Statistica Sinica Preprint No: SS-2022-0263

Title	A Construction Method for Maximin L1-Distance Latin Hypercube Designs
Manuscript ID	SS-2022-0263
URL	http://www.stat.sinica.edu.tw/statistica/
DOI	10.5705/ss.202022.0263
Complete List of Authors	Ru Yuan, Yuhao Yin, Hongquan Xu and Min-Qian Liu
Corresponding Authors	Min-Qian Liu
E-mails	mqliu@nankai.edu.cn

A CONSTRUCTION METHOD FOR MAXIMIN L_1-DISTANCE LATIN HYPERCUBE DESIGNS

Ru Yuan[1], Yuhao Yin[2], Hongquan Xu[2], Min-Qian Liu[3]

[1] *Zhongnan University of Economics and Law,*

[2] *University of California, Los Angeles, and* [3] *Nankai University*

Abstract: Maximin distance designs are a kind of space-filling design, and are widely used in computer experiments. However, although much work has been done on constructing such designs, doing so for a large number of rows and columns remains challenging. In this paper, we propose a theoretical construction method that generates a maximin L_1-distance Latin hypercube design with a run size that is close to the number of columns, or half the number of columns. Our theoretical results show that some of the constructed designs are both maximin L_1-distance and equidistant designs, which means that their pairwise L_1-distances are all equal, and that they are uniform projection designs. Other designs are asymptotically optimal under the maximin L_1-distance criterion. Moreover, the proposed method is efficient for constructing high-dimensional Latin hypercube designs that perform well under the maximin L_1-distance criterion.

Key words and phrases: Computer experiment, Latin square, maximin distance design, space-filling design.

1. Introduction

Computer experiments are increasingly being used to investigate complex systems (Fang, Li, and Sudjianto, 2006). In doing so, it is crucial to use a good space-filling de-

Statistica Sinica: Preprint
doi:10.5705/ss.202022.0263

sign in order to explore the design space effectively and build a high-quality metamodel. Generating a space-filling design involves seeking design points that fill a bounded design region as uniformly as possible. Much work has been done on constructing such designs, including Latin hypercube designs (LHDs; McKay, Beckman, and Conover, 1979) and their extensions (Lin, Mukerjee, and Tang, 2009), maximin distance designs (Johnson, Moore, and Ylvisaker, 1990), and uniform designs (Fang et al., 2018).

One fruitful approach to constructing space-filling designs is to use orthogonal arrays (Hedayat, Sloane, and Stufken, 1999). Owen (1992) and Tang (1993) consider randomized orthogonal arrays and orthogonal array-based LHDs, respectively, representing an important development in this area. Orthogonal arrays have also been used to construct orthogonal LHDs; see Steinberg and Lin (2006), Pang, Liu, and Lin (2009), Sun, Liu, and Lin (2009, 2010), Sun, Pang, and Liu (2011), and Wang et al. (2018). Another approach is to find optimal designs based on other criteria, such as uniformity measures (Fang, Li, and Sudjianto, 2006; Fang et al., 2018), the maximin and minimax distances (Johnson, Moore, and Ylvisaker, 1990), and the integrated mean squared error (Montgomery, 2008). Santner, Williams, and Notz (2018) comprehensively examine various space-filling measures, finding that the maximin distance criterion, which maximizes the minimal distance between all pairs of points, is preferable to the other criteria.

Morris and Mitchell (1995) use a simulated annealing algorithm to search for maximin LHDs. Joseph and Hung (2008) propose an algorithm that generates an orthogonal maximin LHD by combining correlation and distance performance measures. Ba, Myers, and Brenneman (2015) propose an efficient algorithm that searches for maximin distance sliced LHDs, available as an R package called "SLHD." Numerous other algorithms have also been proposed for constructing maximin LHDs; see Lin and Tang (2015) for a review. Such algorithmic methods are useful for generating flexible LHDs, but are not efficient for constructing large designs, owing to their computational complexity. However, large designs are needed for computer experiments; for example, Morris (1991) and Kleijnen (1997) provide many computer models that involve several hundred factors. Xiao and Xu (2017) note that in such cases, it is not unreasonable to assume effect sparsity. Thus, saturated or even supersaturated LHDs are useful for identifying a few active factors using limited runs.

Zhou and Xu (2015) propose constructing maximin LHDs by using a linear-level permutation based on good lattice point sets. Xiao and Xu (2017) propose methods for constructing LHDs with large L_1-distances that use Costas arrays. Wang, Xiao, and Xu (2018) use Williams transformations of good lattice point designs to construct a series of maximin LHDs, some of which are optimal under the maximin L_1-distance criterion and have small pairwise correlations between columns. He (2019) proposes a method for constructing maximin distance designs from interleaved lattices. Zhou, Yang, and Liu (2020) use the rotation method to construct maximin L_2-distance LHDs based on a 2^2 full factorial design and a series of saturated two-level regular designs. Li, Liu, and Tang (2021) propose an easy-to-use method for constructing maximin distance designs based on some carefully selected small designs.

Focusing on two-dimensional projection uniformity, Sun, Wang, and Xu (2019) propose a design criterion called the uniform projection criterion. Uniform projection designs generated under this criterion scatter points uniformly in all dimensions, and have good space-filling properties in terms of distance, uniformity, and orthogonality. Moreover, the authors show that maximin L_1-equidistant designs are uniform projection designs, and provide a method for constructing uniform projection designs based on good lattice point sets when the number of rows is an odd prime.

Lin and Kang (2016) propose a general method for constructing Latin hypercubes with flexible run sizes for computer experiments. The method uses arrays with a special structure and LHDs. They show that their method can be used to generate maximin LHDs with flexible run sizes under the ϕ_r criterion. However, constructing maximin distance LHDs with many rows and columns remains challenging. Here, we propose a method for generating maximin L_1-distance LHDs with run sizes that are close to the number of columns, or half the number of columns. Some of the resulting designs are also Latin squares, which are widely used in designs of experiments and in other fields, see, for example, Hedayat, Sloane, and Stufken (1999) and Keedwell and Dénes (2015). Our theoretical results show that some of the constructed designs are both maximin L_1-distance and equidistant designs, which means their pairwise L_1-distances are all equal, as well as being uniform projection designs. Furthermore, others are asymptotically optimal under the maximin L_1-distance criterion.

The rest of this paper is organized as follows. Section 2 provides preliminaries needed for the development in the subsequent sections. The proposed construction method is presented in Section 3. Theoretical results and comparisons are provided in

Section 4. Section 5 concludes the paper. All proofs are deferred to the Appendix.

2. Preliminaries

For a positive integer b, let $\mathbb{Z}_b$ denote the set $\{1, \ldots, b\}$. Given any two integers a and b, $\gcd(a, b)$ denotes the greatest common divisor of a and b. If $\gcd(a, b) = 1$, then a is coprime to b. For any real number r, $\lfloor r \rfloor$ is the integer part of r.

A Latin square of order n is an $n \times n$ square matrix with n^2 entries of n different elements, none of them occurring twice within any row or column of the matrix. An isotopism of a Latin square L permutes the rows, columns, and elements of L, resulting in another Latin square, which is said to be isotopic to L. These two Latin squares belong to the same isotopy class (an isotopy class of Latin squares is an equivalence class for the isotopy relation). An LHD, denoted by LHD(n, s), is an $n \times s$ matrix in which each column is a permutation of the n different elements from $\mathbb{Z}_n$. A Latin square of order n is a special LHD(n, n) if the n different elements are taken from $\mathbb{Z}_n$.

For an integer $q \geq 1$, define $d_q(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{s} |x_i - y_i|^q \right)^{1/q}$ as the L_q-distance of any two row vectors, $\boldsymbol{x} = (x_1, \ldots, x_s)$ and $\boldsymbol{y} = (y_1, \ldots, y_s)$. In this paper, we take $q = 1$. Define the L_1-distance of design D as

$$d_1(D) = \min\{d_1(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x} \neq \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{y} \in D\}.$$

A maximin L_1-distance design D^* is defined as a design that satisfies

$$d_1(D^*) = \max_{D} d_1(D),$$

from among all possible candidate designs.

3. Construction Method

For a positive integer N, the number of positive integers that are less than and coprime to N is $\phi(N)$, where $\phi(\cdot)$ is the Euler function. It is easy to see that $\phi(N)$ is even for any integer $N > 2$. Define a generator vector $\boldsymbol{h}$ as

$$\boldsymbol{h} = (h_1, \ldots, h_n), \tag{3.1}$$

where $1 = h_1 < \cdots < h_n \leq \lfloor N/2 \rfloor$, and $\gcd(h_i, N) = 1$, for $i = 1, \ldots, n$, and $n = \phi(N)/2$. It is easy to verify that $\boldsymbol{h}$ consists of the first $\phi(N)/2$ elements of the generator vector for the $N \times \phi(N)$ good lattice point sets. Taking $\boldsymbol{h}$ given in (3.1) as the generator vector, we obtain an $n \times n$ square matrix $L = (r_{ij})$, with its (i, j)th element r_{ij} defined by

$$r_{ij} = \min\{h_i * h_j \,(\text{mod } N),\ N - h_i * h_j \,(\text{mod } N)\},\ i, j = 1, \ldots, n. \tag{3.2}$$

Lemma 1. *The $n \times n$ matrix L constructed in* (3.2) *is a Latin square of order n with n different elements $\{h_1, \ldots, h_n\}$.*

Table 1: Latin squares constructed using (3.2) for $N = 11$ and 22.

$N = 11$					$N = 22$				
1	2	3	4	5	1	3	5	7	9
2	4	5	3	1	3	9	7	1	5
3	5	2	1	4	5	7	3	9	1
4	3	1	5	2	7	1	9	5	3
5	1	4	2	3	9	5	1	3	7

Example 1. Let $N = 11$ and 22. Then, $n = \phi(N)/2 = 5$, $\boldsymbol{h} = (1, 2, 3, 4, 5)$ for $N = 11$, and $\boldsymbol{h} = (1, 3, 5, 7, 9)$ for $N = 22$. The Latin squares constructed using (3.2) are listed in Table 1.

For the Latin square L constructed using (3.2), replace each element h_i with i, for $i = 1, \ldots, n$, and denote the obtained matrix as D. Then, D is both an LHD(n, n) and a Latin square of order n with n different elements in $\mathbb{Z}_n$. The following example shows that design D performs well under the maximin L_1-distance criterion.

Example 2. Take the Latin square for $N = 22$ in Table 1 as an example. Replace each element h_i with i, for $i = 1, \ldots, n$, that is, $1 \to 1$, $3 \to 2$, $5 \to 3$, $7 \to 4$, and $9 \to 5$. Then, we have

$$\begin{pmatrix} 1 & 3 & 5 & 7 & 9 \\ 3 & 9 & 7 & 1 & 5 \\ 5 & 7 & 3 & 9 & 1 \\ 7 & 1 & 9 & 5 & 3 \\ 9 & 5 & 1 & 3 & 7 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 4 & 1 & 3 \\ 3 & 4 & 2 & 5 & 1 \\ 4 & 1 & 5 & 3 & 2 \\ 5 & 3 & 1 & 2 & 4 \end{pmatrix}.$$

It is easy to see that the generated matrix is both an LHD$(5, 5)$ and a Latin square of order 5. Furthermore, the L_1-distances of the two LHD$(5, 5)$'s obtained when $N = 11$ and 22 are both equal to $10 = (5 + 1)5/3$.

For an LHD(n, s), the average pairwise L_1-distance is $(n + 1)s/3$ (Zhou and Xu, 2015). In addition, the minimum pairwise L_1-distance cannot exceed the integer part of the average. Hence, the upper bound of the L_1-distance of any LHD(n, s) is $d_{upper} = \lfloor (n + 1)s/3 \rfloor$. It can be verified that the LHDs obtained in Example 2 are maximin L_1-distance designs. Inspired by this, we propose the following method for constructing maximin distance LHDs.

Algorithm 1 (Construction of maximin L_1-distance LHD(n, n))**.**

Step 1. For a given integer N, obtain the generator vector $\boldsymbol{h} = (h_1, \ldots, h_n)$ from (3.1), where $n = \phi(N)/2$.

Statistica Sinica: Preprint
doi:10.5705/ss.202022.0263

Step 2. Generate the $n \times n$ Latin square L using (3.2), where each row and column is a permutation of $\{h_1, \ldots, h_n\}$.

Step 3. Replace each element h_i in L with i, for $i = 1, \ldots, n$, and denote the resulting LHD(n, n) as D.

Note that we can also use $N - \boldsymbol{h}$ as the generator vector in Algorithm 1. In this case, the obtained design is the same as that constructed using the generator vector $\boldsymbol{h}$.

For the LHD(n, n) D constructed by Algorithm 1, let $\boldsymbol{l}_1, \ldots, \boldsymbol{l}_n$ be its 1st to nth rows, and α_i be the bijection from $\boldsymbol{l}_1 = (1, \ldots, n)$ to $\boldsymbol{l}_i = (l_{i1}, \ldots, l_{in})$ with $\alpha_i(k) = l_{ik}$, for $k = 1, \ldots, n$, $i = 1, \ldots, n$. Here, α_1 is obviously an identity mapping, and we have the following result.

Lemma 2. *The transformation set $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is a commutative group.*

Remark 1. For any two distinct rows $\boldsymbol{l}_i$ and $\boldsymbol{l}_j (i < j)$ from D, reorder the elements of $\boldsymbol{l}_i$ such that its elements are in increasing order, that is, $\boldsymbol{l}_i$ is transformed to $\boldsymbol{l}_1$. Apply the same permutation on the elements of row $\boldsymbol{l}_j$, and denote the newly obtained row by $\boldsymbol{l}'_j$. From Lemma 2 and the definition of the L_1-distance criterion, it is easy to verify that $\boldsymbol{l}'_j$ is still a row of D, and $d_1(\boldsymbol{l}_i, \boldsymbol{l}_j) = d_1(\boldsymbol{l}_1, \boldsymbol{l}'_j)$. Hence, the pairwise L_1-distances between rows in D take at most $n - 1$ different values.

Example 3. To illustrate Remark 1, take the LHD$(5, 5)$ D (listed in Table 1) constructed by Algorithm 1 for $N = 11$ as an example. Let $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ be the five bijections corresponding to its rows $\boldsymbol{l}_1, \boldsymbol{l}_2, \boldsymbol{l}_3, \boldsymbol{l}_4, \boldsymbol{l}_5$, respectively. It can be verified that

$\alpha_i^{-1}(\boldsymbol{l}_i) = \boldsymbol{l}_1$, for $i = 1, \ldots, 5$, and the following equalities hold:

$$d_1(\boldsymbol{l}_2, \boldsymbol{l}_3) = d_1(\alpha_2^{-1}(\boldsymbol{l}_2), \alpha_2^{-1}(\boldsymbol{l}_3)) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_4),$$

$$d_1(\boldsymbol{l}_2, \boldsymbol{l}_4) = d_1(\alpha_2^{-1}(\boldsymbol{l}_2), \alpha_2^{-1}(\boldsymbol{l}_4)) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_2),$$

$$d_1(\boldsymbol{l}_2, \boldsymbol{l}_5) = d_1(\alpha_2^{-1}(\boldsymbol{l}_2), \alpha_2^{-1}(\boldsymbol{l}_5)) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_3),$$

$$d_1(\boldsymbol{l}_3, \boldsymbol{l}_4) = d_1(\alpha_3^{-1}(\boldsymbol{l}_3), \alpha_3^{-1}(\boldsymbol{l}_4)) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_5),$$

$$d_1(\boldsymbol{l}_3, \boldsymbol{l}_5) = d_1(\alpha_3^{-1}(\boldsymbol{l}_3), \alpha_3^{-1}(\boldsymbol{l}_5)) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_2),$$

$$d_1(\boldsymbol{l}_4, \boldsymbol{l}_5) = d_1(\alpha_4^{-1}(\boldsymbol{l}_4), \alpha_4^{-1}(\boldsymbol{l}_5)) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_4).$$

This means that the L_1-distance of any two different rows in D is equal to one of the L_1-distances between its 1st row $\boldsymbol{l}_1$ and other rows $\boldsymbol{l}_{j'}$, for $j' = 2, 3, 4, 5$. Hence, the pairwise L_1-distances between rows in D take at most four different values.

Lemma 2 also implies that each transformation in the set $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ has an inverse mapping. Then, the design D generated by Algorithm 1 has the following property.

Corollary 1. *For row $\boldsymbol{l}_1$ and any two other rows $\boldsymbol{l}_i$ and $\boldsymbol{l}_j$ $(2 \le i, j \le n)$ of design D generated by Algorithm 1, with corresponding transformations α_1, α_i, and α_j, respectively, if α_j is the inverse mapping of α_i, then $d_1(\boldsymbol{l}_1, \boldsymbol{l}_i) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_j)$.*

Remark 2. From Lemma 2 and Corollary 1, the pairwise L_1-distances of the LHD(n, n) D generated by Algorithm 1 take at most $\lfloor n/2 \rfloor$ different values, which are included in the set $\{d_1(\boldsymbol{l}_1, \boldsymbol{l}_i), 2 \le i \le n\}$.

Example 4 (Example 3 continued)**.** For $N = 11$, consider the LHD$(5, 5)$ D constructed by Algorithm 1. It is easy to check that $\alpha_2^{-1} = \alpha_5, \alpha_3^{-1} = \alpha_4, \alpha_4^{-1} = \alpha_3$, and $\alpha_5^{-1} = \alpha_2$.

Then, we have

$$d_1(\boldsymbol{l}_1, \boldsymbol{l}_2) = d_1(\alpha_5(\boldsymbol{l}_1), \alpha_5(\boldsymbol{l}_2)) = d_1(\boldsymbol{l}_5, \boldsymbol{l}_1) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_5),$$

$$d_1(\boldsymbol{l}_1, \boldsymbol{l}_3) = d_1(\alpha_4(\boldsymbol{l}_1), \alpha_4(\boldsymbol{l}_3)) = d_1(\boldsymbol{l}_4, \boldsymbol{l}_1) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_4).$$

Therefore, the pairwise L_1-distances between the rows in D take at most two different values.

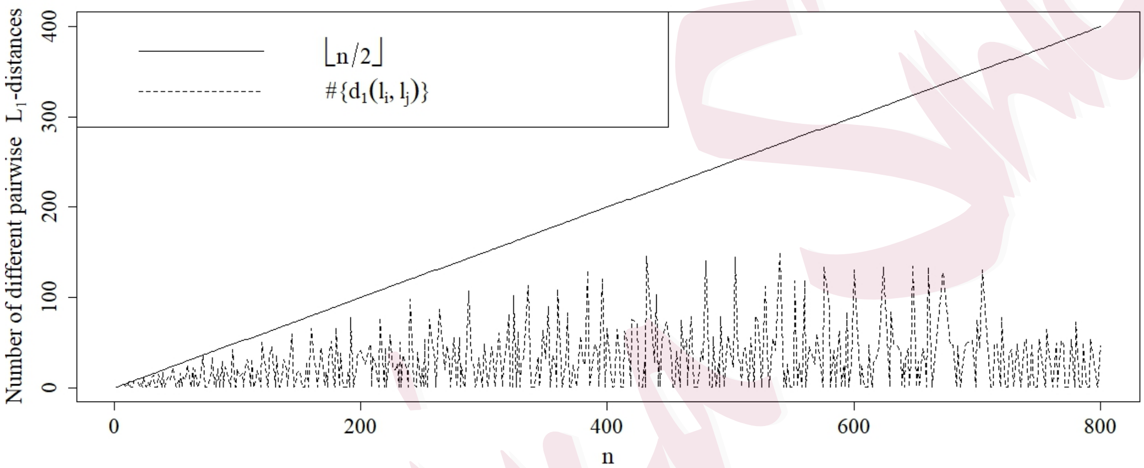

Figure 1: Maximum number of different values of the pairwise L_1-distances in the LHD(n, n)'s constructed by Algorithm 1.

In fact, for the LHD(n, n) D constructed by Algorithm 1, the number of different values of the pairwise L_1-distances between its rows is far less than $\lfloor n/2 \rfloor$ in most cases. For a given positive integer n, because there may be more than one LHD(n, n) that can be constructed from Algorithm 1, Figure 1 plots the maximum number of different values of the pairwise L_1-distances between different rows from among all possible such designs for each n ($n \leq 800$). From Figure 1, it is easy to see that there are few designs with $\lfloor n/2 \rfloor$ different values of the pairwise L_1-distances; in most cases, the number of different values of the pairwise L_1-distances is far less than $\lfloor n/2 \rfloor$.

Table 2: Pairwise L_1-distances of the LHD(n, n)'s generated by Algorithm 1.

N	n	$\#\{d_1(\boldsymbol{l}_i, \boldsymbol{l}_j)\}$	$d_1(D)$	$d_1(D)/d_{upper}$
11, 22	5	1	10	1
13, 26	6	1	14	1
17, 34	8	1	24	1
19, 38	9	1	30	1
25, 33	10	2, 3	34	0.94
23, 46	11	1	44	1
39	12	4	48	0.92
29, 58	14	1	70	1
31, 62	15	1	80	1
51	16	4	86	0.96
37, 74	18	1	114	1
41, 82	20	1	140	1
43, 86	21	1	154	1
69	22	5	162	0.96
47, 94	23	1	184	1
65	24	8	186	0.93
53, 106	26	1	234	1
81	27	3	244	0.97
87, 116	28	5, 6	262	0.97
59, 118	29	1	290	1

For further clarification, we consider $11 \le N \le 118$, and list the possible LHD$(n$,$n)$'s generated by Algorithm 1 with different n values in Table 2. We define the efficiency of an LHD(n, s) D under the maximin L_1-distance criterion as $d_1(D)/d_{upper}$, with $d_{upper} = \lfloor (n+1)s/3 \rfloor$ (Zhou and Xu, 2015). It is obvious that $d_1(D)/d_{upper} \le 1$, and a design with larger efficiency is preferable. When $d_1(D)/d_{upper} < 1$, we select the largest $d_1(D)/d_{upper}$, and give the corresponding two smallest N's (if they exist) with different $\#\{d_1(\boldsymbol{l}_i, \boldsymbol{l}_j)\}$(number of different pairwise L_1-distances for the same n). Table 2 shows that the number of different values of the pairwise L_1-distances is far less than

$\lfloor n/2 \rfloor$, and the LHD(n, n)'s constructed by the proposed method perform well under the maximin L_1-distance criterion.

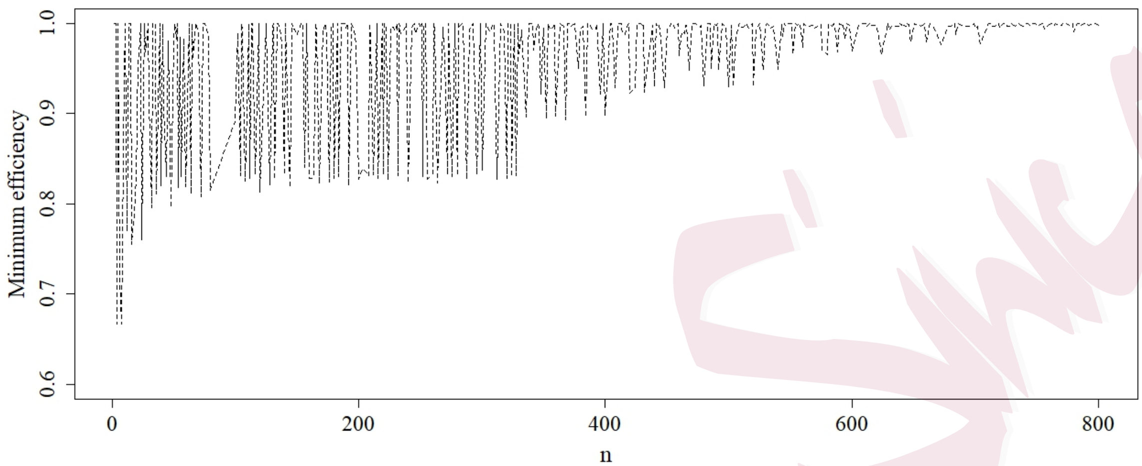

Figure 2: Minimum efficiencies of LHD(n, n)'s generated by Algorithm 1 for general N's, where $n = \phi(N)/2$.

Because there is more than one positive integer N that has the same value of the Euler function $\phi(\cdot)$, for a given positive integer n, there is more than one possible LHD(n, n) that can be constructed by Algorithm 1. To further explore the overall performance of the proposed method under the maximin L_1-distance criterion, Figure 2 plots the minimum efficiency for each n ($n \leq 800$). It is easy to see that the minimum efficiency of the constructed LHD(n, n) converges to one for large n. The proposed method can be used to generate large LHDs with large L_1-distances.

4. Theoretical Results and Comparisons

The proposed method generates optimal LHDs under the maximin L_1-distance criterion for different values of N. Next, we further explore the properties of the LHDs constructed by Algorithm 1 in different cases. Throughout the paper, we assume that p is an odd prime.

4.1 $N = p$ and $2p$

When $N = p$ and $2p$, the generator vectors in (3.1) are $\boldsymbol{h} = (1, 2, \ldots, n)$ and $(1, 3, \ldots, 2n-1)$, respectively, where $n = \phi(N)/2 = (p-1)/2$. It is easy to verify that the integer 3 divides n or $n+1$ for $p \geq 5$. The following result holds for a design D generated by Algorithm 1.

Theorem 1. *For $N = p$ or $2p$, and $n = \phi(N)/2 = (p-1)/2$, the* LHD(n, n) *D generated by Algorithm 1 is a maximin L_1-distance* LHD*, with its pairwise L_1-distances between rows all equal to $n(n+1)/3$.*

Remark 3. (i) Theorem 1 suggests that when N is an odd prime or twice an odd prime, the pairwise L_1-distances of D are all equal to a constant. We call such a design an *equidistant* LHD, which is a maximin L_1-distance LHD. (ii) Hence, by Theorem 3 in Sun, Wang, and Xu (2019), the constructed designs when $N = p$ and $2p$ are also uniform projection designs, which have good space-filling properties, not only in two dimensions, but also in all dimensions. (iii) When $N = p$, the LHD(n, n) D is the same as the design H constructed in Wang, Xiao, and Xu (2018). Then, by Theorem 7 in Wang, Xiao, and Xu (2018), we have that the average pairwise absolute correlation between columns of D, denoted by $\rho_{ave}(D)$, satisfies $\rho_{ave}(D) < 2/(n-1)$.

Example 5. For both $N = 13$ and 26, and $n = \phi(N)/2 = 6$, the generator vectors are $\boldsymbol{h} = (1, 2, 3, 4, 5, 6)$ and $\boldsymbol{h} = (1, 3, 5, 7, 9, 11)$, respectively. Table 3 lists the two LHD$(6, 6)$'s generated by Algorithm 1. Here, the pairwise L_1-distances between the rows of each design are all equal to 14, implying that both D_1 and D_2 are equidistant and maximin L_1-distance LHDs. In addition, if we permute the rows, columns, and elements of D_1 according to the permutation

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 4 & 5 & 2 & 3 & 6 \end{pmatrix}, \tag{4.1}$$

then the obtained design is D_2; that is, D_1 and D_2 are equivalent (i.e., they belong to the same isotopy class). This may not be true in general; see Table 2.

Table 3: Two LHD$(6, 6)$'s D_1 and D_2 generated by Algorithm 1 for $N = 13$ and 26.

D_1						D_2					
1	2	3	4	5	6	1	2	3	4	5	6
2	4	6	5	3	1	2	5	6	3	1	4
3	6	4	1	2	5	3	6	1	5	4	2
4	5	1	3	6	2	4	3	5	2	6	1
5	3	2	6	1	4	5	1	4	6	2	3
6	1	5	2	4	3	6	4	2	1	3	5

Consider two equidistant LHDs D_1 and D_2 for $N = p$ and $2p$, respectively, constructed by Algorithm 1, and let

$$D^* = [D_1, D_2]. \tag{4.2}$$

Then, we have the following result.

Theorem 2. *The* LHD$(n, 2n)$ D^* *defined in* (4.2) *is also equidistant, and thus a maximin L_1-distance LHD, with its pairwise L_1-distances between rows all equal to* $2n(n+1)/3$, *where* $n = (p-1)/2$.

Remark 4. The 1st and $(n+1)$th columns in D^* constructed by (4.2) are the same, and we denote the design obtained by deleting its $(n+1)$th column as D^*_{-1}. When we delete one column from an LHD with n rows, its L_1-distance reduces by at most $n-1$; thus, the L_1-distance of the LHD$(n, 2n-1)$ D^*_{-1} satisfies $d_1(D^*_{-1}) \geq (2n^2 - n + 3)/3$. In addition, it is easy to obtain that $d_1(D^*_{-1})/d_{upper} > 1 - 1/(n+1)$, which means that D^*_{-1} is an asymptotically optimal LHD, where $d_{upper} = \lfloor (n+1)(2n-1)/3 \rfloor$.

Theorem 2 is obvious from the equidistant property of the LHDs constructed by Algorithm 1 when $N = p$ and $2p$. Furthermore, if there are more than two equidistant LHDs with the same number of rows, we can generate larger maximin distance LHDs with more columns that are also equidistant.

Example 6 (Example 5 continued)**.** Consider $p = 13$. The two LHD$(6, 6)$'s D_1 and D_2 generated by Algorithm 1 for $N = p$ and $2p$, respectively, are listed in Table 3. From Theorem 1, it follows that they are both equidistant LHDs with $d_1(D_1) = d_1(D_2) = 14$. The corresponding LHD$(6, 12)$ D^* constructed in (4.2) is also equidistant, with $d_1(D^*) = 28$, which attains the upper bound of the L_1-distance. Because the first columns in each of the two designs listed in Table 3 are the same, we can obtain an LHD$(6, 11)$ D^*_{-1} by deleting one of the repeated columns. Then, $d_1(D^*_{-1}) = 23$, which is very close to the corresponding upper bound $d_{upper} = 25$.

For an LHD(n, n) constructed by Algorithm 1, by adding a row with its n elements all $n+1$, the obtained design has the same L_1-distance as the corresponding LHD(n, n), and the following result holds.

Lemma 3. *Let D be an equidistant* LHD(n, n) *constructed by Algorithm 1 for $N = p$ and $2p$, and let D' be the* LHD$(n+1, n)$ *obtained by adding a row of $(n+1)$'s to D. Then, $d_1(D') = d_1(D) = (n+1)n/3$, and*

$$d_1(D')/d_{upper} \geq 1 - 1/(n+2) \to 1 \text{ as } n \to \infty,$$

where $d_{upper} = \lfloor (n+2)n/3 \rfloor$.

Lemma 3 is obvious, and shows that D' is an asymptotically optimal design under the maximin L_1-distance criterion. In addition, when we delete any column from an LHD(n, s) D, its L_1-distance reduces by at most $n-1$. After repeating this procedure multiple times, we have the following result.

Lemma 4. *Let D be an equidistant* LHD(n, n) *constructed by Algorithm 1. Deleting any k_c columns yields an* LHD$(n, n-k_c)$*, denoted by D'. Then,*

$$d_1(D')/d_{upper} \geq 1 - 2k_c/(n-k_c).$$

If k_c is a fixed constant, not increasing with n, then $d_1(D')/d_{upper} \to 1$ as $n \to \infty$; that is, designs obtained by deleting columns from an equidistant LHD are asymptotically optimal LHDs with different sizes under the maximin L_1-distance criterion. Similar results hold for deleting columns from any (asymptotically) optimal design under the maximin L_1-distance criterion.

4.2 $N = 2^t$ and $2^t p$

When $N(\geq 16)$ is double even, that is, $N/2$ is an even integer, according to Lemma 1 in Elsawah, Fang, and Deng (2021), we have $n = \phi(N)/2 = \phi(N/2)$, and n is even. For a design D generated by Algorithm 1, denote D' as the submatrix of D that consists of its first $n/2$ columns. Then, we have the following result from Theorem 5 in Elsawah, Fang, and Deng (2021). We omit the proof.

Theorem 3. *For any double even integer $N(\geq 16)$, let $D = (l_{ij})$ be the* LHD(n, n) *generated by Algorithm 1, where $n = \phi(N)/2$. We have the following results:*

(i) *The elements in D satisfy $l_{ij} + l_{i(n+1-j)} = n + 1$ and $l_{ij} + l_{(n+1-i)j} = n + 1$, for any $i, j = 1, \ldots, n$, which implies*

$$D = \begin{pmatrix} A_1 & n+1-A_2 \\ n+1-A_3 & A_4 \end{pmatrix},$$

where A_1 is the $n/2 \times n/2$ leading principal submatrix of D, and A_2, A_3, and A_4 can be obtained from A_1 by reversing the orders of the columns, rows, and both, respectively;

(ii) *Denote D' as the $n \times n/2$ submatrix of D that consists of its first $n/2$ columns, that is, $D' = \begin{pmatrix} A_1 \\ n+1-A_3 \end{pmatrix}$. Then, D' is an* LHD$(n, n/2)$, *and $d_1(D') = d_1(D)/2$.*

Theorem 3 (i) shows that when $N(\geq 16)$ is double even, the corresponding LHD(n, n) generated by Algorithm 1 has a fold-over or mirror-symmetric structure with respect to both rows and columns.

Example 7. Consider the double even integers $N = 28$ and 32. The corresponding LHD$(6, 6)$ and LHD$(8, 8)$ constructed by Algorithm 1 are listed in Table 4. Divide

Statistica Sinica: Preprint
doi:10.5705/ss.202022.0263

each of the two LHDs into four blocks, as shown in Table 4. Then, it is easy to verify that property (i) in Theorem 3 holds. Let D_1' and D_2' be the 6×3 and 8×4 submatrices consisting of the first-half columns of D_1 and D_2, respectively. Then, $d_1(D_1') = d_1(D_1)/2 = 6$ and $d_1(D_2') = d_1(D_2)/2 = 11$.

Table 4: LHD(n, n)'s constructed by Algorithm 1 for double even integers $N = 28$ and 32.

D_1: LHD$(6,6)$ for $N = 28$						D_2: LHD$(8,8)$ for $N = 32$							
1	2	3	4	5	6	1	2	3	4	5	6	7	8
2	4	6	1	3	5	2	5	8	6	3	1	4	7
3	6	2	5	1	4	3	8	4	2	7	5	1	6
						4	6	2	8	1	7	3	5
4	1	5	2	6	3								
5	3	1	6	4	2	5	3	7	1	8	2	6	4
6	5	4	3	2	1	6	1	5	7	2	4	8	3
						7	4	1	3	6	8	5	2
						8	7	6	5	4	3	2	1

When $N = 4p$ and $n = \phi(N)/2 = p - 1$, the corresponding generator vector $\boldsymbol{h}$ consists of $p-1$ elements $\{2j-1, j = 1, \ldots, p\} \backslash \{p\}$. When $N = 2^t$ and $n = \phi(N)/2 = 2^{t-2}$, the corresponding generator vector is $\boldsymbol{h} = (1, 3, \ldots, 2n-1)$. We have the following results for $N = 2^t$ and $4p$.

Theorem 4. *Let D be the* LHD(n, n) *generated by Algorithm 1, with $n = \phi(N)/2$.*

(i) *If $N = 4p$ and $p \geq 5$, then $n = \phi(N)/2 = p - 1$ and*

$$d_1(D) = \begin{cases} n^2/3, & \text{if } p \pmod 3 = 1, \\ (n^2+2)/3, & \text{if } p \pmod 3 = 2; \end{cases}$$

(ii) *If $N = 2^t$ and $t \geq 3$, then $n = 2^{t-2}$ and*

$$d_1(D) = (n^2+2)/3.$$

In addition, for both cases, we have $d_1(D)/d_{upper} \geq 1 - 1/(n+1)$*, where* $d_{upper} = \lfloor (n+1)n/3 \rfloor$.

We can establish similar theoretical results for the constructed LHD(n,n)'s when $N = 2^t p$ $(t > 2)$, with more elaborate arguments; the details are omitted here. Figure 3 plots the efficiencies of the LHD(n,n)'s generated by Algorithm 1 when $N = 2^t p$ ($t = 3, 4$ and $16 < p < 200$), showing that the constructed designs perform well under the maximin L_1-distance criterion.

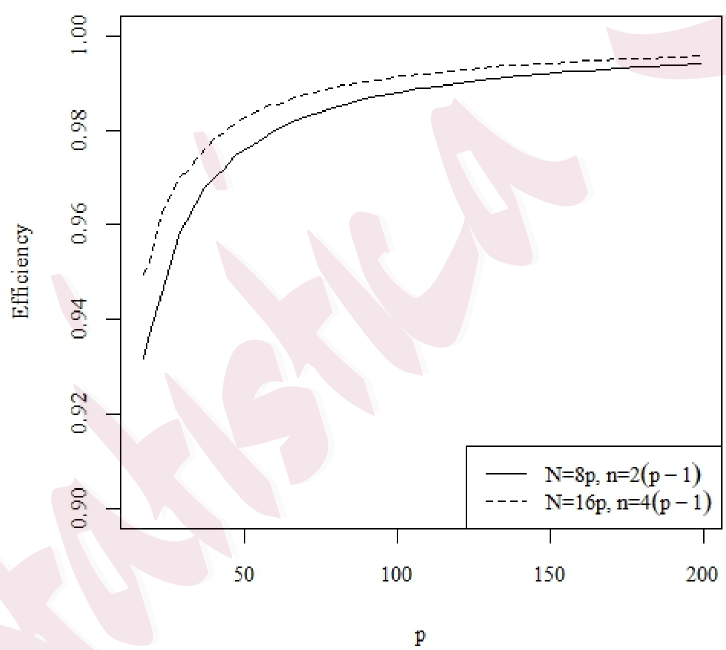

Figure 3: Efficiencies of LHD(n,n)'s generated by the proposed method for $N = 2^3 p$ and $2^4 p$.

Corollary 2. *From Theorems 3 and 4, the following results hold for the* LHD$(n, n/2)$ D'*:*

(i) *if* $N = 4p$ *and* $p \geq 5$, *then* $n = \phi(N)/2 = p - 1$ *and*

$$d_1(D') = \begin{cases} n^2/6, & \textit{if } p \pmod 3 = 1, \\ (n^2+2)/6, & \textit{if } p \pmod 3 = 2; \end{cases}$$

(ii) *if* $N = 2^t$ *and* $t \geq 4$, *then* $n = 2^{t-2}$ *and*

$$d_1(D') = (n^2+2)/6.$$

Because the upper bound of $d_1(D')$ is $d_{upper} = \lfloor (n+1)n/6 \rfloor$, it is easy to verify that $d_1(D')/d_{upper} \to 1$ as $n \to \infty$ for each case listed in Corollary 2; that is, the LHD$(n, n/2)$ D' is an asymptotically optimal design under the maximin L_1-distance criterion. More generally, when N is double even, for each LHD(n, n) constructed by the proposed method, the corresponding submatrix that consists of its first $n/2$ columns is asymptotically optimal under the maximin L_1-distance criterion, as long as the LHD(n, n) itself is asymptotically optimal.

In Figure 4, we compare the efficiencies of the LHD$(p-1, (p-1)/2)$'s generated by the linear permutation of good lattice point sets method ("LP-GLP," Zhou and Xu, 2015), the R package SLHD ("SLHD," Ba, Myers, and Brenneman, 2015), and the proposed method ("new method") in Algorithm 1 for $5 \leq p < 200$. Because the last row of a $p \times (p-1)$ good lattice point set D is $(0, \ldots, 0)$, the last row of the linear permutation good lattice point set D_b is $(b, \ldots, b)$, for $b = 0, 1, \ldots, p-1$. We use the leave-one-out method given in Wang, Xiao, and Xu (2018) to generate an LHD$(p-1, p-1)$ based on each D_b. Then, we can construct p LHD$(p-1, (p-1)/2)$'s by taking the first $(p-1)/2$ columns of each design. Of these p designs, we choose the one with the largest L_1-distance for comparison. The SLHD package generates optimal designs under the average reciprocal inter-point distance measure ϕ_r (Morris and Mitchell, 1995). Therefore, we run the command *maximinSLHD* with the option $t = 1$ and the default settings ($r = 15$) 100 times, and choose the design with the largest L_1-distance. For comparison, from the LHD$(p-1, p-1)$ generated by Algorithm 1 when $N = 4p$, we choose the first $(p-1)/2$ columns to obtain an LHD$(p-1, (p-1)/2)$, as stated in Theorem 3. Figure 4 shows that the proposed method outperforms the other two methods as p becomes larger. Moreover, the proposed method generates LHD$(p-1, (p-1)/2)$'s without a computer search for any given p.

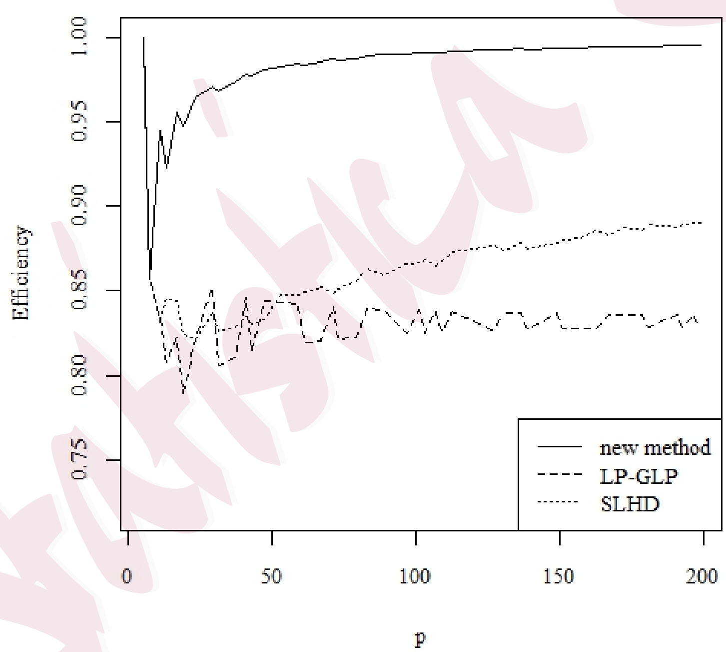

Figure 4: Efficiencies of the LHD$(p-1, (p-1)/2)$'s generated by various methods.

To further explore the performance of the constructed designs, we consider the maximin L_2-distance criterion. We define the efficiency of an LHD(n, s) under the

Statistica Sinica: Preprint
doi:10.5705/ss.202022.0263

L_2-distance as its L_2-distance divided by the corresponding upper bound d_2, where $d_2 = \sqrt{\lfloor n(n+1)s/6 \rfloor}$; see Theorem 3 in Zhou and Xu (2015). Figure 5 shows the efficiencies (under the L_2-distance) of the designs generated by various methods. For the R package SLHD, we run the command *maximinSLHD* with the option $t = 1$ and the default settings 100 times, and record the maximum L_2-distance of these designs. Figure 5 shows that the proposed method still outperforms the other two methods under the L_2-distance. When $p \geq 17$ (except when $p = 173$), the L_2-distances of the LHD$(p-1,(p-1)/2)$'s generated by the proposed method are larger than the maximum L_2-distances of the corresponding LHDs generated by the R package SLHD.

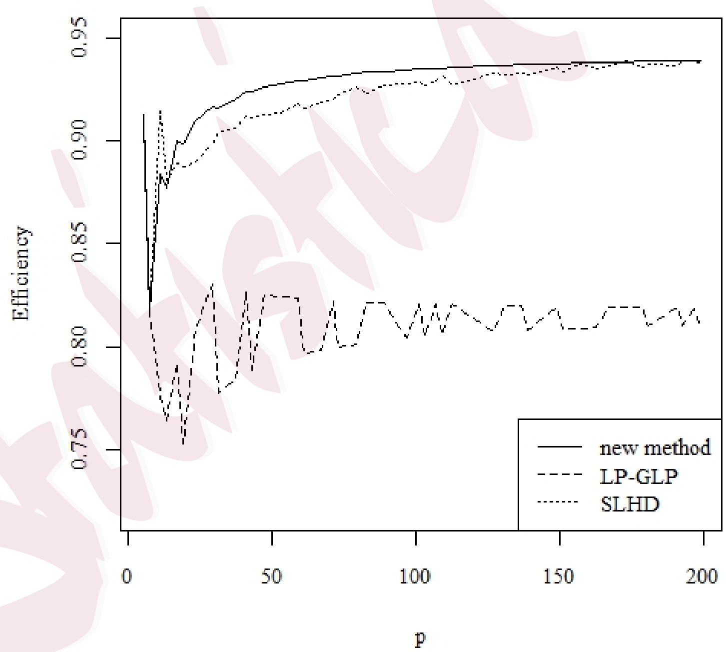

Figure 5: Efficiencies (under the L_2-distance) of the LHD$(p-1,(p-1)/2)$'s generated by various methods.

Statistica Sinica: Preprint
doi:10.5705/ss.202022.0263

We also compare these three methods under the ϕ_r ($r = 15$) criterion (where a smaller value is better). For the R package SLHD, we run the command *maximinSLHD* with the option $t = 1$ and the default settings 100 times, and record the minimum ϕ_r value of these designs. To better illustrate the performance of the three methods, we define the relative ϕ_r efficiency (where a smaller value is better) of a design D as $\phi_r(D)/\phi_r(D_{\text{SLHD}})$, where D_{SLHD} is the design generated by the R package SLHD as a reference. The relative ϕ_r efficiency may be larger or smaller than one, in contrast to the efficiencies defined for Figures 4 and 5. Figure 6 shows the relative ϕ_r efficiencies of the LHD$(p-1,(p-1)/2)$'s generated by various methods. Clearly the designs generated by the proposed method have smaller relative ϕ_r efficiencies. Thus, the proposed method outperforms the other two methods in terms of the ϕ_r criterion.

Note that the method of Lin and Kang (2016) can also be used to generate maximin LHDs under the ϕ_r criterion. Their numerical results show that the designs constructed using their method have larger ϕ_r values (thus, worse) than those of the designs constructed using the R package SLHD. In contrast, because our designs have smaller ϕ_r values than those of the designs constructed by the R package SLHD, our designs perform better than those obtained using the method of Lin and Kang (2016). As an example, using $N = 404$, we obtain an LHD$(100, 50)$. By deleting the last two columns and the last two rows, and rearranging the levels for each column, we obtain an LHD$(98, 48)$ with a ϕ_r value of 0.1096, which is better than any constructed using the method of Lin and Kang (2016) (whose smallest ϕ_r value is 0.1164). The ϕ_r values are evaluated on standardized designs with n levels, scaled to $[0.5/n, 1 - 0.5/n]$.

Statistica Sinica: Preprint
doi:10.5705/ss.202022.0263

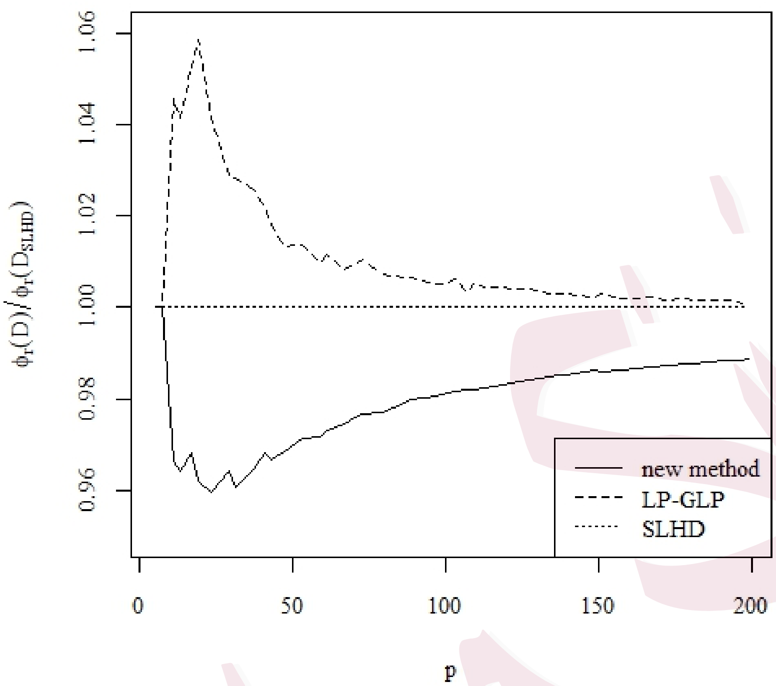

Figure 6: Relative ϕ_r efficiencies of the LHD$(p-1,(p-1)/2)$'s generated by various methods.

To conclude this section, Table 5 lists the possible sizes of the (asymptotically) maximin L_1-distance LHDs that can be obtained directly from the above theoretical results. Some designs are maximin L_1-distance LHDs and others are asymptotically optimal under the maximin L_1-distance criterion, and their efficiencies exceed 95% for $n \geq 50$.

4.3 Numerical studies

In this subsection, we further explore the properties of the LHD(n, n) D obtained from Algorithm 1 for more general N with $n = \phi(N)/2$ using simulations.

Table 5: Maximin L_1-distance LHDs obtained from the theoretical results.

Source	LHD(n, s)	$d_1(D)/d_{upper}$
Theorem 1	LHD$((p-1)/2, (p-1)/2)$	1
Theorem 2	LHD$((p-1)/2, p-1)$	1
Remark 4	LHD$((p-1)/2, p-2)$	$\geq 1 - 1/(n+1)$
Lemma 3	LHD$((p+1)/2, (p-1)/2)$	$\geq 1 - 1/(n+2)$
Lemma 4	LHD$((p-1)/2, (p-1)/2 - k_c)$	$\geq 1 - 2k_c/(n-k_c)$
Theorem 4	LHD$(p-1, p-1)$	$\geq 1 - 1/(n+1)$
Theorem 4	LHD$(2^{t-2}, 2^{t-2})$	$\geq 1 - 1/(n+1)$
Corollary 2	LHD$(p-1, (p-1)/2)$	$\geq 1 - 1/(n+1)$
Corollary 2	LHD$(2^{t-2}, 2^{t-3})$	$\geq 1 - 1/(n+1)$

Figure 7 shows the efficiencies of the LHD(n, n)'s generated by Algorithm 1 for $N = 5p, 7p, 11p$, and $13p$ $(13 < p < 200)$, with $n = 2(p-1), 3(p-1), 5(p-1)$, and $6(p-1)$, respectively. Figure 8 shows the efficiencies of the LHD(n, n)'s generated by the proposed method for $N = p^2$ and p^3 $(5 \leq p < 100)$, with $n = p(p-1)/2$ and $p^2(p-1)/2$, respectively. It is easy to see that the generated LHDs are all asymptotically maximin L_1-distance designs, and $d_1(D)$ approaches d_{upper} as p becomes larger. In general, when $N = p_1p_2$ or p_1^m (p_1, p_2 are odd primes, $m \geq 2$), the LHD(n, n)'s generated by Algorithm 1 are all asymptotically optimal designs under the maximin L_1-distance criterion. Furthermore, we can obtain addtional asymptotically optimal LHDs with different sizes by deleting columns (see Lemma 4) or rows (see Theorem 9 in Wang, Xiao, and Xu (2018)) from the constructed LHDs.

We give the following results on the L_1-distance of the constructed LHD(n, n) D for different N values. We have verified the results up to $p = 1000$:

$$d_1(D) \geq \begin{cases} \lfloor (4p^2 - 10p)/3 \rfloor + 2, & \text{when } N = 5p, n = 2(p-1), \\ 3p^2 - 7p + 6, & \text{when } N = 7p, n = 3(p-1). \end{cases}$$

Statistica Sinica: Preprint
doi:10.5705/ss.202022.0263

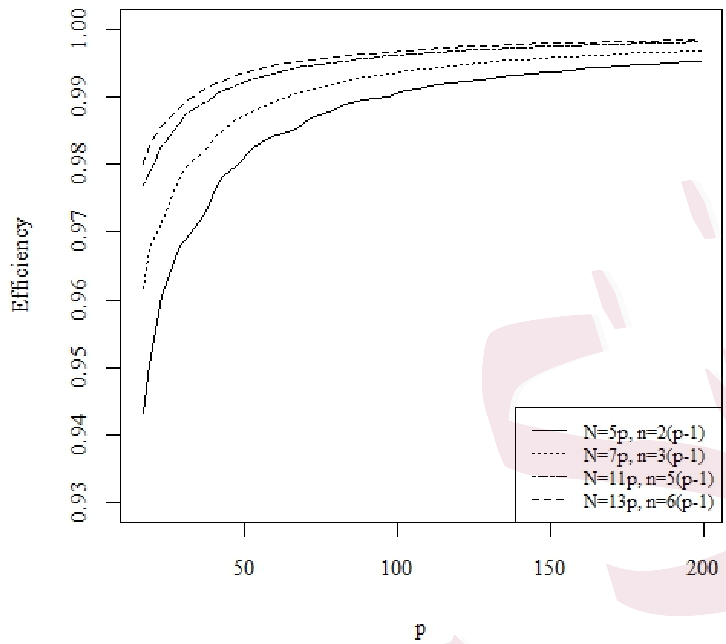

Figure 7: Efficiencies of LHD(n, n)'s generated by the proposed method for $N = 5p, 7p, 11p$, and $13p$.

Using simulations, we find that the lower bound is achieved by some N for either of the two cases. Moreover, the corresponding upper bounds for $N = 5p$ and $7p$ are $d_{upper} = \lfloor (4p^2 - 6p + 2)/3 \rfloor$ and $3p^2 - 5p + 2$, respectively. Thus, the efficiencies of the LHD(n, n)'s generated by Algorithm 1 for $N = 5p$ and $7p$ satisfy

$$d_1(D)/d_{upper} > \begin{cases} 1 - 2p/(2p^2 - 3p + 1), & \text{when } N = 5p, n = 2(p-1), \\ 1 - 2p/(3p^2 - 5p + 2), & \text{when } N = 7p, n = 3(p-1), \end{cases}$$

which implies that $d_1(D)/d_{upper} \to 1$ as $n \to \infty$ for a design D generated by Algorithm 1 when $N = 5p$ and $7p$.

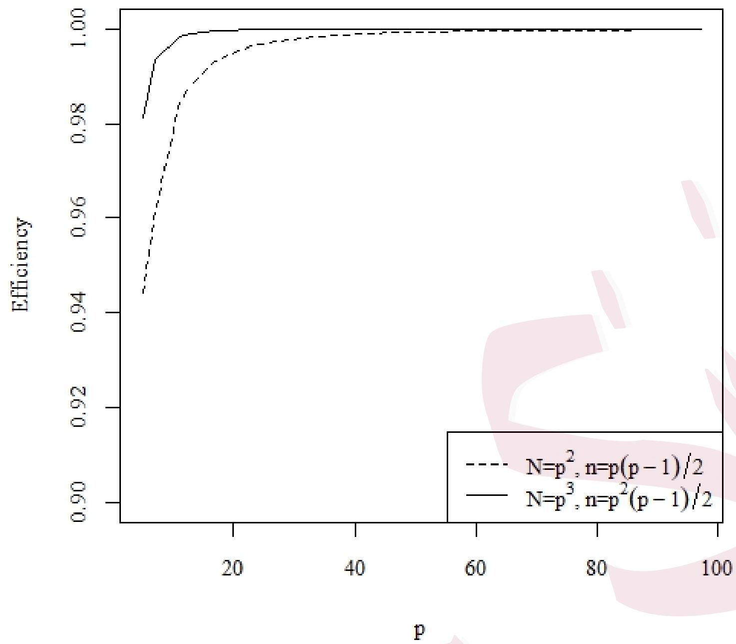

Figure 8: Efficiencies of LHD(n, n)'s generated by the proposed method for $N = p^2$ and p^3.

5. Conclusion

We have proposed a method for constructing maximin L_1-distance LHDs. Our theoretical results and numerical studies show that the proposed method can be used to generate (asymptotically) optimal LHDs that perform well under the maximin L_1-distance criterion. In particular, when $N = p$ and $2p$, the constructed LHDs are all equidistant LHDs; thus, they are maximin L_1-distance LHDs and uniform projection designs. Moreover, larger equidistant LHDs can be constructed by using two or more equidistant LHDs with the same number of rows. Section 4.3 provides lower bounds for the L_1-distances of the constructed LHDs for more general N using numerical computations. Additional theoretical support is possible with more elaborate arguments.

The maximin L_1-distance LHDs constructed using the proposed method are limited to special row and column sizes. This limitation is easy to overcome. Asymptotically optimal LHDs with flexible row and column sizes can be generated easily based on the constructed designs using Theorem 9 in Wang, Xiao, and Xu (2018). Furthermore, the integer programming algorithm of Vazquez and Xu (2024) can be used to obtain more flexible maximin L_1-distance designs based on the constructed LHDs.

Acknowledgments

The authors thank the associate editor and the two reviewers for their helpful comments. This work was supported by the National Natural Science Foundation of China (Grant Nos. 12001540, 12131001, 12226343 and 12371260), the National Ten Thousand Talents Program of China, and the China Scholarship Council (No. 202207080016).

Appendix: Proofs

A.1. Proof of Lemma 1

Let $L = (\boldsymbol{r}_1^T, \ldots, \boldsymbol{r}_n^T)^T$, where $\boldsymbol{r}_i$ is the ith row of L and ^{T} is the notation for transpose. It is obvious that $\boldsymbol{r}_1 = \boldsymbol{h}$ and $L^T = L$. To prove that L is a Latin square, it is sufficient to verify that each $\boldsymbol{r}_i$ $(i = 1, \ldots, n)$ is a permutation on the set $\{h_1, \ldots, h_n\}$.

Let $\boldsymbol{r}_i = (r_{i1}, \ldots, r_{in})$. For $k = 1, \ldots, n$, we have $r_{ik} = \min\{h_i * h_k \,(\mathrm{mod}\ N),\ N - h_i * h_k \,(\mathrm{mod}\ N)\}$. It is easy to check that $r_{ik} \leq \lfloor N/2 \rfloor$ and $\gcd(r_{ik}, N) = 1$, thus r_{ik} is an element of the set $\{h_1, \ldots, h_n\}$. As $\gcd(h_i, N) = 1$ $(1 \leq i \leq n)$, for any two entries r_{ik} and r_{iw} $(k \neq w)$, it is easy to obtain that $r_{ik} \neq r_{iw}$, otherwise, at least one of the following conditions holds: (1) N divides h_i, (2) N divides $h_k - h_w$, (3) N divides $h_k + h_w$, which leads to a contradiction. Consequently, each $\boldsymbol{r}_i$ is a permutation on the set $\{h_1, \ldots, h_n\}$, which completes the proof.

A.2. Proof of Lemma 2

Let $G = \{\alpha_1, \ldots, \alpha_n\}$. G is a commutative group if the following conditions hold:

(C1) if $\alpha, \beta \in G$, then $\alpha\beta \in G$;

(C2) the identity mapping is in G;

(C3) if $\alpha \in G$, then its inverse mapping α^{-1} is in G;

(C4) for any $\alpha, \beta \in G$, the equality $\alpha\beta = \beta\alpha$ holds.

Item (C2) holds obviously as $\alpha_1 (\in G)$ is an identity mapping, so only (C1), (C3), and (C4) need to be verified.

It is easy to see that the elements of Latin square L in (3.2) satisfy $r_{ik} = \min\{\pm h_i * h_k \,(\text{mod } N)\}$. Suppose p is an odd prime, we can prove the lemma in two cases.

(i) When $N = p\ (\geq 5)$ and $n = (p-1)/2$. The generator vector is $\boldsymbol{h} = (1, \ldots, n)$, thus the design $D = (l_{ij})_{n\times n}$ constructed by Algorithm 1 is the same as L. For $i = 1, \ldots, n$, we have

$$\alpha_i(k) = l_{ik} = r_{ik} = \min\{\pm i * k \,(\text{mod } N)\}, \text{where } k = 1, \ldots, n.$$

Choose another transformation α_j $(j \neq i)$ from G, then $\alpha_j(k) = \min\{\pm j * k \,(\text{mod } N)\}$

for $k = 1, \ldots, n$. The resultant of α_i and α_j can then be expressed as

$$\begin{aligned}\alpha_j\alpha_i(k) &= \alpha_j(\min\{\pm i * k \,(\mathrm{mod}\ N)\})\\ &= \min\{\pm j * i * k \,(\mathrm{mod}\ N)\}\\ &= \alpha_i\alpha_j(k),\end{aligned}$$

where $k = 1, \ldots, n$, so item (C4) holds. Since

$$\begin{aligned}\min\{\pm j * i * k \,(\mathrm{mod}\ N)\} &= \min\{\pm(j * i(\mathrm{mod}\ N)) * k \,(\mathrm{mod}\ N)\}\\ &= \min\{\pm \min\{\pm j * i \,(\mathrm{mod}\ N)\} * k \,(\mathrm{mod}\ N)\}\\ &= \min\{\pm w * k \,(\mathrm{mod}\ N)\},\end{aligned}$$

where $w = \min\{\pm j * i \ (\mathrm{mod}\ N)\} \in Z_n$; it is easy to verify that $\alpha_j\alpha_i(k) = \alpha_i\alpha_j(k) = \alpha_w(k)$, that is, $\alpha_j\alpha_i \in G$, so item (C1) holds.

For each α_i, there exists a unique integer j_0 ($1 \le j_0 \le n$) such that $\min\{\pm j_0 * i \,(\mathrm{mod}\ N)\} = 1$. Then α_{j_0} and α_i satisfy the following equality:

$$\begin{aligned}\alpha_{j_0}\alpha_i(k) = \alpha_i\alpha_{j_0}(k) &= \min\{\pm j_0 * i * k \,(\mathrm{mod}\ N)\}\\ &= \min\{\pm \min\{\pm j_0 * i \,(\mathrm{mod}\ N)\} * k \,(\mathrm{mod}\ N)\}\\ &= k,\end{aligned}$$

where $k = 1, \ldots, n$. That is, α_{j_0} is the inverse mapping of α_i, and for each α_i in G, its inverse mapping is also in G, so item (C3) holds.

(ii) When $N \ne p$ and $n = \phi(N)/2$. From Lemma 1, for any two integers i and k ($1 \le i, k \le n$), there exists a unique integer t ($1 \le t \le n$) satisfying

$$h_t = \min\{\pm h_i * h_k \,(\mathrm{mod}\ N)\},$$

which means $\alpha_i(k) = t$. In addition, for each h_i, there exists a unique integer i' ($1 \leq i' \leq n$) such that

$$\min\{\pm h_i * h_{i'} \,(\mathrm{mod}\ N)\} = h_1 = 1.$$

Then, similar to the discussions in case (i), it is easy to verify that items (C1), (C3), and (C4) hold.

In summary, G is a commutative group. This completes the proof.

A.3. Proof of Corollary 1

If the transformation α_j is the inverse mapping of the transformation α_i, that is, $(\alpha_i)^{-1} = \alpha_j$, then $(\alpha_i)^{-1}\alpha_1 = \alpha_j\alpha_1 = \alpha_j$, where α_1 is the identity mapping. Take the transformation α_j on the rows $\boldsymbol{l}_1$ and $\boldsymbol{l}_i$, then, these two rows are transformed to rows $\boldsymbol{l}_j$ and $\boldsymbol{l}_1$, respectively. Thus, according to the definition of L_1-distance of two row vectors, we have $d_1(\boldsymbol{l}_1, \boldsymbol{l}_i) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_j)$, which completes the proof.

A.4. Proof of Theorem 1

For a given integer N, define $w(x)$ as the modified Williams' transformation in Wang, Xiao, and Xu (2018), that is,

$$w(x) = \begin{cases} 2x, & \text{if } x < N/2; \\ 2(N-x), & \text{if } x \geq N/2. \end{cases}$$

When $N = p$, the generator vector in (3.1) is $\boldsymbol{h} = (1, \ldots, n)$, where $n = \phi(N)/2 = (p-1)/2$. Hence, the LHD(n, n) D generated by Algorithm 1 is the same as L in (3.2), and it can be verified that D is also the same as the design H constructed in Wang, Xiao, and Xu (2018) by modified Williams' transformation. Therefore, the result follows from Theorem 4 of Wang, Xiao, and Xu (2018).

For $N = 2p$ and $n = \phi(N)/2 = (p-1)/2$, let $U = (x_{ij})$ be the $N \times \phi(N)$ good lattice point design with generator vector $(1, 3, \ldots, p-2, p+2, \ldots, N-1)$. With proper row and column permutations, U is equivalent to

$$\begin{pmatrix} 2C + p \\ 2C \end{pmatrix} \pmod N$$

where C is the $p \times (p-1)$ good lattice point design.

Then $w(U)$ is equivalent to

$$\begin{pmatrix} w(2C \oplus p) \\ w(2C) \end{pmatrix},$$

where $2C \oplus p = (2C + p) \pmod N$. According to Theorem 1 and the proof of Theorem 8 in Wang, Xiao, and Xu (2018), the following result holds for the ith and kth rows, denoted by $\boldsymbol{r}_i$ and $\boldsymbol{r}_k$, in $w(2C)$,

$$d_1(\boldsymbol{r}_i, \boldsymbol{r}_k) = \frac{2(p^2-1)}{3}, \text{ for } i \neq k,\ i \neq p, k \neq p, \text{and } i + k \neq p. \tag{A.1}$$

Moreover, it can be verified that (A.1) also holds for $w(2C \oplus p)$.

In addition, when $N = 2p$, it can be verified that for the $n \times n$ Latin square L generated in (3.2), the following results hold: (i) its n elements are $\{1, 3, \ldots, p-2\}$; (ii) the L_1-distance of any two distinct rows in L is two times that of the corresponding rows in LHD(n, n) D constructed using Algorithm 1; (iii) under column permutations, L is equivalent to the submatrix of $w(2C \oplus p)/2$ that consists of its $((p+1)/2)$th to $(p-1)$th columns and 1st, 3rd, $\ldots, (p-2)$th rows. Hence, according to (A.1) and properties of good lattice point design U, for any two distinct rows in D, their L_1-distance equals

$(p^2-1)/12 = (n+1)n/3$, which means that $d_1(D) = (n+1)n/3$. Thus, the theorem holds.

A.5. Proof of Theorem 4

(i) For $N = 4p$ and $n = \phi(N)/2 = p-1$, the corresponding generator vector defined in (3.1) is $\boldsymbol{h} = (1, 3, \ldots, p-2, p+2, \ldots, 2p-1)$. Denote rows of the LHD$(n, n)$ D constructed by Algorithm 1 as $\boldsymbol{l}_1, \ldots, \boldsymbol{l}_n$. It is easy to see that the $p-1$ elements of $\boldsymbol{l}_1$ are $l_{1j} = j$, for $j = 1, \ldots, p-1$. For $\boldsymbol{l}_2$, its $p-1$ elements are

$$l_{2j} = \begin{cases} 2+3(j-1), & \text{for } j = 1, \ldots, (p-1)/6; \\ (p+1)/2 + 3[j-(p+5)/6], & \text{for } j = (p+5)/6, \ldots, (p-1)/3; \\ (p+5)/2 + 3[(p-1)/2 - j], & \text{for } j = (p+2)/3, \ldots, (p-1)/2; \\ p - 3[j-(p-1)/3], & \text{for } j = (p+1)/2, \ldots, 2(p-1)/3; \\ 3+3[j-(2p+1)/3], & \text{for } j = (2p+1)/3, \ldots, 5(p-1)/6; \\ 3[j-2(p-1)/3]-1, & \text{for } j = (5p+1)/6, \ldots, p-1, \end{cases}$$

when $p \pmod 3 = 1$, and

$$l_{2j} = \begin{cases} 2+3(j-1), & \text{for } j = 1, \ldots, (p+1)/6; \\ (p+3)/2 + 3[j-(p+7)/6], & \text{for } j = (p+7)/6, \ldots, (p+1)/3; \\ (p+5)/2 + 3[(p-1)/2 - j], & \text{for } j = (p+4)/3, \ldots, (p-1)/2; \\ 3+3[2(p-2)/3 - j], & \text{for } j = (p+1)/2, \ldots, 2(p-2)/3; \\ 1+3[j-(2p-1)/3], & \text{for } j = (2p-1)/3, \ldots, (5p-7)/6; \\ (p+1)/2 + 3[j-(5p-1)/6], & \text{for } j = (5p-1)/6, \ldots, p-1, \end{cases}$$

when $p \pmod 3 = 2$. Then, it can be calculated that $d_1(\boldsymbol{l}_1, \boldsymbol{l}_2) = n^2/3$ for $p \pmod 3) = 1$, and $d_1(\boldsymbol{l}_1, \boldsymbol{l}_2) = (n^2+2)/3$ for $p \pmod 3 = 2$.

For $\boldsymbol{l}_3$, it can be verified that its $p-1$ elements are

$$l_{3j} = \begin{cases} 3+5(j-1), & \text{for } j = 1, \ldots, n/10; \\ 2+5(j-1), & \text{for } j = n/10+1, \ldots, n/5; \\ 4+5(2n/5-j), & \text{for } j = n/5+1, \ldots, 3n/10; \\ 5+5(2n/5-j), & \text{for } j = 3n/10+1, \ldots, 2n/5; \\ 1+5(j-2n/5-1), & \text{for } j = 2n/5+1, \ldots, n/2; \\ n+1-l_{3(n+1-j)}, & \text{for } j = n/2+1, \ldots, n, \end{cases}$$

when $p \pmod 5 = 1$, and the corresponding L_1-distance $d_1(\boldsymbol{l}_1, \boldsymbol{l}_3) = n^2/3 + 4n/15 > d_1(\boldsymbol{l}_1, \boldsymbol{l}_2)$. Similarly, for p with other values or other rows in D, it can be verified that $d_1(\boldsymbol{l}_1, \boldsymbol{l}_i) \geq d_1(\boldsymbol{l}_1, \boldsymbol{l}_2)$ $(i = 3, \ldots, n)$ via some tedious calculations (details are omitted here). Therefore, the L_1-distance of design D is equal to the L_1-distance between its first two rows. That is, $d_1(D) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_2) = n^2/3$ for $p \pmod 3 = 1$, and $d_1(D) = d_1(\boldsymbol{l}_1, \boldsymbol{l}_2) = (n^2 + 2)/3$ for $p \pmod 3 = 2$.

(ii) For $N = 2^t$ and $n = \phi(N)/2 = 2^{t-2}$, the corresponding generator vector is $\boldsymbol{h} = (1, 3, \ldots, 2n - 1)$. Results on $d_1(D)$ can be proved similarly via some tedious calculations, so we omit the details.

In addition, for the constructed LHD(n, n) D in both cases, the upper bound of the L_1-distance is $d_{upper} = \lfloor (n+1)n/3 \rfloor$, hence, it is easy to verify that $d_1(D)/d_{upper} \geq 1 - 1/(n + 1)$. This completes the proof.

References

Ba, S., Myers, W. R. and Brenneman, W. A. (2015). Optimal sliced Latin hypercube designs. *Technometrics* **57**, 479–487.

Elsawah A. M., Fang, K. T. and Deng Y. H. (2021). Some interesting behaviors of good lattice point sets. *Communications in Statistics-Simulation and Computation* **50**, 3650–3668.

Fang, K. T., Li, R. Z. and Sudjianto, A. (2006). *Design and Modeling for Computer Experiments*. Chapman and Hall/CRC, New York.

Fang K. T., Liu M. Q., Qin H. and Zhou Y. D. (2018). *Theory and Application of Uniform Experimental Designs*. Springer and Science Press, Singapore and Beijing.

He, X. (2019). Interleaved lattice-based maximin distance designs. *Biometrika* **106**, 453–464.

Hedayat, A. S., Sloane, N. J. A. and Stufken, J. (1999). *Orthogonal Arrays: Theory and Applications.* Springer, New York.

Johnson, M. E., Moore, L. M. and Ylvisaker, D. (1990). Minimax and maximin distance designs. *J. Statist. Plann. Inference* **26**, 131–148.

Joseph, V. R. and Hung, Y. (2008). Orthogonal-maximin Latin hypercube designs. *Statist. Sinica* **18**, 171–186.

Keedwell, A. D. and Dénes, J. (2015). *Latin Squares and Their Applications (2nd Edition).* Academic Press, New York.

Kleijnen, J. P. (1997). Sensitivity analysis and related analyses: a review of some statistical techniques. *J. Statist. Comp. Simul.* **57**, 111-142.

Li, W., Liu, M. Q. and Tang, B. (2021). A method of constructing maximin distance designs. *Biometrika* **108**, 845–855.

Lin, C. D. and Kang, L. (2016). A general construction for space-filling Latin hypercubes. *Statist. Sinica* **26**, 675–690.

Lin, C. D., Mukerjee, R. and Tang, B. (2009). Construction of orthogonal and nearly orthogonal Latin hypercubes. *Biometrika* **96**, 243–247.

Lin, C. D. and Tang, B. (2015). Latin hypercubes and space-filling designs. In *Handbook of Design and Analysis of Experiments* (Edited by A. Dean, M. Morris, J. Stufken and D. Bingham), pp. 593–625. CRC Press, Boca Raton.

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.

Montgomery, D. C. (2008). *Design and Analysis of Experiments.* John Wiley and Sons, New York.

Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics* **33**, 161–174.

Morris, M. D. and Mitchell, T. J. (1995). Exploratory designs for computational experiments. *J. Statist. Plann. Inference* **43**, 381–402.

Owen, A. B. (1992). Orthogonal arrays for computer experiments, integration and visualization. *Statist. Sinica* **2**, 439–452.

Pang, F., Liu, M. Q. and Lin, D. K. J. (2009). A construction method for orthogonal Latin hypercube designs with prime power levels. *Statist. Sinica* **19**, 1721–1728.

Santner, T. J., Williams, B. J. and Notz, W. I. (2018). *The Design and Analysis of Computer Experiments (2nd Edition)*. Springer, New York.

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

Sun, F. S., Liu, M. Q. and Lin, D. K. J. (2009). Construction of orthogonal Latin hypercube designs. *Biometrika* **96**, 971–974.

Sun, F. S., Liu, M. Q. and Lin, D. K. J. (2010). Construction of orthogonal Latin hypercube designs with flexible run sizes. *J. Statist. Plann. Inference* **140**, 3236–3242.

Sun, F. S., Pang, F. and Liu, M. Q. (2011). Construction of column-orthogonal designs for computer experiments. *Sci. China Math.* **54**, 2683–2692.

Sun, F. S., Wang, Y. and Xu, H. (2019). Uniform projection designs. *Ann. Statist.* **47**, 641–661.

Tang, B. (1993). Orthogonal array-based Latin hypercubes. *J. Amer. Statist. Assoc.* **88**, 1392–1397.

Vazquez, A. R. and Xu, H. (2024). An integer programming algorithm for constructing maximin distance designs from good lattice point sets. *Statist. Sinica* **34**, DOI: 10.5705/ss.202021.0362.

Wang, L., Sun, F. S., Lin, D. K. J. and Liu, M. Q. (2018). Construction of orthogonal symmetric Latin hypercube designs. *Statist. Sinica* **28**, 1503–1520.

Wang, L., Xiao, Q. and Xu, H. (2018). Optimal maximin L_1-distance Latin hypercube designs based on good lattice point designs. *Ann. Statist.* **46**, 3741–3766.

Xiao, Q. and Xu, H. (2017). Construction of maximin distance Latin squares and related Latin hypercube designs. *Biometrika* **104**, 455-464.

Zhou, W., Yang, J. F. and Liu, M. Q. (2020). Optimal maximin L_2-distance Latin hypercube designs. *J. Statist. Plann. Inference* **207**, 113–122.

Zhou, Y. and Xu, H. (2015). Space-filling properties of good lattice point sets. *Biometrika* **102**, 959–966.

School of Statistics and Mathematics, Zhongnan University of Economics and Law, Wuhan 430073, China

E-mail: nkyuan@126.com

Department of Statistics and Data Science, University of California, Los Angeles, California 90095, U.S.A.

E-mail: yuhaoyin@ucla.edu

Department of Statistics and Data Science, University of California, Los Angeles, California 90095, U.S.A.

E-mail: hqxu@stat.ucla.edu

NITFID, LPMC & KLMDASR, School of Statistics and Data Science, Nankai University, Tianjin 300071, China

E-mail: mqliu@nankai.edu.cn