Struct Multidisc Optim (2010) 41:371–395
DOI 10.1007/s00158-009-0432-y

RESEARCH PAPER

Nested maximin Latin hypercube designs

Gijs Rennen · Bart Husslage · Edwin R. Van Dam · Dick Den Hertog

Received: 16 January 2009 / Revised: 19 August 2009 / Accepted: 22 August 2009 / Published online: 6 October 2009

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 variants of the ESE algorithm of Jin et al. (J Stat Plan Inference 134(1):268–287, 2005) are introduced and compared. Our main focus is on GROUPRAND, the most successful of these four variants. In the numerical comparison, we consider the calculation times, space-fillingness of the obtained designs and the performance of different grids. 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 · Multi-fidelity modeling · Nested designs · Sequential simulation · Space-filling · Training and test set · Validation

1 Introduction

Latin hypercube designs are very useful in the approximation of black-box functions. By definition, black-box functions have no explicit description, but can be evaluated to obtain output values for specific input values. As evaluations of a black-box function often involve time-consuming computer simulations, we would like to construct an approximating model (or metamodel) based on evaluations in a (small) number of points. See, e.g., Montgomery (1984), Sacks et al. (1989a, b), Myers (1999), Jones (2001), Booker et al. (1999), Den Hertog and Stehouwer (2002), Santner et al. (2003), Queipo et al. (2005), Wang and Shan (2007), and Kleijnen (2008). A review of metamodeling applications in structural optimization can be found in Barthelemy and Haftka (1993), and in multidisciplinary design optimization in Sobieszczanski-Sobieski and Haftka (1997) and Simpson et al. (2008).

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.

G. Rennen · B. Husslage · E. R. Van Dam (✉) · D. Den Hertog
Department of Econometrics and Operations Research, CentER, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands
e-mail: Edwin.vanDam@uvt.nl

G. Rennen
e-mail: G.Rennen@uvt.nl

B. Husslage
e-mail: Husslage@casema.nl

D. Den Hertog
e-mail: D.denHertog@uvt.nl

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 of experiments for the more traditional response surface methods. Physical experiments exhibit random errors whereas computer experiments are often deterministic (see e.g. Simpson et al. 2004; Forrester et al. 2006, 2008). Therefore, designs for experiments often evaluate certain points multiple times. For designs for computer experiments, replication is redundant because the same input will always result in the same output. This distinction is crucial, so one of the main aims in the field of design of computer experiments (DoCE) is therefore to obtain efficient designs for computer experiments.

As is recognized by several authors, a design for computer experiments should at least satisfy the following two criteria (see Johnson et al. 1990 and Morris and Mitchell 1995). First of all, the design should be *space-filling* in some sense. When no details on the functional behavior of the response parameters are available, it is important to be able to obtain information from the entire design space. Therefore, design points should be "evenly spread" over the entire region. Secondly, the design should be *non-collapsing*. When one of the design parameters has (almost) no influence on the black-box function value, two design points that differ only in this parameter will "collapse", i.e., they can be considered as the same point that is evaluated twice. As evaluation of the deterministic black-box function is often time-consuming, this is not a desirable situation. Therefore, two design points should not share any coordinate values when it is not known a priori which parameters are important. Moreover, we would like the projections of the points onto the axes to be separated as much as possible. When we consider a black-box function on a box-constrained domain, this can be accomplished by using Latin hypercube designs. A Latin hypercube design (LHD) of n points in 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), Audze-Eglais, discrepancy and maximum entropy designs, are also used in the literature. For a good survey of these designs see the book of Santner et al. (2003). Goel et al. (2008) argue that it would be better to use several criteria when selecting a design. However, Santner et al. (2003) show that that maximin Latin hypercube designs generally speaking yield good approximations.

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

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

To start with the first reason, consider the problem of fitting and validating a particular metamodel. In practice, the following approach is often used. First, a metamodel is fitted to the responses obtained when evaluating the design points in the training set. Then, a new set of design points, 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

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

response values are small, the metamodel is considered to be valid. See also Cherkassky and Mulier (1998) for a more detailed description of the use of training and test sets. Because a metamodel should be a global approximation model, i.e., it should be valid for the entire feasible region, both the training set and the test set should cover the entire region. Moreover, the design points in the test set should not lie too close to the design points in the training set, i.e., the total set of design points should be space-filling. This can be accomplished by nesting two designs, which are optimized with respect to, for example, the maximin criterion. The design points that are in both designs then form the training set and the points that are only in the large design make up the test set.

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

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.

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. An alternative method to deal with this situation would be sequential sampling. As this is beyond the scope of this paper, we refer to Jones (2001) and Jin et al. (2002) for more information on sequential sampling.

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 $m \geq 2$. Section 2 gives a more detailed formulation of this problem. When nesting two designs, it is not always possible to satisfy the LHD-structure for both designs. Therefore, we introduce in Section 3 three different grid-structures which approximate the LHD-structure as good as possible. Furthermore, we discuss how to select a suitable grid-structure for a particular application. In Section 4, 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 nested maximin designs, i.e. nested designs which maximize the scaled separation distance d, becomes too time consuming. Therefore in Section 5, we introduce a heuristic which also aims to maximize d but does not guarantee to find the optimal d. We refer to the resulting designs as nested approximate maximin designs. In Section 6, numerical results obtained with this heuristic and the branch-and-bound method are presented and discussed. Finally, Section 7 contains concluding remarks.

2 Problem formulation

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

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

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

is integer. In all other cases, we have to compromise on the LHD-structure of one or both designs. As there are different ways of doing this, we propose three different grid-structures, *nested n_1-grids*, *nested n_2-grids* and *grids with nested maximin axes*, in Section 3. All of these grid-structures are constructed to compromise as little as possible on the LHD-structure. When c_2 is integer, the different grid-structures coincide and are such that both X_1 and X_2 are LHDs.

Secondly, we aim to determine the design points x_i and the set I_1 such that both designs are as much as possible space-filling given the chosen grid-structure. To optimize the space-fillingness, we choose to use the maximin distance criterion. As the distances between the points in X_1 will naturally be greater than the distances between the points in X_2, scaling of these distances is necessary to enable a fair comparison with the maximin distance criterion. Therefore, we define scaled separation distance d_j as the minimal scaled distance between all points in the design X_j:

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

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

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

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 separation distance

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

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

We are aware that the above formulation is just one way of combining the two scaled separation distances into one objective and that other scaling factors or formulations are also possible. Using different scaling factors is no problem as all methods discussed in this paper can also be used for other values of s_1 and s_2. In Section 7, we will discuss some other alternative objectives. Dealing with maximizing d_1 and d_2 as a bi-objective optimization problem is another possibility. For two-dimensional nested designs with small n_1 and n_2, we use this approach in Section 4.2. However, to limit the scope of this paper, our main focus will be on the above maximin objective.

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

3 Grid-structures for nested Latin hypercube designs

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

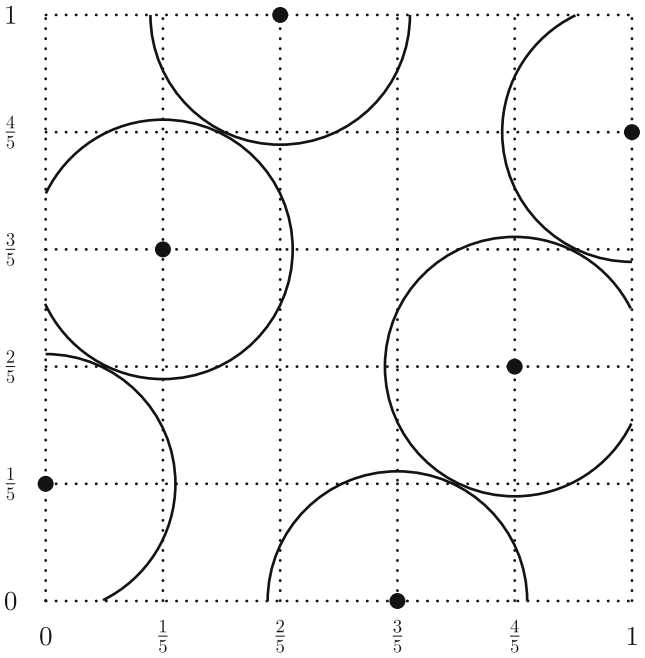

Fig. 1 A maximin Latin hypercube design of 6 points; $d_1 = 1.0000$

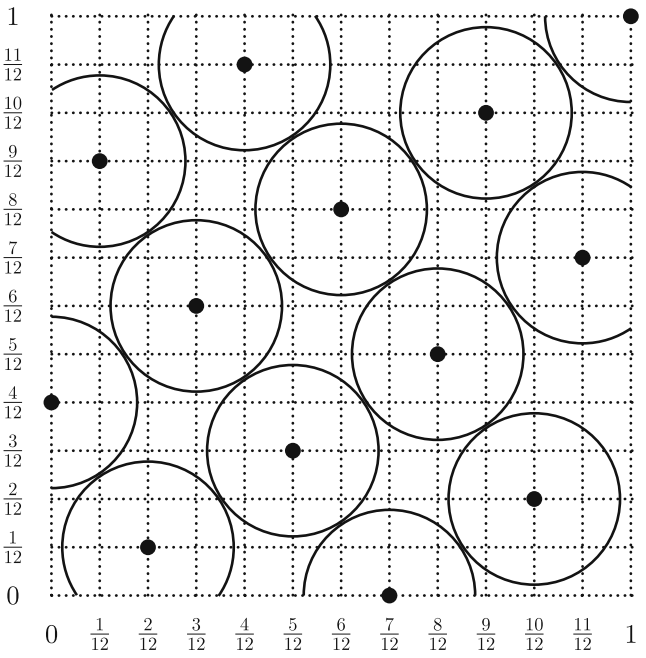

Fig. 2 A maximin Latin hypercube design of 13 points; $d_2 = 1.0408$

which represent different compromises. Furthermore, we discuss how to decide which grid-structure is most suitable for a particular situation.

To illustrate the different structures, examples are provided for the two-dimensional case of $n_1 = 6$ and $n_2 = 13$ points. In Figs. 1 and 2 also the individual maximin Latin hypercube designs of $n_1 = 6$ and $n_2 = 13$ points are depicted to enable comparison with the non-nested case. To compare the design in Fig. 1 with X_1 designs and the design in Fig. 2 with X_2 designs, we calculated the values of d_1 and d_2 for these non-nested designs. The circles illustrate the unscaled separation distance because when we draw circles with the design points as their centers, the separation distance is equal to the largest diameter such that the circles are non-overlapping. Moreover, it shows where the separation distance is attained.

3.1 Nested n_2-grid

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

To construct a nested design where X_2 is an LHD, we have to choose all design points on the n_2-grid, with grid points $\{0, \frac{1}{n_2-1}, \frac{2}{n_2-1}, \ldots, 1\}^m$. 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 Fig. 3. As all grid-structures coincide when c_2 is integer, this design is also a nested maximin design for the other two grid-structures. Therefore, we refer to it as a nested maximin LHD instead of a nested maximin n_2-LHD.

For the case $c_2 \notin \mathbb{N}$, the situation is more complicated. Because we are bound to the n_2-grid, and $n_1 - 1$ is no longer a divisor of $n_2 - 1$, it is 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

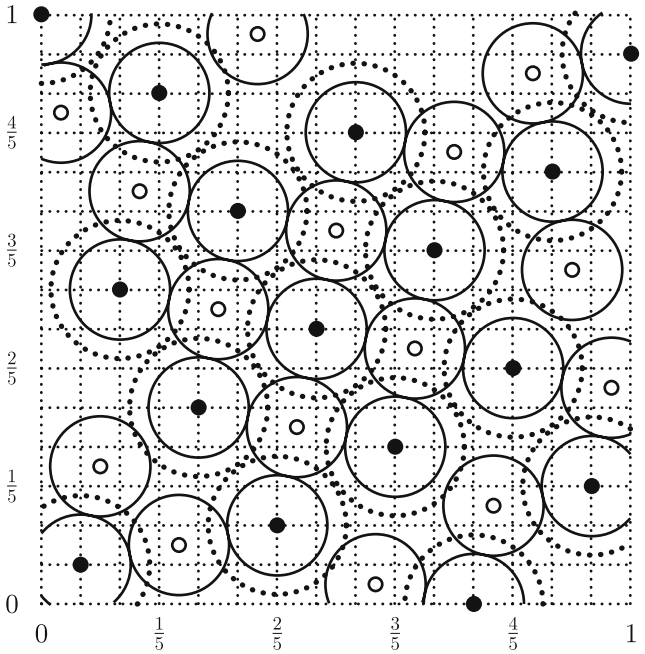

Fig. 3 A nested maximin Latin hypercube design of $n_1 = 16$ and $n_2 = 31$ points; $d = d_1 = d_2 = 0.9309$

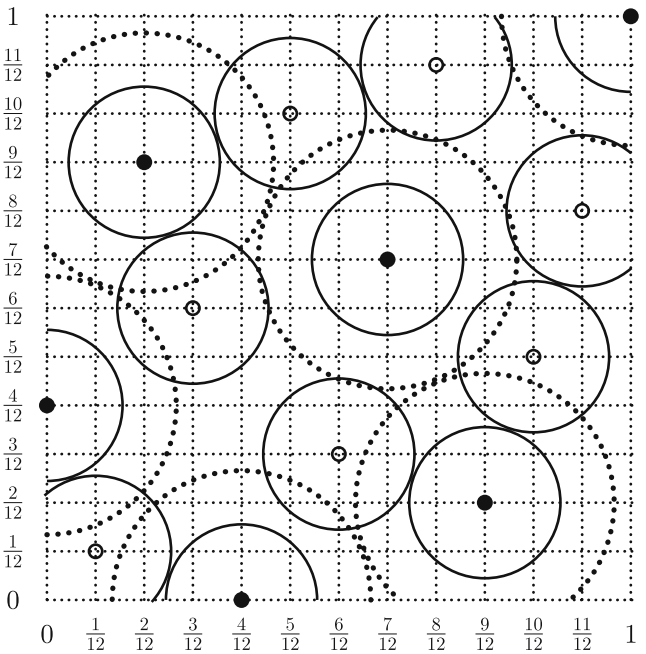

Fig. 4 A nested maximin n_2-Latin hypercube design of $n_1 = 6$ and $n_2 = 13$ points; $d = d_2 = 0.9129$ and $d_1 = 1.0035$

X_1-coordinates; see Van Dam et al. (2009a) (where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ represent the floor and ceiling function, respectively). 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 Fig. 4. In this and following figures, the design points of X_1 are represented by solid dots, the open dots represent the extra design points needed to complete design X_2, hence, the solid and open dots together form the design points of X_2. The diameters of the dotted and solid circles are equal to the unscaled distance $d_1 * s_1$ and $d_2 * s_2$, respectively. They thus illustrate the separation distances of the designs X_1 and X_2.

For the nested n_2-grid, a suitable method to determine nested LHDs would seem to take an existing LHD of n_2-points for X_2 and select a subset of n_1 points for X_1. Forrester et al. (2007), for instance, use an exchange algorithm to implement this approach for multi-fidelity modeling. Although this method is quite attractive because of its simplicity, it does not generally yield a nested LHD satisfying all the restrictions of the nested n_2-grid. We will illustrate this with the example in Fig. 5. 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

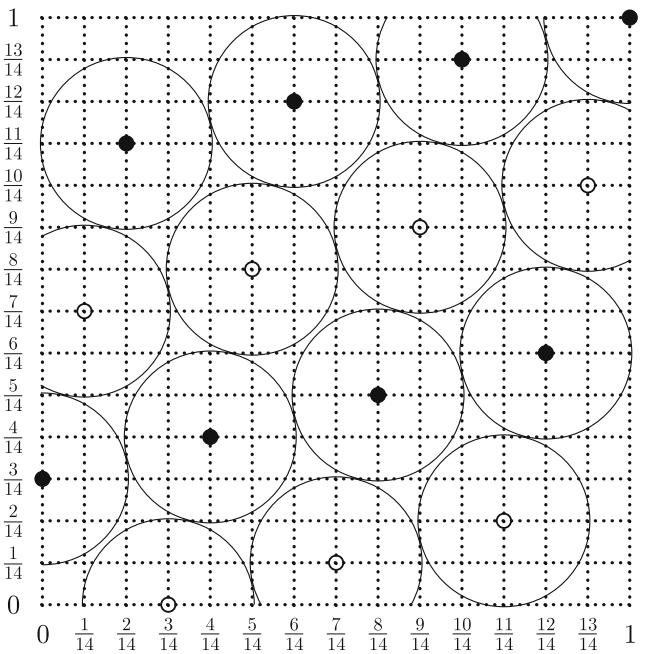

Fig. 5 Example of the problem occurring when taking X_1 equal to a subset of an existing LHD

of the X_1-coordinates on the other axis is certainly not equidistant or space-filling. This problem also occurs for many other Latin hypercube designs and is even more likely to occur when the number of dimensions increases. Therefore, we will not use this method to construct nested LHDs, but use the methods described in Sections 4.1 and 5.1.

3.2 Nested n_1-grid

When we want X_1 to be an LHD instead of X_2, we can use the nested n_1-grid. The design X_1 is then obtained by choosing n_1 design points on the n_1-grid, with grid points $\{0, \frac{1}{n_1-1}, \frac{2}{n_1-1}, \ldots, 1\}^m$. 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 Fig. 6 for an example of a nested maximin design on a nested n_1-grid of $n_1 = 6$ and $n_2 = 13$ points, with $d = d_2 = 0.9522$ and $d_1 = 1.0000$.

3.3 Grid with nested maximin axes

The use of the Latin hypercube structure in the construction of a nested maximin design implies a preference of one design over the other. Design X_1 is assumed to be more important than design X_2 when a nested n_1-grid is used; design X_2 is preferred over design X_1 in case of a nested n_2-grid. If both sets are assumed to be of equal importance, we would like to treat them equally. To deal with this problem, the X_1- and X_2-coordinates could be restricted to take only values at the levels of a (known) one-dimensional nested maximin design of n_1 and n_2 points; see Van Dam et al. (2009a). The design points of X_1 and X_2 could then be chosen from the grid points obtained in this way. Note that in this case the projections of the design points onto the axes are always optimally space-filling with respect to the maximin distance criterion. Furthermore, note that a one-dimensional maximin design, with $c_2 \notin \mathbb{N}$, is (again) not unique, so there are multiple grids possible. Figure 7 depicts an example of a nested maximin design of $n_1 = 6$ and $n_2 = 13$ points on a grid with nested maximin axes, with $d = d_1 = 0.9589$ and $d_2 = 0.9805$.

3.4 Choice of grid-structure

When deciding which grid-structure to use, there are a number of factors which can influence this decision. Firstly, when the space-fillingness of the nested design is very important, we can base our choice on which grid-structure results in the nested design with the highest d. Secondly, when it is not known a priori which design parameters are significant, the non-collapsingness

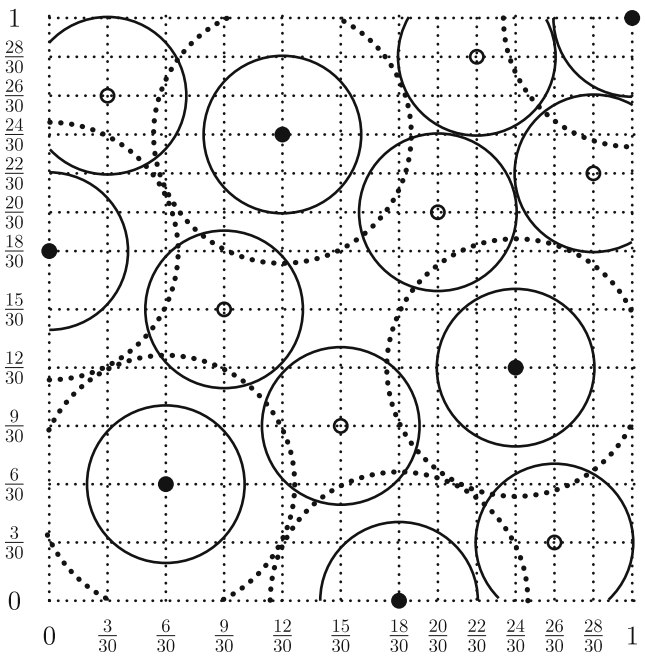

Fig. 6 A nested maximin n_1-Latin hypercube design of $n_1 = 6$ and $n_2 = 13$ points; $d = d_2 = 0.9522$ and $d_1 = 1.0000$

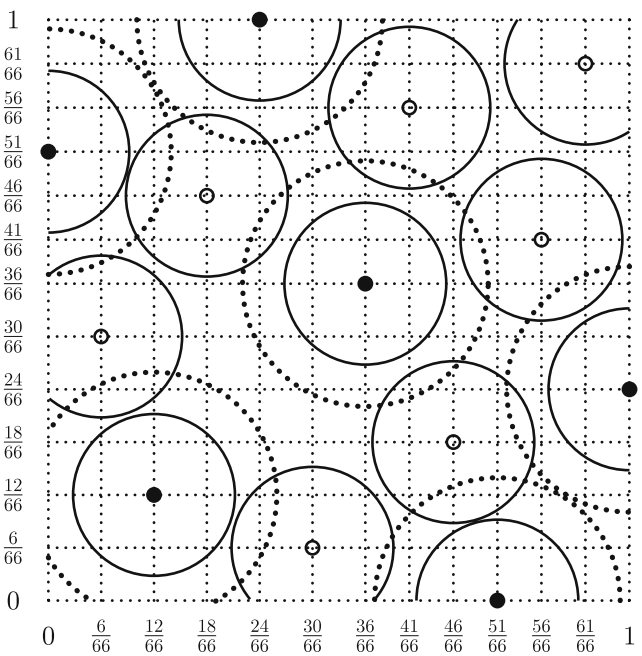

Fig. 7 A nested maximin design of $n_1 = 6$ and $n_2 = 13$ points on a grid with nested maximin axes; $d = d_1 = 0.9589$ and $d_2 = 0.9805$

criterion should be considered. The projections of the design point onto the axes should preferably be space-filling, which is accomplished by choosing a grid with nested maximin axes.

Furthermore, the reason why a nested design is used may also affect the choice for a particular grid. For example, a nested n_1-grid or a grid with the highest d_1 could be preferable for sequential evaluations, because it is known with certainty that the first set of design points is evaluated, whereas the evaluation of an extra set of design points may depend on the previously evaluated set. However in the same setting, a nested n_2-grid is 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" grid-structure depends on the application under consideration and the user's preferences. Fortunately, when $c_2 \in \mathbb{N}$, the comparison of the various nested grid-designs is superfluous. In this case, we do not have to differentiate between different grid-structures, because they will all yield the same nested maximin design (and maximin distance).

4 Two-dimensional nested designs

4.1 Branch-and-bound algorithm

To obtain two-dimensional nested maximin LHDs, we use an extension of the branch-and-bound algorithm of Van Dam et al. (2007). This extended branch-and-bound algorithm works as follows. Given n_1, n_2 and the grid-structure, we first determine all possible nested grids and calculate the possible distances that can occur for X_1 and X_2. These distances form a discrete set that can be efficiently searched and optimized. To determine whether a nested LHD exists with X_1 and X_2 having scaled 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 scaled separation distances smaller than d_1 or d_2.

Using the extended branch-and-bound algorithm, we obtained results for n_2 up to 15 for all three grid-structures. For the cases where $c_2 \in \mathbb{N}$, the algorithm is refined so that nested maximin LHDs could be obtained for n_2 up 32. The maximin distances of these designs can be found in Tables 9 and 10 in Appendix B. The corresponding designs can be found on the website http://www.spacefillingdesigns.nl. In Section 6, the results are compared and discussed.

4.2 Pareto nested designs

Besides nested designs that maximize the objective function $d = \min\{d_1, d_2\}$, there are also some other interesting nested designs 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 ensures that the distances (d_1, d_2) of all nested maximin designs provided in Table 9 (see Appendix B) 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. 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

Table 1 All two-dimensional pairs (n_1, n_2) with more than one Pareto combination; $c_2 \in \mathbb{N}, n_2 \leq 32$

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)

of $n_1 = 11$ and $n_2 = 21$ points there exist two different Pareto combinations, both with a maximin distance equal to $d = 0.7071$. For the (n_1, n_2) pairs $(9, 17)$ and $(10, 19)$, the objective values of the Pareto nested designs are also equal (0.7906 and 0.7454, respectively); however, the individual maximal distances of the second Pareto combination are smaller than the maximal distances of the (optimal) first combination ($1.0607 < 1.1180$ and $1.0000 < 1.0541$, respectively). The Pareto nested designs can also be found on the website http://www.spacefillingdesigns.nl.

5 Higher dimensional nested designs

5.1 Enhanced stochastic evolutionary algorithm

For dimensions higher than two and for larger values of n_1 and n_2, the above branch-and-bound algorithm 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. Furthermore, the algorithm is used in Viana et al. (2007) to generate space-filling LHDs. Although this algorithm was originally designed for non-nested designs, with some changes it is also applicable to nested designs. Before we look at these changes, we first give a short description of the original ESE algorithm. This description is based on 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 using a threshold criterion. The criterion ensures that better designs are always accepted and that worse designs can be accepted with a certain probability depending on the threshold value. If the new design is accepted, it replaces the current design. This process 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 fewer 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.

Figures 8 and 9 give an example where this distortion indeed occurs. The design in Fig. 9 is obtained by exchanging one coordinate value of two points in the lower left part of Fig. 8. As we require in each dimension that the first and last point should be in X_1, the new design is not a valid nested 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

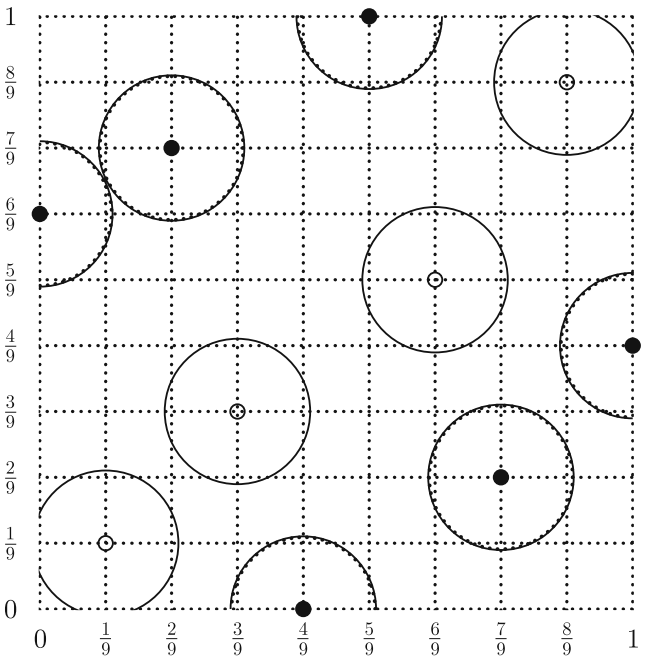

Fig. 8 A nested Latin hypercube design of $n_1 = 6$ and $n_2 = 10$ points; $d = d_1 = 0.3086$ and $d_2 = 0.5556$

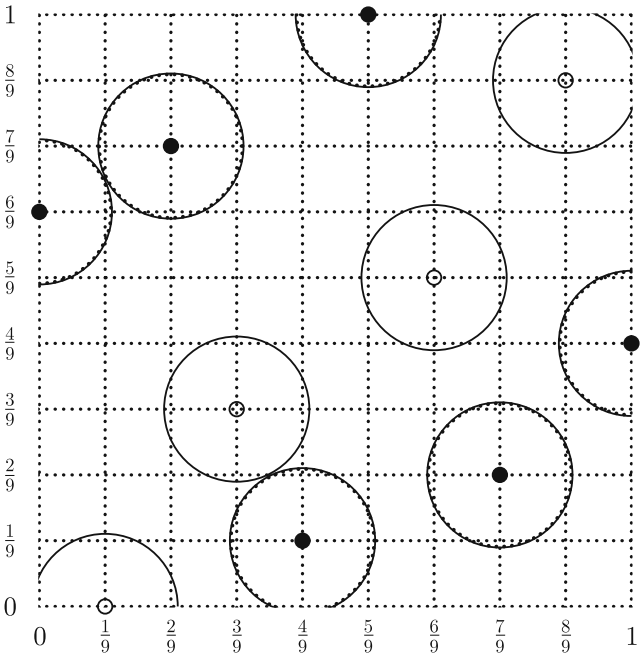

Fig. 9 A nested design obtained by exchanging one coordinate value between a point in X_1 and one point in $X_2 \backslash X_1$

assignment such that the invalid nested design in Fig. 9 becomes a valid nested LHD on a nested n_2-grid.

5.2 Generating new designs

The main problem which needs to be addressed by a new method for generating 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 randomly 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.

We also want to 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 run the ESE algorithm for different initial grids and designs. Furthermore, besides the exchange of coordinate values, we also consider other transformations which are able to change a grid without distorting it.

Based on the above observations, we developed four different methods for generating new designs. We describe the most successful method in this section. A description of the other three methods and a comparison of all four methods can be found in Appendix A.

The most successful method method, called GROUPRAND, can change both the grid and the selected points of a design. 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 definition of the GROUPRAND method.

The method GROUPRAND starts with randomly selecting a first point from X_2. 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

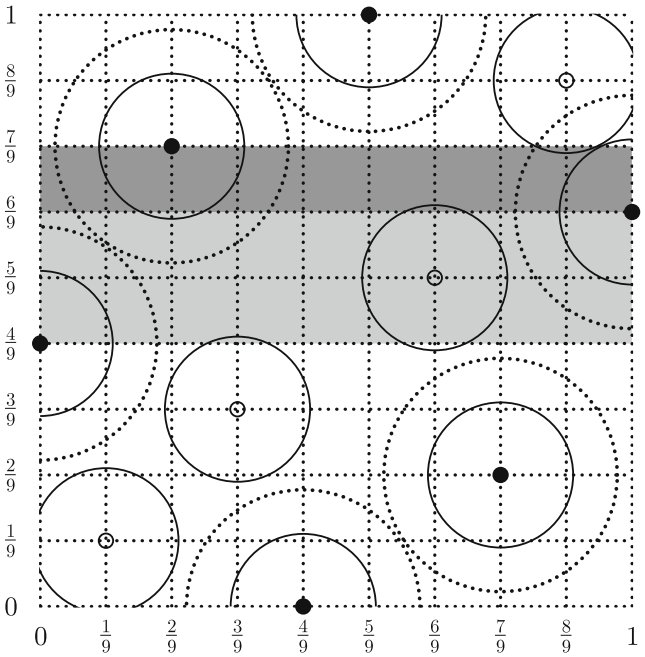

Fig. 10 A nested Latin hypercube design of $n_1 = 6$ and $n_2 = 10$ points; $d = d_2 = 0.7454$ and $d_1 = 0.8958$

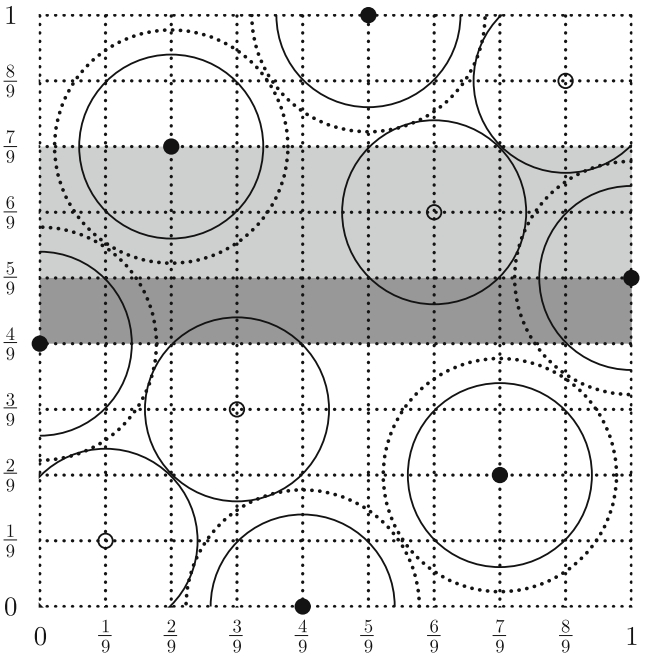

Fig. 11 A nested maximin Latin hypercube design of $n_1 = 6$ and $n_2 = 10$ points; $d = d_1 = 0.8958$ and $d_2 = 0.9428$

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 GROUPRAND method, we will use the design in Fig. 10 and take k equal to 2, i.e. the dimension on the vertical axis. The two differently shaded areas in Fig. 10 now form two possible groups. Notice that the top group is empty as there are no X_2-points between the X_1-points. In Fig. 11, we see the result of exchanging the two groups. Because the groups are of different size, the grid has now changed. Comparing the values of d shows that the design has improved and, in this case, is even optimal.

6 Numerical results

In the numerical comparison in this section, we consider nested designs of dimensions two up to ten. The two-dimensional nested maximin designs are obtained with the branch-and-bound algorithm described in Section 4.1. The three- and four-dimensional nested approximate maximin designs are obtained by performed ten runs of each variant of the ESE algorithm described in Section 5.2 and Appendix A. For dimensions five up to ten, we only used ten runs of the GROUPRAND algorithm. In this comparison, we consider the best obtained designs and all ten designs obtained with GROUPRAND. For each dimension and grid-structure, we ran the ESE algorithm for 65 different pairs (n_1, n_2) with $n_1 = 5, 10, \ldots, 50$ and $n_2 = n_1 + 5, \ldots, 55, 60$. The computations have been performed on PCs with a 2.8-GHz Pentium D processor and the variants were implemented in Matlab 7.4 R2007a. More details on the computations can be found in Appendices A and C.

To get an impression of the calculation times, we determined the mean and average of the calculation times over all ten runs of the GROUPRAND algorithm (for each pair (n_1, n_2) and dimension m). Table 2 contains the results for the nested n_1-grid and Table 13 in Appendix C contains the results for all three grids. When evaluating these results, we note that the calculation times are mainly influenced by two factors. Firstly, generating a new design and calculating d becomes more time-consuming when n_2 or the dimension m increases. Especially, updating the distance matrix containing the distances between all design points requires more time. Secondly, the total number of iterations of the outer loop can vary over different runs of the ESE algorithm. This is caused by our stopping criterion, which terminates the algorithm when no improvement has been found in the last 100 iterations of the outer loop.

The effect of n_2 is most apparent when we compare the calculation times for a fixed n_1. For some cases, especially with $n_1 = 5$, the calculation times are quite large. Note however, that once we have generated a good design for a certain pair (n_1, n_2) in a certain dimension, we can use this design many times without

Table 2 Mean and standard deviation of calculation times in minutes of GROUPRAND for the nested n_1-grid

n_1	n_2	Calculation time							
		Mean				Std			
		4d	6d	8d	10d	4d	6d	8d	10d
5	10	0.1	0.3	0.4	0.3	0.0	0.1	0.1	0.1
5	20	1.6	4.3	4.9	3.6	1.2	1.8	1.7	1.4
5	30	8.0	22.6	21.4	15.0	5.0	6.9	5.0	5.1
5	40	19.2	40.6	51.5	23.7	14.8	5.7	9.6	3.4
5	50	27.0	90.0	118.2	66.0	15.2	19.4	17.9	11.5
5	60	26.4	176.8	216.5	118.5	5.4	47.7	30.7	16.3
10	15	0.1	0.6	0.8	0.7	0.0	0.2	0.2	0.3
10	20	0.3	1.5	2.6	2.1	0.1	0.5	1.1	0.8
10	30	1.5	8.3	10.4	5.3	0.4	2.6	4.8	1.5
10	40	4.4	13.3	18.2	7.4	1.4	4.9	7.7	1.2
10	50	5.6	23.1	24.8	14.4	1.9	12.0	7.4	4.8
10	60	7.0	31.2	37.8	23.1	4.2	7.4	10.3	4.9
15	20	0.3	1.6	2.2	1.3	0.1	0.4	0.6	0.4
15	30	1.2	6.6	5.9	4.5	0.4	2.3	1.6	1.5
15	40	2.0	8.1	8.9	6.2	0.8	2.9	3.2	2.9
15	50	3.4	13.2	15.8	8.6	1.5	4.8	7.4	2.5
15	60	4.7	23.5	25.0	15.7	2.4	13.6	7.3	4.8
20	25	0.6	3.1	3.1	3.4	0.2	1.6	1.1	1.6
20	30	1.2	6.2	7.2	3.8	0.4	2.3	2.8	1.3
20	40	2.1	7.9	8.8	4.4	0.5	2.3	3.1	1.3
20	50	3.2	12.8	16.5	8.9	1.5	5.9	8.6	2.8
20	60	3.0	20.3	17.9	11.7	1.2	7.9	6.8	2.8
25	30	1.2	5.7	6.9	4.4	0.3	1.0	2.7	1.0
25	40	2.4	9.2	10.3	8.2	0.6	2.7	3.0	3.5
25	50	2.8	13.7	17.7	8.4	0.7	6.3	7.7	3.7
25	60	5.4	19.3	25.6	22.0	2.7	9.1	9.3	6.4
30	35	2.0	8.4	9.6	6.6	0.9	2.5	1.7	2.7
30	40	2.6	10.4	13.4	10.6	0.7	3.1	5.3	3.0
30	50	3.3	16.1	19.4	15.1	0.3	4.8	4.1	4.5
30	60	5.8	21.3	27.8	18.3	1.8	6.5	11.4	8.7
35	40	2.8	9.4	12.9	9.9	0.9	2.4	3.6	3.0
35	50	4.3	18.9	24.6	20.2	0.7	6.1	5.7	6.9
35	60	5.6	26.5	32.6	29.2	1.7	13.0	9.4	11.1
40	45	6.1	16.0	16.2	13.9	2.8	6.6	7.3	3.8
40	50	9.9	28.4	29.0	18.7	4.6	11.8	9.1	3.9
40	60	6.0	30.5	42.5	33.2	2.9	7.7	10.4	10.0
45	50	5.3	19.3	22.8	20.9	1.6	6.9	4.2	6.6
45	60	9.0	40.6	52.8	35.8	3.0	10.4	17.4	15.5
50	55	10.4	29.6	33.9	35.8	4.0	10.7	6.6	8.1
50	60	13.0	40.4	48.4	30.9	6.9	10.3	17.8	12.5

incurring these calculation times again. When we compare the times for different dimensions m, we see that the times generally do increase, but not as much as one might expect. For many pairs the times for 10 dimensions are even lower than for 8 dimensions. This result is probably caused by the second factor, i.e., the variance in number of iterations caused by the stopping criterion. When we consider a certain pair (n_1, n_2) and dimension m, this factor is also the main cause for the variance of the calculation times.

Besides the calculation times, we are also interested in the performance of GROUPRAND in terms of the space-fillingness of the obtained nested designs. As there are no other results in literature for (approximatie) maximin nested designs, we cannot compare our results. Therefore, we decided to perform Monte Carlo simulations to get an impression of how the obtained designs are positioned in the objective function space. We performed these Monte Carlo simulations by randomly generating 10,000 nested designs for each grid-structure, pair (n_1, n_2) and dimension m. We denote the mean, standard deviation and maximum of the d-values of these 10,000 designs by $mean_{mc}$, std_{mc} and max_{mc}, respectively. In Table 3, we compare these values to d_{best}, i.e., the d value of the best obtained nested (approximate) maximin design. The first figure shows how many times d_{best} is larger than $mean_{mc}$. The second figures also shows the difference between d_{best} and $mean_{mc}$, but now in terms of the number of standard deviations std_{mc}. The last figure compares d_{best} to max_{mc}, i.e., the d-value of the best design found by the Monte Carlo simulation. All three figures show that d_{best} is considerably larger than $mean_{mc}$ and max_{mc}. The GROUPRAND algorithm is thus able to significantly improve the space-fillingness of the nested designs.

However, compared to non-nested designs of sizes n_1 and n_2, we expect a lower space-fillingness because of the additional condition that X_1 and X_2 should be nested. Therefore, we also evaluate the loss 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 nested (approximate) maximin designs 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 scaled separation distance as:

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

Table 4 represents the averages and the ranges of the percentage losses for the best nested (approximate) maximin designs. The average and range are both taken 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

Table 3 Comparison of d_{best} to the d values of the Monte Carlo simulation

	Average over all (n_1, n_2)-pairs							
	3d	4d	5d	6d	7d	8d	9d	10d
Nested n_1-grid								
$d_{best}/mean_{mc}$	4.0	4.5	4.9	5.3	5.6	5.9	6.2	6.4
$(d_{best} - mean_{mc})/std_{mc}$	10.2	11.9	13.4	14.6	15.7	16.7	17.6	18.4
$(d_{best} - max_{mc})/std_{mc}$	6.4	8.2	9.6	10.9	12.0	13.0	13.9	14.7
Nested n_2-grid								
$d_{best}/mean_{mc}$	3.9	4.3	4.7	5.1	5.4	5.7	5.9	6.1
$(d_{best} - mean_{mc})/std_{mc}$	10.2	11.8	13.2	14.4	15.5	16.5	17.3	18.2
$(d_{best} - max_{mc})/std_{mc}$	6.4	8.0	9.4	10.6	11.7	12.7	13.5	14.4
Nested maximin axes								
$d_{best}/mean_{mc}$	3.9	4.4	4.8	5.1	5.4	5.7	6.0	6.2
$(d_{best} - mean_{mc})/std_{mc}$	10.2	11.8	13.2	14.4	15.5	16.5	17.4	18.2
$(d_{best} - max_{mc})/std_{mc}$	6.4	8.0	9.4	10.6	11.7	12.7	13.6	14.4

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.

The percentage losses in Table 4 are only of the best nested (approximate) maximin designs. To determine how the percentage losses vary over different runs of the ESE algorithm, we also report the mean and standard deviation over all 10 runs of the GROUPRAND algorithm. Table 5 contains the loss $l_1(n_1, n_2)$ for the nested n_1-grid and Tables 14 and 15 in Appendix C contain $l_1(n_1, n_2)$ and $l_2(n_1, n_2)$ for all three grids. To limit the size of the tables, we only report the results for a subset of the evaluated (n_1, n_2)-pairs. We can see in these tables that the standard deviation is fairly low for all cases with $n_1 \geq 10$. For these cases, one run or a small number of runs of the GROUPRAND algorithm is sufficient to determine a good nested approximate maximin design. The relatively high standard deviation for cases with $n_1 = 5$ indicates that for these cases it is beneficial to perform more runs of the GROUPRAND algorithm. When we compare the results for different dimensions, there is no real pattern in the standard deviations.

Besides the performance of GROUPRAND, we are also interested in the performance of the different grid-structures. As can already be seen from the results in Tables 10, 11, and 12, which grid gives the best separation distances depends on the particular pair (n_1, n_2) and dimension. To get an idea of which grid-structures perform well in general, 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 in Table 6. We do not consider the pairs with $c_2 \in \mathbb{N}$ because for these pairs all grids are equal.

Not surprisingly, the grids with the lowest average loss in Table 4 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 strongly depends on

Table 4 Average and range of percentage loss $l_j(n_1, n_2)$ for the best nested (approximate) maximin designs over all evaluated (n_1, n_2)-pairs

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

Table 5 Mean and standard deviation of percentage loss $l_1(n_1, n_2)$ of GROUPRAND for the nested n_1-grid

n_1	n_2	$l_1(n_1, n_2)$							
		Mean				Std			
		4d	6d	8d	10d	4d	6d	8d	10d
5	10	6.06	10.21	10.85	11.49	6.80	3.71	1.85	2.65
5	20	9.64	12.70	15.97	12.59	6.23	3.03	3.89	4.74
5	30	9.19	13.17	18.20	12.78	4.64	4.30	3.02	3.64
5	40	10.61	15.57	15.52	15.31	3.12	4.66	3.89	5.58
5	50	10.42	15.57	14.33	12.98	6.82	4.66	3.41	5.52
5	60	9.36	16.09	13.93	11.09	7.57	6.10	4.29	3.96
10	15	11.92	6.97	5.09	3.90	1.48	0.95	0.59	0.70
10	20	16.24	13.31	10.24	9.39	2.13	1.11	1.68	1.46
10	30	17.69	14.43	12.09	12.05	2.33	1.00	1.22	0.88
10	40	17.93	15.39	13.38	13.33	2.30	1.56	1.08	0.56
10	50	18.77	16.86	14.60	13.97	1.54	1.50	0.69	1.03
10	60	20.91	17.44	15.10	14.54	2.20	1.38	1.47	0.80
15	20	8.64	4.06	5.65	3.43	1.01	0.91	1.36	1.17
15	30	15.01	9.42	10.90	7.69	0.99	0.72	1.04	0.74
15	40	15.74	11.25	11.16	8.60	0.87	1.78	0.64	0.69
15	50	16.42	12.60	13.26	10.82	1.65	1.04	1.37	0.49
15	60	17.43	13.25	14.38	11.74	1.54	0.71	0.87	0.77
20	25	8.67	4.44	2.56	6.42	1.38	1.09	0.68	1.06
20	30	9.91	4.95	3.12	7.61	0.95	0.83	0.90	0.73
20	40	14.31	10.68	7.67	11.41	0.64	1.04	0.69	0.43
20	50	14.65	10.37	6.36	10.06	0.77	1.48	1.06	0.54
20	60	17.58	12.41	10.23	13.37	1.58	0.94	0.67	0.49
25	30	4.88	3.29	2.30	2.37	1.39	0.48	0.51	0.37
25	40	8.91	5.53	3.44	2.96	0.89	0.51	0.48	0.51
25	50	13.07	10.28	7.14	7.11	0.60	1.14	0.80	0.79
25	60	12.73	9.61	5.73	4.57	1.46	1.49	0.97	0.50
30	35	3.99	2.68	3.44	2.35	1.00	0.72	0.48	0.45
30	40	5.74	3.66	3.77	2.53	0.53	0.57	0.56	0.50
30	50	9.80	5.93	4.83	3.06	0.32	0.50	0.43	0.28
30	60	11.72	9.51	8.64	6.58	0.85	0.51	0.93	0.52
35	40	3.42	3.06	2.69	2.21	1.00	0.89	0.44	0.40
35	50	6.20	3.99	3.00	2.51	0.67	1.01	0.43	0.35
35	60	9.35	6.46	4.63	3.40	0.94	0.59	0.58	0.48
40	45	2.31	3.66	2.86	3.99	0.89	0.64	0.57	0.33
40	50	3.01	4.07	3.44	4.45	0.96	0.90	1.47	0.20
40	60	6.01	5.34	3.95	4.56	0.75	0.67	1.23	0.31
45	50	2.43	3.59	1.98	1.61	0.71	0.60	0.43	0.30
45	60	4.30	4.41	2.24	2.13	0.36	0.45	0.60	0.39
50	55	2.48	1.41	1.56	1.71	0.91	0.59	0.32	0.32
50	60	3.17	1.72	2.02	2.49	0.62	0.62	0.62	0.55

the particular pair (n_1, n_2) which grid to choose based on the separation distances.

Furthermore, in many practical situations, the values of n_1 and n_2 are not fixed which leaves some freedom to change these values. In those situations, we can thus also consider nested designs where n_1 and n_2 are slightly lower or higher. Let us, for example, consider the two-dimensional designs with $n_1 = 5$ and $n_2 = 10$. In Table 7, 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.

Table 6 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

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

As mentioned in Section 3.4, the choice for a specific grid or (n_1, n_2)-pair does not only depend on the space-fillingness. The users preferences and the reason why a nested design is used also affect the choice for a particular grid-structure.

7 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

Table 7 Example of reduction in percentage loss $l_j(n_1, n_2)$ by choosing different value for n_1

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

axes. Which grid to use is mainly depends 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, determining nested LHDs which maximize d becomes too time-consuming. Instead, we determine nested approximate maximin LHDs which do not guarantee optimality of d. To determine these designs, we considered four variants of the ESE algorithm. We mainly focused our attention on the GROUPRAND method as this method performed best in a comparison of three- and four-dimensional designs. Using this method, designs for dimensions five up to ten and for up to 100 design points were obtained. Note that all variants of the ESE algorithms can also be used for higher dimensions and larger numbers of points. When the number of points or dimensions increases, changing the design and calculating the new value for d will become more time-consuming. Comparing the calculation times over different dimensions shows that the times indeed do increase, but not as much as one might expect. A method to compensate the increase in calculation time could be to perform fewer iterations of the inner loop of the ESE algorithm, although this might reduce the quality of the final nested design.

Besides comparing the different variants, we also studied the performance using Monte Carlo simulation. The results of this simulation show that the ESE algorithm is able to significantly improve the space-fillingness of the nested designs. Furthermore, we considered the percentage 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. We also noted that we can reduce the amount of loss by choosing slightly different values for n_1 and n_2.

We remark that the objective $d = \min\{d_1, d_2\}$ used in this paper is only one way of combining the separation distances of X_1 and X_2. As mentioned in the introduction, alternative scaling factors and formulations are possible. Taking the weighted sum of both objectives instead of the minimum would be a possible alternative objective. When using this objective, the branch-and-bound and ESE algorithms can still be used with little or no adjustments. Dealing with maximizing d_1 and d_2 as a bi-objective optimization problem is another possibility. In that case, different Pareto optimal nested designs could be found by using the weighted sum objective with various scaling factors.

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

Acknowledgement The authors would like to thank the referees for their valuable comments.

Appendix A: Variants of ESE algorithm

Besides the GROUPRAND method, we developed and tested three other methods which differ in two aspects. Firstly, they differ in the method used to choose between X_1 and $X_2 \backslash X_1$. In GROUPRAND random selection is used, 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 $X_2 \backslash X_1$ might 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.

Secondly, the methods differ in the possible exchanges. To test the effect of exchanging groups, we also tested methods which only allow the exchange of points. These methods might be faster as a group exchange is more difficult than a exchange of points. However, a drawback of not using group exchanges is that the initial grid cannot change.

As we have two options for each of the above aspects, we developed the following four methods for generating new designs. The method 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.

The method POINTDMIN is similar to POINTRAND but does take the values of d_1 and d_2 into account. When d_2 is smallest, the method works in exactly 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 similar to POINTRAND and GROUPRAND, respectively. The only difference is that these two methods also allow group exchanges as described in Section 5.2.

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. 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. Also the calculation times of the ESE algorithm did not significantly differ. Therefore, we do not make a distinction between these two types in the 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 11 and 12 of Appendix B, 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 8. 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 cannot take the sum of two columns to determine the combined performance of two methods. When we study the results, we see that the two RAND methods find the best design for most cases. One 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 approximate maximin designs for dimensions five up to ten. However, in this paper we will only discuss the results of GROUPRAND.

Table 8 Percentage of the (n_1, n_2)-pairs for which a certain variant of the ESE algorithm finds a best design

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

Appendix B: Values of d for best nested (approximate) maximin designs

Table 9 provides the maximin distances for nested maximin Latin hypercube designs in two dimensions 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 10 provides the maximin distances for the two-dimensional nested maximin designs for all three grid-structures. Tables 11 and 12 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.

Table 9 Maximin distances for two-dimensional nested maximin Latin hypercube designs; $c_2 \in \mathbb{N}$

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

Table 10 Maximin distances for two-dimensional nested designs; $c_2 \notin \mathbb{N}$

n_1	n_2	$d(n_1)$	$d(n_2)$	Nested n_1-grid			Nested n_2-grid			Grid with nested axes		
				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

Table 10 (continued)

n_1	n_2	$d(n_1)$	$d(n_2)$	Nested n_1-grid			Nested n_2-grid			Grid with nested axes		
				d	d_1	d_2	d	d_1	d_2	d	d_1	d_2
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 11 Scaled separation distances for three-dimensional nested approximate maximin designs

n_1	n_2	$d(n_1)$	$d(n_2)$	Nested n_1-grid			Nested n_2-grid			Grid with nested axes		
				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

Table 11 (continued)

n_1	n_2	$d(n_1)$	$d(n_2)$	Nested n_1-grid			Nested n_2-grid			Grid with nested axes		
				d	d_1	d_2	d	d_1	d_2	d	d_1	d_2
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 12 Scaled separation distances for four-dimensional nested approximate maximin designs

n_1	n_2	$d(n_1)$	$d(n_2)$	Nested n_1-grid			Nested n_2-grid			Grid with nested axes		
				d	d_1	d_2	d	d_1	d_2	d	d_1	d_2
5	10	1.3693	1.3608	1.2141	1.2748	1.2141	1.2472	1.3240	1.2472	1.2201	1.3279	1.2201
5	15	1.3693	1.3035	1.2069	1.2247	1.2069	1.2203	1.3325	1.2203	1.2001	1.3880	1.2001
5	20	1.3693	1.2862	1.1683	1.2247	1.1683	1.1784	1.1839	1.1784	1.1888	1.2094	1.1888
5	25	1.3693	1.2407	1.1738	1.3693	1.1738	1.1738	1.3693	1.1738	1.1738	1.3693	1.1738
5	30	1.3693	1.2241	1.1689	1.3693	1.1689	1.1706	1.2250	1.1706	1.1612	1.2926	1.1612
5	35	1.3693	1.2074	1.1735	1.2247	1.1735	1.1735	1.3323	1.1735	1.1588	1.2132	1.1588
5	40	1.3693	1.1902	1.1558	1.2247	1.1558	1.1640	1.2146	1.1640	1.1577	1.2222	1.1577
5	45	1.3693	1.1881	1.1560	1.2748	1.1560	1.1560	1.2748	1.1560	1.1560	1.2748	1.1560
5	50	1.3693	1.1830	1.1459	1.2247	1.1459	1.1492	1.3715	1.1492	1.1428	1.3652	1.1428
5	55	1.3693	1.1773	1.1490	1.2247	1.1490	1.1502	1.2642	1.1502	1.1475	1.2695	1.1475
5	60	1.3693	1.1734	1.1463	1.3693	1.1463	1.1487	1.2699	1.1487	1.1378	1.2282	1.1378
10	15	1.3608	1.3035	1.2347	1.2472	1.2347	1.1966	1.1995	1.1966	1.1871	1.1878	1.1871
10	20	1.3608	1.2862	1.1599	1.1706	1.1599	1.1419	1.1710	1.1419	1.1265	1.1265	1.1327
10	25	1.3608	1.2407	1.1564	1.1706	1.1564	1.1319	1.1319	1.1333	1.1365	1.1511	1.1365
10	30	1.3608	1.2241	1.1410	1.1706	1.1410	1.1373	1.1473	1.1373	1.1362	1.1564	1.1362
10	35	1.3608	1.2074	1.1482	1.1706	1.1482	1.1496	1.1672	1.1496	1.1362	1.1362	1.1374
10	40	1.3608	1.1902	1.1359	1.1386	1.1359	1.1427	1.1700	1.1427	1.1389	1.1462	1.1389
10	45	1.3608	1.1881	1.1364	1.1547	1.1364	1.1410	1.1497	1.1410	1.1371	1.1392	1.1371

Table 12 (continued)

n_1	n_2	$d(n_1)$	$d(n_2)$	Nested n_1-grid			Nested n_2-grid			Grid with nested axes		
				d	d_1	d_2	d	d_1	d_2	d	d_1	d_2
10	50	1.3608	1.1830	1.1547	1.1547	1.1599	1.1454	1.1606	1.1454	1.1382	1.1425	1.1382
10	55	1.3608	1.1773	1.1425	1.1547	1.1425	1.1425	1.1547	1.1425	1.1425	1.1547	1.1425
10	60	1.3608	1.1734	1.1222	1.1222	1.1268	1.1391	1.1647	1.1391	1.1298	1.2066	1.1298
15	20	1.3035	1.2862	1.2124	1.2124	1.2207	1.1741	1.1741	1.1886	1.1751	1.1751	1.1852
15	25	1.3035	1.2407	1.1510	1.1560	1.1510	1.1341	1.1341	1.1407	1.1451	1.1456	1.1451
15	30	1.3035	1.2241	1.1126	1.1139	1.1126	1.1059	1.1101	1.1059	1.1020	1.1150	1.1020
15	35	1.3035	1.2074	1.1394	1.1394	1.1535	1.1139	1.1163	1.1139	1.1178	1.1178	1.1197
15	40	1.3035	1.1902	1.1135	1.1225	1.1135	1.1117	1.1267	1.1117	1.1128	1.1128	1.1225
15	45	1.3035	1.1881	1.1207	1.1225	1.1207	1.1060	1.1303	1.1060	1.1074	1.1074	1.1105
15	50	1.3035	1.1830	1.1239	1.1309	1.1239	1.1184	1.1221	1.1184	1.1103	1.1157	1.1103
15	55	1.3035	1.1773	1.1218	1.1309	1.1218	1.1112	1.1219	1.1112	1.1223	1.1234	1.1223
15	60	1.3035	1.1734	1.1139	1.1139	1.1155	1.1156	1.1166	1.1156	1.1036	1.1036	1.1091
20	25	1.2862	1.2407	1.1987	1.1987	1.2162	1.1671	1.1671	1.1702	1.1747	1.1747	1.1765
20	30	1.2862	1.2241	1.1732	1.1732	1.1904	1.1401	1.1429	1.1401	1.1386	1.1386	1.1418
20	35	1.2862	1.2074	1.1152	1.1313	1.1152	1.1155	1.1155	1.1162	1.1336	1.1336	1.1368
20	40	1.2862	1.1902	1.0988	1.0988	1.1083	1.0912	1.0945	1.0912	1.0970	1.1097	1.0970
20	45	1.2862	1.1881	1.1249	1.1260	1.1249	1.1013	1.1108	1.1013	1.0952	1.0952	1.1015
20	50	1.2862	1.1830	1.1098	1.1098	1.1145	1.0971	1.0971	1.1000	1.0917	1.0917	1.0945
20	55	1.2862	1.1773	1.1062	1.1098	1.1062	1.1010	1.1010	1.1078	1.0931	1.0931	1.0963
20	60	1.2862	1.1734	1.0988	1.0988	1.1024	1.1123	1.1123	1.1156	1.0933	1.0933	1.0947
25	30	1.2407	1.2241	1.2024	1.2024	1.2192	1.1600	1.1600	1.1651	1.1670	1.1670	1.1685
25	35	1.2407	1.2074	1.1629	1.1629	1.1690	1.1297	1.1313	1.1297	1.1431	1.1431	1.1474
25	40	1.2407	1.1902	1.1407	1.1407	1.1548	1.1227	1.1251	1.1227	1.1193	1.1193	1.1204
25	45	1.2407	1.1881	1.0996	1.1143	1.0996	1.0951	1.0975	1.0951	1.1057	1.1057	1.1108
25	50	1.2407	1.1830	1.0969	1.0990	1.0969	1.0946	1.0963	1.0946	1.0817	1.0817	1.0827
25	55	1.2407	1.1773	1.1182	1.1182	1.1204	1.1021	1.1127	1.1021	1.0875	1.0898	1.0875
25	60	1.2407	1.1734	1.1182	1.1182	1.1193	1.0916	1.0957	1.0916	1.0869	1.0869	1.0895
30	35	1.2241	1.2074	1.1950	1.1950	1.1980	1.1522	1.1522	1.1605	1.1509	1.1513	1.1509
30	40	1.2241	1.1902	1.1596	1.1596	1.1784	1.1258	1.1258	1.1264	1.1255	1.1266	1.1255
30	45	1.2241	1.1881	1.1401	1.1401	1.1477	1.1076	1.1076	1.1091	1.1198	1.1198	1.1204
30	50	1.2241	1.1830	1.1061	1.1146	1.1061	1.1105	1.1147	1.1105	1.1123	1.1123	1.1130
30	55	1.2241	1.1773	1.0861	1.0972	1.0861	1.0987	1.0998	1.0987	1.0935	1.0956	1.0935
30	60	1.2241	1.1734	1.0943	1.0943	1.1022	1.0822	1.0822	1.0835	1.0824	1.0824	1.0831
35	40	1.2074	1.1902	1.1799	1.1799	1.2144	1.1567	1.1567	1.1622	1.1488	1.1493	1.1488
35	45	1.2074	1.1881	1.1670	1.1670	1.1905	1.1365	1.1367	1.1365	1.1266	1.1266	1.1340
35	50	1.2074	1.1830	1.1496	1.1496	1.1568	1.1184	1.1259	1.1184	1.1078	1.1078	1.1123
35	55	1.2074	1.1773	1.1341	1.1341	1.1360	1.1062	1.1062	1.1067	1.1097	1.1097	1.1100
35	60	1.2074	1.1734	1.1071	1.1071	1.1147	1.0952	1.0952	1.0976	1.0892	1.0892	1.0939
40	45	1.1902	1.1881	1.1780	1.1780	1.2141	1.1556	1.1556	1.1574	1.1450	1.1457	1.1450
40	50	1.1902	1.1830	1.1728	1.1728	1.1848	1.1243	1.1243	1.1262	1.1319	1.1329	1.1319
40	55	1.1902	1.1773	1.1485	1.1498	1.1485	1.1146	1.1174	1.1146	1.1146	1.1146	1.1166
40	60	1.1902	1.1734	1.1354	1.1354	1.1437	1.0986	1.0988	1.0986	1.1064	1.1064	1.1075
45	50	1.1881	1.1830	1.1736	1.1736	1.2056	1.1407	1.1407	1.1454	1.1454	1.1454	1.1471
45	55	1.1881	1.1773	1.1780	1.1780	1.2026	1.1225	1.1236	1.1225	1.1225	1.1225	1.1244
45	60	1.1881	1.1734	1.1455	1.1455	1.1494	1.1121	1.1121	1.1126	1.1112	1.1112	1.1152
50	55	1.1830	1.1773	1.1718	1.1718	1.1981	1.1449	1.1449	1.1480	1.1443	1.1443	1.1445
50	60	1.1830	1.1734	1.1606	1.1606	1.2065	1.1254	1.1256	1.1254	1.1152	1.1152	1.1163

Appendix C: Calculation times and percentage losses for GROUPRAND

Table 13 Mean and standard deviation of calculation times of GROUPRAND in minutes

n_1	n_2	Nested n_1-grid								Nested n_2-grid								Grid with nested axes							
		Mean				Std				Mean				Std				Mean				Std			
		4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d
5	10	0.1	0.3	0.4	0.3	0.0	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.0	0.1	0.1	0.1	0.1	0.3	0.4	0.3	0.1	0.1	0.1	0.1
5	20	1.6	4.3	4.9	3.6	1.2	1.8	1.7	1.4	0.8	1.7	1.8	2.4	0.4	0.5	0.4	0.7	1.4	2.1	2.9	3.7	0.8	0.7	0.6	2.4
5	30	8.0	22.6	21.4	15.0	5.0	6.9	5.0	5.1	2.9	6.1	7.0	7.1	1.2	2.4	2.0	2.0	5.9	9.3	9.2	7.6	1.8	3.3	2.4	3.4
5	40	19.2	40.6	51.5	23.7	14.8	5.7	9.6	3.4	5.6	8.8	9.6	12.3	1.6	2.6	1.5	2.8	13.3	13.1	14.9	13.7	5.1	2.0	2.6	4.8
5	50	27.0	90.0	118.2	66.0	15.2	19.4	17.9	11.5	9.1	16.3	17.0	18.0	3.4	5.0	4.6	3.3	16.9	21.1	27.6	26.5	4.6	3.8	3.9	9.4
5	60	26.4	176.8	216.5	118.5	5.4	47.7	30.7	16.3	16.0	25.7	32.8	47.4	8.9	13.9	12.5	12.0	28.4	37.2	50.7	46.5	8.8	9.1	8.8	14.3
10	15	0.1	0.6	0.8	0.7	0.0	0.2	0.2	0.3	0.2	0.4	0.6	0.8	0.1	0.1	0.1	0.3	0.4	0.8	1.2	1.0	0.2	0.4	0.5	0.4
10	20	0.3	1.5	2.6	2.1	0.1	0.5	1.1	0.8	0.5	0.8	1.2	1.7	0.3	0.2	0.4	0.6	0.8	1.4	2.0	1.9	0.3	0.5	0.7	0.8
10	30	1.5	8.3	10.4	5.3	0.4	2.6	4.8	1.5	1.6	3.4	3.7	3.6	0.5	1.3	1.1	1.1	3.9	6.2	6.9	4.6	1.2	1.8	2.2	2.1
10	40	4.4	13.3	18.2	7.4	1.4	4.9	7.7	1.2	6.1	5.2	6.4	6.8	3.1	1.2	1.8	1.3	8.9	12.4	13.3	8.2	4.4	6.9	5.4	3.8
10	50	5.6	23.1	24.8	14.4	1.9	12.0	7.4	4.8	5.9	10.6	11.2	13.3	2.5	3.2	3.6	3.1	10.5	15.1	13.9	15.3	3.0	5.9	3.5	5.6
10	60	7.0	31.2	37.8	23.1	4.2	7.4	10.3	4.9	8.0	14.5	24.8	27.9	3.5	2.1	9.2	6.3	19.2	17.3	25.9	30.3	9.6	4.8	7.7	11.3
15	20	0.3	1.6	2.2	1.3	0.1	0.4	0.6	0.4	0.4	1.2	1.5	2.2	0.1	0.5	0.4	0.4	1.1	1.9	2.3	3.0	0.3	0.6	0.5	0.8
15	30	1.2	6.6	5.9	4.5	0.4	2.3	1.6	1.5	0.9	3.1	3.1	3.2	0.2	1.0	1.0	0.8	2.3	3.6	4.4	3.6	0.9	1.1	1.1	1.1
15	40	2.0	8.1	8.9	6.2	0.8	2.9	3.2	2.9	2.4	6.9	5.9	5.3	1.3	2.7	1.3	1.1	7.9	10.1	9.6	8.7	3.6	4.0	4.3	3.6
15	50	3.4	13.2	15.8	8.6	1.5	4.8	7.4	2.5	4.5	9.4	12.3	10.1	2.3	4.1	5.2	2.5	14.0	14.6	14.1	13.8	6.3	7.1	6.1	4.6
15	60	4.7	23.5	25.0	15.7	2.4	13.6	7.3	4.8	6.5	13.5	13.4	18.4	2.8	9.3	2.6	3.2	12.4	20.0	20.8	20.5	5.1	9.1	5.6	6.2
20	25	0.6	3.1	3.1	3.4	0.2	1.6	1.1	1.6	0.8	2.4	3.4	4.0	0.2	0.4	0.9	1.0	2.2	3.5	5.3	5.2	0.7	0.8	1.1	3.2
20	30	1.2	6.2	7.2	3.8	0.4	2.3	2.8	1.3	1.3	4.9	4.6	4.2	0.6	1.4	1.6	1.4	3.4	8.5	8.0	7.1	1.2	3.1	2.8	3.4
20	40	2.1	7.9	8.8	4.4	0.5	2.3	3.1	1.3	1.8	5.3	5.3	4.4	0.6	2.3	2.6	1.5	4.5	5.5	5.4	5.5	1.4	1.7	1.0	2.1
20	50	3.2	12.8	16.5	8.9	1.5	5.9	8.6	2.8	4.3	9.1	9.6	10.6	2.0	3.8	2.4	1.9	7.4	20.9	18.8	15.0	1.4	10.4	7.2	4.1
20	60	3.0	20.3	17.9	11.7	1.2	7.9	6.8	2.8	5.3	11.4	13.0	15.6	2.6	4.8	4.4	3.4	10.8	14.5	17.4	16.4	5.6	3.2	4.1	4.2
25	30	1.2	5.7	6.9	4.4	0.3	1.0	2.7	1.0	1.3	5.2	5.0	3.9	0.2	2.6	1.2	0.9	4.0	7.0	6.8	5.6	1.4	2.1	1.8	2.3
25	40	2.4	9.2	10.3	8.2	0.6	2.7	3.0	3.5	3.1	6.6	6.8	6.2	1.0	3.6	2.2	1.3	8.1	10.7	17.2	12.2	2.7	6.4	6.2	4.9
25	50	2.8	13.7	17.7	8.4	0.7	6.3	7.7	3.7	3.9	6.0	6.8	7.6	0.9	1.8	1.6	2.5	8.7	7.9	8.7	8.8	3.6	2.4	1.7	2.1
25	60	5.4	19.3	25.6	22.0	2.7	9.1	9.3	6.4	5.0	15.5	14.2	20.4	2.5	10.2	4.7	5.5	18.3	16.3	23.4	25.9	10.1	2.7	8.4	7.0
30	35	2.0	8.4	9.6	6.6	0.9	2.5	1.7	2.7	2.4	5.3	4.8	5.7	0.7	1.9	1.0	1.7	6.6	7.9	7.6	7.4	3.1	2.7	1.9	2.6
30	40	2.6	10.4	13.4	10.6	0.7	3.1	5.3	3.0	3.0	5.9	7.0	6.4	1.3	1.4	2.3	1.1	7.2	9.8	10.7	10.3	2.2	2.7	3.2	4.2
30	50	3.3	16.1	19.4	15.1	0.3	4.8	4.1	4.5	5.2	9.7	10.9	10.3	3.1	3.5	2.9	2.9	12.3	14.5	22.3	19.7	5.6	5.3	5.8	8.9
30	60	5.8	21.3	27.8	18.3	1.8	6.5	11.4	8.7	7.3	12.9	13.0	15.3	2.5	7.5	3.6	3.8	12.2	11.6	18.1	16.7	4.8	2.6	1.9	4.7
35	40	2.8	9.4	12.9	9.9	0.9	2.4	3.6	3.0	3.2	6.6	6.7	5.9	0.7	1.7	1.7	1.7	9.2	8.5	10.2	8.4	3.2	2.7	3.2	2.7
35	50	4.3	18.9	24.6	20.2	0.7	6.1	5.7	6.9	4.3	10.5	13.2	11.9	2.0	3.6	5.4	1.9	9.4	13.1	17.5	16.4	3.3	2.4	2.9	5.1
35	60	5.6	26.5	32.6	29.2	1.7	13.0	9.4	11.1	7.6	14.8	14.5	20.1	3.1	8.2	3.8	3.2	10.6	27.1	27.3	30.2	1.8	10.5	9.6	7.4
40	45	6.1	16.0	16.2	13.9	2.8	6.6	7.3	3.8	3.6	8.1	8.7	8.1	2.1	2.3	2.6	1.4	8.2	9.1	11.4	12.0	2.7	2.4	1.7	2.7
40	50	9.9	28.4	29.0	18.7	4.6	11.8	9.1	3.9	4.4	11.4	10.2	10.6	2.3	4.6	1.6	1.9	10.8	15.0	15.3	13.5	3.5	3.5	2.8	3.9
40	60	6.0	30.5	42.5	33.2	2.9	7.7	10.4	10.0	7.6	16.2	24.2	23.2	2.7	5.8	6.1	3.9	16.9	23.5	31.8	29.7	8.6	4.7	5.6	5.0
45	50	5.3	19.3	22.8	20.9	1.6	6.9	4.2	6.6	5.1	10.8	9.4	10.6	1.9	3.7	3.6	1.9	8.8	13.8	16.6	13.5	1.2	5.8	3.3	4.2
45	60	9.0	40.6	52.8	35.8	3.0	10.4	17.4	15.5	7.9	16.5	19.2	24.1	3.9	6.5	6.0	4.0	14.6	24.1	29.9	29.2	5.3	3.7	1.3	7.3
50	55	10.4	29.6	33.9	35.8	4.0	10.7	6.6	8.1	7.9	16.8	12.3	15.1	3.3	5.5	2.2	3.3	13.6	17.1	22.6	20.4	5.8	3.5	4.3	6.7
50	60	13.0	40.4	48.4	30.9	6.9	10.3	17.8	12.5	10.1	17.3	20.7	23.3	5.2	6.0	4.7	3.7	14.9	22.3	30.4	27.2	5.6	6.0	2.8	6.0

Table 14 Mean and standard deviation of percentage loss $l_1(n_1, n_2)$ of GROUPRAND

n_1	n_2	Nested n_1-grid								Nested n_2-grid								Grid with nested axes							
		Mean				Std				Mean				Std				Mean				Std			
		4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d
5	10	6.06	10.21	10.85	11.49	6.80	3.71	1.85	2.65	−0.77	12.71	11.55	10.93	6.52	1.93	3.13	4.20	8.21	13.65	13.59	11.45	3.76	2.45	2.87	3.49
5	20	9.64	12.70	15.97	12.59	6.23	3.03	3.89	4.74	7.64	12.39	14.44	12.49	5.48	2.97	3.96	4.94	10.23	12.11	14.08	13.97	6.52	5.03	4.18	3.68
5	30	9.19	13.17	18.20	12.78	4.64	4.30	3.02	3.64	13.04	8.65	16.28	17.08	3.39	3.56	4.52	4.34	9.15	11.78	14.31	12.62	4.55	3.45	5.49	4.22
5	40	10.61	15.57	15.52	15.31	3.12	4.66	3.89	5.58	11.83	10.42	16.95	12.94	6.24	3.45	3.67	5.78	9.43	12.10	16.65	16.12	6.12	5.05	2.45	4.79
5	50	10.42	15.57	14.33	12.98	6.82	4.66	3.41	5.52	15.23	10.99	12.02	12.78	5.64	6.06	3.30	3.16	10.67	13.55	16.47	11.88	7.20	5.81	4.86	4.08
5	60	9.36	16.09	13.93	11.09	7.57	6.10	4.29	3.96	11.10	15.39	14.52	12.73	6.64	3.09	6.05	3.46	13.77	9.29	14.78	16.02	6.52	5.03	3.27	6.01
10	15	11.92	6.97	5.09	3.90	1.48	0.95	0.59	0.70	12.90	10.82	8.04	8.29	1.20	0.80	0.66	0.51	13.29	9.32	6.82	6.48	0.96	0.74	0.28	0.56
10	20	16.24	13.31	10.24	9.39	2.13	1.11	1.68	1.46	16.89	13.69	11.43	10.44	1.93	0.83	0.75	0.72	17.94	14.62	11.57	11.09	1.57	0.94	0.35	0.94
10	30	17.69	14.43	12.09	12.05	2.33	1.00	1.22	0.88	17.07	14.79	12.68	12.08	1.70	0.58	0.71	0.81	17.59	14.89	13.06	12.58	1.42	0.84	0.68	0.36
10	40	17.93	15.39	13.38	13.33	2.30	1.56	1.08	0.56	17.35	16.29	14.20	13.69	1.46	0.92	0.87	0.39	18.07	15.30	13.62	13.49	1.58	1.53	0.93	0.76
10	50	18.77	16.86	14.60	13.97	1.54	1.50	0.69	1.03	18.19	16.61	14.99	13.96	1.97	0.78	1.32	0.75	17.72	16.80	15.02	13.97	1.99	1.60	0.92	1.09
10	60	20.91	17.44	15.10	14.54	2.20	1.38	1.47	0.80	18.65	17.13	14.83	13.76	1.48	0.78	0.47	1.47	19.60	18.33	14.91	14.64	1.67	0.99	1.42	0.55
15	20	8.64	4.06	5.65	3.43	1.01	0.91	1.36	1.17	11.67	6.73	8.93	5.39	0.94	1.06	0.38	0.72	11.04	6.36	8.17	4.76	0.52	0.70	0.53	0.43
15	30	15.01	9.42	10.90	7.69	0.99	0.72	1.04	0.74	15.69	11.31	12.07	9.53	0.83	0.64	0.48	0.37	16.55	11.30	12.26	9.80	1.19	0.80	0.41	0.47
15	40	15.74	11.25	11.16	8.60	0.87	1.78	0.64	0.69	16.15	11.71	13.12	10.85	1.21	0.39	0.48	0.40	16.04	11.07	12.20	9.56	0.89	0.69	0.50	0.65
15	50	16.42	12.60	13.26	10.82	1.65	1.04	1.37	0.49	16.82	12.98	13.89	11.26	1.08	0.94	0.64	0.57	16.78	13.12	13.96	11.60	1.20	0.91	0.87	0.76
15	60	17.43	13.25	14.38	11.74	1.54	0.71	0.87	0.77	16.97	13.43	14.57	11.80	1.70	0.97	0.40	0.51	16.63	13.70	14.66	12.08	1.09	0.77	0.49	0.54
20	25	8.67	4.44	2.56	6.42	1.38	1.09	0.68	1.06	10.91	6.34	4.93	8.76	0.98	0.52	0.64	0.18	9.96	6.49	4.42	8.43	0.81	0.58	0.50	0.55
20	30	9.91	4.95	3.12	7.61	0.95	0.83	0.90	0.73	12.86	8.15	6.52	10.44	1.42	0.57	0.71	0.49	12.80	7.21	5.43	9.26	0.94	0.77	0.69	0.51
20	40	14.31	10.68	7.67	11.41	0.64	1.04	0.69	0.43	15.42	11.62	8.85	12.17	0.62	0.47	0.31	0.38	16.20	11.36	9.15	12.44	0.73	0.34	0.33	0.16
20	50	14.65	10.37	6.36	10.06	0.77	1.48	1.06	0.54	15.46	12.23	9.91	13.00	0.75	0.72	0.51	0.20	16.35	10.92	8.65	12.15	0.76	0.77	0.63	0.59
20	60	17.58	12.41	10.23	13.37	1.58	0.94	0.67	0.49	16.30	13.02	10.34	13.78	1.94	0.78	0.63	0.41	17.09	12.93	10.31	13.72	0.85	0.62	0.57	0.46
25	30	4.88	3.29	2.30	2.37	1.39	0.48	0.51	0.37	7.57	5.57	3.95	4.49	1.15	0.73	0.27	0.23	7.40	5.95	4.82	4.10	1.08	0.39	2.99	0.49
25	40	8.91	5.53	3.44	2.96	0.89	0.51	0.48	0.51	10.61	9.33	7.15	6.47	0.63	0.53	0.55	0.39	10.49	8.09	5.44	5.98	0.41	1.17	0.67	2.14
25	50	13.07	10.28	7.14	7.11	0.60	1.14	0.80	0.79	12.82	11.25	8.62	7.82	0.66	0.45	0.59	0.46	13.75	11.52	8.59	7.94	0.54	0.82	0.39	0.35
25	60	12.73	9.61	5.73	4.57	1.46	1.49	0.97	0.50	13.93	11.32	9.28	8.67	0.96	0.62	0.63	0.76	13.36	11.85	9.39	7.89	0.70	0.74	0.88	0.37
30	35	3.99	2.68	3.44	2.35	1.00	0.72	0.48	0.45	6.50	5.03	5.87	4.01	0.59	0.69	1.01	0.39	6.65	5.50	5.66	3.93	0.42	0.78	0.31	0.46
30	40	5.74	3.66	3.77	2.53	0.53	0.57	0.56	0.50	8.71	7.57	7.07	5.46	0.68	0.61	0.33	0.30	8.80	6.54	6.61	4.82	0.64	0.31	0.60	0.41
30	50	9.80	5.93	4.83	3.06	0.32	0.50	0.43	0.28	10.85	9.22	8.38	6.68	1.12	0.43	0.34	0.48	10.33	8.08	7.36	5.64	0.78	0.71	0.87	0.38
30	60	11.72	9.51	8.64	6.58	0.85	0.51	0.93	0.52	12.30	10.38	9.67	7.71	0.69	0.88	0.35	0.21	12.79	11.12	9.61	7.54	0.84	0.50	0.15	0.36
35	40	3.42	3.06	2.69	2.21	1.00	0.89	0.44	0.40	5.19	4.80	4.71	4.27	0.67	0.68	0.32	0.26	5.76	5.51	4.69	3.85	0.48	0.81	0.60	0.41
35	50	6.20	3.99	3.00	2.51	0.67	1.01	0.43	0.35	9.13	7.87	6.97	5.77	1.24	0.68	0.68	0.22	8.90	7.59	6.33	5.34	0.55	0.69	0.34	0.33
35	60	9.35	6.46	4.63	3.40	0.94	0.59	0.58	0.48	10.02	9.08	8.22	6.77	0.64	0.79	0.59	0.24	10.45	7.69	7.31	6.00	0.39	0.83	1.03	0.29
40	45	2.31	3.66	2.86	3.99	0.89	0.64	0.57	0.33	4.57	6.09	5.15	6.03	1.07	0.59	1.20	1.45	4.78	6.77	4.93	5.68	0.64	0.55	0.30	0.30
40	50	3.01	4.07	3.44	4.45	0.96	0.90	1.47	0.20	6.78	7.77	6.28	6.89	0.87	0.55	0.34	0.41	6.46	7.34	5.79	6.86	0.89	0.78	0.36	0.37
40	60	6.01	5.34	3.95	4.56	0.75	0.67	1.23	0.31	8.30	9.11	7.21	7.89	0.70	0.59	0.43	0.27	8.42	8.63	6.86	7.37	0.85	0.67	0.42	0.40
45	50	2.43	3.59	1.98	1.61	0.71	0.60	0.43	0.30	4.62	5.67	4.07	3.55	0.85	0.42	0.65	0.18	5.02	5.99	4.14	3.70	0.50	0.55	0.24	1.26
45	60	4.30	4.41	2.24	2.13	0.36	0.45	0.60	0.39	7.64	8.23	5.77	4.93	0.62	0.52	0.52	0.37	7.82	7.60	5.62	4.58	0.54	0.37	0.27	0.27
50	55	2.48	1.41	1.56	1.71	0.91	0.59	0.32	0.32	4.49	3.66	3.59	3.69	1.05	0.62	0.48	0.40	4.60	3.91	3.44	3.44	0.54	0.52	0.33	0.42
50	60	3.17	1.72	2.02	2.49	0.62	0.62	0.62	0.55	5.90	5.03	4.50	4.36	0.92	0.61	0.43	0.19	6.29	5.38	4.50	4.36	0.50	0.51	0.35	0.38

Table 15 Mean and standard deviation of percentage loss $l_2(n_1, n_2)$ of GROUPRAND

n_1	n_2	Nested n_1-grid								Nested n_2-grid								Grid with nested axes							
		Mean				Std				Mean				Std				Mean				Std			
		4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d	4d	6d	8d	10d
5	10	13.46	8.35	4.99	5.54	6.80	3.71	1.85	2.65	11.25	10.20	7.87	8.21	6.52	1.93	3.13	4.20	12.58	10.84	6.91	7.75	3.76	2.45	2.87	3.49
5	20	11.46	7.70	6.42	11.04	6.23	3.03	3.89	4.74	11.10	7.66	6.44	10.25	5.48	2.97	3.96	4.94	11.70	8.28	6.53	10.46	6.52	5.03	4.18	3.68
5	30	7.06	6.16	6.59	5.33	4.64	4.30	3.02	3.64	7.03	6.49	6.83	5.56	3.39	3.56	4.52	4.34	7.35	5.82	6.98	5.52	4.55	3.45	5.49	4.22
5	40	5.25	7.34	6.07	7.46	3.12	4.66	3.89	5.58	5.14	7.46	6.37	7.22	6.24	3.45	3.67	5.78	4.97	6.95	6.26	7.00	6.12	5.05	2.45	4.79
5	50	6.11	4.91	4.52	4.53	6.82	4.66	3.41	5.52	5.59	5.18	5.17	5.29	5.64	6.06	3.30	3.16	5.21	5.03	4.51	4.64	7.20	5.81	4.86	4.08
5	60	5.16	4.49	4.61	3.94	7.57	6.10	4.29	3.96	4.74	5.53	5.45	4.50	6.64	3.09	6.05	3.46	4.69	4.68	4.88	4.44	6.52	5.03	3.27	6.01
10	15	7.06	0.68	1.80	−0.86	1.48	0.95	0.59	0.70	10.26	6.56	6.92	4.81	1.20	0.80	0.66	0.51	9.54	4.46	5.15	2.88	0.96	0.74	0.28	0.56
10	20	12.30	7.81	4.59	7.92	2.13	1.11	1.68	1.46	13.63	8.77	6.48	9.70	1.93	0.83	0.75	0.72	14.00	9.34	6.18	9.92	1.57	0.94	0.35	0.94
10	30	8.89	6.60	5.94	4.55	2.33	1.00	1.22	0.88	8.93	7.22	7.07	5.46	1.70	0.58	0.71	0.81	8.82	7.54	7.97	5.43	1.42	0.84	0.68	0.36
10	40	6.47	7.83	6.15	7.03	2.30	1.56	1.08	0.56	6.57	8.09	6.82	7.41	1.46	0.92	0.87	0.39	6.85	7.44	6.18	7.31	1.58	1.53	0.93	0.76
10	50	6.61	6.02	5.15	4.85	1.54	1.50	0.69	1.03	6.54	5.86	5.71	4.95	1.97	0.78	1.32	0.75	7.32	6.05	5.54	5.06	1.99	1.60	0.92	1.09
10	60	8.25	6.33	6.00	4.76	2.20	1.38	1.47	0.80	6.31	6.27	5.45	4.73	1.48	0.78	0.47	1.47	6.83	7.09	6.09	4.73	1.67	0.99	1.42	0.55
15	20	5.54	−0.77	−2.44	2.41	1.01	0.91	1.36	1.17	10.56	5.80	4.29	7.19	0.94	1.06	0.38	0.72	9.92	5.18	3.57	6.47	0.52	0.70	0.53	0.43
15	30	9.78	6.04	5.79	3.29	0.99	0.72	1.04	0.74	10.66	7.95	7.35	5.52	0.83	0.64	0.48	0.37	11.27	7.95	7.53	5.61	1.19	0.80	0.41	0.47
15	40	7.82	7.32	4.56	5.03	0.87	1.78	0.64	0.69	8.24	8.25	6.83	7.60	1.21	0.39	0.48	0.40	7.82	7.21	5.68	6.07	0.89	0.69	0.50	0.65
15	50	7.69	5.71	4.84	4.71	1.65	1.04	1.37	0.49	8.23	6.36	5.79	5.34	1.08	0.94	0.64	0.57	8.07	6.45	5.75	6.31	1.20	0.91	0.87	0.76
15	60	8.10	6.19	6.11	4.90	1.54	0.71	0.87	0.77	7.70	6.53	6.34	5.16	1.70	0.97	0.40	0.51	7.76	6.77	6.40	5.29	1.09	0.77	0.49	0.54
20	25	2.08	−0.65	−1.60	−1.49	1.38	1.09	0.68	1.06	7.32	4.76	3.66	3.34	0.98	0.52	0.64	0.18	6.22	4.86	3.14	2.93	0.81	0.58	0.50	0.55
20	30	4.09	0.91	0.87	−0.55	0.95	0.83	0.90	0.73	8.86	5.60	5.90	4.40	1.42	0.57	0.71	0.49	8.10	4.50	4.69	2.98	0.94	0.77	0.69	0.51
20	40	8.21	7.80	5.48	6.27	0.64	1.04	0.69	0.43	8.87	9.04	6.76	7.14	0.62	0.47	0.31	0.38	9.49	8.58	7.04	7.39	0.73	0.34	0.33	0.16
20	50	6.97	4.28	2.02	1.83	0.77	1.48	1.06	0.54	8.55	6.72	5.99	5.26	0.75	0.72	0.51	0.20	9.01	4.96	4.53	4.23	0.76	0.77	0.63	0.59
20	60	9.48	6.18	6.20	4.76	1.58	0.94	0.67	0.49	8.36	6.93	6.37	5.28	1.94	0.78	0.63	0.41	9.02	6.85	6.26	5.21	0.85	0.62	0.57	0.46
25	30	1.31	−1.06	0.51	−0.32	1.39	0.48	0.51	0.37	6.14	4.08	4.28	3.65	1.15	0.73	0.27	0.23	5.88	4.48	5.16	3.14	1.08	0.39	2.99	0.49
25	40	4.41	3.25	1.75	2.40	0.89	0.51	0.48	0.51	7.05	7.78	6.21	6.62	0.63	0.53	0.55	0.39	6.53	6.44	4.39	6.01	0.41	1.17	0.67	2.14
25	50	9.74	5.81	4.24	4.48	0.60	1.14	0.80	0.79	8.81	6.99	5.85	5.27	0.66	0.45	0.59	0.46	9.52	7.28	5.77	5.33	0.54	0.82	0.39	0.35
25	60	7.28	4.48	2.49	0.85	1.46	1.49	0.97	0.50	9.30	6.70	6.48	5.39	0.96	0.62	0.63	0.76	8.49	7.11	6.50	4.44	0.70	0.74	0.88	0.37
30	35	−0.78	−0.89	0.06	0.28	1.00	0.72	0.48	0.45	4.75	4.01	4.43	3.43	0.59	0.69	1.01	0.39	4.88	4.48	4.22	3.42	0.42	0.78	0.31	0.46
30	40	1.63	1.05	−0.47	1.05	0.53	0.57	0.56	0.50	6.12	7.25	5.55	6.40	0.68	0.61	0.33	0.30	6.11	5.99	5.01	5.66	0.64	0.31	0.60	0.41
30	50	6.62	2.07	0.94	0.78	0.32	0.50	0.43	0.28	7.95	6.14	5.07	4.89	1.12	0.43	0.34	0.48	7.17	4.72	3.83	3.70	0.78	0.71	0.87	0.38
30	60	8.51	5.99	5.22	3.95	0.85	0.51	0.93	0.52	8.66	6.83	6.31	5.11	0.69	0.88	0.35	0.21	9.04	7.54	6.22	4.95	0.84	0.50	0.15	0.36
35	40	−1.20	1.38	0.86	2.29	1.00	0.89	0.44	0.40	3.33	5.31	4.48	5.57	0.67	0.68	0.32	0.26	4.28	5.85	4.44	5.20	0.48	0.81	0.60	0.41
35	50	3.01	0.44	−0.43	−0.24	0.67	1.01	0.43	0.35	7.55	5.55	4.96	4.38	1.24	0.68	0.68	0.22	6.83	5.16	4.24	3.94	0.55	0.69	0.34	0.33
35	60	6.75	3.53	2.21	0.95	0.94	0.59	0.58	0.48	7.60	6.55	6.23	4.65	0.64	0.79	0.59	0.24	7.77	4.84	5.15	3.74	0.39	0.83	1.03	0.29
40	45	−0.56	0.87	0.07	0.00	0.89	0.64	0.57	0.33	4.07	5.13	3.86	3.22	1.07	0.59	1.20	1.45	4.42	5.81	3.61	2.90	0.64	0.55	0.30	0.30
40	50	0.15	−1.43	−0.96	−0.64	0.96	0.90	1.47	0.20	6.12	4.80	4.35	4.12	0.87	0.55	0.34	0.41	5.77	4.35	3.80	4.10	0.89	0.78	0.36	0.37
40	60	4.04	1.33	1.32	0.15	0.75	0.67	1.23	0.31	7.11	5.90	5.28	4.39	0.70	0.59	0.43	0.27	6.99	5.30	4.84	3.78	0.85	0.67	0.42	0.40
45	50	−0.30	−0.39	−0.05	0.51	0.71	0.60	0.43	0.30	3.58	3.47	3.24	3.47	0.85	0.42	0.65	0.18	4.27	3.75	3.39	3.57	0.50	0.55	0.24	1.26
45	60	1.99	0.11	0.03	−0.46	0.36	0.45	0.60	0.39	6.52	5.78	5.04	4.08	0.62	0.52	0.52	0.37	6.57	5.04	4.84	3.73	0.54	0.37	0.27	0.27
50	55	−0.25	−0.20	0.07	0.30	0.91	0.59	0.32	0.32	3.62	3.47	3.23	3.18	1.05	0.62	0.48	0.40	3.80	3.85	3.09	2.94	0.54	0.52	0.33	0.42
50	60	0.01	−0.93	−0.27	−0.18	0.62	0.62	0.62	0.55	4.96	4.59	4.43	3.55	0.92	0.61	0.43	0.19	5.49	4.96	4.41	3.52	0.50	0.51	0.35	0.38

References

Barthelemy JFM, Haftka RT (1993) Approximation concepts for optimum structural design—a review. Struct Multidisc Optim 5(3):129–144

Booker AJ, Dennis JE, Frank PD, Serafini DB, Torczon V, Trosset MW (1999) A rigorous framework for optimization of expensive functions by surrogates. Struct Multidisc Optim 17(1):1–13

Cherkassky V, Mulier F (1998) Learning from data: concepts, theory, and methods. Wiley, New York

Cressie NAC (1993) Statistics for spatial data (revised ed), vol 605. Wiley, New York

Den Hertog D, Stehouwer HP (2002) Optimizing color picture tubes by high-cost nonlinear programming. Eur J Oper Res 140(2):197–211

Forrester AIJ, Keane AJ, Bressloff NW (2006) Design and analysis of "noisy" computer experiments. AIAA J 44(10): 2331–2339

Forrester AIJ, Sóbester A, Keane AJ (2007) Multi-fidelity optimization via surrogate modelling. In: Proceedings of the royal society a: mathematical, physical and engineering sciences, vol 463. The Royal Society, London, pp 3251–3269

Forrester AIJ, Sóbester A, Keane AJ (2008) Engineering design via surrogate modelling: a practical guide. Wiley, Chichester

Goel T, Haftka RT, Shyy W, Watson LT (2008) Pitfalls of using a single criterion for selecting experimental designs. Int J Numer Methods Eng 75(2):127–155

Grosso A, Jamali ARMJU, Locatelli M (2009) Finding maximin Latin hypercube designs by iterated local search heuristics. Eur J Oper Res 197(2):541–547

Husslage BGM, Van Dam ER, Den Hertog D, Stehouwer HP, Stinstra ED (2003) Collaborative metamodeling: coordinating simulation-based product design. Concurr Eng Res Appl 11(4):267–278

Husslage BGM, Van Dam ER, Den Hertog D (2005) Nested maximin Latin hypercube designs in two dimensions. CentER Discussion Paper 2005-79. Tilburg University, Tilburg, pp 1–11

Husslage BGM, Rennen G, Van Dam ER, Den Hertog D (2008) Space-filling Latin hypercube designs for computer experiments. CentER Discussion Paper 2008-104. Tilburg University, Tilburg, pp 1–14

Jin R, Chen W, Sudjianto A (2002) On sequential sampling for global metamodeling in engineering design. In: Proceedings of the ASME 2002 design engineering technical conferences and computers and information in engineering conference. Montreal, pp 1–10

Jin R, Chen W, Sudjianto A (2005) An efficient algorithm for constructing optimal design of computer experiments. J Stat Plan Inference 134(1):268–287

Johnson ME, Moore LM, Ylvisaker D (1990) Minimax and maximin distance designs. J Stat Plan Inference 26:131–148

Jones DR (2001) A taxonomy of global optimization methods based on response surfaces. J Glob Optim 21(4):345–383

Kennedy MC, O'Hagan A (2000) Predicting the output from a complex computer code when fast approximations are available. Biometrika 87(1):1–13

Kleijnen JPC (2008) Design and analysis of simulation experiments. In: International series in operations research & management science, vol 111. Springer, New York

Montgomery DC (1984) Design and analysis of experiments, 2nd ed. Wiley, New York

Morris MD, Mitchell TJ (1995) Exploratory designs for computer experiments. J Stat Plan Inference 43:381–402

Myers RH (1999) Response surface methodology—current status and future directions. J Qual Technol 31:30–74

Qian Z, Seepersad CC, Joseph VR, Allen JK, Wu CFJ (2006) Building surrogate models based on detailed and approximate simulations. J Mech Des 128(4):668–677

Queipo NV, Haftka RT, Shyy W, Goel T, Vaidyanathan R, Tucker PK (2005) Surrogate-based analysis and optimization. Prog Aerosp Sci 41(1):1–28

Sacks J, Schiller SB, Welch WJ (1989a) Designs for computer experiments. Technometrics 31:41–47

Sacks J, Welch WJ, Mitchell TJ, Wynn HP (1989b) Design and analysis of computer experiments. Stat Sci 4:409–435

Santner ThJ, Williams BJ, Notz WI (2003) The design and analysis of computer experiments. Springer Series in Statistics. Springer, New York

Simpson TW, Booker AJ, Ghosh D, Giunta AA, Koch PN, Yang R-J (2004) Approximation methods in multidisciplinary analysis and optimization: a panel discussion. Struct Multidisc Optim 27(5):302–313

Simpson TW, Toropov VV, Balabanov V, Viana FAC (2008) Design and analysis of computer experiments in multidisciplinary design optimization: a review. In: Proceedings of the 12th AIAA/ISSMO multidisciplinary analysis and optimization conference, pp 1–22

Sobieszczanski-Sobieski J, Haftka RT (1997) Multidisciplinary aerospace design optimization: survey of recent developments. Struct Multidisc Optim 14(1):1–23

Van Dam ER, Husslage BGM, Den Hertog D, Melissen JBM (2007) Maximin Latin hypercube designs in two dimensions. Oper Res 55(1):158–169

Van Dam ER, Husslage BGM, Den Hertog D (2009a) One-dimensional nested maximin designs. J Glob Optim (in press)

Van Dam ER, Rennen G, Husslage BGM (2009b) Bounds for maximin Latin hypercube designs. Oper Res 57:595–608

Viana FAC, Balabanov V, Venter G, Garcelon J, Steffen V (2007) Generating optimal Latin hypercube designs in real time. In: 7th world congress on structural and multidisciplinary optimization, pp 2310–2315

Wang GG, Shan S (2007) Review of metamodeling techniques in support of engineering design optimization. J Mech Des 129(4):370–380

Ye KQ, Li W, Sudjianto A (2000) Algorithmic construction of optimal symmetric Latin hypercube designs. J Stat Plan Inference 90(1):145–159