Maximin designs for computer experiments

Maximin designs for computer experiments

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit van Tilburg, op gezag van de rector magnificus, prof. dr. F.A. van der Duyn Schouten, in het openbaar te verdedigen ten overstaan van een door het college voor promoties aangewezen commissie in de aula van de Universiteit op vrijdag 17 november 2006 om 14.15 uur door

Bartholomeus Gerardus Martinus Husslage

geboren op 6 juli 1980 te Breda.

Promotores:	prof. dr. ir. D. den Hertog
	prof. dr. E.H.L. Aarts
Copromotor:	dr. ir. E.R. van Dam

The research of B.G.M. Husslage has been financially supported by the SamenwerkingsOrgaan Brabantse Universiteiten (SOBU).

From the start to the end
no matter what I pretend
the journey is more important than the end or the start
and what it meant to me
will eventually be a memory
of the time when I tried so hard...

(Linkin Park, *Enth E Nd*)

Preface

> The truth of a situation most often turns out to be that one with the simplest explanation.
>
> (Terry Goodkind, *Faith of the Fallen*)

Although there is only a single name printed on the front cover, the writing of this thesis would not have been possible without the help of several people.

The many brainstorming sessions I have had with Edwin van Dam and Dick den Hertog have not only been very motivating, they have been fun as well. I would like to thank them for their guidance and enthusiasm during these four years. I would also like to thank Emile Aarts for his advice and the inspiring discussions we have had. My research position has been financially supported by the SamenwerkingsOrgaan Brabantse Universiteiten (SOBU), for which I am grateful.

Several papers and ideas in this thesis are a result of collaborations with fellow researchers. I would like to thank Hans Melissen, János Pintér, Gijs Rennen, Peter Stehouwer, and Erwin Stinstra, for the fruitful cooperation. Furthermore, I would like to thank Jack Kleijnen, Hans Melissen, Dolf Talman, and Vassili Toropov, for joining Dick, Edwin, and Emile, in my thesis committee. Special thanks go to Els Kiewied, who has been so kind as to proofread this thesis and whose suggestions have led to several improvements to the text.

I would like to thank the members of the Department of Econometrics & Operations Research, and in particular the people of the "lunchtafelgroep", for creating a pleasant working environment in which there was always time for a joke or two.

Finally, I am very grateful for having always been surrounded by a wonderful family and good friends and I would like to thank them for all the good times we have had and (hopefully) will have. My last word of thanks goes to the most important people in my life: my parents, brothers, and sister, for their constant love and support.

Contents

Chapter 1

Introduction

> Don't you watch television?
> I thought all children despise effort and enjoy cartoons.
>
> (BtVS, *Episode 05.17*)

1.1 Simulation-based product design

In the last two decades advances in the field of computer technology have had a tremendous impact on the design processes engineers face every day. The use of sophisticated computer programs to aid engineers in the design of technical devices, such as television sets and cellular phones, is common practice nowadays. These computer programs are able to provide much more (detailed) information about the devices than the engineers had before. On the downside, however, this stream of new-found knowledge has led to an increasingly more complex decision process.

When designing a new device, engineers try to find a product design that fulfills their requirements. These requirements are stated in terms of (quantifiable) criteria that the final product should meet. The expected lifetime of a product is an example of such a criterion. Unfortunately, it is very hard to determine whether all criteria are met *before* a product is actually manufactured. Therefore, engineers resort to prototyping, i.e. they test different *prototypes* of the product, during various stages of the design process, in order to find one that meets all design criteria.

In the early days *physical prototyping* was used most often, which meant that several different product designs, or scaled versions of it, were manufactured and then tested on how they performed on the design criteria, i.e. these prototypes acted as test-scenarios for the product. Physical prototyping, however, takes a lot of time and large costs are incurred at the production of the prototypes. Furthermore, the increased complexity

of many technical devices, as well as the increased pressure on the time-to-market, has made this type of prototyping more and more obsolete. Therefore, physical prototyping is nowadays often replaced by *virtual prototyping*. Instead of actually manufacturing the prototypes they are now represented by computer simulation models (cf. Oden et al. (2006)). Such models can be constructed using *Computer Aided Engineering* tools, such as Finite Element Analysis and Computational Fluid Dynamics. These are special computer packages that are able to simulate the behavior of a product. Hence, the engineers can monitor directly how different prototypes will perform without the need to go through the timely and costly process of manufacturing, which is especially helpful when implementing new product designs. Furthermore, this premature testing minimizes the possibility of flaws in the final product.

There are many other fields, besides engineering, in which the decision process is facilitated by simulation tools. Examples of such fields are: logistics, military, social science, and finance; see e.g. Law and Kelton (2000). In this thesis, however, we mainly consider the problems that arise when designing a product or a process in engineering. Note that each time the term *product* is utilized in the text, the term *process* could be read instead.

Due to the complexity of the mathematical systems underlying the computer simulation tools there are, unfortunately, often no (simple) explicit input-output formulas known; such tools are therefore referred to as *black boxes*. It is then up to the engineer to set the design (or input) parameters in such a way that the observed response (or output) parameters meet all requirements of the final product; see Figure 1.1. Although computer power has significantly increased during the last years, the evaluation of a particular setting of the design parameters (also called a *scenario*) may still be very time-consuming. It is not unusual for one evaluation to take several minutes, or even up to several hours, of computation time. To gain more insight into a computer simulation tool the unknown black-box function is often replaced by an approximation model, based on a set of evaluations of the black-box function. Since computation time, and, hence, the number of evaluated scenarios, is limited in practice, the question as to which set of scenarios to evaluate becomes one of vital importance. Answering this question is the main focus of this thesis.

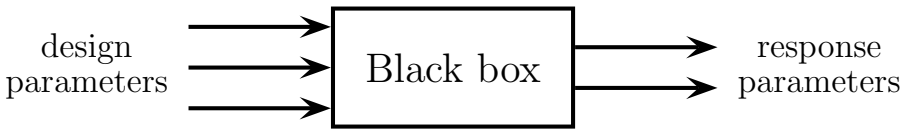


Figure 1.1: A black-box function.

The change from physical prototyping to virtual prototyping clearly has had influence on the way experiments are dealt with these days. On the side of setting up experiments, i.e. determining which product scenarios to evaluate, things have changed significantly. As a result, the traditional statistical *design of experiments*, such as full and fractional factorial designs, is no longer able to correctly deal with deterministic computer experiments. Some reasons that underlie this inability are the following; see Stehouwer and Den Hertog (1999):

- Due to the presence of noise in traditional physical experiments, replicating the evaluation of a particular design point will result in different response values. These replicates are used to form confidence intervals for the expected main and interaction effects of design parameters on response values (cf. Law and Kelton (2000)). With deterministic computer simulations, however, lack of noise will yield exactly the same outcome when a design point is evaluated twice.

- Another effect of noise in physical experiments is that design points in traditional design of experiments will often be located on or near the border of the design space (or the feasible region). With computer experiments, however, the absence of noise no longer restricts the design points to the borders of the feasible region. Since the behavior in the interior of the design space is equally important as the behavior on the border of this region, a design for computer experiments should have its design points spread out over the entire feasible region.

- Most traditional designs for experiments are applicable only to problems with constraints on parameter ranges, i.e. rectangular design spaces. In the practice of expensive computer simulations, however, there is sometimes a need for designs on arbitrarily shaped feasible regions. Stinstra, Den Hertog, Stehouwer, and Vestjens (2003) propose a method to obtain designs on different shaped regions, such as a strip and a quarter of a disk.

For above reasons a *design of computer experiments* should be used instead of a traditional design of experiments when dealing with deterministic computer simulations. The main part of this thesis focuses on the construction of so-called maximin Latin hypercube designs. Such designs for computer experiments have been shown to lead to good approximation models, see e.g. Simpson et al. (2001), Santner et al. (2003), and Bursztyn and Steinberg (2006).

1.2 Metamodel approach

It has been proposed to replace black boxes by global approximation models, also called metamodels; see e.g. Kleijnen (1987), Barton (1998), Jones et al. (1998), and Booker

et al. (1999). Equivalent terms that appear in the literature are: compact models, surrogate models, and response surface models. With such metamodels product designs can be evaluated relatively fast. Hence, these models can be used to gain insight into the product over the whole design space. Furthermore, the explicit approximating functions enable the search for optimal and robust product designs within an admissible time.

Alternatively, several sequential optimization methods have been introduced in the literature to deal with design optimization involving expensive (or time-consuming) simulations, see Driessen (2006) for a comprehensive overview of such methods. These sequential methods try to find an optimal product design by means of derivative-free optimization and search methods; see e.g. Toropov et al. (1993), Glover et al. (1996), Conn et al. (1997), Powell (2000), and Brekelmans et al. (2005).

Note that these sequential techniques do not lead to a global approximation model, and, hence, less information about the behavior of the product is obtained. Furthermore, optimal product designs found by optimizing a global approximation model remain feasible under slight changes in the optimization problem, whereas with sequential optimization methods new evaluations would be needed. The advantage of sequential optimization, however, is that the number of required evaluations is in general lower. This stresses the importance of determining a good set of evaluation points when using a global approximation model, i.e. a set that is expected to yield as much information as possible concerning the underlying black-box function.

In this thesis we consider the *Metamodel approach*; see Den Hertog and Stehouwer (2002). This approach replaces the (unknown) black-box function by a global approximation model, based on evaluations of some scenarios. The design process can be divided into four basic steps: problem specification, design of computer experiments, metamodeling, and design analysis and optimization. Next, the four steps in the Metamodel approach are summarized, as well as the problems that are encountered when applying this procedure to product design problems. For a detailed discussion of these steps the reader is referred to Stinstra (2006).

Step 1: Problem specification

In the first step of the Metamodel approach the product design problem is formulated. This includes the identification and definition of both the design and response parameters. The expected number of simulations needed, as well as the corresponding simulation times, are also determined. The latter times are of particular importance for cases where there is a budgetary maximum on the time spent on simulation. To explore the design space, restrictions on design parameter settings, such as lower and upper bounds, have to be investigated. Furthermore, physical limitations may apply. For example, it may be known beforehand that particular settings of the design parameters do not lead to

good designs or are even infeasible, and, hence, restrictions on combinations of designs parameters should be considered. The collection of parameter settings that satisfy all restrictions then constitutes the design space (or feasible region). There may also be restrictions imposed on some of the response parameters. Since response values will be known only *after* the scenarios have been evaluated, feasibility of the observed responses has to be checked afterwards. In order to use the fitted metamodels (see Step 3) to find a good product design (see Step 4) the requirements that the final product has to meet also have to be defined in the first step.

Step 2: Design of computer experiments

With the design space determined the question arises as to which scenarios (or design points) to evaluate. Such a set of evaluation points is called a *design*. Note that the term *design* has two different meanings in this thesis; depending on the context, it either refers to the design of (computer) experiments or to the design of a product. When no details on the functional behavior of the response parameters are available, it is important to obtain information from the entire design space. One way to accomplish this is to construct a space-filling design, i.e. to have the design points "evenly spread" over the entire feasible region. In Chapter 2 several different criteria that will lead to a proper distribution of the design points over the design space are discussed. Furthermore, the main subject of most subsequent chapters of this thesis is the construction of good designs for computer experiments.

Step 3: Metamodeling

After the design points have been evaluated the observed response values are used to fit metamodels to the black box. Polynomials, neural networks, radial basis functions, and Kriging models, are popular choices for these approximation models. To validate the obtained models, techniques such as cross-validation could be used; see e.g. Kleijnen and Sargent (2000). Should a metamodel appear to be invalid, then either a different metamodel should be fitted to the data or an additional set of evaluations has to be carried out to improve the current model. Chapters 7 and 8 introduce a way to choose such extra scenarios.

Step 4: Design analysis and optimization

Once valid metamodels have been found, these models, in combination with optimization techniques, will help to gain insight into the product and to find a good product design, within the design space, that satisfies all response criteria set by the engineers. Due to the fact that metamodels are explicit functions, function evaluations are relatively fast, and, hence, mathematical programming approaches (cf. Birge and Murty (1994))

could be applied. Since the resulting best-found product design is an approximation of the real (unknown) optimum, it is wise to simulate the corresponding design parameter settings once more. When the observed response values do not deviate too much from the response values estimated by the metamodels, the product design is very likely a good one. Note, however, that during the manufacture of the product some of the design parameters may be subject to noise, e.g. due to small errors in their actual settings. To deal with this problem robustness should be taken into account; see Stinstra and Den Hertog (2005) for a more detailed discussion on how to obtain a robust product design.

1.3 Contribution

The contribution of this thesis is twofold. On the one hand, many new (approximate) maximin designs are obtained for the class of Latin hypercube designs. On the other hand, coordination methods and nested maximin designs are introduced as means to deal with interdependencies among black-box functions and/or among function evaluations of a single black box.

Part I considers the use of maximin Latin hypercube designs in the design of computer experiments for box-constrained design spaces. These maximin Latin hypercube designs are extremely useful in the approximation and optimization of black-box functions. In this thesis general formulas are derived for two-dimensional maximin Latin hypercube designs of n points, when the distance measure is the maximum norm or the rectangular distance. For the Euclidean distance measure, maximin Latin hypercube designs are obtained for $n \leq 70$ and approximate maximin Latin hypercube designs are obtained for $n \leq 1000$. Furthermore, we investigate the trade-off between the space-fillingness and the non-collapsingness of designs for computer experiments and show that highly non-collapsing designs can be constructed without reducing the space-fillingness too much. Moreover, for two-dimensional maximin designs we show that the reduction in the maximin distance caused by imposing the Latin hypercube structure is in general small. This justifies the use of maximin Latin hypercube designs instead of the traditional unrestricted designs. Moreover, for up to ten dimensions approximate maximin Latin hypercube designs are constructed for $n \leq 100$. These designs present a significant extension of the previously known results.

Part II presents a collaborative extension of the Metamodel approach (cf. Den Hertog and Stehouwer (2002)). In this Collaborative Metamodel approach, which acts as a framework for dealing with multi-component product design problems, coordination of the interrelated black-box functions plays a crucial role. Such interrelations occur, for example, in the design of high-tech products, such as automobiles and aircrafts, where the products often consist of many interrelated components, each of them represented

by their own black-box functions. To deal with black-box functions that depend on each other by some output-input relations the concept of coordination methods is introduced. Several aspects of such coordination methods are discussed and compared. For the throughput time, i.e. the total time needed for all simulations, general formulas are derived. Another important step in the Collaborative Metamodel approach is the construction of nested designs. Such designs are useful when dealing with black-box functions that have some design parameters in common. In this thesis general formulas are derived for one-dimensional nested maximin designs, when nesting two designs, and approximate maximin designs are obtained when nesting three or four designs. Furthermore, it is shown that the loss in space-fillingness, with respect to traditional maximin designs, is relatively small. Moreover, in two dimensions, non-collapsing nested maximin designs are obtained for $n \leq 15$ (and some larger values), when nesting two designs, for different types of grids. Although the concept of sequential evaluations, i.e. first evaluating an initial set of design points and then, if needed, evaluating an additional set of points, is not new, the usage of nested designs leads to new ways to facilitate this process. In the same light, the obtained nested maximin designs could also be used as training and test sets for fitting and validating metamodels, respectively.

Note that all maximin Latin hypercube designs and nested maximin designs that are obtained in this thesis can be downloaded from the website

`http://www.spacefillingdesigns.nl`.

1.4 Outline

This thesis consists of two parts. The main focus of both parts is on designs for computer experiments. The current section provides a short description of the contents of all the following chapters.

Part I considers the construction of (box-constrained) maximin designs, and maximin Latin hypercube designs in particular, for a single black-box function. Chapter 2 first gives an overview of the literature in the field of design of computer experiments. Chapter 3 derives construction methods for two-dimensional maximin Latin hypercube designs for the maximum norm and the rectangular distance measure. Furthermore, for the Euclidean distance measure a heuristic construction method to obtain two-dimensional approximate maximin Latin hypercube designs is proposed. Chapter 4 extends this heuristic to higher dimensions and uses this extension, in combination with a simulated annealing algorithm, to obtain approximate maximin Latin hypercube designs for up to ten dimensions. Finally, Chapter 5 illustrates the trade-off between the space-fillingness and non-collapsingness of two-dimensional maximin designs.

Part II considers, among others, the problem of dealing with multi-component product design problems. Chapter 6 introduces the Collaborative Metamodel approach as a framework to deal with such design problems. Furthermore, it proposes to use coordination methods in order to efficiently deal with the relationships present among the various components. The next two chapters consider the construction of nested maximin designs. Chapter 7 provides explicit and heuristic construction methods for one-dimensional nested maximin designs. The construction of two-dimensional nested maximin designs for different types of grids is considered in Chapter 8. Finally, Chapter 9 presents the main conclusions and gives some directions for further research.

This thesis is based on the following research papers:

Chapters 3 & 5	Dam, E.R. van, B.G.M. Husslage, D. den Hertog, and J.B.M. Melissen (2006). Maximin Latin hypercube designs in two dimensions, *Operations Research*. To appear.
Chapter 4	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*, Tilburg University.
Chapter 6	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.
Chapter 7	Dam, E.R. van, B.G.M. Husslage, and D. den Hertog (2004). One-dimensional nested maximin designs, *CentER Discussion Paper 2004-66*, Tilburg University.
Chapter 8	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*, Tilburg University.

Part I

Maximin designs

Chapter 2

Design of computer experiments

History is rarely made by reasonable men.

(Terry Goodkind, *Blood of the Fold*)

2.1 Introduction

The second step of the Metamodel approach encompasses the construction of a design of computer experiments (see Section 1.2). Such a design is a collection of points at which the underlying black-box function will be evaluated. Response values obtained at these evaluations are used to quantify the effect that the design parameters have on the characteristics of the product. Furthermore, based on the observed data, metamodels can be built to approximate the unknown black-box function. Not only does this lead to a better understanding of the final product, it also opens the way to the use of optimization techniques to find a good product design.

The prediction accuracy of a metamodel is not only affected by the type of model used, e.g. a polynomial, it also heavily depends on the data onto which the model is fitted, i.e. on the design points that are evaluated. Hence, well-chosen design points increase the accuracy of the constructed metamodels, which, in turn, improves the approximation of the true behavior of the unknown black-box function. Therefore, it is vitally important to use a proper design of computer experiments. This chapter discusses several classes of designs and different measures that are used, both in literature and in practice, to obtain good designs for computer experiments. As is recognized by several authors, a design of computer experiments should at least incorporate the following two features. First of all, the design should be *space-filling* in some sense. Secondly, the design should be *non-collapsing*. These two features are discussed in Sections 2.2 and 2.3, respectively.

We assume that all parameters are equally important in the construction of the design of computer experiments. Therefore, box constraints, i.e. lower and upper bounds, on

the design parameters can (and must) be scaled to equally sized intervals, e.g. $[0, 1]$ or $[0, n-1]$, for every parameter. Note that in this thesis we will sometimes choose to scale designs to the $[0, 1]^k$-box and at other times choose to use the $[0, n-1]^k$-box. In the current chapter all distance computations are based on the $[0, 1]^k$-box.

2.2 Design criteria

It has been stressed before that it is important to have a good design, i.e. a collection of evaluation points, for computer experiments. The problem is to define what makes a design "good". We need some kind of criterion that tells us when one particular design is preferred over another one in order to find a good (and possibly the best) design. In this section several criteria for good designs that are often used in the literature and in practice are considered.

2.2.1 Geometrical criteria

As noted in Section 1.2, it is important to obtain information from the entire feasible region when there are no details available on the functional behavior of the response parameters. Therefore, design points should be "evenly spaced" over the entire region. A design that fills the whole design space is called *space-filling*.

Maximin design

Intuitively it appeals to spread design points over the design space in such a way that the separation distance (i.e. the minimal distance between pairs of points) is maximized. Let $x_i \in \mathbb{R}^k$, $i = 1, \ldots, n$, represent the n design points of a k-dimensional design X within the feasible region Ω and let $d(\cdot, \cdot)$ be a certain distance measure. A maximin design X then has a distance

$$d = \max_{\substack{X \subset \Omega \\ |X|=n}} \min_{\substack{x_i, x_j \in X \\ i \neq j}} d(x_i, x_j). \tag{2.1}$$

Figure 2.1 gives an example of a maximin design of 7 points in the unit square, with respect to the Euclidean (or ℓ^2) distance measure, i.e.

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^{k} (x_{il} - x_{jl})^2}. \tag{2.2}$$

The optimal design in this figure has the separation distance $d = 4 - 2\sqrt{3} \approx 0.5359$; see e.g. Melissen (1997). Note that the design point located in the upper-right of the square is a so-called *rattler*, i.e. a point that can be moved somewhat without affecting the separation distance. The region within which this point can be moved freely is depicted by the gray-colored area.

Minimax design

Another intuitively appealing criterion is to require every point in the region to have a design point close by, or, put differently, to minimize the maximal distance from any point to the design. Let $y \in \mathbb{R}^k$ represent an arbitrary point in the feasible region, X a design of n points, and $\rho(y, X)$ the distance between y and its closest design point, i.e.

$$\rho(y, X) = \min_{x_i \in X} d(y, x_i). \tag{2.3}$$

A minimax design X of n points then has a distance

$$\rho = \min_{\substack{X \subset \Omega \\ |X|=n}} \max_{y \in \Omega} \rho(y, X). \tag{2.4}$$

The distance ρ is referred to as the *minimal covering radius* of the design. For example, in case of 7 congruent ℓ^2-circles the minimal radius needed to cover the unit square is $\rho \approx 0.2743$; see Figure 2.2 (from Johnson et al. (1990)). In this figure the diamonds ($\diamond$) depict *remote sites*, i.e. points in the square that are at distance ρ from the design.

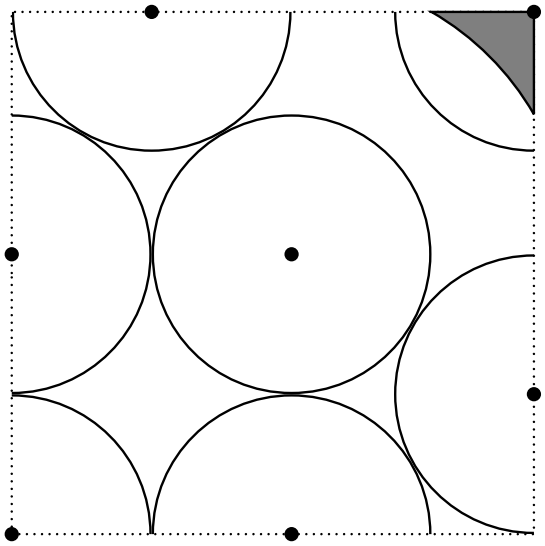


Figure 2.1: Two-dimensional ℓ^2-maximin design of 7 points; $d \approx 0.5359$.

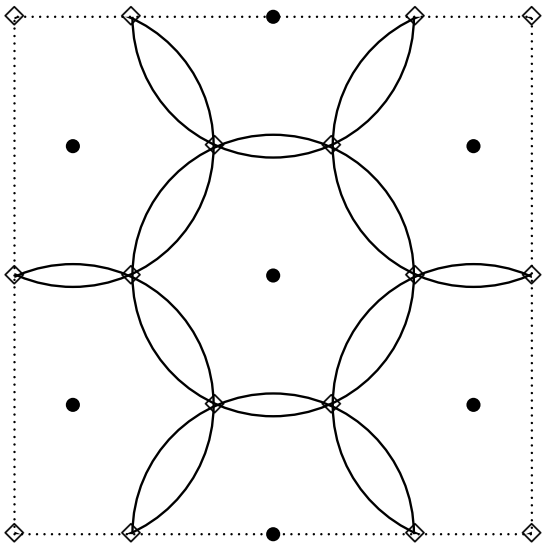


Figure 2.2: Two-dimensional ℓ^2-minimax design of 7 points; $\rho \approx 0.2743$.

Uniform design

As a third criterion, consider the problem of finding a design that is as uniformly distributed as possible. Fang et al. (2000) use the L_p-discrepancy to measure the *uniformity* of a design. The lower this discrepancy, the more uniform the design points are scattered

over the feasible region Ω. More formally, the minimal L_p-discrepancy is given by

$$\min_{\substack{X \subset \Omega \\ |X|=n}} \left(\int_{\Omega} |F_n(y, X) - F(y)|^p \right)^{1/p}, \tag{2.5}$$

with $F_n(y, X)$ the empirical distribution function of design X (of n points) and $F(y)$ the uniform distribution function on Ω. Popular choices for the parameter p are 2 and ∞. The so-called U-type design is the most widely used uniform design. Since this particular type of design is non-collapsing, an example of such a uniform design is postponed until Section 2.3.

Audze-Eglais design

Another criterion that leads to a space-filling distribution of the design points has been proposed by Audze and Eglais (1977). The authors consider the physical analogy of a system of points with potential energy U. This energy is caused by repulsive forces between the points, and, naturally, the system will move to a state with minimal potential energy. Bates et al. (2004) apply this idea to construct non-collapsing, space-filling designs. Under the assumption that the repulsive forces are inversely proportional to the squared distances between the points, an Audze-Eglais design is obtained:

$$U = \min_{x_i, x_j \in X} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{d^2(x_i, x_j)}. \tag{2.6}$$

Here, X is a non-collapsing design of n points; see Section 2.3.

2.2.2 Statistical criteria

Instead of using a criterion that optimizes the distribution of the design points over the feasible region in some sense, i.e. a geometrical criterion, it may be interesting to use a criterion based on some statistical arguments. For example, when it is expected that the (unknown) black-box function can be approximated by a second-order polynomial it may be wiser to choose the design points in such a way that the expected error of fitting the polynomial to the observed data will be minimal.

Integrated mean squared error design

Let $R(y)$ represent the response function, which depends on the design points y. Assume that R has the form

$$R(y) = \sum_{j=1}^{t} \beta_j f_j(y) + Z(y). \tag{2.7}$$

Here, each $f_j(y)$ is a *known* polynomial and each β_j is the corresponding *unknown* coefficient. Furthermore, $Z(y)$ is some stochastic process that represents the deviation of the (unknown) black-box function from the assumed linear model; see Sacks et al. (1989). For a given design X of n points, let the best linear predictor of $R(y)$ be defined by $\hat{R}(y, X)$. The mean squared error (MSE) of this predictor is then given by

$$MSE\left(\hat{R}(y,X)\right) = \mathbb{E}\left\{\left(\hat{R}(y,X) - R(y)\right)^2\right\}. \tag{2.8}$$

To obtain a design that works well for the entire design space, the integrated mean squared error (IMSE) is often considered. This criterion averages the mean squared error over the region of interest, i.e. the feasible region Ω, possibly using some weight function. For the normalized IMSE criterion the best design is found by solving the following problem (with σ_Z^2 the variance of process Z):

$$\min_{\substack{X\subset\Omega \\ |X|=n}} \frac{1}{\sigma_Z^2} \int_\Omega MSE\left(\hat{R}(y,X)\right) dy. \tag{2.9}$$

Note that the above expression depends on the correlation structure of Z, and, hence, it is important to choose a proper setting of the correlation parameters, which may be hard. Another disadvantage is that even when dealing with multiple responses for each response the same correlation structure Z has to be used. Figure 2.3 gives an example of a two-dimensional integrated mean squared error design for a quadratic model, where Z is assumed to be a Gaussian process (from Sacks et al. (1989)).

Crary et al. (2000) have developed I-OPT$^{\text{TM}}$, to generate designs with minimal integrated mean squared error. They find that IMSE-optimal designs may have proximate design points, which they call "twin points"; see Crary (2002).

Maximum entropy design

Entropy was introduced by Shannon (1948) to measure the amount of available information (about some process). In the field of design of experiments Lindley (1956) used this notion to determine the information provided by the experiments. The lower the entropy, the better the understanding of the underlying process. Let π represent the prior distribution (i.e. before the experiments) and π_X the posterior distribution (i.e. after the experiments). The prior and posterior information on the process are then defined as

$$I = \int_\Omega \pi(y) \log\Big(\pi(y)\Big) dy = \mathbb{E}_\pi\Big\{\log \pi\Big\}, \text{ and} \tag{2.10}$$

$$I_X = \int_\Omega \pi_X(y) \log\Big(\pi_X(y)\Big) dy = \mathbb{E}_{\pi_X}\Big\{\log \pi_X\Big\}, \tag{2.11}$$

respectively. The change in information, and thus the value of the experiments, is equal to $I_X - I$. Farhangmehr (2003) shows that this difference can be rewritten as $H - H_X$, where H and H_X are Shannon's prior and posterior entropy:

$$H = -\mathbb{E}_\pi\Big\{I\Big\} \overset{(2.10)}{=} -\mathbb{E}_\pi\Big\{\mathbb{E}_\pi\{\log \pi\}\Big\}, \text{ and} \tag{2.12}$$

$$H_X = -\mathbb{E}_{\pi_X}\Big\{I_X\Big\} \overset{(2.11)}{=} -\mathbb{E}_{\pi_X}\Big\{\mathbb{E}_{\pi_X}\{\log \pi_X\}\Big\}, \tag{2.13}$$

respectively. Hence, a design is of maximum entropy if it minimizes the posterior entropy H_X. This corresponds to selecting those design points about which the least is known. Note that under the Gaussian assumption a maximum entropy design maximizes the determinant of the prior covariance matrix; see Koehler and Owen (1996). An example of a two-dimensional maximum entropy design is depicted in Figure 2.4 (from Farhangmehr (2003)).

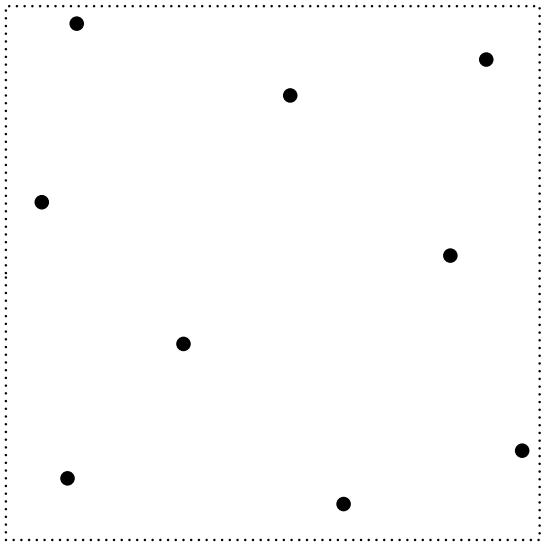


Figure 2.3: Two-dimensional IMSE design of 9 points for a quadratic model.

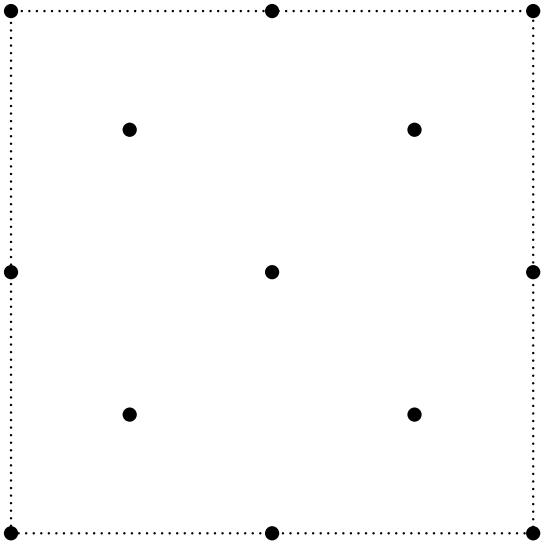


Figure 2.4: Two-dimensional maximum entropy design of 13 points.

2.2.3 Other criteria and related problems

The above list of criteria is by no means meant to be exhaustive. For a more thorough discussion of these, and other, criteria the reader is referred to Koehler and Owen (1996) and Santner et al. (2003). In the rest of this thesis we will consider the maximin distance criterion. The resulting maximin (Latin hypercube) designs generally speaking yield the best approximations, see e.g. Simpson et al. (2001), Santner et al. (2003), and Bursztyn and Steinberg (2006).

The two-dimensional maximin design problem has been studied in location theory. In this field of research, the problem is usually referred to as the *continuous multiple facility location problem* or the *max-min facility dispersion problem*, see e.g. Erkut (1990) and Dimnaku et al. (2005). Facilities, such as power plants, are placed in the plane such that the minimal distance to any other facility is maximal. In the case of power plants, such a placement minimizes the probability that a failure of one of the power plants will affect the other plants.

There is also much literature on packing and covering with circles. The problem of finding the maximal common radius of n circles that can be packed into a square (or, in higher dimensions, the packing of n congruent spheres into a k-dimensional cube) is equivalent to the maximin design problem. The problem of finding the minimal common radius of n circles that cover a square is equivalent to the minimax design problem. Melissen (1997) gives a comprehensive overview of the historical developments and state-of-the-art research in these fields. For the ℓ^2-distance measure optimal two-dimensional maximin 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 larger values of n; see the Packomania website of Specht (2005). Baer (1992) solved the maximum ℓ^∞-circle packing problem in a k-dimensional unit cube. The maximum ℓ^1-circle packing problem in a square has been solved for many values of n; see Fejes Tóth (1971) and Florian (1989). Chapters 3 to 5 discuss maximin designs in more detail, and (their relation with) non-collapsing maximin designs in particular.

2.3 Non-collapsing designs

Designs for computer experiments are mostly used to gain insight into, and optimize, black-box functions. Since there is often no information available about the black-box behavior, design points should be chosen in such a way that the expected amount of information obtained is maximized. Section 2.2 discusses several criteria that can be used to address this problem.

Unfortunately, designs that are optimized for the space-fillingness criterion (or one of the statistical criteria) often turn out to be highly *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. For deterministic black-box functions this is not a desirable situation. For example, should one of the two design parameters in Figure 2.4 have no significant influence, then the 13 evaluated design points would collapse onto only 5 different points, thereby losing 8 time-consuming evaluations. Therefore, two design points

should not share any coordinate values when it is not known a priori which dimensions are important. Of course, the *screening* of design parameters, i.e. to determine which parameters are important based on experience with or knowledge about the underlying process, *before* an experiment is set up may provide useful information about which design parameters appear to have a significant influence on the responses. However, the true effect of a design parameter on the black-box function value will still be known only *after* the computer experiments have taken place. Hence, the non-collapsingness of a design remains an important issue to consider.

2.3.1 Latin hypercube designs

To guarantee non-collapsingness, when searching for a good design, the search space is often restricted to some class of designs. One of such classes that is widely used in both theory and practice is the class of Latin hypercube designs (LHDs). In our definition a Latin hypercube design is an $n \times k$ matrix, where each column y_j, $j = 1, \ldots, k$, is a permutation of the set $\{0, 1, \ldots, n-1\}$. The rows $x_i = (x_{i1}, x_{i2}, \ldots, x_{ik})$, $i = 1, \ldots, n$, of this matrix define the design points. Note that the design points lie on the $[0, n-1]^k$-grid, and, since every column is a permutation, no coordinate values are shared by any pair of design points. As an example, consider the following two-dimensional Latin hypercube design of $n = 12$ points:

$$X^T = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 0 & 9 & 5 & 8 & 6 & 10 & 2 & 4 & 1 & 11 & 3 & 7 \end{pmatrix}. \tag{2.14}$$

The design corresponding to matrix X is depicted in Figure 2.5.

McKay et al. (1979) were the first to use Latin hypercube designs in computer experiments by introducing a technique called *Latin hypercube sampling*. The idea is to divide the design space into n^k equally sized cells and to randomly select n cells, under the restriction that the projections of the selected cells onto any axis do not overlap. A design point is chosen randomly within each of the n selected cells. See Figure 2.6 for an example of a two-dimensional Latin hypercube sample of $n = 12$ points. The use of this sampling method reduces the variance of the expected response values (asymptotically), when compared to independent and identically distributed (iid) sampling; see e.g. Stein (1987). A slightly adapted version of Latin hypercube sampling lets the centers of the selected cells represent the design points, instead of choosing them randomly. Note that these latter designs (after scaling) correspond to our definition of a Latin hypercube design.

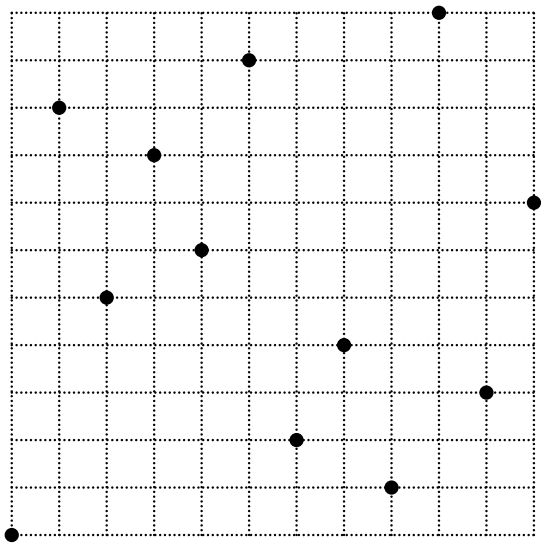


Figure 2.5: Two-dimensional Latin hypercube design of 12 points.

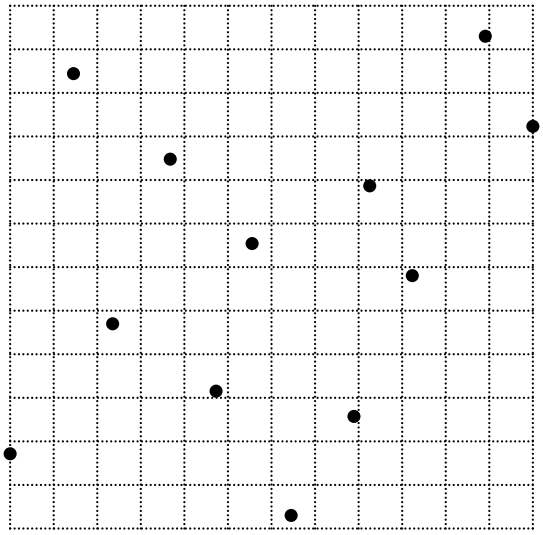


Figure 2.6: Two-dimensional Latin hypercube sample of 12 points.

2.3.2 Orthogonal arrays

Several researchers have considered Latin hypercube designs that exhibit some special structure. For example, both Owen (1992) and Tang (1993), independently and contemporaneously, have used *orthogonal arrays* to construct designs for computer experiments. An $n \times k$ matrix OA, with its elements taken from the set $\{1, 2, \ldots, s\}$, is called an orthogonal array of strength t if in any $n \times t$ submatrix of OA each of the s^t possible rows occurs with the same frequency λ; clearly, $n = \lambda s^t$. Furthermore, note that a Latin hypercube design is an orthogonal array of strength 1, i.e. $s = n$ and $\lambda = t = 1$. The advantage of orthogonal arrays is their uniformity in each t-variate margin, i.e. when projected onto t (or fewer) dimensions the points in the array form a regular grid. Latin hypercube designs exhibit this property only in one dimension. A major disadvantage, however, is that orthogonal arrays exist only for certain values of k and when $n = \lambda s^t$. Tang (1993) proposes a method to construct Latin hypercube designs by extending orthogonal arrays, thereby (partly) preserving some of the features of the latter. Owen (1992) uses randomization to derive Latin hypercube designs from the columns of orthogonal arrays. An example of such a *randomized orthogonal array* is depicted in Figure 2.7. In fact, the figure shows one of the two-dimensional projections of an orthogonal array with $(n, k, s, \lambda, t) = (16, 5, 4, 1, 2)$ (see the StatLib website of Meyer and Vlachos (1993)) and with the design points randomly centered (otherwise, the projection would yield a 4×4 regular grid). Note that each of the 16 subsquares contains exactly one design point.

To extend the class of designs with orthogonal properties Ye et al. (2000) consider

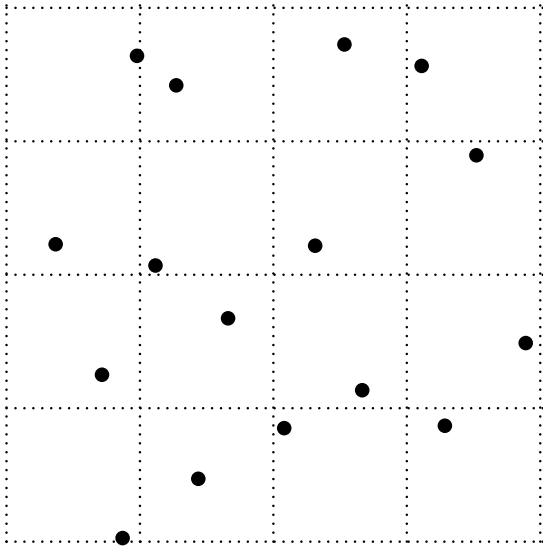


Figure 2.7: Two-dimensional projection of a five-dimensional, randomly centered, randomized orthogonal array of 16 points.

symmetric Latin hypercube designs. A Latin hypercube design is called symmetric when for every point x_i in the design there exists another point x_j in the design that is the reflection of x_i through the center. These symmetric LHDs can be viewed as generalizations of orthogonal-array based LHDs that still retain some of the orthogonality of the latter. Morris and Mitchell (1995) are the first to mention symmetric properties of some Latin hypercube designs. They observe symmetry in maximin designs for which $n = 2k$ and refer to them as *foldover designs*.

Finally, Steinberg and Lin (2006) present a construction method for orthogonal Latin hypercube designs, for the special case where $n = 2^k$ and $k = 2^m$, which is based on rotating the design points in a two-level factorial design.

2.3.3 Space-filling Latin hypercube designs

Section 2.2 discusses several criteria that can be used to obtain a space-filling distribution of the design points over some specified feasible region. Moreover, for the class of Latin hypercube designs one of these criteria could be applied to obtain a space-filling design of computer experiments.

Figure 2.8 shows an optimal Latin hypercube design of 12 points on the unit square for the ℓ^2-maximin distance criterion, with $d = \frac{\sqrt{13}}{11} \approx 0.3278$. Maximin Latin hypercube designs are discussed extensively in Chapters 3 and 4. Van Dam (2005) considers two-dimensional Latin hypercube designs that are optimized for the minimax criterion. The case of 12 design points on the unit square is depicted in Figure 2.9. The minimal radius needed to cover the square is equal to $\rho = \frac{5}{22} \approx 0.2273$. Again, the diamonds ($\diamond$) rep-

resent remote sites. Furthermore, note that the Latin hypercube designs in Figures 2.8 and 2.9 are both symmetric (see Section 2.3.2).

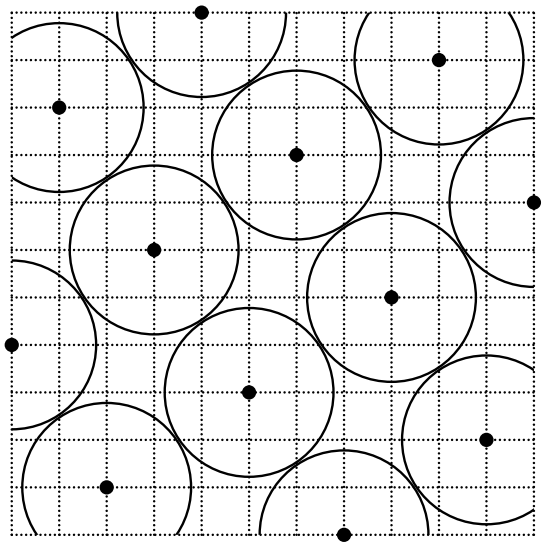


Figure 2.8: Two-dimensional ℓ^2-maximin Latin hypercube design of 12 points; $d \approx 0.3278$.

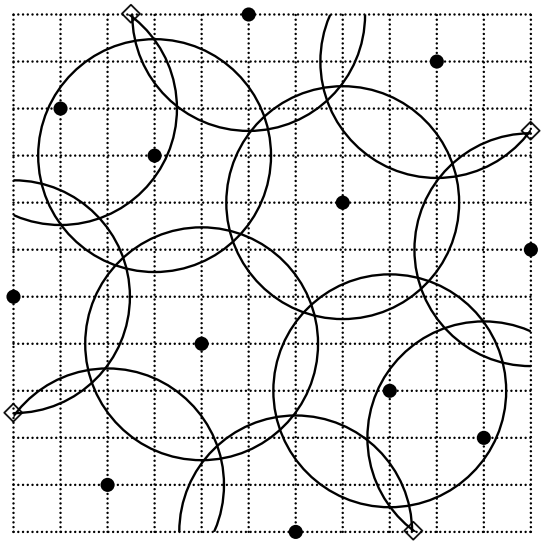


Figure 2.9: Two-dimensional ℓ^2-minimax Latin hypercube design of 12 points; $\rho \approx 0.2273$.

As mentioned in Section 2.2.1, the U-type design is the most widely used uniform design. Since each column of this type of design is a permutation of $\{0, 1, \ldots, n-1\}$ the resulting design points form a Latin hypercube design. Figure 2.10 (from the Uniform Design website of Fang et al. (1999)) gives an example of a two-dimensional U-type uniform design of 12 points on the unit square that minimizes the centered L_2-discrepancy measure: $CL_2 \approx 0.0456$. Note that this centered measure does not only take into account the uniformity of the design points, but also the uniformity of all the projections of these points, see Fang et al. (2002).

An Audze-Eglais Latin hypercube design of 10 points, with minimal potential energy $U \approx 2.0662$ (with respect to the squared Euclidean distance measure), is depicted in Figure 2.11 (from Bates et al. (2003)).

Several other criteria are used to optimize over the class of Latin hypercube designs. Morris and Mitchell (1995), for example, introduce a scalar-valued design criterion that is used to break ties between multiple designs that are maximin (and of minimum index), thereby extending the definition used in Johnson et al. (1990). A simulated annealing method is used to explore the set of possible Latin hypercube designs. Jin et al. (2005) propose a stochastic evolutionary algorithm to search this set. To achieve space-fillingness the authors optimize the criterion of Morris and Mitchell, as well as maximum entropy. Park (1994), finally, uses an exchange algorithm to find designs that minimize

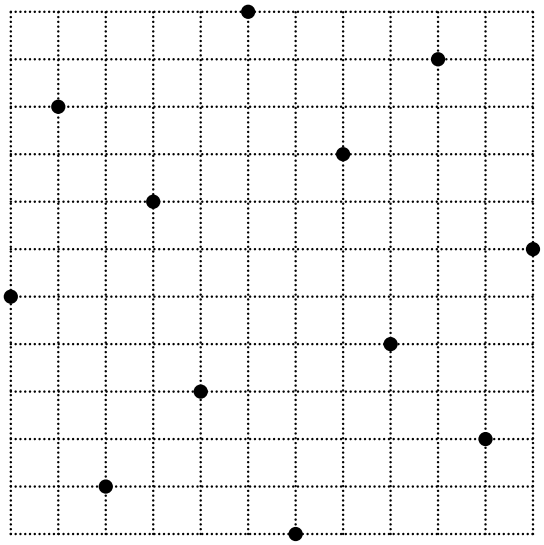


Figure 2.10: Two-dimensional centered L_2-discrepancy U-type uniform design of 12 points; $CL_2 \approx 0.0456$.

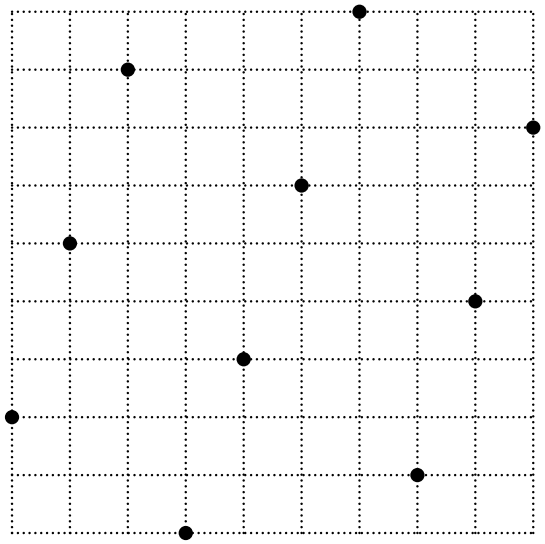


Figure 2.11: Two-dimensional Audze-Eglais Latin hypercube design of 10 points; $U \approx 2.0662$.

the integrated mean squared error or that maximize entropy.

In location theory there exists a discrete version of the continuous multiple facility location problem. In this case the facilities are chosen from a fixed set of candidate (grid) points in such a way that, for example, the sum of the separation distances between pairs of facilities is maximal (cf. Daskin (1995)). Note, however, that the obtained solution may still be a collapsing design, and, hence, extra restrictions have to be added to the discrete location problem to enforce the Latin hypercube structure.

All aforementioned authors deal with box-constrained design spaces. Stehouwer and Den Hertog (1999) are among the few that consider space-filling Latin hypercube designs on a non-box feasible region. To have the design points fall into the interior of the constrained design space the authors use a refined grid. The density of this grid depends on the content of the non-box region, relative to the content of its enveloping box. In this thesis, however, we only consider box-constrained design spaces. Furthermore, to distinguish between designs with some specific structure, e.g. Latin hypercube designs and orthogonal arrays, and designs without an implied structure, the latter are referred to as *unrestricted designs* in this thesis.

2.4 Sequential and nested designs

Since computer simulations are time-consuming there is often a (budgetary) maximum on the allowed number of evaluations. Therefore, one could choose to first evaluate a small

number of design points to get a better understanding of the design space. After all the computer simulations have been performed the response values obtained at the evaluation points could be used to fit a metamodel. This approximation model may, or may not, turn out to be valid (see Section 1.2). In case of an invalid model, either a different model should be fitted, or more data are needed to find a proper approximation of the (unknown) black-box function. In the latter case, the remaining (allowed) simulations could be used to extend the current design of computer experiments with extra evaluation points, resulting in a so-called *sequential design*. Jin et al. (2002) apply both the maximum entropy and the integrated mean squared error criterion to the problem of finding such an augmenting set. These two statistical criteria are able to adapt the placement of additional points to the existing metamodel, i.e. to let the choice of new evaluation points depend, among others, on the correlation parameters of the current metamodel. A major drawback, however, is that this adaptation is limited to Kriging models. To deal with other types of approximation models a geometrical criterion, such as maximin, could be used. Since such a criterion lacks adaptation to the fitted metamodel Jin et al. (2002) propose to use a maximin scaled-distance or a cross-validation approach to (partly) deal with this problem. In the maximin scaled-distance approach weights are introduced to reflect the importance of each design parameter (as identified by the fitted metamodel). In the cross-validation approach the point with the largest estimated prediction error is added to the current set of design points. Similarly, Van Beers and Kleijnen (2005) consider several candidate design points and add the point for which the estimated variance of the predicted response value is maximal, using both cross-validation and jackknifing.

In principle, sequential optimization methods (see Section 1.2) also use sequential designs. For these methods, however, the determination of new evaluation points depends on the (local) value of the objective function to optimize, instead of the validity of the global approximation model. Furthermore, methods that, after evaluating an initial design, explore interesting areas of the design space by running extra computer simulations in these areas can, within this framework, also be viewed as sequential design methods.

In Part II of this thesis we introduce *nested designs*. We call a design *nested* when it consists of m separate designs, say, $X_1, X_2, \ldots, X_m$, one being a subset of the other, i.e. $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_m$. Note that in this case the placement of additional points depends only on the current set of design points and not on the fitted metamodel. Clearly, nested designs can be considered as a type of sequential designs. For example, when $m = 2$, X_1 can be considered as the initial design, which is augmented by the design points in $X_2 \setminus X_1$, leading to the new (extended) design X_2.

More importantly, in the particular case where $m = 2$ the set X_1 could be used as

a *training set* for fitting a metamodel; set $X_2 \setminus X_1$ could then be the *test set* used for validating the obtained metamodel.

Besides acting like sequential designs or training and test sets for single black-box functions, nested designs can also be used as designs for multiple black-box functions that share some design parameters. This latter feature is very useful when dealing with collaborative optimization problems; see Part II of this thesis.

Chapter 3

Two-dimensional Latin hypercube designs

You're here again?
Kids really dig the library, don't cha?
– *We're literary!*
– *To read makes our speaking English good.*

(BtVS, *Episode 01.08*)

3.1 Introduction

In Chapter 2 we argue that a design of computer experiments should cover the entire feasible region and should not replicate any of the design parameter coordinates, i.e. the design should be space-filling and non-collapsing. To obtain good designs for computer experiments several papers combine space-filling criteria with the (non-collapsing) Latin hypercube structure, see e.g. Bates et al. (2004), Van Dam (2005), and Jin et al. (2005). Although it is impossible to define which type of design is the "best", the overall conclusion in literature tends to be that maximum entropy and distance-based criteria often lead to better designs for computer experiments than other measures, see e.g. Simpson et al. (2001), Santner et al. (2003), and Bursztyn and Steinberg (2006). Furthermore, maximin Latin hypercube designs (LHDs) are frequently used in real-life 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). This validates our choice to consider maximin Latin hypercube designs when constructing a design of computer experiments.

In the current chapter we consider two-dimensional maximin Latin hypercube designs. We derive explicit descriptions of maximin Latin hypercube designs and general formulas for the maximin distance when the distance measure is ℓ^∞ or ℓ^1. Furthermore, for

the distance measure ℓ^2 we obtain maximin Latin hypercube designs for $n \leq 70$ by using a branch-and-bound algorithm, and approximate maximin Latin hypercube designs for larger values of n. All these (approximate) maximin Latin hypercube designs can be downloaded from the website `http://www.spacefillingdesigns.nl`. As far as we know, this is the first catalogue of maximin Latin hypercube designs, although there are several catalogues for classical design of experiments, see e.g. the WebDOE$^{\text{TM}}$ website of Crary (2001). In higher dimensions we have not been able to derive explicit constructions. Nonetheless, by extending some of the ideas in the current chapter, we have obtained approximate maximin Latin hypercube designs. The construction of such designs is the subject of Chapter 4.

The problem of finding a maximin Latin hypercube design in two dimensions can easiest be described as a rook problem. This problem aims to position n rooks on an $n \times n$ chessboard, such that the rooks do not attack each other, and such that the separation distance (i.e. the minimal distance between pairs of rooks) is maximized. More formally, a two-dimensional maximin Latin hypercube design can be defined as a set of points $x_i = (x_{i1}, x_{i2}) \in \{0, 1, \ldots, n-1\}^2$, $i = 1, \ldots, n$, such that $x_{i1} \neq x_{j1}$ and $x_{i2} \neq x_{j2}$, $i \neq j$, and such that the separation distance $d = \min_{i \neq j} d(x_i, x_j)$ is maximal, where $d(\cdot, \cdot)$ is a certain distance measure. Note that in this, and the next, chapter the $[0, n-1]^k$-grid is considered, which will cause all (squared) separation distances to be integer-valued.

3.2 Maximum norm

The problem of arranging n points in the box $[0, n-1]^k$ to maximize the minimal ℓ^∞-distance between all pairs of points has been completely solved by Baer (1992). In two dimensions, i.e. $k = 2$, the corresponding maximin distance equals $d = \frac{n-1}{\lfloor \sqrt{n-1} \rfloor}$ and is attained, for example, by choosing n points from the set $\{id \,|\, i = 0, \ldots, \lfloor \sqrt{n-1} \rfloor\}^2$. This unrestricted design is of course highly collapsing (see Section 2.3), and, although there is in general some freedom to change the design to decrease the "collapsingness" (without decreasing the distance), only in the cases where $n-1$ is a square is it possible to obtain a maximin Latin hypercube design. This latter observation follows implicitly from the following construction, which attains the maximin distance, i.e. $\lfloor \sqrt{n} \rfloor$, among the set of Latin hypercube designs.

Construction 3.1 *Let n and d be positive integers such that $n \geq d^2$. Let the sequence $(t_0, t_1, \ldots, t_d)$ be defined by $t_0 = 0$ and $t_{j+1} = t_j + \left\lfloor \frac{n+j}{d} \right\rfloor$, $j = 0, \ldots, d-1$. Then*

$$X = \left\{ (id - j - 1, t_j + i - 1) \,|\, j = 0, \ldots, d-1;\ i = 1, \ldots, t_{j+1} - t_j \right\} \tag{3.1}$$

is a Latin hypercube design of n points with separation ℓ^∞-distance d.

Proof. First note that X indeed consists of $t_d = \sum_{j=0}^{d-1} \lfloor \frac{n+j}{d} \rfloor = n$ points. Since all first coordinates of the points in X are distinct elements of $\{0, 1, \ldots, n-1\}$, as are all second coordinates, it follows that X is a Latin hypercube design. From facts such as $t_{j+1} - t_j \geq d$ we find that the separation distance is d. □

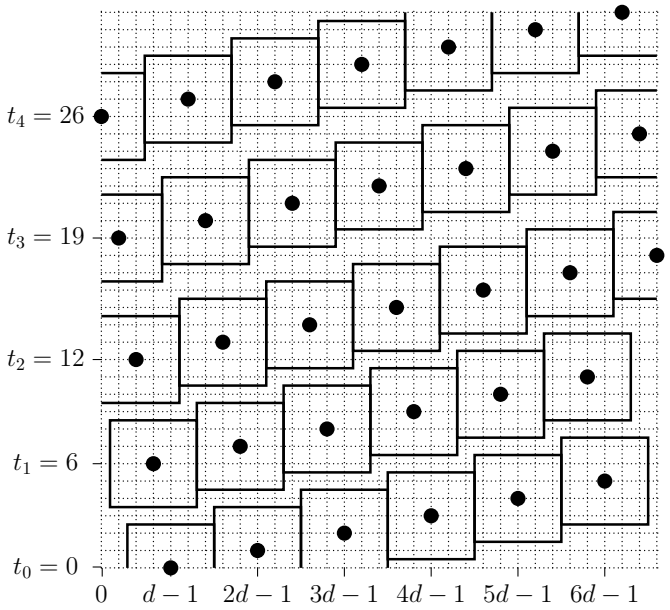


Figure 3.1: Two-dimensional ℓ^∞-maximin Latin hypercube design of 33 points; $d = 5$.

This construction (see Figure 3.1 for an example) shows that Latin hypercube designs of n points with separation distance $\lfloor \sqrt{n} \rfloor$ exist. The following proposition shows that this distance is optimal.

Proposition 3.1 *Let $n \geq 2$. An ℓ^∞-maximin Latin hypercube design of n points in two dimensions has a separation distance of $\lfloor \sqrt{n} \rfloor$.*

Proof. Consider a Latin hypercube design of n points in two dimensions, as a subset of $\{0, 1, \ldots, n-1\}^2$, with separation distance d. Consider the point $(d-1, x_{d-1,2})$ of the design. Without loss of generality we may assume that $x_{d-1,2} \leq \frac{n-1}{2}$. First, note that $x_{d-1,2} + d - 1 \leq n - 1$ because of this assumption and the easily proven fact that $d - 1 \leq \frac{n-1}{2}$. Now, the d points with second coordinates $x_{d-1,2}, x_{d-1,2}+1, \ldots, x_{d-1,2}+d-1$ must all have first coordinates in $\{d-1, d, \ldots, n-1\}$ and these coordinates must all be at least d apart. This shows that $n - d \geq (d-1)d$, and, hence, $d \leq \lfloor \sqrt{n} \rfloor$. This bound and Construction 3.1 show that a maximin Latin hypercube design of n points has a separation distance of $d = \lfloor \sqrt{n} \rfloor$. □

It is easy to see that the difference between the maximin distance for unrestricted designs and the maximin distance for Latin hypercube designs is less than two; hence, the relative difference tends to zero. For example, the reduction in the maximin distance due to the Latin hypercube constraints is less than 10% for $n \geq 324$, and less than 1% for $n \geq 39{,}204$. See also Figure 3.2, where the two maximin distances are displayed as a function of the number of points. The trade-off between space-fillingness and non-collapsingness for the maximum norm, as well as for the rectangular and Euclidean distance measure, is illustrated in more detail in Chapter 5.

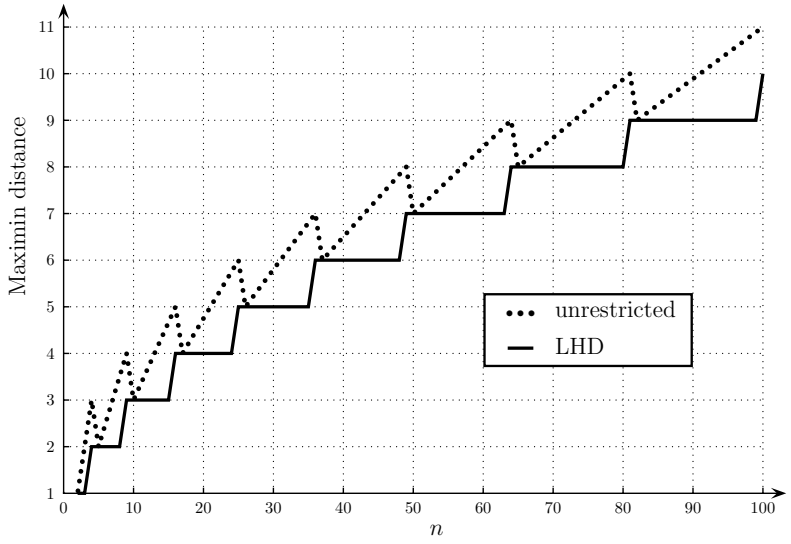


Figure 3.2: Maximin ℓ^∞-distances for unrestricted designs and for Latin hypercube designs.

3.3 Rectangular distance

For the ℓ^1-distance measure the situation is more complicated than for the ℓ^∞-distance measure. Fejes Tóth (1971) shows that the maximin distance for unrestricted designs is at most $1 + \sqrt{2n-1}$, with equality if and only if the number of points n is the sum of two consecutive squares. The unique design giving equality for $n = r^2 + (r+1)^2$, $r \in \mathbb{N}$, is the set

$$\left\{ \frac{i(n-1)}{r} \,|\, i = 0, \ldots, r \right\}^2 \bigcup \left\{ \frac{(2i+1)(n-1)}{2r} \,|\, i = 0, \ldots, r-1 \right\}^2, \tag{3.2}$$

which is highly collapsing. Also for some other values of n the maximin distance has been determined, see Florian (1989). Typically, the corresponding optimal designs are highly collapsing too; only the cases $n = 2$, 4, and 7, seem to be exceptions. For these latter cases there is an optimal design which is a Latin hypercube design. For most

(approximately "3 out of 4") values of n, however, the maximin distance for unrestricted designs has not been determined yet. Next, we derive the maximin distance explicitly for the class of Latin hypercube designs, for all n: it equals $\lfloor\sqrt{2n+2}\rfloor$. This bound is, for example, attained by the designs in the following constructions, which distinguish between even d and odd d.

Construction 3.2 *Let n and d be positive integers, d even, such that $n \geq \frac{1}{2}d^2 - 1$. Let the sequence $(t_0, t_1, \ldots, t_{d-1})$ be defined by $t_0 = 0$ and $t_{j+1} = t_j + \left\lfloor \frac{n + \frac{j}{2} + \frac{1}{2}(1-(-1)^j)(\frac{1}{2}d - \frac{1}{2})}{d-1} \right\rfloor$, $j = 0, \ldots, d-2$. Then*

$$\begin{aligned} X \;=\; \Bigg\{ & \left(i(d-1) - \frac{j}{2} - \frac{1}{2}(1-(-1)^j)(\frac{1}{2}d - \frac{1}{2}) - 1, t_j + i - 1 \right) \mid \\ & j = 0, \ldots, d-2;\ i = 1, \ldots, t_{j+1} - t_j \Bigg\} \end{aligned} \tag{3.3}$$

is a Latin hypercube design of n points with separation ℓ^1-distance d.

Proof. Also here X indeed consists of $t_{d-1} = n$ points (although it is more tedious to check). Checking that X is a Latin hypercube design with separation distance d is tedious, but routine. Important here are the facts that $t_{j+1} - t_j \geq \frac{1}{2}d$ for even j, and $t_{j+1} - t_j \geq \frac{1}{2}d + 1$ for odd j. □

Construction 3.3 *Let n and d be positive integers, d odd, such that $n \geq \frac{1}{2}d^2 - \frac{1}{2}$. Let the sequence $(s_0, s_1, \ldots, s_d)$ be defined by $s_0 = 0$ and $s_{j+1} = s_j + \left\lfloor \frac{n + \frac{j}{2} + \frac{1}{2}(1-(-1)^j)(\frac{1}{2}d)}{d} \right\rfloor$, $j = 0, \ldots, d-1$. Then*

$$\begin{aligned} X \;=\; \Bigg\{ & \left(id - \frac{j}{2} - \frac{1}{2}(1-(-1)^j)(\frac{1}{2}d) - 1, s_j + i - 1 \right) \mid \\ & j = 0, \ldots, d-1;\ i = 1, \ldots, s_{j+1} - s_j \Bigg\} \end{aligned} \tag{3.4}$$

is a Latin hypercube design of n points with separation ℓ^1-distance d.

Proof. The proof is similar as before. One can check that X has $s_d = n$ points and separation distance d by using that $s_{j+1} - s_j \geq \frac{1}{2}(d-1)$ for even j, and $s_{j+1} - s_j \geq \frac{1}{2}(d+1)$ for odd j. □

Particular examples of Constructions 3.2 and 3.3 are depicted in Figure 3.3 (d even) and Figure 3.4 (d odd), respectively. As before, these constructions can be used to construct optimal designs.

Proposition 3.2 *Let $n \geq 2$. An ℓ^1-maximin Latin hypercube design of n points in two dimensions has a separation distance of $\lfloor\sqrt{2n+2}\rfloor$.*

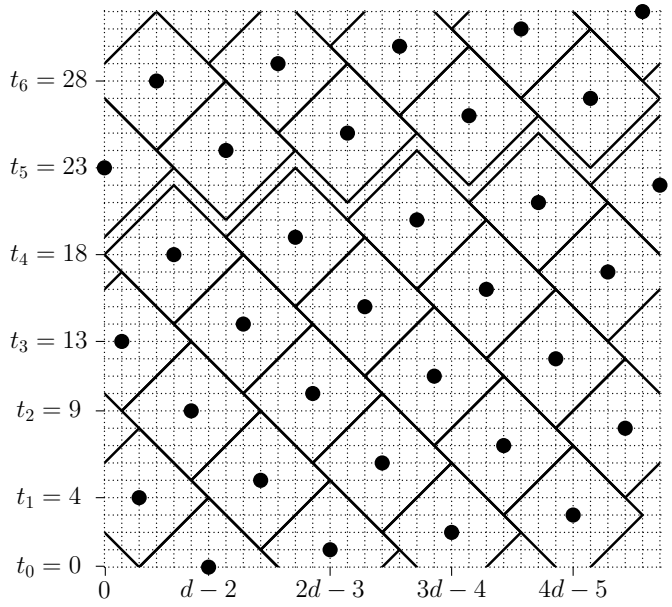


Figure 3.3: Two-dimensional ℓ^1-maximin Latin hypercube design of 33 points; $d = 8$.

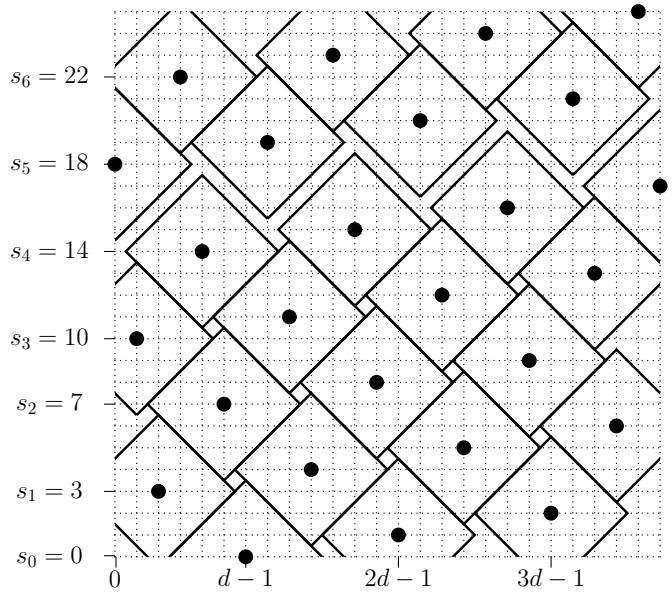


Figure 3.4: Two-dimensional ℓ^1-maximin Latin hypercube design of 26 points; $d = 7$.

Proof. We shall prove that $n \geq \frac{1}{2}d^2 - 1$ for any Latin hypercube design of n points with a separation distance of d. For $d \leq 3$ this is obvious, so we may assume that $d \geq 4$.

Consider the Latin hypercube design as a subset of $\{0, 1, \ldots, n-1\}^2$ embedded in $\mathbb{R}^2$, together with the ℓ^1-circles (diamonds) with radius $\frac{1}{2}d$ centered at the n design points; let us call these *design circles*. As the interiors of these design circles are disjoint, they cover a total area of $n \cdot \frac{1}{2}d^2$. We shall next find a bound on this total area that implies

the bound for n in terms of d.

First, let d be even and fixed. The total covered area below the (horizontal) line $h = \frac{1}{2}d-2$ is equal to

$$\frac{1}{4}d^3 - \frac{3}{4}d^2 + 1. \tag{3.5}$$

This can be seen by observing that the area below the line $h = \frac{1}{2}d - 2$ that is covered by the two design circles centered at the design points with second coordinates i and $d-4-i$ equals $\frac{1}{2}d^2$, for $i = 0, \ldots, \frac{1}{2}d-3$. What remains is to account for the areas covered by the design circles that are centered at the design points with second coordinates $\frac{1}{2}d-2$ and $d-3$, which are $\frac{1}{4}d^2$ and 1, respectively. The sum of these areas gives the expression in (3.5). It thus follows that the total covered area outside the square $[\frac{1}{2}d-2, n-\frac{1}{2}d+1]^2$ is at most d^3-3d^2+4, and we therefore find that

$$n \cdot \frac{1}{2}d^2 \leq d^3 - 3d^2 + 4 + (n-d+3)^2. \tag{3.6}$$

This equation implies that $n^2 - n(2d-6+\frac{1}{2}d^2) + d^3 - 2d^2 - 6d + 13 \geq 0$, so

$$\begin{aligned} n &\geq d - 3 + \frac{1}{4}d^2 + \frac{1}{4}\sqrt{d^4 - 8d^3 + 24d^2 - 64} \\ &> d - 3 + \frac{1}{4}d^2 + \frac{1}{4}\sqrt{d^4 - 8d^3 + 24d^2 - 32d + 16} = \frac{1}{2}d^2 - 2, \end{aligned} \tag{3.7}$$

which proves that $n \geq \frac{1}{2}d^2 - 1$. Note that we used that $d \geq 4$ to obtain the last inequality, and that the case where $n \leq d - 3 + \frac{1}{4}d^2 - \frac{1}{4}\sqrt{d^4 - 8d^3 + 24d^2 - 64} < 2d - 4$ is easily excluded.

Next, let d be odd and fixed. As above, we first find that the total covered area below the line $h = \frac{1}{2}(d-5)$ equals

$$\frac{1}{4}d^3 - d^2 + \frac{5}{2}. \tag{3.8}$$

As before, this can be seen by observing that the area below the line $h = \frac{1}{2}(d-5)$ that is covered by the two design circles centered at the design points with second coordinates i and $d-5-i$ is equal to $\frac{1}{2}d^2$, for $i = 0, \ldots, \frac{1}{2}(d-5)-1$. The areas covered by the design circles that are centered at the design points with second coordinates $\frac{1}{2}(d-5)$, $d-4$, and $d-3$, are $\frac{1}{4}d^2$, $\frac{9}{4}$, and $\frac{1}{4}$, respectively. The sum of these areas results in the expression in (3.8). It follows that the total covered area outside the square $[\frac{1}{2}(d-5), n-\frac{1}{2}d+\frac{3}{2}]^2$ is at most $d^3 - 4d^2 + 10$.

In order to derive a useful inequality we have to look more carefully at the covered area inside the above-mentioned square. We claim that each design point $x_i = (x_{i1}, x_{i2})$ has the property that the interior of at least one of the two ℓ^1-circles with radius $\frac{1}{2}$, centered at $(x_{i1} - \frac{1}{2}, x_{i2} + \frac{1}{2}d)$ and $(x_{i1} + \frac{1}{2}, x_{i2} + \frac{1}{2}d)$, is not covered, and we call such

an uncovered circle a hole (such holes can clearly be identified in Figure 3.4). Indeed, a design circle that covers any of these two mentioned smaller circles also covers the circle with radius $\frac{1}{2}$ around $(x_{i1}, x_{i2} + \frac{1}{2}(d+1))$. Since the two small circles clearly cannot be covered by the same design circle, this proves the claim. We note now that the interiors of all holes are disjoint, and, moreover, all holes lie above the line $h = \frac{1}{2}(d-5)$. Since there are $d-2$ design points with holes above the line $h = n - \frac{1}{2}d + \frac{3}{2}$, there are at least $n - d + 3 - (d-2) = n - 2d + 5$ holes (among those coming from design points with first coordinates $\frac{1}{2}(d-5)+1, \ldots, n - \frac{1}{2}d + \frac{1}{2}$) that lie entirely inside the square $[\frac{1}{2}(d-5), n - \frac{1}{2}d + \frac{3}{2}]^2$. We thus obtain that

$$n \cdot \frac{1}{2}d^2 \leq d^3 - 4d^2 + 10 + (n-d+4)^2 - \frac{1}{2}(n - 2d + 5), \tag{3.9}$$

which implies that $n^2 - n(2d - \frac{15}{2} + \frac{1}{2}d^2) + d^3 - 3d^2 - 7d + \frac{47}{2} \geq 0$. Therefore,

$$\begin{aligned} n &\geq d - \frac{15}{4} + \frac{1}{4}d^2 + \frac{1}{4}\sqrt{d^4 - 8d^3 + 34d^2 - 8d - 151} \\ &> d - \frac{15}{4} + \frac{1}{4}d^2 + \frac{1}{4}\sqrt{d^4 - 8d^3 + 34d^2 - 72d + 81} = \frac{1}{2}d^2 - \frac{3}{2}, \end{aligned} \tag{3.10}$$

and, hence, $n \geq \frac{1}{2}d^2 - 1$.

To obtain the inequality in (3.10) we used that $d \geq 4$; the case $n \leq d - \frac{15}{4} + \frac{1}{4}d^2 - \frac{1}{4}\sqrt{d^4 - 8d^3 + 34d^2 - 8d - 151} < 2d - 6$ is easily excluded. We have thus proven the inequality $n \geq \frac{1}{2}d^2 - 1$ for all d, and, hence, that $d \leq \lfloor \sqrt{2n+2} \rfloor$. Constructions 3.2 and 3.3 show that equality can be attained. □

The difference between the maximin distance for unrestricted designs and the maximin distance for Latin hypercube designs is again less than two. The reduction in the maximin distance due to the Latin hypercube constraints is less than 10% for $n \geq 144$, and less than 1% for $n \geq 19{,}404$. See also Figure 3.5, where the maximin distance for Latin hypercube designs and the upper bound/exact value for the maximin distance for unrestricted designs are displayed as a function of the number of points.

3.4 Euclidean distance

Sections 3.2 and 3.3 consider maximin designs for the ℓ^∞- and ℓ^1-distance measures, respectively. For many real-world applications, however, the ℓ^2-distance measure is often the first choice. Unfortunately, for this Euclidean distance measure the situation is much more complicated than for the other two measures. There is no known infinite class of optimal designs in the unrestricted situation, as is the case, for instance, for the ℓ^1-measure, let alone a complete solution like for the ℓ^∞-measure. Optimal designs are known only for up to 30 points and the single case of 36 points. Melissen (1997) summarizes the optimal

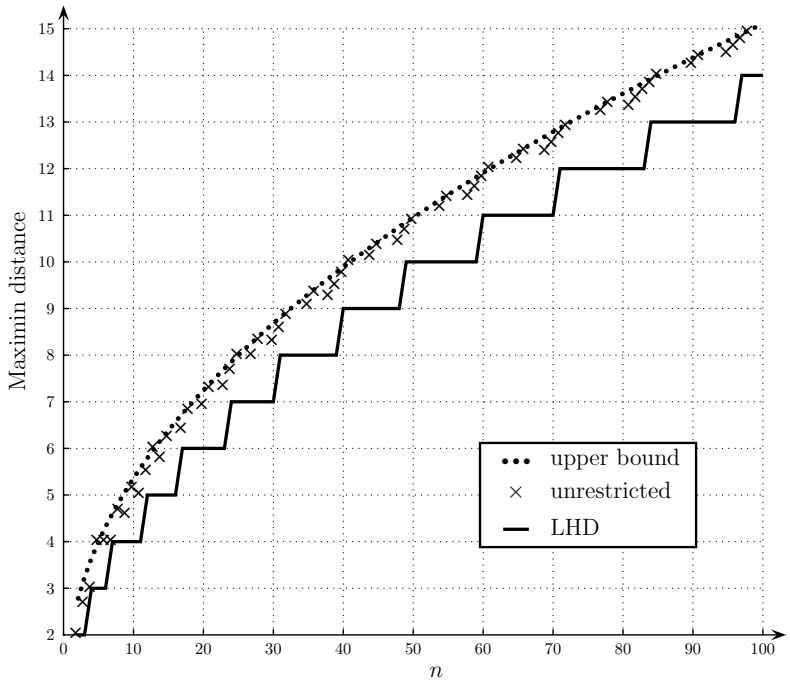


Figure 3.5: General upper bound for maximin ℓ^1-distance and maximin ℓ^1-distances for unrestricted designs and Latin hypercube designs.

arrangements of $n \leq 20$ points and Kirchner and Wengerodt (1987) provide the proof for the case of $n = 36$. Many of the designs require dedicated optimality proofs and some of the larger cases have even been proven by computer-assisted proof techniques, see e.g. Peikert et al. (1991) ($n = 11$–$13, 15, 17$–20), Nurmela and Östergård (1999) ($n = 21$–27), and Markót and Csendes (2005) ($n = 28$–30). The optimal designs may be devoid of any symmetry or nice structure (for instance, for 10 or 13 points), and there can be multiple optimal solutions (e.g. for 17 points). Moreover, like in the case of the ℓ^∞- and ℓ^1-distance measures, there are even optimal designs that have points that are not fixed, but that can move around a little (for instance, for 7, 11, and 13 points). These so-called rattlers have already been identified in Section 2.2.1; see e.g. Figure 2.1.

As there are no general results for maximin designs in the ℓ^2-measure, this is still a field of research where world records can be broken, see e.g. Casado et al. (2001). A list of the best-known circle packings in a square (and also in a circle and in a rectangle) is on the Packomania website, maintained by Specht (2005). So far, the list contains many very good (and probably close to optimal) designs for up to 300 points, and a few larger numbers. This supports the belief that a complete solution for all points is not likely to be ever found. To a lesser extent, the same seems to be the case for the problem of finding ℓ^2-maximin Latin hypercube designs.

3.4.1 Branch-and-bound

To find maximin Latin hypercube designs for the ℓ^2-distance measure (for small n), we designed a branch-and-bound algorithm. This algorithm searches for Latin hypercube designs of n points with a separation distance of at least d, for given n and d, by examining all designs $\{(i, z_i) \mid i = 0, \ldots, n-1\}$, represented by the sequence $(z_0, z_1, \ldots, z_{n-1}) \in \{0, 1, \ldots, n-1\}^n$, while checking whether they are non-collapsing and have a separation distance of at least d. Note that this formulation implies that the coordinates of the first dimension are, without loss of generality, fixed to the sequence $(0, 1, \ldots, n-1)$.

As a first approach, one could use the search tree where the root has n branches, giving the value of z_0, and each corresponding node further branches into n parts, giving the value of z_1, et cetera, until the end nodes are reached, giving the value of z_{n-1}. One can cut branches from the node corresponding to the partial design $(z_0, z_1, \ldots, z_t)$ if points are already collapsing or are separated by a distance less than d. In this way, we obtained maximin Latin hypercube designs for n up to 40.

A disadvantage of the above approach is that it does not use the fact that useless partial designs occur as part of other partial designs (for example, $(0, 3, 4)$ is part of $(9, 12, 15, 0, 3, 4)$) in different parts of the tree, and, hence, are not cut off by just one cut. Note also, in this respect, that it is beneficial to cut the tree at small depth. To (partly) solve this disadvantage, a different tree is used. For this, the value of z_i is first fixed to $z_i = z \neq \frac{n-1}{2}$, where the index i will be determined later, and will depend on the particular end node in the tree. Because of symmetry, i is assumed to be at most $f = \lfloor \frac{n}{2} \rfloor - 1$. This z_i will be the root of the tree, and it branches into n parts, giving the value of z_{i+1}. The corresponding nodes further branch into n parts, giving the value of z_{i+2}, et cetera, up to the nodes giving the value of z_{i+n-1} (and at these end nodes we take $i = 0$). Moreover, for $t = 0, \ldots, f-1$, the nodes corresponding to the value of $z_{i+n-1-f+t}$ (roughly speaking: when over "half" of the points in the design are chosen) have n additional branches giving the value of z_{i-1} (we now start extending the partial design on the other side of i), and these branch further corresponding to the values of z_{i-2}, et cetera, up to the values of z_{i-f+t}. Taking $i = f - t$ at these end nodes yields the design coordinates $(z_0, z_1, \ldots, z_{n-1})$ for the second dimension. With the branch-and-bound algorithm based on this tree we managed to find optimal designs, or prove optimality of some designs found by hand, for $n \leq 70$ by taking $z = \lfloor \sqrt{d^2 - 2} \rfloor$ (but this value does not seem to be crucial). For the instance $(n, d) = (69, \sqrt{80})$ we took $z = \frac{n-1}{2}$. This has the advantage that, because of symmetry, only the cases $z_{i+1} < z$ (i.e. only half the tree) have to be searched; however, the disadvantage is that also the value $i = \frac{n-1}{2}$ must be considered (this is implemented by letting $f = \frac{n-1}{2}$). In this particular case there was no disadvantage since all cutting turned out to be performed far before half of the points in the design were chosen.

Using these branch-and-bound techniques we were able to obtain maximin Latin hy-

percube designs for up to 70 points. These maximin Latin hypercube designs, which were also found by our periodic construction method (see Section 3.4.2), can be derived from Table 3.1 displayed below. Unlike the situation without Latin hypercube constraints, many of the optimal designs exhibit some nice regularity, i.e. the designs turn out to be either periodic arrangements or slightly adapted periodic arrangements. For example, see the ℓ^2-maximin Latin hypercube designs depicted in Figures 3.6 and 3.7.

Remark: we have learned that similar reduction techniques, as the ones described above, have been used for the (unrestricted) packing problem, see Markót and Csendes (2005).

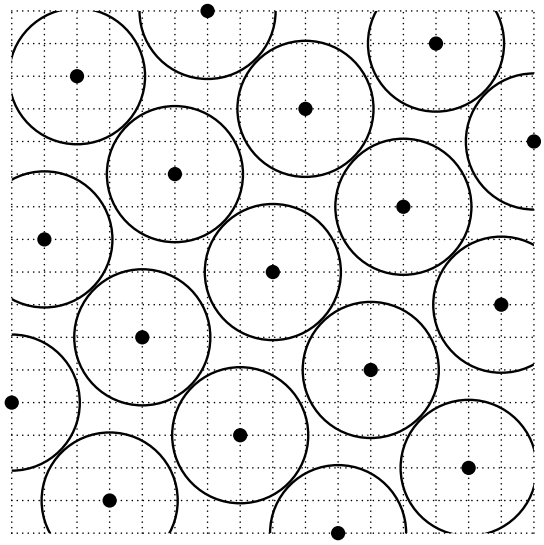


Figure 3.6: Two-dimensional ℓ^2-maximin Latin hypercube design of 17 points; $d^2 = 18$.

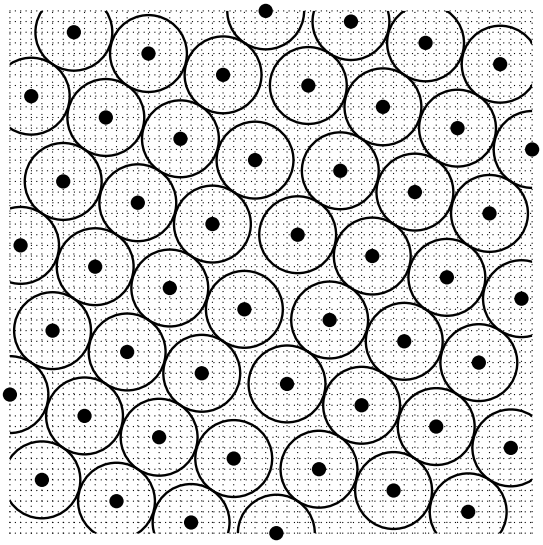


Figure 3.7: Two-dimensional ℓ^2-maximin Latin hypercube design of 50 points; $d^2 = 52$.

3.4.2 Heuristics

Due to increasing computational effort as n increases, the applicability of the presented branch-and-bound algorithm is restricted to smaller designs. To extend the range of designs several heuristics have been tried.

One option, for instance, is to consider the ℓ^∞- and ℓ^1-maximin Latin hypercube designs. When picking the best of the two, with respect to the ℓ^2-measure, some good designs are obtained. We have tried simulated annealing to improve these designs. In our algorithm a neighborhood-solution is obtained by randomly selecting two points, one of them being a point at separation distance to another point, and then switch one of the coordinate values. The performance of the neighborhood-solution is defined by the minimal distance of these two new points to all other points. Unfortunately,

when starting from the initial ℓ^∞- and ℓ^1-designs the algorithm could not produce better ones. When starting from a random design the algorithm consumed excessive amounts of computation time without turning up solutions that were at least as good as the ℓ^∞- or ℓ^1-designs. In higher dimensions, however, our algorithm *did* result in good Latin hypercube designs. This algorithm, and the reason why it works in higher dimensions, is discussed in Chapter 4.

Another approach uses the nice, periodic structure of many of the maximin Latin hypercube designs that have been found by the branch-and-bound algorithm, and looks for periodic designs. Following the definition in Section 2.3.1, a two-dimensional Latin hypercube design of n points can be represented as an $n \times 2$ matrix, where the columns y_1 and y_2 are permutations of the set $\{0, 1, \ldots, n-1\}$. Without loss of generality, y_1 is fixed to the sequence $(0, 1, \ldots, n-1)$. What remains is to decide on a permutation y_2 such that the separation distance of the corresponding Latin hypercube design is maximal. The use of a periodic sequence for y_2 turned out to be very successful.

For given n, start with choosing a period p such that $\gcd(n+1, p) = 1$ and construct a Latin hypercube design with points (y_{1i}, y_{2i}), where $y_{1i} = i$ and

$$y_{2i} = (i+1)p \bmod (n+1) - 1, \text{ for } i = 0, \ldots, n-1. \tag{3.11}$$

This heuristic often resulted in maximin Latin hypercube designs and, in other instances, good designs.

Note that the periodic designs obtained in this way resemble *lattices*; see e.g. Bates et al. (1996). The design points in Figure 3.6, for example, are a subset of the lattice points generated by the primitive vectors $(1, 5)$ and $(4, 2)$. The main difference is that lattices are infinite sets of points, which may collapse, and, hence, to construct a (finite) Latin hypercube design a proper subset of non-collapsing lattice points should be chosen. For given n, the structure of the lattice, however, will not always lead to a Latin hypercube design with a sufficient number of points. This in contrast to periodic designs, for which the modulo-operator insures that for every period p, with $\gcd(n+1, p) = 1$, a feasible Latin hypercube design exists.

To improve the results obtained with the sequence in (3.11), consider the more general sequence $w_i = (s + ip) \bmod n$ (note that the modulus has been changed), for all periods $p = 1, \ldots, \lfloor \frac{n}{2} \rfloor$, and different starting points $s = 0, \ldots, \lfloor \frac{n}{2} \rfloor$. 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 $\{(i, w_i) \,|\, i = 0, \ldots, n-1\}$ may not be a Latin hypercube design. 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, and, hence, a Latin hypercube design, is by shifting parts of the sequence by, say, q, and repeating this when necessary. The adapted periodic sequence y_2 is then given by

the updated sequence w, for which it holds that

$$w_i = (s+ip+jq) \bmod n, \text{ for } i = jr, \ldots, (j+1)r-1, \text{ and } j = 0, \ldots, \gcd(n,p)-1. \quad (3.12)$$

For n up to 200 points all "shifts" q, with q such that $\gcd(q, \gcd(n,p)) = 1$, in the range $[1-p, p-1]$ and all starting points $s = 0, \ldots, \lfloor \frac{n}{2} \rfloor$ were tested. It turned out that taking q equal to either $1-p$ or -1, and s equal to $p-1$, yielded the best designs. Additional tests indicated that the value $q = 1$ should also be considered. Therefore, the final heuristic considered only $q \in \{1-p, -1, 1\}$ and $s = p-1$.

Combining both periodic construction methods we found results for n up to 1000; the obtained Latin hypercube designs for $n \leq 70$ are optimal. The designs, with their corresponding maximin distances, are provided in Table 3.1. In this table the tuple (p, q, m) defines the permutation y_2 of the Latin hypercube design (y_1, y_2). If $m = n+1$ then y_2 is given by (3.11), whereas it is equal to the sequence defined in (3.12) when $m = n$.

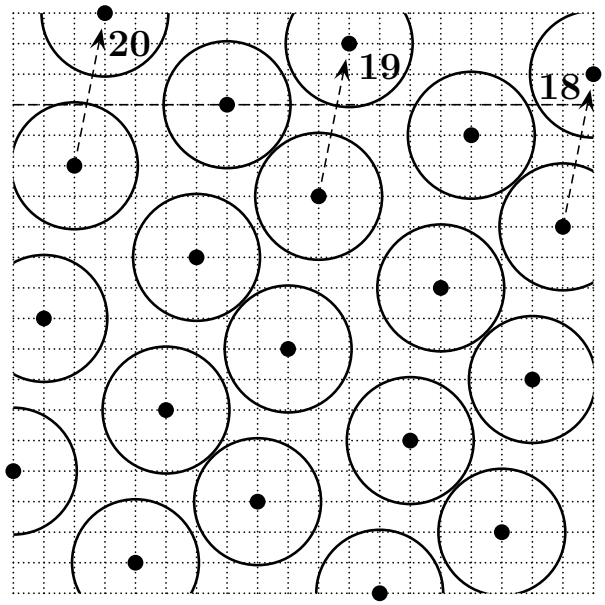


Figure 3.8: Latin hypercube design constructions for 18, 19, and 20 points, based on an ℓ^2-maximin Latin hypercube design of 17 points; $d^2 = 18$.

Table 3.1 provides only designs for which n is a *break point*, i.e. the values of n for which $d_n^2 > d_i^2$, for all $i < n$. Periodic designs for intermediate values of n may have a minimal distance that is smaller than the minimal distance of their preceding break point. For these n, however, better designs can easily be derived. Every two-dimensional (adapted) periodic Latin hypercube design of n points, with n a break point, is defined by its periodic sequence of y_2-values, which can be split up into several increasing subsequences. For example, the ℓ^2-maximin Latin hypercube design of 17 points in Figure 3.6 consists of

n	d^2	p	q	m
2	2	1	–	3
4	5	2	–	5
7	8	3	–	8
9	10	3	–	10
12	13	5	–	13
14	17	4	–	15
17	18	5	–	18
21	20	5	–	22
22	25	5	–	23
23	26	5	–	24
28	29	12	–	29
31	32	7	–	32
33	34	13	–	34
34	37	6	–	35
38	41	7	–	39
44	50	19	–	45
50	52	14	−13	50
52	58	8	–	53
58	61	9	–	59
60	65	8	–	61
65	68	25	–	66
67	74	9	–	68
75	80	9	–	76
76	85	34	–	77
83	90	25	–	84
86	97	10	−9	86
90	98	27	–	91
93	100	11	–	94
95	101	10	−1	95
100	109	30	–	101
102	113	28	−27	102
104	117	11	–	105
111	128	41	–	112
121	130	51	–	122
126	145	12	–	127
136	149	13	–	137
146	157	56	−55	146
148	160	34	−1	148
149	170	13	–	150
156	178	36	–	157
162	180	14	−13	162
166	181	36	–	167
170	185	52	−51	170
171	194	37	–	172
176	197	14	−13	176
180	202	39	–	181
184	205	66	−65	184
187	208	15	–	188
194	212	52	−51	194
200	218	16	–	201

n	d^2	p	q	m
202	226	15	–	203
208	241	56	–	209
216	245	16	–	217
225	250	99	–	226
232	257	16	–	233
240	269	71	–	241
246	277	17	–	247
253	290	45	–	254
260	292	46	−45	260
267	296	79	–	268
268	305	63	–	269
279	306	18	−1	279
280	320	18	−17	280
291	328	81	–	292
298	338	116	–	299
306	346	113	–	307
313	356	19	–	314
324	360	51	−1	324
326	365	120	−119	326
330	370	20	–	331
335	386	71	–	336
350	401	20	–	351
358	409	54	–	359
367	410	21	–	368
374	425	118	–	375
388	442	21	–	389
395	450	139	–	396
408	461	22	−21	408
415	466	79	–	416
422	481	96	−95	422
429	482	22	−1	429
430	485	22	−21	430
433	490	59	–	434
448	509	61	–	449
462	530	141	–	463
470	533	193	–	471
474	545	62	−61	474
488	549	64	–	489
492	565	86	–	493
509	578	89	–	510
520	586	136	1	520
534	593	64	–	535
537	610	25	–	538
550	613	154	–	551
552	629	199	–	553
559	640	67	–	560
575	650	155	–	576
582	661	93	–	583
586	673	26	−25	586
600	674	168	–	601

n	d^2	p	q	m
607	680	27	–	608
613	692	71	–	614
626	722	265	–	627
634	725	27	–	635
641	738	119	–	642
658	745	28	–	659
666	746	119	–	667
672	761	100	–	673
678	765	130	−129	678
679	778	101	–	680
686	785	28	–	687
694	793	124	–	695
706	808	288	−287	706
710	809	76	–	711
717	818	249	–	718
730	820	78	−77	730
732	829	76	–	733
738	850	192	–	739
756	853	209	–	757
758	865	340	−339	758
761	866	79	–	762
766	872	30	−29	766
776	882	295	–	777
777	884	107	–	778
783	898	183	–	784
795	901	30	−1	795
800	909	187	–	801
808	914	287	–	809
814	925	169	–	815
821	932	31	–	822
828	949	266	–	829
840	954	298	−297	840
843	962	175	–	844
850	977	205	–	851
866	981	196	–	867
875	986	137	–	876
880	1009	32	–	881
888	1013	115	–	889
896	1025	116	–	897
914	1037	194	–	915
919	1042	119	–	920
922	1060	268	−267	922
940	1073	33	–	941
957	1076	145	–	958
962	1090	204	−1	962
970	1105	147	–	971
985	1124	277	–	986
998	1129	258	−257	998

Table 3.1: Two-dimensional (adapted) periodic (approximate) ℓ^2-maximin Latin hypercube designs on break points.

the sequences $(4, 9, 14)$, $(1, 6, 11, 16)$, $(3, 8, 13)$, $(0, 5, 10, 15)$, and $(2, 7, 12)$. Each of these sequences can be augmented by extra points, starting with the sequence with the smallest end value (i.e. 12 in the example above), while retaining the minimal distance. Hence, a given periodic Latin hypercube design of n points can be extended to an LHD of $\hat{n} > n$ points with the same minimal distance. Figure 3.8 shows how to extend an ℓ^2-maximin Latin hypercube design of 17 points, with $d^2 = 18$, to ℓ^2-maximin LHDs of 18, 19, and 20 points, all with d^2 equal to 18. The Latin hypercube design of 17 points could also be

extended further to LHDs of $\hat{n} \geq 21$ points with $d^2 = 18$; however, Table 3.1 shows that this is no longer optimal.

Figure 3.9 displays the best found ℓ^2-distances d for unrestricted designs and Latin hypercube designs for up to 300 points. The upper bound depicted in this figure can easily be derived when applying Oler's theorem (cf. Oler (1961)) to the square $[0, n-1]^2$, resulting in:

$$d \leq 1 + \sqrt{1 + (n-1)\frac{2}{\sqrt{3}}}. \tag{3.13}$$

Like in the case of the ℓ^∞- and the ℓ^1-distance measure, the reduction in the maximin distance caused by imposing the Latin hypercube structure is small, see also Chapter 5. This justifies the use of maximin Latin hypercube designs instead of unrestricted designs.

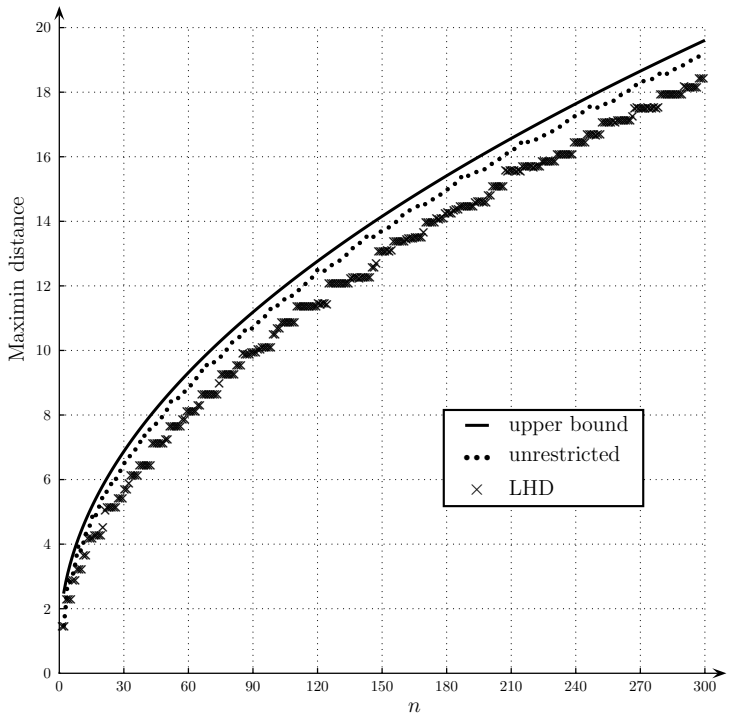


Figure 3.9: General upper bound for maximin ℓ^2-distance d and approximate maximin ℓ^2-distances for unrestricted designs and Latin hypercube designs.

Chapter 4

High-dimensional Latin hypercube designs

> To be absolutely certain about something, one must know everything, or nothing, about it.
>
> (Olin Miller)

4.1 Introduction

Chapters 1 and 2 discuss the practical importance of a proper design of computer experiments. It has been argued that such a design should be space-filling and non-collapsing. To obtain non-collapsing designs the Latin hypercube structure is often enforced. Within this class of Latin hypercube designs (LHDs), the use of maximum entropy or distance-based criteria will result in good space-filling designs for computer experiments. Chapter 3 introduces ways to construct two-dimensional maximin Latin hypercube designs for the maximum norm, the rectangular distance, and the Euclidean distance measure. Since the latter distance measure is most often the first choice in practice, the current chapter focuses on constructing approximate ℓ^2-maximin Latin hypercube designs.

Following the definition in Section 2.3.1, a k-dimensional Latin hypercube design of n points is a set of n design points $x_i = (x_{i1}, x_{i2}, \ldots, x_{ik}) \in \{0, 1, \ldots, n-1\}^k$, $i = 1, \ldots, n$, such that for each dimension j all x_{ij} are distinct, $j = 1, \ldots, k$. Or, put differently, a k-dimensional Latin hypercube design of n points can be represented by a matrix that consists of k columns y_j, $j = 1, \ldots, k$, where each column y_j is a permutation of the set $\{0, 1, \ldots, n-1\}$. A Latin hypercube design is called maximin when the separation distance $\min_{i \neq j} d(x_i, x_j)$ is maximal among all Latin hypercube designs of given size n,

where $d(\cdot,\cdot)$ is a certain distance measure. In this chapter the squared Euclidean distance measure is considered, i.e.

$$d^2(x_i, x_j) = \sum_{l=1}^{k} (x_{il} - x_{jl})^2. \tag{4.1}$$

Furthermore, the idea of (adapted) periodic designs, as introduced in Chapter 3, is extended to more than two dimensions, and a simulated annealing algorithm, with a special neighborhood structure, is proposed, to obtain approximate maximin Latin hypercube designs for up to ten dimensions and for up to 100 design points.

4.2 Periodic designs

Chapter 3 shows that two-dimensional maximin Latin hypercube designs often exhibit a nice, periodic structure. By constructing (adapted) periodic designs, many maximin Latin hypercube designs, and, otherwise, good LHDs, have been found for up to 1000 points. Therefore, extending this idea to higher dimensions seems natural.

Consider the sequences $y_1, y_2, \ldots, y_k$, with every y_j a permutation of the set $\{0, 1, \ldots, n-1\}$, that define a k-dimensional Latin hypercube design of n points. As in the two-dimensional case, a design is constructed by fixing the first dimension, without loss of generality, to the sequence $y_1 = (0, 1, \ldots, n-1)$ and assigning (adapted) periodic sequences to all other dimensions. Two types of periodic sequences are considered. The first one is the sequence $(v_0, v_1, \ldots, v_{n-1})$, where

$$v_i = (i+1)p \bmod (n+1) - 1, \text{ for } i = 0, \ldots, n-1. \tag{4.2}$$

Here, p is the period of the sequence, which is chosen such that $\gcd(n+1, p) = 1$, resulting in a permutation of the set $\{0, 1, \ldots, n-1\}$. The periodic designs obtained in this way resemble k-dimensional lattices; see Section 3.4.2.

The second type of sequence that is considered is the more general sequence $(w_0, w_1, \ldots, w_{n-1})$, where $w_i = (s + ip) \bmod n$ (note that the modulus has been changed), for $i = 0, \ldots, n-1$. In this case, all starting points $s = 0, \ldots, p$ and all periods $p = 1, \ldots, \lfloor \frac{n}{2} \rfloor$ are considered. As is the case in two dimensions, the resulting sequence w may no longer be one-to-one, resulting in a non-LHD. To deal with this problem, we consider the following updated unique sequence w (see Section 3.4.2):

$$w_i = (s + ip + jq) \bmod n, \text{ for } i = jr, \ldots, (j+1)r - 1, \text{ and } j = 0, \ldots, \gcd(n, p) - 1. \tag{4.3}$$

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 type. For given n, the parameters (p, q, s, m) have to be set for every sequence $y_2, y_3, \ldots, y_k$. To find the best settings for these parameters, it would be best to test all possible values. However, when the

dimension and the number of points increase, the number of possible values 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 and used throughout the whole process. 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 $m \in \{n, n + 1\}$. Testing in three and four dimensions indicated that almost all adapted periodic designs are based on a shift of $1 - p$, -1, or 1 (as was the case for two dimensions; see Section 3.4.2). Furthermore, most Latin hypercube 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 4.1 shows the different classes used in the computations for each dimension.

Dimension	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 4.1: Different classes of periodic sequences are considered for each dimension.

As an example, consider a three-dimensional adapted periodic Latin hypercube design of 22 points. A best parameter setting was 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 Latin hypercube design, with (squared) separation distance 69, is defined by the sequences

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

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 Latin hypercube design in Figure 4.1. 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 Latin hypercube design has a separation distance that is smaller than the distance of an

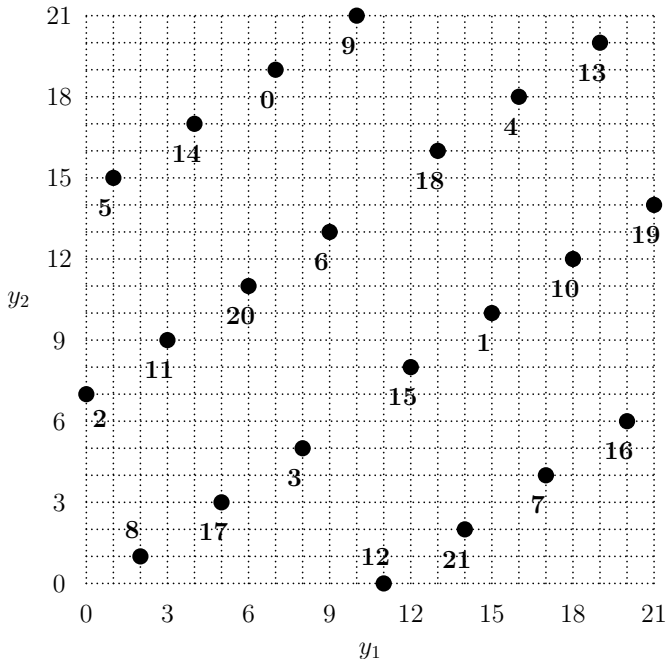


Figure 4.1: Two-dimensional projection of a three-dimensional Latin hypercube design (y_1, y_2, y_3) of 22 points; $d^2 = 69$. The y_3-values are depicted at the design points.

LHD of $n-1$ points. For these n, however, better designs can be derived by adding an extra point (usually a *corner point*) to the Latin hypercube design of $n-1$ points. In this way, a monotone non-decreasing sequence of separation distances has been obtained for all dimensions; see Table 4.3.

4.3 Simulated annealing

Another heuristic method that can be used to approximate ℓ^2-maximin Latin hypercube designs is simulated annealing; see Aarts and Lenstra (1997). The general simulated annealing algorithm that we have implemented is described in Algorithm 4.1. In this algorithm, the acceptance probability function, the annealing schedule, the terminating condition, and the neighborhood, still need to be specified. These parameters influence the performance of the algorithm, and, hence, should be carefully set. In our implementation, we focus on the choice of the neighborhood and the terminating condition.

For the acceptance probability function the commonly used classic formula of Kirkpatrick et al. (1983) is taken:

$$P\left(E_{\text{current}}, E_{\text{neighbor}}, T\right) = \exp\left(\frac{E_{\text{current}} - E_{\text{neighbor}}}{T}\right), \tag{4.4}$$

where E_{current} and E_{neighbor} are the separation distances of the current Latin hypercube

Algorithm 4.1 General simulated annealing algorithm for approximating ℓ^2-maximin Latin hypercube designs (LHDs).

```
Randomly select an initial LHD and calculate its separation distance;
best LHD = initial LHD;
REPEAT
  Create a neighbor LHD of the current LHD;
  Calculate separation distance of the neighbor LHD;
  IF (separation distance of the neighbor LHD ≥ separation distance of
      the current LHD) THEN
    current LHD = neighbor LHD;
    IF (separation distance of the current LHD ≥ separation distance of
        the best LHD) THEN
      best LHD = current LHD;
    END
  ELSE (with probability depending on the annealing temperature and the
        difference in separation distance)
    current LHD = neighbor LHD;
  END
  Update the annealing temperature;
UNTIL (terminating condition is met).
```

design and the neighbor LHD, respectively, and T is the annealing temperature.

The implemented annealing schedule starts with an initial temperature of 5. After each iteration the annealing temperature is decreased by 0.1 percent, as long as the temperature is at least 0.5 (otherwise, it remains 0.5). Furthermore, every 1,000 iterations the algorithm checks the number of improvements on the best solution found so far. If there are no improvements during the last 1,000 iterations, the temperature is reset by multiplying it by 2.7, which is approximately $0.999^{-1,000}$, i.e. the inverse of 1,000 times a 0.1% decrease.

Four different terminating conditions have been tested. The first two conditions terminate the algorithm after a fixed number of 25,000 and 50,000 iterations, respectively. The third and fourth condition let the number of iterations depend on the results obtained by the algorithm in the following way. Every 1,000 iterations it is checked whether the best design has improved. If during five subsequent checks (i.e. during the last 5,000 iterations) no improvement has occurred, the algorithm terminates. To avoid excessive running times, the total number of iterations is limited to 125,000 and 250,000 in the third and fourth condition, respectively.

For the definition of a neighborhood four different choices have been considered. In all four neighborhoods the main idea is to change two design points of the current Latin

hypercube design by exchanging one or more of their coordinate values. In three of the four neighborhoods one point is required to be a critical point. A critical point is a point which is at separation distance to at least one of the other points.

In the first neighborhood, one design point x_i is randomly selected from all critical points and the other design point x_j randomly from all remaining points. This implies that the second design point will either be a critical or a non-critical point. Once the design points have been selected, the number of coordinates to be changed is randomly determined. Due to symmetry, at most $\lfloor \frac{k}{2} \rfloor$ coordinates are changed. Subsequently, the coordinates to be changed are randomly selected. The values of the two design points at these coordinates are then exchanged, which results in a new Latin hypercube design, i.e. a neighbor LHD.

As an example, consider the four-dimensional Latin hypercube design of 10 points defined by the sequences

$$\begin{aligned} y_1 &= (5,\ 6,\ 9,\ 3,\ 1,\ 4,\ 2,\ 8,\ 0,\ 7),\\ y_2 &= (4,\ 5,\ 8,\ 6,\ 0,\ 2,\ 9,\ 7,\ 3,\ 1),\\ y_3 &= (0,\ 4,\ 6,\ 1,\ 9,\ 7,\ 3,\ 5,\ 2,\ 8),\\ y_4 &= (2,\ 3,\ 6,\ 5,\ 4,\ 9,\ 0,\ 7,\ 8,\ 1). \end{aligned}$$

The critical points of this design are the design points x_3 and x_8, i.e. $(9, 8, 6, 6)$ and $(8, 7, 5, 7)$. If the critical point x_8, the random point x_4, and the coordinates 2 and 3, are selected, the following neighbor is obtained:

$$\begin{aligned} y_1 &= (5,\ 6,\ 9,\ \ 3,\ \ 1,\ 4,\ 2,\ \ 8,\ \ 0,\ 7),\\ y_2 &= (4,\ 5,\ 8,\ \boxed{7},\ 0,\ 2,\ 9,\ \boxed{6},\ 3,\ 1),\\ y_3 &= (0,\ 4,\ 6,\ \boxed{5},\ 9,\ 7,\ 3,\ \boxed{1},\ 2,\ 8),\\ y_4 &= (2,\ 3,\ 6,\ \ 5,\ \ 4,\ 9,\ 0,\ \ 7,\ \ 8,\ 1). \end{aligned}$$

The second neighborhood is very similar to the first. The only difference is that only one coordinate is exchanged, instead of a random number of coordinates. Note that for $k = 3$ both neighborhoods are the same.

In the third neighborhood also one coordinate is exchanged. However, this time the coordinate is not randomly selected. Instead, all coordinate changes are tried and the one that results in the neighbor with the largest separation distance is executed. If there are more coordinate changes that result in the same (largest) separation distance, the coordinate with the lowest index is selected.

The fourth neighborhood is again very similar to the second neighborhood. The difference is that the first point is randomly chosen from all design points, instead of only the critical points.

Although the described approach appears to be quite similar to simulated annealing algorithms for finding good Latin hypercube designs used by other authors, it is

different in the following ways. First, our approach does not impose a certain additional structure on the Latin hypercube design, such as, for instance, symmetry; see e.g. Ye et al. (2000). Secondly, the maximin distance criterion is used as the objective function. This in contrast to the approach of, for example, Morris and Mitchell (1995), who minimize a surrogate measure. Their reason for using a surrogate measure is to minimize the number of critical points. The main disadvantage of this measure, however, is that it contains an extra parameter, which needs to be set for every value of k and n. An inaccurate setting of this parameter could lead to the situation where designs with a larger maximin distance have, incorrectly, a larger value for the surrogate measure. On the other hand, a disadvantage of using the maximin distance criterion is that several designs may result in the same objective value. We, however, have reduced this problem by implementing neighborhoods that use critical points and by accepting equally good designs. By using critical points, we also implicitly reduce the number of critical points, without the need to introduce a surrogate measure.

4.4 Computational results

Periodic and adapted periodic Latin hypercube designs have been constructed for up to seven dimensions and for up to 100 design points, using the different classes provided in Table 4.1. Using simulated annealing, approximate maximin Latin hypercube designs have also been obtained for dimensions 8 to 10. All computations have been performed on PCs with an 800-MHz Pentium III processor. Table 4.2 shows the total CPU-times needed to construct approximate maximin Latin hypercube designs, for up to 100 points, for each dimension.

Dimension	3	4	5	6	7	8	9	10
CPU-time (hrs) PD	145	61	267	108	232	–	–	–
CPU-time (hrs) SA	500	181	152	520	246	460	470	470

Table 4.2: Total CPU-times needed to construct approximate maximin Latin hypercube designs, up to 100 points, using periodic designs (PD) and simulated annealing (SA).

Although our heuristics consider only a subset of all possible Latin hypercube designs, it can be seen from the table that still a considerable amount of time is needed to find good Latin hypercube designs in higher dimensions and for a large number of points. Fortunately, these computation times are a one-time cost, i.e. once a good Latin hypercube design has been found, and its coordinates saved, the design can be used over and over again in various applications, without incurring the computational costs again.

Table 4.3 provides the squared ℓ^2-maximin distances that were obtained by applying both heuristics. From this table it can be seen that (adapted) periodic designs work

	$k=3$		$k=4$		$k=5$		$k=6$		$k=7$		$k=8$	$k=9$	$k=10$
n	PD	SA	PD	SA	PD	SA	PD	SA	PD	SA	SA	SA	SA
2	**3**	**3**	**4**	**4**	**5**	**5**	**6**	**6**	**7**	**7**	**8**	**9**	**10**
3	3	**6**	4	**7**	5	**8**	6	**12**	7	**13**	**14**	**18**	**19**
4	**6**	**6**	**12**	**12**	11	**14**	15	**20**	16	**21**	**26**	**28**	**33**
5	6	**11**	12	**15**	11	**24**	15	**27**	16	**32**	**40**	**43**	**50**
6	**14**	**14**	16	**22**	23	**32**	28	**40**	29	**47**	**54**	**61**	**68**
7	14	**17**	16	**28**	23	**40**	28	**52**	31	**61**	**70**	**80**	**89**
8	**21**	**21**	25	**42**	32	**50**	42	**66**	46	**79**	**91**	**101**	**114**
9	21	**22**	25	**42**	39	**61**	45	**76**	47	**93**	**112**	**126**	**141**
10	21	**27**	36	**50**	55	**82**	62	**91**	68	**110**	**130**	**154**	**172**
11	24	**30**	39	**55**	55	**80**	62	**108**	69	**128**	**152**	**178**	**206**
12	30	**36**	46	**63**	62	**91**	91	**136**	95	**150**	**176**	**204**	**235**
13	35	**41**	51	**68**	64	**101**	91	**136**	95	**174**	**202**	**232**	**267**
14	35	**42**	70	**75**	86	**112**	104	**152**	119	**204**	**228**	**265**	**298**
15	42	**48**	71	**83**	88	**124**	111	**167**	129	**211**	**257**	**296**	**337**
16	42	**50**	85	**90**	101	**136**	130	**186**	155	**238**	**286**	**330**	**378**
17	42	**53**	85	**97**	113	**150**	131	**203**	161	**256**	**312**	**367**	**415**
18	50	**56**	94	**103**	123	**162**	155	**223**	186	**281**	**344**	**398**	**458**
19	57	**59**	94	**113**	136	**174**	169	**241**	195	**305**	**370**	**438**	**498**
20	57	**62**	106	**123**	139	**184**	210	**260**	226	**332**	**403**	**472**	**542**
21	65	**66**	116	**127**	165	**201**	210	**283**	236	**361**	**438**	**517**	**592**
22	**69**	**69**	117	**137**	174	**215**	223	**304**	270	**384**	**467**	**555**	**643**
23	72	**74**	130	**146**	178	**224**	236	**324**	273	**410**	**501**	**596**	**685**
24	76	**78**	138	**154**	201	**242**	258	**343**	308	**444**	**538**	**639**	**739**
25	**91**	81	156	**162**	205	**255**	286	**368**	350	**467**	**583**	**688**	**792**
26	**91**	86	156	**171**	226	**269**	296	**387**	365	**499**	**612**	**726**	**854**
27	**91**	90	157	**178**	238	**287**	310	**410**	382	**526**	**648**	**780**	**896**
28	**94**	**94**	174	**188**	258	**302**	339	**427**	406	**561**	**693**	**826**	**953**
29	94	**98**	174	**196**	269	**322**	346	**452**	417	**593**	**733**	**876**	**1015**
30	**105**	102	194	**209**	310	**335**	390	**473**	458	**620**	**787**	**925**	**1086**
31	**107**	106	212	**215**	310	**347**	390	**504**	482	**657**	**812**	**976**	**1138**
32	**114**	110	212	**228**	341	**371**	419	**529**	518	**695**	**866**	**1026**	**1194**
33	**114**	113	215	**234**	341	**379**	430	**548**	537	**723**	**900**	**1084**	**1253**
34	**133**	117	230	**244**	358	**403**	470	**586**	561	**751**	**945**	**1135**	**1329**
35	**133**	122	234	**255**	366	**418**	495	**601**	586	**811**	**1002**	**1190**	**1398**
36	**133**	129	250	**261**	400	**427**	518	**631**	636	**831**	**1042**	**1257**	**1459**
37	**152**	131	266	**275**	408	**454**	528	**648**	668	**863**	**1079**	**1300**	**1516**
38	**152**	134	**283**	279	415	**464**	561	**681**	709	**923**	**1127**	**1367**	**1597**
39	**152**	139	283	**290**	439	**486**	561	**706**	726	**938**	**1192**	**1434**	**1665**
40	**155**	146	291	**301**	492	**505**	632	**739**	786	**970**	**1224**	**1489**	**1742**
41	**162**	147	293	**309**	492	**525**	632	**776**	802	**1016**	**1271**	**1562**	**1820**
42	**168**	152	319	**325**	496	**543**	670	**791**	903	**1064**	**1333**	**1639**	**1920**
43	**168**	157	323	**329**	520	**558**	670	**830**	903	**1112**	**1377**	**1683**	**1973**
44	**186**	161	331	**349**	548	**582**	696	**862**	903	**1140**	**1463**	**1752**	**2072**
45	**186**	166	347	**362**	565	**615**	737	**891**	926	**1192**	**1480**	**1820**	**2130**
46	**189**	169	366	**370**	592	**615**	797	**918**	985	**1243**	**1548**	**1906**	**2208**
47	**189**	173	**378**	**378**	611	**634**	797	**940**	985	**1268**	**1616**	**1958**	**2331**
48	**189**	178	**413**	385	632	**673**	857	**976**	1054	**1325**	**1658**	**2017**	**2387**
49	**196**	180	**415**	399	634	**680**	893	**1015**	1074	**1356**	**1729**	**2103**	**2470**
50	**213**	185	**415**	414	663	**699**	893	**1042**	1113	**1397**	**1772**	**2179**	**2556**
51	**213**	189	421	**426**	692	**727**	917	**1067**	1161	**1450**	**1855**	**2243**	**2639**
52	**213**	198	**455**	429	709	**742**	1003	**1100**	1231	**1486**	**1888**	**2325**	**2745**
53	**216**	200	**455**	447	716	**765**	1003	**1136**	1241	**1537**	**1949**	**2429**	**2825**
54	**233**	213	**477**	454	760	**783**	1019	**1171**	1288	**1577**	**2006**	**2473**	**2892**
55	**243**	214	**483**	477	760	**805**	1082	**1198**	1325	**1639**	**2084**	**2570**	**3054**
56	**243**	216	**515**	479	784	**830**	1104	**1236**	1358	**1701**	**2162**	**2623**	**3100**
57	**261**	221	**515**	490	846	**854**	1136	**1265**	1479	**1721**	**2194**	**2704**	**3215**
58	**261**	227	**539**	500	846	**878**	1166	**1303**	1479	**1795**	**2258**	**2796**	**3305**
59	**266**	229	**544**	519	849	**905**	1223	**1328**	1509	**1821**	**2356**	**2881**	**3399**
60	**273**	237	**568**	530	904	**928**	1242	**1381**	1577	**1899**	**2393**	**2939**	**3500**
61	**274**	244	**620**	538	904	**939**	1258	**1413**	1615	**1928**	**2488**	**3021**	**3588**
62	**283**	245	**620**	554	934	**991**	1306	**1450**	1680	**2023**	**2541**	**3132**	**3700**

Table 4.3: (Maximin) squared ℓ^2-distance found using periodic designs (PD) and simulated annealing (SA) (*table continues on next page*).

	$k=3$		$k=4$		$k=5$		$k=6$		$k=7$		$k=8$	$k=9$	$k=10$
n	PD	SA	PD	SA	PD	SA	PD	SA	PD	SA	SA	SA	SA
63	**297**	249	**620**	575	967	**989**	1380	**1497**	1680	**2035**	**2607**	**3215**	**3767**
64	**297**	258	**625**	579	985	**1009**	1430	**1526**	1769	**2093**	**2734**	**3292**	**3955**
65	**314**	260	**630**	582	997	**1035**	1430	**1565**	1786	**2132**	**2723**	**3357**	**4034**
66	**314**	269	**666**	602	1050	**1051**	1476	**1590**	1857	**2180**	**2841**	**3474**	**4143**
67	**314**	270	**666**	614	1072	**1085**	1482	**1646**	1868	**2238**	**2868**	**3543**	**4224**
68	**314**	278	**685**	623	1087	**1119**	1538	**1664**	1940	**2295**	**2956**	**3647**	**4360**
69	**324**	280	**698**	650	1112	**1114**	1588	**1704**	1965	**2351**	**3075**	**3716**	**4455**
70	**325**	285	**716**	658	**1150**	1135	1633	**1759**	2130	**2417**	**3130**	**3841**	**4539**
71	**325**	289	**716**	665	1150	**1187**	1644	**1783**	2130	**2451**	**3161**	**3936**	**4689**
72	**341**	296	**750**	678	**1203**	1197	1768	**1862**	2177	**2503**	**3220**	**4027**	**4812**
73	**350**	299	**759**	688	1229	**1242**	1768	**1872**	2206	**2598**	**3305**	**4134**	**4873**
74	**350**	306	**767**	703	1229	**1269**	1774	**1910**	2244	**2614**	**3432**	**4224**	**5038**
75	**350**	310	**771**	714	1274	**1282**	1862	**1963**	2295	**2703**	**3513**	**4298**	**5171**
76	**363**	324	**813**	750	1300	**1318**	1935	**2024**	2375	**2756**	**3559**	**4395**	**5254**
77	**363**	325	**823**	762	1308	**1331**	1947	**2051**	2403	**2819**	**3617**	**4492**	**5399**
78	**387**	337	**844**	761	**1382**	1360	2014	**2079**	2505	**2870**	**3684**	**4577**	**5489**
79	**387**	333	**848**	788	1382	**1399**	2037	**2120**	2525	**2950**	**3775**	**4705**	**5633**
80	**403**	344	**873**	786	1395	**1430**	2037	**2152**	2590	**2979**	**3877**	**4807**	**5773**
81	**406**	338	**916**	782	1406	**1431**	2064	**2217**	2642	**3086**	**4001**	**4888**	**5901**
82	**406**	353	**938**	825	1475	**1482**	2141	**2239**	2753	**3118**	**3998**	**5030**	**6013**
83	**417**	369	**940**	829	1501	**1509**	2141	**2290**	2767	**3195**	**4076**	**5102**	**6097**
84	**426**	363	**967**	838	**1534**	1510	2229	**2325**	2838	**3227**	**4183**	**5222**	**6273**
85	**426**	369	**967**	877	1552	**1566**	2232	**2399**	2874	**3299**	**4324**	**5340**	**6397**
86	**428**	376	**967**	867	1573	**1578**	2375	**2437**	3103	**3335**	**4397**	**5423**	**6491**
87	**428**	374	**976**	877	**1598**	1589	2375	**2476**	3103	**3450**	**4474**	**5538**	**6622**
88	**437**	374	**1050**	890	**1685**	1629	2398	**2513**	3183	**3500**	**4524**	**5667**	**6803**
89	**443**	378	**1050**	907	**1690**	1654	2400	**2562**	3183	**3541**	**4578**	**5774**	**6872**
90	**481**	384	**1060**	940	**1710**	1696	2516	**2633**	3190	**3661**	**4699**	**5832**	**7040**
91	**481**	393	**1089**	951	**1748**	1724	2516	**2674**	3234	**3677**	**4850**	**5969**	**7163**
92	**481**	394	**1089**	966	**1805**	1750	2599	**2729**	3277	**3760**	**4873**	**6081**	**7286**
93	**481**	402	**1098**	962	**1813**	1795	2604	**2726**	3361	**3811**	**4984**	**6231**	**7488**
94	**481**	405	**1124**	986	**1881**	1811	2747	**2788**	3474	**3888**	**5067**	**6329**	**7536**
95	**481**	413	**1135**	1010	**1901**	1846	2747	**2817**	3531	**3940**	**5154**	**6396**	**7741**
96	**509**	414	**1261**	1023	**1965**	1863	2769	**2911**	3639	**4070**	**5220**	**6516**	**7777**
97	**515**	419	**1261**	1027	**1965**	1899	2817	**2960**	3639	**4069**	**5316**	**6649**	**8038**
98	**531**	429	**1261**	1055	**1965**	1929	2850	**3001**	3690	**4147**	**5445**	**6776**	**8242**
99	**531**	449	**1261**	1040	**2009**	1950	2878	**3043**	3731	**4214**	**5477**	**6912**	**8344**
100	**554**	451	**1261**	1074	**2053**	1975	3000	**3117**	3903	**4335**	**5597**	**6983**	**8450**

Table 4.3: (Maximin) squared ℓ^2-distance found using periodic designs (PD) and simulated annealing (SA) (*continued*).

particularly well for larger values of n. For dimensions 3 to 5 a *break-even point*, i.e. a point (or, better, an interval) where the preference shifts from the Latin hypercube designs found by simulated annealing to (adapted) periodic designs, is clearly visible in the table. 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 Latin hypercube designs, with $k \geq 6$, will occur for larger values of n, i.e. $n > 100$. This behavior could be explained by the *border effect*, i.e. the irregularity of Latin hypercube designs that is caused by the borders of the design space. Clearly, the number of "borders" of a 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 construction method works well only when the number of design points is relatively large, when compared to the dimension. Section 3.4 already shows the presence of this particular behavior in two-dimensional maximin Latin hypercube designs, i.e. the optimal designs found can all be represented by periodic designs. The results in Table 4.3 suggest that this behavior also occurs in higher dimensions.

Simulated annealing, however, does not depend on an underlying structure of the design and can therefore often find better Latin hypercube designs, especially for smaller values of n. Since all six- and seven-dimensional (adapted) periodic designs, of 3 to 100 points, are dominated by the designs found by simulated annealing, maximin distances of the former are computed only for up to seven dimensions. Concerning the different neighborhoods for the simulated annealing algorithm (see Section 4.3), it turned out that the second neighborhood yields, in general, the best results. For the terminating conditions, the first two conditions, generally speaking, result in the best Latin hypercube designs for $n \leq 50$, whereas the third and fourth condition are better for larger values of n.

Our heuristics are able to generate all best-known maximin Latin hypercube designs (see Morris and Mitchell (1995)), except for the cases $k = 6,\ n = 12$ and $k = 7,\ n = 14$, for which slightly worse designs are obtained. For the case $k = 3,\ n = 11$, however, we obtained an improved (and optimal) maximin Latin hypercube design. Furthermore, using a branch-and-bound algorithm, the three-dimensional designs of up to 13 points have been verified to be optimal.

Chapter 5

Quasi non-collapsing designs

I knew it! I knew it!
Well, not "knew it" in the sense of having the slightest idea, but I knew there was something I didn't know.

(BtVS, *Episode 02.14*)

5.1 Introduction

The problem of finding a two-dimensional unrestricted ℓ^2-maximin design (in a box-constrained domain) is equivalent to the problem of finding a packing of n circles with maximal common radius in a square; see Section 2.2.3. The Packomania website of Specht (2005) contains many good approximating solutions to this latter circle packing problem for up to 300 circles. Figure 3.9 depicts the best known maximal distances for the unrestricted case, i.e. the circle packing problem, as well as the case where the Latin hypercube structure is enforced. That figure shows that the loss incurred by adding the latter restriction is relatively small. The same seems to hold for three-dimensional maximin designs. The website of Pfoertner (2005) contains, among others, the best known packings of n congruent spheres into a cube for up to 72 spheres. Figure 5.1 depicts the corresponding best known maximal distances, as well as the best known maximal distances in case the Latin hypercube structure is enforced; see Table 4.3 (take the square roots of the distances provided in this table). Although the loss in space-fillingness is larger than in the two-dimensional case, the relative difference still decreases. Furthermore, as the border effect, caused by imposing the Latin hypercube structure, is less pronounced when n increases, it is to be expected that the relative difference tends to zero for larger values of n.

Remember that a Latin hypercube design of n points is obtained by requiring all the coordinates of the set of design points to be equidistantly distributed over the interval

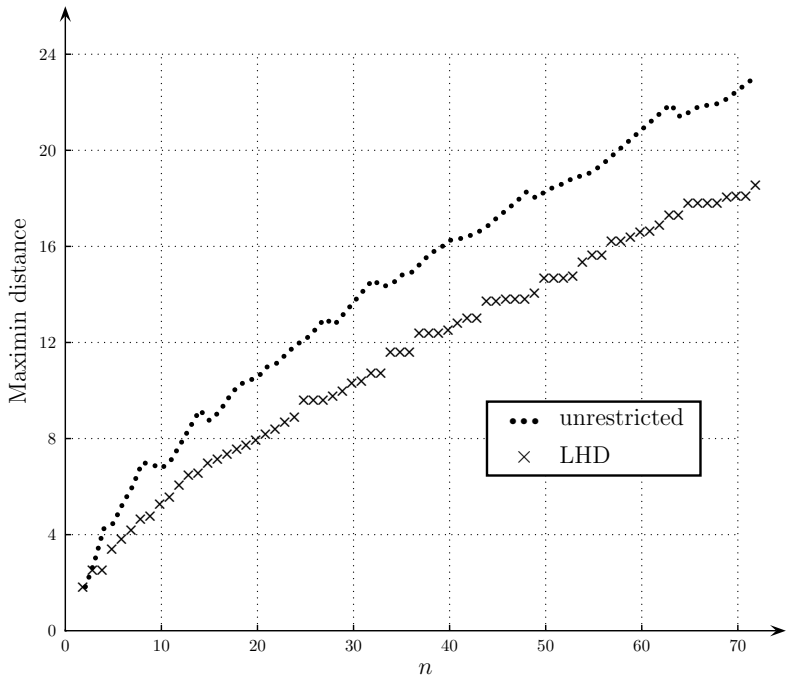


Figure 5.1: (Maximin) ℓ^2-distances d for three-dimensional unrestricted designs and Latin hypercube designs.

$[0, n-1]$. Now, instead, let the coordinates be required to be separated by at least some distance $\alpha \in [0,1]$. Note that $\alpha = 0$ results in an unrestricted (possibly collapsing) design, whereas $\alpha = 1$ yields a (non-collapsing) Latin hypercube design. We will call designs that depend on such a given $\alpha \in [0,1]$ *quasi non-collapsing*. It is interesting to investigate how the maximin distance is affected by the choice of α. In the rest of this chapter we consider two-dimensional, quasi non-collapsing, maximin designs. In this case, for a given value of $\alpha \in [0,1]$, the corresponding maximin distance is obtained by solving the following optimization problem:

$$\begin{array}{lll} \max \min\limits_{i \neq j} & d(x_i, x_j) & \\ \text{s.t.} & \alpha \leq |x_{i1} - x_{j1}| & i,j = 0,\ldots,n-1;\ i \neq j \\ & \alpha \leq |x_{i2} - x_{j2}| & i,j = 0,\ldots,n-1;\ i \neq j \\ & 0 \leq x_{i1} \leq n-1 & i = 0,\ldots,n-1 \\ & 0 \leq x_{i2} \leq n-1 & i = 0,\ldots,n-1. \end{array} \tag{5.1}$$

Here, $d(\cdot,\cdot)$ is a certain distance measure. In Sections 5.2 to 5.4 the computation of the maximin distances (and the corresponding designs) is illustrated for the rectangular distance, the maximum norm, and the Euclidean distance measure, respectively, for several values of $\alpha \in [0,1]$. Although results are obtained only for small values of n, it is interesting to see the, often non-trivial, trade-off between space-fillingness and non-collapsingness.

5.2 Rectangular distance

For the rectangular distance measure ℓ^1 the objective function in (5.1) reduces to $|x_{i1} - x_{j1}| + |x_{i2} - x_{j2}|$, which results in a non-convex, non-linear program. Furthermore, (5.1) can be rewritten as the following mixed integer linear program:

$$
\begin{array}{lll}
\max & d & \\
\text{s.t.} & d \le x_{j1} - x_{i1} + z_{ij} & i,j = 0,\ldots,n-1;\ i<j \\
& \alpha \le x_{i+1,1} - x_{i1} & i = 0,\ldots,n-2 \\
& \alpha \le z_{ij} & i,j = 0,\ldots,n-1;\ i<j \\
& z_{ij} \le x_{i2} - x_{j2} + 2(n-1)(1-h_{ij}) & i,j = 0,\ldots,n-1;\ i<j \\
& z_{ij} \le x_{j2} - x_{i2} + 2(n-1)h_{ij} & i,j = 0,\ldots,n-1;\ i<j \\
& 0 \le x_{i1} \le n-1 & i = 0,\ldots,n-1 \\
& 0 \le x_{i2} \le n-1 & i = 0,\ldots,n-1 \\
& 0 \le z_{ij} \le n-1 & i,j = 0,\ldots,n-1;\ i<j \\
& h_{ij} \in \{0,1\} & i,j = 0,\ldots,n-1;\ i<j.
\end{array}
\tag{5.2}
$$

Here, $h_{ij} = 1$ if $x_{i2} \ge x_{j2}$, and $h_{ij} = 0$ otherwise, resulting in $z_{ij} \le |x_{i2} - x_{j2}|$. Since d is maximized, this yields $z_{ij} = |x_{i2} - x_{j2}|$ in the optimal solution. Solving (5.2) for several given values of α results in the maximin distance d as a function of the quasi non-collapsingness parameter $\alpha \in [0,1]$. Figure 5.2 shows two examples of such a function for maximin designs of 10 and 11 points, respectively. The graphs are a result of solving (5.2), using the *XA Binary and Mixed Integer Solver* of Sunset Software Technology (2003), for 200 equidistantly distributed values of $\alpha \in [0,1]$. Unfortunately, solving the mixed integer linear program for a given α may take a lot of computation time, e.g. solving (5.2) for $n = 11$ takes (on average) 20 minutes. Therefore, (5.2) has been solved for small values of n only.

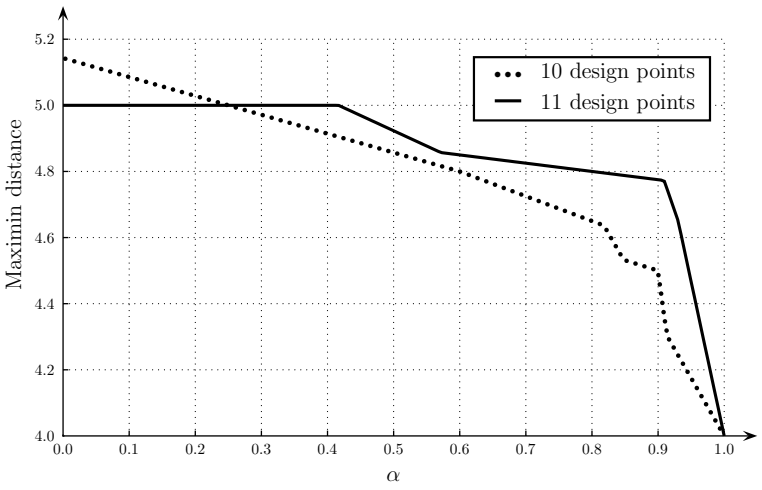
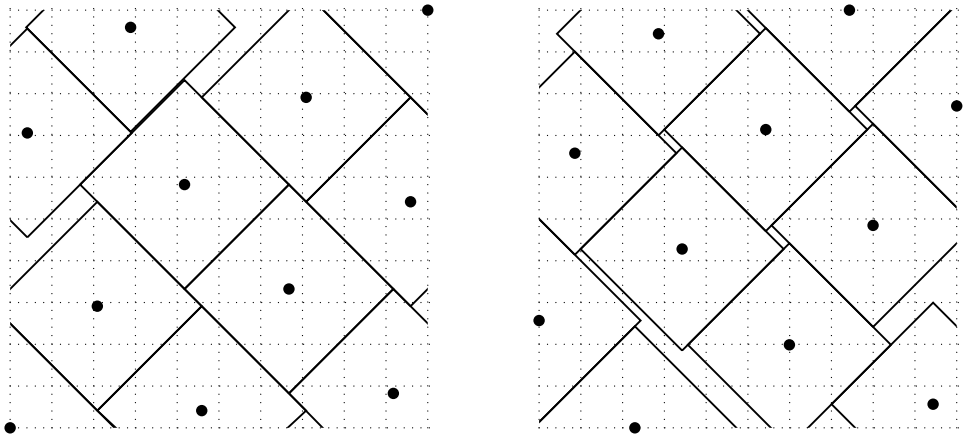
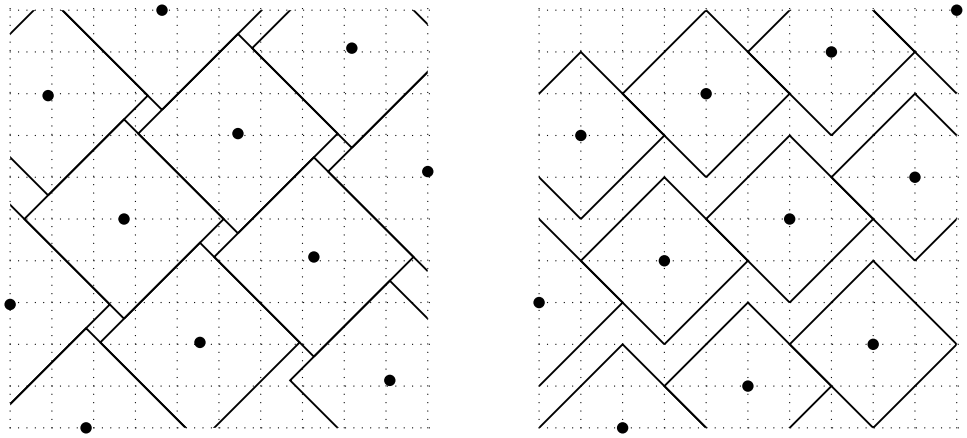


Figure 5.2: Maximin ℓ^1-distance as function of the quasi non-collapsingness parameter $\alpha \in [0,1]$, for 10 and 11 design points.

Both graphs in Figure 5.2 indicate non-concave, non-increasing, piecewise-linear functions. This behavior can be explained as follows. Fixing all h_{ij} in (5.2) results in a linear program with continuous variables only, and α in the right-hand side of the constraints. From the sensitivity analysis of a linear program it is known that the optimal value as a function of α is a non-increasing, concave, piecewise-linear function, see Roos et al. (1997). For every realization of the binary variables such a function is obtained. The maximal d is found by taking the maximum over all these functions, resulting in a non-increasing, piecewise-linear function that is not necessarily concave.



$\alpha = 0.41$, $d = 5.00$ $\alpha = 0.57$, $d = 4.86$



$\alpha = 0.91$, $d = 4.77$ $\alpha = 1.00$, $d = 4.00$

Figure 5.3: Quasi non-collapsing maximin ℓ^1-distance designs of 11 points for $\alpha = 0.41$, $\alpha = 0.57$, $\alpha = 0.91$, and $\alpha = 1.00$.

An interesting observation can be made from the plotted function for 11 design points. It is seen that α can be increased to a value of 0.41 without affecting the unrestricted maximin distance. Furthermore, for α between 0.41 and 0.91 the maximin distance stays within 5% of its unrestricted value; dropping sharply only for values larger than 0.91.

Apparently, it is possible to construct a highly non-collapsing design of 11 points, without decreasing the unrestricted maximin distance much. For example, Figure 5.3 depicts four quasi non-collapsing maximin designs, corresponding to the points of inflection in Figure 5.2: $\alpha = 0.41$, $\alpha = 0.57$, $\alpha = 0.91$, and $\alpha = 1.00$.

5.3 Maximum norm

When considering the maximum norm ℓ^∞ the objective function in (5.1) reduces to $\max\{|x_{i1}-x_{j1}|, |x_{i2}-x_{j2}|\}$. Following the same kind of reasoning as with the ℓ^1-distance measure, the optimization problem can be rewritten as a mixed integer linear program. Unfortunately, extra binary variables have to be included to deal with the maximum-operator in the objective function, which may increase the computation time:

$$
\begin{array}{lll}
\max & d & \\
\text{s.t.} & d \leq x_{j1} - x_{i1} + (n-1)(1-k_{ij}) & i,j = 0,\ldots,n-1;\ i<j \\
& d \leq z_{ij} + (n-1)k_{ij} & i,j = 0,\ldots,n-1;\ i<j \\
& \alpha \leq x_{i+1,1} - x_{i1} & i = 0,\ldots,n-2 \\
& \alpha \leq z_{ij} & i,j = 0,\ldots,n-1;\ i<j \\
& z_{ij} \leq x_{i2} - x_{j2} + 2(n-1)(1-h_{ij}) & i,j = 0,\ldots,n-1;\ i<j \\
& z_{ij} \leq x_{j2} - x_{i2} + 2(n-1)h_{ij} & i,j = 0,\ldots,n-1;\ i<j \\
& 0 \leq x_{i1} \leq n-1 & i = 0,\ldots,n-1 \\
& 0 \leq x_{i2} \leq n-1 & i = 0,\ldots,n-1 \\
& 0 \leq z_{ij} \leq n-1 & i,j = 0,\ldots,n-1;\ i<j \\
& h_{ij} \in \{0,1\} & i,j = 0,\ldots,n-1;\ i<j \\
& k_{ij} \in \{0,1\} & i,j = 0,\ldots,n-1;\ i<j.
\end{array}
\tag{5.3}
$$

The binary variables h_{ij} serve the same purpose as in (5.2); for the extra binary variables k_{ij} it holds that $k_{ij} = 1$ if $|x_{i1} - x_{j1}| \geq |x_{i2} - x_{j2}|$, and $k_{ij} = 0$ otherwise, resulting in $d \leq \max\{|x_{i1} - x_{j1}|, |x_{i2} - x_{j2}|\}$. Like in the case of the ℓ^1-distance measure, this mixed integer linear program can be used to compute the maximin distance d for several values of $\alpha \in [0,1]$. Figure 5.4 depicts two examples, for maximin designs of 6 and 7 points, respectively. The graphs are a result of solving (5.3) for 200 uniformly spaced values of $\alpha \in [0,1]$. Again, solving the mixed integer linear program is computationally demanding, and, hence, only small values of n have been considered. Furthermore, it can be argued that the maximin distance is a non-increasing, piecewise-linear function of α. Note that this function appears to be linear in the case of 6 design points. For 7 design points, highly non-collapsing maximin designs can be constructed without decreasing the maximin distance more than 15%, by taking $\alpha \leq 0.85$.

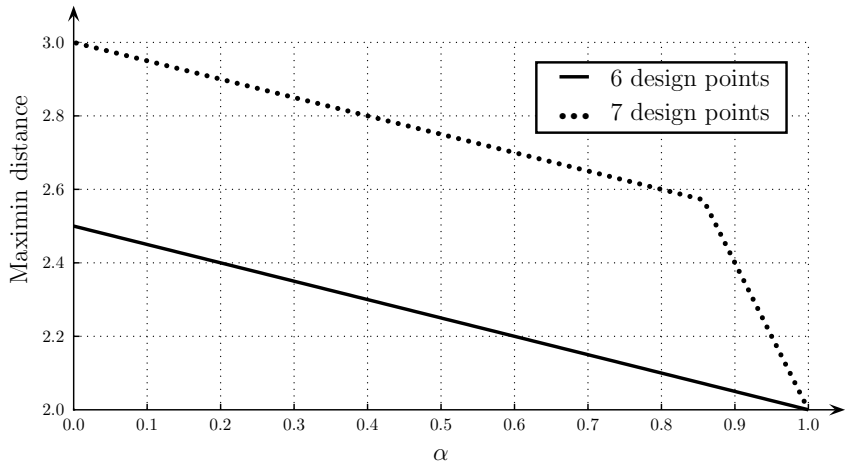


Figure 5.4: Maximin ℓ^∞-distance as function of the quasi non-collapsingness parameter $\alpha \in [0, 1]$, for 6 and 7 design points.

5.4 Euclidean distance

For the Euclidean distance measure ℓ^2 the situation is even more complicated than for the previously considered ℓ^1- and ℓ^∞-distance measures. In this case, the objective function in (5.1) reduces to the quadratic function $(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2$ (for the sake of convenience, we consider the squared Euclidean distance). The resulting non-linear program is in fact a multi-extremal optimization problem, which calls for a global optimizer. We have used the Lipschitz Global Optimizer (LGO) (see Pintér (1995)) to compute the maximin distance as a function of the quasi non-collapsingness parameter $\alpha \in [0, 1]$. Within LGO the multi-start global search option has been applied, followed by a local search phase, to increase the probability of obtaining a good solution.

Although the obtained distances yield only lower bounds for the (unknown) global maximin distances, some information about the behavior of the maximin distances can still be extracted. For example, Figure 5.5 depicts the maximal distances corresponding to (approximate) maximin designs of 5 and 6 points for several values of $\alpha \in [0, 1]$. To obtain this figure, (5.1), with objective function $(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2$, has been solved for 50 equidistantly distributed values of $\alpha \in [0, 1]$. Both plotted functions in Figure 5.5 indicate a non-trivial behavior. For 5 design points a small change in α heavily affects the maximal distance for values of α less than 0.53 and larger than 0.86, whereas this effect is less pronounced when α lies between 0.53 and 0.86. For (approximate) maximin designs of 6 points the maximal distance is heavily affected only by large values of α, i.e. $\alpha > 0.80$. This facilitates the construction of highly non-collapsing (approximate) maximin designs with a maximal distance that does not deviate too much from the unrestricted maximin distance.

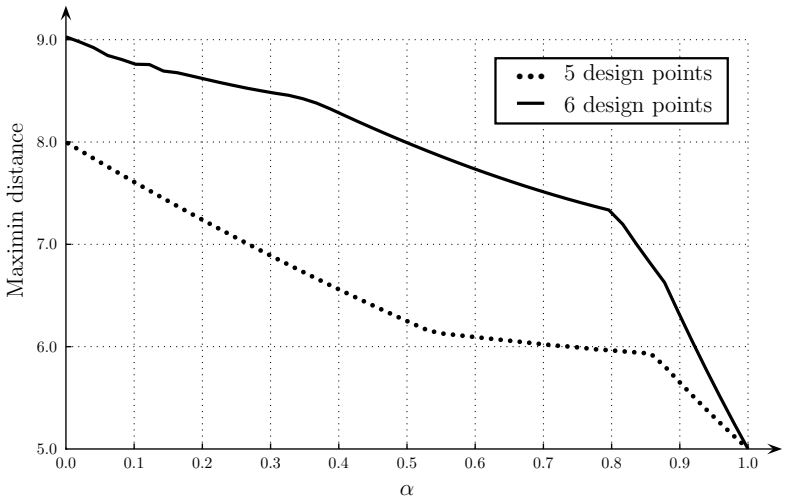


Figure 5.5: (Maximin) squared ℓ^2-distance as function of the quasi non-collapsingness parameter $\alpha \in [0, 1]$, for 5 and 6 design points.

Part II

Nested maximin designs

Chapter 6

Collaborative Metamodeling

> Your theory is crazy, but it's not crazy enough to be true.
>
> (Niels Bohr, *to a young physicist*)

6.1 Introduction

High-tech products, such as automobiles and aircrafts, consist of many components. Since these components are often complex, their design processes are distributed among specialized design teams of engineers. Each of those teams use their own simulation tools to evaluate the individual component designs.

For example, in multidisciplinary industries, such as aeronautics and astronautics, engineers have to deal with many disciplines in their design of the final product. Examples of disciplines in the aeronautic industry are aerodynamics, (wing) structures, and mission performance. Unfortunately, there are often several conflicting aspects among the disciplines. As an example, consider the design of an aircraft wing in aeronautics. The main aspects of a wing are its shape and its weight. Obviously, a heavier wing can cope with a higher pressure on the surface of the wing. However, the power needed to lift the wing at take-off increases with the weight. Hence, there is a trade-off between the strength of the wing and the power needed to lift the plane from the ground. Note, however, that this trade-off is limited to just one discipline, i.e. the structural (wing) optimization. A trade-off between two disciplines occurs when determining the shape of the wing. An aerodynamically shaped wing may reduce the amount of kerosine needed at take-off and during the flight; however, the fabrication of such a wing may require the use of special materials, thereby increasing the production costs. In this case, the trade-off is between the disciplines of aerodynamics and structures. Due to this latter type of conflicting aspects, engineers like to speak of multidisciplinary problems.

Similarly, in the case of multi-component products, relations among components correlate the design problems that the engineers face and create interdependencies among the different black-box functions. Designing the final product is therefore a very hard task that can be accomplished only by a proper coordination of all the individual component design processes. Up to now, mainly sequential optimization methods have been proposed in the literature to deal with this kind of problems. Such sequential techniques are based on optimization procedures that iteratively solve several small optimization problems in order to gradually converge to the optimal solution. Several of these methods are discussed in Section 6.2.

In practice, however, engineers are not just interested in the optimal solution, but they also like to gain insight into the behavior of the product, in order to design a reliable product. This may be accomplished by an efficient construction of a global approximation model for the product as a whole. Unfortunately, because of the large number of design (or input) and response (or output) parameters of the product, it is impossible to do this all-at-once, e.g. using the Metamodel approach; see Section 1.2. In this chapter we propose a collaborative extension to the former approach, which we call the *Collaborative Metamodel approach*, or simply *Collaborative Metamodeling*. This approach exploits the architecture of the product by first constructing metamodels for all black boxes individually and then combining these models into a metamodel for the product. To account for the relations among the black-box functions, coordination methods are introduced. Such coordination methods control the order of the evaluations and the construction of metamodels during the whole modeling process. The set of resulting approximation models then implicitly forms the required metamodel for the product.

6.2 Collaborative approaches

Due to the increased complexity of products and organizational structures, engineers have spent much time on developing new techniques that can help them with the design process. This section gives a short overview of some techniques that are found in literature and in practice.

6.2.1 Multidisciplinary Design Optimization

Multidisciplinary Design Optimization is a collective term for several solution techniques that are able to deal with multidisciplinary design problems. The basic idea of these solution techniques is to integrate different (coupled) design problems, i.e. the subproblems at the various disciplines, into one large design problem. This large problem is then solved iteratively until the solution has converged.

Solution techniques that are often applied to large-scale design problems are the *All-at-once*, *Individual Discipline Feasible*, and *Multidisciplinary Feasible* method. The main difference between these methods is the requirement, or the lack of it, to have a feasible design at each iteration. For the All-at-once method only the final design has to be feasible, whereas the Individual Discipline Feasible method requires every discipline to yield (locally) feasible designs at each iteration. Hence, an intermediate design for a particular discipline is still allowed to be incompatible with other disciplines. For the Multidisciplinary Feasible method, however, incompatibility is not allowed at any step, i.e. the product design must be feasible at each iteration.

Cramer et al. (1994) and Kodiyalam and Sobieszczanski-Sobieski (2001) discuss these, and other, methods in more detail, and Hulme (2000) uses some real-life test cases to determine how each of these methods perform. Sobieszczanski-Sobieski and Haftka (1997) give a comprehensive survey concerning the most recent developments in Multidisciplinary Design Optimization in aeronautics.

6.2.2 Collaborative Optimization

The Multidisciplinary Design Optimization techniques discussed above do not always suffice to solve coupled design problems, especially not when the number of disciplines is large. For cases where the design problem involves a large number of disciplines with many local design parameters, i.e. parameters that affect only a certain discipline, engineers have developed a technique called *Collaborative Optimization*; see Braun and Kroo (1995). The idea of this technique is not to *integrate* all subproblems into one large problem, but rather to *coordinate* these problems. To achieve this, a bi-level optimization architecture is used in which a coordination level controls the subproblems at a lower level.

Instead of satisfying all restrictions at once, the coordination level imposes local restrictions and targets on the various subproblems at the lower level. Each subproblem searches for the best solution that satisfies these requirements (as closely as possible), and the result is returned to the coordination level. At this latter level the results of the different subproblems are compared with the imposed targets. Any adjustments needed on the local restrictions and targets are processed and then again sent back to the subproblem level. After several of these level-switching iterations have been executed, an optimal solution is obtained – if one exists.

From this description it follows that several optimizers may be used at different places in the optimization process. In order to obtain an optimal feasible solution to the (large) design problem, these optimizers have to collaborate. Just as with Multidisciplinary Design Optimization, Collaborative Optimization is a collective term for several related solution techniques. A mathematical discussion of some of these techniques is provided by Alexandrov and Lewis (2000) and DeMiguel and Murray (2000). Several applications of

Collaborative Optimization are discussed by Kroo and Manning (2000). Finally, Alexandrov and Lewis (1999) compare Collaborative Optimization with other Multidisciplinary Design Optimization methods.

6.2.3 Analytical Target Cascading

Since evaluations of scenarios in multidisciplinary problems are often very time-consuming, the time the solution process takes may easily grow out of bounds. Papalambros (2000) proposes to use mathematical functions that locally approximate the behavior of the more complex models and processes to reduce the order of complexity. Furthermore, Papalambros (2001) introduces *Analytical Target Cascading* as an approach to find an optimal product design in case there are dependencies among these local approximations.

Analytical Target Cascading is based on the assumption that the engineers, or some coordinator of the optimization process, set targets for the characteristics of the final product. Like Collaborative Optimization, this is done at a coordination level. The total problem is hierarchically decomposed into sublevels, subsublevels, et cetera. Among the subproblems at a certain level only weak dependencies, i.e. a small number of common design parameters, are allowed.

After initial targets have been set at the coordination level, these targets are sent down to the underlying sublevels. At each of these sublevels an optimization problem is solved, resulting in new targets, which are in turn sent down to their underlying sublevels, i.e. the subsublevels. At each subsequent level the targets are adjusted, until the lowest level has been reached. At this point, the direction of the information flow reverses and the targets are sent back up through the levels, all the way up to the coordination level. This procedure is repeated until all targets have converged.

6.3 Collaborative Metamodel approach

Section 1.2 discusses the Metamodel approach for black-box optimization problems. In this case, the unknown black-box function is replaced by approximation models, based on evaluations of some scenarios. These metamodels can then be used to gain insight into, or optimize over, the product design space. For the Metamodel approach the design process is divided into four basic steps: problem specification, design of computer experiments, metamodeling, and design analysis and optimization.

This section introduces Collaborative Metamodeling to deal with optimization problems when there are several interdependent black boxes to consider. The steps to be taken are the same as in the Metamodel approach, be it that some extra work has to be performed at each step, in order to deal with the relations among the black boxes. Next,

the four steps in the collaborative variant are discussed, as well as the problems that are encountered when applying this procedure to multi-component product design problems.

6.3.1 Step 1: Problem specification

In the first step the architecture of the product is investigated and all components (or black boxes), and the relationships among them, are identified. Furthermore, for each black box its design and response parameters are defined, as well as the expected simulation time per evaluated scenario and the expected number of evaluations needed. Design parameters can be divided into *local design parameters* and *linking design parameters*. Local design parameters are input to a single black box, whereas linking design parameters are input to multiple black boxes. Note that restrictions on (combinations of) linking design parameters increase the complexity of the product design problem since these restrictions cause dependencies among the different components. Moreover, such dependencies may also be caused by *response input parameters*, which are black-box response parameters that are input to other black boxes. An example of the interdependence of design and response parameters for two coupled black boxes is depicted in Figure 6.1.

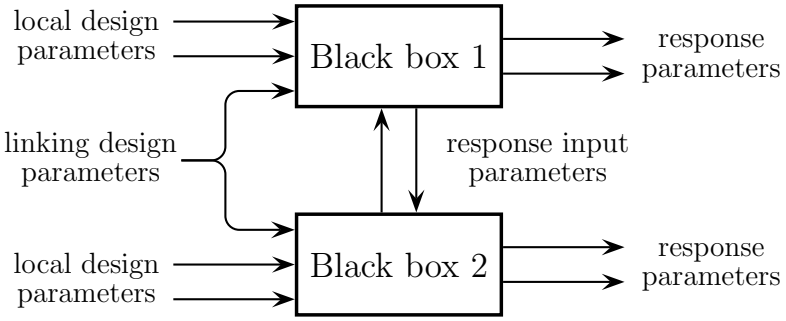


Figure 6.1: Design and response parameters for two coupled black boxes.

From Figure 6.1 it can be seen that the coupling of black boxes is caused by linking design parameters and response input parameters. For linking design parameters the same settings should be chosen as much as possible, when constructing designs for computer experiments for the different black boxes. The reasons behind this restriction are discussed in Section 6.4.1; Chapters 7 and 8 introduce methods to construct such designs.

The presence of response input parameters gives rise to the need for a coordination method. This latter type of coupling can be represented by a directed graph in which the nodes represent the black boxes and the arcs represent the links between the black boxes. We assume that there exist no cycles in the directed graph, which is a common assumption in the literature, see e.g. Assine et al. (1999) and Tang et al. (2000).

Moreover, this assumption is substantiated by design problems found in practice. The absence of cycles results in a directed graph with a forward structure, i.e. there exists an explicit precedence ordering of the nodes, and, thus, the black boxes. Therefore, we refer to such a directed graph as a *black-box chain*. See Figure 6.2 for an example of such a chain.

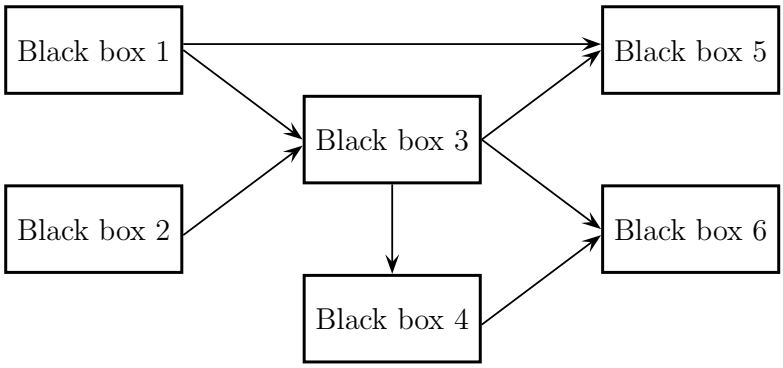


Figure 6.2: A black-box chain.

In Figure 6.2 an arc represents one or more response input parameters. Note that there may be multiple independent black-box chains within the product design problem. All these chains can be dealt with concurrently in the way described in the current chapter.

To reduce the total simulation time needed, several black boxes may be clustered. Such a cluster of black boxes can then be considered as one (big) black box, thereby decreasing the number of design parameters. Since the expected number of evaluations needed often depends on the number of design parameters, clustering will lead to a reduction of the number of evaluations, and, hence, the total simulation time. Kusiak and Park (1990) discuss a methodology for clustering, based on the grouping and decomposition of design activities, in order to reduce the design project make-span. Note, however, that clustering may have unwanted effects on the construction of metamodels, such as less accuracy, which may not weigh up against the reduction in simulation time. The coordination methods that are introduced in Section 6.4, however, do reduce the total time needed for simulation, while retaining the structure of the black-box chain.

6.3.2 Step 2: Design of computer experiments

With the design spaces for each component determined, or estimated, each design team has to construct a design of computer experiments. Depending on the applied coordination method these designs can either be constructed individually, or have to take into

account previously obtained evaluations of the own black-box function or from other black boxes. This latter dependency is caused by response input parameters. Again, the designs should be space-filling and non-collapsing to insure that good global approximation models are obtained for all black-box functions.

Depending on the coordination method, it may be possible to carry out *system-level simulations*, i.e. the evaluation of particular product designs (instead of just component designs). Note, however, that this is possible only when the same settings are chosen for all linking design parameters and the observed response input parameter values are used as input to succeeding black-box functions. Section 6.4.1 explains how system-level simulations can help in the design process of the product. Furthermore, Chapters 7 and 8 introduce methods to construct designs for computer experiments that can be used to obtain system-level simulations. Such designs are referred to as *nested designs*.

6.3.3 Step 3: Metamodeling

After all scenarios at the individual black boxes have been evaluated, metamodels can be fitted to the obtained data. The relations among the individual components can then be used to combine these models into metamodels for the product as a whole. Combining these approximation models, however, may lead to serious error propagation, which may result in poor metamodels for the product. Therefore, the metamodels of the product should be validated, e.g. using cross-validation; see Stehouwer and Den Hertog (1999). Should the system-level metamodels appear to be invalid, then either different models should be fitted or an additional set of scenarios has to be evaluated to improve the current models. The problem to determine which components to grant extra evaluations, and which design points to evaluate in this case, is very complex and further research is therefore needed in this area. One option may be the use of nested designs, see Chapters 7 and 8.

6.3.4 Step 4: Design analysis and optimization

The obtained metamodels of the product give a lot of information about the behavior of the product. Since the approximation models are explicit functions, function evaluations are relatively fast, and, hence, optimization techniques can be applied to find a good product design. Furthermore, the metamodels for the components of the product are also an interesting source of information. From these approximation models the significance of each black box, and all its design parameters, with respect to the characteristics of the final product, becomes more transparent. Hence, changes on the component level that will result in changes on the system level can be detected (and tested) more easily. Furthermore, with a multi-component product, and all corresponding interrelations on

the component level, robustness of the final product design becomes an important, and, unfortunately, very difficult aspect to consider.

6.3.5 Comparison with other approaches

Multidisciplinary Design Optimization, Collaborative Optimization, and Analytical Target Cascading, can all be considered as sequential optimization methods since they aim to establish a convergence of the (intermediate) designs, or targets, to some optimum. Collaborative Metamodeling, however, aims at constructing global metamodels for the underlying product, in order to not only find a good product design, but also to gain more insight into the behavior of the product as a whole.

Collaborative Optimization uses the same bi-level structure as Collaborative Metamodeling, i.e. a coordination (or system) level and a component level. The coordination process in Collaborative Optimization, however, aims at achieving the targets set at the system level, whereas in Collaborative Metamodeling the coordination process intends to obtain metamodels for all components that properly reflect the dependencies that are present.

Like Analytical Target Cascading, Collaborative Metamodeling exploits the architecture of the product by considering a decomposition of the product into its components and disciplines. Both approaches use approximation models to reduce model complexity, albeit that Analytical Target Cascading considers local approximations, whereas Collaborative Metamodeling uses global approximation models. Furthermore, Analytical Target Cascading takes the decomposition one step further by considering, besides the system and component level, also the part and process levels underlying the components.

The rest of this chapter focuses on the first step of Collaborative Metamodeling, i.e. the problem specification, and on coordination methods in particular.

6.4 Coordination methods

The coupling of black boxes has a direct effect on the design of computer experiments. Due to the presence of response input parameters, the constructions of designs for computer experiments for different black-box functions are interrelated. To steer the simulation process in the right direction, coordination methods are introduced in this section. We define a coordination method to be a rule that determines the order in which simulations are carried out and metamodels are constructed by the different component design teams. Next, we introduce and analyze the following three coordination methods.

Parallel Simulation

The first method does not take into account the interdependencies among the black boxes. Every black box is dealt with separately, i.e. independent of all others. Linking and response input parameters are considered as local design parameters and a design of computer experiments is based on local parameter restrictions only. Every design team carries out their simulations concurrently.

Sequential Simulation

The second method uses the response values obtained at black boxes preceding the one in question. That is, once a component design has been evaluated the obtained responses are transferred to all its successors (if any). When a particular black box has received the evaluation results for a specific scenario from all its predecessors, a simulation run is carried out, i.e. one scenario is evaluated. This procedure is repeated until the number of required evaluations has been reached. It is important to note that the simulation runs at every design team are carried out *one-by-one*, following the precedence ordering in the black-box chain and using the responses observed at predecessors.

Sequential Modeling

The third method closely resembles Sequential Simulation. The main difference is that the simulation runs for a particular black box are carried out *all-at-once* and that the observed response values, along with the constructed metamodels, are transferred to all its successors. Again, the precedence ordering in the black-box chain is followed, but now the design teams have to wait until their predecessors have completely finished their simulation and metamodeling processes. Note, however, that this procedure provides the engineers with maximal information about their predecessors.

6.4.1 Aspects of coordination methods

Once a design of computer experiments has been evaluated for a particular black box, metamodels can be constructed based on the observed responses. The metamodels for all black-box functions together are then used in the product design optimization process. The validity of these metamodels, however, depends heavily on the availability of proper data. This, in turn, depends on the way the evaluations have been carried out, and, even more important, it depends on which scenarios have been evaluated. Since coordination methods play a role in this process, the current section discusses and compares the three coordination methods in more detail. As a measure for comparison, the following five aspects are considered:

- Use of precedent information;

- Coordination complexity;
- System-level simulations;
- Flexibility;
- Throughput time.

Next, all these five aspects are defined. Furthermore, the different effects that the three coordination methods have on each aspect are discussed and compared. In Section 6.4.2 the results found are summarized and recommendations on the choice of a coordination method are provided.

Use of precedent information

This aspect refers to the use of evaluation and modeling results from preceding black boxes. Such results are helpful in the determination of design parameter settings that are expected to yield the most valuable information about the components. Note that in case of response input parameters the use of precedent information is a necessity to obtain system-level simulations. Clearly, both sequential coordination methods use precedent information by means of response input parameter values. In Parallel Simulation no precedent information is used since all design parameters are considered as local design parameters.

Coordination complexity

Coordination complexity refers to the amount of communication and time that is needed to implement a certain coordination method. It also includes extra costs that are incurred by, for example, the need for an automated communication system. In Parallel Simulation every design team operates independently and there is no need for a complex organizational structure; see e.g. Krishnan (1996), where managing the simultaneous execution of two coupled development phases plays a central role. In Sequential Simulation, communication is needed after each global design simulation (see Section 6.5.2) at every black box. Therefore, this coordination method results in a complex coordination process that needs sophisticated communication methods, which have to be supported by the design tools. Communication between design teams is also required in Sequential Modeling, be it only after a complete design of computer experiments has been evaluated. Hence, the coordination process is relatively simple.

System-level simulations

A product design is a particular setting of all the design parameters in the product specification. Evaluating such a setting yields a system-level simulation. Obtaining

system-level simulations may require some effort, but it is a great help in the product design process. These simulations provide information about the characteristics of feasible product designs, which lead to more insight into the product, and may even yield an improved product design. Furthermore, system-level simulations increase the credibility of the applied optimization and robust design approaches, and can be used to validate the obtained metamodels of the product.

Since simulations are carried out by evaluating black-box functions at the component level, all these functions will have only a subset of the total set of the design parameters of the product as their input. Furthermore, the presence of linking design parameters and response input parameters creates overlap in these sets of design parameters and leads to coupled black-box functions. Hence, the same settings should be used for the linking design parameters at every black box, and observed response input parameter values should be used as input parameter settings in succeeding simulation tools, in order to obtain a consistent system-level simulation.

Nested designs can be used to deal with the linking design parameter settings (see Chapters 7 and 8); what remains is to account for the response input parameters. Clearly, it is not possible to obtain system-level simulations in Parallel Simulation, since all response input parameters are considered to act as local design parameters. The sequential coordination methods take responses from preceding black boxes as inputs, and, hence, may yield system-level simulations.

Flexibility

A coordination method is called flexible when it does not take a lot of effort to validate or adjust the constructed metamodels, in case of small changes to one or more black-box functions. In Parallel Simulation the metamodels for the various black boxes are constructed independently, so changes in a particular component do not directly affect other metamodels and can be offset by evaluating an additional set of design points for the corresponding black box. When applying a sequential coordination method one must be more careful, since coupling among black-box functions is preserved in the construction of metamodels. Therefore, invalidity of one metamodel may affect the validity of the metamodels for succeeding black-box functions. Since small changes can require much effort in the validation and, possibly, adjustment of many metamodels, the sequential coordination methods are not flexible with respect to changes in the black boxes, whereas Parallel Simulation is flexible.

Throughput time

The throughput time of a coordination method is defined as the total time it takes to carry out all evaluations needed to construct proper metamodels for every black box

in the chain. From a time-to-market perspective it is desirable to have short product development times, so the throughput time should preferably be small. We assume that construction times of metamodels are negligible, relative to the time-consuming computer simulations, and, hence, they are ignored in our analysis. Derivations of formulas for the throughput times for all three coordination methods are provided in Section 6.5. Furthermore, it is proven that Parallel Simulation always leads to the shortest throughput time, Sequential Simulation takes a longer time, and Sequential Modeling the longest. Using the throughput-time formulas it is easy to compute the exact throughput time for each coordination method for a particular problem instance. This information, along with the four other aspects above, can be used to decide on the coordination method to be used.

6.4.2 Comparison of coordination methods

To compare the three coordination methods, Table 6.1 assigns a performance score for each of the five aspects to each method. Two pluses (++) indicate that the coordination method has a very positive effect on a particular aspect; one plus (+) indicates a moderately positive effect. With one minus (−) the effect of the coordination method on a particular aspect is slightly negative; with two minuses (−−) this effect is very negative. Note that in Table 6.1 a positive effect, i.e. + or ++, on the coordination complexity implies that the coordination process is *not* complex.

Aspect	Parallel Simulation	Sequential Simulation	Sequential Modeling
Use of precedent information	−−	+	++
Coordination complexity	++	−−	+
System-level simulations	−	++	++
Flexibility	++	−−	−−
Throughput time	++	+	−−

Table 6.1: Comparison of the three coordination methods.

The main advantages of Parallel Simulation are a small throughput time, much flexibility, and a less complex coordination process. The obtained metamodels, however, may not approximate the characteristics of the product accurately enough since relations among components are not considered. Hence, an additional set of evaluations, besides the scenarios already evaluated, may be needed to include the coupling among black boxes properly in the metamodels. Furthermore, Section 6.4.1 illustrates the importance of system-level simulations, which Parallel Simulation unfortunately lacks. These drawbacks make a sequential coordination method more suitable than the parallel coordination

method, particularly in cases with response input parameters.

Choosing between the two sequential coordination methods mainly depends on the throughput time and the availability of good means of communication among the design teams. Sequential Simulation results in a more complex coordination process, whereas Sequential Modeling generates a longer throughput time. Therefore, when dealing with time-consuming simulations and an automated communication system, Sequential Simulation is preferable. Sequential Modeling is a good choice when communication among design teams is hard and the simulation times are not too lengthy.

Of course, the determination of the best coordination method is not so strict, and it depends on the kind of product design problem that is dealt with. This is why a careful study of all aspects for each of the three coordination methods is extremely important. In this respect, the discussions in Section 6.4.1, Table 6.1, and the formulas derived in Section 6.5, may be of help in the decision process.

Section 6.3.3 mentions the problem of constructing proper metamodels. Since the initial sets of evaluated scenarios may not suffice to construct proper metamodels for each black box, a two-stage simulation procedure is often applied. For the first stage we advise to use Parallel Simulation and to evaluate all designs concurrently. This gives a good idea about the general black-box behavior and the most important parts of the component and product design spaces. Furthermore, the small number of evaluations may turn out to be sufficient for constructing valid metamodels for (some of) the black-box functions. In this latter case, additional (time-consuming) evaluations are not needed. Otherwise, we advise that in the second stage the previous evaluation results are combined with a sequential coordination method, and that extra sets of component designs are simulated, which may lead to more accurate metamodels.

6.5 Computation of the throughput time

This section derives mathematical formulas for the throughput time, which has been introduced in Section 6.4.1. The following notation is used:

- B: set of black boxes, $B = \{1, 2, \ldots\}$;
- P_b: set of black boxes that directly precede black box $b \in B$;
- B_{end}: set of black boxes with no successors, i.e. $B_{\text{end}} = \{b \,|\, b \in B;\ b \notin P_{b'},\ \forall\, b' \in B\}$;
- n_b: number of required simulations at black box $b \in B$;
- s_b: time per simulation run at black box $b \in B$.

Using this notation, the black boxes can always be numbered in such a way that their numbering reflects the precedence ordering in the chain, i.e. $b \notin P_{\tilde{b}}$, $\tilde{b} = 1, \ldots, b'$, if $b \geq b'$.

Note that $P_b = \emptyset$ if black box $b \in B$ is at the beginning of the chain, and that $b \in B_{\text{end}}$ if b is at the end of the chain.

Figure 6.3 provides a numerical example, which will be used to clarify the throughput-time computation for each coordination method in the rest of the text. This figure shows eleven black boxes (BB1 up to BB11) that are coupled by response input parameters. The actual time-unit of the simulation times is irrelevant for our discussion; we let it be minutes.

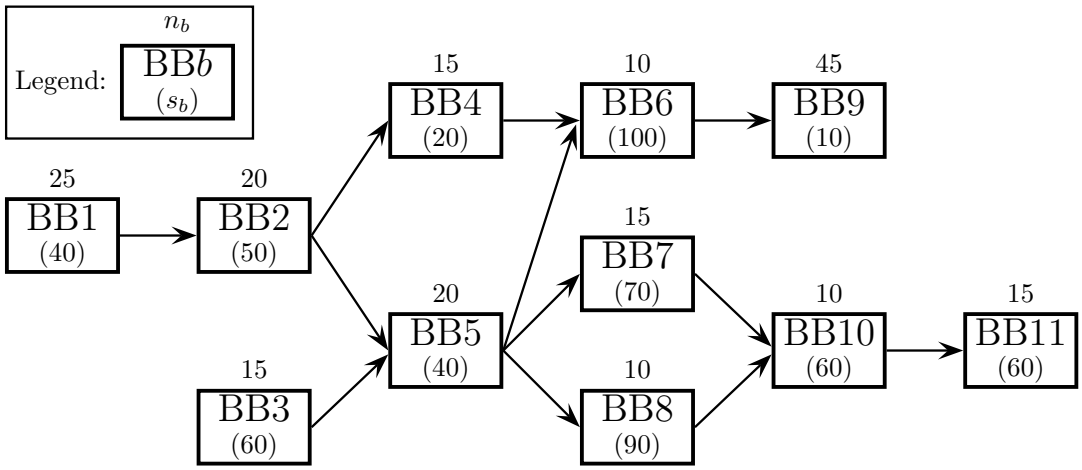


Figure 6.3: Example of eleven coupled black boxes.

6.5.1 Parallel Simulation

In Parallel Simulation all design teams carry out their evaluations concurrently. Hence, the corresponding throughput time, denoted by TT_{parallel}, is equal to the maximum of the total simulation times at every black box:

$$TT_{\text{parallel}} = \max_{b \in B} n_b s_b. \tag{6.1}$$

We call a black box a *bottleneck* when a (small) increase of its simulation time s_b results in an increase of the throughput time. The bottlenecks in Parallel Simulation are all black boxes $\hat{b} \in B$ that satisfy the equation

$$\hat{b} = \arg\max_{b \in B} n_b s_b. \tag{6.2}$$

For the black-box chain in Figure 6.3 it can readily be computed that black box 7 (BB7) forms the bottleneck and that TT_{parallel} is equal to 1050 minutes.

6.5.2 Sequential Simulation

In Sequential Simulation observed response values at a particular black box are passed on to its successors. This process can be viewed as a flow of information objects through the black-box chain, where each information object contains a scenario evaluated at a preceding black box. We refer to these objects as *global (component) design evaluations*. The maximum number of this type of evaluations, say, $\hat{n}$, is restricted by the minimum number of required evaluations per black box, i.e. $\hat{n} = \min_{b \in B} n_b$. Since $\hat{n}$ is a minimum, there may be several black boxes that require more evaluations. The observed responses of the latter are used only locally, i.e. at a certain black box, and are therefore referred to as *local design evaluations*. Clearly, every black box $b \in B$ invokes $\hat{n}$ global and $n_b - \hat{n}$ local design evaluations. A global design evaluation can be started only when all preceding black boxes have finished (at least) one such evaluation. Now, let the throughput-time function $f_b(n)$ represent the minimal time it takes for n global design evaluations to be finished at black box $b \in B$, then:

$$f_b(n) = \max \left\{ s_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(n),\ s_b + f_b(n-1) \right\},\ n \geq 1. \tag{6.3}$$

The interpretation of this formula is that black box b can only start its n-th evaluation when all its predecessors have finished n global design evaluations and it has evaluated $n-1$ of such evaluations itself. Note that $f_b(0) = 0$ implies that

$$f_b(1) = s_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(1). \tag{6.4}$$

This equation basically computes the longest path up to black box b, see e.g. Bazaara et al. (1990), starting from a black box at the beginning of the chain. Since (6.3) is dynamic in the variables b and n, it can be rewritten as

$$\begin{aligned}
f_b(n) &= \max \left\{ s_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(n),\ s_b + \max \left\{ s_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(n-1),\ s_b + f_b(n-2) \right\} \right\} \\
&= \max \left\{ s_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(n),\ 2s_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(n-1),\ 2s_b + f_b(n-2) \right\} \\
&= \max \left\{ s_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(n),\ 2s_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(n-1),\ 3s_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(n-2), \right. \\
&\qquad \left. \ldots,\ (n-1)s_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(2),\ ns_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(1) \right\} \\
&= \max_{r=1,\ldots,n} \left\{ rs_b + \max_{\tilde{b} \in P_b} f_{\tilde{b}}(n+1-r) \right\},\ n \geq 1.
\end{aligned} \tag{6.5}$$

It can be proven that $f_b(n)$ is convex in n. Therefore, (6.5) can be simplified to the following maximum function, which is dynamic in the variable b only.

$$\begin{aligned} f_b(n) &= \max_{r\in\{1,n\}} \left\{ r s_b + \max_{\tilde{b}\in P_b} f_{\tilde{b}}(n+1-r) \right\} \\ &= \max \left\{ s_b + \max_{\tilde{b}\in P_b} f_{\tilde{b}}(n),\ n s_b + \max_{\tilde{b}\in P_b} f_{\tilde{b}}(1) \right\} \\ &\overset{(6.4)}{=} \max \left\{ s_b + \max_{\tilde{b}\in P_b} f_{\tilde{b}}(n),\ (n-1)s_b + f_b(1) \right\},\ n \geq 2. \end{aligned} \tag{6.6}$$

Next, consider the following two sets:

$C_{b,n}$: set of possible bottlenecks up to black box $b \in B$ for n global design evaluations;

C_n: set of bottlenecks of the whole black-box chain for n global design evaluations.

Note that the black-box chain structure can cause both sets $C_{b,n}$ and C_n to differ significantly for distinct values of n. For $n = 1$, $f_b(1)$ follows from (6.4), and

$$C_{b,1} = \{b\} \cup \bigcup_{\tilde{b}\in I} C_{\tilde{b},1},\ \text{with } I = \left\{ \tilde{b} \,|\, \tilde{b} \in P_b;\ f_{\tilde{b}}(1) = f_b(1) - s_b \right\}, \tag{6.7}$$

yields the corresponding bottlenecks for all black boxes $b \in B$. Combining this information with (6.6) enables the computation of the throughput time $f(n)$ and the corresponding sets $C_{b,n}$ and C_n for every arbitrary integer $n \geq 2$, see Algorithm 6.2.

Algorithm 6.2 Throughput-time algorithm.

`FOR` $b = 1, 2, \ldots, |B|$ `DO`

$$f_b(n) = \max \left\{ s_b + \max_{\tilde{b}\in P_b} f_{\tilde{b}}(n),\ (n-1)s_b + f_b(1) \right\}; \tag{6.8}$$

$$C_{b,n} = \begin{cases} C_{b,1} \cup \bigcup_{\tilde{b}\in I} C_{\tilde{b},n} & \texttt{if } f_b(n) = (n-1)s_b + f_b(1), \\ \{b\} \cup \bigcup_{\tilde{b}\in I} C_{\tilde{b},n} & \texttt{otherwise,} \end{cases} \tag{6.9}$$

$$\texttt{where } I = \left\{ \tilde{b} \mid \tilde{b} \in P_b;\ f_{\tilde{b}}(n) = f_b(n) - s_b \right\};$$

`END`

$$f(n) = \max_{b\in B_{\text{end}}} f_b(n); \tag{6.10}$$

$$C_n = \bigcup_{b\in J} C_{b,n},\ \texttt{where } J = \{b \,|\, b \in B_{\text{end}};\ f_b(n) = f(n)\}. \tag{6.11}$$

The determination of the minimal time needed to evaluate all $\hat{n}$ global designs, i.e. $f(\hat{n})$, directly follows from Algorithm 6.2. To compute the throughput time, the time needed to evaluate the local designs at every black box must be included as well. In this respect, note that a black box may be idle for several periods of time during the whole simulation process of global designs. When it is possible to stop a simulation run at some point in time and later on proceed from that point on, we call the simulation runs *preemptive*. In this case, local designs can be evaluated within the idle periods. The throughput time of the Sequential Simulation method, denoted by $TT^{\text{pre}}_{\text{seqsim}}$, is then given by

$$TT^{\text{pre}}_{\text{seqsim}} = \max\Big\{\max_{b\in B} n_b s_b,\ f(\hat{n})\Big\} \overset{(6.1)}{=} \max\Big\{TT_{\text{parallel}},\ f(\hat{n})\Big\}. \tag{6.12}$$

In case $TT^{\text{pre}}_{\text{seqsim}} = TT_{\text{parallel}}$, the black boxes $\hat{b} \in B$ that satisfy (6.2) form the bottlenecks. If $TT^{\text{pre}}_{\text{seqsim}} = f(\hat{n})$, the black boxes $b \in C_{\hat{n}}$ are bottlenecks. Note that in the latter case not all bottlenecks may have the same impact on the throughput time, since the impact depends on the placement of the bottleneck within the black-box chain. The exact impact of each bottleneck, however, can easily be computed from Algorithm 6.2.

Unfortunately, simulation runs are often *non-preemptive*. Furthermore, since set-up times of computer experiments are generally not negligible, switching between different component designs within the simulation process may not be very practical. Finally, waiting until all global designs have been evaluated results in much more information for the design teams, which can be used to determine which local designs to evaluate best. For these reasons we suggest to use non-preemptive simulation runs and suggest to evaluate all local designs after the global designs have been evaluated. The throughput time, denoted by TT_{seqsim}, then becomes

$$TT_{\text{seqsim}} = \max_{b\in B} \left\{ f_b(\hat{n}) + (n_b - \hat{n}) s_b \right\}. \tag{6.13}$$

In this case, the bottlenecks are given by the set

$$C^*_{\hat{n}} = \bigcup_{b\in I} C_{b,\hat{n}},\ \text{with } I = \left\{ b \,|\, b \in B;\ f_b(\hat{n}) + (n_b - \hat{n}) s_b = TT_{\text{seqsim}} \right\}. \tag{6.14}$$

As above, the impact of these bottlenecks may vary.

For the numerical example in Figure 6.3 it follows from (6.8) and (6.10) that

$$f(n) = \begin{cases} 250 + 90n & \text{if } n \le 11, \\ 140 + 100n & \text{if } n \ge 11. \end{cases} \tag{6.15}$$

This equation and the fact that $\hat{n} = 10$ yield $f(\hat{n}) = 1150$. Recall that $TT_{\text{parallel}} = 1050$ minutes, and, hence, (6.12) results in $TT^{\text{pre}}_{\text{seqsim}} = 1150$ minutes. Furthermore, since $f_{11}(\hat{n}) = f(\hat{n})$ it follows from (6.9) and (6.11) that the bottlenecks are given by the set

$C_{\hat{n}} = C_{11,\hat{n}} = \{1, 2, 5, 8, 10, 11\}$. Using (6.13), the throughput time TT_{seqsim} is found to be equal to 1490 minutes. Since $f_9(\hat{n}) + (n_9 - \hat{n})s_9 = TT_{\text{seqsim}}$, all black boxes in the set $C^*_{\hat{n}} = C_{9,\hat{n}} = \{1, 2, 5, 6, 9\}$ are bottlenecks.

6.5.3 Sequential Modeling

In Sequential Modeling all required simulation runs at a particular black box are carried out before any result is passed on to succeeding black boxes. Therefore, the throughput time, denoted by TT_{seqmod}, is equal to the longest path in the black-box chain, when the total simulation time per black box ($n_b s_b$) is taken on the nodes of the directed graph:

$$TT_{\text{seqmod}} = \max_{\hat{B} \subset B} \sum_{b \in \hat{B}} n_b s_b, \text{ where } \hat{B} \text{ is a path in the chain.} \tag{6.16}$$

In (6.16) a path is defined as a sequence of black boxes starting at a beginning of the chain, i.e. at a black box b for which $P_b = \emptyset$, and ending at an end of the chain, i.e. at a black box $b \in B_{\text{end}}$. All black boxes on a longest, or critical, path form bottlenecks. In Figure 6.3, with $n_b s_b$ on the nodes, the black boxes $1, 2, 5, 7, 10$, and 11, form the (unique) longest path, and, therefore, the bottlenecks. The corresponding throughput time is equal to $TT_{\text{seqmod}} = 5350$ minutes.

6.5.4 Throughput-time relations

From the preceding observations some general relations between the throughput times of the different coordination methods can be derived:

$$TT_{\text{parallel}} \overset{(6.12)}{\leq} TT^{\text{pre}}_{\text{seqsim}} \overset{(6.13)}{\leq} TT_{\text{seqsim}} \leq TT_{\text{seqmod}}. \tag{6.17}$$

The first two inequalities readily follow from (6.12) and (6.13). It can easily be proven that the last inequality also holds.

6.6 Case study: Color picture tube design

This section summarizes the results obtained in an application of Collaborative Metamodeling to the design process of a color picture tube at LG.Philips Displays in Eindhoven. For a detailed discussion of this application the reader is referred to Stinstra, Stehouwer, and Van der Heijden (2003). The topic of this study is the collaborative design of several aspects of the shadow mask and screen for a color picture tube. The problem specification results in the problem structure that is depicted in Figure 6.4.

In this figure there are four black boxes, labelled Landing, MicMac center, MicMac northeast, and Microphony. The numbers in brackets denote the time needed per

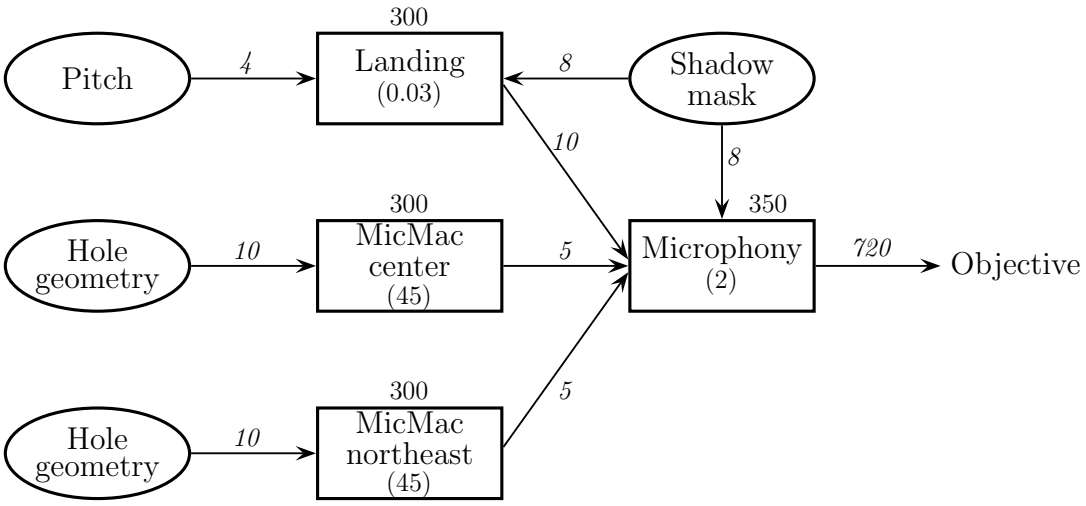


Figure 6.4: The problem structure.

evaluated scenario (in minutes) and the numbers above the black boxes show the number of evaluations performed. Ellipses represent design and response parameters, with the number of parameters in italics. The first columns of Table 6.2 summarize these black-box characteristics. Note in Figure 6.4 that MicMac center and MicMac northeast have only local design parameters as input, whereas Landing also has linking design parameters. Microphony takes its input from both linking design and response input parameters.

In this case study, Parallel Simulation was chosen as the coordination method. The main reason for this choice was that communication among the different departments was very hard. The design optimization tool COMPACT (see Stehouwer and Den Hertog (1999)) was used to construct designs for computer experiments and to obtain metamodels for all four black boxes. The quality of these metamodels, defined by the cross-validation root mean squared error (CV-RMSE), can be found in the last column of Table 6.2. COMPACT-CO, the collaborative version of COMPACT (cf. Stinstra, Stehouwer, and Van der Heijden (2003)), was used to combine the individual metamodels into a system-level metamodel. For the validation of this system-level metamodel a test set, taken from the predicted feasible product design space, was simulated.

Black box	# Design parameters	# Response parameters	# Simulations performed	Average relative CV-RMSE (%)
Landing	12	10	300	2.86
MicMac center	10	5	300	3.06
MicMac northeast	10	5	300	4.08
Microphony	28	720	350	8.60

Table 6.2: Black-box characteristics and metamodel validation results.

All approximation models used were quadratic, which lead to a quadratic optimization problem. CONOPT (cf. Drud (1994)) and a multi-start technique were used to find a global optimum. The optimal Microphony design found turned out to be an improvement of 50% with respect to the current design. In order to test the robustness of the design, Monte-Carlo analysis was applied to the design. Since metamodels are explicit functions, this type of analysis is very fast.

This case study showed that Collaborative Metamodeling improves the insight into the product design problem. Furthermore, the constructed metamodels can be used for Monte-Carlo analysis, to ensure that the product design remains valid under small perturbations. In our particular problem instance, however, it is to be expected that the quality of the metamodels for Microphony are improved when a sequential coordination method is applied, since the latter black box takes response parameters coming from other black boxes as input. Since Microphony directly affects the objective, applying a sequential coordination method would probably lead to better system-level metamodels, for the same number of evaluated scenarios. Unfortunately, using a sequential method complicates the coordination process significantly; the question is whether these costs outweigh the possibility of better metamodels.

Chapter 7

One-dimensional nested designs

Does it ever get easy?
– *You mean life?*
Yeah. Does it get easy?
– *What do you want me to say?*
Lie to me.
– *Yes. It's terribly simple. The good guys are always stalwart and true, the bad guys are easily distinguished by their pointy horns or black hats, and, uh, we always defeat them and save the day. No one ever dies, and everybody lives happily ever after.*
Liar.

(BtVS, *Episode 02.07*)

7.1 Introduction

In Part I of this thesis maximin designs, and maximin Latin hypercube designs in particular, have been considered as designs for computer experiments. In many practical design problems, however, there is often a need for *nested designs*. As defined in Section 2.4, a design is called "nested" when it consists of m separate designs, say, $X_1, X_2, \ldots, X_m$, one being a subset of the other, i.e. $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_m$. There are three main reasons for using nested designs: *linking design parameters*, *sequential evaluations*, and *training and test sets*.

To start with the first, consider a product that consists of two components, each of them represented by a black-box function. To obtain proper approximation models, a different number of function evaluations may be required for each black-box function. Moreover, in practice, it may occur that these two functions share one or more linking design parameters; see Figure 6.1. The evaluation of linking design parameters at the

same parameter settings in both functions (i.e. component-wise) leads to a system-level simulation, i.e. an evaluation of the product. Not only do such product evaluations provide a better understanding of the product, they are also very useful in the metamodel validation and product optimization processes; see Section 6.4.1. Another reason for using the same settings for (linking) design parameters is caused by physical restrictions on the simulation tools. Setting the parameters for computer experiments can be a time-consuming job in practice, since 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. Nested designs enable these common settings for the design parameters.

As an example of a real-life problem in which linking design parameters play a role, consider the case-study in Section 6.6. In this case-study, Stinstra, Stehouwer, and Van der Heijden (2003) apply Collaborative Metamodeling to optimize the design of a color picture tube. Such a tube consists of the main components: screen, electron gun, and shadow mask, and the relations among these components. Stinstra, Stehouwer, and Van der Heijden (2003) consider the collaborative design of several aspects of the shadow mask and the screen. Two of these aspects are the black-box functions describing Landing and Microphony; see Figure 6.4. The Landing function measures the quality of the image, whereas the Microphony function measures how vulnerable the shadow mask is to external vibrations. Since the response parameters of both Landing and Microphony depend on the settings of the design parameters of the shadow mask, linking design parameters play an important role; see Figure 7.1. As is argued in Section 6.4.1, the same settings should be used for these linking design parameters as much as possible, which gives rise to the need for a nested design.

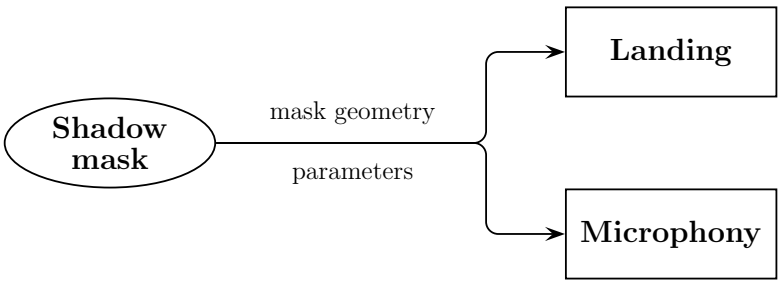


Figure 7.1: Linking design parameters in tube design optimization.

Sequential evaluations are a second reason for using nested designs. In practice, it may happen that after evaluating an initial set of design points extra evaluations are needed. For example, suppose that an approximation model is constructed for a particular black-box function, based on n_1 function evaluations. Should this model turn out to be invalid,

then either a different model should be fitted or an extra set of function evaluations is required to improve the current metamodel. In this latter case, the problem of constructing a design on a total of, say, n_2 points, given the initial design of n_1 points, arises. To anticipate the possibility of extra evaluations, one can construct the two designs (of n_1 and n_2 points) concurrently; hence, by constructing a nested design (or sequential design, see Section 2.4). Note that in this case the placement of the additional $n_2 - n_1$ points depends only on the current set of n_1 design points, and not on the fitted metamodel.

A third reason for using nested designs originates from the field of training and test sets. Consider the problem of fitting and validating a particular metamodel. First, the approximation model is fitted to the obtained data, i.e. the response values obtained when evaluating the design points in the training set. Then, a new set of design points, i.e. the test set, is evaluated and the obtained 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 said to be *valid*. Since the metamodel should be a global approximation model, i.e. it should be valid for the entire feasible region, the evaluation points, in both the training set and the test set, should cover the entire region. Moreover, the evaluation points in the test set should not lie too close to the evaluation points in the training set, i.e. the total set of evaluation points should be space-filling. Note that this is accomplished by nesting two designs, say, X_1 and X_2, with respect to, for example, the maximin criterion. The resulting sets X_1 and $X_2 \setminus X_1$ can then be used as the training set and the test set, respectively.

More formally, a nested design consists of $m \in \mathbb{N}$ separate k-dimensional designs (or sets of design points) $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_m$ and index sets $I_1 \subseteq I_2 \subseteq \cdots \subseteq I_m = \{1, 2, \ldots, n_m\}$, with $X_j = \{x_i = (x_{i1}, x_{i2}, \ldots, x_{ik}) \mid i \in I_j\}$ and $|I_j| = n_j$, $j = 1, \ldots, m$. Thus, the index set I_j defines which design points x_i are part of design X_j, $j = 1, \ldots, m$; the nested design is defined by the collection of individual designs X_j.

To obtain space-filling nested designs, the class of nested designs is optimized with respect to the maximin distance criterion. The current chapter deals with one-dimensional nested maximin designs, i.e. $k = 1$; the two-dimensional case is considered in Chapter 8. Without loss of generality, the lower and upper bounds on the design parameters are scaled such that all parameters take values in the interval $[0, 1]$. Note that when considering a particular design X_j independently, a space-filling distribution of the design points x_i (with $i \in I_j$) over the interval $[0, 1]$ is obtained by spreading the points equidistantly over the interval, resulting in a minimal distance of $\frac{1}{n_j - 1}$ between the points. We aim to determine the design points x_i and the index sets I_j such that every design X_j is as much space-filling as possible, with respect to the maximin criterion, under the "nesting" restriction. To this end, define d_j as the minimal scaled distance between all points in

the design X_j:

$$d_j = \min_{\substack{k,l \in I_j \\ k \neq l}} (n_j - 1)|x_k - x_l|, \text{ for all } j. \tag{7.1}$$

Maximizing the minimal distance $d = \min_j d_j$ over all index sets $I_1 \subseteq I_2 \subseteq \cdots \subseteq I_m$, with $|I_j| = n_j$, and design points $x_i \in [0, 1]$, yields a nested maximin design.

7.2 Nesting two designs

Let us first consider the case of nesting two designs, i.e. $m = 2$. Note that this case is of particular interest when using the sets X_1 and $X_2 \setminus X_1$ as a training set and a test set, respectively.

7.2.1 Maximin distance

The general problem of nesting two designs can be formalized as the following mathematical program:

$$\begin{aligned} \max \quad & \min_{\substack{k,l \in I_j \\ j=1,2;\, k \neq l}} (n_j - 1)\,|x_k - x_l| \\ \text{s.t.} \quad & I_1 \subseteq I_2 \\ & |I_j| = n_j, \qquad j = 1, 2 \\ & 0 \leq x_i \leq 1, \quad i \in I_2. \end{aligned} \tag{7.2}$$

To obtain a feasible solution that maximizes the objective function in (7.2), we may choose, without loss of generality, $x_1 = 0$, $x_{n_2} = 1$, $x_i < x_{i+1}$, $1 \in I_1$, and $n_2 \in I_1$. For a given I_1, containing the indices, say, $1 = a_1 < a_2 < \cdots < a_{n_1} = n_2$, consider the sequence $v = (v_1, v_2, \ldots, v_{n_1 - 1})$ given by $v_i = a_{i+1} - a_i$, $i = 1, \ldots, n_1 - 1$. Thus, $v_i - 1$ represents the number of additional points of X_2 between the i-th and $(i + 1)$-st point of X_1. It is clear that the set of possible I_1 is in one-to-one correspondence to the set of positive integer sequences v, summing up to $n_2 - 1$. Now, the approach to solve (7.2) is to first fix I_1, and its corresponding $a = (a_1, a_2, \ldots, a_{n_1})$ and v, and to obtain an expression for the maximal distance δ_v, subject to the remaining constraints, and then to maximize δ_v over all v. It turns out that finding δ_v is quite simple.

Lemma 7.1 *For fixed I_1, and corresponding a and v, the optimal value δ_v equals*

$$\left(\sum_{i=1}^{n_1 - 1} \max \left\{ \frac{v_i}{n_2 - 1}, \frac{1}{n_1 - 1} \right\} \right)^{-1}. \tag{7.3}$$

Proof. Fix a and v, and let δ_v be the corresponding maximal distance. Since $x_{i+1} - x_i \geq \frac{\delta_v}{n_2 - 1}$ for all i, it follows that $x_{a_{i+1}} - x_{a_i} \geq v_i \frac{\delta_v}{n_2 - 1}$. Furthermore, it also holds that

$x_{a_{i+1}} - x_{a_i} \geq \frac{\delta_v}{n_1-1}$, and, hence,

$$x_{a_{i+1}} - x_{a_i} \geq \max\left\{ v_i \frac{\delta_v}{n_2 - 1}, \frac{\delta_v}{n_1 - 1} \right\}. \tag{7.4}$$

From (7.4) it can be derived that

$$1 = x_{a_{n_1}} - x_{a_1} \geq \delta_v \sum_{i=1}^{n_1-1} \max\left\{ \frac{v_i}{n_2 - 1}, \frac{1}{n_1 - 1} \right\}, \tag{7.5}$$

which shows that the stated expression for δ_v (i.e. (7.3)) is an upper bound. It is clear from the above that, and how, this upper bound can be attained, which proves the lemma. □

What remains is to maximize δ_v over all appropriate sequences v. For ease of notation, define $c_2 = \frac{n_2-1}{n_1-1}$.

Proposition 7.1 *Let $2 \leq n_1 \leq n_2$. The maximin distance in (7.2) is given by*

$$d = \frac{1}{1 + \lfloor c_2 \rfloor + \lceil c_2 \rceil - c_2 - \lfloor c_2 \rfloor \lceil c_2 \rceil \frac{1}{c_2}}. \tag{7.6}$$

Proof. As mentioned before, δ_v has to be maximized, which is equivalent to minimizing

$$\sum_{i=1}^{n_1-1} \max\left\{ \frac{v_i}{n_2 - 1}, \frac{1}{n_1 - 1} \right\} \tag{7.7}$$

over all integer-valued sequences v, such that $\sum_{i=1}^{n_1-1} v_i = n_2 - 1$.

We claim that it is optimal to let v take only the values $\lfloor c_2 \rfloor$ and $\lceil c_2 \rceil$. This is clearly true if $n_2 - 1$ is a multiple of $n_1 - 1$ (i.e. $c_2 \in \mathbb{N}$), since, in that case, picking a larger value than c_2 for any of the elements v_i will increase the objective function. Therefore, let $n_2 - 1$ not be a multiple of $n_1 - 1$. To prove our claim, first assume that $v_i \leq \lfloor c_2 \rfloor - 1$ for some i, and let j be such that $v_j \geq \lceil c_2 \rceil$ (such a j exists). Then, by adding 1 to v_i, and subtracting 1 from v_j, the sequence v', for which the objective function is strictly smaller than for v, is obtained. This follows from the inequality

$$\begin{aligned} & \max\left\{ \frac{v_i}{n_2 - 1}, \frac{1}{n_1 - 1} \right\} + \max\left\{ \frac{v_j}{n_2 - 1}, \frac{1}{n_1 - 1} \right\} \\ > \; & \max\left\{ \frac{v_i + 1}{n_2 - 1}, \frac{1}{n_1 - 1} \right\} + \max\left\{ \frac{v_j - 1}{n_2 - 1}, \frac{1}{n_1 - 1} \right\}, \end{aligned} \tag{7.8}$$

which is easily checked to be true. Hence, the original v is not optimal. Similarly, the case where $v_i \geq \lceil c_2 \rceil + 1$ for some i is ruled out. Thus, it follows that the optimal v has

$v_i = \lfloor c_2 \rfloor$ for $p = (n_1 - 1)(\lceil c_2 \rceil - c_2)$ values of i, and $v_i = \lceil c_2 \rceil$ for the remaining i. The value for d now easily follows from Lemma 7.1. □

Figure 7.2 gives a graphical representation of the maximin distance as a function of n_1 and n_2.

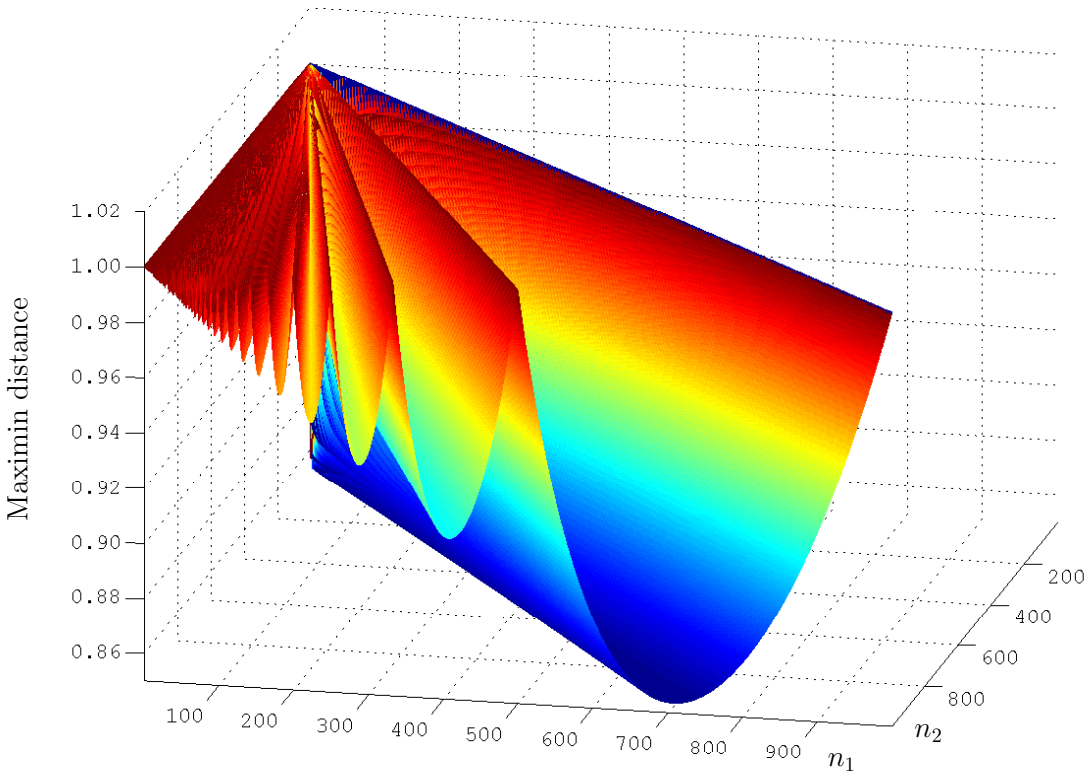


Figure 7.2: Maximin distance as a function of n_1 and n_2.

Using the preceding derivations, a nested maximin design can easily be constructed.

Construction 7.1 *Let $2 \leq n_1 \leq n_2$. A nested maximin design, with maximin distance d as in Proposition 7.1, is given by*

$$x_{i+1} = \begin{cases} \frac{d}{n_1 - 1} \frac{i}{\lfloor c_2 \rfloor} & i = 0, \ldots, p \lfloor c_2 \rfloor ; \\ \frac{d}{n_1 - 1} p + \frac{d}{n_2 - 1} (i - p \lfloor c_2 \rfloor) & i = p \lfloor c_2 \rfloor + 1, \ldots, n_2 - 1; \end{cases} \tag{7.9}$$

$$\begin{aligned} I_1 = & \left\{ 1 + j \lfloor c_2 \rfloor \mid j = 0, \ldots, p \right\} \bigcup \\ & \left\{ 1 + p \lfloor c_2 \rfloor + (j - p) \lceil c_2 \rceil \mid j = p + 1, \ldots, n_1 - 1 \right\}. \end{aligned} \tag{7.10}$$

As an example, consider a nested maximin design of $n_1 = 4$ and $n_2 = 8$ points. From Proposition 7.1 it follows that the maximin distance equals $d = \frac{21}{23} \approx 0.9130$. Substituting

d and $p = 2$ in Construction 7.1 yields the points $x_1 = 0$, $x_2 = \frac{7}{46}$, $x_3 = \frac{14}{46}$, $x_4 = \frac{21}{46}$, $x_5 = \frac{28}{46}$, $x_6 = \frac{34}{46}$, $x_7 = \frac{40}{46}$, and $x_8 = 1$, and the set $I_1 = \{1, 3, 5, 8\}$, implying that $X_1 = \{x_1, x_3, x_5, x_8\}$. This nested maximin design is depicted in Figure 7.3.

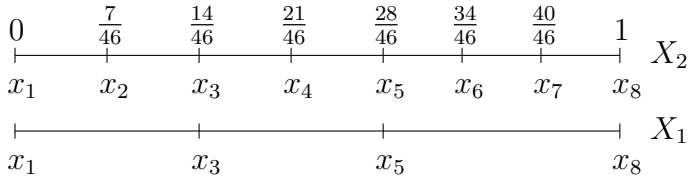


Figure 7.3: A nested maximin design of $n_1 = 4$ and $n_2 = 8$ points; $d = \frac{21}{23} \approx 0.9130$.

Besides computing the maximin distance for given n_1 and n_2, Proposition 7.1 can also be used to prove a general lower bound on the maximin distance.

Proposition 7.2 *Let* $2 \leq n_1 \leq n_2$. *Then* $1 \geq d > \left(4 - 2\sqrt{2}\right)^{-1} \approx 0.853553$.

Proof. Consider the function $z : [1, \infty) \to \mathbb{R}$, given by

$$z(c_2) = 1 + \lfloor c_2 \rfloor + \lceil c_2 \rceil - c_2 - \lfloor c_2 \rfloor \lceil c_2 \rceil \frac{1}{c_2} = 1 + \frac{(c_2 - \lfloor c_2 \rfloor)(\lceil c_2 \rceil - c_2)}{c_2} \geq 1. \tag{7.11}$$

If $c_2 \in \mathbb{N}$ then $z(c_2) = 1$, i.e. z is minimal, and, hence, $d = z(c_2)^{-1} \leq 1$. Else

$$z(c_2 + 1) = 1 + \frac{(c_2 - \lfloor c_2 \rfloor)(\lceil c_2 \rceil - c_2)}{c_2 + 1} < z(c_2). \tag{7.12}$$

Therefore, in this latter case z is maximal for some $c_2 \in (1, 2)$. Restricting z to $(1, 2)$ leads to:

$$z(c_2) = 1 + 1 + 2 - c_2 - \frac{2}{c_2} = 4 - c_2 - \frac{2}{c_2}, \tag{7.13}$$

which is maximal for $c_2 = \sqrt{2}$. For $c_2 \in \mathbb{Q}$ and $c_2 \geq 1$ it follows that

$$z(c_2) < z(\sqrt{2}) = 4 - 2\sqrt{2}, \tag{7.14}$$

and then

$$d > \frac{1}{z(\sqrt{2})} = \frac{1}{4 - 2\sqrt{2}} = \frac{1}{2} + \frac{1}{4}\sqrt{2} \approx 0.853553. \tag{7.15}$$

□

Note that the obtained lower bound is tight since c_2 can be taken arbitrarily close to $\sqrt{2}$. The interpretation of this lower bound is that for all values of n_1 and n_2, when nesting the designs X_1 and X_2, never more than 14.64% is lost, with respect to the "restriction free" maximin distance. In practice this implies that a linking design parameter can be included in the maximin designs, or that the designs can be used as training and test sets, at a cost of using designs that are at most 14.64% worse with respect to space-fillingness.

In case of sequential evaluations the interpretation is somewhat different. A standard way to perform two-stage sequential evaluations is to first evaluate n_1 design points, equidistantly distributed over the interval $[0, 1]$. After the evaluations, $n_2 - n_1$ extra design points are evaluated, resulting in an extended design of computer experiments with a minimal distance of $d' = \frac{c_2}{\lceil c_2 \rceil}$ between the design points; see Section 7.2.2. Clearly, $d \geq d'$ and $d' = \frac{c_2}{\lceil c_2 \rceil} \geq \frac{c_2}{c_2+1} > \frac{1}{2}$, for $c_2 > 1$.

If one evaluation stage turns out to be sufficient, a nested maximin design is at most 14.64% less space-filling, with respect to the (standard) equidistant design (since we lose $1 - d$). However, if a second evaluation stage is needed, then a nested maximin design is better space-filling than the extended equidistant design (since we gain $d - d'$). Figure 7.4 depicts the net gain of using a nested maximin design, i.e. $(d-1)+(d-d')$, as a function of c_2. For $n_2 \leq 100$ the net gain takes values in the interval $[-0.07, 0.48]$.

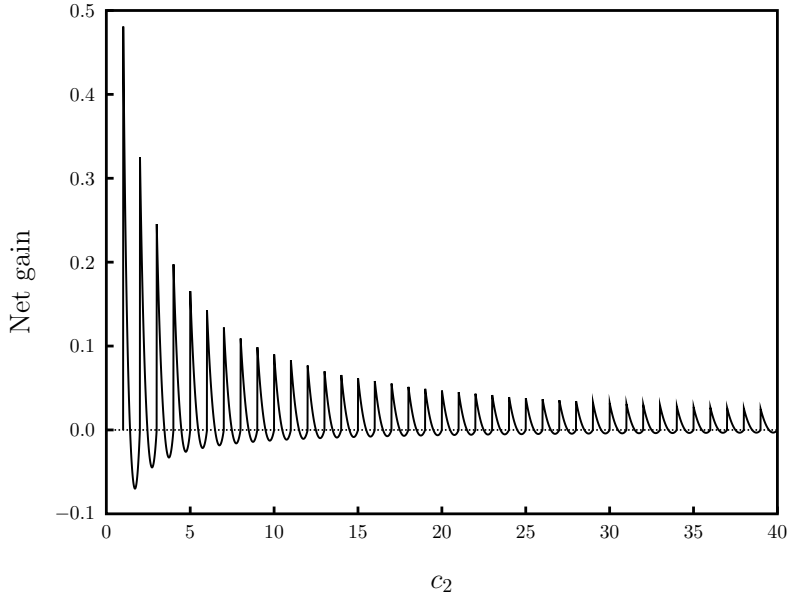


Figure 7.4: Net gain of using a nested maximin design, as a function of c_2.

7.2.2 Dominance

So far, the designs X_1 and X_2 have been assumed to be equally important. However, what if one design turns out to be more important than the other design? Or, put differently, given a fixed value for d_1, what is the corresponding maximal value of d_2? To examine this, we first introduce the notion of *dominance*. We call a combination (d_1, d_2) dominant if it is not possible to improve one of the coordinates without deteriorating the other coordinate. Knowledge of the dominant combinations is very useful in practice since it enables the determination of the trade-off between d_1 and d_2. This helps in finding a

combination that best satisfies a particular requirement, like, for example, "design X_2 is more important than design X_1". Note that the maximin combination (d, d), with d as in Proposition 7.1, is dominant. The combinations $(1, \frac{c_2}{\lceil c_2 \rceil})$ and $(\frac{\lfloor c_2 \rfloor}{c_2}, 1)$ are also dominant, which can be argued as follows:

- Fixing $d_1 = 1$, the design points of X_1 must be equidistantly distributed, i.e. $X_1 = \left\{0, \frac{1}{n_1-1}, \frac{2}{n_1-1}, \ldots, 1\right\}$. Due to the restriction $I_1 \subseteq I_2$, the $n_2 - n_1$ extra design points in X_2 need to be chosen such that d_2 is maximal. This is accomplished by choosing these extra points as equally as possible spread over the $n_1 - 1$ intervals formed by the points in X_1, which corresponds to v taking only the values $\lfloor c_2 \rfloor$ and $\lceil c_2 \rceil$. Hence, after scaling, this results in a minimal distance of

$$d = d_2 = (n_2 - 1)\left(\frac{1}{n_1 - 1} \cdot \frac{1}{\lceil c_2 \rceil}\right) = \frac{c_2}{\lceil c_2 \rceil}. \tag{7.16}$$

- Next, fixing $d_2 = 1$, the design points of X_2 must be equidistantly distributed, i.e. $X_2 = \left\{0, \frac{1}{n_2-1}, \frac{2}{n_2-1}, \ldots, 1\right\}$. To maximize d_1, the $n_2 - 1$ intervals should as equally as possible be spread over the $n_1 - 1$ intervals that are to be formed by the design points in X_1. Every interval of X_1 then contains either $\lfloor c_2 \rfloor$ or $\lceil c_2 \rceil$ intervals of length $\frac{1}{n_2-1}$. Hence, the minimal distance, after scaling, is given by

$$d = d_1 = (n_1 - 1)\left(\frac{1}{n_2 - 1}\lfloor c_2 \rfloor\right) = \frac{\lfloor c_2 \rfloor}{c_2}. \tag{7.17}$$

Since $\left(1, \frac{c_2}{\lceil c_2 \rceil}\right)$ and $\left(\frac{\lfloor c_2 \rfloor}{c_2}, 1\right)$ bound the values of d_1 and d_2, they are referred to as *extreme dominant combinations*. Moreover, note that these bounds imply that $d \geq \max\left\{\frac{c_2}{\lceil c_2 \rceil}, \frac{\lfloor c_2 \rfloor}{c_2}\right\}$. For given n_1 and n_2, all corresponding dominant combinations can be characterized by a linear function.

Proposition 7.3 *Let $2 \leq n_1 \leq n_2$. All dominant combinations (d_1, d_2) are characterized by the linear function $f : \left[\frac{\lfloor c_2 \rfloor}{c_2}, 1\right] \to \left[\frac{c_2}{\lceil c_2 \rceil}, 1\right]$, where*

$$d_2 = f(d_1) = \Big((c_2 - \lceil c_2 \rceil)\, d_1 + 1\Big)\frac{c_2}{\lceil c_2 \rceil(c_2 - \lfloor c_2 \rfloor)}. \tag{7.18}$$

Proof. Like in Lemma 7.1, for a given I_1, and corresponding a and v, it holds that

$$1 \geq \sum_{i=1}^{n_1-1} \max\left\{\frac{v_i}{n_2 - 1}d_2,\ \frac{1}{n_1 - 1}d_1\right\}. \tag{7.19}$$

Hence, for a given a, v, and $d_1 \leq 1$, it is optimal to choose d_2 as large as possible, such that equality is attained in (7.19).

We claim that for any d_1, with $\frac{\lfloor c_2 \rfloor}{c_2} \leq d_1 \leq 1$, a maximal d_2 is obtained by letting v take only the values $\lfloor c_2 \rfloor$ and $\lceil c_2 \rceil$, just like in Proposition 7.1. Note that this needs no further proof for $d_1 = \frac{\lfloor c_2 \rfloor}{c_2}$ and $d_1 = 1$; therefore, we may assume that $\frac{\lfloor c_2 \rfloor}{c_2} < d_1 < 1$, and, hence, that c_2 is not an integer.

To prove the claim, fix d_1, and suppose that there is a v resulting in an optimal d_2 with $v_i \geq \lceil c_2 \rceil + 1$ for some i. Let j be such that $v_j \leq \lfloor c_2 \rfloor$ (such a j exists). Since d_2 is optimal, we may assume that $d_2 \geq \frac{c_2}{\lceil c_2 \rceil}$. Now, let v' be obtained from v by subtracting 1 from v_i, and adding 1 to v_j. Since d_2 is optimal, the d_2' corresponding to v' is at most d_2. From the equalities in (7.19) for the pairs (v, d_2) and (v', d_2'), and the inequality $d_2' \leq d_2$, it follows that

$$\begin{aligned} & \max\left\{\frac{v_i}{n_2-1}d_2,\ \frac{1}{n_1-1}d_1\right\} + \max\left\{\frac{v_j}{n_2-1}d_2,\ \frac{1}{n_1-1}d_1\right\} \\ \leq\ & \max\left\{\frac{v_i-1}{n_2-1}d_2,\ \frac{1}{n_1-1}d_1\right\} + \max\left\{\frac{v_j+1}{n_2-1}d_2,\ \frac{1}{n_1-1}d_1\right\}. \end{aligned} \tag{7.20}$$

Because of the inequalities $v_i \geq \lceil c_2 \rceil + 1$, $v_j \leq \lfloor c_2 \rfloor$, $1 \geq d_2 \geq \frac{c_2}{\lceil c_2 \rceil}$, and $\frac{\lfloor c_2 \rfloor}{c_2} < d_1 < 1$, this reduces to

$$\frac{v_i}{n_2-1}d_2 + \frac{1}{n_1-1}d_1 \leq \frac{v_i-1}{n_2-1}d_2 + \max\left\{\frac{v_j+1}{n_2-1}d_2,\ \frac{1}{n_1-1}d_1\right\}. \tag{7.21}$$

This latter inequality implies that $\frac{v_j+1}{n_2-1}d_2 \geq \frac{1}{n_1-1}d_1$, and, hence, (7.21) further reduces to $\frac{1}{n_1-1}d_1 \leq \frac{v_j}{n_2-1}d_2$. Using that $\frac{\lfloor c_2 \rfloor}{c_2} < d_1$ and $d_2 \leq 1$, this implies that $v_j > \lfloor c_2 \rfloor$, which is a contradiction; hence, the considered v does not give an optimal d_2. Similarly, it can be shown that the case where $v_i < \lfloor c_2 \rfloor$ for some i is not optimal.

Thus, for any d_1 it is optimal to take a such that $v_i = \lfloor c_2 \rfloor$ for $p = (n_1 - 1)(\lceil c_2 \rceil - c_2)$ values of i, and $v_i = \lceil c_2 \rceil$ for the remaining i. The value for d_2 as a function of d_1 then easily follows from equality in (7.19). □

Note that for fixed a and v, the relation between d_1 and d_2 is a piecewise-linear function. Furthermore, note that for $c_2 \in \mathbb{N}$ the graph of this function results in the single point $(1, 1)$, and that setting d_1 equal to d_2 in (7.18) yields the maximin distance d, with d as in Proposition 7.1. In Figure 7.5 a graphical example of the linear function f is depicted. This figure shows the set of dominant combinations for $n_1 = 4$ and $n_2 = 8$ points, including the two extreme dominant combinations $(1, \frac{c_2}{\lceil c_2 \rceil}) \approx (1, 0.7778)$ and $(\frac{\lfloor c_2 \rfloor}{c_2}, 1) \approx (0.8571, 1)$ (depicted by the points on the border). Moreover, the line $d_1 = d_2$ intersects the dominant set exactly in the maximin combination $(d, d) = (\frac{21}{23}, \frac{21}{23}) \approx (0.9130, 0.9130)$ (depicted by the point in the middle).

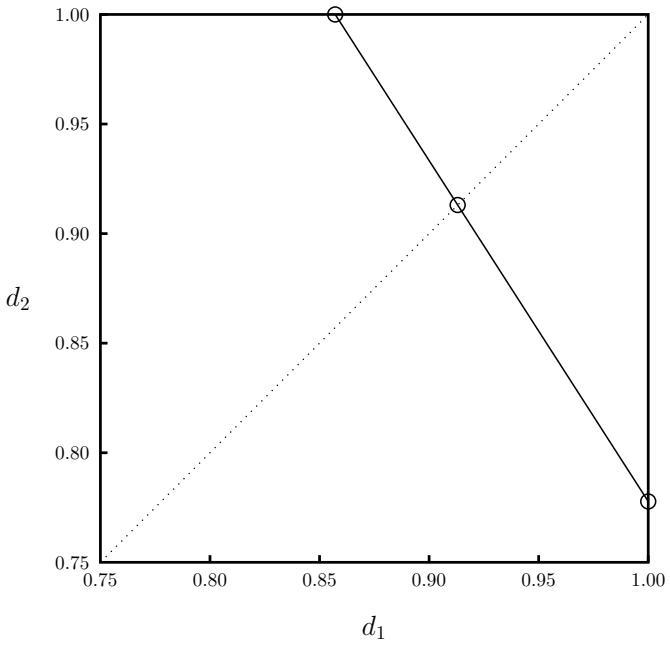


Figure 7.5: All dominant combinations (d_1, d_2) for $n_1 = 4$ and $n_2 = 8$ points, and the line $d_1 = d_2$.

7.3 Nesting three designs

Next, consider the case of nesting three designs, i.e. $m = 3$.

7.3.1 Maximin distance

The general problem of nesting three designs can be formalized as the following mathematical program:

$$\begin{aligned} \max \quad & \min_{\substack{k,l \in I_j \\ j=1,2,3;\, k \neq l}} (n_j - 1)\,|x_k - x_l| \\ \text{s.t.} \quad & I_1 \subseteq I_2 \subseteq I_3 \\ & |I_j| = n_j, \qquad j = 1, 2, 3 \\ & 0 \leq x_i \leq 1, \qquad i \in I_3. \end{aligned} \tag{7.22}$$

As in Section 7.2.1, we may choose, without loss of generality, $x_1 = 0$, $x_{n_3} = 1$, $x_i < x_{i+1}$, $1 \in I_1$, $n_3 \in I_1$, $1 \in I_2$, and $n_3 \in I_2$. For a given I_2, containing the indices, say, $1 = b_1 < b_2 < \cdots < b_{n_2} = n_3$, consider the sequence $w = (w_1, w_2, \ldots, w_{n_2-1})$ given by $w_j = b_{j+1} - b_j$, $j = 1, \ldots, n_2 - 1$. Given an I_1 contained in this I_2, let $1 = a_1 < a_2 < \cdots < a_{n_1} = n_2$ be such that $b_{a_i} \in I_1$ for $i = 1, \ldots, n_1$. Note that in this case the set $\{a_i \mid i = 1, \ldots, n_1\} \neq I_1$. As before, let $v_i = a_{i+1} - a_i$. Thus, $v_i - 1$ represents the number of additional points of X_2 between the i-th and $(i+1)$-st point of X_1, while $w_j - 1$ represents the number of additional points of X_3 between the j-th and $(j+1)$-st point of X_2. Now, the analogue of Lemma 7.1 is the following.

Lemma 7.2 *For fixed I_1, I_2, and corresponding a, b, v, and w, the optimal value $\delta_{a,w}$ equals*

$$\left(\sum_{i=1}^{n_1-1} \max \left\{ \sum_{j=a_i}^{a_{i+1}-1} \max \left\{ \frac{w_j}{n_3-1}, \frac{1}{n_2-1} \right\}, \frac{1}{n_1-1} \right\} \right)^{-1}. \tag{7.23}$$

We would now have to maximize $\delta_{a,w}$ over all appropriate sequences a and w. Unfortunately, we are not able to come up with an explicit formula for the maximin distance, as we did for the case of nesting two designs; see Section 7.2.1. However, (7.22) can be rewritten as a mixed integer linear program:

$$\begin{array}{lll} \max & d & \\ \text{s.t.} & d \le (n_3-1)(x_{i+1}-x_i), & i \in I_3 \setminus \{n_3\} \\ & d \le (n_j-1)(x_l-x_k)+2-z_{jk}-z_{jl}, & j=1,2;\ k,l \in I_3;\ k<l \\ & \sum_{k=1}^{n_3} z_{jk} = n_j, & j=1,2 \\ & z_{1k} \le z_{2k}, & k \in I_3 \\ & 0 \le x_i \le 1, & i \in I_3 \\ & z_{jk} \in \{0,1\}, & j=1,2;\ k \in I_3. \end{array} \tag{7.24}$$

Here, $z_{jk}=1$ if $k \in I_j$, and $z_{jk}=0$ otherwise. The constraints $\sum_{k=1}^{n_3} z_{jk} = n_j$ and $z_{1k} \le z_{2k}$ insure that $|I_j| = n_j$ and $I_1 \subseteq I_2$, respectively. Solving (7.24) with the *XA Binary and Mixed Integer Solver* of Sunset Software Technology (2003), we obtained results up to $n_3 = 25$ points. Computation times varied from 1 second to almost 2.5 hours of CPU-time for some instances, on a PC with an 800-MHz Pentium III processor.

As an example, consider a nested maximin design of $n_1 = 4$, $n_2 = 8$, and $n_3 = 18$ points. Solving (7.24) for this instance yields the sets $I_1 = \{1, 7, 12, 18\}$ and $I_2 = \{1, 4, 7, 10, 12, 14, 16, 18\}$, which results in the designs $X_1 = \{x_1, x_7, x_{12}, x_{18}\}$ and $X_2 = \{x_1, x_4, x_7, x_{10}, x_{12}, x_{14}, x_{16}, x_{18}\}$, with maximin distance $d = \frac{357}{398} \approx 0.8970$. Figure 7.6 depicts a graphical representation of this nested maximin design.

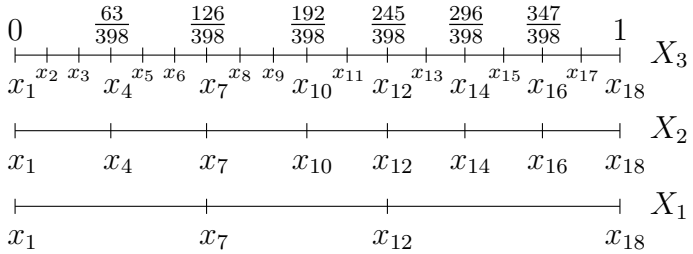


Figure 7.6: A nested maximin design of $n_1 = 4$, $n_2 = 8$, and $n_3 = 18$ points; $d = \frac{357}{398} \approx 0.8970$.

Although we do not have an explicit formula for the maximin distance, we can prove

a general lower bound on this distance. To accomplish this, let $d(n_1, n_2, n_3)$ denote the optimal value for d as function of (n_1, n_2, n_3).

Lemma 7.3 *Let* $2 \leq n_1 \leq n_2 \leq n_3$. *Then*

$$d(n_1, n_2, n_3) \leq d(n_1, n_2, n_3 + n_2 - 1). \tag{7.25}$$

Proof. Consider any a and w for the problem of (n_1, n_2, n_3). For the problem of $(n_1, n_2, n_3 + n_2 - 1)$, consider the same a, and w', which is given by $w'_j = w_j + 1$ for all j. Since

$$\max\left\{\frac{w_j + 1}{n_3 + n_2 - 1 - 1},\ \frac{1}{n_2 - 1}\right\} \leq \max\left\{\frac{w_j}{n_3 - 1},\ \frac{1}{n_2 - 1}\right\}, \tag{7.26}$$

which is easy to show, this implies that $\delta_{a,w'}(n_1, n_2, n_3 + n_2 - 1) \geq \delta_{a,w}(n_1, n_2, n_3)$, and the result follows. □

Proposition 7.4 *Let* $2 \leq n_1 \leq n_2 \leq n_3$. *Then* $1 \geq d(n_1, n_2, n_3) > \left(6 - 3\sqrt[3]{4}\right)^{-1} \approx 0.807887$.

Proof. Let (again) $c_2 = \frac{n_2 - 1}{n_1 - 1}$ and $c_3 = \frac{n_3 - 1}{n_2 - 1}$. First, note that $d(n_1, n_2, n_3) = 1$ if and only if $c_2, c_3 \in \mathbb{N}$. Because of Lemma 7.3, we may assume, without loss of generality, that $c_3 < 2$. To prove the stated inequality, we shall give an a and w such that $\delta_{a,w}(n_1, n_2, n_3) > (6 - 3\sqrt[3]{4})^{-1}$.

Let a be such that the corresponding v takes the value $v_i = \lfloor c_2 \rfloor$ for $i = 1, \ldots, p$, with $p = (n_1 - 1)(\lceil c_2 \rceil - c_2)$, and $v_i = \lceil c_2 \rceil$ for the remaining i, i.e. it is the optimal a when nesting two designs. Since $c_3 < 2$, it is possible to take w such that each w_j is equal to 1 or 2, and we shall do so. To further describe w, two cases are distinguished.

In the first case, $n_3 - n_2 \geq \lfloor c_2 \rfloor (n_1 - 1)(\lceil c_2 \rceil - c_2)$, and w is chosen such that $w_j = 2$ for $j = 1, \ldots, n_3 - n_2$, and $w_j = 1$ for the remaining j. Since $\max\left\{\frac{2}{n_3 - 1},\ \frac{1}{n_2 - 1}\right\} = \frac{2}{n_3 - 1}$, it follows that

$$\begin{aligned}
\delta_{a,w}^{-1} &= (n_1 - 1)(\lceil c_2 \rceil - c_2) \max\left\{\frac{2\lfloor c_2 \rfloor}{n_3 - 1},\ \frac{1}{n_1 - 1}\right\} + \\
&\quad (n_3 - n_2 - \lfloor c_2 \rfloor (n_1 - 1)(\lceil c_2 \rceil - c_2)) \frac{2}{n_3 - 1} + (n_2 - 1 - (n_3 - n_2)) \frac{1}{n_2 - 1} \\
&= (\lceil c_2 \rceil - c_2) \max\left\{0,\ 1 - \frac{2\lfloor c_2 \rfloor}{c_2 c_3}\right\} + 4 - c_3 - \frac{2}{c_3}.
\end{aligned} \tag{7.27}$$

Thus, if $2\lfloor c_2 \rfloor < c_2 c_3$, then $\delta_{a,w}^{-1} = (\lceil c_2 \rceil - c_2)(1 - \frac{2\lfloor c_2 \rfloor}{c_2 c_3}) + 4 - c_3 - \frac{2}{c_3}$. Call this expression $f(c_2)$, then it is easy to see that $f(c_2 + 1) < f(c_2)$; hence, we may restrict our attention

to the case where $1 < c_2 < 2$. From the above it is obtained that $\delta_{a,w}^{-1} = 6 - c_2 - c_3 - \frac{4}{c_2 c_3}$. This expression is at most $6 - 3\sqrt[3]{4}$, a value that is attained only if $c_2 = c_3 = \sqrt[3]{4}$. The case $2\lfloor c_2 \rfloor \geq c_2 c_3$ is straightforward (in this case $\delta_{a,w}^{-1} \leq 4 - 2\sqrt{2}$), so for the case $n_3 - n_2 \geq \lfloor c_2 \rfloor (n_1 - 1)(\lceil c_2 \rceil - c_2)$ the lower bound on d is proven.

In the second case, $n_3 - n_2 < \lfloor c_2 \rfloor (n_1 - 1)(\lceil c_2 \rceil - c_2)$, and we may assume that c_2 is not an integer. Let (again) $p = (n_1 - 1)(\lceil c_2 \rceil - c_2)$, and introduce $t = \frac{n_3 - n_2}{p} = \frac{c_2(c_3-1)}{\lceil c_2 \rceil - c_2}$; it follows that $\lceil t \rceil \leq \lfloor c_2 \rfloor$. Take w as follows: for $m(t - \lfloor t \rfloor)$ values of $i = 1, \ldots, m$ there are $\lceil t \rceil$ values of j, $a_i \leq j < a_{i+1}$, for which $w_j = 2$, and the remaining $\lfloor c_2 \rfloor - \lceil t \rceil$ of such elements j have $w_j = 1$; for the other values of $i = 1, \ldots, m$ there are $\lfloor t \rfloor$ values of j, $a_i \leq j < a_{i+1}$, for which $w_j = 2$, and the remaining $\lfloor c_2 \rfloor - \lfloor t \rfloor$ of such elements j have $w_j = 1$; and all $j \geq a_{m+1}$ have $w_j = 1$. From Lemma 7.2 it then follows that

$$\begin{aligned}
\delta_{a,w}^{-1} &= m(t - \lfloor t \rfloor) \max\left\{ \lceil t \rceil \frac{2}{n_3 - 1} + (\lfloor c_2 \rfloor - \lceil t \rceil)\frac{1}{n_2 - 1},\ \frac{1}{n_1 - 1} \right\} + \\
&\quad m(1 - t + \lfloor t \rfloor) \max\left\{ \lfloor t \rfloor \frac{2}{n_3 - 1} + (\lfloor c_2 \rfloor - \lfloor t \rfloor)\frac{1}{n_2 - 1},\ \frac{1}{n_1 - 1} \right\} + \\
&\quad (n_1 - 1)(c_2 - \lfloor c_2 \rfloor)\lceil c_2 \rceil \frac{1}{n_2 - 1} \\
&= \frac{\lceil c_2 \rceil - c_2}{c_2}(t - \lfloor t \rfloor) \max\left\{ \lceil t \rceil (\frac{2}{c_3} - 1),\ c_2 - \lfloor c_2 \rfloor \right\} + \\
&\quad \frac{\lceil c_2 \rceil - c_2}{c_2}(1 - t + \lfloor t \rfloor) \max\left\{ \lfloor t \rfloor (\frac{2}{c_3} - 1),\ c_2 - \lfloor c_2 \rfloor \right\} + 1. \qquad (7.28)
\end{aligned}$$

Now, assume that

$$\lfloor t \rfloor (\frac{2}{c_3} - 1) < c_2 - \lfloor c_2 \rfloor < \lceil t \rceil (\frac{2}{c_3} - 1), \qquad (7.29)$$

otherwise it is straightforward to show that $\delta_{a,w}^{-1} \leq 4 - 2\sqrt{2}$. Then

$$\delta_{a,w}^{-1} = \frac{\lceil c_2 \rceil - c_2}{c_2}(t - \lfloor t \rfloor)\lceil t \rceil (\frac{2}{c_3} - 1) + \frac{\lceil c_2 \rceil - c_2}{c_2}(1 - t + \lfloor t \rfloor)(c_2 - \lfloor c_2 \rfloor) + 1. \qquad (7.30)$$

Using the left inequality in (7.29) leads to

$$\begin{aligned}
\delta_{a,w}^{-1} &< \frac{\lceil c_2 \rceil - c_2}{c_2}(t - \lfloor t \rfloor)(\frac{2}{c_3} - 1) + \frac{\lceil c_2 \rceil - c_2}{c_2}(c_2 - \lfloor c_2 \rfloor) + 1 \\
&\leq (c_3 - 1)(\frac{2}{c_3} - 1) + \frac{\lceil c_2 \rceil - c_2}{c_2}(c_2 - \lfloor c_2 \rfloor) + 1 \\
&\leq 4 - 2\sqrt{2} + \frac{\lceil c_2 \rceil - c_2}{c_2}(c_2 - \lfloor c_2 \rfloor). \qquad (7.31)
\end{aligned}$$

If $c_2 > 4$, then this upper bound suffices (its maximum is attained at $c_2 = \sqrt{20}$), as one can easily check. For $c_2 < 4$, fix $k = \lceil t \rceil$ (≤ 3), and let $c_2 > k$. Then (7.30) reduces to

$$\begin{aligned}
\delta_{a,w}^{-1} &= 1 + 3k - kc_3 - \frac{2k}{c_3} - (c_3 - 1)(c_2 - \lfloor c_2 \rfloor) + \\
&\quad k\frac{\lceil c_2 \rceil - c_2}{c_2}\left(c_2 - \lfloor c_2 \rfloor - (k - 1)(\frac{2}{c_3} - 1) \right), \qquad (7.32)
\end{aligned}$$

the maximum of which is attained for some c_2 between k and $k+1$, i.e. $\lfloor c_2 \rfloor$ is minimal. For each $k = 1, 2, 3$ (separately) it is now possible to obtain an appropriate upper bound on $\delta_{a,w}^{-1}$, under the assumptions that $k \leq c_2 \leq k+1$ and $1 \leq c_3 \leq 2$. For $k = 1$, this upper bound is $6 - 3\sqrt[3]{4}$, and it is attained when $c_2 = c_3 = \sqrt[3]{2}$. □

Note that the obtained lower bound is tight since c_2 and c_3 can be taken arbitrarily close to $\sqrt[3]{2}$, and in these cases the given a and w are optimal; see Proposition 7.6. The interpretation of this lower bound is that for all values of n_1, n_2, and n_3, when nesting the designs X_1, X_2, and X_3, never more than 19.21% is lost, with respect to the "restriction free" maximin distance. In practice this implies that a linking design parameter can be included in the maximin designs, at a cost of using designs that are at most 19.21% worse with respect to space-fillingness.

Using a nested maximin design in case of three-stage sequential evaluations incurs a loss of $1 - d$ when one stage suffices. If two stages are sufficient, a net gain of $d - d'$ is obtained, where $d' = \frac{c_2}{\lceil c_2 \rceil}$ (see Section 7.2.1). Finally, when all three stages are needed $d - d''$ is gained, where $d'' \leq d' \leq d$ and $d'' > \frac{1}{2}$ for $c_3 > 1$. Thus, the net gain of using a nested maximin design is equal to $(d-1) + (d-d') + (d-d'')$, which takes values in the interval $[-0.19, 0.84]$ for $n_3 \leq 25$.

7.3.2 Dominance

The notion of dominance is introduced in Section 7.2.2. Similar as before, a combination (d_1, d_2, d_3) is called dominant if it is not possible to improve one of the coordinates without deteriorating another coordinate. Unlike the case of nesting two designs, the maximin combination (d, d, d) is not necessarily dominant. For example, for the combination $(n_1, n_2, n_3) = (4, 8, 17)$ the maximin distance equals $d(4, 8, 17) = 0.9130$; however, $(0.9130, 0.9130, 0.9275)$ is dominant. In Section 7.2.2 it is shown that $(1, \frac{c_2}{\lceil c_2 \rceil})$ and $(\frac{\lfloor c_2 \rfloor}{c_2}, 1)$ are extreme dominant combinations when nesting two designs. Extending this idea, i.e. fixing $d_j = 1$ and maximizing d_k, $k \neq j$, leads to extreme dominant combinations for the case of nesting three designs. Note that the extreme dominant combinations are again lower bounds on the maximin distance $d = d(n_1, n_2, n_3)$. An upper bound on d is obtained by the simple observation that $d(n_1, n_2, n_3) \leq \max\{d(n_1, n_2),\ d(n_1, n_3),\ d(n_2, n_3)\}$. Furthermore, it is easily shown that $d(n_1, n_2, n_3) = d(n_2, n_3)$, if and only if $c_2 \in \mathbb{N}$, and $d(n_1, n_2, n_3) = d(n_1, n_2)$, if and only if $c_3 \in \mathbb{N}$.

All this may lead to the belief that the idea of finding the maximin distance by means of extreme dominant combinations, as has been done for the case of nesting two designs, can be extended to the case of nesting three designs. For example, in Figure 7.7 it can be seen that the dominant combinations for $n_1 = 4$, $n_2 = 8$, and $n_3 = 18$ points fall in a plane through the extreme dominant combinations $(1, 0.7778, 0.9444)$,

$(0.8571, 1, 0.8095)$, and $(0.8824, 0.8235, 1)$. This plane intersects the line $d_3 = d_2 = d_1$ exactly in the maximin combination $(0.8970, 0.8970, 0.8970)$, strengthening the belief that this method also works for arbitrary values of (n_1, n_2, n_3). Unfortunately, dominant combinations do not always fall in a plane through the extreme dominant points, see e.g. Figure 7.8. Hence, such a plane can not always be used to find the maximin combination. As another example, consider the values $n_1 = 6$, $n_2 = 8$, and $n_3 = 12$. In this case, the plane through the extreme dominant combinations $(1, 0.7, 0.7333)$, $(0.7143, 1, 0.7857)$, and $(0.9091, 0.6364, 1)$, results in the unattainable combination $(0.8324, 0.8324, 0.8324)$, when intersected with the line $d_3 = d_2 = d_1$, thereby "missing" the correct maximin combination $(0.8262, 0.8262, 0.8262)$.

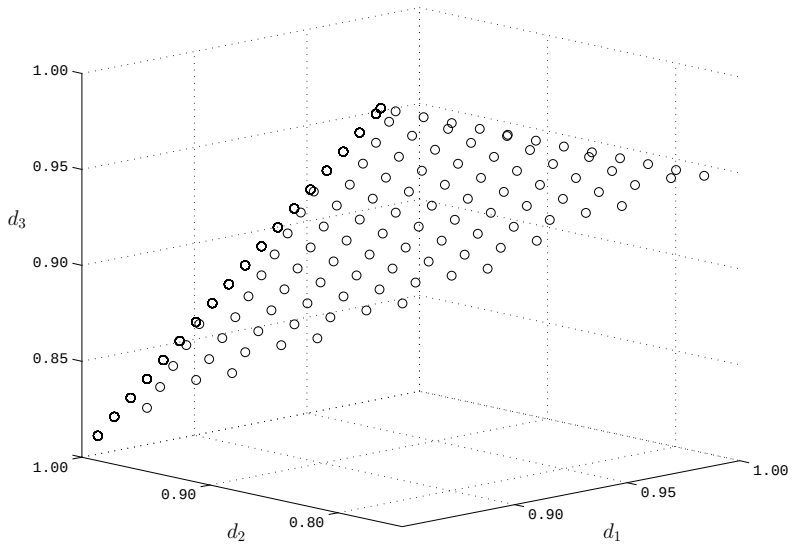


Figure 7.7: Dominant combinations for $n_1 = 4$, $n_2 = 8$, and $n_3 = 18$ points.

7.3.3 Heuristic

The previous section shows that it is not always possible to obtain the maximin distance by means of extreme dominant combinations. Note, however, that even if this method would work, this would still not result in the construction of a nested maximin design. Mixed integer linear programming could be used to find a nested design; unfortunately, due to lengthy computation times, this latter method is applicable only to small values of (n_1, n_2, n_3). To deal with these problems, we have built a heuristic that is able to find good nested designs, relatively fast. Furthermore, the obtained nested designs are conjectured to be optimal.

Our heuristic is based on the observation that all nested maximin designs, which have been found by solving (7.24), contain the optimal *assignments* of the corresponding cases of nesting two designs, as provided in (7.10), as part of their solutions. Compare, for

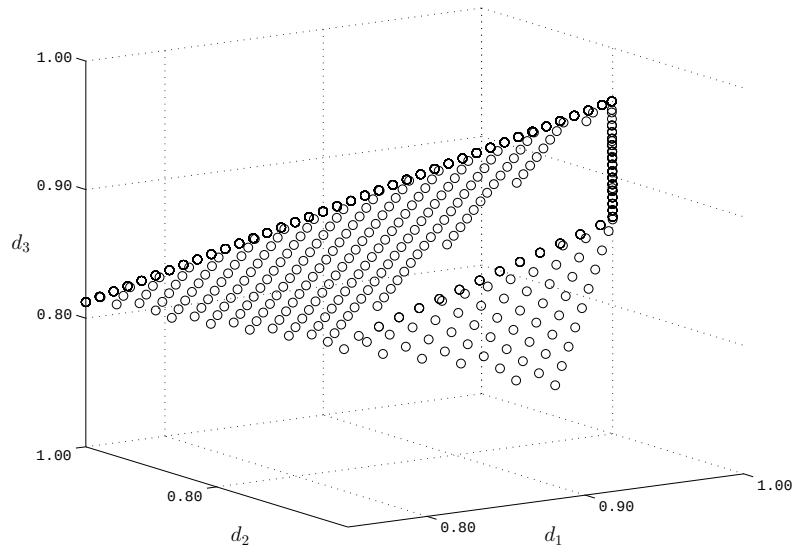


Figure 7.8: Dominant combinations for $n_1 = 4$, $n_2 = 9$, and $n_3 = 14$ points.

example, the designs depicted in Figures 7.3 and 7.6. Therefore, for given (n_1, n_2, n_3), with $c_2, c_3 \notin \mathbb{N}$, start by constructing a nested maximin design of n_1 and n_2 points, using Construction 7.1. Every interval $[x_l, x_{l+1}]$, $l \in I_2 \setminus \{n_2\}$, then has a width of at least $\frac{d}{n_2-1}$, with $d = d(n_1, n_2)$ as in (7.6). This implies that up to q points can be added to each interval, without decreasing d, as long as q satisfies the inequality

$$\frac{d(q+1)}{n_3 - 1} \leq \frac{d}{n_2 - 1}, \tag{7.33}$$

or, equivalently, $q + 1 \leq c_3$, which results in at most $q = \lfloor c_3 \rfloor - 1$ additional points per interval and $(\lfloor c_3 \rfloor - 1)(n_2 - 1)$ points in total. Hence, if $n_3 - n_2 \leq (\lfloor c_3 \rfloor - 1)(n_2 - 1)$, we are finished, since spreading the $n_3 - n_2$ points equally over the $n_2 - 1$ intervals will yield a nested maximin design with distance $d(n_1, n_2, n_3) = d(n_1, n_2)$. Note that this will hold only in case $c_3 \in \mathbb{N}$; however, we have chosen (n_1, n_2, n_3) such that $c_2, c_3 \notin \mathbb{N}$.

In case $n_3 - n_2 > (\lfloor c_3 \rfloor - 1)(n_2 - 1)$, $q = \lfloor c_3 \rfloor - 1$ points can be added to every interval and $r = (n_3 - n_2) - (\lfloor c_3 \rfloor - 1)(n_2 - 1) < (n_3 - n_2) - (c_3 - 2)(n_2 - 1) = n_2 - 1$ points remain to be added. These remaining r points are sequentially added to one of the $n_2 - 1$ intervals as follows. Consider the case where s points, $s \in \{0, \ldots, r-1\}$, have already been assigned and consider the index sets $I_1^s \subseteq I_2^s \subseteq I_3^s = \{1, \ldots, n_3'\}$, which describe the current nested design on (n_1, n_2, n_3'), with $n_3' = n_2 + (\lfloor c_3 \rfloor - 1)(n_2 - 1) + s$. The corresponding maximal distance can then readily be computed from Lemma 7.2. When assigning the $(s+1)$-st point, first compute (using Lemma 7.2) for each of the $n_2 - 1$ intervals what the new maximal distance will be, should the point be assigned to that particular interval. The interval for which this maximal distance is the largest is chosen, and the corresponding sets $I_1^{s+1} \subseteq I_2^{s+1} \subseteq I_3^{s+1}$ will describe the new nested design. In

case of a tie, the interval j, $j = 1, \ldots, n_2 - 1$, for which

$$\max\left\{\frac{(I_2^{s+1})_{j+1} - (I_2^{s+1})_j}{n_3 - 1},\ \frac{1}{n_2 - 1}\right\} - \max\left\{\frac{(I_2^{s})_{j+1} - (I_2^{s})_j}{n_3 - 1},\ \frac{1}{n_2 - 1}\right\} \tag{7.34}$$

is the smallest is chosen. Here, $(I_2^s)_j$ and $(I_2^{s+1})_j$ are defined as the j-th smallest elements of the sets I_2^s and I_2^{s+1}, respectively. The value in (7.34) can be interpreted as the relative cost of adding an extra design point to a particular interval. Leaving out this second objective may result in bad nested designs.

For given index sets I_1, I_2, and I_3, it takes $\mathcal{O}(n_1 n_2)$ time to compute the maximal distance, using Lemma 7.2. There are $s \leq r < n_2$ additional points to be added and for each of these points $n_2 - 1$ index sets have to be considered, hence, Lemma 7.2 has to be applied $\mathcal{O}({n_2}^2)$ times. Therefore, a nested design for (n_1, n_2, n_3) is found in $\mathcal{O}(n_1 {n_2}^3)$ time. Note that the complexity does not depend on n_3. Moreover, it turns out that the heuristic yields an optimal nested design for all values of (n_1, n_2, n_3) that have been considered so far, i.e. for $n_3 \leq 25$. This supports our conjecture that above heuristic will find a nested maximin design for all values of (n_1, n_2, n_3).

7.4 Nesting four or more designs

The general case of nesting $m \geq 4$ designs can be formalized as the following mathematical program:

$$\begin{array}{lll} \max & \min\limits_{\substack{k,l \in I_j;\, k \neq l \\ j=1,\ldots,m}} (n_j - 1)\,|x_k - x_l| & \\ \text{s.t.} & I_1 \subseteq I_2 \subseteq \cdots \subseteq I_m & \\ & |I_j| = n_j, & j = 1, \ldots, m \\ & 0 \leq x_i \leq 1, & i \in I_m. \end{array} \tag{7.35}$$

Furthermore, Lemmas 7.2 and 7.3 can easily be generalized. In particular, the following holds.

Lemma 7.4 *Let $2 \leq n_1 \leq \cdots \leq n_m$. Then*

$$d(n_1, n_2, \ldots, n_{m-1}, n_m) \leq d(n_1, n_2, \ldots, n_{m-1}, n_m + n_{m-1} - 1). \tag{7.36}$$

Now, consider the case where $n_m < 2n_1$. Let $c_j = \frac{n_j - 1}{n_{j-1} - 1}$, $j = 2, \ldots, m$, and $d = d(n_1, n_2, \ldots, n_m)$. For fixed I_1, let it contain the indices $1 = a_1 < a_2 < \cdots < a_{n_1} = n_m$. Note that this a is somewhat different from a in the previous section, in the sense that it now gives the relation between I_1 and I_m (and not between I_1 and I_2). As before, let the sequence $v = (v_1, v_2, \ldots, v_{n_1 - 1})$ be given by $v_i = a_{i+1} - a_i$. Thus, $v_i - 1$ represents the number of additional points of X_m between the i-th and $(i+1)$-st point of X_1.

Proposition 7.5 *Let $m \geq 3$ and $2 \leq n_1 \leq \cdots \leq n_m < 2n_1$. Then the maximal value for d equals*

$$\frac{1}{2m - \frac{2}{c_2} - \cdots - \frac{2}{c_m} - c_2 c_3 \cdots c_m}. \tag{7.37}$$

Proof. Consider an I_1 such that the corresponding v takes only values 1 and 2, i.e. between two neighboring points from X_1 there is at most one point from X_m. Since $\max\left\{\frac{2}{n_j-1}, \frac{1}{n_1-1}\right\} = \frac{2}{n_j-1}$, and since the number of i for which $v_i = 1$ equals $2n_1 - n_m - 1$, it follows that

$$\begin{aligned} \delta_v &= \left((2n_1 - n_m - 1)\frac{1}{n_1 - 1} + \sum_{j=2}^{m}(n_j - n_{j-1})\frac{2}{n_j - 1}\right)^{-1} \\ &= \left(2m - 2c_2^{-1} - \cdots - 2c_m^{-1} - c_2 c_3 \cdots c_m\right)^{-1}. \end{aligned} \tag{7.38}$$

That this v indeed yields the optimal d can be shown by comparing $\delta_{v'}^{-1}$, for a v' with $v'_i \geq 3$ for some i, to $\delta_{v''}^{-1}$, where v'' is obtained from v' by letting $v''_i = v'_i - 1$, and taking $v''_j = 2$ for a j with $v'_j = 1$. Such a j exists because of the condition $n_m < 2n_1$. Further technical details are omitted. □

Using Proposition 7.5, it is easy to show that the following holds.

Proposition 7.6 *Let $m \geq 2$ and $2 \leq n_1 \leq \cdots \leq n_m < 2n_1$. Then $1 \geq d > \left(2m(1 - \sqrt[m]{\frac{1}{2}})\right)^{-1}$.*

The lower bound for d is attained when $c_i = \sqrt[m]{2}$ for all i. We conjecture that this lower bound for d holds in all cases. This conjecture is supported by the results for $m = 2$ and $m = 3$, see Propositions 7.2 and 7.4, respectively. Furthermore, note that the sequence $\left(2m(1 - \sqrt[m]{\frac{1}{2}})\right)^{-1}$ is decreasing in m and converges to $\frac{1}{2\log 2} \approx 0.721348$. Hence, if our conjecture is correct, never more than 27.87% is lost, with respect to the "restriction free" maximin distance, when nesting the designs $X_1, X_2, \ldots, X_m$.

The mixed integer linear program for the case of nesting three designs (see (7.24)) has been extended to the case of nesting four designs, i.e. $m = 4$, and results have been obtained for n_4 up to 19. Unfortunately, as n_4 gradually increases the computation time rapidly grows, leading to some instances that take about 4 hours of CPU-time (with the *XA Binary and Mixed Integer Solver*). Therefore, the heuristic described in Section 7.3.3 has been extended to search for good nested designs, for given values of (n_1, n_2, n_3, n_4). This extended heuristic first constructs a nested design for the combination (n_1, n_2, n_3), which is conjectured to be an optimal nested design. Then, the $n_4 - n_3$ additional points are sequentially added, in the way described in Section 7.3.3.

As can be observed from Figure 7.9, for $n_4 \leq 19$ the heuristic often finds the maximin distance (and, hence, the corresponding nested maximin design), and it is not too far off in most other cases. Unfortunately, there is an instance, i.e. $(n_1, n_2, n_3, n_4) = (4, 6, 9, 14)$, for which the distance found by the heuristic is smaller than the (conjectured) lower bound in Proposition 7.6 (which is depicted by the dotted lines in Figure 7.9). For this instance, the heuristic obtains a distance of 0.7796, which is smaller than both the conjectured lower bound of 0.7857 and the maximin distance of $d(4, 6, 9, 14) = 0.7923$. Note, however, that, although the heuristic does not yield a distance that is at least as large as the conjectured lower bound, the conjecture may still be correct.

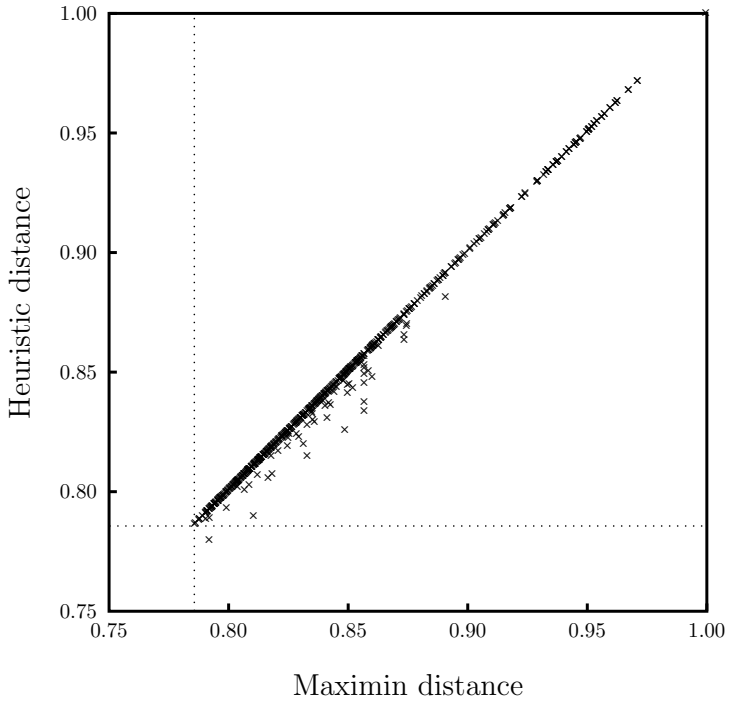


Figure 7.9: Maximal distances found by the heuristic versus the maximin distances, for all values of (n_1, n_2, n_3, n_4) with $n_4 \leq 19$.

Chapter 8

Two-dimensional nested designs

If the road is easy, you're likely going the wrong way.

(Terry Goodkind, *Soul of the Fire*)

8.1 Introduction

Chapter 7 introduces the concept of nested designs and shows how to construct one-dimensional nested maximin designs. In the current chapter we construct two-dimensional nested maximin designs. We focus on the problem of nesting two designs, X_1 and X_2, with $X_1 \subseteq X_2$, $X_j = \{x_i = (x_{i1}, x_{i2}) \mid i \in I_j\}$, and $|I_j| = n_j$, $j = 1, 2$. Thus, the index set $I_1 \subseteq I_2 = \{1, 2, \ldots, n_2\}$ 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.

As in Chapter 7, all design parameters are scaled such that they take values in the interval $[0, 1]$, and the class of nested designs is again optimized with respect to the maximin distance criterion. Furthermore, scaling factors s_1 and s_2 are introduced to enable a comparison of the minimal distances between the points in the designs X_1 and X_2, respectively. 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 with respect to the maximin criterion. To this end, define d_j as the minimal scaled distance between all points in the design X_j:

$$d_j = \min_{\substack{k,l \in I_j \\ k \neq l}} \frac{d(x_k, x_l)}{s_j}, \; j = 1, 2. \tag{8.1}$$

In this chapter, we consider the Euclidean distance measure (see (2.2)) for $d(\cdot, \cdot)$. Remember that the upper bound for the ℓ^2-maximin distance of two-dimensional unrestricted designs (and, hence, also for Latin hypercube designs) in (3.13) is of the order $\frac{1}{\sqrt{n-1}}$

(when rescaled to the unit square). Furthermore, for the ℓ^2-distances of the obtained two-dimensional (approximate) maximin Latin hypercube designs in Table 3.1 it often holds that d is approximately equal to $\frac{1}{\sqrt{n-1}}$ (again, when rescaled to the unit square). For these reasons we use the scaling factors $s_j = \frac{1}{\sqrt{n_j-1}}$, $j = 1, 2$, in (8.1). What remains is to maximize the minimal distance $d = \min\{d_1, d_2\}$ over all $I_1 \subseteq I_2$, with $|I_1| = n_1$, and $x_i \in [0, 1]^2$, to obtain a nested maximin design in terms of the set I_1 and the design points x_i.

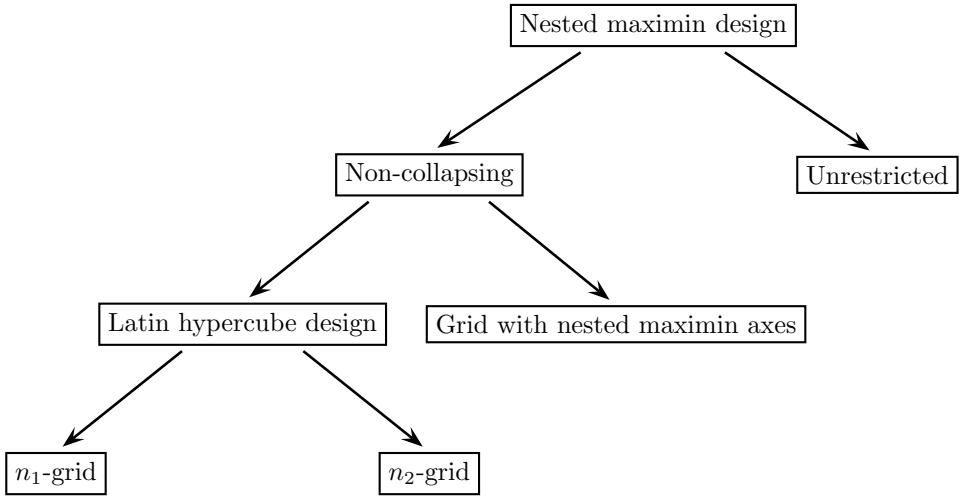


Figure 8.1: Several types of nested maximin designs.

Note that several different types of nested maximin designs can be distinguished, see Figure 8.1. A first division can be made by distinguishing between *unrestricted* (possibly collapsing) and *non-collapsing* designs. When nesting two designs, an unrestricted nested maximin design is obtained by solving the following optimization problem:

$$\begin{aligned} \max \quad & \min_{\substack{k,l \in I_j \\ j=1,2;\, k \neq l}} \sqrt{n_j - 1}\sqrt{(x_{k1} - x_{l1})^2 + (x_{k2} - x_{l2})^2} \\ \text{s.t.} \quad & I_1 \subseteq I_2 \\ & |I_j| = n_j, \qquad j = 1, 2 \\ & 0 \leq x_{ij} \leq 1, \quad i \in I_2,\, j = 1, 2. \end{aligned} \tag{8.2}$$

An example of such an unrestricted nested maximin design, of $n_1 = 4$ and $n_2 = 9$ points, is depicted in Figure 8.2. In this figure, 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. In this particular example, the "nesting" restriction does not reduce the maximin distances of the individual designs, i.e. they are both optimal (see Melissen (1997)), and, hence, the nested design is easy to construct. For most combinations of n_1 and n_2, however, the "nesting"

restriction does reduce the individual maximin distances, making it very hard to obtain a nested design. No general methods for nesting unrestricted designs exist, as of yet.

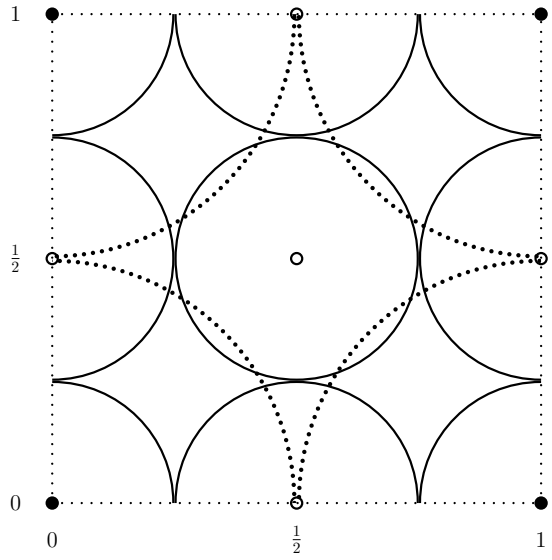


Figure 8.2: An unrestricted nested maximin design of $n_1 = 4$ and $n_2 = 9$ points; $d = d_2 = 1.4142$ and $d_1 = 1.7321$.

As has been argued in Chapter 2, it is important to use non-collapsing designs when dealing with deterministic computer experiments. Therefore, we consider this type of designs in the rest of the current chapter. Note that a non-collapsing nested maximin design is obtained by adding constraints that enforce the x_i-coordinates (in each dimension) to be separated by at least some distance $\alpha \in (0, 1]$ (see Chapter 5) to (8.2). Several choices for α are proposed in Sections 8.2 and 8.3. Section 8.2 considers the class of Latin hypercube designs, and n_1-*grids* and n_2-*grids* in particular. In Section 8.3 the underlying grid depends on the one-dimensional nested maximin designs obtained in Chapter 7. This latter type of grid is therefore referred to as a *grid with nested maximin axes*.

8.2 Latin hypercube designs

When nesting two designs, there are two ways to apply the Latin hypercube structure. The nested design will either be based on a Latin hypercube design of n_1 points, i.e. n_1 design points are chosen on the n_1-grid, with grid points $\left\{0, \frac{1}{n_1-1}, \frac{2}{n_1-1}, \ldots, 1\right\}^2$, and $n_2 - n_1$ design points are added to this design, or it will be based on a Latin hypercube design of n_2 points, i.e. all design points are chosen on the n_2-grid, with grid points $\left\{0, \frac{1}{n_2-1}, \frac{2}{n_2-1}, \ldots, 1\right\}^2$. Next, both types of grids are discussed. Furthermore, examples are provided for the case of $n_1 = 6$ and $n_2 = 13$ points. To enable comparison with the

non-nested case, the individual maximin Latin hypercube designs of $n_1 = 6$ and $n_2 = 13$ points (see Section 3.4) are depicted in Figures 8.3 and 8.4, respectively.

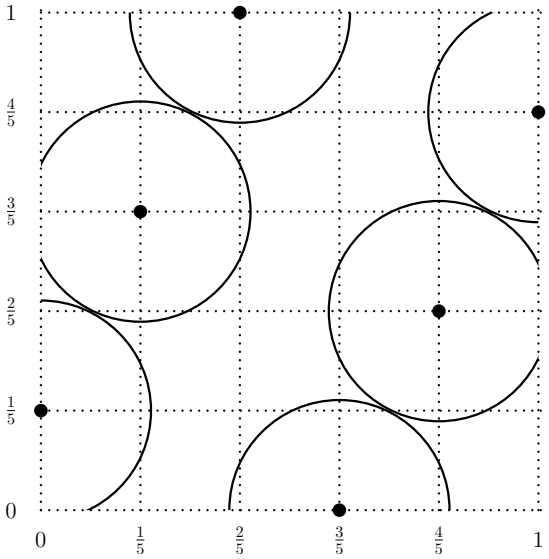


Figure 8.3: A maximin Latin hypercube design of 6 points; $d_1 = 1.0000$.

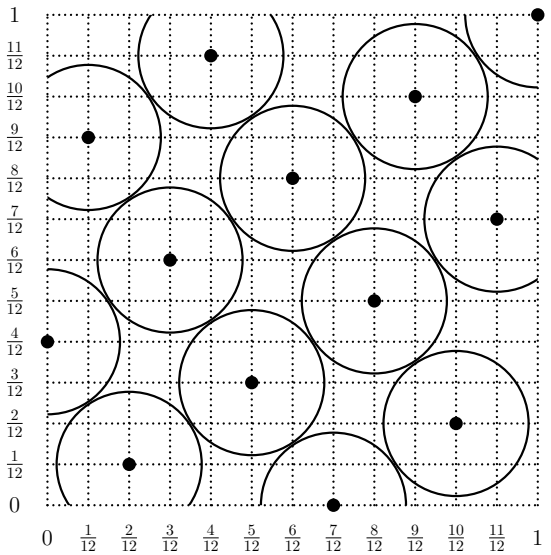


Figure 8.4: A maximin Latin hypercube design of 13 points; $d_2 = 1.0408$.

The n_2-grid

To construct a nested Latin hypercube design on an n_2-grid, the n_1 points that will form the design X_1 have to be chosen on the grid; the $n_2 - n_1$ extra points that, together with the design points of X_1, will form the design X_2 have to be added. Consider the following two measures: the space-fillingness of each design, represented by the distances d_j, and the non-collapsingness of each design, with respect to the projections of the design points onto the axes.

A space-filling nested design is obtained by selecting design points that maximize the minimal scaled distances between these points in the two underlying designs X_1 and X_2.

Let the term *X_j-coordinates* denote the levels obtained when projecting the design points of design X_j onto one of the axes (or dimensions), for $j = 1, 2$. Note that the use of an n_2-grid already yields a non-collapsing design X_2 since the X_2-coordinates are equidistantly distributed. Hence, what remains is to add restrictions that lead to a space-filling 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 $\left\{0, \frac{1}{n_1-1}, \frac{2}{n_1-1}, \ldots, 1\right\}^2$. See, for example, the nested maximin Latin hypercube design of $n_1 = 16$ and $n_2 = 31$ points (with $c_2 = 2$) depicted in Figure 8.5. Using an extension of the branch-and-bound algorithm of Chapter 3, we have

been able to obtain nested maximin Latin hypercube designs for n_2 up to 32 points, in case $c_2 \in \mathbb{N}$. Section 8.6 provides the corresponding maximin distances.

For the case $c_2 \notin \mathbb{N}$ the situation is more complicated. Since we are bound to the n_2-grid, and $n_1 - 1$ is no longer a divisor of $n_2 - 1$, it is no longer 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 the proof of Proposition 7.1. Hence, should the design collapse onto one dimension, having chosen the design points of X_1 such that its coordinates satisfy the above restriction will result in an optimal one-dimensional nested maximin design. Therefore, the X_1-coordinates are required 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. An example of a nested maximin design on an n_2-grid of $n_1 = 6$ and $n_2 = 13$ points, with $d = d_2 = 0.9129$ and $d_1 = 1.0035$, is depicted in Figure 8.6. The results obtained with the extended branch-and-bound algorithm, for $n_2 \leq 15$, are provided in Section 8.6.

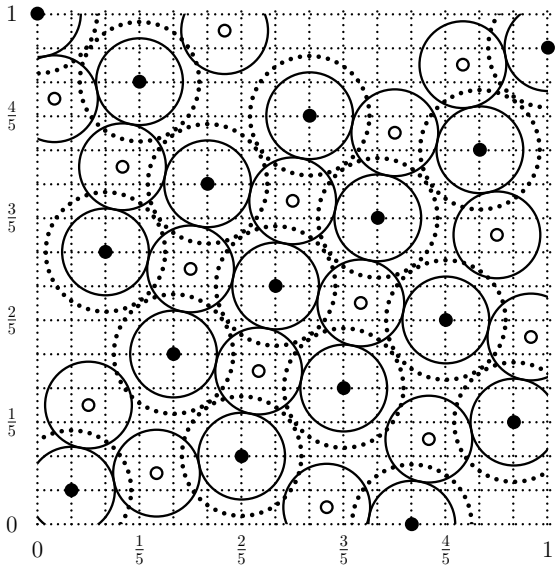


Figure 8.5: An optimal nested maximin Latin hypercube design of $n_1 = 16$ and $n_2 = 31$ points; $d = d_1 = d_2 = 0.9309$.

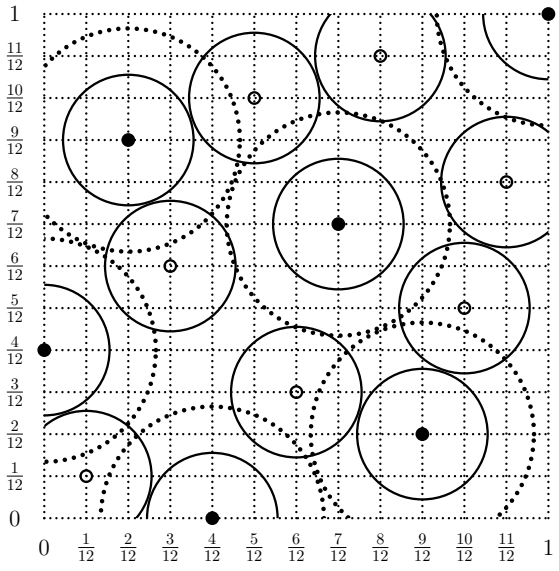


Figure 8.6: A nested maximin n_2-Latin hypercube design of $n_1 = 6$ and $n_2 = 13$ points; $d = d_2 = 0.9129$ and $d_1 = 1.0035$.

The n_1-grid

The idea here is the same as with the 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 will be separated by either $\frac{1}{\lfloor c_2 \rfloor} \frac{1}{n_1-1}$ or $\frac{1}{\lceil c_2 \rceil} \frac{1}{n_1-1}$. See Figure 8.7 for an example of a nested maximin design on an n_1-grid of

$n_1 = 6$ and $n_2 = 13$ points, with $d = d_2 = 0.9522$ and $d_1 = 1.0000$. More results, for n_2 up to 15 points, can again be found in Section 8.6.

8.3 Grids 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 an n_1-grid is used; design X_2 is preferred over design X_1 in case of an 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 the (known) one-dimensional nested maximin design of n_1 and n_2 points; see Section 7.2. 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 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 8.8 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 $d_2 = 0.9805$. Section 8.6 provides the maximin distances for values of n_2 up to 15.

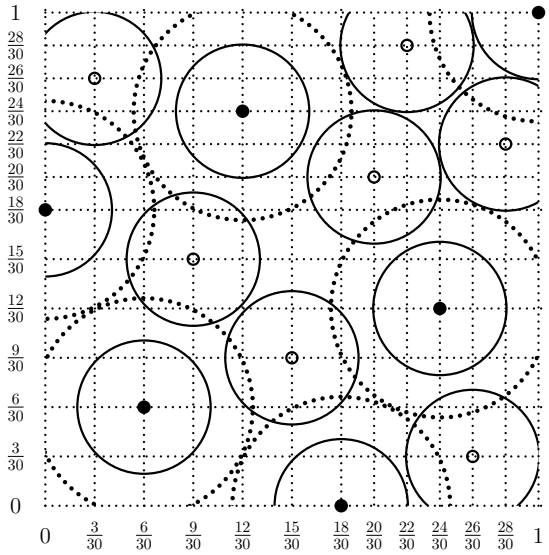


Figure 8.7: A nested maximin n_1-Latin hypercube design of $n_1 = 6$ and $n_2 = 13$ points; $d = d_2 = 0.9522$ and $d_1 = 1.0000$.

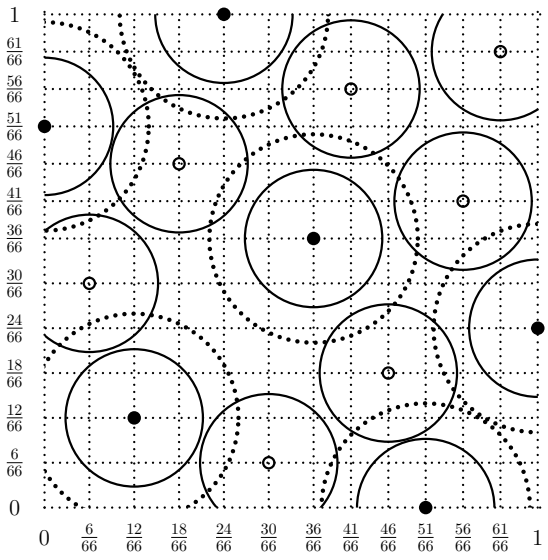


Figure 8.8: A nested maximin design of $n_1 = 6$ and $n_2 = 13$ points on a grid with nested maximin axes; $d = d_1 = 0.9589$ and $d_2 = 0.9805$.

8.4 Comparing the different types of grids

Sections 8.2 and 8.3 introduce three types of grids for constructing non-collapsing nested maximin designs. The question that arises is when to use which type? As an example, consider Table 8.1, which summarizes the results found for the case of $n_1 = 6$ and $n_2 = 13$ points.

Grid type	d	d_1	d_2	Figure
n_2-grid	0.9129	1.0035	0.9129	8.6
n_1-grid	0.9522	1.0000	0.9522	8.7
Grid with nested maximin axes	0.9589	0.9589	0.9805	8.8

Table 8.1: Maximin distances for different types of nested grid-designs of $n_1 = 6$ and $n_2 = 13$ points.

When determining which type of grid to use, there are a few aspects to consider.

First, if we are interested in the space-fillingness of a design, then the grid that yields the largest maximin distance should be chosen, i.e. the grid with nested maximin axes in Table 8.1. Note, however, that the maximin distance does not depend only on the grid used, but also on the values of n_1 and n_2. Therefore, it may be wise to compare several different pairs (n_1, n_2) for each type of grid in order to find a satisfying nested maximin design. When it is not known a priori which design parameters are significant, the non-collapsingness criterion should also be considered. If the design collapses then the one-dimensional design should preferably be space-filling, which is accomplished by choosing a grid with nested maximin axes. Hence, in case of $n_1 = 6$ and $n_2 = 13$ points, the nested maximin design depicted in Figure 8.8 is a good choice, with respect to both space-fillingness and non-collapsingness.

Besides the space-fillingness and the non-collapsingness criteria, the reason why a nested design is used may also affect the choice for a particular grid. For example, an n_1-grid is preferable for sequential evaluations, since it is known for sure that the first set of design points is evaluated (moreover, these evaluation points should yield information about the whole feasible region, and, hence, should be equally spread over this region), whereas the evaluation of an extra set of design points may depend on the previously evaluated set. In the same setting, an n_2-grid is preferable 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, i.e. X_1 or X_2, is considered to be the most important one, and, thus, using an n_1-grid or an n_2-grid, respectively. This latter question is of particular importance in cases where a training set and a test set are used. Since the prediction accuracy of a metamodel is, among others, affected by the choice of the evaluation points, 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 may be preferred. A grid with nested maximin axes should be used when there is no explicit preference for either one of the designs.

From this discussion it follows that the notion of the "best" nested grid-design depends on the application under consideration and the users' preferences. Fortunately, there are some special cases that make the comparison of the various nested grid-designs superfluous, i.e. when $c_2 \in \mathbb{N}$. In these cases we do not have to differentiate between different grid types, since they will all yield the same nested maximin design (and maximin distance).

8.5 Dominance

Section 7.2.2 introduces the notion of dominance. Dominant combinations can also be identified for two-dimensional nested maximin Latin hypercube designs. For $c_2 \in \mathbb{N}$ and $n \leq 32$ we have been able to compute all these dominant combinations. Furthermore, all optimal (i.e. maximin) combinations that are provided in Table 8.3 turned out to be dominant. Table 8.2 provides all dominant 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.

n_1	n_2	Dominant 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 8.2: All pairs (n_1, n_2) with more than one dominant combination; $c_2 \in \mathbb{N}$, $n_2 \leq 32$.

In this table the first entry corresponds to the optimal maximin combination (d_1, d_2), followed by the other dominant combination(s). Note that in case of $n_1 = 11$ and $n_2 = 21$

points there exist two different optimal designs, 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 dominant designs are also equal (0.7906 and 0.7454, respectively); however, the individual maximal distances of the second dominant combination are smaller than the maximal distances of the (optimal) first combination ($1.0607 < 1.1180$ and $1.0000 < 1.0541$, respectively). Figures 8.9 and 8.10 depict the other two dominant nested maximin Latin hypercube designs of $n_1 = 16$ and $n_2 = 31$ points; the optimal nested maximin Latin hypercube design for this case is depicted in Figure 8.5.

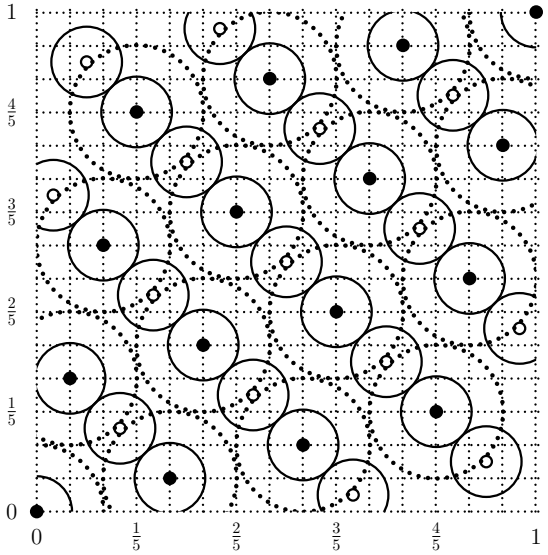


Figure 8.9: A dominant nested maximin Latin hypercube design of $n_1 = 16$ and $n_2 = 31$ points; $d = d_2 = 0.7746$ and $d_1 = 1.0646$.

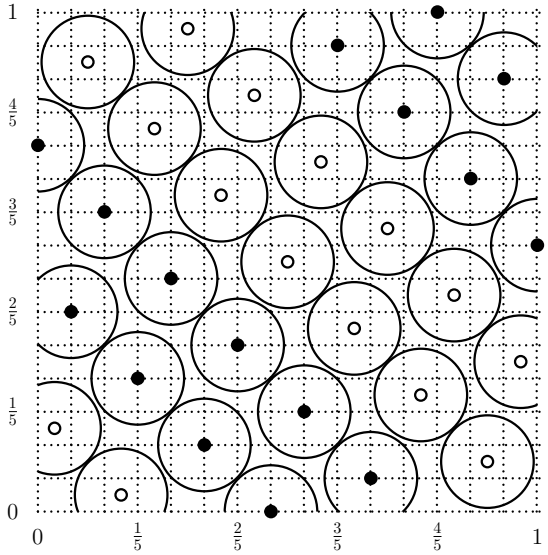


Figure 8.10: A dominant nested maximin Latin hypercube design of $n_1 = 16$ and $n_2 = 31$ points; $d = d_1 = 0.7303$ and $d_2 = 1.0328$.

8.6 Computational results

Table 8.3 provides the maximin distances for nested maximin Latin hypercube designs when $n_1 - 1$ is a divisor of $n_2 - 1$, i.e. $c_2 \in \mathbb{N}$, and for $n_2 \leq 32$. For n_2 up to 15, and $c_2 \notin \mathbb{N}$, Table 8.4 provides the maximin distances for non-collapsing nested maximin grid-designs in case of an n_1-grid, an n_2-grid, and a grid with nested maximin axes.

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

Table 8.3: Maximin distances for nested maximin Latin hypercube designs; $c_2 \in \mathbb{N}$.

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

Table 8.4: Maximin distances for nested maximin grid-designs; $c_2 \notin \mathbb{N}$.

Chapter 9

Conclusions and further research

> If you realize you aren't so wise today as you thought you were yesterday, you're wiser today.
>
> (Olin Miller)

9.1 Summary and conclusions

Decision processes are nowadays often facilitated by simulation tools. In the field of engineering, for example, such tools are used to simulate the behavior of products (and processes). Due to the nature of these simulation tools they are referred to as black-box functions. Since the evaluation of a black-box function, i.e. the simulation of a particular product design, may take a lot of computation time, the unknown function is often replaced by approximation models (or metamodels). These metamodels give a lot of insight into the underlying (and unknown) black-box function. Furthermore, since metamodels are explicit functions, function evaluations are relatively fast, so mathematical programming techniques can be applied in the search for a good product design.

Chapter 1 discusses the Metamodel approach, as described by Den Hertog and Stehouwer (2002), which provides an efficient way to construct metamodels. An important step in this approach is to set up a proper design of computer experiments, i.e. to determine which scenarios are to be evaluated. Since metamodels are fitted on the data obtained by evaluating these scenarios, the design of computer experiments has a direct effect on the accuracy of the metamodels.

The rest of this thesis is divided into two parts. We start the first part by giving an overview of the literature in the field of design of computer experiments in Chapter 2. Several design criteria, such as maximin, minimax, uniformity, Audze-Eglais, integrated mean squared error, and maximum entropy, are discussed in that chapter. Furthermore,

several different types of designs are identified, such as non-collapsing designs, sequential designs, and nested designs. Since we focus on deterministic computer simulations, a proper design of computer experiments should at least be space-filling and non-collapsing, in some sense. The main focus in the first part of this thesis therefore lies on the construction of non-collapsing maximin designs, and in particular maximin Latin hypercube designs (LHDs). Furthermore, since we assume that all design parameters in a design of computer experiments are equally important, all box constraints (i.e. lower and upper bounds) on these design parameters can (and must) be scaled to equally sized intervals. As a result, the (approximate) maximin Latin hypercube designs obtained in this thesis are not bound to a particular problem instance, but can be used in various applications (that is, after proper re-scaling). To facilitate the general use of these designs (and the nested designs obtained in Part II) an online database of (nested) maximin designs is maintained at the website `http://www.spacefillingdesigns.nl`.

Chapter 3 considers two-dimensional maximin Latin hypercube designs, which are very useful in the approximation and optimization of black-box functions. In that chapter we derive general formulas for the maximin distance and provide explicit construction methods to obtain the corresponding maximin Latin hypercube designs for the maximum norm and the rectangular distance measure. Baer (1992) and both Fejes Tóth (1971) and Florian (1989) have already (partly) solved the two-dimensional unrestricted maximin problem for these two distance measures, respectively. Comparing their results with the maximin distances that we obtain, we see that the difference in the maximin distances is less than two, and, hence, the relative difference tends to zero. For the Euclidean distance measure we obtain maximin Latin hypercube designs for up to 70 points using a branch-and-bound algorithm. Inspired by the periodicity present in many of these optimal designs, we develop a periodic construction method and obtain approximate maximin Latin hypercube designs for up to 1000 points. This heuristic method is able to reproduce all the optimal distances found by the branch-and-bound algorithm. The best-known maximin distances of the circle packing problem, i.e. the unrestricted equivalent to our LHD-problem, are provided on the Packomania website of Specht (2005). Comparing our obtained results with these best-known maximin distances, we see that the reduction in the maximin distance caused by imposing the Latin hypercube structure is small. Hence, for all three distance measures the use of Latin hypercube designs instead of unrestricted designs is justified.

Chapter 4 considers the more general case of maximin Latin hypercube designs for up to ten dimensions. In that chapter we extend the periodic construction method of Chapter 3 and obtain approximate maximin Latin hypercube designs of up to 100 points. Furthermore, we also present a simulated annealing algorithm to construct high-dimensional approximate maximin Latin hypercube designs. This latter algorithm works

particularly well in case the number of design points is small, whereas the periodic construction method performs better in case the number of design points is large. This difference can be explained by the irregularity, i.e. the lack of a nice, periodic structure, of Latin hypercube designs of a small number of points. Since (approximate) maximin Latin hypercube designs have been known only for some small numbers of points (see Morris and Mitchell (1995)) our designs present a significant extension of the previously known results.

In Chapter 5 we investigate the trade-off between the space-fillingness and the non-collapsingness of designs for computer experiments. Requiring the coordinates of the design points to be separated by at least some distance α, we obtain quasi non-collapsing designs. When varying the value of α, we find that sometimes highly non-collapsing designs can be constructed without even reducing the space-fillingness (i.e. the maximin distance). This behavior is illustrated by several examples for the maximum norm, the rectangular distance, and the Euclidean distance measure.

In the second part of this thesis the focus lies on product design problems in which multiple black-box functions play a role, and on non-collapsing nested designs.

Chapter 6 considers multi-component product design problems. Such problems occur, for example, in the automobile and aerospace industry, where products consist of many components, each of them represented by their own black-box functions. Since interdependencies often exist among the various components, a proper coordination of the design process is vitally important. We propose a collaborative extension of the Metamodel approach, called the Collaborative Metamodel approach, which acts as a framework for dealing with the aforementioned type of design problems. Furthermore, we introduce three different coordination methods to facilitate the control over the inter-relations among the black-box functions. We distinguish between sequential coordination methods, in which information obtained at a particular black-box function is passed on to other functions, and parallel simulation, in which all black-box functions are considered concurrently. Several aspects of these coordination methods are discussed and compared in order to give recommendations on when to use which method. We show that the main reason for using a sequential coordination method is the presence of strong relations among the black-box functions, whereas a parallel simulation method is preferable when these relations are less pronounced. For the corresponding throughput time, i.e. the total time needed for all simulations, of each coordination method we derive general formulas. We find that the throughput times of sequential coordination methods are always longer than the time needed by a parallel simulation method. The differences in the throughput times for a particular problem instance can be quantified using our formulas.

Another important step in the Collaborative Metamodel approach is the construction of nested designs. In case of multiple black-box functions such designs are useful when the

functions have one or more design parameters in common. In case of a single black-box function, nested designs can either be considered as sequential designs or can be applied as training and test sets in the process of fitting and validating metamodels.

In Chapter 7 we introduce one-dimensional nested maximin designs. In that chapter we derive general formulas for the maximin distance and provide explicit construction methods to obtain the corresponding nested maximin designs for the case of nesting two designs. Furthermore, we obtain nested approximate maximin designs, for the case of nesting three and four designs, by means of a heuristic construction method. We conjecture that this heuristic is optimal when nesting three designs; this claim is supported by the results obtained. Moreover, we show that the loss in space-fillingness, with respect to traditional maximin designs, is relatively small. This justifies the use of nested maximin designs instead of traditional maximin designs.

Chapter 8 considers the extension to two-dimensional non-collapsing nested maximin designs. We obtain such nested designs for up to 15 points (and some larger values), for the case of nesting two designs. Non-collapsingness is met by requiring the design points to lie on a grid. We show that the choice for a particular grid depends on the underlying design problem, and in particular on the question which of the two designs (within the nested design) is considered to be more important. Furthermore, we give some recommendations on when to choose which type of grid.

9.2 Directions for further research

This section proposes several topics for further research:

- Chapter 3 derives general construction methods for two-dimensional maximin Latin hypercube designs for the maximum norm and for the rectangular distance measure. It would be very useful to also have such construction methods for general k-dimensional Latin hypercube designs. Furthermore, lower and upper bounds for the maximin distances of these designs would provide useful information on the performance of the approximate maximin Latin hypercube designs obtained in Chapter 4. In an initial study, Van Dam, Den Hertog, Husslage, and Rennen (2006) provide a construction method to obtain maximin Latin hypercube designs of $n = m^k$ points for the maximum norm. For other values of n and other distance measures, however, none of such explicit constructions exist yet. That study also provides several upper bounds for three-dimensional maximin Latin hypercube designs for different distance measures.

- Chapters 3 and 4 propose a periodic construction method to obtain (approximate) maximin Latin hypercube designs. This method, however, is not limited to the

maximin criterion and could be applied to construct other types of Latin hypercube designs, such as minimax, uniform, Audze-Eglais, integrated mean squared error, or maximum entropy. For example, initial tests on minimizing the potential energy (see (2.6)) of (adapted) periodic Latin hypercube designs show promising results. The Audze-Eglais Latin hypercube designs obtained in this way either exhibit a lower potential energy than the (best-known) designs found by Bates et al. (2004), or fewer iterations are needed to obtain designs with a potential energy that closely approximates the best-known values.

- The first part of this thesis considers the construction of Latin hypercube designs for box-constrained design spaces. In practice, however, the need for a design of computer experiments on a non-box feasible region may arise. Stehouwer and Den Hertog (1999), for example, introduce a refined-grid method, in combination with a simulated annealing algorithm, to obtain non-collapsing designs on arbitrarily shaped feasible regions. Using such a refined grid, it may be possible to generalize the periodic construction method presented in Chapters 3 and 4 to the case of non-box feasible regions.

- Chapter 5 considers quasi non-collapsing designs and the corresponding trade-off between the space-fillingness and the non-collapsingness of a design. For a small number of points, and for the rectangular distance, the maximum norm, and the Euclidean distance measure, this trade-off is often non-trivial (as is illustrated by Figures 5.2, 5.4, and 5.5, respectively). An interesting research topic is to determine how the trade-off behaves for a larger number of points for these different distance measures.

- Kusiak and Park (1990) discuss a methodology to reduce the product-design makespan by decomposing and grouping design activities. For a multi-component product the architecture of the product defines the decomposition of the product into multiple black boxes and their relations, which results in the corresponding black-box chain. An initial study by Rennen (2003) considers the clustering (or grouping) of black boxes within this chain for Parallel Simulation and Sequential Modeling. That study shows that the clustering of black boxes for these two coordination methods results in a significant reduction of the throughput time. A disadvantage of clustering, however, remains the amount of time involved to compute the optimal clusters, and, hence, more research in this area is needed.

- As noted in Section 6.3.3, the invalidity of system-level metamodels leads to the choice between fitting different (types of) models on the one hand, or evaluating additional sets of scenarios on the other hand. In this latter case, nested designs

can be used to determine which extra design points should be evaluated in order to improve the accuracy of the system-level metamodels. What remains is to decide which of the components to grant extra evaluations. This problem, however, is very complex and further research is needed in this area.

- Chapter 8 constructs non-collapsing nested maximin designs in two dimensions, using a branch-and-bound algorithm. In its search for the best nested design the algorithm consumes a lot of CPU-time, in particular when the number of design points is large. Furthermore, when extending the algorithm to three or more dimensions the search space (i.e. all possible combinations of design points) increases exponentially with the dimension of the nested design. Since nested designs can be used as training and test sets, they play an important role in the process of fitting and validating metamodels. Hence, the development of a heuristic construction method to obtain nested designs for a larger number of design points and in higher dimensions would be very useful.

Bibliography

Aarts, E.H.L. and J.K. Lenstra (1997). *Local search in combinatorial optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester: John Wiley & Sons.

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. and R.M. Lewis (1999). Comparative properties of collaborative optimization and other approaches to MDO. *ICASE Report 99-24*. NASA Langley Research Center, Hampton, VA, United States.

Alexandrov, N.M. and R.M. Lewis (2000). Analytical and computational aspects of collaborative optimization. *Proceedings of the Eight AIAA / USAF / NASA / ISSMO Symposium on Multidisciplinary Analysis and Optimization*. Long Beach, CA, United States.

Assine, A., D. Falkenburg, and K. Chelst (1999). Engineering design management: An information structure approach. *International Journal of Product Research*, **37**, 2957–2975.

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. *Wiss. Z. Pädagog. Hochsch. Erfurt/Mühlhausen, Math.-Naturwiss. Reihe*, **28**, 87–92.

Barton, R.R. (1998). Simulation metamodels. *Proceedings of the 1998 Winter Simulation Conference*, 167–176. Washington, DC, United States.

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 D.S. Langley (2003). Formulation of the Audze-Eglais

Uniform Latin Hypercube design of experiments. *Advances in Engineering Software*, **34**, 493–506.

Bates, S.J., J. Sienz, and V.V. Toropov (2004). Formulation of the optimal Latin hypercube design of experiments using a permutation genetic algorithm. *AIAA 2004-2011*, 1–7.

Bazaara, M.S., J.J. Jarvis, and H.D. Sherali (1990). *Linear programming and network flows*. New York: John Wiley & Sons.

Beers, W.C.M. van and J.P.C. Kleijnen (2005). Customized sequential designs for random simulation experiments: Kriging metamodeling and bootstrapping. *CentER Discussion Paper 2005-55*. Tilburg University, Tilburg, The Netherlands.

Birge, J.R. and K.G. Murty (1994). *Mathematical programming: State of the art*. Ann Arbor, MI: University of Michigan Press.

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 Optimization*, **17(1)**, 1–13.

Braun, R.D. and I.M. Kroo (1995). Development and application of the collaborative optimization architecture in a multidisciplinary design environment. In *Multidisciplinary design optimization: State of the art*. SIAM Publications.

Brekelmans, R., L.T. Driessen, H. Hamers, and D. den Hertog (2005). Constrained optimization involving expensive function evaluations: A sequential approach. *European Journal of Operational Research*, **160**, 121–138.

Bursztyn, D. and D.M. Steinberg (2006). Comparison of designs for computer experiments. *Journal of Statistical Planning and Inference*, **136(3)**, 1103–1119.

Casado, L.G., I. Garcia, P.G. Szabó, and T. Csendes (2001). Packing equal circles in a square II – New results for up to 100 circles using the TAMSASS-PECS algorithm. In: F. Giannessi et al. (eds.): *Optimization Theory*, 207–224. Dordrecht: Kluwer Academic Publishers.

Conn, A.R., K. Scheinberg, and P.L. Toint (1997). Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, **79(1–3)**, 397–414.

Cramer, E.J., J.E. Dennis Jr., P.D. Frank, R.M. Lewis, and G.R. Shubin (1994). Problem formulation for multidisciplinary optimization. *SIAM Journal on Optimization*, **4**, 754–776.

Crary, S.B. (2001). WebDOE^{TM}. `http://www.webdoe.cc`.

Crary, S.B. (2002). Design of computer experiments for metamodel generation. *Analog Integrated Circuits and Signal Processing*, **32(1)**, 7–16.

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^{TM}. *Computer Modeling in Engineering & Sciences*, **1(1)**, 127–139.

Dam, E.R. van (2005). Two-dimensional minimax Latin hypercube designs. *CentER Discussion Paper 2005-105*. Tilburg University, Tilburg, The Netherlands.

Dam, E.R. van, D. den Hertog, B.G.M. Husslage, and G. Rennen (2006). Maximin Latin hypercube designs. *Working paper*. Tilburg University, Tilburg, The Netherlands.

Dam, E.R. van, B.G.M. Husslage, and D. den Hertog (2004). One-dimensional nested maximin designs. *CentER Discussion Paper 2004-66*. Tilburg University, Tilburg, The Netherlands.

Dam, E.R. van, B.G.M. Husslage, D. den Hertog, and J.B.M. Melissen (2006). Maximin Latin hypercube designs in two dimensions. *Operations Research*. To appear.

Daskin, M.S. (1995). *Network and discrete location: Models, algorithms, and applications*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester: John Wiley & Sons.

DeMiguel, A.-V. and W. Murray (2000). An analysis of collaborative optimization methods. *Proceedings of the Eight AIAA / USAF / NASA / ISSMO Symposium on Multidisciplinary Analysis and Optimization*. Long Beach, CA, United States.

Dimnaku, A., R. Kincaid, and M.W. Trosset (2005). Approximate solutions of continuous dispersion problems. *Annals of Operations Research*, **136(1)**, 65–80.

Driessen, L.T. (2006). *Simulation-based optimization for product and process design*. Ph. D. thesis, 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. *Proceedings of the European Conference on Spacecraft, Structures, Materials & Mechanical Testing*. Toulouse, France.

Drud, A.S. (1994). CONOPT – A large scale GRG code. *ORSA Journal of Computing*, **6**, 207–216.

Erkut, E. (1990). The discrete p-dispersion problem. *European Journal of Operational Research*, **46(1)**, 48–60.

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., C.X. Ma, D. Maringer, Y. Tang, and P. Winker (1999). The uniform design. `http://www.math.hkbu.edu.hk/UniformDesign/`, September 1999.

Fang, K.T., C.X. Ma, and P. Winker (2002). Centered L_2-discrepancy of random sampling and Latin hypercube design, and construction of uniform designs. *Mathematics of Computation*, **71**, 275–296.

Farhangmehr, A. (2003). *Entropy approach to meta-modeling, multi-objective genetic algorithm, and quality assessment of solution sets for design optimization*. Ph. D. thesis, University of Maryland, College Park, MD, United States.

Fejes Tóth, L. (1971). Punktverteilungen in einem Quadrat. *Studia Sci. Math. Hung.*, **6**, 439–442.

Florian, A. (1989). Verteilung von Punkten in einem Quadrat. *Sitzungsber., Abt. II, Österr. Akad. Wiss., Math.-Naturwiss. Kl.*, **198**, 27–44.

Glover, F., J.P. Kelly, and M. Laguna (1996). New advances and applications of combining simulation and optimization. *Proceedings of the 1996 Winter Simulation Conference*, 144–152. Coronado, CA, United States.

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.

Hulme, K.F. (2000). *The design of a simulation-based framework for the development of solution approaches in multidisciplinary design optimization*. Ph. D. thesis, University at Buffalo, Buffalo, NY, United States.

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*. 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*. Tilburg University, Tilburg, The Netherlands.

Jin, R., W. Chen, and A. Sudjianto (2002). On sequential sampling for global metamodeling in engineering design. *Proceedings of the ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. 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.

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., M. Schonlau, and W.J. Welch (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, **13**, 455–492.

Kirchner, K. and G. Wengerodt (1987). Die dichteste Packung von 36 Kreisen in einem Quadrat. *Beiträge Algebra Geom.*, **25**, 147–159.

Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi (1983). Optimization by simulated annealing. *Science*, **220**, 671–680.

Kleijnen, J.P.C. (1987). *Statistical tools for simulation practitioners*. New York: Marcel Dekker, Inc.

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.

Kodiyalam, S. and J. Sobieszczanski-Sobieski (2001). Multidisciplinary design optimization – Some formal methods, framework requirements, and application to vehicle design. *International Journal of Vehicle Design*, **25**, 3–22.

Koehler, J.R. and A.B. Owen (1996). *Computer experiments*, Volume 13 of *Handbook of Statistics*. New York: Elsevier Science B.V.

Krishnan, V. (1996). Managing the simultaneous execution of coupled phases in concurrent product development. *IEEE Transactions on Management*, **43(2)**, 210–217.

Kroo, I.M. and V. Manning (2000). Collaborative optimization: Status and directions. *Proceedings of the Eight AIAA / USAF / NASA / ISSMO Symposium on Multidisciplinary Analysis and Optimization*. Long Beach, CA, United States.

Kusiak, A. and K. Park (1990). Concurrent engineering: Decomposition and scheduling of design activities. *International Journal of Production Research*, **28**, 1883–1900.

Law, A.M. and W.D. Kelton (2000). *Simulation modeling and analysis*. Singapore: McGraw-Hill.

Lindley, D.V. (1956). On a measure of information provided by an experiment. *Annals of Mathematical Statistics*, **27**, 986–1005.

Markót, M.Cs. 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.

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*, **16**, 239–245.

Melissen, J.B.M. (1997). *Packing and covering with circles*. Ph. D. thesis, Utrecht University, Utrecht, The Netherlands.

Meyer, M. and P. Vlachos (1993). StatLib – Collection of orthogonal arrays, by Owen. `http://lib.stat.cmu.edu/designs/owen.html`.

Morris, M.D. and T.J. Mitchell (1995). Exploratory designs for computer experiments. *Journal of Statistical Planning and Inference*, **43**, 381–402.

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). Revolutionizing engineering science through simulation. *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.

Papalambros, P.Y. (2000). Extending the optimization paradigm in engineering design. *Proceedings of the Third International Symposium on Tools and Methods of Competitive Engineering*. Delft, The Netherlands.

Papalambros, P.Y. (2001). Analytical target cascading in product development. *Proceedings to the Third ASMO UK / ISSMO Conference on Engineering Design Optimization*, 3–16. Harrogate, United Kingdom.

Park, J.-S. (1994). Optimal Latin-hypercube designs for computer experiments. *Journal of Statistical Planning and Inference*, **39(1)**, 95–111.

Peikert, R., D. Würtz, M. Monagan, and C. den Groot (1991). Packing circles in a sphere: A review and new results. *Proceedings of the 15th IFIP Conference on System Modeling and Optimization, Springer Lecture Notes in Control and Information Sciences*, **180**, 111–124.

Pfoertner, H. (2005). Collection of the densest packings of equal spheres in a cube. `http://www.pfoertner.org`, June 2005.

Pintér, J.D. (1995). *LGO: A model development and solver system for continuous global optimization*. Halifax, Nova Scotia, Canada: Pintér Consulting Services Inc.

Powell, M.J.D. (2000). UOBYQA: Unconstrained optimization by quadratic approximation. *Mathematical Programming*, **92(3)**, 555–582.

Rennen, G. (2003). Optimal grouping of black boxes to reduce the throughput time. *Unpublished note*. Tilburg University, Tilburg, The Netherlands.

Rikards, R. and J. Auzins (2004). Response surface method for solution of structural identification problems. *Inverse Problems in Engineering*, **12(1)**, 59–70.

Roos, C., T. Terlaky, and J.-P. Vial (1997). *Theory and algorithms for linear optimization: An interior point approach*. Somerset, New Jersey: John Wiley & Sons.

Sacks, J., S.B. Schiller, and W.J. Welch (1989). Designs for computer experiments. *Technometrics*, **31**, 41–47.

Santner, Th.J., B.J. Williams, and W.I. Notz (2003). *The design and analysis of computer experiments*. Springer Series in Statistics. New York: Springer-Verlag.

Shannon, C.E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, **27**, 379–423.

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.

Sobieszczanski-Sobieski, J. and R.T. Haftka (1997). Multidisciplinary aerospace design optimization: Survey of recent developments. *Structural and Multidisciplinary Optimization*, **14(1)**, 1–23.

Specht, E. (2005). Packomania. `http://www.packomania.com`, July 2005.

Stehouwer, H.P. and D. den Hertog (1999). Simulation-based design optimisation: Methodology and applications. *Proceedings of the First ASMO UK / ISSMO Conference on Engineering Design Optimization*. Ilkley, United Kingdom.

Stein, M. (1987). Large sample properties of simulations using Latin hypercube sampling. *Technometrics*, **29(2)**, 143–151.

Steinberg, D.M. and D.K.J. Lin (2006). A construction method for orthogonal Latin hypercube designs. *Biometrika*, **93(2)**, 279–288.

Stinstra, E.D. (2006). *The meta-model approach for simulation-based design optimization*. Ph. D. thesis, Tilburg University, Tilburg, The Netherlands.

Stinstra, E.D. and D. den Hertog (2005). Robust design using computer experiments. *CentER Discussion Paper 2005-90*. 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.

Stinstra, E.D., H.P. Stehouwer, and J. van der Heijden (2003). Collaborative tube design optimization: An integral meta-modeling approach. *Proceedings of the Fifth ISSMO Conference on Engineering Design Optimization*. Como, Italy.

Sunset Software Technology (2003). XA binary and mixed integer solver. `http://www.sunsetsoft.com`.

Tang, B. (1993). Orthogonal array-based Latin hypercubes. *Journal of the American Statistical Association*, **88**, 1392–1397.

Tang, D., L. Zheng, Z. Li, D. Li, and S. Zhang (2000). Re-engineering of the design process for concurrent engineering. *Computers and Industrial Engineering*, **38**, 479–491.

Toropov, V.V., A.A. Filatov, and A.A. Polynkin (1993). Multiparameter structural optimization using FEM and multipoint explicit approximations. *Structural Optimization*, **6**, 7–14.

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.

Author index

Subject index

Bold page numbers refer to definitions.

Optimale schema's voor computerexperimenten

Samenvatting

Simulatieprogramma's worden tegenwoordig veel gebruikt in beslissingsprocessen. In het gebied van de engineering worden zulke programma's bijvoorbeeld gebruikt om het gedrag van producten en processen te simuleren. Dit simuleren kan erg tijdrovend zijn, vandaar dat simulatiemodellen vaak vervangen worden door benaderende functies. Deze functies geven veel inzicht in de onderliggende complexe modellen. Verder kunnen evaluaties van deze expliciete functies snel uitgevoerd worden, waardoor het vinden van een goed ontwerp, middels het gebruik van optimaliseringstechnieken, binnen handbereik komt te liggen.

Hoofdstuk 1 van dit proefschrift bespreekt een algemeen raamwerk om benaderende functies op te stellen voor complexe simulatiemodellen. Een belangrijke stap in dit raamwerk is het opstellen van een schema voor de computerexperimenten. Zo'n schema bepaalt welke scenario's gesimuleerd zullen worden en heeft daardoor een direct effect op de nauwkeurigheid van de benaderende functies.

De rest van dit proefschrift is opgesplitst in twee delen. We beginnen het eerste gedeelte met een overzicht van de literatuur op het gebied van schema's voor computerexperimenten in Hoofdstuk 2. Verschillende criteria om een goed schema op te stellen worden in dat hoofdstuk besproken. Verder behandelen we verscheidene typen schema's, zoals sequentiële en gekoppelde schema's. In de praktijk is gebleken dat, in het geval van deterministische computersimulaties, een schema in ieder geval hoort te voldoen aan de volgende twee eisen: het schema dient de gehele ontwerpruimte te overdekken en de gesimuleerde scenario's mogen elkaar niet (geheel of gedeeltelijk) overlappen. De klasse van de zogenaamde *maximin Latin hypercube* schema's voldoet aan beide eisen en de constructie van zulke schema's vormt daarom dan ook het onderwerp van de eerste hoofdstukken.

In dit proefschrift beschouwen we alle ontwerpparameters in het ontwerpprobleem als even belangrijk. Door de parameters te schalen maken we de geconstrueerde schema's

onafhankelijk van een bepaalde probleeminstantie, zodat deze schema's (na herschaling) gebruikt kunnen worden in verschillende applicaties. Om het algemeen gebruik van deze optimale schema's (en de gekoppelde schema's verkregen in het tweede deel van dit proefschrift) te vergemakkelijken wordt er een online database van optimale schema's bijgehouden op de website `http://www.spacefillingdesigns.nl`.

Hoofdstuk 3 beschouwt tweedimensionale maximin Latin hypercube schema's. In dat hoofdstuk leiden we algemene formules af voor de *maximin* afstand en geven expliciete constructiemethoden voor het opstellen van de bijbehorende maximin Latin hypercube schema's voor verschillende afstandsmaten. Voor de meest gebruikte afstandsmaat, de Euclidische afstand, vinden we maximin Latin hypercube schema's, bestaande uit maximaal 70 scenario's, door het gebruik van een *branch-and-bound* algoritme. Geïnspireerd door de periodiciteit die voorkomt in veel van deze optimale schema's ontwikkelen we een periodieke constructiemethode die goede schema's oplevert tot 1000 scenario's. Deze heuristiek vindt ook de optimale schema's die gevonden worden door het branch-and-bound algoritme. Verder laten we zien dat door het toevoegen van de Latin hypercube structuur de maximin afstand niet teveel afneemt voor de verschillende afstandsmaten. Deze observatie rechtvaardigt het gebruik van maximin Latin hypercube schema's in plaats van traditionele schema's voor computerexperimenten.

Hoofdstuk 4 bespreekt het algemenere geval van meerdimensionale maximin Latin hypercube schema's. In dat hoofdstuk breiden we de periodieke constructiemethode van Hoofdstuk 3 uit en verkrijgen op deze manier schema's tot 100 scenarios. We presenteren ook een *simulated annealing* algoritme om goede schema's tot in tien dimensies op te stellen. Dit laatste algoritme blijkt erg goed te werken wanneer het aantal scenario's relatief klein is, terwijl de periodieke constructiemethode goed werkt voor een groter aantal scenario's. Dit verschijnsel kan verklaard worden door de onregelmatigheid, dat is het gebrek aan een mooie, periodieke structuur, van Latin hypercube schema's bestaande uit een klein aantal scenario's. Aangezien maximin Latin hypercube schema's eerder alleen bekend waren voor een klein aantal scenario's leveren onze gevonden schema's een significante uitbreiding op van de tot nu toe bekende resultaten.

In Hoofdstuk 5 onderzoeken we de wisselwerking tussen de eis dat een schema de ontwerpruimte dient te overdekken en de eis dat de gesimuleerde scenario's elkaar niet mogen overlappen. We bekijken deze wisselwerking voor verschillende afstandsmaten en we laten zien dat de relatie tussen de twee eisen interessant, niet-triviaal gedrag kan vertonen. In sommige gevallen is het bijvoorbeeld mogelijk om niet-overlappende scenario's te kiezen, zonder dat dit effect heeft op de mate waarin het schema de ontwerpruimte overdekt.

In het tweede gedeelte van dit proefschrift richten we ons op de problemen die optreden bij

het ontwerp van multi-component producten. Zulke problemen komen bijvoorbeeld voor in de auto- en de luchtvaartindustrie, waar producten uit veel verschillende componenten bestaan. Tussen deze componenten is er vaak enige samenhang aanwezig, waardoor een juiste coördinatie binnen het ontwerpproces onontbeerlijk wordt.

In Hoofdstuk 6 presenteren we een raamwerk om bovengenoemde ontwerpproblemen efficiënt op te lossen. Belangrijke stappen in onze aanpak zijn de constructie en het gebruik van coördinatiemethoden en van gekoppelde schema's om de relaties tussen de verschillende componenten onder controle te houden. We introduceren drie verschillende coördinatiemethoden, die we onderscheiden in parallelle en sequentiële methoden. Bij dit laatste type geven de ontwerpers van de verschillende componenten informatie door aan elkaar, terwijl bij parallelle methoden alle componenten afzonderlijk beschouwd worden. Verscheidene aspecten van de drie coördinatiemethoden worden besproken en met elkaar vergeleken. Vanuit deze analyse geven we aanbevelingen betreffende de beste keus voor één van de coördinatiemethoden voor bepaalde probleeminstanties. Zo zijn sequentiële methoden bijvoorbeeld te prefereren wanneer er een sterke samenhang bestaat tussen de componenten, terwijl een parallelle coördinatiemethode beter werkt wanneer deze samenhang minder duidelijk aanwezig is. Een belangrijk kwantitatief aspect is de doorlooptijd, ofwel de totale tijd die nodig is voor alle simulaties. We leiden algemene formules af voor deze doorlooptijd en laten zien dat sequentiële coördinatiemethoden een kortere doorlooptijd opleveren dan een parallele methode.

De constructie van gekoppelde schema's voor computerexperimenten vormt het onderwerp van de volgende twee hoofdstukken. In het geval van multi-component producten zijn zulke gekoppelde schema's nuttig wanneer er componenten zijn die één of meerdere ontwerpparameters gemeenschappelijk hebben. In het geval van een enkele component kunnen gekoppelde schema's gebruikt worden als sequentiële schema's voor computerexperimenten of kunnen ze worden toegepast als training- en testschema's bij het opstellen van benaderende functies.

In Hoofdstuk 7 introduceren we eendimensionale gekoppelde maximin schema's. In dat hoofdstuk leiden we algemene formules af voor de maximin afstand en geven we constructiemethoden voor de bijbehorende schema's, in het geval dat er twee schema's gekoppeld worden. Voor de gevallen waarin we drie of vier schema's koppelen presenteren we een heuristische constructiemethode. We vermoeden dat deze heuristiek optimaal is in het geval van drie gekoppelde schema's; dit vermoeden wordt onderbouwd door de gevonden resultaten. Verder laten we zien dat gekoppelde maximin schema's nog steeds goed de gehele ontwerpruimte overdekken. Deze observatie rechtvaardigt het gebruik van gekoppelde maximin schema's in plaats van traditionele schema's voor computerexperimenten.

In Hoofdstuk 8 breiden we de ideeën van Hoofdstuk 7 uit en construeren we tweedi-

mensionale gekoppelde maximin schema's, in het geval dat er twee schema's gekoppeld worden. De restrictie dat de gesimuleerde scenario's elkaar niet mogen overlappen wordt opgevangen door de scenario's op een rooster te kiezen. We laten zien dat de keuze voor een bepaald type rooster voornamelijk afhangt van het antwoord op de vraag welke van de twee gekoppelde schema's het meest belangrijk wordt geacht. Uiteindelijk geven we nog enkele aanbevelingen betreffende de beste keus voor een bepaald type rooster.

Hoofdstuk 9 beëindigt dit proefschrift met de belangrijkste conclusies en geeft enkele aanbevelingen voor verder onderzoek.