Optimal Sliced Latin Hypercube Designs

Shan Ba, William R. Myers, and William A. Brenneman

The Procter and Gamble Company,
Mason, OH 45040
(*ba.s@pg.com*; *myers.wr@pg.com*; *brenneman.wa@pg.com*)

Sliced Latin hypercube designs (SLHDs) have important applications in designing computer experiments with continuous and categorical factors. However, a randomly generated SLHD can be poor in terms of space-filling, and based on the existing construction method that generates the SLHD column by column using sliced permutation matrices, it is also difficult to search for the optimal SLHD. In this article, we develop a new construction approach that first generates the small Latin hypercube design in each slice and then arranges them together to form the SLHD. The new approach is intuitive and can be easily adapted to generate orthogonal SLHDs and orthogonal array-based SLHDs. More importantly, it enables us to develop general algorithms that can search for the optimal SLHD efficiently.

KEY WORDS: Computer experiment; Continuous and categorical factors; Maximin distance criterion; Space-filling design.

1. INTRODUCTION

Computer experiments, which simulate real-world phenomena using computational methods such as finite element analysis or computational fluid dynamics, are becoming widely used in almost every field of scientific research and product development. Different from conducting experiments physically, computer simulations are usually deterministic (not subject to random error) and are often characterized by a large number of input factors whereas only few of them are important. As a result, replication is unnecessary in computer experiments and a good design should be noncollapsing, in the sense that when the design points are projected onto the subspace of these few important factors, replications can be avoided (Santner, Williams, and Notz 2003; Fang, Li, and Sudjianto 2005). These requirements are perfectly met by the *Latin hypercube designs* (LHDs; McKay, Beckman, and Conover 1979), which are most popularly used in designing computer experiments with *continuous (quantitative) inputs* and possess the desirable property that when projecting an n-run LHD onto any factor, it always achieves n different levels for that variable. Suppose we denote the n levels of a factor by $1, \dots, n$, then an n-run LHD in p factors can be generated by using a random permutation of $\{1, \dots, n\}$ for each column of its $(n \times p)$ design matrix.

The motivation of this work originates from two computer experiments from The Procter & Gamble Company, whose inputs not only contain continuous variables but also include one or more *categorical (qualitative) variables*. Examples of categorical variables are the type or shape of equipment, the presence or absence of a part or when a continuous variable is restricted to two or three discrete levels. The presence of such categorical inputs makes it quite challenging to design computer experiments because the categorical factor usually cannot accommodate as many as n different levels and consequently the LHD structure cannot be directly applied.

For computer experiments with both continuous and categorical inputs, Qian (2012) proposed generating a *sliced Latin hypercube design* (SLHD) for the p continuous factors. An n-run SLHD is a special type of LHD that can be partitioned into t slices (blocks), each of which is also an LHD containing n/t runs, and then each slice can be used under one of the t different level combinations of categorical factors. This structure guarantees the maximum one-dimension uniformity of continuous factors in the whole design as well as in each slice. For example, with two continuous variables ($p = 2$) and one categorical variable with three levels ($t = 3$), a 12-run SLHD $\boldsymbol{X}$ for the two continuous factors is shown (in transpose) in (1), where the solid lines divide $\boldsymbol{X}$ into three slices $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, and $\boldsymbol{X}_3$. Obviously, $\boldsymbol{X}$ itself is an LHD with 12 different levels and for each $c = 1, \dots, 3$, $\lceil \boldsymbol{X}_c/3 \rceil$ is a smaller LHD with four levels, where $\lceil\,\rceil$ denotes the ceiling function such that $\lceil a \rceil$ equals the smallest integer no less than a. After assigning each slice to one level of the categorical variable, the full design for both the continuous and categorical inputs can be easily obtained as $[\boldsymbol{X}, \boldsymbol{C}]$. The benefit of using an SLHD is manifest. For instance, even if the categorical factor turns out to be insignificant, the design projected onto the remaining continuous factors still forms a 12-run LHD with no projection redundancy.

$$\boldsymbol{X}^T = \left[\begin{array}{cccc|cccc|cccc} 7 & 12 & 1 & 6 & 9 & 2 & 10 & 5 & 3 & 4 & 11 & 8 \\ 4 & 9 & 3 & 11 & 1 & 6 & 12 & 7 & 10 & 2 & 5 & 8 \end{array}\right],$$
$$\boldsymbol{C}^T = \left[\begin{array}{cccc|cccc|cccc} 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \end{array}\right]. \qquad (1)$$

A construction method was provided by Qian (2012) to *randomly* generate an n-run SLHD in t slices with each slice con-

TECHNOMETRICS, NOVEMBER 2015, VOL. 57, NO. 4
DOI: 10.1080/00401706.2014.957867
Color versions of one or more of the figures in the article can be found online at *www.tandfonline.com/r/tech*.

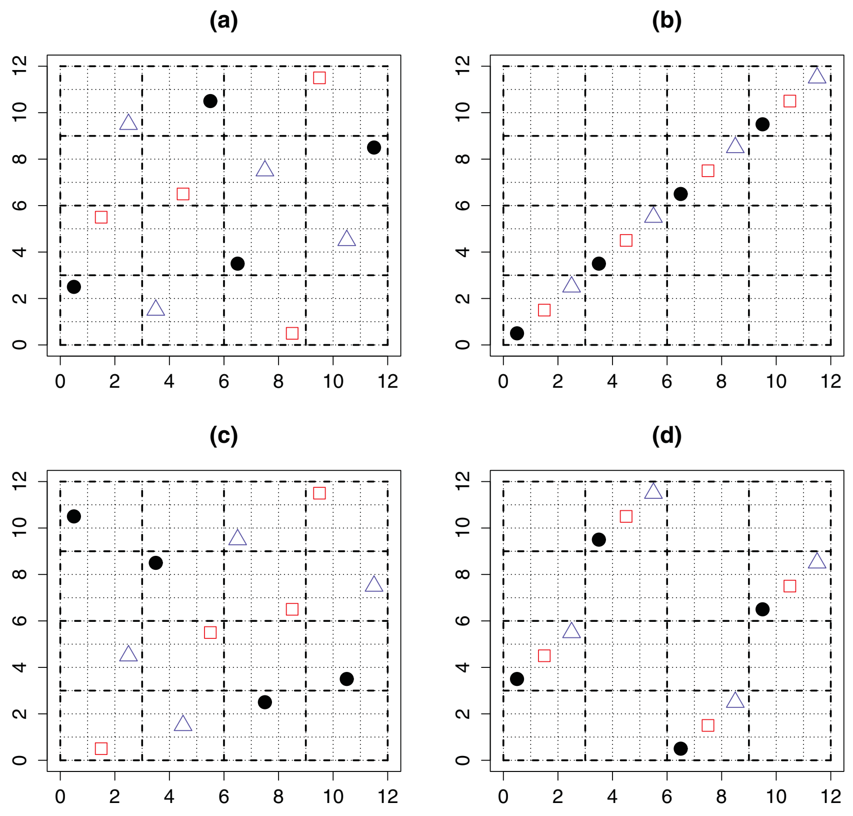

Figure 1. Several random SLHDs ($n = 12$, $p = 2$, $m = 4$, $t = 3$), where symbols (○, □, △) represent points belonging to the three different slices.

sisting of m points ($n = mt$). To impose the sliced structure, Qian (2012) proposed determining the p random permutations of $\{1, \ldots, n\}$ for each column of the SLHD by p-independent *sliced permutation matrices* (SPMs) $\boldsymbol{H}_1, \ldots, \boldsymbol{H}_p$. In this way, the construction effort of SLHD goes into the generation of p such SPMs, where each $\boldsymbol{H}_j$ is defined to be an $(m \times t)$ matrix containing $\{1, \ldots, n\}$ as its elements and each column of $\lceil \boldsymbol{H}_j/t \rceil$ forms a permutation of $\{1, \ldots, m\}$. Qian (2012) proposed randomly generating each SPM in three steps: (i) fill in the ith ($i = 1, \ldots, m$) row of SPM with $\{a \in \{1, \ldots, n\} | \lceil a/t \rceil = i\}$; (ii) randomly permute the entries in each row of SPM; (iii) randomly permute the entries in each column of SPM. After stacking the columns of each $\boldsymbol{H}_j$ into a single vector, Qian (2012) showed that it can be used as the jth column in the SLHD design matrix.

Figure 1(a) depicts the 12-run two-dimensional SLHD matrix $\boldsymbol{X}$ in (1), whose two columns correspond to the following two SPMs:

$$\boldsymbol{H}_1 = \begin{bmatrix} 7 & 9 & 3 \\ 12 & 2 & 4 \\ 1 & 10 & 11 \\ 6 & 5 & 8 \end{bmatrix}, \qquad \boldsymbol{H}_2 = \begin{bmatrix} 4 & 1 & 10 \\ 9 & 6 & 2 \\ 3 & 12 & 5 \\ 11 & 7 & 8 \end{bmatrix}. \tag{2}$$

In Figure 1(b)–1(d), we also provide three other random SLHDs of the same size that are produced by different random SPMs. Although they also achieve maximum one-dimensional uniformity, all of them are inferior to the design in Figure 1(a) for various reasons. In Figure 1(b), the two factors are perfectly correlated in the whole design as well as in each slice. If this design was used, the effects of the two factors would become indistinguishable. In Figure 1(c), although overall the points have been spread out in the design region, in one slice (denoted by symbol □) the two factors are still highly correlated. This design is undesirable because if the responses across different categories (slices) have very weak correlation, then the model cannot borrow too much information among the data from different categories resulting in poor predictions based mainly on this slice. When it comes to Figure 1(d), the small design in each slice is desirable, but the design points for the whole SLHD get clustered together, which leaves a large area of the design region unexplored. This design would be very inefficient if the categorical variable turns out to be insignificant and the design points need to be projected onto the subspace of continuous variables. In general, a desirable SLHD for computer experiments should have its design points well spread out for the whole design as well as for each slice. However, for an n-run design consisting of t slices with each slice containing m points in p factors, there can be as many as $((m!)^t(t!)^m)^p$ different SLHDs. How to avoid the undesirable ones such as in Figure 1(b)–1(d) and how to *efficiently* find the optimal SLHD is the main focus of this article.

Extensive work has been done in the literature on finding "good" LHDs. Due to the huge combinatorial nature of such a problem, a popular strategy is to use an exchange algorithm to iteratively search in the design space and optimize the LHD based on some criteria. For example, in a simulated annealing algorithm proposed by Morris and Mitchell (1995), two randomly chosen elements within a randomly selected column are exchanged in each step to find a new design. Other exchange algorithms have been discussed by Jin, Chen, and Sudjianto (2005) and Joseph and Hung (2008). Extending these algorithms to find optimal SLHDs, however, is not straightforward. This is because in the construction method proposed in Qian (2012), permuting elements in any row of an SPM *must precede* the random shuffling of its column elements. After elements in any column of an SPM are permuted, the elements in its rows can no longer be freely exchanged. This constraint on the order of permutations creates obstacles for freely generating new designs in the *neighborhood* of an existing SLHD, which prohibits the implementation of exchange algorithms for combinatorial optimization. For example, suppose we want to further improve the current SLHD shown in Figure 1(a). If we choose to search in the neighboring design region by exchanging elements "7" and "3" in the first row of $\boldsymbol{H}_1$ in (2), it would immediately violate the definition of SPM and thus cannot generate a valid candidate of the SLHD.

$$
\boldsymbol{X} = \left(\begin{array}{l} \boldsymbol{X}_1 = \left[\begin{array}{cc} 3 & 2 \\ 4 & 3 \\ 1 & 1 \\ 2 & 4 \end{array}\right] \\ \boldsymbol{X}_2 = \left[\begin{array}{cc} 3 & 1 \\ 1 & 2 \\ 4 & 4 \\ 2 & 3 \end{array}\right] \\ \boldsymbol{X}_3 = \left[\begin{array}{cc} 1 & 4 \\ 2 & 1 \\ 4 & 2 \\ 3 & 3 \end{array}\right] \end{array} \right)
\xrightarrow[\substack{\text{In column 2:} \\ \text{three 1's} \to \{3,1,2\} \\ \text{three 2's} \to \{4,6,5\} \\ \text{three 3's} \to \{9,7,8\} \\ \text{three 4's} \to \{11,12,10\}}]{\substack{\text{In column 1:} \\ \text{three 1's} \to \{1,2,3\} \\ \text{three 2's} \to \{6,5,4\} \\ \text{three 3's} \to \{7,9,8\} \\ \text{three 4's} \to \{12,10,11\}}}
\boldsymbol{X} = \left(\begin{array}{cc} 7 & 4 \\ 12 & 9 \\ 1 & 3 \\ 6 & 11 \\ \hline 9 & 1 \\ 2 & 6 \\ 10 & 12 \\ 5 & 7 \\ \hline 3 & 10 \\ 4 & 2 \\ 11 & 5 \\ 8 & 8 \end{array} \right).
$$

In this work, we develop a new construction method for the SLHD that is not based on the SPMs. Unlike in Qian (2012) that generates the whole design column by column, we propose to first construct the small LHD for each slice and then arrange their levels to form an overall SLHD. This new construction method is intuitive and it can be used to generate SLHDs with many desirable properties. The rest of this article is organized as follows. In Section 2, we present the new construction approach for the SLHD and also point out two of its interesting variants that can generate SLHDs with orthogonal columns or with higher-dimensional uniformity. In Section 3, a general algorithm for searching for the optimal SLHD is proposed, and we also develop strategies to substantially improve its efficiency. Two examples for designing computer experiments at The Procter & Gamble Company are provided in Section 4, and finally we give some concluding remarks in Section 5.

2. A NEW CONSTRUCTION METHOD

In this section, we present a simple and intuitive method to construct an n-run SLHD with t slices, where each slice contains m points in p factors (p, n, t, m are positive integers and $n = mt$). This construction method consists of only two steps:

Step 1. Independently generate t small LHDs $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_t$ for the t slices, where each LHD contains m points in p factors. Denote their factor levels by $1, \ldots, m$ and stack them to form an $(n \times p)$ matrix $\boldsymbol{X} = \bigcup_{i=1}^{t} \boldsymbol{X}_i$.

Step 2. Independently in each column of the matrix $\boldsymbol{X}$, replace the t entries of level l ($l = 1, \ldots, m$) with a random permutation $\boldsymbol{\Pi}_l$ of elements $\{(l-1)t + 1, \ldots, lt\}$.

It is quite obvious that the design matrix $\boldsymbol{X}$ constructed above is an n-run SLHD with t slices. Unlike the procedure introduced by Qian (2012) that generates the whole design column by column, we first construct the small design in each slice and then arrange them together. As a result, we will refer to this new method as *slice-wise construction method*. An example is provided below to illustrate the proposed procedure, where we generate three small LHDs $\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3$ and arrange their levels to form a 12-run SLHD:

Comparing this design with Figure 1(a), it can be seen that this SLHD is the same as the one generated by the SPMs in (2) using the procedure proposed by Qian (2012). In general, the statistical equivalency of the two methods in generating random SLHDs can be established by recognizing the following facts. For any SLHD generated by the proposed slice-wise method, we can construct p SPMs by assigning the jth column of the constructed design matrix in slice $\boldsymbol{X}_i$ to the ith column of the SPM $\boldsymbol{H}_j$. Then permuting elements in the jth column of the ith small LHD in our Step 1 is equivalent to permuting entries in the ith column of the jth SPM $\boldsymbol{H}_j$ in Qian (2012). In addition, permuting the t elements in each $\boldsymbol{\Pi}_i$ in our Step 2 is equivalent to permuting elements in each row of the (initial) SPMs in Qian (2012). Due to this equivalency, all the sampling properties of the SLHD discussed in Qian (2012) are also true for the SLHDs generated with the new method. Unlike the procedure introduced by Qian (2012), the proposed slice-wise construction has no constraint on the order of its two steps. After completing Step 2, we can still go back to Step 1 to modify the design in any slice and then use the same permutations $\boldsymbol{\Pi}_l$ ($l = 1, \ldots, m$) in Step 2 to generate new SLHDs in the neighborhood of the previous SLHD. This property is very important in applying the exchange algorithms for finding optimal SLHDs, which will be discussed in detail in Section 3.2. In addition, by envisioning the slice-wise construction method as a two-stage sampling process, we can further develop strategies to substantially improve the efficiency of generating space-filling SLHDs, and this topic will be exploited in Section 3.3.

Before concluding this section, we briefly point out two interesting variants of the proposed slice-wise construction method, which can produce some random SLHDs with improved properties. (1) As an improvement on random LHDs that only stratify univariate margins, Owen (1992) and Tang (1993) independently proposed to use *orthogonal arrays* (OAs; Hedayat, Sloane, and Stufken 1999; Wu and Hamada 2009) to generate OA-based LHDs, which can achieve uniformity in more than one dimensions. Because concatenating several OAs row by row together leads to a larger OA with similar strength, by using t OA-based LHDs (constructed as in Tang 1993) for the t slices in Step 1, the proposed slice-wise construction method can produce an *OA-based SLHD* in which the whole design as well as each of its slices can all achieve two- or higher-dimensional uniformities. (2) Another type of improvement over a random LHD is to construct the *orthogonal LHD* (Ye 1998) in which every pair of its columns is required to have zero correlation and/or the *second-order orthogonal LHD* in which the element-wise product of every two columns also need to be orthogonal to all the columns in the design (Sun, Liu, and Lin 2009). By choosing some second-order orthogonal LHDs for the t slices in Step 1 and transforming their levels wisely in Step 2, the proposed slice-wise construction method can also be adapted to produce the *second-order orthogonal SLHDs* in which the small LHD in each slice as well as the whole LHD are all second-order orthogonal. Designs generated by this approach can achieve more flexible run sizes and accommodate larger number of columns than the existing ones in the literature (Yang et al. 2013). Since the orthogonal SLHD may not necessarily be space-filling, we only give a brief description of their construction in the appendix. We next study how to improve the space-filling property of an SLHD, which is more important in designing computer experiments.

3. OPTIMAL SLHD

For any given m, t, p values, the slice-wise construction method proposed in Section 2 can generate $((m!)^t(t!)^m)^p$ possible SLHDs in total. Among them, we define the optimal SLHD as the one that optimizes a particular design optimality criterion. In this section, we first discuss how to develop performance measures for evaluating the goodness of an SLHD, and then in Section 3.2, we propose an algorithm that can generally be applied for finding the optimal SLHD with respect to any criteria. In Section 3.3, we further develop a very efficient algorithm for finding the optimal space-filling SLHDs.

3.1 Optimality Criteria for SLHD

For designing computer experiments, one of the most popularly used criteria is the *maximin distance criteria* (Johnson, Moore, and Ylvisaker 1990). By definition, this space-filling criterion tries to spread out the points in the design region in such a way that the minimum distance among the design points is maximized. Suppose we denote the distance between any two design points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \{\sum_{k=1}^{p} |x_{ik} - x_{jk}|^q\}^{1/q}$, in which $q = 1$ and $q = 2$ correspond to the rectangular and Euclidean distances, respectively. Because many different designs may have the same minimum interpoint distance, an extension of this criterion is to minimize the average reciprocal interpoint distance of the design $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ (Morris and Mitchell 1995; Jin, Chen, and Sudjianto 2005), given by

$$\phi_r(\boldsymbol{X}) = \left(\frac{2}{n(n-1)} \sum_{1 \le i < j \le n} \frac{1}{d(\boldsymbol{x}_i, \boldsymbol{x}_j)^r} \right)^{\frac{1}{r}}.$$

It can be seen that in a special case when $r \to \infty$, minimizing the above ϕ_r is equivalent to maximizing the minimum distance among the design points.

When extending the maximin distance criterion for evaluating the goodness of an SLHD $\boldsymbol{X} = (\boldsymbol{X}_1^T, \ldots, \boldsymbol{X}_t^T)^T$, we need to consider both the space-filling properties of the whole design as well as that of the small design in each slice. As a result, the *maximin distance SLHD* not only should minimize the $\phi_r(\boldsymbol{X})$ for all the design points, but it should also minimize the $\phi_r(\boldsymbol{X}_i)$ for each slice ($i = 1, \ldots, t$). To solve this multi-objective optimization problem, we propose a single objective function that takes a weighted average of all the above individual measures as follows:

$$\phi_{Mm}(\boldsymbol{X}) = \frac{1}{2} \left(\phi_r(\boldsymbol{X}) + \frac{1}{t} \sum_{i=1}^{t} \phi_r(\boldsymbol{X}_i) \right). \quad (3)$$

Here, more weights are given to $\phi_r(\boldsymbol{X})$ because the whole design matrix $\boldsymbol{X}$ contains t times as many points as each $\boldsymbol{X}_i$ does and thus its space-filling property is more important. Based on this combined measure, we can define the maximin distance SLHD as the one that minimizes the above $\phi_{Mm}(\boldsymbol{X})$. Similar structure of the objective function has also appeared in Jones and Nachtsheim (2011).

Other popular criteria for designing computer experiments can be extended for selecting the SLHD in a similar way. For example, we can define a *uniform SLHD* as the one that minimizes $\phi_{\text{unif}}(\boldsymbol{X}) = (\phi_{\text{CD}}(\boldsymbol{X}) + \sum_{i=1}^{t} \phi_{\text{CD}}(\boldsymbol{X}_i)/t)/2$, where ϕ_{CD} represents the centered L_2-discrepancy measure proposed by Hickernell (1998). In the next section, we develop a general algorithm to construct the optimal SLHD with respect to any criterion defined above.

3.2 A General Algorithm

In the literature, a version of the simulated annealing algorithm proposed by Morris and Mitchell (1995) has been widely used for constructing optimal LHDs. It uses an exchange algorithm to iteratively search in the design space in such a way that in each step two randomly chosen elements within a randomly selected column in the design matrix are interchanged. When it comes to the SLHDs, the construction process becomes more complex. In the slice-wise construction method proposed in the beginning of Section 2, we can see that the randomness in generating an SLHD comes from two sources: (i) the random permutation of elements in each column of the $\boldsymbol{X}_i$ ($i = 1, \ldots, t$) in Step 1; (ii) the random permutation of elements in each $\boldsymbol{\Pi}_l$ ($l = 1, \ldots, m$) in Step 2. In this section, we present a general algorithm to search for the optimal SLHDs, which uses an exchange algorithm to randomly perturb both of these two types of permutations. Suppose we want to minimize a prescribed performance measure $\phi(\boldsymbol{X})$ for the SLHD and let $p_0 = 0.5$. Our algorithm starts with a randomly chosen SLHD and then examines a sequence of new designs, each generated as a perturbation of the preceding one:

Step (i). Denote the current SLHD as $\boldsymbol{X}$. Draw a random variate $z \sim \text{Uniform}(0, 1)$.

Step (ii). If $z \le p_0$, randomly select a slice in $\boldsymbol{X}$, and interchange two randomly chosen elements within a randomly selected column in this slice. Denote the new SLHD as $\boldsymbol{X}_{\text{try}}$ and go to Step (iv).

Step (iii). If $z > p_0$, randomly choose a column in $\boldsymbol{X}$, and interchange two randomly chosen elements within a randomly selected $\boldsymbol{\Pi}_l$ $(l \in \{1, \dots, m\})$ for this column. Denote the resulting SLHD as $\boldsymbol{X}_{\text{try}}$ and go to Step (iv).

Step (iv). If $\phi(\boldsymbol{X}_{\text{try}}) < \phi(\boldsymbol{X})$, replace the current $\boldsymbol{X}$ with $\boldsymbol{X}_{\text{try}}$; otherwise, replace $\boldsymbol{X}$ with $\boldsymbol{X}_{\text{try}}$ with probability $\pi = \exp\{-[\phi(\boldsymbol{X}_{\text{try}}) - \phi(\boldsymbol{X})]/T\}$, where T is a preset parameter known as "temperature."

Step (v). Repeat Steps (i) to (iv) until some convergence requirements are met.

All the parameters in the above algorithm are set to be the same as those in a standard simulated annealing algorithm for which the convergence is already established (Lundy and Mees 1986). As a result, this proposed algorithm is guaranteed to ultimately converge to the global optimum. In the end, the SLHD with the smallest $\phi(\boldsymbol{X})$ value found by the algorithm is reported as our optimal design.

We note that since each iteration of the above algorithm only interchanges two elements within a column, its objective function $\phi(\boldsymbol{X})$ can usually be efficiently updated without recalculating all of its components at each step of the algorithm. Jin, Chen, and Sudjianto (2005) discussed the computational efficiency of evaluating several design optimality criteria. As an illustration, now we derive the updating formula of the most commonly used maximin distance measure $\phi_{Mm}(\boldsymbol{X})$ for our algorithm. As defined in (3), the value of $\phi_{Mm}(\boldsymbol{X})$ depends on the $n(n-1)/2$ interpoint distances $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ $(1 \le i < j \le n)$ among the n design points $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$. Whenever two elements x_{wh} and x_{vh} are interchanged within the hth column of the design matrix, only the interpoint distances that are related with the points $\boldsymbol{x}_w$ and $\boldsymbol{x}_v$ need to be updated:

$$d^*(\boldsymbol{x}_w, \boldsymbol{x}_k) = ([d(\boldsymbol{x}_w, \boldsymbol{x}_k)]^q + s(w, v, h, k))^{1/q}, \quad \text{for } 1 \le k \le n, k \ne w, v \tag{4}$$

$$d^*(\boldsymbol{x}_v, \boldsymbol{x}_k) = ([d(\boldsymbol{x}_v, \boldsymbol{x}_k)]^q - s(w, v, h, k))^{1/q}, \quad \text{for } 1 \le k \le n, k \ne w, v, \tag{5}$$

where $d^*(\cdot)$ represents the new distance and $d(\cdot)$ represents the distance from the previous step, $s(w, v, h, k) = |x_{vh} - x_{kh}|^q - |x_{wh} - x_{kh}|^q$ and all the other interpoint distances remain unchanged. Based on this observation, it can be shown that the new $\phi^*_{Mm}(\boldsymbol{X})$ can be efficiently calculated using the previous $\phi_r(\boldsymbol{X})$ and $\phi_r(\boldsymbol{X}_i)$ values as follows:

$$\phi^*_{Mm}(\boldsymbol{X}) = \frac{1}{2}\left(\phi^*_r(\boldsymbol{X}) + \frac{1}{t}\sum_{i=1}^{t} \phi^*_r(\boldsymbol{X}_i)\right), \tag{6}$$

in which

$$\begin{aligned}\phi^*_r(\boldsymbol{X}) = \Bigg(\frac{2}{n(n-1)}\Bigg[&\frac{n(n-1)}{2}[\phi_r(\boldsymbol{X})]^r \\ &+ \sum_{1\le k\le n, k\ne w,v} \left[d^*(\boldsymbol{x}_w, \boldsymbol{x}_k)^{-r} - d(\boldsymbol{x}_w, \boldsymbol{x}_k)^{-r}\right] \\ &+ \sum_{1\le k\le n, k\ne w,v} \left[d^*(\boldsymbol{x}_v, \boldsymbol{x}_k)^{-r} - d(\boldsymbol{x}_v, \boldsymbol{x}_k)^{-r}\right]\Bigg]\Bigg)^{1/r}, \end{aligned} \tag{7}$$

and (i) when $z \le p_1$, suppose the two points $\boldsymbol{x}_w$ and $\boldsymbol{x}_v$ belong to the cth slice $(c \in \{1, \dots, t\})$. Then we have: $w, v \in I_c = \{(c-1)m + 1, (c-1)m + 2, \dots, cm\}$, and

$$\phi^*_r(\boldsymbol{X}_i) = \begin{cases} \phi_r(\boldsymbol{X}_i), & \text{if } i \ne c, \\ \Big(\frac{2}{m(m-1)}\Big[\frac{m(m-1)}{2}[\phi_r(\boldsymbol{X}_i)]^r \\ \quad + \sum_{k\in I_c, k\ne w,v}\left[d^*(\boldsymbol{x}_w, \boldsymbol{x}_k)^{-r} - d(\boldsymbol{x}_w, \boldsymbol{x}_k)^{-r}\right] \\ \quad + \sum_{k\in I_c, k\ne w,v}\left[d^*(\boldsymbol{x}_v, \boldsymbol{x}_k)^{-r} - d(\boldsymbol{x}_v, \boldsymbol{x}_k)^{-r}\right]\Big]\Big)^{1/r}, & \text{if } i = c. \end{cases} \tag{8}$$

(ii) When $z > p_2$, suppose $\boldsymbol{x}_w$ belongs to slice c and $\boldsymbol{x}_v$ belongs to slice c' $(c, c' \in \{1, \dots, t\})$. Then we have $w \in I_c = \{(c-1)m + 1, (c-1)m + 2, \dots, cm\}$, $v \in I_{c'} = \{(c'-1)m + 1, (c'-1)m + 2, \dots, c'm\}$, and

$$\phi^*_r(\boldsymbol{X}_i) = \begin{cases} \phi_r(\boldsymbol{X}_i), & \text{if } i \ne c, c' \\ \Big(\frac{2}{m(m-1)}\Big[\frac{m(m-1)}{2}[\phi_r(\boldsymbol{X}_i)]^r \\ \quad + \sum_{k\in I_c, k\ne w}\left[d^*(\boldsymbol{x}_w, \boldsymbol{x}_k)^{-r} - d(\boldsymbol{x}_w, \boldsymbol{x}_k)^{-r}\right]\Big]\Big)^{1/r}, & \text{if } i = c, \\ \Big(\frac{2}{m(m-1)}\Big[\frac{m(m-1)}{2}[\phi_r(\boldsymbol{X}_i)]^r \\ \quad + \sum_{k\in I'_c, k\ne v}\left[d^*(\boldsymbol{x}_v, \boldsymbol{x}_k)^{-r} - d(\boldsymbol{x}_v, \boldsymbol{x}_k)^{-r}\right]\Big]\Big)^{1/r}, & \text{if } i = c'. \end{cases} \tag{9}$$

By using the above updating formulas, considerable computation can be saved in evaluating the objective functions in each iteration of our proposed algorithm.

3.3 Efficient Two-Stage Algorithm for Generating Space-Filling SLHDs

In the algorithm presented in Section 3.2, the performance measure $\phi(\boldsymbol{X})$ actually is very generally defined and may not necessarily be restricted to the space-filling measures. When n or p is large, however, the convergence may be slow due to the enormous number of candidate designs. In this section, we show that when our interest focuses on finding the optimal space-filling SLHDs, we can modify the algorithm to significantly reduce the required computation and improve its efficiency. The relevant design criteria include, for example, the maximin distance criterion $\phi_{Mm}(\boldsymbol{X})$ or the uniformity criterion $\phi_{\text{unif}}(\boldsymbol{X})$, which are most commonly used in designing computer experiments.

The key to improving the efficiency of our algorithm is to perceive the proposed construction method for SLHD as a two-stage sampling process. Particularly, the two steps in our slice-wise construction method (described in the beginning of Section 2) can be envisioned as follows: Step 1 partitions the p-dimensional input region into m^p cells using an $\underbrace{(m \times \cdots \times m)}_{p}$ *coarser* grid. To generate an m-run LHD, a set of m cells are sampled from the m^p population of cells in such a way that the projections of these m cells onto any axis fall into m equally spaced intervals. Such m cells are sampled repeatedly (with replacement) from the coarser grid for t times to form t independent m-run LHDs. In total, a sample of $n = mt$ cells are chosen in this step, and some of them may be the same. In Step 2, the input region is partitioned using a much *finer* $\underbrace{(n \times \cdots \times n)}_{p}$ grid, which further divides each large cell in Step 1 into t^p small subcells. One small subcell is sampled within each of the n sampled large cells from Step 1 in such a way that the projections of these n selected small subcells onto any axis fall into n equally spaced small intervals. Overall, these n selected subcells eliminate the one-dimensional projection redundancy and constitute a random SLHD. Figure 2 illustrates this two-stage sampling process using a coarser grid and a finer grid, respectively. It can be seen that since the sampling of subcells in Step 2 only takes place locally within the previously selected large cells, it has a much smaller impact on the overall space-filling property of the SLHD. In addition, to obtain n space-filling subcells in Step 2, its n mother cells from Step 1 must be well spread out. When $n < m^p$, this means that these n sampled large cells in Step 1 must be space-filling and different slices cannot sample the same cell from the coarser grid.

The above observations motivate the development of an efficient two-stage algorithm that optimizes the two steps of the slice-wise construction method sequentially (instead of simultaneously) for finding the space-filling SLHDs. In the Stage-I algorithm, we focus on how to construct the t small LHDs $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_t$ so that their combined design matrix $\boldsymbol{X} = \bigcup_{i=1}^{t} \boldsymbol{X}_i$ is optimal with respect to a chosen space-filling measure $\phi(\boldsymbol{X})$. After fixing these optimal small LHDs, we can run a Stage-II algorithm that only optimizes the permutations $\boldsymbol{\Pi}_l$ ($l = 1, \ldots, m$) for each of the p columns in $\boldsymbol{X}$. Through this way, the candidate SLHDs that we need to search have been greatly reduced from $((m!)^t(t!)^m)^p$ into a much smaller subset of only $(m!)^{tp} + (t!)^{mp}$ designs. This can lead to substantial savings in the computation.

A simple way to implement the above two-stage algorithm is to run the general algorithm presented in Section 3.2 twice: we first start with a randomly generated SLHD and run the algorithm with parameter p_0 fixed as 1; after obtaining the "optimized" SLHD, we use it as the starting design and run the algorithm again in which we fix the parameter p_0 as 0. It can easily be seen that the first stage of this procedure optimizes the small LHDs in the t slices and then in the second stage it only improves the $\boldsymbol{\Pi}_l$ permutations.

When $m^p > n$, the above Stage-I algorithm can be further sped up by cutting out a significant amount of undesirable designs from its candidate set. Note that the assumption $m^p > n$ is quite general since it can easily be met when the input dimension p is moderate or large. Because computer experiments are usually characterized by a large number of inputs, it is quite a standard for us to have $m^p \gg n$ in practice. As we have discussed earlier, since the sample of n cells chosen by the Stage-I algorithm needs to be space-filling, no cell from the $\underbrace{(m \times \cdots \times m)}_{p}$ grid can be sampled more than once for different slices. In other words, an efficient Stage-I algorithm should quickly eliminate all the undesirable designs that contain replicated rows in the combined design matrix $\boldsymbol{X} = \bigcup_{i=1}^{t} \boldsymbol{X}_i$. In light of this, the improved two-stage algorithm can be stated as follows:

Stage-I algorithm. The algorithm starts with an $(n \times p)$ design matrix, which is comprised of t randomly chosen $(m \times p)$ LHDs. In each step, suppose the current combined design matrix is $\boldsymbol{X} = \bigcup_{i=1}^{t} \boldsymbol{X}_i$, and the interpoint distances among its n design points are $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ $(1 \le i \ne j \le n)$. Denote $S(\boldsymbol{X}) = \sum_{1 \le i < j \le n} \mathbf{1}\{d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0\}$, where $\mathbf{1}\{A\} = 1$ if A is true and $\mathbf{1}\{A\} = 0$ otherwise. A positive $S(\boldsymbol{X})$ indicates that some rows in the current design matrix $\boldsymbol{X}$ are identical, and we refer to them as the *duplicated rows*. This algorithm proceeds as follows:

(i) If $S(\boldsymbol{X}) = 0$, directly go to Step (iv); otherwise, go to Step (ii) to reduce the number of duplicated rows in the combined design matrix $\boldsymbol{X}$.
(ii) Randomly pick a duplicated row from the current $\boldsymbol{X}$, and then randomly select another row within the same slice of this row. Interchange the two elements that belong to a randomly chosen column of these two rows. Denote the new combined design matrix as $\boldsymbol{X}_{\text{try}}$, and update the corresponding interpoint distances using formulas (4) and (5).

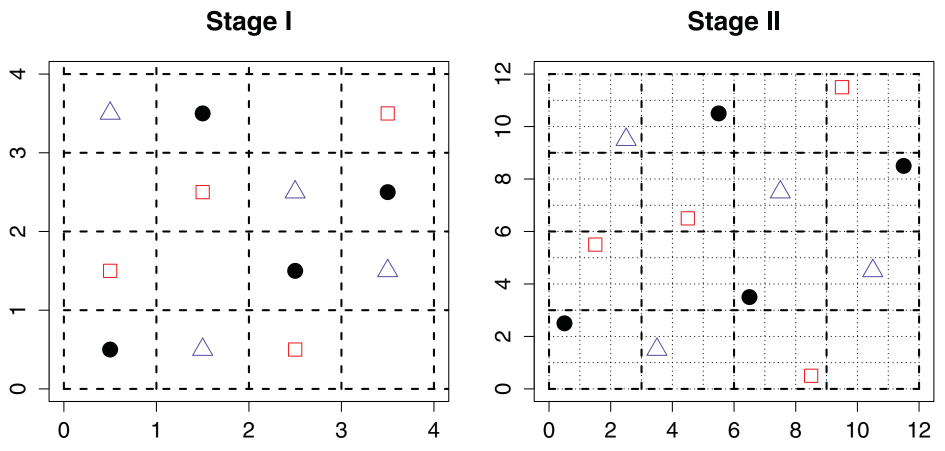

Figure 2. Illustration of the slice-wise construction method as a two-stage sampling process.

(iii) If $S(\boldsymbol{X}_{\text{try}}) < S(\boldsymbol{X})$, replace $\boldsymbol{X}$ with $\boldsymbol{X}_{\text{try}}$ and go to Step (i); otherwise, discard $\boldsymbol{X}_{\text{try}}$ and go back to Step (ii).

(iv) Randomly select a slice from the current $\boldsymbol{X}$, and interchange two randomly chosen elements within a randomly selected column in this slice. Denote the new combined design matrix as $\boldsymbol{X}_{\text{try}}$, and update the corresponding interpoint distances using formulas (4) and (5).

(v) If $S(\boldsymbol{X}_{\text{try}}) > 0$, discard $\boldsymbol{X}_{\text{try}}$ and go back to Step (iv); otherwise, go to Step (vi).

(vi) If $\phi(\boldsymbol{X}_{\text{try}}) < \phi(\boldsymbol{X})$, replace the current $\boldsymbol{X}$ with $\boldsymbol{X}_{\text{try}}$; otherwise, replace $\boldsymbol{X}$ with $\boldsymbol{X}_{\text{try}}$ with probability $\pi = \exp\{-[\phi(\boldsymbol{X}_{\text{try}}) - \phi(\boldsymbol{X})]/T\}$, where T is a preset parameter known as "temperature."

(vii) Repeat Steps (iv) to (vi) until some convergence requirements are met.

It can be seen that after quickly eliminating the initial row duplications in Steps (i) to (iii), the above algorithm never considers any candidate design that contains duplicated rows. Compared to running the general algorithm in Section 3.2 with p_0 set as 1, the above Stage-I algorithm cuts out all the candidate designs with positive $S(\boldsymbol{X})$ from the search space, and thus it can be expected to have a much faster convergence rate.

Stage-II algorithm. After obtaining the optimal combined design matrix $\boldsymbol{X}$ from the Stage I, we can use it (with some randomly chosen $\boldsymbol{\Pi}_l$'s) as the starting design and run the general algorithm in Section 3.2 with p_0 set as 0. This Stage-II algorithm optimizes the permutations $\boldsymbol{\Pi}_l$ $(l = 1, \ldots, m)$ for each column of the matrix $\boldsymbol{X}$, which arrange the t small space-filling LHDs found by the Stage-I algorithm into an optimal space-filling SLHD.

As we have discussed in the beginning of this section, since the Stage-I algorithm plays a much more important role in determining the space-filling property of the SLHD, more resources should be allocated for running the Stage-I algorithm if our computational budgets are tight. When p is very large and $m^p \gg n$, we may even completely focus on the Stage-I algorithm and skip the optimization in Stage II by directly using some randomly generated $\boldsymbol{\Pi}_l$'s. In such cases, because the n sampled cells from the Stage-I algorithm are very sparse (far away from each other) in the $(\underbrace{m \times \cdots \times m}_{p})$ grid, the Stage-II algorithm has a very minor effect in determining the space-filling property of the SLHD.

4. EXAMPLES

Example 1. A new bottle design was needed for a beauty product of The Procter & Gamble Company that is shipped under changing air pressure due to altitude changes and the solution is mixed by the consumer through shaking prior to use. The new bottle design was optimized through a computer simulation of both the air pressure changes and the solution mixing process with the goal to minimize bottle deformation under pressure while at the same time maximize mixing properties. There were five continuous variables and three two-level categorical variables in the computer experiment. The computer experiment along with the subsequent analysis allowed the project team to define a new bottle design that was strong enough to handle changes in air pressure seen through shipping, had very good mixing properties, and was aesthetically pleasing.

Based on the available resources a 256-run computer experiment was proposed. Since the three categorical factors have eight different level combinations, we constructed an SLHD containing eight slices, each of which is a 32-run LHD for the five continuous factors ($n = 256$, $m = 32$, $t = 8$, and $p = 5$). To demonstrate the advantage of using an optimal SLHD over a random SLHD, in this example we randomly generate SLHDs for 1000 times and draw the density plot of their minimum interpoint distances in Figure 3(a). Then a maximin distance SLHD of the same size is constructed using the algorithm in Section 3 and its corresponding minimum interpoint distance is depicted by the red arrow in Figure 3(a). It can be seen that even the best random SLHD out of the previous 1000 samples is substantially worse than the optimal SLHD. This clearly shows the necessity and benefits for optimizing the SLHD using the proposed algorithm. Since it is well known that the space-filling property of a design has a huge impact on the predictive accuracy of the surrogate model (Johnson, Moore, and Ylvisaker 1990), we can expect to obtain a much more accurate surrogate model when using the proposed optimal design instead of a random design.

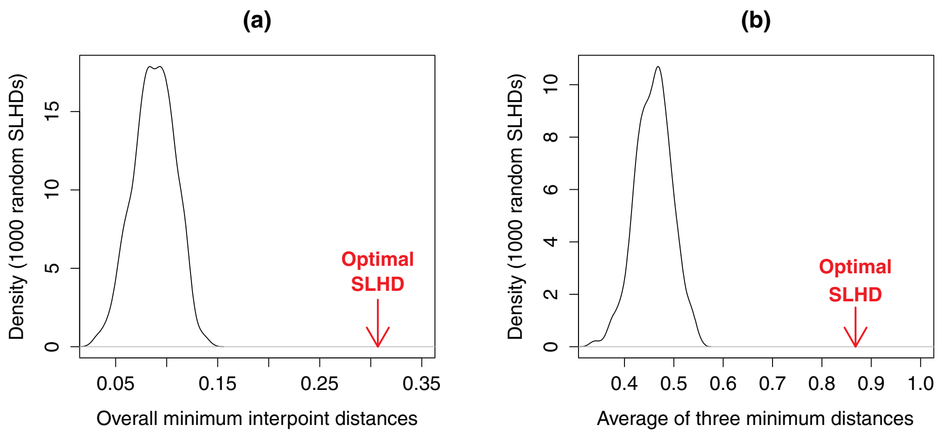

Figure 3. Density plots for 1000 random SLHDs: (a) Example 1: overall minimum interpoint distances. (b) Example 2: average of the minimum interpoint distances of the designs in three slices.

Example 2. Engineers at the Procter & Gamble employed computer experiments to model the packing line process. The following experiment involves modeling a particular part (or transformation) of an oral care packing line. The computer simulation consisted of nine continuous variables and one three-level categorical variable representing three types of a particular equipment part. A 132-run SLHD containing three slices, each of which is a 44-run LHD for the continuous factors was used ($n = 132, m = 44, t = 3$, and $p = 9$). Interestingly, after the data were collected, it was found that the correlations of responses across different categories were quite weak. Since the model cannot borrow too much information among the data from different categories, any new prediction needs to be made mainly based on the information from a single slice. This clearly demonstrates the importance of also having a good space-filling design for each slice. In Figure 3(b), we compare the average of the minimum interpoint distances of the designs in three slices for an optimal SLHD with that for 1000 random SLHDs of the same size. We can see that the optimal SLHD is also much superior in this aspect.

5. CONCLUSIONS

In this article, we developed a new approach for constructing the SLHDs in which we first generate the small LHD in each slice and then arrange them together. Unlike the method introduced by Qian (2012), the proposed slice-wise construction method can be easily adapted to produce SLHDs with many desirable properties. Not only can it construct the OA-based SLHDs, but it can also be extended to generate the second-order orthogonal SLHDs with more flexible run sizes and larger number of factors than those existing designs in the literature. More importantly, a general algorithm has been developed based on the proposed slice-wise construction method for finding the optimal SLHDs with respect to any criterion. When our interests focus on generating the space-filling SLHDs, a two-stage strategy has also been presented to substantially improve the efficiency of our algorithm. An R package "SLHD" for implementing the proposed algorithm can be downloaded from *http://cran.r-project.org/*

APPENDIX: CONSTRUCTION PROCEDURE FOR THE SECOND-ORDER ORTHOGONAL SLHD

For simplicity, we center the n equally spaced levels in LHD to have mean equal zero, so that the orthogonality of any two columns only requires their inner product to be zero. The proposed procedure is based on Sun, Liu, and Lin (2009), which can generate any second-order orthogonal LHD in 2^{c+1} or $2^{c+1}+1$ runs ($c \geq 1$) with 2^c columns as follows:

For $c = 1$, let

$$\boldsymbol{S}_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \ \boldsymbol{T}_1 = \begin{pmatrix} 1 & 2 \\ 2 & -1 \end{pmatrix}.$$

For $c \geq 2$, iteratively define $\boldsymbol{S}_c$ and $\boldsymbol{T}_c$ as

$$\boldsymbol{S}_c = \begin{pmatrix} \boldsymbol{S}_{c-1} & -\boldsymbol{S}^*_{c-1} \\ \boldsymbol{S}_{c-1} & \boldsymbol{S}^*_{c-1} \end{pmatrix},$$
$$\boldsymbol{T}_c = \begin{pmatrix} \boldsymbol{T}_{c-1} & -(\boldsymbol{T}^*_{c-1} + 2^{c-1}\boldsymbol{S}^*_{c-1}) \\ \boldsymbol{T}_{c-1} + 2^{c-1}\boldsymbol{S}_{c-1} & \boldsymbol{T}^*_{c-1} \end{pmatrix},$$

where the * operator works on any matrix with an even number of rows by multiplying the entries in the top half of the matrix by -1 and leaving those in the bottom half unchanged. Sun, Liu, and Lin (2009) showed that $\boldsymbol{L}_c = (\boldsymbol{T}^T_c, \boldsymbol{0}_{2^c}, -\boldsymbol{T}^T_c)^T$ is a second-order orthogonal LHD in $2^{c+1}+1$ runs with 2^c columns, where $\boldsymbol{0}_{2^c}$ denotes a $(2^c \times 1)$ column vector with all elements equal zero, and $\boldsymbol{L}_c = (\boldsymbol{H}^T_c, -\boldsymbol{H}^T_c)^T$ is a second-order orthogonal LHD in 2^{c+1} runs with 2^c columns, where $\boldsymbol{H}_c = \boldsymbol{T}_c - \boldsymbol{S}_c/2$.

Based on Sun, Liu, and Lin's (2009) method, the slice-wise construction procedure can be modified as follows to generate the n-run second-order orthogonal SLHD consisting of t slices, in which each slice is a small second-order orthogonal LHD in $m = 2^{c+1}$ or $2^{c+1}+1$ runs ($c \geq 1$ and $n = mt$):

(i) For $m = 2^{c+1}$, any $t \geq 1$ and $p \leq 2^c$ ($\forall c \geq 1$):

Step 1. Use Sun, Liu, and Lin's (2009) method to independently generate t small $(m \times p)$ orthogonal LHDs $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_t$ as the design matrices for the t slices. Code their factor levels as $\{-2^c + 0.5, \ldots, -1.5, -0.5, 0.5, 1.5, \ldots, 2^c - 0.5\}$, and combine them row by row to form an $(n \times p)$ matrix $\boldsymbol{X} = \bigcup_{i=1}^t \boldsymbol{X}_i$.

Step 2. Within each $\boldsymbol{X}_i$ (for $i = 1, \ldots, t$), transform every positive element l to $(l - 0.5)t + (i - 0.5)$ and transform every negative element l to $(l + 0.5)t - (i - 0.5)$.

(ii) For $m = 2^{c+1}+1$, $t = 2^{c'+1}$ or $2^{c'+1}+1$ and $p \leq 2^{\min(c,c')}$ ($\forall c, c' \geq 1$):

Step 1. Use Sun, Liu, and Lin's (2009) method to independently generate t small $(m \times p)$ orthogonal LHDs $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_t$ for the t slices. Code their factor levels as $\{-2^c, \ldots, -1, 0, 1, \ldots, 2^c\}$, and combine them row by row to form an $(n \times p)$ matrix $\boldsymbol{X} = \bigcup_{i=1}^t \boldsymbol{X}_i$.

Step 2. Within each $\boldsymbol{X}_i$ (for $i = 1, \ldots, t$), transform every positive element l to $(l - 0.5)t + (i - 0.5)$ and transform every negative element l to $(l + 0.5)t - (i - 0.5)$.

Step 3. Use Sun, Liu, and Lin's (2009) method to construct a $(t \times p)$ orthogonal LHD whose levels are taken as $\{1 - (1+t)/2, 2 - (1+t)/2, \ldots, t - (1+t)/2\}$. For $j = 1, \ldots, p$, replace the t entries of level 0 in the jth column of the matrix $\boldsymbol{X}$ with the sequence in the jth column of this $(t \times p)$ orthogonal LHD.

Lemma A.1. For any constant a and b, define $\boldsymbol{W}_c = a\boldsymbol{T}_c - b\boldsymbol{S}_c$. We have $\boldsymbol{W}^T_c \boldsymbol{W}_c = w_c \boldsymbol{I}_{2^c}$, where $w_c = (a^2 2^c(2^c+1)(2^{c+1}+1)/6 - ab(2^{2c} + 2^c) + b^2 2^c)$ and $\boldsymbol{I}_{2^c}$ is a $(2^c \times 2^c)$ identity matrix.

Proof. $\boldsymbol{W}^T_c \boldsymbol{W}_c = (a\boldsymbol{T}^T_c - b\boldsymbol{S}^T_c)(a\boldsymbol{T}_c - b\boldsymbol{S}_c) = a^2 \boldsymbol{T}^T_c \boldsymbol{T}_c - ab(\boldsymbol{S}^T_c \boldsymbol{T}_c + \boldsymbol{T}^T_c \boldsymbol{S}_c) + b^2 \boldsymbol{S}^T_c \boldsymbol{S}_c$. According to the Lemma 1 and the proof of Theorem 1 in Sun, Liu, and Lin (2009), we know that $\boldsymbol{S}^T_c \boldsymbol{S}_c = 2^c \boldsymbol{I}_{2^c}$, $\boldsymbol{S}^T_c \boldsymbol{T}_c + \boldsymbol{T}^T_c \boldsymbol{S}_c = (2^{2c} + 2^c)\boldsymbol{I}_{2^c}$ and $\boldsymbol{T}^T_c \boldsymbol{T}_c = 2^c(2^c+1)(2^{c+1}+1)/6 \boldsymbol{I}_{2^c}$. Therefore, we have $\boldsymbol{W}^T_c \boldsymbol{W}_c = (a^2 2^c(2^c+1)(2^{c+1}+1)/6 - ab(2^{2c}+2^c) + b^2 2^c)\boldsymbol{I}_{2^c}$. □

Theorem A.1. The design matrix $\boldsymbol{X}$ generated from the above procedure is an SLHD, in which the whole design as well as each of its slices are all second-order orthogonal LHDs.

Proof. Since the design matrix in each slice $\boldsymbol{X}_i$ ($i = 1, \ldots, t$) is constructed according to Sun, Liu, and Lin (2009), each of them is guaranteed to be a second-order orthogonal LHD. From the above construction procedure, it is also obvious that each column of the matrix $\boldsymbol{X}$ forms a permutation of $\{1 - (1+n)/2, 2 - (1+n)/2, \ldots, n - (1+n)/2\}$, and thus $\boldsymbol{X}$ is an n-run SLHD.

Now we show that the columns in $\boldsymbol{X}$ are orthogonal to each other. First, it is easy to see that $\boldsymbol{S}_c$ is the matrix whose entries are taken as 1 or -1 according to the signs of the elements in matrix $\boldsymbol{T}_c$ (or equivalently in matrix $\boldsymbol{H}_c$). When $m = 2^{c+1}$, the proposed procedure constructs each $\boldsymbol{X}_i$ ($i = 1, \ldots, t$) as a set of p columns from

$\tilde{\boldsymbol{L}}_c^{(i)} = (\boldsymbol{M}_c^{(i)T}, -\boldsymbol{M}_c^{(i)T})^T$, where $\boldsymbol{M}_c^{(i)} = t\boldsymbol{H}_c + (i - 0.5 - 0.5t)\boldsymbol{S}_c = t\boldsymbol{T}_c + (i - 0.5 - t)\boldsymbol{S}_c$. According to Lemma 1, $\boldsymbol{M}_c^{(i)T}\boldsymbol{M}_c^{(i)}$ is a diagonal matrix. Therefore, all the columns in each $\tilde{\boldsymbol{L}}_c^{(i)}$ are orthogonal to each other ($i = 1, \ldots, t$), and thus $\boldsymbol{X} = (\boldsymbol{X}_1^T, \ldots, \boldsymbol{X}_t^T)^T$ is an orthogonal LHD. When $m = 2^{c+1} + 1$, the proposed procedure constructs each $\boldsymbol{X}_i$ $(i = 1, \ldots, t)$ as a set of p columns from $\tilde{\boldsymbol{L}}_c^{(i)} = (\boldsymbol{M}_c^{(i)T}, \tilde{\boldsymbol{0}}_{2^c}^{(i)}, -\boldsymbol{M}_c^{(i)T})^T$, where $\boldsymbol{M}_c^{(i)} = t\boldsymbol{T}_c + (i - 0.5 - 0.5t)\boldsymbol{S}_c$ and $\tilde{\boldsymbol{0}}_{2^c}^{(i)}$ corresponds to the original $2^c \times 1$ zero vector but its elements are relabeled according to Step 3. Using the result in Lemma 1, we know that $\boldsymbol{M}_c^{(i)T}\boldsymbol{M}_c^{(i)}$ is a diagonal matrix. Since Step 3 has relabeled the t original zero vectors from $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_t$ in such a way that they form a $(t \times p)$ second-order orthogonal LHD, it is obvious that $\boldsymbol{X} = (\boldsymbol{X}_1^T, \ldots, \boldsymbol{X}_t^T)^T$ is an orthogonal LHD.

Now we only need to prove that the columns in $\boldsymbol{X}$ also satisfies the second-order orthogonality property. Observe that the matrix $\boldsymbol{X}$ can always be arranged into blocks as $\boldsymbol{X} = (\boldsymbol{K}^T, -\boldsymbol{K}^T)^T$ when n is even or $\boldsymbol{X} = (\boldsymbol{K}^T, \boldsymbol{0}_p, -\boldsymbol{K}^T)^T$ when n is odd. Thus, for any three columns in $\boldsymbol{X}$, no matter whether they are distinct or not, they can be denoted as $\boldsymbol{c}_j = (\boldsymbol{k}_j^T, -\boldsymbol{k}_j^T)^T$ when n is even or $\boldsymbol{c}_j = (\boldsymbol{k}_j^T, 0, -\boldsymbol{k}_j^T)^T$ when n is odd, for $j = 1, 2, 3$. Then, we have $(\boldsymbol{c}_1 \odot \boldsymbol{c}_2 \odot \boldsymbol{c}_3)^T \boldsymbol{1}_n = (\boldsymbol{k}_1 \odot \boldsymbol{k}_2 \odot \boldsymbol{k}_3)^T \boldsymbol{1}_m - (\boldsymbol{k}_1 \odot \boldsymbol{k}_2 \odot \boldsymbol{k}_3)^T \boldsymbol{1}_m = 0$, where $\boldsymbol{u} \odot \boldsymbol{v}$ represents the element-wise product of $\boldsymbol{u}$ and $\boldsymbol{v}$, $\boldsymbol{1}_m$ represents the $m \times 1$ vector with all elements equal one, $m = n/2$ when n is even and $m = (n - 1)/2$ when n is odd. Now, it follows immediately that the element-wise square of each column and the element-wise product of every two columns in $\boldsymbol{X}$ are orthogonal to all the columns in the design. □

Since any 2^{c+1}-run second-order orthogonal LHD can at most contain 2^c columns (Sun, Liu, and Lin 2009), it is easy to see that for $m = 2^{c+1}$ $(\forall c \geq 1)$, the second-order orthogonal SLHDs constructed by the proposed procedure have achieved the *maximum* 2^c number of factors (for any number of slices $t \geq 1$). This is a substantial improvement compared to the existing method in Yang et al. (2013), whose second-order orthogonal SLHDs must satisfy the following two restrictions: (i) the number of slices is a power of two: $t = 2^r, r = 1, \ldots, c$; and (ii) each slice contains 2^{2c-r+1} rows but only 2^c columns. Obviously, the slice number t in Yang et al. (2013) is much more restrictive, and unless the design contains exactly 2^c slices $(r = c)$, the number of columns these designs can accommodate are much smaller.

ACKNOWLEDGMENTS

The authors thank the editor, an associate editor, and two referees for their valuable comments and suggestions. This work was partially completed when the first author was an assistant professor at the Fariborz Maseeh Department of Mathematics and Statistics in the Portland State University.

[Received December 2012. Revised July 2014.]

REFERENCES

Fang, K. T., Li, R., and Sudjianto, A. (2005), *Design and Modeling for Computer Experiments*, London: Chapman and Hall. [479]

Hedayat, A., Sloane, N., and Stufken, J. (1999), *Orthogonal Arrays*, New York: Springer Verlag. [482]

Hickernell, F. J. (1998), "A Generalized Discrepancy and Quadrature Error Bound," *Mathematics of Computation*, 67, 299–322. [482]

Jin, R., Chen, W., and Sudjianto, A. (2005), "An Efficient Algorithm for Constructing Optimal Design of Computer Experiments," *Journal of Statistical Planning and Inference*, 134, 268–287. [481,482,483]

Johnson, M. E., Moore, L. M., and Ylvisaker, D. (1990), "Minimax and Maximin Distance Designs," *Journal of Statistical Planning and Inference*, 26, 131–148. [482,485]

Jones, B., and Nachtsheim, C. J. (2011), "Efficient Designs with Minimal Aliasing," *Technometrics*, 53, 62–71. [482]

Joseph, V. R., and Hung, Y. (2008), "Orthogonal-Maximin Latin Hypercube Designs," *Statistica Sinica*, 18, 171–186. [481]

Lundy, M., and Mees, A. (1986), "Convergence of an Annealing Algorithm," *Mathematical Programming*, 34, 111–124. [483]

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979), "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code," *Technometrics*, 21, 239–245. [479]

Morris, M. D., and Mitchell, T. J. (1995), "Exploratory Designs for Computational Experiments," *Journal of Statistical Planning and Inference*, 43, 381–402. [480,482]

Owen, A. B. (1992), "Orthogonal Arrays for Computer Experiments, Integration and Visualization," *Statistica Sinica*, 2, 439–452. [482]

Qian, P. Z. G. (2012), "Sliced Latin Hypercube Designs," *Journal of the American Statistical Association*, 107, 393–399. [479,481,486]

Santner, T. J., Williams, B. J., and Notz, W. (2003), *The Design and Analysis of Computer Experiments*, New York: Springer. [479]

Sun, F. S., Liu, M. Q., and Lin, D. K. J. (2009), "Construction of Orthogonal Latin Hypercube Designs," *Biometrika*, 96, 971–974. [482,486,487]

Tang, B. (1993), "Orthogonal Array-Based Latin Hypercubes," *Journal of the American Statistical Association*, 88, 1392–1397. [482]

Wu, C. F. J., and Hamada, M. S. (2009), *Experiments: Planning, Analysis, and Optimization* (2nd ed.), New York: Wiley. [482]

Yang, J., Lin, C. D., Qian, P. Z. G., and Lin, D. K. J. (2013), "Construction of Sliced Orthogonal Latin Hypercube Designs," *Statistica Sinica*, 23, 1117–1130. [482,487]

Ye, K. Q. (1998), "Orthogonal Column Latin Hypercubes and Their Application in Computer Experiments," *Journal of the American Statistical Association*, 93, 1430–1439. [482]