevant facets. According to the result of Proposition 7.21, we must use $\frac{b - w^\top \bar{z}}{w^\top \epsilon}$ instead of $b - w^\top \bar{z}$ as the error of a facet where $\bar{z}$ is the solution of problem (7.3). By calculating the error of the facets in this way, $\alpha(OPS, IPS)$ is equal to the maximum of the errors of all relevant facets.

Although the algorithms of Craft et al. (2006) and Klamroth et al. (2002) can also be enhanced, we have not tested the enhanced versions of these algorithms. If we would enhance the algorithm of Craft et al. (2006) through dummy points, the algorithm would

become very similar to the enhanced version of the algorithm of Solanki et al. (1993). The only difference that would remain is the difference in the error used to select the facets. Enhancing the algorithm of Craft et al. (2006) with both the dummy points and the error measure would make the algorithms completely the same. The algorithms of Klamroth et al. (2002) were not enhanced because initial tests showed that these algorithms are much less efficient than the algorithms of Solanki et al. (1993) and Craft et al. (2006). A main reason for this is that not all obtained Pareto points are added to IPS. As this drawback is not reduced by our enhancements, it seems unlikely that an enhanced version of the algorithm of Klamroth et al. (2002) would become more efficient than the other sandwich algorithms.

7.8 Numerical comparison of sandwich algorithms

7.8.1 Comparison method

In our numerical comparison, we consider the following five algorithms:

- Solanki et al. (1993) (SOLANKI).
- Klamroth et al. (2002) with equality constraint (KLAMROTH$^{=}$).
- Klamroth et al. (2002) with inequality constraint (KLAMROTH$^{\leq}$).
- Craft et al. (2006) (CRAFT).
- Enhanced version of Solanki et al. (1993) (ENHANCED).

To compare these algorithms, we use three different test cases. The first two cases are artificial; the third case is an IMRT problem. To show the effect of transforming convex objective functions, a fourth case is used that consists of a geometric programming problem. A more elaborate description of these four cases is given in Sections 7.8.2, 7.8.3, 7.8.4, and 7.8.5.

For all cases, we must set a number of parameters. The parameter z^{ub} is taken equal to the pseudo-nadir point z^{pN}, defined in Section 7.2. To calculate the error measure, we must also set the vector ϵ that specifies for each objective a maximal allowable error. As the objectives in cases 1 and 2 have no practical interpretation, it is not immediately clear how to select values for ϵ. A reasonable choice could be to take ϵ equal to the difference between the vectors z^{U} and z^{ub}, which contain the minimal and maximal possible values of all objectives. By setting ϵ in this way, we try to give each objective equal importance. However, we are not interested in an ϵ-approximation for this particular ϵ, because any point $z \in Z$ forms an ϵ-approximation of PS. Instead, we are interested in reaching a certain value for $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$. For example, if we want a maximal

error of 5 percent of the difference between the minimal and maximal objective values, then we want to determine when $\alpha(PS, IPS) \leq 0.05$ or $\alpha(OPS, IPS) \leq 0.05$ is reached. Note, that this is not the same as putting a bound on the relative error discussed in Section 7.5.1. The relative error in a point $z \in IPS$ is a percentage of the value of z, whereas the above bound is a percentage of the difference between z^U and z^{ub}. The maximal absolute error allowed by this bound is thus the same for all points $z \in IPS$. Therefore, it is a bound on the absolute error and not on the relative error.

To measure the efficiency, we assume that solving an optimization problem to find a Pareto point is a difficult problem and relatively the most time intensive part of each approximation algorithm. Because the optimizations in the first two test cases are quite easy to solve, the used CPU-time is not representative for real-life cases. Therefore, we measure the computational effort by the number of performed optimizations. In CRAFT, SOLANKI, and ENHANCED, we thus measure the number of weighted sum problems. In KLAMROTH$^{\leq}$ and KLAMROTH$^{=}$, we measure the number of runs of problems (7.4) and (7.5). For each algorithm, we denote the number of optimizations performed at a certain stage of the algorithm by n^{opt}. Although the anchor points must also be determined by solving optimization problems, these optimizations are not included in n^{opt} as they are often easier to solve and they are the same for all algorithms.

As we mentioned in the description of KLAMROTH$^{\leq}$ and KLAMROTH$^{=}$ in Section 7.3.3, not all points and inequalities determined by problems (7.4) and (7.5) are used to improve IPS and OPS. This does not seem very efficient, as determining these points and inequalities is time-consuming. Therefore, we also determine the inner and outer approximations that use all available points and inequalities; we denote these approximations by IPS^* and OPS^*.

Lastly, using only $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$ to measure the accuracy seems to give an unfair advantage to ENHANCED as this algorithm uses $\alpha(OPS, IPS)$ to select a facet in each iteration. Therefore, we also calculated the accuracy of the different IPS and OPS using the error measures used in SOLANKI and CRAFT. However, the measure used in SOLANKI also calculates the distance between IPS and OPS for facets with an 'undesirable' normal. The maximum of all distances was often attained by one of these facets. A large distance for one of these facets does not necessarily imply a large inaccuracy near these facets. Especially for IPSs obtained by CRAFT, the measure of SOLANKI often returned the same value for IPSs with significantly different values for $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$. Because of this drawback, we decided not to use this measure. The measure used in CRAFT has a different drawback. When adding a new point to IPS to obtain a new inner approximation IPS^*, the measure could give a higher value for IPS^* than for IPS. As an improvement of IPS could result in a deterioration of this error measure, we also decided not to use this measure in our comparison.

7.8.2 Test case 1: artificial 3-dimensional case

The first test case has the following three objectives and three constraints:

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= x_2 \\ f_3(x) &= x_3 \\ x_1 &\geq (x_2-9)^2+(x_3-3)^2 \\ x_2 &\geq (x_1-4)^2+(x_3-3)^2 \\ x_3 &\geq (x_1-4)^2+(x_2-9)^2. \end{aligned}$$

It is easy to see that all objectives and the feasible region are convex.

We compare the IPS and OPS of the different algorithms on both $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$. The first measure is important because it shows the real accuracy of IPS. In practice, however, we usually have only $\alpha(OPS, IPS)$ as calculating $\alpha(PS, IPS)$ is too time-consuming. The decision maker will thus use $\alpha(OPS, IPS)$ to determine if a certain approximation is accurate enough.

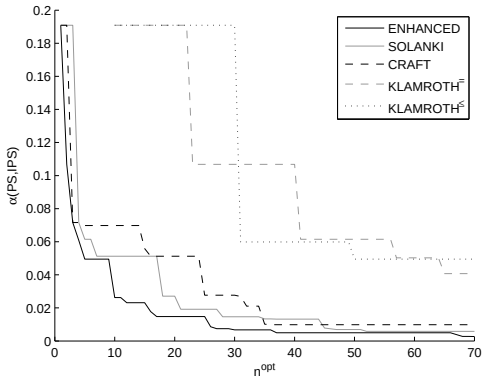


Figure 7.5: $\alpha(PS, IPS)$ of test case 1 as function of n^{opt}.

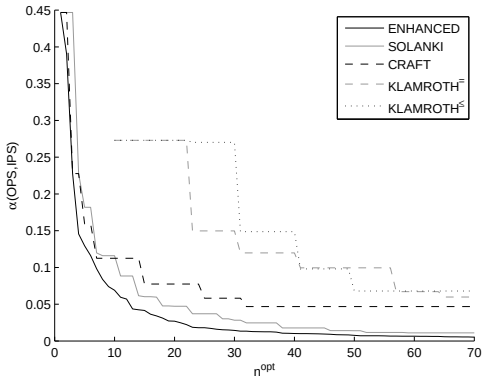


Figure 7.6: $\alpha(OPS, IPS)$ of test case 1 as function of n^{opt}.

In Figures 7.5 and 7.6, we show the achieved accuracy of the five algorithms for different values of n^{opt}. As mentioned before, the inefficiency of adding only a selection of the obtained points and inequalities to IPS and OPS, make KLAMROTH$^{\leq}$ and KLAMROTH$^{=}$ the least efficient. When considering IPS^* and OPS^*, the results improve but remain considerably worse than the results of the other algorithms. Therefore, we did not plot $\alpha(PS, IPS^*)$ and $\alpha(OPS^*, IPS^*)$ in the above figures. The algorithms CRAFT, SOLANKI and ENHANCED perform considerably better. For most values of n^{opt}, ENHANCED performs the best of the five tested algorithms.

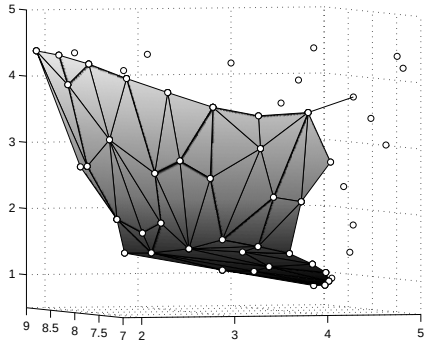


Figure 7.7: PS of test case 1.

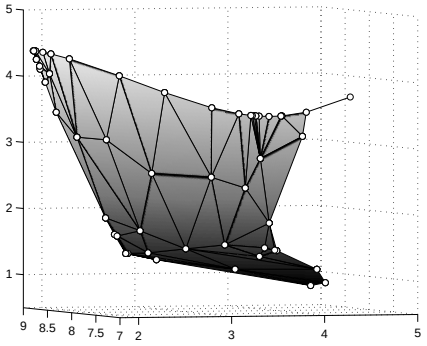


Figure 7.8: Approximation of CRAFT.

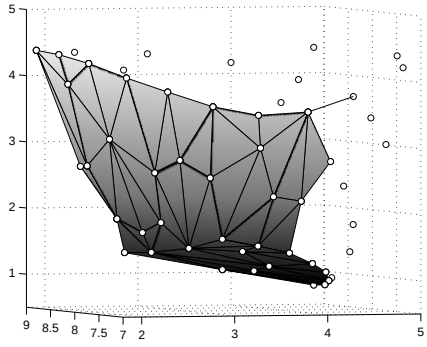


Figure 7.9: Approximation of SOLANKI.

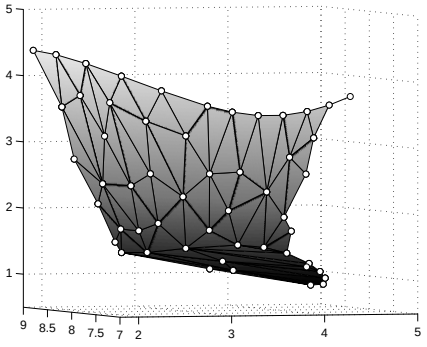


Figure 7.10: Approximation of ENHANCED.

As this MOP has three objectives, it is possible to draw PS and compare the different results graphically. In Figure 7.7, we show the set PS of this problem. In Figures 7.8, 7.9 and 7.10, the lines and shaded area show IPS^{nId} obtained when $n^{opt} = 50$. The dots represent the anchor points and all points found by the optimizations in the sandwich algorithms. When looking at the approximation obtained with CRAFT in Figure 7.8, we notice that there are some clusters of points. These clusters may be caused by the method used in CRAFT for selecting a next weighting vector for the weighted sum method. The approximations of SOLANKI and ENHANCED also appear to contain a cluster of points in the lower left part of the figure. These points, however, only appear to be clustered due to the angle at which we view the approximation and are thus not really clustered. Although SOLANKI generates no clusters, it does have another drawback as we can see in Figure 7.9. Of the 50 points found by the optimizations, 14 are IPS-dominated and thus no element of IPS^{nId}. All these points are obtained by using the weighted sum with an 'undesirable' normal. Figure 7.10 shows that the IPS obtained through ENHANCED does not have these drawbacks as all points are nicely distributed and are part of IPS^{nId}.

7.8.3 Test case 2: artificial 5-dimensional case

To test the performance for MOPs with more dimensions, we use a linear five-dimensional MOP as the second test case. Similar to the first case, we set the objective functions equal to the variables. To determine the constraints, 30 points were randomly drawn from a uniform distribution on the interval $[0, 1]^5$. The convex hull of these points was taken as the feasible region X.

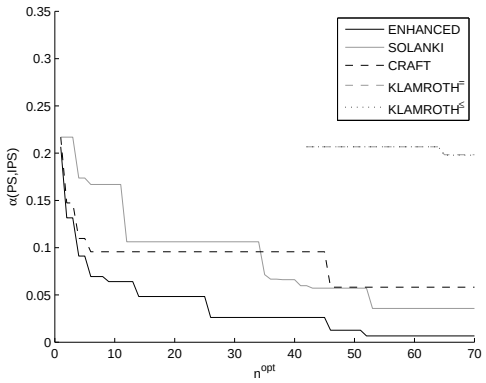


Figure 7.11: $\alpha(PS, IPS)$ of test case 2 as function of n^{opt}.

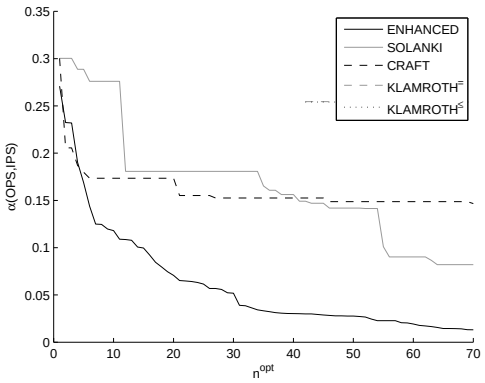


Figure 7.12: $\alpha(OPS, IPS)$ of test case 2 as function of n^{opt}.

In Figures 7.11 and 7.12, we plot the values of $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$ for different values of n^{opt}. Again KLAMROTH$^{\leq}$ and KLAMROTH$^{=}$ are the least efficient algorithms. CRAFT initially performs better than SOLANKI, but for n^{opt} above 34 and 45 SOLANKI performs better on $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$, respectively. However for almost all values of n^{opt}, ENHANCED performs considerably better than the four other algorithms. If, for instance, we want a quality guarantee of $\alpha(OPS, IPS) \leq 0.1$, ENHANCED needs only 15 optimizations. SOLANKI, on the other hand, needs 56 optimizations and the other three algorithms need more than 70 optimizations. These figures show that our enhancements can improve the efficiency considerably. Furthermore, Figure 7.11 shows that not only the upper bound $\alpha(OPS, IPS)$ found by ENHANCED is better but also the true accuracy $\alpha(PS, IPS)$. Although a better $\alpha(OPS, IPS)$ generally implies a better $\alpha(PS, IPS)$, this is not always true as we can see when comparing the values of $\alpha(OPS, IPS)$ and $\alpha(PS, IPS)$ at $n^{opt} = 40$ for SOLANKI and CRAFT.

7.8.4 Test case 3: IMRT problem

As IMRT is one of the common application areas, we also include a test case from this field to compare the algorithms. The IMRT optimization problem used in this comparison is a 2D phantom pancreas case. Figure 7.13 shows the tumor and the five nearby organs. The high-energy photon beams used in radiation therapy to treat cancer tumors have

to pass through surrounding tissue to reach the tumor. To reduce the risk of damaging healthy tissue, the radiation dose delivered to this tissue should be minimized. The five organs indicated in Figure 7.13 are especially sensitive to radiation and are therefore referred to as the organs-at-risk (OARs). The radiation dose delivered to these OARs should be limited, while enough radiation should be delivered to the tumor to eradicate it. To calculate these doses, the relevant part of the body of the patient is discretized by dividing it into voxels. Using a dose influence matrix, the radiation dose delivered to each voxel can be calculated for a specific radiation plan. The objectives are often formulated in terms of the mean and maximum dose delivered to all voxels belonging to a tumor or OAR.

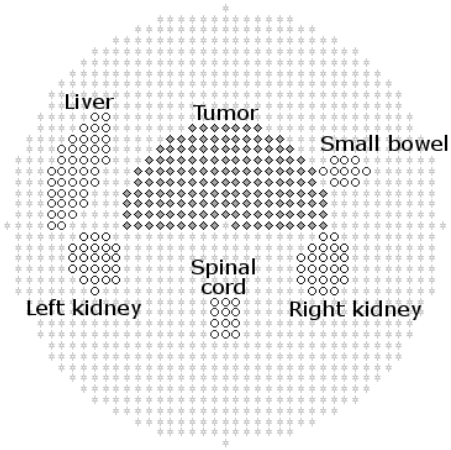


Figure 7.13: 2D pancreatic phantom case.

	Dose in Gy	
	Min.	Max.
Tumor	50	75
Left kidney		70
Right kidney		70
Spinal cord		45
Small bowel		75
Liver		75
Other tissue		75

Table 7.1: Minimal and maximal dose allowed for the tumor, OARs and other tissue.

In our problem, we consider four objectives. The first objective is aimed at delivering a dose of 60 Gray (Gy) to the tumor. Any voxel of the tumor that receives less than 60 Gy is penalized; the mean of these penalties forms the first objective. The three other objectives measure the maximum dose delivered to any of the voxels belonging to the left kidney, right kidney, or spinal cord, respectively. Furthermore, constraints are added to ensure a minimal or maximal dose for the OARs and other tissue. The values for these bounds can be found in Table 7.1. As all constraint and objectives are linear, this IMRT problem is a multi-objective LP problem and thus convex.

Figures 7.14 and 7.15 again contain the values of $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$. KLAMROTH$^{\leq}$ and KLAMROTH$^{=}$ are not used in this test case as they performed considerably worse than the other algorithms in the previous two test cases. For this IMRT case, CRAFT performs better than SOLANKI for all n^{opt}, but ENHANCED still performs better than the former two algorithms. The quality guarantee of $\alpha(OPS, IPS) \leq 0.1$ is reached by CRAFT after 42 optimizations whereas ENHANCED needs only 14 optimizations. After 24 optimizations, ENHANCED even gives a quality guarantee of $\alpha(OPS, IPS) \leq 0.05$.

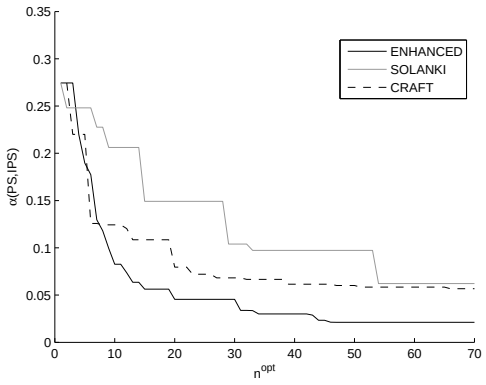


Figure 7.14: $\alpha(PS, IPS)$ of test case 3 as function of n^{opt}.

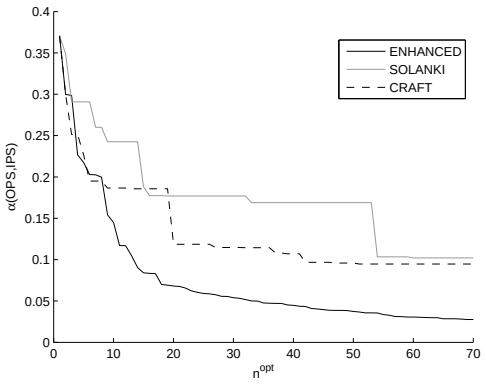


Figure 7.15: $\alpha(OPS, IPS)$ of test case 3 as function of n^{opt}.

7.8.5 Test case 4: geometric programming problem

In the fourth and final test case, we consider the effect of transforming convex objectives through a geometric programming example. For a general introduction to geometric programming, we refer to Boyd et al. (2007). Although we use only one example to show the effects of transforming the objectives, the transformation described below can be applied to any geometric program.

In this test case, we consider the following geometric programming problem:

$$\begin{aligned} f_1(x) &= e^{-x_1-x_2-x_3} \\ f_2(x) &= e^{x_4} \\ f_3(x) &= e^{x_5} \\ 2e^{-x_4+x_1}\left(e^{x_2}+e^{x_3}\right) &\leq 1 \\ e^{-x_5+x_2+x_3} &\leq 1 \\ e^{|x(2)-x(1)|} &\leq 2 \\ e^{|x(2)-x(3)|} &\leq 2 \\ f(x) &\leq \left[e^3\ e^3\ e^3\right]^\top \end{aligned}$$

This problem corresponds to Example 5 in Boyd et al. (2007) after applying the logarithmic change of variables and adding the upper bound $z^{ub} = [e^3\ e^3\ e^3]^\top$. We can easily show that this MOP is convex. When applying the transformation function $h(z) = log(z)$, the function $h \circ f$ becomes a linear function and thus remains convex. As $h(z) = log(z)$ is a strictly increasing and concave transformation, we can apply the results attained in Section 7.6. In this test case we did not use the transformation for the selection of facets. Furthermore, as this test case is intended to show the benefits of applying transformations and not to compare different algorithms, we used only the ENHANCED-algorithm.

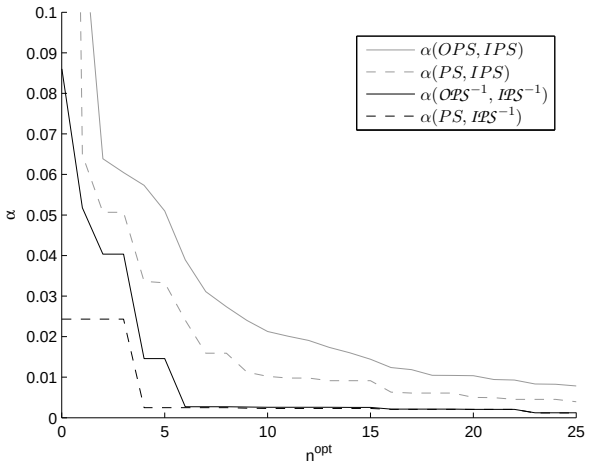


Figure 7.16: α-values of test case 4 as function of n^{opt}.

Figure 7.16 shows the values of $\alpha(OPS, IPS)$, $\alpha(PS, IPS)$, $\alpha(\mathcal{OPS}^{-1}, \mathcal{IPS}^{-1})$, and $\alpha(PS, \mathcal{IPS}^{-1})$ for different values of n^{opt}. Comparing $\alpha(OPS, IPS)$ with $\alpha(\mathcal{OPS}^{-1}, \mathcal{IPS}^{-1})$ and $\alpha(PS, IPS)$ with $\alpha(PS, \mathcal{IPS}^{-1})$ shows that both the upper bound and the real accuracy improves considerably by applying the transformation. At $n^{opt} = 4$, the real accuracy $\alpha(PS, \mathcal{IPS}^{-1})$ is already equal to 0.0025. After three more optimizations, the value of $\alpha(\mathcal{OPS}^{-1}, \mathcal{IPS}^{-1})$ is almost the same as the real accuracy. Without the transformations, more than 25 optimizations are needed to reach the same level of accuracy.

7.9 Conclusions and future research

In this chapter, we have introduced several enhancements to improve sandwich algorithms for approximating multi-dimensional convex Pareto sets. Firstly, dummy points were introduced to overcome the problem of 'undesirable' normals of IPS-facets. Adding these dummy points, we can determine also the set IPS^{nId} of non-IPS-dominated points more easily. Secondly, we introduced $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$, which can be used to determine when a set IPS^{nId} is both an ϵ-approximation and ε-optimal. As both concepts of ϵ-approximation and ε-optimal have a clear interpretation, these two measures provide quality guarantees that are easy to understand by a decision maker. Furthermore, we introduced a method to calculate $\alpha(PS, IPS)$ and $\alpha(OPS, IPS)$. This method simplifies the calculation of $\alpha(OPS, IPS)$ to solving a finite number of simple LP problems and thus improves the practical applicability of this error measure. Thirdly, we showed that transformations of the objective functions can improve OPS and IPS for certain convex MOPs and to extend the application of sandwich algorithm to certain non-convex MOPs. The calculation of the error measure when using these transformations was also discussed.

To test the benefits of these enhancements, we constructed the algorithm ENHANCED

by enhancing SOLANKI with dummy points and the error measure $\alpha(OPS, IPS)$. Three test cases showed a considerable efficiency improvement of ENHANCED compared with the other four tested methods. A fourth test case shows that using suitable transformations can still further improve the efficiency.

A limitation of these comparisons is that they consider only sandwich algorithms. Generally non-sandwich algorithms for approximating convex Pareto sets cannot provide quality guarantees, but they still can provide good approximations of the Pareto set. Therefore, it would be interesting to perform more extensive comparisons among different methods for approximating multi-dimensional convex Pareto sets.

Another interesting subject for further research would be to determine if the efficiency could be further improved by using a more interactive approach. Klamroth and Miettinen (2008) describe an approach where decision makers can refine their preferences to identify regions of PS where the approximation should be improved. A similar refinement might also be incorporated into the sandwich approaches by allowing the decision maker to change z^{ub}. However, more research is necessary to determine the effects and benefits of such an interactive approach.

Bibliography

Aarts, E.H.L. and J. Korst (1989). *Simulated annealing and Boltzmann machines: A stochastic approach to combinatorial optimization and neural computing*. New York, NY, USA: John Wiley & Sons, Inc.

Agca, S., B. Eksioglu, and J.B. Ghosh (2000). Lagrangian solution of maximum dispersion problems. *Naval Research Logistics*, **47(2)**, 97–114.

Alam, F.M., K.R. McNaught, and T.J. Ringrose (2004). A comparison of experimental designs in the development of a neural network simulation metamodel. *Simulation Modelling: Practice and Theory*, **12(7-8)**, 559–578.

Alexandrov, N.M., J.E. Dennis, R.M. Lewis, and V. Torczon (1998). A trust-region framework for managing the use of approximation models in optimization. *Structural and Multidisciplinary Optimization*, **15(1)**, 16–23.

Audze, P. and V. Eglais (1977). New approach for planning out of experiments. *Problems of Dynamics and Strengths*, **35**, 104–107.

Baer, D. (1992). Punktverteilungen in Würfeln beliebiger Dimension bezüglich der Maximum-norm. *Wissenschaftliche Zeitschrift der Pädagogischen Hochschule Erfurt/Mühlhausen, Mathematisch-Naturwissenschaftliche Reihe*, **28**, 87–92.

Banzhaf, W., F.D. Francone, R.E. Keller, and P. Nordin (1998). *Genetic Programming: An introduction on the automatic evolution of computer programs and its applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Barber, C.B., D.P. Dobkin, and H. Huhdanpaa (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, **22(4)**, 469–483.

Barthelemy, J.F.M. and R.T. Haftka (1993). Approximation concepts for optimum structural design – A review. *Structural and Multidisciplinary Optimization*, **5(3)**, 129–144.

Bates, R.A., R.J. Buck, E. Riccomagno, and H.P. Wynn (1996). Experimental design and observation for large systems. *Journal of the Royal Statistical Society: Series B*, **58**, 77–94.

Bates, S.J., J. Sienz, and V.V. Toropov (2004). Formulation of the optimal Latin hypercube design of experiments using a permutation genetic algorithm. In *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 1–7.

Beers, W.C.M. van and J.P.C. Kleijnen (2008). Customized sequential designs for random simulation experiments: Kriging metamodeling and bootstrapping. *European Journal of Operational Research*, **186(3)**, 1099–1113.

Bermond, J.C. and D. Sotteau (1976). Graph decompositions and G-designs. In Nash-Williams and Sheehan (Eds.), *Proceedings of the 5th British Combinatorial Conference 1975*, Winnipeg, Canada, 53–72. Utilitas Mathematica Publishing Inc.

Bisschop, J. and R. Entriken (1993). *AIMMS: The Modeling System*. Haarlem, The Netherlands: Paragon Decision Technology.

Booker, A.J., J.E. Dennis, P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset (1999). A rigorous framework for optimization of expensive functions by surrogates. *Structural and Multidisciplinary Optimization*, **17(1)**, 1–13.

Boyd, S., S.J. Kim, L. Vandenberghe, and A. Hassibi (2007). A tutorial on geometric programming. *Optimization and Engineering*, **8(1)**, 67–127.

Branke, J., K. Deb, K. Miettinen, and R. Slowinski (2008). *Multiobjective Optimization - Interactive and Evolutionary Approaches*. New York, NY, USA: Springer-Verlag.

Bulik, M., M. Liefvendahl, R. Stocki, and C. Wauquiez (2004). Stochastic simulation for crashworthiness. *Advances in Engineering Software*, **35(12)**, 791–803.

Bursztyn, D. and D.M. Steinberg (2006). Comparison of designs for computer experiments. *Journal of Statistical Planning and Inference*, **136(3)**, 1103–1119.

Campos, T.P.R. (2006). Computational simulations in medical radiation - a new approach to improve therapy. *Boletim da Sociedade Brasileira de Matemática VII*, **2**, 7–20.

Carlyle, W.M., J.W. Fowler, E.S. Gel, and B. Kim (2003). Quantitative comparison of approximate solution sets for bi-criteria optimization problems. *Decision Sciences*, **34(1)**, 63–82.

Chen, V.C.P., K.L. Tsui, R.R. Barton, and M. Meckesheimer (2006). A review on design, modeling and applications of computer experiments. *IIE Transactions*, **38(4)**, 273–291.

Cherkassky, V. and F. Mulier (1998). *Learning from data : Concepts, theory, and methods*. New York, NY, USA: John Wiley & Sons, Inc.

Cioppa, T.M. and T.W. Lucas (2007). Efficient nearly orthogonal and space-filling Latin hypercubes. *Technometrics*, **49(1)**, 45–55.

CPLEX (2005). *ILOG CPLEX 9.1 User's Manual*. Gentilly Cedex, France: ILOG S.A.

Craft, D.L., T.F. Halabi, H.A. Shih, and T.R. Bortfeld (2006). Approximating convex Pareto surfaces in multiobjective radiotherapy planning. *Medical Physics*, **33(9)**, 3399–3407.

Craft, D.L., T.F. Halabi, H.A. Shih, and T.R. Bortfeld (2007). An approach for practical multiobjective IMRT treatment planning. *International Journal of Radiation Oncology, Biology, Physics*, **69(5)**, 1600–1607.

Crary, S.B. (2002). Design of computer experiments for metamodel generation. *Analog Integrated Circuits and Signal Processing*, **32(1)**, 7–16.

Crary, S.B. (2008). WebDOE$^{\text{TM}}$. `http://www.webdoe.cc`, January 2008.

Crary, S.B., P. Cousseau, D. Armstrong, D.M. Woodcock, E.H. Mok, O. Dubochet, P. Lerch, and P. Renaud (2000). Optimal design of computer experiments for metamodel generation using I-OPT$^{\text{TM}}$. *Computer Modeling in Engineering & Sciences*, **1(1)**, 127–139.

Cressie, N.A.C. (1993). *Statistics for Spatial Data* (revised ed.), Volume 605. New York, NY, USA: John Wiley & Sons, Inc.

Cuyt, A. and B. Verdonk (1992). Multivariate rational data fitting: general data structure, maximal accuracy and object orientation. *Numerical Algorithms*, **3(1)**, 159–172.

Czyżak, P. and A. Jaszkiewicz (1998). Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 34–47.

Dam, E.R. van (2008). Two-dimensional minimax Latin hypercube designs. *Discrete Applied Mathematics*, **156(18)**, 3483–3493.

Dam, E.R. van, B.G.M. Husslage, and D. den Hertog (2009a). One-dimensional nested maximin designs. *Journal of Global Optimization*. To appear.

Dam, E.R. van, B.G.M. Husslage, D. den Hertog, and J.B.M. Melissen (2007). Maximin Latin hypercube designs in two dimensions. *Operations Research*, **55(1)**, 158–169.

Dam, E.R. van, G. Rennen, and B.G.M. Husslage (2009b). Bounds for maximin Latin hypercube designs. *Operations Research*, **57**, 595–608.

Das, I. (1999a). An improved technique for choosing parameters for Pareto surface generation using Normal-Boundary Intersection. In *Proceedings of WCSMO-3*, Volume 2, Buffalo, NY, USA, 411–413.

Das, I. (1999b). A preference ordering among various Pareto optimal alternatives. *Structural and Multidisciplinary Optimization*, **18(1)**, 30–35.

Das, I. and J.E. Dennis (1997). A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural and Multidisciplinary Optimization*, **14(1)**, 63–69.

Davis, G.J. and M.D. Morris (1997). Six factors which affect the condition number of matrices associated with Kriging. *Mathematical Geology*, **29**, 669–683.

Dimnaku, A., R. Kincaid, and M.W. Trosset (2005). Approximate solutions of continuous dispersion problems. *Annals of Operations Research*, **136(1)**, 65–80.

Dixon, L.C.W and G.P. Szegö (1978). The global optimization problem: An introduction. In L.C.W Dixon and G.P. Szegö (Eds.), *Toward Global Optimization*, Volume 2, 1–15. Amsterdam, The Netherlands: North-Holland.

Driessen, L.T. (2006). *Simulation-based optimization for product and process design*. Ph. D. thesis, CentER for Economic Research, Tilburg University, Tilburg, The Netherlands.

Driessen, L.T., H.P. Stehouwer, and J.J. Wijker (2002). Structural mass optimization of the engine frame of the Ariane 5 ESC-B. In *Proceedings of the European Conference on Spacecraft, Structures, Materials & Mechanical Testing*, Toulouse, 1–9. Toulouse, France.

Ehrgott, M. (2005). *Multicriteria Optimization*. Berlin, Germany: Springer-Verlag.

Ehrgott, M. and M.M. Wiecek (2005). Multiobjective Programming. In J. Figueira, S. Greco, and M. Ehrgott (Eds.), *Multiple Criteria Decision Analysis: State of the Art Surveys*, 667–722. New York, NY, USA: Springer-Verlag.

Erkut, E. (1990). The discrete p-dispersion problem. *European Journal of Operational Research*, **46(1)**, 48–60.

Erkut, E. and S. Neuman (1989). Analytical models for locating undesirable facilities. *European Journal of Operational Research*, **50**, 275–291.

Eskelinen, P., K. Miettinen, K. Klamroth, and J. Hakanen (2007). Interactive learning-oriented decision support tool for nonlinear multiobjective optimization: Pareto navigator. Technical report, Working Paper W-439, Helsinki School of Economics, Helsinki.

Evtushenko, Y.G. and M.A. Potapov (1987). Methods of numerical solution of multicriterion problem. *Soviet mathematics – doklady*, **34**, 420–423.

Fang, K.T., D.K.J. Lin, P. Winker, and Y. Zhang (2000). Uniform design: Theory and application. *Technometrics*, **42(3)**, 237–248.

Fang, K.T. and A. Sudjianto (2006). *Design and Modeling for Computer Experiments*. Boca Raton, FL, USA: Chapman & Hall/CRC.

Fejes Tóth, L. (1971). Punktverteilungen in einem Quadrat. *Studia Scientiarum Mathematicarum Hungarica*, **6**, 439–442.

Florian, A. (1989). Verteilung von Punkten in einem Quadrat. *Sitzungsberichte, Abteilung II, Österreichische Akademie der Wissenschaften, Mathematisch-Naturwissenschaftliche Klasse*, **198**, 27–44.

Fonseca, C.M. and P.J. Fleming (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, **3(1)**, 1–16.

Forrester, A.I.J., A.J. Keane, and N.W. Bressloff (2006). Design and analysis of "noisy" computer experiments. *AIAA journal*, **44(10)**, 2331–2339.

Forrester, A.I.J., A. Sóbester, and A.J. Keane (2007). Multi-fidelity optimization via surrogate modelling. In *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Volume 463, 3251–3269. The Royal Society.

Forrester, A.I.J., A. Sóbester, and A.J. Keane (2008). *Engineering Design via Surrogate Modelling: A Practical Guide*. Chichester, UK: Wiley.

Gambling, M., R.D. Jones, V.V. Toropov, and L.F. Alvarez (2001). Application of optimization strategies to problems with highly non-linear response. In *Proceedings of the 3rd ASMO-UK/ISSMO Conference on Engineering Design Optimization*, Harrogate, UK, 249–256.

Garishina, N.V. and C.J. Vladislavleva (2004). On development of a complexity measure for symbolic regression via genetic programming. Technical report, Mathematics for industry program of the Stan Ackermans Institute, Eindhoven, The Netherlands.

Ghosh, J.B. (1996). Computational aspects of the maximum diversity problem. *Operations Research Letters*, **19**, 175–181.

Giunta, A.A., S.F. Wojtkiewicz, and M.S. Eldred (2003). Overview of modern design of experiments methods for computational simulations. *AIAA 2003*, **649**, 1–17.

Goel, T., R.T. Haftka, W. Shyy, and L.T. Watson (2008). Pitfalls of using a single criterion for selecting experimental designs. *International Journal for Numerical Methods in Engineering*, **75(2)**, 127–155.

Golbraikh, A., M. Shen, Z. Xiao, Y.D. Xiao, K.H. Lee, and A. Tropsha (2003). Rational selection of training and test sets for the development of validated QSAR models. *Journal of Computer-Aided Molecular Design*, **17(2)**, 241–253.

Golbraikh, A. and A. Tropsha (2002). Predictive QSAR modeling based on diversity sampling of experimental datasets for the training and test set selection. *Journal of Computer-Aided Molecular Design*, **16(5–6)**, 357–369.

Golub, G.H. and C.F. van Loan (1996). *Matrix computations* (3th ed.). Baltimore, MD, USA: Johns Hopkins University Press.

Grosso, A., A.R.M.J.U. Jamali, and M. Locatelli (2009). Finding maximin Latin hypercube designs by Iterated Local Search heuristics. *European Journal of Operational Research*, **197(2)**, 541–547.

Haimes, Y.Y., L.S. Lasdon, and D.A. Wismer (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, **1(3)**, 296–297.

Hansen, P. and I.J. Moon (1994). Dispersing facilities on a network. *Cahiers du CERO*, **36**, 221–234.

Hardy, R.L. (1971). Multiquadratic equations of topography and other irregular surfaces. *Journal of Geophysical Research*, **76(8)**, 1905–1915.

Hedayat, A.S., N.A.J. Sloane, and J. Stufken (1999). *Orthogonal Arrays: Theory and Applications*. New York, NY, USA: Springer.

Heinrich, K. (1996). Graph decompositions and designs. In Colbourn and Dinitz (Eds.), *Handbook of Combinatorial Designs*, Boca Raton, FL, USA, 361–365. CRC Press.

Hertog, D. den and H.P. Stehouwer (2002). Optimizing color picture tubes by high-cost nonlinear programming. *European Journal of Operational Research*, **140(2)**, 197–211.

Hino, R., F. Yoshida, and V.V. Toropov (2006). Optimum blank design for sheet metal forming based on the interaction of high-and low-fidelity FE models. *Archive of Applied Mechanics*, **75(10)**, 679–691.

Hoffmann, A.L., D. den Hertog, A.Y.D. Siem, J.H.A.M. Kaanders, and H. Huizenga (2008). Convex reformulation of biologically-based multi-criteria intensity-modulated radiation therapy optimization including fractionation effects. *Physics in Medicine and Biology*, **53(22)**, 6345–6362.

Hoffmann, A.L., A.Y.D. Siem, D. den Hertog, J.H.A.M. Kaanders, and H. Huizenga (2006). Derivative-free generation and interpolation of convex Pareto optimal IMRT plans. *Physics in Medicine and Biology*, **51**, 6349–6369.

Husslage, B.G.M. (2006). *Maximin Designs for Computer Experiments*. Ph. D. thesis, CentER for Economic Research, Tilburg University, Tilburg, The Netherlands.

Husslage, B.G.M., E.R. van Dam, and D. den Hertog (2005). Nested maximin Latin hypercube designs in two dimensions. *CentER Discussion Paper 2005-79*, 1–11. Tilburg University, Tilburg, The Netherlands.

Husslage, B.G.M., E.R. van Dam, D. den Hertog, H.P. Stehouwer, and E.D. Stinstra (2003). Collaborative metamodeling: Coordinating simulation-based product design. *Concurrent Engineering: Research and Applications*, **11(4)**, 267–278.

Husslage, B.G.M., G. Rennen, E.R. van Dam, and D. den Hertog (2006). Space-filling Latin hypercube designs for computer experiments. *CentER Discussion Paper 2006-18*, 1–11. Tilburg University, Tilburg, The Netherlands.

Husslage, B.G.M., G. Rennen, E.R. van Dam, and D. den Hertog (2008). Space-filling Latin hypercube designs for computer experiments. *CentER Discussion Paper 2008-104*, 1–14. Tilburg University, Tilburg, The Netherlands.

Hwang, C.L. and A.S.M. Masud (1979). *Multiple Objective Decision Making-Methods and Applications: A State of the Art Survey*, Volume 164 of *Lecture Notes in Economics and Mathematical Systems*. Berlin, Germany: Springer-Verlag.

Jin, R., W. Chen, and T.W. Simpson (2001). Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization*, **23(1)**, 1–13.

Jin, R., W. Chen, and A. Sudjianto (2002). On sequential sampling for global metamodeling in engineering design. In *Proceedings of the ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 1–10. Montreal, Canada.

Jin, R., W. Chen, and A. Sudjianto (2005). An efficient algorithm for constructing optimal design of computer experiments. *Journal of Statistical Planning and Inference*, **134(1)**, 268–287.

Jin, R., X. Du, and W. Chen (2003). The use of metamodeling techniques for optimization under uncertainty. *Structural and Multidisciplinary Optimization*, **25(2)**, 99–116.

Jin, Y. and B. Sendhoff (2004). Constructing dynamic optimization test problems using the multi-objective optimization concept. In G. Raidl et al. (Eds.), *Applications of Evolutionary Algorithms*, Volume 3005 of *LNCS*, 525–536. Springer.

Johnson, M.E., L.M. Moore, and D. Ylvisaker (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, **26**, 131–148.

Jones, D.R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, **21(4)**, 345–383.

Jones, D.R., M. Schonlau, and W.J. Welch (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, **13(4)**, 455–492.

Karasakal, E. and M. Koksalan (2009). Generating a representative subset of the nondominated frontier in multiple criteria decision making. *Operations Research*, **57(1)**, 187–199.

Keijzer, M. (2003). Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. In C. Ryan, Soule T., M. Keijzer, E. Tsang, R. Poli, and E. Costa (Eds.), *Genetic Programming, Proceedings of EuroGP 2003*, Volume 2610 of *LNCS*, Berlin, Germany, 71–83. Springer-Verlag.

Kennedy, MC and A. O'Hagan (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, **87(1)**, 1–13.

Kirchner, K. and G. Wengerodt (1987). Die dichteste Packung von 36 Kreisen in einem Quadrat. *Beiträge zur Algebra und Geometrie*, **25**, 147–159.

Klamroth, K. and K. Miettinen (2008). Integrating approximation and interactive decision making in multicriteria optimization. *Operations Research*, **56(1)**, 222–234.

Klamroth, K., J. Tind, and M.M. Wiecek (2002). Unbiased approximation in multicriteria optimization. *Mathematical Methods of Operations Research (ZOR)*, **56(3)**, 413–437.

Kleijen, J.P.C. (1999). Validation of models: statistical techniques and data availability. In P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans (Eds.), *Proceedings of the 1999 Winter Simulation Conference*, Volume 1, 647–654.

Kleijnen, J.P.C. (2005). Supply chain simulation tools and techniques: A survey. *International Journal of Simulation and Process Modelling*, **1(1)**, 82–89.

Kleijnen, J.P.C. (2008). *Design and Analysis of Simulation Experiments*. Springer.

Kleijnen, J.P.C. (2009). *Kriging metamodeling in simulation: A review*, Volume 192.

Kleijnen, J.P.C. and W.C.M. van Beers (2004). Application-driven sequential designs for simulation experiments: Kriging metamodelling. *Journal of the Operational Research Society*, **55(8)**, 876–883.

Kleijnen, J.P.C., G. van Ham, and J. Rotmans (1992). Techniques for sensitivity analysis of simulation models: A case study of the CO2 greenhouse effect. *Simulation*, **58(6)**, 410–417.

Kleijnen, J.P.C. and R.G. Sargent (2000). A methodology for fitting and validating metamodels in simulation. *European Journal of Operational Research*, **120**, 14–29.

Koehler, J.R. and A.B. Owen (1996). Computer experiments. In S. Ghosh and C.R. Rao (Eds.), *Design and analysis of experiments*, Volume 13 of *Handbook of Statistics*, 261–308. Amsterdam, The Netherlands: North-Holland.

Kordon, A. (2006). Evolutionary computation at Dow Chemical. *SIGEVOlution*, **1(3)**, 4–9.

Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press.

Krige, D.G. (1951). A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, **52(6)**, 119–139.

Kuo, C.C., F. Glover, and K.S. Dhir (1993). Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Sciences*, **24**, 1171–1185.

Lam, R.L.H., W.J. Welch, and S.S. Young (2002). Uniform coverage designs for molecule selection. *Technometrics*, **44**, 99–109.

Liefvendahl, M. and R. Stocki (2006). A study on algorithms for optimization of Latin hypercubes. *Journal of Statistical Planning and Inference*, **136(9)**, 3231–3247.

Locatelli, M. and U. Raber (2002). Packing equal circles in a square: A deterministic global optimization approach. *Discrete Applied Mathematics*, **122(1–3)**, 139–166.

Lophaven, S.N., H.B. Nielsen, and J. Sondergaard (2002). DACE: A Matlab Kriging toolbox version 2.0. Technical Report IMM-TR-2002-12, Technical Univeristy of Denmark, Copenhagen.

Loridan, P. (1984). ε-Solutions in vector minimization problems. *Journal of Optimization Theory and Applications*, **43(2)**, 265–276.

Markót, M.C. and T. Csendes (2005). A new verified optimization technique for the "packing circles in a unit square" problems. *SIAM Journal on Optimization*, **16(1)**, 193–219.

Marler, R.T. and J.S. Arora (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, **26(6)**, 369–395.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, **58(8)**, 1246–1266.

Mathews, J.H. and K.D. Fink (2004). *Numerical Methods Using Matlab* (4th ed.). Upper Saddle River, NJ, USA: Prentice-Hall Inc.

McKay, M.D., R.J. Beckman, and W.J. Conover (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, **21(2)**, 239–245.

Melissen, J.B.M. (1997). *Packing and Covering with Circles*. Ph. D. thesis, Utrecht University, Utrecht, The Netherlands.

Messac, A., A. Ismail-Yahaya, and C.A. Mattson (2003). The normalized normal constraint method for generating the Pareto frontier. *Structural and Multidisciplinary Optimization*, **25(2)**, 86–98.

Messac, A. and C.A. Mattson (2002). Generating well-distributed sets of Pareto points for engineering design using physical programming. *Optimization and Engineering*, **3(4)**, 431–450.

Messac, A. and C.A. Mattson (2004). Normal constraint method with guarantee of even representation of complete Pareto frontier. *AIAA Journal*, **42(10)**, 2101–2111.

Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Boston, MA, USA: Kluwer Academic Publishers.

Miller, G.A. (1956). The magical number seven plus or minus two: some limits on our capacity for processing information. *Psychological review*, **63(2)**, 81.

Montgomery, D.C. (2009). *Design and Analysis of Experiments* (7th ed.). New York, NY, USA: John Wiley & Sons, Inc.

Monz, M. (2006). *Pareto Navigation - Interactive multiobjective optimisation and its application in radiotherapy planning*. Ph. D. thesis, Department of Mathematics, University of Kaiserslautern, Kaiserslautern, Germany.

Morris, M.D. and T.J. Mitchell (1995). Exploratory designs for computer experiments. *Journal of Statistical Planning and Inference*, **43**, 381–402.

Myers, R.H. (1999). Response surface methodology – Current status and future directions. *Journal of Quality Technology*, **31**, 30–74.

Nguyen, T.A. (2007). Application of optimization methods to controller design for active suspensions. *Mechanics Based Design of Structures and Machines*, **35(3)**, 291–318.

Nurmela, K.J. and P.R.J. Östergård (1999). More optimal packings of equal circles in a square. *Discrete and Computational Geometry*, **22**, 439–547.

Oden, J.T., T. Belytschko, J. Fish, T.J.R. Hughes, C. Johnson, D. Keyes, A. Laub, L. Petzold, D. Srolovitz, and S. Yip (2006). Simulation-Based Engineering Science: Revolutionizing Engineering Science through Simulation. Technical report, Report of the National Science Foundation Blue Ribbon Panel on Simulation-Based Engineering Science.

Oler, N. (1961). A finite packing problem. *Canadian Mathematical Bulletin*, **4(2)**, 153–155.

Owen, A.B. (1992). Orthogonal arrays for computer experiments, integration and visualization. *Statistica Sinica*, **2**, 439–452.

Padula, S.L., N. Alexandrov, and L.L. Green (1996). MDO test suite at NASA Langley Research Center. *6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 410–420.

Palmer, K. and K.L. Tsui (2001). A minimum bias Latin hypercube design. *IIE Transactions*, **33(9)**, 793–808.

Peikert, R., D. Würtz, M. Monagan, and C. den Groot (1991). Packing circles in a sphere: A review and new results. In *Proceedings of the 15th IFIP Conference on System Modeling and Optimization, Springer Lecture Notes in Control and Information Sciences*, Volume 180 of *Springer Lecture Notes in Control and Information Sciences*, 111–124.

Pisinger, D. (2006). Upper bounds and exact algorithms for p-dispersion problems. *Computers and Operations Research*, **33(5)**, 1380–1398.

Powell, M.J.D. (1987). Radial basis functions for multivariable interpolation: A review. *Clarendon Press Institute of Mathematics and its Applications Conference Series*, 143–167.

Qian, Z., C.C. Seepersad, V.R. Joseph, J.K. Allen, and C.F.J. Wu (2006). Building surrogate models based on detailed and approximate simulations. *Journal of Mechanical Design*, **128(4)**, 668–677.

Queipo, N.V., R.T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P.K. Tucker (2005). Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, **41(1)**, 1–28.

Ravi, S.S., D.J. Rosenkrantz, and G.K. Tayi (1991). Facility dispersion problems: Heuristics and special cases (extended abstract). In *Algorithms and Data Structures, 2nd Workshop WADS '91, Ottawa, Canada, August 14-16*, 355–366.

Ravi, S.S., D.J. Rosenkrantz, and G.K. Tayi (1994). Heuristic and special case algorithms for dispersion problems. *Operations Research*, **42**, 299–310.

Reuter, H. (1990). An approximation method for the efficiency set of multiobjective programming problems. *Optimization*, **21(6)**, 905–911.

Rikards, R. and J. Auzins (2004). Response surface method for solution of structural identification problems. *Inverse Problems in Engineering*, **12(1)**, 59–70.

Rikards, R., A. Chate, and G. Gailis (2001). Identification of elastic properties of laminates based on experiment design. *International Journal of Solids and Structures*, **38(30–31)**, 5097–5115.

Rockafellar, R.T. and R.J.B. Wets (1998). *Variational Analysis*. Berlin, Germany: Springer-Verlag.

Romeijn, H.E. and J. Dempsey (2008). Intensity modulated radiation therapy treatment plan optimization. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, **16(2)**, 215–243.

Romeijn, H.E., J.F. Dempsey, and J.G. Li (2004). A unifying framework for multicriteria fluence map optimization models. *Physics in Medicine and Biology*, **49(10)**, 1991–2013.

Ruhe, G. and B. Fruhwirth (1990). ε-Optimality for bicriteria programs and its application to minimum cost flows. *Computing*, **44(1)**, 21–34.

Ruzika, S. and M.M. Wiecek (2003). A survey of approximation methods in multiobjective programming. *Journal of Optimization Theory and Applications*, **126(3)**, 473–501.

Saaty, T.L. and M.S. Ozdemir (2003). Why the magic number seven plus or minus two. *Mathematical and Computer Modelling*, **38(3-4)**, 233–244.

Sacks, J., S.B. Schiller, and W.J. Welch (1989a). Designs for computer experiments. *Technometrics*, **31**, 41–47.

Sacks, J., W.J. Welch, T.J. Mitchell, and H.P. Wynn (1989b). Design and analysis of computer experiments. *Statistical Science*, **4**, 409–435.

Santner, Th.J., B.J. Williams, and W.I. Notz (2003). *The Design and Analysis of Computer Experiments*. Springer Series in Statistics. New York, Ny, USA: Springer-Verlag.

Schandl, B., K. Klamroth, and M.M. Wiecek (2002). Norm-based approximation in multicriteria programming. *Computers and Mathematics with Applications*, **44(7)**, 925–942.

Shao, L. and M. Ehrgott (2008). Approximately solving multiobjective linear programmes in objective space and an application in radiotherapy treatment planning. *Mathematical Methods of Operations Research*, **68(2)**, 257–276.

Siem, A.Y.D. and D. den Hertog (2007). Kriging models that are robust with respect to simulation errors. *CentER Discussion Paper 2007-68*, 1–28. Tilburg University, Tilburg, The Netherlands.

Siem, A.Y.D., D. den Hertog, and A.L. Hoffmann (2008). The effect of transformations on the approximation of univariate (convex) functions with applications to Pareto curves. *European Journal of Operational Research*, **189(2)**, 347–362.

Simpson, T.W., A.J. Booker, D. Ghosh, A.A. Giunta, P.N. Koch, and R.-J. Yang (2004). Approximation methods in multidisciplinary analysis and optimization: A panel discussion. *Structural and Multidisciplinary Optimization*, **27(5)**, 302–313.

Simpson, T.W., D.K.J. Lin, and W. Chen (2001). Sampling strategies for computer experiments: Design and analysis. *International Journal of Reliability and Applications*, **2(3)**, 209–240.

Simpson, T.W., J. Peplinski, P.N. Koch, and J.K. Allen (2001). Metamodels for computer-based engineering design: Survey and recommendations. *Engineering with Computers*, **17**, 129–150.

Simpson, T.W., V.V. Toropov, V. Balabanov, and F.A.C. Viana (2008). Design and Analysis of Computer Experiments in Multidisciplinary Design Optimization: A Review. In *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 1–22.

Smith, K.I., R.M. Everson, and J.E. Fieldsend (2004). Dominance Measures for Multi-Objective Simulated Annealing. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, 23–30. IEEE Press.

Smits, G.F. and M. Kotanchek (2004). Pareto-Front Exploitation in Symbolic Regression. In U. O'Reilly, T. Yu, R.L. Riolo, and B. Worzel (Eds.), *Genetic Programming Theory and Practice II*, Chapter 17, 283–300. Ann Arbor, MI, USA: Kluwer Academic Publishers.

Sobieszczanski-Sobieski, J. and R.T. Haftka (1997). Multidisciplinary aerospace design optimization: Survey of recent developments. *Structural and Multidisciplinary Optimization*, **14(1)**, 1–23.

Solanki, R.S., P.A. Appino, and J.L. Cohon (1993). Approximating the noninferior set in multiobjective linear programming problems. *European Journal of Operational Research*, **68(3)**, 356–373.

Specht, E. (2008). Packomania. `http://www.packomania.com`, January 2008.

Srivastava, A., K. Hacker, K. Lewis, and T.W. Simpson (2004). A method for using legacy data for metamodel-based design of large-scale systems. *Structural and Multidisciplinary Optimization*, **28**, 145–155.

Stehouwer, H.P. and D. den Hertog (1999). Simulation-based design optimisation: Methodology and applications. *Proceedings of the 1st ASMO-UK/ISSMO Conference on Engineering Design Optimization*, 349–355. Ilkley, United Kingdom.

Stein, M. (1987). Large sample properties of simulations using Latin hypercube sampling. *Technometrics*, **29(2)**, 143–151.

Steuer, R.E. (1986). *Multiple Criteria Optimization: Theory and Application*. New York, NY, USA: John Wiley & Sons, Inc.

Stewart, T., O. Bandte, H. Braun, N. Chakraborti, M. Ehrgott, M. Göbelt, Y. Jin, H. Nakayama, S. Poles, and D. Stefano (2008). Real-World Applications of Multiobjective Optimization. In J. Branke, K. Deb, K. Miettinen, and R. Slowinski (Eds.), *Multiobjective Optimization - Interactive and Evolutionary Approaches*, 285–327. New York, NY, USA: Springer-Verlag.

Stinstra, E.D. (2006). *The meta-model Approach for simulation-based design optimization*. Ph. D. thesis, CentER for Economic Research, Tilburg University, Tilburg, The Netherlands.

Stinstra, E.D., D. den Hertog, H.P. Stehouwer, and A. Vestjens (2003). Constrained maximin designs for computer experiments. *Technometrics*, **45(4)**, 340–346.

Stocki, R. (2005). A method to improve design reliability using optimal Latin hypercube sampling. *Computer Assisted Mechanics and Engineering Sciences*, **12(4)**, 393–412.

Tang, B. (1993). Orthogonal array-based Latin hypercubes. *Journal of the American Statistical Association*, **88**, 1392–1397.

Trosset, M.W. (1999). Approximate maximin distance designs. In *Proceedings of the Section on Physical and Engineering Sciences*, Alexandria, VA, USA, 223–227.

Vapnik, V., S.E. Golowich, and A. Smola (1997). Support vector method for function approximation, regression estimation, and signal processing. In *Advances in Neural Information Processing Systems 9*, Cambridge, MA, USA, 281–287. MIT Press.

Viana, F.A.C., V. Balabanov, G. Venter, J. Garcelon, and V. Steffen (2007). Generating optimal Latin hypercube designs in real time. In *7th World Congress on Structural and Multidisciplinary Optimization*, 2310–2315.

Volgenant, A. (1990). Symmetric traveling salesman problems. *European Journal of Operational Research*, **49(1)**, 153–154.

Volgenant, A. and R. Jonker (1982). A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. *European Journal of Operational Research*, **9**, 83–89.

Wang, G., Z. Dong, and P. Aitchison (2001). Adaptive response surface method - A global optimization scheme for approximation-based design problems. *Journal of Engineering Optimization*, **33**, 707–734.

Wang, G.G. and S. Shan (2007). Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, **129(4)**, 370–380.

White, D.J. (1990). A bibliography on the applications of mathematical programming multiple-objective methods. *Journal of the Operational Research Society*, **41(8)**, 669–691.

Wilson, R.M. (1976). Decompositions of complete graphs into subgraphs isomorphic to a given graph. In Nash-Williams and Sheehan (Eds.), *Proceedings of the 5th British Combinatorial Conference 1975*, Winnipeg, Canada, 647–659. Utilitas Mathematica Publishing Inc.

Ye, K.Q., W. Li, and A. Sudjianto (2000). Algorithmic construction of optimal symmetric Latin hypercube designs. *Journal of Statistical Planning and Inference*, **90(1)**, 145–159.

Zadeh, L. (1963). Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, **8(1)**, 59–60.

Zheng, X, H. Yu, and A. Atkins (2008). An Overview of Simulation in Supply Chains. In *Advanced Design and Manufacture to Gain a Competitive Edge*, 407–416. London, UK: Springer.

Zitzler, E., L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, **7(2)**, 117–132.

Samenvatting

> Ja, ik ben gek op het citeren van mensen. Of zoals president Kennedy al zei: "Was het Homerus niet die schreef: 'Originaliteit is het mooiste wat er is"'.
>
> (Herman Finkers)

Computers zijn in de afgelopen decennia steeds krachtiger geworden en daardoor in staat steeds complexere problemen op te lossen. Ondanks de toegenomen rekenkracht, vragen bepaalde problemen nog steeds erg veel rekentijd. Twee voorbeelden zijn optimalisatieproblemen waarbij complexe simulatiemodellen of meerdere doelstellingsfuncties een rol spelen. Bij het eerste type problemen valt bijvoorbeeld te denken aan het optimaliseren van de veiligheid van een auto met behulp van een simulatiemodel dat de effecten van een botsing berekent. Als daarnaast ook het optimaliseren van het brandstofverbruik een doelstelling is, is er sprake van het tweede type problemen. Aangezien beide typen problemen erg tijdrovend kunnen zijn, is het vaak moeilijk om een optimale of goede oplossing te vinden binnen een redelijke tijd. Het centrale onderwerp van dit proefschrift is daarom het ontwikkelen en verbeteren van methodes die op een efficiënte manier omgaan met deze optimalisatieproblemen. Ondanks dat de gebruikte technieken voor deze problemen verschillend zijn, hebben ze een belangrijke overeenkomst. Bij beide typen problemen wordt een benadering bepaald aan de hand van datapunten die tijdrovend zijn om te berekenen.

In het geval van complexe simulatiemodellen benaderen we de black-boxfunctie die de relatie tussen input- en outputvariabelen van het simulatiemodel beschrijft. We beschouwen deze relatie als een black-boxfunctie omdat er geen expliciete wiskundige beschrijving bekend is, maar er wel uitkomsten (output) berekend kunnen worden voor bepaalde waarden van de invoervariabelen (input). We benaderen deze black-boxfunctie met een zogeheten "metamodel". Dit metamodel heeft wel een expliciete wiskundige beschrijving en kan zodoende gebruikt worden om de uitkomsten te optimaliseren of om inzicht te krijgen in de relatie tussen de invoer en uitkomst. Voor het bepalen van een metamodel zijn datapunten nodig die de outputwaarden bevatten bij bepaalde waarden van de in-

putvariabelen. Wanneer voor het bepalen van de outputwaarde een complexe simulatie uitgevoerd moet worden, kan het berekenen van deze datapunten erg tijdrovend zijn. Het is daarom belangrijk deze datapunten zo efficiënt mogelijk te kiezen.

Bij multi-objective optimalisatieproblemen benaderen we de onbekende verzameling van Pareto-oplossingen, d.w.z. oplossingen waarvoor het niet mogelijk is de waarde van een doelstellingsfunctie te verbeteren zonder andere te verslechteren. Het kiezen van een oplossing die niet aan deze eigenschap voldoet is sub-optimaal omdat we in dat geval een doelstelling zouden kunnen verbeteren zonder dat dit ten koste gaat van anderen. Het bepalen van de verzameling van Pareto-oplossingen is daarom een belangrijke stap in het oplossen van een multi-objective optimalisatieprobleem. Het bepalen van Pareto-oplossingen vereist echter het formuleren en oplossen van een, vaak tijdrovend, optimalisatieprobleem waardoor het bepalen van de complete verzameling meestal niet haalbaar is. In plaats daarvan kan een benadering van deze verzameling bepaald worden waaruit een goede oplossing geselecteerd kan worden.

Aangezien het berekenen van de datapunten tijdrovend is voor beide typen problemen, proberen we met zo min mogelijk datapunten een goede benadering te bepalen van de black-boxfunctie of de verzameling van Pareto-oplossingen. In dit proefschrift ligt daarom de nadruk op de efficiënte selectie van simulaties of optimalisaties voor het bepalen van de datapunten. Door deze efficiënt te kiezen, streven we dus naar een kleiner aantal benodigde datapunten voor het verkrijgen van een nauwkeurige benadering.

Bij de selectie van datapunten voor de constructie van een metamodel maken we gebruik van een zogeheten "design of computer experiments". Dit is een schema dat bepaalt welke datapunten gesimuleerd gaan worden. Aangezien de keuze van het schema invloed heeft op de kwaliteit van het metamodel, houden we ons in hoofdstukken 2 tot en met 5 van dit proefschrift bezig met het bepalen van schema's met verscheidene gunstige eigenschappen.

In hoofdstuk 2 beschouwen we de zogeheten "space-filling Latin hypercube designs" (LHDs). Schema's uit deze klasse hebben twee belangrijke eigenschappen. Ten eerste hebben LHDs de eigenschap dat voor elke inputvariabele geldt dat alle datapunten een andere waarde hebben. Hiermee wordt voorkomen dat verschillende datapunten elkaar overlappen als een of meerdere inputvariabelen geen (beduidende) invloed blijken te hebben op de outputvariabele. Ten tweede zijn deze schema's space-filling wat wil zeggen dat de datapunten gelijkmatig verspreid zijn over de gehele inputruimte. De mate van spreiding wordt in dit hoofdstuk gemeten met de ℓ^2-maximin en Audze-Eglais criteria. Gebruikmakend van periodieke designs en het ESE algoritme bepalen we space-filling LHDs voor maximaal 10 inputvariabelen en 300 datapunten.

Hoofdstuk 3 focust op de spreidingsmaten ℓ^2-, ℓ^1- en ℓ^∞-maximin. Om zo space-filling mogelijke LHDs te vinden, wordt vaak geprobeerd deze maten te maximaliseren. Naar-

mate het aantal inputvariabelen en het aantal datapunten toeneemt, wordt het echter steeds moeilijker een LHD te vinden dat één van deze maten maximaliseert. In hoofdstuk 3 bepalen we daarom bovengrenzen voor de maximale waardes die deze maten kunnen aannemen. Deze bovengrenzen bepalen we voor LHDs met verschillende aantallen variabelen en datapunten. Afhankelijk van deze eigenschappen en de spreidingsmaat worden verschillende technieken en formuleringen gebruikt om goede bovengrenzen te vinden. Voorbeelden hiervan zijn Mixed Integer Programming, het Traveling Salesman Probleem en het Graph Covering Probleem.

In bepaalde situaties hebben we niet één schema nodig, maar twee schema's met verschillende aantallen datapunten. Wanneer het kleine schema een deelverzameling vormt van het grote schema, spreken we van een genest schema. Deze geneste schema's kunnen bijvoorbeeld worden gebruikt wanneer er twee simulatiemodellen beschikbaar zijn die eigenschappen van hetzelfde product of proces beschrijven maar met verschillende mates van nauwkeurigheid. Aangezien het minder nauwkeurige simulatiemodel meestal minder rekentijd vergt, kunnen hiermee meer datapunten doorgerekend worden dan met het nauwkeurigere model. Wanneer bepaalde datapunten door beide modellen worden doorgerekend, kunnen de resultaten van de modellen gecombineerd worden om zo een beter metamodel te verkrijgen dan mogelijk zou zijn met slecht één simulatiemodel. Naast de geneste eigenschap willen we ook graag dat beide schema's een space-filling LHD vormen. In hoofdstuk 4 beschrijven we daarom verschillende methodes om geneste space-filling LHDs te bepalen. Afhankelijk van het aantal datapunten in beide schema's is het helaas niet altijd mogelijk beide schema's een LHD te laten zijn. We introduceren daarom drie alternatieven waarbij de LHD-structuur zoveel mogelijk intact word gelaten. Voor alle drie de alternatieven worden geneste space-filling LHDs bepaald met maximaal 50 datapunten in het kleine schema en 60 in het grote schema. De gevonden schema's en bovengrenzen uit de hoofdstukken 2, 3 en 4 zijn beschikbaar op `http://www.spacefillingdesigns.nl`.

In hoofdstuk 5 gaan we uit van een andere situatie dan in de voorgaande hoofdstukken. We bekijken hierin namelijk de situatie waarin een dataset beschikbaar is met de input- en outputwaarden van een groot aantal simulaties of experimenten. Een grote dataset is normaal gesproken een goed uitgangspunt voor het schatten van een nauwkeurig metamodel. Wanneer de Kriging-methode gebruikt wordt om een metamodel te bepalen, kan een grote dataset echter ook nadelen hebben. Met name als de datapunten in de dataset niet gelijkmatig verspreid zijn over de inputruimte, kan dit de kwaliteit van het Kriging-model negatief beïnvloeden. In hoofdstuk 5 laten we zien dat het soms beter is slechts een deel van de dataset te gebruiken in plaats van de complete dataset. Ook hier proberen we de datapunten zo te kiezen dat ze gelijkmatig verspreid liggen over de inputruimte. Aangezien de datapunten waaruit we kunnen kiezen vastliggen en de outputwaarden bekend zijn, gebruiken we hiervoor andere methodes dan voor het bepalen van space-

filling LHDs. Vijf bestaande methodes en drie varianten van een nieuwe methode worden vergeleken op vier soorten datasets. De vergelijking betreft nauwkeurigheid, robuustheid, kans op numerieke onnauwkeurigheid, tijd voor het bepalen van de subset en tijd voor het schatten van het Kriging-model. Aangezien geen methode het beste presteert op alle criteria, hangt het van de situatie en gebruiker af welke methode het meest geschikt is.

Naast Kriging bestaan er nog vele andere technieken om een metamodel te schatten. In hoofdstuk 6 beschouwen we "symbolic regression". Deze techniek heeft het voordeel dat er zeer weinig eisen opgelegd worden aan de vorm van het metamodel, wat de mogelijkheid geeft om nauwkeurigere en beter interpreteerbare modellen te vinden. Een nadeel is echter dat de modellen ook te ingewikkeld kunnen worden of ruis in de data proberen te verklaren. Om dit te voorkomen, introduceren we een maat voor de complexiteit van een functie. Deze maat is gebaseerd op het idee dat de complexiteit gecorreleerd is met de minimale graad van een polynoom die nodig is om een functie met een bepaalde nauwkeurigheid te benaderen. Met behulp van Pareto simulated annealing proberen we metamodellen te vinden die een goede afweging vormen tussen complexiteit en nauwkeurigheid.

In het laatste hoofdstuk van dit proefschrift behandelen we het benaderen van de verzameling van Pareto-oplossingen. Aanleiding voor dit onderzoek is het probleem van het bepalen van goede bestralingsplannen voor tumoren. Hierbij moet steeds een afweging gemaakt worden tussen de kans op succesvolle verwijdering van de tumor en het risico op beschadiging van gezond weefsel. Dit afwegingsprobleem kan geformuleerd worden als een convex multi-objective optimalisatieprobleem. Wanneer er geen oplossing bestaat die alle doelstellingen tegelijk optimaliseert, zal er een goed compromis bepaald moeten worden. Een benadering van de verzameling van Pareto-oplossingen kan de arts in deze keuze ondersteunen. Voor convexe multi-objective optimalisatieproblemen is het mogelijk om door middel van zogenaamde sandwich-algoritmes benaderingen te vinden waarvoor garanties gegeven kunnen worden voor de nauwkeurigheid. In hoofdstuk 7 introduceren we verschillende verbeteringen voor bestaande sandwich-algoritmes. Ten eerste voegen we dummy-punten toe aan de benadering om beter te kunnen bepalen welke optimalisaties we moeten uitvoeren in verschillende stappen van de sandwich-algoritmes. Ten tweede beschrijven we een foutmaat die ons in staat stelt goed interpreteerbare kwaliteitsgaranties te geven voor de nauwkeurigheid van de benadering. We beschrijven ook hoe deze foutmaat eenvoudig te berekenen is met behulp van de bovengenoemde dummy-punten. Ten derde laten we zien hoe transformaties van doelstellingsfuncties de nauwkeurigheid van de benadering kunnen vergroten. Door deze transformaties wordt het verder ook mogelijk sandwich-algoritmes toe te passen bij bepaalde niet-convexe doelstellingsfuncties. Om het effect van deze verbeteringen te testen, vergelijken we bestaande sandwich-algoritmes met een nieuw algoritme waarin bovenstaande verbeteringen zijn

toegepast. Vergelijking aan de hand van vier testcases laat zien dat het nieuwe algoritme inderdaad aanzienlijk efficiënter is, d.w.z. minder tijdrovende optimalisaties nodig heeft om eenzelfde (gegarandeerde) nauwkeurigheid te bereiken.