Nested Maximin Latin Hypercube Designs

Rennen, G.; Husslage, B.G.M.; van Dam, E.R.; den Hertog, D.

Publication date:
2009

No. 2009–06

NESTED MAXIMIN LATIN HYPERCUBE DESIGNS

By Gijs Rennen, Bart Husslage, Edwin R. van Dam,
Dick den Hertog

January 2009

ISSN 0924-7815

Nested maximin Latin hypercube designs

Gijs Rennen • Bart Husslage* • Edwin R. van Dam† • Dick den Hertog

Department of Econometrics and Operations Research, Tilburg University,
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

G.Rennen@uvt.nl • Husslage@casema.nl • Edwin.vanDam@uvt.nl • D.denHertog@uvt.nl

Abstract

In the field of design of computer experiments (DoCE), Latin hypercube designs are frequently used for the approximation and optimization of black-boxes. In certain situations, we need a special type of designs consisting of two separate designs, one being a subset of the other. These nested designs can be used to deal with training and test sets, models with different levels of accuracy, linking parameters, and sequential evaluations. In this paper, we construct nested maximin Latin hypercube designs for up to ten dimensions. We show that different types of grids should be considered when constructing nested designs and discuss how to determine which grid to use for a specific application. To determine nested maximin designs for dimensions higher than two, four different variants of the ESE-algorithm of Jin et al. (2005) are introduced and compared. In the appendix, maximin distances for different numbers of points are provided; the corresponding nested maximin designs can be found on the website http://www.spacefillingdesigns.nl.

Keywords: Design of computer experiments, Latin hypercube design, linking parameter, nested designs, sequential simulation, space-filling, training and test set.
JEL Classification: C90.

1 Introduction

Latin hypercube designs are extremely useful in the approximation of black-box functions. By definition, black-box functions have no explicit description, but can be evaluated to obtain output values for specific input values. As evaluations of a black-box function often involve time-consuming computer simulations, we would like to construct an approximating model (or metamodel) based on evaluations in a (small) number of points. See, e.g., Montgomery (1984), Sacks et al. (1989), (1989), Myers (1999), Jones et al. (1998), Booker et al. (1999), Den Hertog and Stehouwer (2002), Santner et al. (2003), and Kleijnen (2008). A review of meta-modeling applications in structural optimization can be found in Barthelemy and Haftka (1993), and in multidisciplinary design optimization in Sobieszczanski-Sobieski and Haftka (1997).

We will use the term *design* to denote the set of evaluation points. As observed by many researchers, there is an important distinction between designs for computer experiments and designs for the more traditional response surface methods. Physical experiments exhibit random errors whereas computer experiments are often deterministic (cf. Simpson et al. (2004)). This distinction is crucial and one of the main aims in the field of design of computer experiments (DoCE) is therefore to obtaining efficient designs for computer experiments.

As is recognized by several authors, a design for computer experiments should at least satisfy the following two criteria (see Johnson et al. (1990) and Morris and Mitchell (1995)). First of all, the design should be *space-filling* in some sense. When no details on the functional behavior of the response parameters are available, it is important to be able to obtain information from the entire design space. Therefore, design points should be "evenly spread" over the entire region. Secondly, the design should be

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

†The research of E.R. van Dam has been made possible by a fellowship of the Royal Netherlands Academy of Arts and Sciences.

non-collapsing. When one of the design parameters has (almost) no influence on the black-box function value, two design points that differ only in this parameter will "collapse", i.e., they can be considered as the same point that is evaluated twice. As evaluation of the deterministic black-box function is often time-consuming, this is not a desirable situation. Therefore, two design points should not share any coordinate values when it is not known a priori which parameters are important. Moreover, we would like the projections of the points onto the axes to be separated as much as possible. When we consider a black-box function on a box-constrained domain, this can be accomplished by using Latin hypercube designs. A Latin hypercube design (LHD) of n points in m dimensions can be defined as an $n \times m$ matrix, were each column is a permutation of the set $\{0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\}$. The rows $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$, $i = 1, \ldots, n$, then define the n design points. Because the columns are permutations of the above set, for all of the m coordinates it holds that no two design points have the same value.

To obtain space-filling designs, the evaluation points are chosen in such a way that the separation distance (i.e., the minimal distance among any pair of points) is maximized, leading to so-called maximin designs. Other space-filling designs, like minimax, integrated mean squared error (IMSE), and maximum entropy designs, are also used in the literature. For a good survey of these designs see the book of Santner et al. (2003). In this book it is also shown that maximin Latin hypercube designs generally speaking yield good approximations.

Maximin Latin hypercube designs where first constructed by Morris and Mitchell (1995) using simulated annealing. Ye et al. (2000) only considered the class of symmetric approximate maximin LHDs to reduce the computing effort. Jin et al. (2005) introduce the enhanced stochastic evolutionary (ESE) algorithm for finding various space-filling designs, among which approximate maximin LHDs. In Husslage et al. (2008), the ESE-algorithm is also used to construct approximate maximin LHDs for up to 10 dimensions and up to 300 design points. Furthermore, they also construct approximate maximin LHDs by optimizing the maximin criterion over all LHDs having a certain periodic structure. This approach is an extension of the method used in Van Dam et al. (2007) to obtain two-dimensional approximate maximin LHDs. In that paper, two-dimensional maximin LHDs are also found using a branch-and-bound algorithm. Finally, Grosso et al. (2008) use Iterated Local Search heuristics to find good approximate maximin LHDs for up to 10 dimensions. The best designs found in these papers are published on-line at `http://www.spacefillingdesigns.nl` where they can be downloaded for free. This website also contains the upper bounds on the separation distance for certain classes of maximin LHDs found by Van Dam et al. (2009). These upper bounds can be used to asses the quality of approximate maximin LHDs.

In real-life, there are situations where we need a special type of designs called *nested designs*. This type of design consists of two separate designs, with the requirement that one design is a subset of the other design. Van Dam et al. (2004) show how to construct one-dimensional nested maximin designs; the current paper focuses on two and higher dimensional designs*. Four main reasons for nesting maximin designs are: *training and test sets*, *models with different levels of accuracy*, *linking parameters*, and *sequential evaluations*.

To start with the first, consider the problem of fitting and validating a particular metamodel. In practice, the following approach is often used. First, a metamodel is fitted to the obtained training data, i.e., the responses 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 to the response values predicted by the metamodel. If the differences between the predicted and the actual response values are small the metamodel is considered to be valid. See also Cherkassky and Mulier (1998) for a more detailed description of the use of training and test sets. Because a metamodel should be a global approximation model, i.e., it should be valid for the entire feasible region, 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. This can be accomplished by nesting two designs, which are optimized with respect to, for example, the maximin criterion. The evaluation points that are part of both designs form the training set, the remaining points make up the test set.

The second motivation comes from the situation where an output variable of a process, product, or system is modeled by two black-box functions with different levels of accuracy. These black-box functions could, for instance, be simulation models with different levels of detail. As a more accurate model is in

*This paper is a revision and extension of Husslage et al. (2005).

general also more time-consuming, we can perform fewer evaluations of the high accuracy model than of the low accuracy model in the same amount of time. Instead of choosing to use either the high or low accuracy model, we can also choose to use both. In Qian et al. (2006), a method is introduced which combines the results from the high and low accuracy models. This way, they try to obtain a model which is as accurate as possible, given the resources available. The high accuracy model is evaluated at all points in the small design and the low accuracy model at all points in the large design. By using a nested design, high and low accuracy evaluations are performed at all points in the small design. In Qian et al. (2006), the differences in these points are used to adjust and improve the metamodel fitted to the low accuracy evaluations.

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

As an example of a real-life problem in which linking parameters play a role, we consider a collaborative optimization approach to optimize the design of a color picture tube, see Stinstra et al. (2003). Such a tube consists of the main components screen, electron gun, and shadow mask. Stinstra et al. (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. The Landing function measures the quality of the image, whereas the Microphony function measures how vulnerable the shadow mask is to external vibrations. Because the response parameters of both Landing and Microphony depend on the settings of the design parameters of the shadow mask, linking parameters play an important role, see Figure 1. As is argued by Husslage et al. (2003), the same settings should be used for these linking parameters as much as possible, giving rise to the need for nested designs.

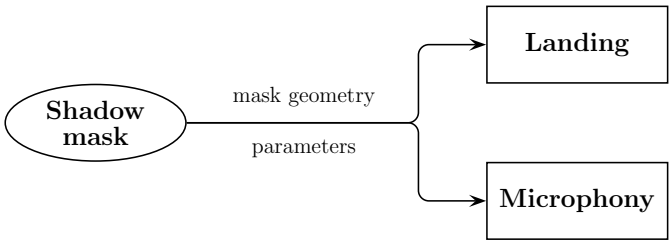

Figure 1: Linking parameters in tube design optimization.

Nested designs are also useful when dealing with sequential evaluations. In practice it is common that after evaluating an initial set of points, extra evaluations are needed. As an example, suppose we construct an approximating model for some black-box function based on n_1 function evaluations. However, after validating the obtained model it turns out that an extra set of function evaluations is needed to build a proper model. We then face the problem of constructing a design on a total of, say, n_2 points, given the initial design on n_1 points. To anticipate the possibility of extra evaluations, one can construct the two designs (on n_1 and n_2 points) at once, hence, by constructing a nested design.

We have just described why both Latin hypercube designs and nested designs are important. In this paper we construct nested maximin Latin hypercube designs in m dimensions with $m \geq 2$. Moreover, we focus on the problem of nesting two designs, X_1 and X_2, with $X_1 \subset X_2$, $X_j = \{x_i = (x_{i1}, x_{i2}, \ldots, x_{im}) \mid i \in I_j\}$, and $|I_j| = n_j$, $j = 1, 2$. Thus, the index set $I_1 \subset I_2 = \{1, 2, \ldots, n_2\}$ defines which design points x_i are

part of both designs. The nested design is defined by the combination of X_1 and X_2.

All design parameters are scaled such that they take values in the interval $[0, 1]$, and the class of nested designs is optimized with respect to the maximin distance criterion. Furthermore, scaling factors s_1 and s_2 are introduced to enable a fair comparison of the distances between the points in the designs X_1 and the distances between the points in 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:

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

In this paper, we consider the Euclidean distance measure for $d(\cdot, \cdot)$. Because one-dimensional designs of n points have distance $1/(n-1)$ and the minimum distance of n points in an m-dimensional hypercube is at most of the order $1/\sqrt[m]{n-1}$, it seems natural to use scaling factors $s_j = 1/\sqrt[m]{n_j - 1}$, $j = 1, 2$, in (1) for m-dimensional designs. As we use the maximin distance criterion, we have to maximize the minimal scaled distance between any pair of points in X_1 and X_2. Therefore, what remains is to maximize the minimal distance $d = \min\{d_1, d_2\}$ over all $I_1 \subset I_2$, with $|I_1| = n_1$, and $x_i \in [0, 1]^2$.

We are aware that the above formulation is just one way of combining the two separation distances into one objective and that also other scaling factors or formulations are possible. Using different scaling factors is no problem as all methods discussed in this paper can also be used for other values of s_1 and s_2. In Section 6, we will discuss some other alternative objectives. Dealing with maximizing d_1 and d_2 as a bi-objective optimization problem is one of the possibilities. For two-dimensional nested designs with small n_1 and n_2, we use this approach in Section 3.2. However, to limit the scope of this paper, our main focus will be on the above maximin objective.

Combining the objective function and the conditions for a nested design, we can formulate the following optimization problem to obtain a nested maximin design:

$$\begin{array}{lll} \max & d & \\ \text{s.t.} & I_1 \subset I_2 & \\ & |I_j| = n_j, & j = 1, 2 \\ & 0 \leq x_{ij} \leq 1, & i \in I_2, \, j = 1, 2. \end{array} \tag{2}$$

Note that these conditions do not enforce non-collapsingness, which is essential for the LHD structure. As mentioned before, the non-collapsingness is important when dealing with deterministic computer experiments. We can obtain a non-collapsing nested maximin design by adding to Problem (2) constraints that enforce the x_i-coordinates (in each dimension) to be separated (see also Van Dam et al. (2007)). These constraint can, for instance, impose that the design points x_i must be positioned on a grid with a certain structure. Three possible grid structures, *nested n_1-grids*, *nested n_2-grids* and *grids with nested maximin axes*, are proposed in Section 2. In Section 3, a branch-and-bound method for determining two dimensional nested maximin designs is presented and Pareto optimal nested designs in two dimensions are discussed. For higher dimensions, determining the nested LHD which maximizes d becomes too time consuming. Therefore, we introduce in Section 4 a heuristic which also aims to maximize d but does not guarantee to find the optimal d. In Section 5, numerical results obtained with different variants of this heuristic are presented and compared. Furthermore, we discuss how to select a grid-structure and design based on the obtained results. Finally, Section 6 contains concluding remarks.

2 Grid structures for nested Latin hypercube designs

One of the properties of a regular LHD is that all points, when projected onto one of the axes, are equidistantly distributed. To form an LHD, the points in X_1 must thus be projected onto the set $\{0, \frac{1}{n_1-1}, \frac{2}{n_1-1}, \ldots, 1\}$ and the points in X_2 onto $\{0, \frac{1}{n_2-1}, \frac{2}{n_2-1}, \ldots, 1\}$. In order for X_1 and X_2 to both form a Latin hypercube design, the first set must be a subset of the second. However, this only holds when $n_2 - 1$ is a multiple of $n_1 - 1$ or, stated differently, when $c_2 := \frac{n_2-1}{n_1-1}$ is integer. In all other cases, we have to compromise on the LHD structure of one or both designs. As there are different ways of doing this, we introduce three different grid structures. Note, however that all three grids coincide when c_2 is integer.

To illustrate these structures, examples are provided for the two-dimensional case of $n_1 = 6$ and $n_2 = 13$ points. In Figures 2 and 3 also the individual maximin Latin hypercube designs of $n_1 = 6$ and $n_2 = 13$ points are depicted to enable comparison with the non-nested case.

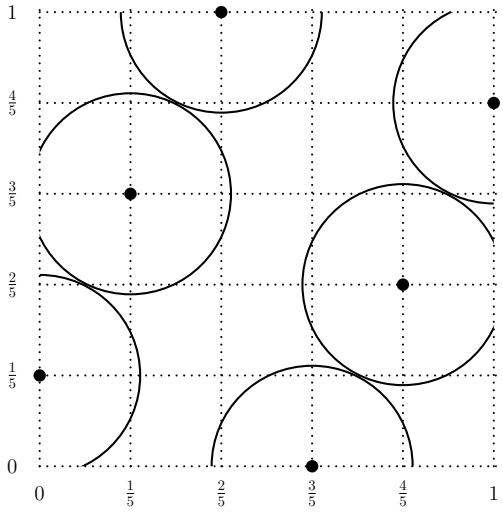

Figure 2: A maximin Latin hypercube design of 6 points; $d_1 = 1.0000$.

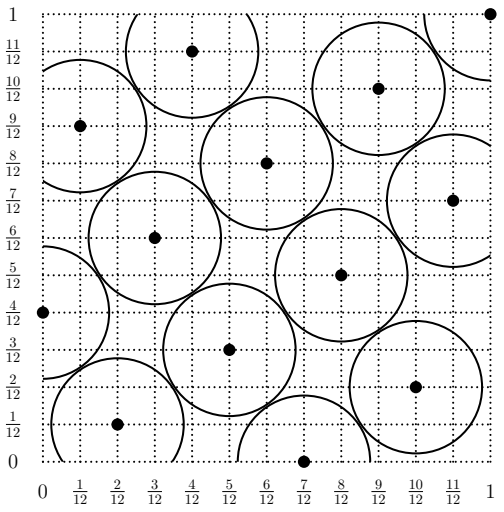

Figure 3: A maximin Latin hypercube design of 13 points; $d_2 = 1.0408$.

2.1 Nested n_2-grid

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

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

To start, consider the case where $c_2 = \frac{n_2-1}{n_1-1} \in \mathbb{N}$. In this case, a non-collapsing design X_1 is obtained by limiting the choice of design points (of X_1) to the set of equidistantly distributed X_1-coordinates $\{0, \frac{1}{n_1-1}, \frac{2}{n_1-1}, \ldots, 1\}^m$. See, for example, the two-dimensional nested maximin Latin hypercube design of $n_1 = 16$ and $n_2 = 31$ points (with $c_2 = 2$) depicted in Figure 4.

For the case $c_2 \notin \mathbb{N}$, the situation is more complicated. Because we are bound to the n_2-grid, and $n_1 - 1$ is no longer a divisor of $n_2 - 1$, it is 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 Van Dam et al. (2004). Therefore, the X_1-coordinates are required to be separated by either $\lfloor c_2 \rfloor \frac{1}{n_2-1}$ or $\lceil c_2 \rceil \frac{1}{n_2-1}$. Hence, should the design then collapse (onto one of the axes), then it collapses onto an optimal one-dimensional nested maximin design.

Note that this restriction still leaves multiple grids possible for design X_1 when $c_2 \notin \mathbb{N}$. An example of a nested maximin design on a nested n_2-grid of $n_1 = 6$ and $n_2 = 13$ points, with $d = d_2 = 0.9129$ and $d_1 = 1.0035$, is depicted in Figure 5. 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.

For the nested n_2-grid, a very simple method to determine nested LHDs would seem to take an existing LHD of n_2-points for X_2 and select a subset of n_1 points for X_1. Although this method is quite attractive

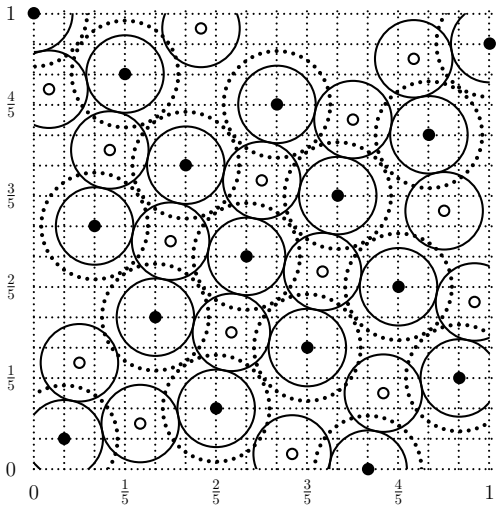

Figure 4: A nested maximin Latin hypercube design of $n_1 = 16$ and $n_2 = 31$ points; $d = d_1 = d_2 = 0.9309$.

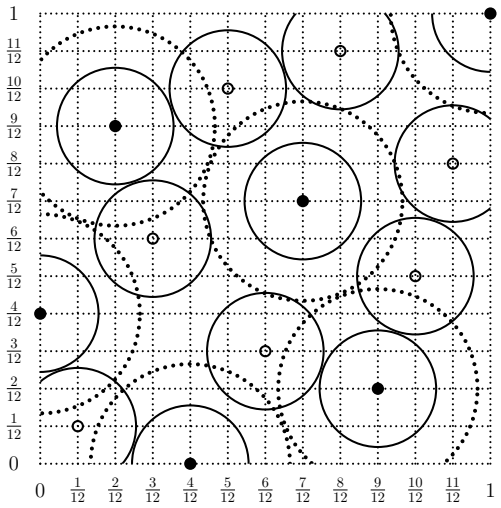

Figure 5: A nested maximin n_2-Latin hypercube design of $n_1 = 6$ and $n_2 = 13$ points; $d = d_2 = 0.9129$ and $d_1 = 1.0035$.

because of its simplicity, it does not generally yield a nested LHD satisfying all the restrictions of the nested n_2-grid. We will illustrate this with the example in Figure 6. The figure shows a maximin Latin hypercube design for $n = 15$ obtained in Van Dam et al. (2007). Assume we want to construct a nested LHD with $n_1 = 8$ and $n_2 = 15$. Because in this case $c_2 = 2$, the nested n_2-grid is unique and both the X_1- and X_2-coordinates must be equidistantly distributed for both dimensions. The solid dots represent X_1 when we satisfy this latter restriction for the dimension on the horizontal axis. We can easily see that the distribution of the X_1-coordinates on the other axis is certainly not equidistantly. This probleem also occurs for many other Latin hypercube designs and is even more likely to occur when the number of dimensions increases. Therefore, we will not use this method to construct nested LHDs, but use the methods described in Sections 3.1 and 4.1.

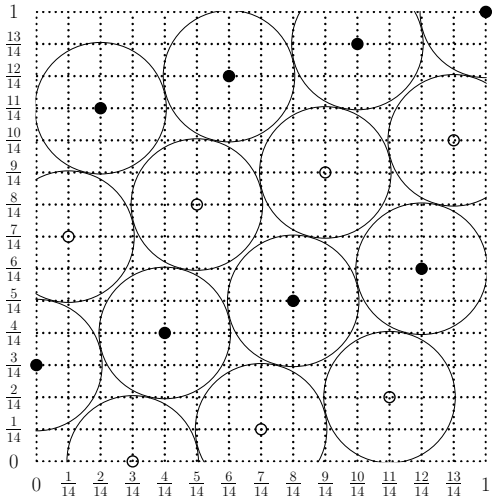

Figure 6: Example of the problem occurring when taking X_1 equal to a subset of an existing LHD.

2.2 Nested n_1-grid

When we want X_1 to be an LHD instead of X_2, we can use the nested n_1-grid. The design X_1 is then obtained by choosing n_1 design points on the n_1-grid, with grid points $\{0, \frac{1}{n_1-1}, \frac{2}{n_1-1}, \ldots, 1\}^m$. The

additional X_2-coordinates are placed equidistantly between the X_1-coordinates. Similar to the nested n_2-grid, the (interiors of the) intervals formed by consecutive X_1-coordinates are again required to contain either $\lfloor c_2 \rfloor - 1$ or $\lceil c_2 \rceil - 1$ X_2-coordinates. Hence, consecutive X_2-coordinates will be separated by either $\frac{1}{\lfloor c_2 \rfloor}\frac{1}{n_1-1}$ or $\frac{1}{\lceil c_2 \rceil}\frac{1}{n_1-1}$. Again, this leaves multiple grids possible when $c_2 \notin \mathbb{N}$. See Figure 7 for an example of a nested maximin design on a nested n_1-grid of $n_1 = 6$ and $n_2 = 13$ points, with $d = d_2 = 0.9522$ and $d_1 = 1.0000$.

2.3 Grid with nested maximin axes

The use of the Latin hypercube structure in the construction of a nested maximin design implies a preference of one design over the other. Design X_1 is assumed to be more important than design X_2 when a nested n_1-grid is used; design X_2 is preferred over design X_1 in case of a nested n_2-grid. If both sets are assumed to be of equal importance we would like to treat them equally. To deal with this problem, the X_1- and X_2-coordinates could be restricted to take only values at the levels of a (known) one-dimensional nested maximin design of n_1 and n_2 points; see Van Dam et al. (2004). The design points of X_1 and X_2 could then be chosen from the grid points obtained in this way. Note that in this case the projections of the design points onto the axes are always optimally space-filling with respect to the maximin distance criterion. Furthermore, note that a one-dimensional maximin design, with $c_2 \notin \mathbb{N}$, is (again) not unique, so there are multiple grids possible. Figure 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 $d_2 = 0.9805$.

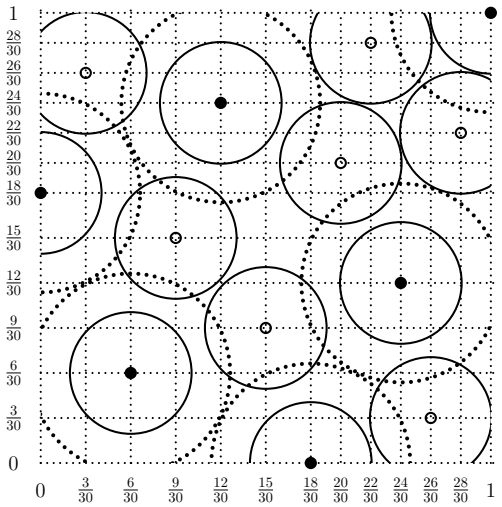

Figure 7: A nested maximin n_1-Latin hypercube design of $n_1 = 6$ and $n_2 = 13$ points; $d = d_2 = 0.9522$ and $d_1 = 1.0000$.

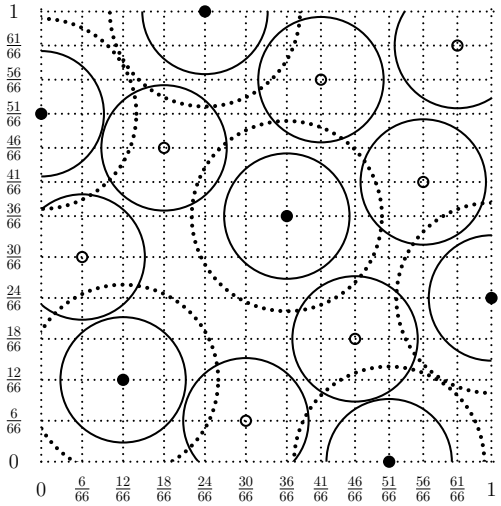

Figure 8: A nested maximin design of $n_1 = 6$ and $n_2 = 13$ points on a grid with nested maximin axes; $d = d_1 = 0.9589$ and $d_2 = 0.9805$.

3 Two-dimensional nested designs

3.1 Branch-and-bound

To obtain two-dimensional nested maximin LHDs, we use an extension of the branch-and-bound algorithm of Van Dam et al. (2007). This extended branch-and-bound algorithm works as follows. Given n_1, n_2 and the grid structure, first all possible nested grids are determined and the possible distances that can occur for X_1 and X_2 are calculated. These distances form a discrete set that can be efficiently searched and optimized. To determine whether a nested LHD exists with X_1 and X_2 having separation distances at least d_1 and d_2, respectively, a branch-and-bound search is performed for each possible nested grid. This branch-and-bound method is similar to the one used for usual LHDs as described in Van Dam et al. (2007), however in general the nested grid structures do not allow for the refinements given there. In the search tree, a node at level t corresponds to a partial nested design consisting of t design points

$(x_1, \ldots, x_t)$, where the first n_1 points are in X_1 and the points are, furthermore, ordered by increasing first coordinate. Nodes in the tree are pruned when they correspond to partial nested designs that are collapsing or that have separation distances smaller than d_1 or d_2.

Using the extended branch-and-bound algorithm, we obtained results for n_2 up to 15 for all three grid structures. For the cases where $c_2 \in \mathbb{N}$, the algorithm is refined so that nested maximin LHDs could be obtained up to $n_2 = 32$. The maximin distances of these designs can be found in Tables 6 and 7 in the appendix. In Section 5, the results are compared and discussed.

3.2 Pareto nested designs

Besides nested designs that maximize the objective function $d = \min\{d_1, d_2\}$, there are also some other interesting nested designs to consider: *Pareto nested designs*. We will call a combination of distances (d_1, d_2) Pareto optimal (or Pareto) if it is not possible to improve one of the distances, without deteriorating the other distance. A Pareto nested design is a nested design of which the distances (d_1, d_2) form a Pareto combination. For $c_2 \in \mathbb{N}$ and $n_2 \leq 32$, we have found all the Pareto combinations using a slightly adjusted version of the branch-and-bound algorithm. Furthermore, the original branch-and-bound algorithm already made sure that the distances (d_1, d_2) of all nested maximin designs provided in Table 6 (see the appendix) are Pareto optimal. Table 1 provides all Pareto combinations (d_1, d_2) corresponding to the pairs (n_1, n_2), with $c_2 \in \mathbb{N}$ and $n_2 \leq 32$, for which there exist more than one such combination.

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

Table 1: All pairs (n_1, n_2) with more than one Pareto 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 Pareto combination(s). Note that in case of $n_1 = 11$ and $n_2 = 21$ points there exist two different Pareto combinations, both with a maximin distance equal to $d = 0.7071$. For the (n_1, n_2) pairs $(9, 17)$ and $(10, 19)$, the objective values of the Pareto nested designs are also equal (0.7906 and 0.7454, respectively); however, the individual maximal distances of the second Pareto combination are smaller than the maximal distances of the (optimal) first combination ($1.0607 < 1.1180$ and $1.0000 < 1.0541$, respectively).

4 Higher dimensional nested designs

4.1 Enhanced stochastic evolutionary algorithm

For dimensions higher than two and for larger values of n_1 and n_2, using the above branch-and-bound algorithm to find nested LHDs which maximize d becomes too time-consuming. In these cases, we can use heuristics to find nested approximate maximin LHDs, where approximate indicates that optimality is not guaranteed. One possible heuristic is the ESE-algorithm of Jin et al. (2005). In Husslage et al. (2008), this algorithm obtains good results for approximate maximin LHDs. Although this algorithm

was originally designed for non-nested designs, with some changes it is also applicable to nested designs. Before we look at these changes, we first give a short description of the original ESE-algorithm. This description is based on the description given in Husslage et al. (2008).

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

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

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

To use the ESE-algorithm for nested designs, the step which needs to be changed most is the generation of new designs. When one point is selected from X_1 and the other from $X_2 \backslash X_1$, exchanging the k^{th} coordinate value can distort the nested grid structure.

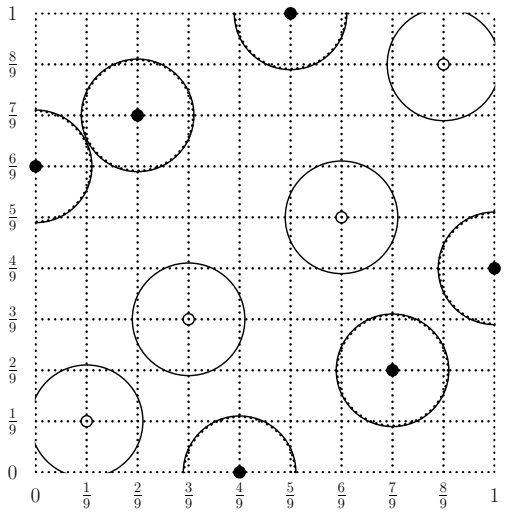

Figure 9: A nested Latin hypercube design of $n_1 = 6$ and $n_2 = 10$ points; $d = d_1 = 0.3086$ and $d_2 = 0.5556$.

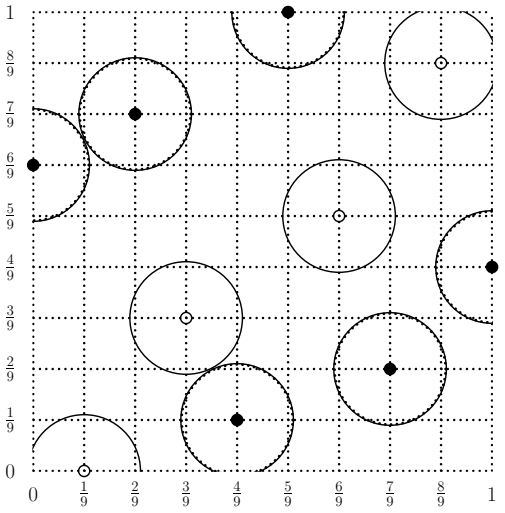

Figure 10: A nested design obtained by exchanging one coordinate value between a point in X_1 and one point in $X_2 \backslash X_1$.

Figures 9 and 10 give an example where this indeed occurs. The design in Figure 10 is obtained by exchanging one coordinate value of two points in the lower left part of Figure 9. As we require in each dimension that the first and last point should be in X_1, the new design is not a valid nested design. We could try to repair this by changing the assignment of the points to the sets X_1 and X_2. However, there exists no assignment such that the invalid nested design in Figure 10 becomes a valid nested LHD on a nested n_2-grid.

Besides solving the above problem, a new method for generating designs can also take into account the different objective function. As we consider the minimum of d_1 and d_2, we can, for instance, use different

methods depending on which of the two values is smallest. In the next section, we discuss some different methods of generating new designs which take the above two aspects into account.

4.2 Generating new designs

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

We can also take into account that the grids are not unique when c_2 is non-integer. For instance, when $n_1 = 6$ and $n_2 = 13$, it can be verified that there are 21 different two-dimensional nested n_2-grids after accounting for reflection and rotational symmetry. In cases like these, the choice of a specific grid can affect the maximal attainable value of d. Therefore, we consider different methods of selecting the grid of the initial design. Furthermore, we also look at methods which can change a grid without it becoming distorted.

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

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

To illustrate the different methods, we will again use the design in Figure 9. To simplify notation, we will use the terms DMIN-, RAND-, POINT- or GROUP-method to refer to any of the two methods whose name contain these words, e.g. POINTDMIN and GROUPDMIN are both DMIN-methods. As $d_1 < d_2$, a DMIN-method would exchange a coordinate value of two points in X_1. In the ESE-algorithm, a fixed number of designs is generated and the best is selected for comparison against the current design. We could for instance obtain the design in Figure 11. This design is obtained by exchanging one coordinate value between the points with coordinates $(0, \frac{6}{9})$ and $(1, \frac{4}{9})$ in the current design of Figure 9. Looking at the d-values, we see that this design is an improvement and will thus be selected by the ESE-algorithm to become the new current design. Let us next consider a GROUP-method and take k equal to 2, i.e. the dimension on the vertical axis. The two differently shaded areas in Figure 11 now form two possible groups. Notice that the top group is empty as there are no X_2-points between the X_1-points. In Figure

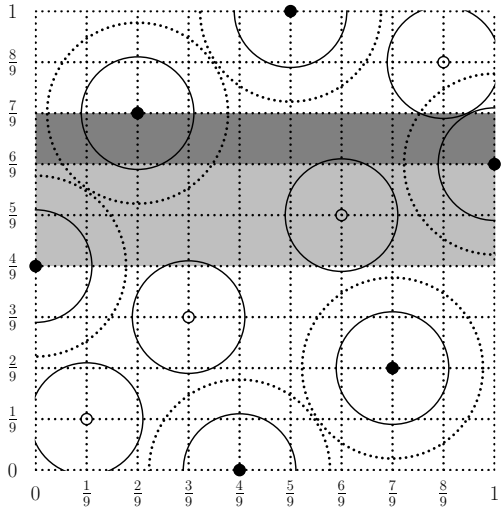

Figure 11: A nested Latin hypercube design of $n_1 = 6$ and $n_2 = 10$ points; $d = d_2 = 0.5556$ and $d_1 = 0.8025$.

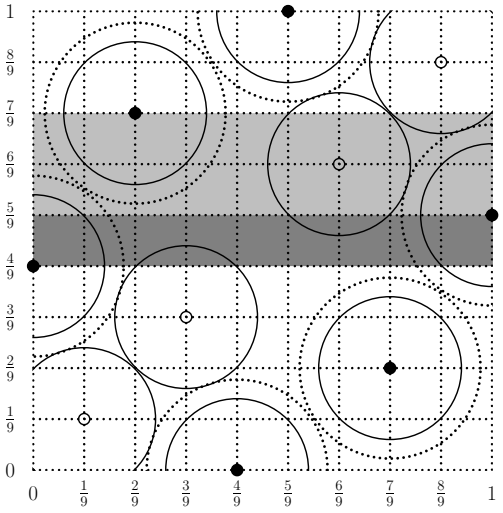

Figure 12: A nested maximin Latin hypercube design of $n_1 = 6$ and $n_2 = 10$ points; $d = d_1 = 0.8025$ and $d_1 = 0.8889$.

12, we see the result of exchanging the two groups. Because the groups are of different size, the grid has now changed. Again the design has improved and in this case the design is even optimal.

5 Numerical results

To compare the four different variants of the ESE-algorithm, we generated three- and four-dimensional nested designs with $n_1 = 5, 10, \ldots, 50$ and $n_2 = n_1 + 5, \ldots, 55, 60$ for each of the three grid structures. We thus consider 65 different pairs (n_1, n_2) for each dimension and grid structure. As the grid is not unique when $c_2 \notin \mathbb{N}$, which grid we select might affect the space-fillingness of the final design. For each combination of n_1, n_2, dimension, and grid structure, we therefore ran the ESE-algorithm ten times with a different grid and initial design. These computations have been performed on PCs with a 2.8-GHz Pentium D processor and the variants were implemented in Matlab R2007a. Per variant, grid and dimension, it took between 2 and 4 days to perform the ten runs of the ESE-algorithm for all 65 pairs.

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

Using the best results of the ten runs of the ESE-algorithm, we determine for each (n_1, n_2)-pair which method(s) obtained a best design. In Tables 8 and 9 of the appendix, the separation distances of the best generated designs are given for each of the three grid-structures. The percentage of the 65 (n_1, n_2)-pairs for which a certain method performs best are presented in Table 2. Note that the sum of the percentages per row is larger than 100%, because for some cases, a best design is found by multiple variants of the ESE-algorithm. Due to the same reason, we can not take the sum of two columns to determine the combined performance of two methods. When we study the results, we see that the two RAND-methods find the best design for most cases. One reason for the relative poor performance of the DMIN-methods could be that the number of neighbor designs is smaller. With neighbor designs, we mean all designs which can be obtained by making one possible change to the current design. When $d_1 < d_2$, a DMIN-method produces less neighbor designs than a RAND-method, because the DMIN-methods allow fewer possible changes. This can make it more difficult for a DMIN-method to escape from a local minimum, which could result in a worse performance. Of the two RAND-methods, GROUPRAND performs the best for most cases. This indicates that the possibility to change the grid indeed improves the performance of the ESE-algorithm. Based on these results, we decided to use both RAND-methods to obtain nested

	RAND		DMIN	
Three-dimensional	**GROUP**	**POINT**	**GROUP**	**POINT**
Nested n_1-grid	60	37	12	17
Nested n_2-grid	63	34	17	17
Grid with nested maximin axes	55	23	6	17
Four-dimensional				
Nested n_1-grid	66	26	2	11
Nested n_2-grid	57	29	6	15
Grid with nested maximin axes	52	23	9	15

Table 2: Percentage of the (n_1, n_2)-pairs for which a certain variant of the ESE-algorithm finds a best design.

approximate maximin designs for dimensions five up to ten. For dimension ten, calculating all 65 pairs took approximately 8 days per grid and variant. All generated nested designs can be found on the website `http://www.spacefillingdesigns.nl`.

	$\mathbf{l_1(n_1, n_2)}$		$\mathbf{l_2(n_1, n_2)}$	
Two-dimensional	**Average**	**Range**	**Average**	**Range**
Nested n_1-grid	4.13	[0.00,36.75]	11.12	[-5.41,52.56]
Nested n_2-grid	3.10	[-33.33,36.75]	8.92	[0.00,36.75]
Grid with nested maximin axes	3.83	[-14.29,36.75]	10.08	[-11.61,45.79]
Three-dimensional				
Nested n_1-grid	11.00	[0.00,17.60]	7.23	[-1.07,15.81]
Nested n_2-grid	11.14	[3.45,19.60]	7.90	[3.03,12.92]
Grid with nested maximin axes	11.38	[0.19,17.63]	8.02	[2.88,13.69]
Four-dimensional				
Nested n_1-grid	8.90	[0.00,17.54]	4.01	[-2.82,10.78]
Nested n_2-grid	9.89	[-0.16,16.82]	5.63	[2.11,11.21]
Grid with nested maximin axes	9.95	[-1.37,17.03]	5.87	[2.22,11.63]

Table 3: Average and range of percentage loss $l_j(n_1, n_2)$ caused by using nested (approximate) maximin designs instead of (approximate) maximin LHDs.

Furthermore, we are interested to see how much we lose in terms of space-fillingness by using nested instead of non-nested designs. To determine this, we compare the d_1- and d_2-values of the best nested design to the scaled separation distances of the (approximate) maximin LHDs of the same size. For a pair (n_1, n_2), we denote the first distances by $d_1(n_1, n_2)$ and $d_2(n_1, n_2)$ and the latter distances by $d(n_1)$ and $d(n_2)$. For $d(n_1)$ and $d(n_2)$, we use the best known (approximate) maximin LHDs available on `http://www.spacefillingdesigns.nl` (December 2008). We now define the percentage loss in separation distance as $l_j(n_1, n_2) := (d_j(n_1, n_2) - d(n_j))/d(n_j)$ for $j = 1, 2$. Table 3 represents the averages and the ranges of these percentage losses over all evaluated (n_1, n_2)-pairs. Note that for two dimensions, we evaluated different pairs than for the other dimensions. When we consider the two-dimensional results, we see that the n_2-grid on average gives the best space-fillingness for both designs X_1 and X_2. For the higher dimensions, the averages and ranges are more similar, but the nested n_1-grid performs slightly better on both d_1 and d_2. These results are a bit surprising as we expected the nested n_2-grid to perform better on d_2 and the nested n_1-grid to perform better on d_1. Another observation which might be surprising at first sight is that some ranges also contain negative values. This means that for some (n_1, n_2)-pairs, the d_1- or d_2-distance is better than the distance of the corresponding (approximate) maximin LHD. The main

explanation for this is that the designs X_1 or X_2 do not always have to satisfy the LHD structure. In some cases, this enables X_1 or X_2 to attain a larger separation distance than the (approximate) maximin LHD.

Although the above results give us some indication on which grids perform well in general, the results do not tell us for a specific pair (n_1, n_2) which grid gives the highest d, d_1 and d_2 values. In Table 4, we present the percentages of pairs (n_1, n_2), with $c_2 \notin \mathbb{N}$, for which a grid type performs best on a particular distance. We do not consider the pairs with $c_2 \in \mathbb{N}$ because for these pairs all grids are equal.

	Percentage best designs		
Two-dimensional	**d**	$\mathbf{d_1}$	$\mathbf{d_2}$
Nested n_1-grid	17	36	16
Nested n_2-grid	67	56	72
Grid with nested maximin axes	16	11	13
Three-dimensional			
Nested n_1-grid	52	45	53
Nested n_2-grid	19	37	21
Grid with nested maximin axes	29	18	26
Four-dimensional			
Nested n_1-grid	69	73	66
Nested n_2-grid	26	19	27
Grid with nested maximin axes	6	8	8

Table 4: Percentage of the (n_1, n_2)-pairs, with $c_2 \notin \mathbb{N}$, for which a particular grid type performs best on d, d_1, or d_2.

Not surprisingly, the grids with the lowest average loss in Table 3 also have the highest percentage of pairs for which they perform best. However, there is still a considerable percentage of pairs where one of the other two grids perform better. It thus depends on the particular pair (n_1, n_2) which grid to choose based on the separation distances. Furthermore in many situations in practice, the values of n_1 and n_2 are not fixed which leaves some freedom to change these values. In those situations, we can thus also consider nested designs where n_1 and n_2 are slightly lower or higher. Let us, for example, consider the two-dimensional designs with $n_1 = 5$ and $n_2 = 10$. In Table 5, we compare the losses of these designs to the losses of the designs with $n_1 = 6$ and $n_2 = 10$. The comparison shows that all losses either reduce or stay the same. Choosing n_1 equal to 6 instead of 5 thus seems to be a better choice in terms of space-fillingness.

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

Table 5: Example of reduction in percentage loss $l_j(n_1, n_2)$ by choosing different value for n_1.

The choice for a specific grid or (n_1, n_2)-pair does not only depend on the space-fillingness. 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.

Furthermore, the reason why a nested design is used may also affect the choice for a particular grid. For example, a nested n_1-grid or the grid with the highest d_1 could be preferable for sequential evaluations, because it is known for sure that the first set of design points is evaluated, whereas the evaluation of an extra set of design points may depend on the previously evaluated set. However in the same setting, a nested n_2-grid is 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 should, thus, have an LHD structure or have the largest separation distance.

A grid with nested maximin axes should be used when there is no explicit preference for either one of the designs. When constructing a training set and a test set, design X_1, which forms the training set, is in general the most important of the two designs. This is because the prediction accuracy of a metamodel is, among others, affected by the choice of the evaluation points in the training set. A space-filling distribution of these points over the feasible region is desirable and, hence, the grid for which the design points of X_1 have the largest separation distance may be preferred. When combining high and low accuracy models, it is hard to say which of the two designs is more important. The X_2 design is important because it is used to fit the initial model, but the X_1 design is also important as it is used to evaluate the accurate model whose results must improve the initial model.

From this discussion it follows that the notion of the "best" nested grid-design depends on the application under consideration and the user's 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 structures, because they will all yield the same nested maximin design (and maximin distance).

6 Concluding remarks

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

For dimensions higher than two, we introduced four variants of the ESE-algorithm. Using a comparison of three- and four-dimensional designs, we determined that the POINTRAND- and GROUPRAND-methods obtained the best results. Therefore, these methods are also used to obtain designs for dimensions five up to ten. Furthermore, we studied the loss in space-fillingness by using nested designs instead of non-nested designs. The results show that the nested n_2-grid in general gives the smallest losses in two dimensions and the nested n_1-grid in higher dimensions. Furthermore, we noted that we can positively influence the amount of loss by choosing slightly different values for n_1 and n_2.

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

All nested (approximate) maximin designs generated for this paper can be found on the website `http://www.spacefillingdesigns.nl`.

References

Barthelemy, J.F.M. and R.T. Haftka (1993). Approximation concepts for optimum structural design – A review. *Structural and Multidisciplinary Optimization*, **5(3)**, 129–144.

Booker, A.J., J.E. Dennis, P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset (1999). A rigorous framework for optimization of expensive functions by surrogates. *Structural and Multidisciplinary*

Optimization, **17(1)**, 1–13.

Cherkassky, V. and F. Mulier (1998). *Learning from data: Concepts, theory, and methods*. Wiley-Interscience Series on Adaptive and Learning Systems for Signal Processing, Communications, and Control. Chichester: John Wiley & Sons.

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 (2007). Maximin Latin Hypercube Designs in Two Dimensions. *Operations Research* **55(1)**, 158–169.

Dam, E.R. van, G. Rennen, and B.G.M. Husslage (2009). Bounds for Maximin Latin Hypercube Designs. *Operations Research*. (to appear).

Grosso, A., A.R.M.J.U. Jamali, and M. Locatelli (2008). Finding maximin latin hypercube designs by Iterated Local Search heuristics. *European Journal of Operational Research*, **In Press, Corrected Proof**, –.

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.

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 (2008). Space-filling Latin hypercube designs for computer experiments. *CentER Discussion Paper 2008-104*. Tilburg University, Tilburg, The Netherlands.

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.

Kleijnen, Jack P.C. (2008). *Design and Analysis of Simulation Experiments*, Volume 111 of *International Series in Operations Research & Management Science*. Springer.

Montgomery, D.C. (1984). *Design and analysis of experiments* (Second ed.). New York: John Wiley & Sons.

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

Myers, R.H. (1999). Response surface methodology – Current status and future directions. *Journal of Quality Technology*, **31**, 30–74.

Qian, Z., C.C. Seepersad, V.R. Joseph, J.K. Allen, and C.F.J. Wu (2006). Building Surrogate Models Based on Detailed and Approximate Simulations. *Journal of Mechanical Design* **128(4)**, 668–677.

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

Sacks, J., W.J. Welch, T.J. Mitchell, and H.P. Wynn (1989). Design and analysis of computer experiments. *Statistical Science*, **4**, 409–435.

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

Simpson, T.W., A.J. Booker, D. Ghosh, A.A. Giunta, P.N. Koch, and R.-J. Yang (2004). Approximation methods in multidisciplinary analysis and optimization: A panel discussion. *Structural and Multidisciplinary Optimization*, **27(5)**, 302–313.

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

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.

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.

Appendix

Table 6 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 7 provides the maximin distances for the two-dimensional nested maximin designs for all three grid structures. Tables 8 and 9 provide the separation distances of three- and four-dimensional nested approximate maximin designs with $n_1 = 5, 10, \ldots, 50$ and $n_2 = n_1 + 5, \ldots, 55, 60$ for all three grid structures. Besides these distances, all tables also contain the scaled separation distances $d(n_1)$ and $d(n_2)$ of the approximate maximin LHDs available on `http://www.spacefillingdesigns.nl` (December 2008). The nested approximate maximin designs for dimensions five up to ten can also be found on this website.

n_1	n_2	$d(n_1)$	$d(n_2)$	d	d_1	d_2
2	3	1.4142	1.0000	1.0000	1.4142	1.0000
2	4	1.4142	1.2910	0.8165	1.4142	0.8165
2	5	1.4142	1.1180	0.7071	1.4142	0.7071
3	5	1.0000	1.1180	0.7071	1.0000	0.7071
2	6	1.4142	1.0000	1.0000	1.4142	1.0000
2	7	1.4142	1.1547	0.9129	1.4142	0.9129
3	7	1.0000	1.1547	0.9129	1.0000	0.9129
4	7	1.2910	1.1547	0.8165	0.8165	1.1547
2	8	1.4142	1.0690	0.8452	1.4142	0.8452
2	9	1.4142	1.1180	1.0000	1.4142	1.0000
3	9	1.0000	1.1180	1.0000	1.0000	1.0000
5	9	1.1180	1.1180	1.1180	1.1180	1.1180
2	10	1.4142	1.0541	0.9428	1.4142	0.9428
4	10	1.2910	1.0541	0.8165	0.8165	0.9428
2	11	1.4142	1.0000	1.0000	1.4142	1.0000
3	11	1.0000	1.0000	1.0000	1.0000	1.0000
6	11	1.0000	1.0000	1.0000	1.0000	1.0000
2	12	1.4142	1.0871	0.9535	1.4142	0.9535
2	13	1.4142	1.0408	0.9129	1.4142	0.9129
3	13	1.0000	1.0408	0.9129	1.0000	0.9129
4	13	1.2910	1.0408	0.9129	1.2910	0.9129
5	13	1.1180	1.0408	0.8165	1.1180	0.8165
7	13	1.1547	1.0408	0.9129	1.1547	0.9129
2	14	1.4142	1.1435	1.0000	1.4142	1.0000
2	15	1.4142	1.1019	0.9636	1.4142	0.9636
3	15	1.0000	1.1019	0.8452	1.0000	0.8452
8	15	1.0690	1.1019	0.8452	1.0690	0.8452
2	16	1.4142	1.0646	1.0646	1.4142	1.0646
4	16	1.2910	1.0646	0.9309	1.2910	0.9309
6	16	1.0000	1.0646	0.7303	1.0000	0.7303
2	17	1.4142	1.0607	1.0308	1.4142	1.0308
3	17	1.0000	1.0607	1.0000	1.0000	1.0308
5	17	1.1180	1.0607	0.9014	1.1180	0.9014
9	17	1.1180	1.0607	0.7906	1.1180	0.7906
2	18	1.4142	1.0290	1.0000	1.4142	1.0000
2	19	1.4142	1.0000	1.0000	1.4142	1.0000
3	19	1.0000	1.0000	1.0000	1.0000	1.0000
4	19	1.2910	1.0000	0.9718	1.2910	0.9718
7	19	1.1547	1.0000	0.9718	1.1547	0.9718
10	19	1.0541	1.0000	0.7454	1.0541	0.7454
2	20	1.4142	0.9733	0.9733	1.4142	0.9733

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

Table 6: Maximin distances for two-dimensional nested maximin Latin hypercube designs; $c_2 \in \mathbb{N}$.

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

Table 7: Maximin distances for two-dimensional nested designs; $c_2 \notin \mathbb{N}$.

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

Table 8: Scaled separation distances for three-dimensional nested approximate maximin designs.

				Nested n_1-grid			**Nested n_2-grid**			**Grid with nested axes**		
n_1	n_2	$d(n_1)$	$d(n_2)$	d	d_1	d_2	d	d_1	d_2	d	d_1	d_2
5	10	1.3693	1.3608	1.2141	1.2748	1.2141	1.2472	1.3240	1.2472	1.2201	1.3279	1.2201
5	15	1.3693	1.3035	1.2069	1.2247	1.2069	1.2203	1.3325	1.2203	1.2001	1.3880	1.2001
5	20	1.3693	1.2862	1.1683	1.2247	1.1683	1.1784	1.1839	1.1784	1.1714	1.2810	1.1714
5	25	1.3693	1.2407	1.1738	1.3693	1.1738	1.1738	1.3693	1.1738	1.1738	1.3693	1.1738
5	30	1.3693	1.2241	1.1689	1.3693	1.1689	1.1706	1.2250	1.1706	1.1612	1.2926	1.1612
5	35	1.3693	1.2074	1.1735	1.2247	1.1735	1.1735	1.3323	1.1735	1.1621	1.2191	1.1621
5	40	1.3693	1.1902	1.1558	1.2247	1.1558	1.1640	1.2146	1.1640	1.1577	1.2222	1.1577
5	45	1.3693	1.1881	1.1560	1.2748	1.1560	1.1560	1.2748	1.1560	1.1560	1.2748	1.1560
5	50	1.3693	1.1830	1.1459	1.2247	1.1459	1.1492	1.3715	1.1492	1.1428	1.3652	1.1428
5	55	1.3693	1.1773	1.1490	1.2247	1.1490	1.1502	1.2642	1.1502	1.1389	1.2285	1.1389
5	60	1.3693	1.1734	1.1463	1.3693	1.1463	1.1487	1.2699	1.1487	1.1473	1.3674	1.1473
10	15	1.3608	1.3035	1.2347	1.2472	1.2347	1.1966	1.1995	1.1966	1.1871	1.1878	1.1871
10	20	1.3608	1.2862	1.1599	1.1706	1.1599	1.1419	1.1710	1.1419	1.1339	1.1339	1.1366
10	25	1.3608	1.2407	1.1564	1.1706	1.1564	1.1319	1.1319	1.1333	1.1365	1.1511	1.1365
10	30	1.3608	1.2241	1.1410	1.1706	1.1410	1.1373	1.1473	1.1373	1.1362	1.1564	1.1362
10	35	1.3608	1.2074	1.1482	1.1706	1.1482	1.1496	1.1672	1.1496	1.1291	1.1291	1.1306
10	40	1.3608	1.1902	1.1359	1.1386	1.1359	1.1427	1.1700	1.1427	1.1389	1.1462	1.1389
10	45	1.3608	1.1881	1.1364	1.1547	1.1364	1.1410	1.1497	1.1410	1.1331	1.1385	1.1331
10	50	1.3608	1.1830	1.1547	1.1547	1.1599	1.1454	1.1606	1.1454	1.1382	1.1425	1.1382
10	55	1.3608	1.1773	1.1425	1.1547	1.1425	1.1425	1.1547	1.1425	1.1425	1.1547	1.1425
10	60	1.3608	1.1734	1.1222	1.1222	1.1268	1.1391	1.1647	1.1391	1.1298	1.2066	1.1298
15	20	1.3035	1.2862	1.2124	1.2124	1.2207	1.1741	1.1741	1.1886	1.1751	1.1751	1.1852
15	25	1.3035	1.2407	1.1510	1.1560	1.1510	1.1341	1.1341	1.1407	1.1451	1.1456	1.1451
15	30	1.3035	1.2241	1.1126	1.1139	1.1126	1.1059	1.1101	1.1059	1.1011	1.1014	1.1011
15	35	1.3035	1.2074	1.1394	1.1394	1.1535	1.1139	1.1163	1.1139	1.1178	1.1178	1.1197
15	40	1.3035	1.1902	1.1135	1.1225	1.1135	1.1117	1.1267	1.1117	1.1128	1.1128	1.1225
15	45	1.3035	1.1881	1.1207	1.1225	1.1207	1.1060	1.1303	1.1060	1.1031	1.1031	1.1044
15	50	1.3035	1.1830	1.1239	1.1309	1.1239	1.1184	1.1221	1.1184	1.1103	1.1157	1.1103
15	55	1.3035	1.1773	1.1218	1.1309	1.1218	1.1112	1.1219	1.1112	1.1223	1.1234	1.1223
15	60	1.3035	1.1734	1.1139	1.1139	1.1155	1.1156	1.1166	1.1156	1.1036	1.1036	1.1091
20	25	1.2862	1.2407	1.1987	1.1987	1.2162	1.1671	1.1671	1.1702	1.1747	1.1747	1.1765
20	30	1.2862	1.2241	1.1732	1.1732	1.1904	1.1401	1.1429	1.1401	1.1386	1.1386	1.1418
20	35	1.2862	1.2074	1.1152	1.1313	1.1152	1.1155	1.1155	1.1162	1.1336	1.1336	1.1368
20	40	1.2862	1.1902	1.0988	1.0988	1.1083	1.0912	1.0945	1.0912	1.0878	1.0878	1.0929
20	45	1.2862	1.1881	1.1249	1.1260	1.1249	1.1013	1.1108	1.1013	1.0952	1.0952	1.1015
20	50	1.2862	1.1830	1.1098	1.1098	1.1145	1.0971	1.0971	1.1000	1.0917	1.0917	1.0945
20	55	1.2862	1.1773	1.1062	1.1098	1.1062	1.1010	1.1010	1.1078	1.0904	1.0904	1.0958
20	60	1.2862	1.1734	1.0988	1.0988	1.1024	1.1123	1.1123	1.1156	1.0885	1.0928	1.0885
25	30	1.2407	1.2241	1.2024	1.2024	1.2192	1.1600	1.1600	1.1651	1.1670	1.1670	1.1685
25	35	1.2407	1.2074	1.1629	1.1629	1.1690	1.1297	1.1313	1.1297	1.1431	1.1431	1.1474
25	40	1.2407	1.1902	1.1407	1.1407	1.1548	1.1227	1.1251	1.1227	1.1193	1.1193	1.1204
25	45	1.2407	1.1881	1.0996	1.1143	1.0996	1.0951	1.0975	1.0951	1.1057	1.1057	1.1108
25	50	1.2407	1.1830	1.0969	1.0990	1.0969	1.0946	1.0963	1.0946	1.0817	1.0817	1.0827
25	55	1.2407	1.1773	1.1182	1.1182	1.1204	1.1021	1.1127	1.1021	1.0875	1.0898	1.0875
25	60	1.2407	1.1734	1.1182	1.1182	1.1193	1.0916	1.0957	1.0916	1.0869	1.0869	1.0895
30	35	1.2241	1.2074	1.1950	1.1950	1.1980	1.1522	1.1522	1.1605	1.1509	1.1513	1.1509
30	40	1.2241	1.1902	1.1596	1.1596	1.1784	1.1258	1.1258	1.1264	1.1255	1.1266	1.1255
30	45	1.2241	1.1881	1.1401	1.1401	1.1477	1.1076	1.1076	1.1091	1.1198	1.1198	1.1204
30	50	1.2241	1.1830	1.1061	1.1146	1.1061	1.1105	1.1147	1.1105	1.1123	1.1123	1.1130
30	55	1.2241	1.1773	1.0861	1.0972	1.0861	1.0987	1.0998	1.0987	1.0935	1.0956	1.0935
30	60	1.2241	1.1734	1.0943	1.0943	1.1022	1.0822	1.0822	1.0835	1.0824	1.0824	1.0831
35	40	1.2074	1.1902	1.1799	1.1799	1.2144	1.1567	1.1567	1.1622	1.1488	1.1493	1.1488
35	45	1.2074	1.1881	1.1670	1.1670	1.1905	1.1365	1.1367	1.1365	1.1266	1.1266	1.1340
35	50	1.2074	1.1830	1.1496	1.1496	1.1568	1.1184	1.1259	1.1184	1.1078	1.1078	1.1123
35	55	1.2074	1.1773	1.1341	1.1341	1.1360	1.1062	1.1062	1.1067	1.1097	1.1097	1.1100
35	60	1.2074	1.1734	1.1071	1.1071	1.1147	1.0952	1.0952	1.0976	1.0892	1.0892	1.0939
40	45	1.1902	1.1881	1.1780	1.1780	1.2141	1.1556	1.1556	1.1574	1.1450	1.1457	1.1450
40	50	1.1902	1.1830	1.1728	1.1728	1.1848	1.1243	1.1243	1.1262	1.1319	1.1329	1.1319
40	55	1.1902	1.1773	1.1485	1.1498	1.1485	1.1146	1.1174	1.1146	1.1146	1.1146	1.1166
40	60	1.1902	1.1734	1.1354	1.1354	1.1437	1.0986	1.0988	1.0986	1.1064	1.1064	1.1075
45	50	1.1881	1.1830	1.1736	1.1736	1.2056	1.1407	1.1407	1.1454	1.1454	1.1454	1.1471
45	55	1.1881	1.1773	1.1780	1.1780	1.2026	1.1225	1.1236	1.1225	1.1225	1.1225	1.1244
45	60	1.1881	1.1734	1.1455	1.1455	1.1494	1.1121	1.1121	1.1126	1.1112	1.1112	1.1152
50	55	1.1830	1.1773	1.1718	1.1718	1.1981	1.1449	1.1449	1.1480	1.1443	1.1443	1.1445
50	60	1.1830	1.1734	1.1606	1.1606	1.2065	1.1254	1.1256	1.1254	1.1152	1.1152	1.1163

Table 9: Scaled separation distances for four-dimensional nested approximate maximin designs.