

Statistica Sinica Preprint No: SS-2025-0169	
Title	Maximin Distance Designs for Mixed Continuous, Ordinal, and Binary Variables
Manuscript ID	SS-2025-0169
URL	http://www.stat.sinica.edu.tw/statistica/
DOI	10.5705/ss.202025.0169
Complete List of Authors	Hui Lan and Xu He
Corresponding Authors	Xu He
E-mails	hexu@amss.ac.cn
Notice: Accepted author version.	

MAXIMIN DISTANCE DESIGNS FOR MIXED CONTINUOUS, ORDINAL, AND BINARY VARIABLES

Hui Lan and Xu He*

Beijing University of Technology and Chinese Academy of Sciences

Abstract: Computer experiments are widely used to study complex real-world systems. Space-filling designs, which aim to provide adequate coverage of the input domain, are commonly used to facilitate informative sampling and accurate surrogate modeling. Although maximin distance designs for continuous variables have been extensively studied, general-purpose constructions for mixed input types remain relatively limited. This paper develops interleaved lattice designs under the maximin distance criterion for input spaces comprising continuous, ordinal, and binary variables. The proposed framework accommodates a range of run sizes, combinations of variable types, and prespecified level sets for ordinal variables. We develop three main construction algorithms, supplemented by seven supporting procedures and corresponding theoretical results. These algorithms include an exhaustive search for low-dimensional problems and computationally tractable constructions for moderate- and high-dimensional problems. In the numerical settings considered, the proposed designs attain larger separation distances than the benchmark designs in most cases and yield lower or comparable prediction errors under the specified Gaussian process emulators.

Key words and phrases: Maximin distance design, mixed-variable design, interleaved lattice.

*Corresponding author. E-mail: hexu@amss.ac.cn

1. Introduction

Computer experiments are widely used to study complex real-world systems (Santner et al., 2003). To maximize the information gained from computer runs and facilitate accurate surrogate modeling, design points are typically spread throughout the input space, a property commonly referred to as “space-filling”. A commonly used criterion for comparing designs in computer experiments is the L_2 separation distance, defined as the minimum Euclidean distance between any two distinct design points, $\rho(\mathcal{D}) = \min_{\mathbf{x}, \mathbf{y} \in \mathcal{D}, \mathbf{x} \neq \mathbf{y}} \|\mathbf{x} - \mathbf{y}\|_2$. A design is called a maximin distance design if it attains the largest possible separation distance among all designs with the same input space and run size n (Johnson et al., 1990). Under the conditions considered by Haaland and Qian (2011); Haaland et al. (2018), the maximin distance criterion is also related to asymptotic D-optimality for Gaussian process emulation and to bounds on numerical and approximation errors.

Considerable effort has been devoted to constructing maximin distance designs for the continuous input space $\mathcal{G} = [0, 1]^p$. For two- and three-dimensional problems, the best-known configurations are catalogued in the Packomania database. Search-based methods for general p include those developed by Trosset et al. (1999), Stinstra et al. (2003), and Mu and Xiong (2017), but these methods can become computationally demanding when either n or p is large. To reduce the search space, Morris and Mitchell (1995) considered Latin hypercube designs (LHDs), while He (2019) restricted the search to interleaved lattice-based (IL-based) designs and identified most of the best-known designs for $p > 3$. Other studies have developed direct constructions under specific structural conditions (Xiao and Xu, 2017;

Wang et al., 2018; Li et al., 2021; Yuan et al., 2025). Although some of these constructions attain optimal separation distances, their applicability can be restricted to particular values of n and p or to particular relationships between them.

However, computer experiments often involve ordinal or nominal categorical variables. An ordinal variable is a discrete factor with finitely many naturally ordered levels, whereas a nominal categorical variable has levels with no inherent ordering. As an illustration, Magmasoft V4.4, a software package for simulating alloy casting processes, includes five grades of grey cast iron whose names and compositions are listed in Table 1. When the ordering of these grades is scientifically relevant, this input can be represented as an ordinal variable with five allowable levels. Ordinal levels can be assigned numerical scores that preserve the ordering, enabling space-filling designs based on distances between the assigned scores. For nominal categorical variables, which have no intrinsic ordering or metric, design construction commonly emphasizes balanced representation and estimability, with dummy-variable coding used for subsequent modeling. In this work, the two levels of a binary categorical variable are numerically coded and handled within the same design framework as ordinal inputs with two allowable levels, i.e., 0 and 1. This coding is used only for design construction and does not imply an ordinal scale.

A central challenge in designing experiments with mixed continuous, ordinal, and binary variables is that the design space is no longer a regular hypercube but a hybrid continuous-discrete domain contained in $[0, 1]^d$. Consequently, the design construction must ensure that all design points lie within this subset. For the alloy casting example, the ordinal input is restricted to the five prescribed levels. In such hybrid spaces, the feasible set is

Table 1: Composition percentages for five grey cast iron grades.

Name	Carbon	Silicon	Manganese	Phosphorus	Sulfur
GJL-150	3.55	2.35	0.65	0.20	0.10
GJL-200	3.35	2.05	0.70	0.20	0.10
GJL-250	3.25	2.05	0.70	0.15	0.10
GJL-300	3.20	1.60	0.80	0.10	0.10
GJL-350	3.00	1.40	0.90	0.10	0.10

disconnected with respect to the discrete coordinates, so standard continuous optimization methods and gradient-based search strategies do not apply directly. One heuristic approach first constructs a continuous maximin design and then projects it onto the discrete levels. This procedure can degrade the space-filling properties and may even produce duplicate design points.

Methods for constructing designs involving ordinal inputs are relatively scarce. A representative approach is the sliced LHD of Ba et al. (2015), which was developed for computer experiments involving continuous and categorical variables and optimized under a maximin distance criterion. If ordinal variables are treated as categorical and their ordering is ignored, this construction can also be applied to designs involving both continuous and ordinal variables. In addition, Sun et al. (2019) considered maximin designs restricted to a finite candidate set of allowable input points.

Beyond maximin distance designs, Deng et al. (2015) proposed marginally coupled designs (MCDs) for experiments involving mixed qualitative and quantitative variables. For

MCDs constructed from orthogonal arrays, the admissible run sizes and level combinations depend on the availability of suitable orthogonal arrays. Joseph et al. (2020) proposed a maximum projection (MaxPro) method for constructing designs involving continuous, nominal, discrete numeric, and ordinal variables. These designs are obtained through numerical optimization under the MaxPro criterion (Joseph et al., 2015). The MaxPro criterion emphasizes space-filling in low-dimensional projections and is particularly useful when only a subset of variables is expected to be active and their identities are unknown in advance (Joseph et al., 2015).

By contrast, the maximin distance criterion emphasizes separation in the full unprojected input space. When prior information suggests that different variables should receive different levels of emphasis, IL-based maximin distance designs (He, 2019) can incorporate such information through a weighted L_2 distance, $d_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = \{\sum_{k=1}^p [w_k(x_k - y_k)]^2\}^{1/2}$, where $\mathbf{w} = (w_1, \dots, w_p) \in (0, \infty)^p$ is a prespecified weight vector. A larger w_k places greater emphasis on separation along the k th coordinate, whereas the standard L_2 distance corresponds to $\mathbf{w} = (1, \dots, 1)$. For mixed variable types, the use of a weighted distance is particularly important because the difference between the two levels of a binary variable is generally on a different scale from that between the two endpoint values of a numeric variable. Relative weights may be informed by sensitivity measures obtained from prior studies or pilot analyses, such as Sobol' indices (Sobol', 1993). The resulting weighted maximin problem can be reduced to an unweighted problem by rescaling the coordinate ranges. For a set $\mathcal{S} \subset \mathbb{R}^p$ and a vector $\mathbf{v} = (v_1, \dots, v_p)$, let $\mathcal{S} \otimes \mathbf{v} = \{(x_1 v_1, \dots, x_p v_p) : \mathbf{x} \in \mathcal{S}\}$ denote coordinatewise rescaling. As shown by He (2019), for a fixed run size, a design $\mathcal{D}$

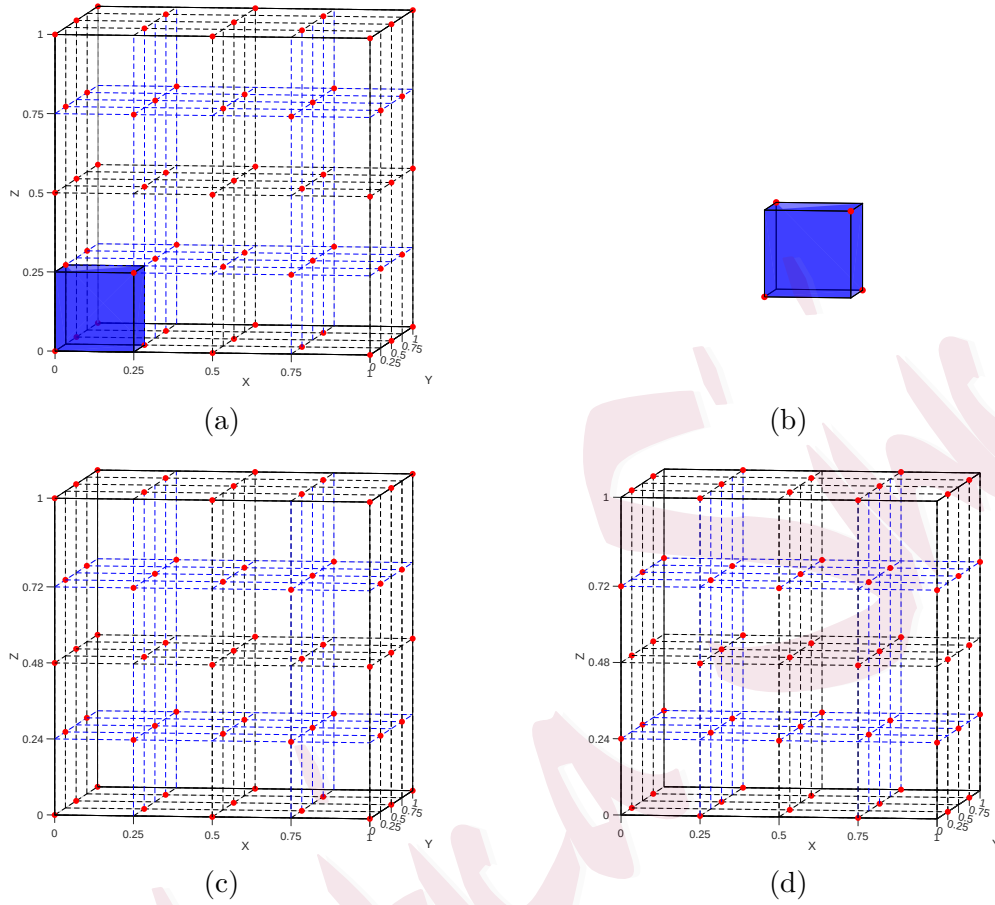


Figure 1: (a) An IL-based maximin distance design for continuous variables (He, 2019), with $p = 3$ and $n = 63$. (b) The local structure $\mathcal{H} = \mathcal{L} \cap [0, 1]^p$ of the design in (a), with its formal definition given in Section 2. (c) An IL-based maximin distance design for mixed-type variables, with $p = 3$, $n = 63$, and shift vector $\mathbf{u} = (0, 0, 0)$. (d) An IL-based maximin distance design for mixed-type variables, with $p = 3$, $n = 62$, and shift vector $\mathbf{u} = (0, 0, 1)$.

is maximin in $[0, 1]^p$ under $d_{\mathbf{w}}$ if and only if $\mathcal{D} \otimes \mathbf{w}$ is maximin in $\prod_{k=1}^p [0, w_k]$ under the standard L_2 distance. For simplicity, we use the unweighted distance measure in this paper and assume that the design space is a subset of $\prod_{k=1}^d [0, w_k]^d$. In particular, the allowable levels for binary variables are 0 and w_k .

To construct maximin distance designs with mixed continuous and ordinal variables, we restrict attention to interleaved lattice structures following He (2019). A fully unrestricted numerical search over mixed design spaces quickly becomes computationally infeasible for

moderate to large problems. Although Latin hypercube structures are commonly used to reduce search complexity in purely continuous settings, they are not suitable for ordinal variables and therefore cannot be imposed in the present setting. Algebraic construction techniques developed for continuous design spaces in $[0, 1]^p$ likewise do not extend directly to settings involving discretely valued factors. For instance, although space-filling designs constructed from orthogonal arrays can accommodate discretely valued factors, the limited availability of mixed-level orthogonal arrays makes it difficult to construct, by algebraic means, designs with unequal numbers of levels, as required when ordinal variables have heterogeneous allowable level sets and categorical variables are also present. Interleaved lattices are naturally compatible with the geometry of design spaces containing discrete components and allow the number of levels to be expanded systematically and scalably. Most importantly, interleaved lattices achieve asymptotically optimal separation distances as $n \rightarrow \infty$ for $p = 2, 3, 4, 5, 7, 8$ (Conway and Sloane, 2013).

Figure 1 illustrates how an IL structure accommodates ordinal and continuous variables. Panel (a) shows the 63-run maximin distance design of He (2019) for $\mathcal{G} = [0, 1]^3$, formed from translated copies of the local configuration in Panel (b). Because all points lie on $\{0, 0.25, 0.5, 0.75, 1\}^3$, the design is also maximin on this discrete grid and on $[0, 1]^2 \times \{0, 0.25, 0.5, 0.75, 1\}$. More generally, coordinate levels can be modified to accommodate pre-specified, possibly unequally spaced ordinal levels while preserving the underlying lattice structure. Our method maximizes separation distance over admissible coordinate configurations within the corresponding IL-based design class. Panels (c) and (d) illustrate this modification, showing designs with the same separation distance but different run sizes and

shifts, namely 63 runs with $\mathbf{u} = (0, 0, 0)$ and 62 runs with $\mathbf{u} = (0, 0, 1)$, respectively.

Motivated by these considerations, we make three main contributions. First, we develop an IL-based methodology for constructing maximin distance designs with mixed continuous, ordinal, and binary variables. The methodology accommodates a range of run sizes, dimensions, numbers of ordinal variables, and prespecified ordinal level sets, subject to the admissibility conditions of the construction. Second, we establish theoretical results for the proposed class of IL-based designs and develop three construction algorithms tailored to different dimensional settings. The low-dimensional algorithm exhaustively searches the specified IL class, while the other two algorithms reduce the search space for moderate- and high-dimensional problems. Seven auxiliary procedures support these constructions by handling tasks such as level selection, lattice construction, candidate initialization, and shift selection. Third, we assess the proposed designs through numerical studies. In most settings examined, they attain larger separation distances than the benchmark designs and yield lower or comparable prediction errors under the specified emulators. The reported computing times also indicate that the proposed procedures can construct large designs using standard computing resources in the settings considered. Source code for generating the proposed designs is provided in the R package “LatticeDesign”, available on CRAN.

2. Preliminaries

In this section, we briefly review interleaved lattices (ILs) and IL-based designs for continuous variables following He (2019).

A full-rank lattice $\mathcal{L} \subset \mathbb{R}^p$ consists of all integer combinations of p linearly independent

vectors and is therefore a discrete additive subgroup of $\mathbb{R}^p$. Equivalently, $\mathcal{L} = \mathcal{L}(\mathbf{G}) = \{\mathbf{z}^T \mathbf{G} : \mathbf{z} \in \mathbb{Z}^p\}$, where $\mathbf{G}$ is a full-rank $p \times p$ generator matrix. In particular, the p -dimensional integer lattice and even integer lattice are $\mathbb{Z}^p = \mathcal{L}(\mathbf{I}_{p \times p})$ and $\mathbb{E}^p = \mathcal{L}(2\mathbf{I}_{p \times p})$, respectively, where $\mathbf{I}_{p \times p}$ denotes the $p \times p$ identity matrix. A lattice $\mathcal{L}$ is called a standard IL if $\mathbb{E}^p \subseteq \mathcal{L} \subseteq \mathbb{Z}^p$ and its projection onto each coordinate is $\mathbb{Z}$. Standard ILs in low dimensions are enumerated in He (2019). For a set $\mathcal{S} \subset \mathbb{R}^p$ and a vector $\mathbf{u} \in \mathbb{R}^p$, let $\mathcal{S} \oplus \mathbf{u}$ denote the translated set $\{\mathbf{x} + \mathbf{u} : \mathbf{x} \in \mathcal{S}\}$. Any standard IL $\mathcal{L}$ can then be expressed as $\mathcal{L} = \bigcup_{\mathbf{z} \in \mathbb{Z}^p} (\mathcal{H} \oplus (2\mathbf{z}))$, where $\mathcal{H} = \mathcal{L} \cap \{0, 1\}^p$ is a finite local point set that determines $\mathcal{L}$. For instance, Figure 1(a) illustrates a three-dimensional IL-based design constructed from $\mathcal{L} = \mathcal{L}(\mathbf{G}_{3,2})$, where $\mathbf{G}_{3,2}$ is defined in (3.4). Figure 1(b) shows the corresponding local point set $\mathcal{H}$, whose translated copies generate the entire lattice $\mathcal{L}$.

A design $\mathcal{D}$ in $\prod_{k=1}^p [0, w_k]$ is called an interleaved lattice-based design for continuous variables (ILDC) if $\mathcal{D} = [(\mathcal{L} \oplus \mathbf{u}) \otimes \mathbf{v}] \cap \prod_{k=1}^p [0, w_k]$, where $\mathcal{L}$ is a standard IL, $\mathbf{v} \in (0, \infty)^p$ is a scale vector, and $\mathbf{u} \in \mathbb{Z}^p$ is a shift vector. Thus, $\mathcal{D}$ consists of shifted and coordinatewise rescaled lattice points that lie within the input space.

For a target run size n , an interleaved lattice-based maximin distance design for continuous variables (ILMDC) is an ILDC with at least n runs that maximizes the separation distance within the ILDC class. To parameterize the ILDC construction, let $\mathbf{s} = (s_1, \dots, s_p)$ denote the span vector, where $s_k \geq 2$ is the number of distinct levels in the k th coordinate. Setting $v_k = w_k/s_k - 1$, $k = 1, \dots, p$, yields

$$\mathcal{D} = \left[(\mathcal{L} \oplus \mathbf{u}) \cap \prod_{k=1}^p [0, s_k - 1] \right] \otimes \left(\frac{w_1}{s_1 - 1}, \dots, \frac{w_p}{s_p - 1} \right). \quad (2.1)$$

Because $\mathbb{E}^p = 2\mathbb{Z}^p \subseteq \mathcal{L}$, it suffices to consider shift vectors $\mathbf{u} \in \{0, 1\}^p$. Moreover, He

(2019) showed that, without loss of optimality, the shift vector can be set to $\mathbf{u} = \mathbf{0}_p$, where $\mathbf{0}_p$ denotes the zero vector. Consequently, constructing an LMDC reduces to searching over feasible pairs $(\mathcal{L}, \mathbf{s})$ that satisfy the run-size requirement and maximize the separation distance. By comparison, directly optimizing an unrestricted n -run design in $\prod_{k=1}^p [0, w_k]$ requires optimizing np coordinates, resulting in a nonsmooth, nonconvex optimization problem with potentially many local optima.

3. Methodology for mixed-type variables

3.1 General expression

This section presents the construction of interleaved lattice-based maximin distance designs for mixed-type variables, hereafter referred to as ILMDMs.

Let $\mathcal{G}_k$ denote the input space of the k th variable. For continuous variables, $\mathcal{G}_k = [0, w_k]$, where $w_k > 0$ is the prespecified coordinate weight introduced in Section 2. For ordinal variables, $\mathcal{G}_k$ is a finite subset of $[0, w_k]$ containing both 0 and w_k . For binary variables, $\mathcal{G}_k = \{0, w_k\}$. The overall input space is $\mathcal{G} = \prod_{k=1}^p \mathcal{G}_k$. Given $\mathcal{G}$ and a target run size n , an ILMDM is an interleaved lattice-based design for mixed-type variables (ILDM) $\mathcal{D} \subset \mathcal{G}$ with at least n runs that maximizes the separation distance within the ILDM class.

We first present a general representation of ILDMs that serves as the basis for the subsequent design construction. We consider ILDMs of the form

$$\mathcal{D} = \left\{ (\mathcal{Y}_{1(i_1)}, \dots, \mathcal{Y}_{p(i_p)}) : (i_1, \dots, i_p) \in [(\mathcal{L} \oplus \mathbf{u}) \cap \prod_{k=1}^p [0, s_k - 1]] \oplus \mathbf{1}_p \right\}, \quad (3.2)$$

where $\mathbf{1}_p$ denotes the p -dimensional vector of ones, $\mathcal{Y}_{k(i)}$ denotes the i th smallest element of $\mathcal{Y}_k$, and $\mathcal{Y}_k \subset \mathcal{G}_k$ specifies the levels used in the k th coordinate of $\mathcal{D}$. Let $\mathcal{Y} = \prod_{k=1}^p \mathcal{Y}_k \subset \mathcal{G}$

3.1 General expression

denote the overall level set of the design. This representation generalizes (2.1) by allowing coordinate-specific level sets. If $\mathcal{Y}_k = \{iw_k/(s_k - 1) : i = 0, 1, \dots, s_k - 1\}$ for every k , then (3.2) reduces to (2.1). Thus, $\mathcal{D}$ is fully determined by $\mathcal{L}$, $\mathbf{s}$, $\mathcal{Y}$, and $\mathbf{u}$, and we write $\mathcal{D} = \mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{u})$.

Figures 1(c) and 1(d) illustrate two ILDMs. They share the same lattice $\mathcal{L}$, span vector $\mathbf{s} = (5, 5, 5)$, and overall level set $\mathcal{Y} = \{0, 0.24, 0.48, 0.72, 1\} \times \{0, 0.25, 0.5, 0.75, 1\}^2$. Both have separation distance $\rho(\mathcal{D}) = 0.3466$, which is slightly smaller than the separation distance 0.3535 of the design in Figure 1(a). Both designs qualify as ILMDMs for input spaces such as $\mathcal{G} = \{0, 0.24, 0.48, 0.72, 1\} \times \{0, 0.25, 0.5, 0.75, 1\}^2$ and $\mathcal{G} = \{0, 0.24, 0.48, 0.72, 1\} \times [0, 1]^2$. The difference in run size arises from the choice of $\mathbf{u}$. We use $m(\mathcal{S})$ to denote the number of elements in $\mathcal{S}$ when $\mathcal{S}$ is finite and set $m(\mathcal{S}) = +\infty$ when $\mathcal{S}$ is infinite. The design in Figure 1(c) has $\mathbf{u} = (0, 0, 0)$ and 63 runs, whereas the design in Figure 1(d) has $\mathbf{u} = (0, 0, 1)$ and 62 runs. By the definition of an ILDM, $m(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{u})) = m((\mathcal{L} \oplus \mathbf{u}) \cap \prod_{k=1}^p [0, s_k - 1])$, so the run size depends only on $\mathcal{L}$, $\mathbf{s}$, and $\mathbf{u}$ and is independent of $\mathcal{Y}$. Theoretical results for the run size of ILDCs therefore apply directly to ILDMs. Theorems 1 through 3 summarize several key results. In particular, Theorem 1 shows that ILDMs with $\mathbf{u} = \mathbf{0}_p$ attain the largest possible run size among all ILDMs constructed from the same $\mathcal{L}$ and $\mathbf{s}$.

Theorem 1. *Suppose $\mathcal{L}$ is an IL, $\mathcal{Y} \subset \mathcal{G}$, $\mathbf{s} \in \mathbb{N}^p$, $\mathbf{u} \in \mathbb{Z}^p$, and $\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{u})$ is an ILDM in (3.2). Then $m(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{u})) \leq m(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{0}_p))$.*

For a standard IL $\mathcal{L}$, let $\mathbf{e}_k$ denote the k th standard basis vector in $\mathbb{R}^p$, and let $q(\mathcal{L}) = \log_2 m(\mathcal{H})$ and $r(\mathcal{L}) = m(\{k : \mathbf{e}_k \in \mathcal{L}\})$. Thus, $q(\mathcal{L})$ is the base-2 logarithm of the number of lattice points in $\mathcal{H}$, whereas $r(\mathcal{L})$ is the number of standard basis vectors contained in

3.1 General expression

$\mathcal{L}$. When $\mathcal{L}$ is clear from the context, we write $q = q(\mathcal{L})$ and $r = r(\mathcal{L})$ for simplicity. These integers satisfy either $r = q = p$ or $0 \leq r < q < p$ (He, 2017). The next two results provide useful screening rules for a target run size n . For fixed $\mathcal{L}$ and $\mathbf{s}$, Theorem 2 provides lower and upper bounds on the run size. During the search in the subsequent algorithms for eligible pairs $(\mathcal{L}, \mathbf{s})$ that produce designs with at least n points, any pair $(\mathcal{L}, \mathbf{s})$ whose upper bound is below n can be excluded. For fixed q , $\mathbf{s}$, and $\mathbf{u}$, Theorem 3 shows that the maximum attainable run size over the corresponding lattice class is nondecreasing in r . Consequently, if this maximum is below n for a given r , all smaller values of r can also be excluded.

Theorem 2. Suppose that $\mathcal{L}$ is an IL, $\mathcal{Y} \subseteq \mathcal{G}$, $\mathbf{s} \in \mathbb{N}^p$, $\mathbf{u} \in \mathbb{Z}^p$, and $\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{u})$ is the ILDM defined in (3.2). Then

$$2^{q-p} \prod_{k: \mathbf{e}_k \notin \mathcal{L}} 2^{\left\lfloor \frac{s_k}{2} \right\rfloor} \prod_{k: \mathbf{e}_k \in \mathcal{L}} s_k \leq m(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{u})) \leq 2^{q-p} \prod_{k: \mathbf{e}_k \notin \mathcal{L}} 2^{\left\lceil \frac{s_k}{2} \right\rceil} \prod_{k: \mathbf{e}_k \in \mathcal{L}} s_k.$$

Theorem 3. Suppose that $\mathbf{s} \in \mathbb{N}^p$ and $\mathcal{Y} = \prod_{k=1}^p \mathcal{Y}_k \subseteq \mathcal{G}$, with $m(\mathcal{Y}_k) = s_k$ for every k .

Let z_1, z_2, z_3 be integers satisfying $0 \leq z_1 \leq z_2 < z_3 < p$. Then

$$\max_{q(\mathcal{L})=z_3, r(\mathcal{L})=z_2} m(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{0}_p)) \geq \max_{q(\mathcal{L})=z_3, r(\mathcal{L})=z_1} m(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{0}_p)),$$

where both maxima are taken over all lattices $\mathcal{L}$ satisfying $\mathbb{E}^p \subseteq \mathcal{L} \subseteq \mathbb{Z}^p$.

Unlike run-size properties, the separation distance of an ILDM is not inherited directly from that of an ILDC. We next establish several basic properties of the separation distance for ILDMs. In particular, Theorem 4 shows that the separation distance $\rho(\mathcal{D})$ is invariant to the choice of the shift vector $\mathbf{u}$.

3.1 General expression

Theorem 4. *Suppose that $\mathcal{L}$ is a standard IL, $\mathcal{Y} \subseteq \mathcal{G}$, $\mathbf{s} \in \mathbb{N}^p$, $\mathbf{u} \in \mathbb{Z}^p$, and $\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{u})$ is the ILDM defined in (3.2). Then $\rho(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{u})) = \rho(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{0}_p))$.*

Combining Theorems 1 and 4, we see that $\mathbf{u} = \mathbf{0}_p$ maximizes the run size over the admissible shifts while attaining the same separation distance as any other shift for fixed $(\mathcal{L}, \mathbf{s}, \mathcal{Y})$. We therefore optimize the lattice and level configuration at $\mathbf{u} = \mathbf{0}_p$ and subsequently allow $\mathbf{u}$ to vary to obtain a run size closer to n . This separates the optimization of the separation distance from the subsequent run-size adjustment.

We next present several results concerning $\mathcal{Y}$ and $\mathbf{s}$. Let γ be a finite set containing at least two elements, and let $\gamma_{(i)}$ denote its i th smallest element. Let

$$d^*(\gamma) = \min_{i=1, \dots, m(\gamma)-1} \{\gamma_{(i+1)} - \gamma_{(i)}\}$$

be the minimum distance between adjacent levels of γ , and

$$d^+(\gamma) = \min_{i=1, \dots, m(\gamma)-2} \{\gamma_{(i+2)} - \gamma_{(i)}\}$$

be the minimum distance between nonadjacent levels of γ . For $m(\gamma) = 2$, we use the convention $d^+(\gamma) = +\infty$.

Theorem 5. *For any fixed $\mathcal{G} \subseteq \prod_{k=1}^p [0, w_k]$ and $n \geq 2$, among all ILDMs $\mathcal{D}$ in (3.2) satisfying $m(\mathcal{D}) \geq n$ and $\mathbf{u} = \mathbf{0}_p$, the maximum separation distance is attained by at least one design satisfying $s_k \geq 2$, $\mathcal{Y}_{k(1)} = 0$, and $\mathcal{Y}_{k(s_k)} = w_k$ for all $k = 1, \dots, p$. Furthermore, for such designs,*

$$\rho(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{0}_p)) = \min \left[\min_{\mathbf{x} \in \mathcal{H} \setminus \{\mathbf{0}_p\}} \left\{ \sum_{k=1}^p d^*(\mathcal{Y}_k)^2 \right\}^{1/2}, \min_{s_k \geq 2} d^+(\mathcal{Y}_k) \right]. \quad (3.3)$$

3.2 Level selection strategy

Theorem 5 reduces the search space by showing that the endpoint and span conditions above can be imposed without loss of optimality within the ILDM class. Furthermore, $\rho(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{0}_p))$ depends on $\mathcal{Y}$ only through $d^*(\mathcal{Y}_k)$ and $d^+(\mathcal{Y}_k)$ for $k = 1, \dots, p$.

3.2 Level selection strategy

Although the preceding results restrict attention to ILDMs of the form in (3.2), where $\mathcal{L}$ is a standard IL, $\mathbf{s} \in \{2, 3, \dots\}^p$, $\prod_{k=1}^p \{0, w_k\} \subseteq \mathcal{Y} \subseteq \mathcal{G}$, and $\mathbf{u} = \mathbf{0}_p$, selecting $(\mathcal{L}, \mathbf{s}, \mathcal{Y})$ remains challenging because of the large number of possible configurations. In this subsection, we further reduce the search space by focusing on the selection of $\mathcal{Y}$ for a given pair $(\mathcal{L}, \mathbf{s})$. Because the run size is independent of $\mathcal{Y}$, the objective is to select $\mathcal{Y}$ so as to maximize the separation distance.

We first determine, independently of $\mathcal{L}$, a minimal sufficient collection of candidate level sets for $\mathcal{Y}_k$, given fixed $\mathcal{G}_k$ and s_k . By Theorem 5, $\mathcal{Y}_k$ affects the separation distance only through $d^*(\mathcal{Y}_k)$ and $d^+(\mathcal{Y}_k)$. If a level set $\tilde{\mathcal{Y}}_k$ simultaneously maximizes both quantities among all admissible $\mathcal{Y}_k \subseteq \mathcal{G}_k$ with $m(\mathcal{Y}_k) = s_k$, then it is the only candidate that needs to be retained. When the two maxima cannot be attained simultaneously, let $\mathcal{F}(\mathcal{G}_k, s_k)$ denote a minimal sufficient collection of candidate level sets. For every admissible $\gamma \subseteq \mathcal{G}_k$ with $m(\gamma) = s_k$, there exists $\tilde{\gamma} \in \mathcal{F}(\mathcal{G}_k, s_k)$ such that $d^*(\tilde{\gamma}) \geq d^*(\gamma)$ and $d^+(\tilde{\gamma}) \geq d^+(\gamma)$. Moreover, each $\tilde{\gamma} \in \mathcal{F}(\mathcal{G}_k, s_k)$ is undominated in the sense that no other admissible γ satisfies $d^*(\gamma) \geq d^*(\tilde{\gamma})$ and $d^+(\gamma) \geq d^+(\tilde{\gamma})$, with at least one strict inequality.

We construct $\mathcal{F}(\mathcal{G}_k, s_k)$ by considering three cases. In the first case, $\mathcal{G}_k = [0, w_k]$ and s_k is odd. Theorem 6 shows that $\mathcal{F}(\mathcal{G}_k, s_k)$ contains only the equally spaced level set.

3.2 Level selection strategy

Algorithm 1 Construct $\mathcal{F}(\mathcal{G}_k, s_k)$ for a finite level set $\mathcal{G}_k$.

Input: $\mathcal{G}_k$ and s_k .

- 1: If $s_k = 2$, **return** $\{\{0, \max(\mathcal{G}_k)\}\}$.
 - 2: Initialize $\tilde{d}^+ \leftarrow 0$ and $\mathcal{F} \leftarrow \emptyset$.
 - 3: **loop**
 - 4: Use Algorithm 2 to obtain $\tilde{d}^*$, the largest attainable value of $d^*(\gamma)$ over all $\gamma \subseteq \mathcal{G}_k$ satisfying $m(\gamma) = s_k$ and $d^+(\gamma) > \tilde{d}^+$. If no such γ exists, set $\tilde{d}^* = 0$.
 - 5: If $\tilde{d}^* = 0$, **return** $\mathcal{F}$.
 - 6: Use Algorithm 3 to obtain a set $\tilde{\gamma}$ that maximizes $d^+(\gamma)$ over all $\gamma \subseteq \mathcal{G}_k$ satisfying $m(\gamma) = s_k$ and $d^*(\gamma) \geq \tilde{d}^*$.
 - 7: Set $\mathcal{F} \leftarrow \mathcal{F} \cup \{\tilde{\gamma}\}$ and $\tilde{d}^+ \leftarrow d^+(\tilde{\gamma})$.
 - 8: **end loop**
-

Theorem 6. Suppose that $\mathcal{G}_k = [0, w_k]$ and $s_k \geq 2$ is odd. Let $\bar{\gamma}_i = w_k(i-1)/(s_k-1)$ for $i = 1, \dots, s_k$, and let $\bar{\gamma} = \{\bar{\gamma}_1, \dots, \bar{\gamma}_{s_k}\}$. Then $\mathcal{F}([0, w_k], s_k) = \{\bar{\gamma}\}$ is a minimal sufficient collection.

In the second case, $\mathcal{G}_k = [0, w_k]$ and s_k is even. The maxima of d^* and d^+ cannot be attained simultaneously by a single level set. For $0 < v_1 \leq w_k/(s_k-1)$, let $v_2 = (2w_k - s_kv_1)/(s_k-2)$ and $\tilde{\gamma}_{v_1, i} = \lfloor i/2 \rfloor v_1 + (\lceil i/2 \rceil - 1)v_2$, $i = 1, \dots, s_k$. Let $\tilde{\gamma}_{v_1} = \{\tilde{\gamma}_{v_1, 1}, \dots, \tilde{\gamma}_{v_1, s_k}\}$. Theorem 7 characterizes the corresponding minimal sufficient collection. Within this collection, d^* is maximized at $v_1 = w_k/(s_k-1)$, yielding equally spaced levels with $d^*(\tilde{\gamma}_{w_k/(s_k-1)}) = w_k/(s_k-1)$ and $d^+(\tilde{\gamma}_{w_k/(s_k-1)}) = 2w_k/(s_k-1)$. At the other endpoint, as $v_1 \rightarrow 0^+$, $d^*(\tilde{\gamma}_{v_1}) \rightarrow 0$, while $d^+(\tilde{\gamma}_{v_1}) \rightarrow 2w_k/(s_k-2)$. Thus, this family captures the tradeoff between d^* and d^+ .

Theorem 7. Suppose that $\mathcal{G}_k = [0, w_k]$ and $s_k > 2$ is even. Then $\mathcal{F}([0, w_k], s_k) = \{\tilde{\gamma}_{v_1} : 0 < v_1 \leq w_k/(s_k-1)\}$ is a minimal sufficient collection.

3.2 Level selection strategy

Algorithm 2 Find the largest attainable value of $d^*(\gamma)$ subject to $d^+(\gamma) > \tilde{d}^+$.

Input: $\mathcal{G}_k$, $s_k \geq 3$, and $\tilde{d}^+$.

```

1: Initialize  $\tilde{d}^* \leftarrow 0$ .
2: loop
3:   Set  $\gamma_1 \leftarrow 0$ . If  $\{l \in \mathcal{G}_k : l > \tilde{d}^*\} = \emptyset$ , return  $\tilde{d}^*$ ; otherwise, set  $\gamma_2 \leftarrow \min\{l \in \mathcal{G}_k : l > \tilde{d}^*\}$ .
4:   for  $i = 3$  to  $s_k - 1$  do
5:     If  $\{l \in \mathcal{G}_k : l > \gamma_{i-1} + \tilde{d}^*, l > \gamma_{i-2} + \tilde{d}^+\} = \emptyset$ , return  $\tilde{d}^*$ .
6:     Set  $\gamma_i \leftarrow \min\{l \in \mathcal{G}_k : l > \gamma_{i-1} + \tilde{d}^*, l > \gamma_{i-2} + \tilde{d}^+\}$ .
7:   end for
8:   If  $\max(\mathcal{G}_k) \leq \gamma_{s_k-1} + \tilde{d}^*$  or  $\max(\mathcal{G}_k) \leq \gamma_{s_k-2} + \tilde{d}^+$ , return  $\tilde{d}^*$ .
9:   Set  $\gamma_{s_k} \leftarrow \max(\mathcal{G}_k)$ ,  $\gamma \leftarrow \{\gamma_1, \dots, \gamma_{s_k}\}$ , and  $\tilde{d}^* \leftarrow d^*(\gamma)$ .
10: end loop

```

In the third case, $\mathcal{G}_k \subset [0, w_k]$ is a finite set. A simple characterization of $\mathcal{F}(\mathcal{G}_k, s_k)$ is generally unavailable, and exhaustive enumeration becomes impractical because the number of possible level sets grows combinatorially with $m(\mathcal{G}_k)$. We therefore construct $\mathcal{F}(\mathcal{G}_k, s_k)$ using Algorithm 1. The algorithm avoids exhaustive enumeration by sequentially identifying level sets that are nondominated with respect to d^* and d^+ . Starting from $\tilde{d}^+ = 0$, Algorithm 2 determines the largest attainable value of $d^*(\gamma)$ among level sets satisfying $d^+(\gamma) > \tilde{d}^+$. Algorithm 3 then identifies, subject to the resulting value of d^* , a level set that maximizes $d^+(\gamma)$. This level set is added to $\mathcal{F}(\mathcal{G}_k, s_k)$, and the d^+ threshold is updated for the next iteration. We write each candidate level set as $\gamma = \{\gamma_1 < \dots < \gamma_{s_k}\}$. For illustration, Table 2 reports the collections obtained for $\mathcal{G}_k = \{0, 0.12, 0.24, 0.36, 0.48, 0.60, 0.72, 0.84, 1\}$ and $s_k \in \{2, \dots, 9\}$. Theorem 8 establishes that Algorithm 1 produces a minimal sufficient collection.

Theorem 8. Suppose that $\mathcal{G}_k \subseteq [0, w_k]$ is finite, $2 \leq s_k \leq m(\mathcal{G}_k)$, and $\mathcal{F}(\mathcal{G}_k, s_k)$ is obtained

3.2 Level selection strategy

Algorithm 3 Find a level set γ that maximizes $d^+(\gamma)$ subject to $d^*(\gamma) \geq \tilde{d}^*$.

Input: $\mathcal{G}_k$, $s_k \geq 3$, and $\tilde{d}^*$.

- 1: Initialize $\tilde{d}^+ \leftarrow 0$ and $\tilde{\gamma} \leftarrow \emptyset$.
 - 2: **loop**
 - 3: Set $\gamma_1 \leftarrow 0$ and $\gamma_2 \leftarrow \min\{l \in \mathcal{G}_k : l \geq \tilde{d}^*\}$.
 - 4: **for** $i = 3$ to $s_k - 1$ **do**
 - 5: If no $l \in \mathcal{G}_k$ satisfies $l \geq \gamma_{i-1} + \tilde{d}^*$ and $l > \gamma_{i-2} + \tilde{d}^+$, **return** $\tilde{\gamma}$.
 - 6: Set $\gamma_i \leftarrow \min\{l \in \mathcal{G}_k : l \geq \gamma_{i-1} + \tilde{d}^*, l > \gamma_{i-2} + \tilde{d}^+\}$.
 - 7: **end for**
 - 8: If $\max(\mathcal{G}_k) < \gamma_{s_k-1} + \tilde{d}^*$ or $\max(\mathcal{G}_k) \leq \gamma_{s_k-2} + \tilde{d}^+$, **return** $\tilde{\gamma}$; otherwise, set $\gamma_{s_k} \leftarrow \max(\mathcal{G}_k)$.
 - 9: Set $\gamma \leftarrow \{\gamma_1, \dots, \gamma_{s_k}\}$, $\tilde{\gamma} \leftarrow \gamma$, and $\tilde{d}^+ \leftarrow d^+(\gamma)$.
 - 10: **end loop**
-

using Algorithm 1. Then $\mathcal{F}(\mathcal{G}_k, s_k)$ is a minimal sufficient collection.

Among the candidate level sets retained in $\mathcal{F}(\mathcal{G}_k, s_k)$, we seek a configuration $\mathcal{Y}_s = \prod_{k=1}^p \mathcal{Y}_k$, $\mathcal{Y}_k \in \mathcal{F}(\mathcal{G}_k, s_k)$, that maximizes the separation distance for given $\mathcal{L}$ and $\mathbf{s}$. As illustrated in Example 2, most designs that are optimal within the examined ILDM class are associated with relatively large values of d^* . This observation motivates the strategy in Algorithm 4. The search begins with a level set that maximizes d^* in each coordinate, as specified in Line 2, and allows d^* to decrease only when the corresponding increase in d^+ can improve the separation distance. Let $\rho^{(t)}$ denote the separation distance at iteration t . If $\rho^{(t)} = \min_{\mathbf{x} \in \mathcal{H} \setminus \{\mathbf{0}_p\}} \{\sum_{k: x_k=1} d^*(\mathcal{Y}_k^{(t)})^2\}^{1/2}$, then the first term in (3.3) determines the separation distance, and thus any increase in d^+ at the expense of d^* cannot improve the design. The search therefore terminates at Line 6. Otherwise, coordinates with $d^+(\mathcal{Y}_k^{(t)}) > \rho^{(t)}$ are retained at Line 8, whereas those attaining the limiting value $\rho^{(t)}$ are considered for further adjustment. For finite $\mathcal{G}_k$, Line 9 selects, among candidates with a larger d^+ , the

3.2 Level selection strategy

Table 2: Minimal sufficient collections $\mathcal{F}(\mathcal{G}_k, s_k)$ for $\mathcal{G}_k = \{0, 0.12, 0.24, 0.36, 0.48, 0.60, 0.72, 0.84, 1\}$.

s_k	$\mathcal{F}(\mathcal{G}_k, s_k)$	d^*	d^+
2	$\{0, 1\}$	1	$+\infty$
3	$\{0, 0.48, 1\}$	0.48	1
4	$\{0, 0.36, 0.72, 1\}$	0.28	0.64
	$\{0, 0.24, 0.72, 1\}$	0.24	0.72
	$\{0, 0.24, 0.84, 1\}$	0.16	0.76
	$\{0, 0.12, 0.84, 1\}$	0.12	0.84
5	$\{0, 0.24, 0.48, 0.72, 1\}$	0.24	0.48
6	$\{0, 0.12, 0.36, 0.48, 0.72, 1\}$	0.12	0.36
7	$\{0, 0.12, 0.36, 0.48, 0.72, 0.84, 1\}$	0.12	0.28
8	$\{0, 0.12, 0.36, 0.48, 0.60, 0.72, 0.84, 1\}$	0.12	0.24
9	$\{0, 0.12, 0.24, 0.36, 0.48, 0.60, 0.72, 0.84, 1\}$	0.12	0.24

level set with the largest possible d^* . For continuous coordinates with even s_k , the limiting coordinates are adjusted jointly according to the family characterized in Theorem 7. Their common value of d^+ is increased subject to the constraints implied by (3.3), balancing the increase in d^+ against the accompanying reduction in d^* . If the resulting update does not increase the separation distance, the best level configuration obtained so far is returned, as specified in Line 5.

Algorithm 4 Select the level configuration $\mathcal{Y}$ to maximize $\rho(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{0}_p))$ for fixed $\mathcal{L}$ and $\mathbf{s}$.

Input: $\mathcal{G} = \prod_{k=1}^p \mathcal{G}_k$, $\mathcal{L}$, and $\mathbf{s} = (s_1, \dots, s_p)$.

- 1: For each k , obtain $\mathcal{F}(\mathcal{G}_k, s_k)$ using Algorithm 1 when $\mathcal{G}_k$ is finite and using Theorems 6 and 7 when $\mathcal{G}_k = [0, w_k]$, with $\mathcal{F}([0, w_k], 2) = \{\{0, w_k\}\}$.
 - 2: Set $\rho^{(0)} \leftarrow 0$ and $t \leftarrow 1$. For each k , choose $\mathcal{Y}_k^{(1)} \in \mathcal{F}(\mathcal{G}_k, s_k)$ to maximize $d^*(\mathcal{Y}_k^{(1)})$.
 - 3: **loop**
 - 4: Set $\mathcal{Y}^{(t)} = \prod_{k=1}^p \mathcal{Y}_k^{(t)}$ and $\rho^{(t)} \leftarrow \rho(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}^{(t)}, \mathbf{0}_p))$.
 - 5: If $\rho^{(t)} > \rho^{(t-1)}$, set $\tilde{\rho} \leftarrow \rho^{(t)}$ and $\tilde{\mathcal{Y}} \leftarrow \mathcal{Y}^{(t)}$; otherwise, **return** $\tilde{\rho}$ and $\tilde{\mathcal{Y}}$.
 - 6: If $\rho^{(t)} = \min_{\mathbf{x} \in \mathcal{H} \setminus \{\mathbf{0}_p\}} \{\sum_{k: x_k=1} d^*(\mathcal{Y}_k^{(t)})^2\}^{1/2}$, **return** $\tilde{\rho}$ and $\tilde{\mathcal{Y}}$.
 - 7: If there exists k such that $d^+(\mathcal{Y}_k^{(t)}) = \rho^{(t)}$ and no $\gamma \in \mathcal{F}(\mathcal{G}_k, s_k)$ satisfies $d^+(\gamma) > d^+(\mathcal{Y}_k^{(t)})$, **return** $\tilde{\rho}$ and $\tilde{\mathcal{Y}}$.
 - 8: Set $\mathcal{Y}_k^{(t+1)} \leftarrow \mathcal{Y}_k^{(t)}$ for all k such that $d^+(\mathcal{Y}_k^{(t)}) > \rho^{(t)}$.
 - 9: For each k with finite $\mathcal{G}_k$ and $d^+(\mathcal{Y}_k^{(t)}) = \rho^{(t)}$, choose $\mathcal{Y}_k^{(t+1)} \in \mathcal{F}(\mathcal{G}_k, s_k)$ to maximize $d^*(\mathcal{Y}_k^{(t+1)})$ subject to $d^+(\mathcal{Y}_k^{(t+1)}) > d^+(\mathcal{Y}_k^{(t)})$.
 - 10: Set $\mathcal{A}_c^{(t)} = \{k : \mathcal{G}_k = [0, w_k], s_k > 2 \text{ is even, } d^+(\mathcal{Y}_k^{(t)}) = \rho^{(t)}\}$, and set $\mathcal{Y}_k^{(t+1)} \leftarrow \mathcal{Y}_k^{(t)}$ for all $k \in \mathcal{A}_c^{(t)}$.
 - 11: Set $\rho_{\text{pre}} \leftarrow \rho(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}^{(t+1)}, \mathbf{0}_p))$. If $\rho_{\text{pre}} < \rho^{(t)}$, **return** $\tilde{\rho}$ and $\tilde{\mathcal{Y}}$.
 - 12: **if** $\mathcal{A}_c^{(t)} \neq \emptyset$ **then**
 - 13: Choose the largest feasible $\hat{d}_c^+$ satisfying $\hat{d}_c^+ \geq \rho_{\text{pre}}$, $\hat{d}_c^+ < \min_{k \in \mathcal{A}_c^{(t)}} 2w_k/(s_k - 2)$, and, if $\mathcal{A}_c^{(t)} \neq \{1, \dots, p\}$, $\hat{d}_c^+ \leq \min_{k \notin \mathcal{A}_c^{(t)}} d^+(\mathcal{Y}_k^{(t+1)})$. For every $\mathbf{x} \in \mathcal{H} \setminus \{\mathbf{0}_p\}$, also require $(\hat{d}_c^+)^2 \leq \sum_{k \in \mathcal{A}_c^{(t)}} x_k \{w_k - \hat{d}_c^+(s_k - 2)/2\}^2 + \sum_{k \notin \mathcal{A}_c^{(t)}} x_k d^*(\mathcal{Y}_k^{(t+1)})^2$.
 - 14: For each $k \in \mathcal{A}_c^{(t)}$, choose $\mathcal{Y}_k^{(t+1)} \in \mathcal{F}(\mathcal{G}_k, s_k)$ such that $d^+(\mathcal{Y}_k^{(t+1)}) = \hat{d}_c^+$.
 - 15: **end if**
 - 16: Set $t \leftarrow t + 1$.
 - 17: **end loop**
-

3.3 Design construction

For given p , n , and $\mathcal{G}$, we construct ILMDMs by first searching over designs $\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{0}_p)$ of the form in (3.2) with $s_k \geq 2$ for every k and $m(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{0}_p)) \geq n$, as justified by Theorems 1 through 5. Algorithm 5 then selects the shift yielding the smallest run size of at least n for each retained candidate and returns the design with the smallest resulting run

3.3 Design construction

Algorithm 5 Select the candidate design with the smallest run size that is at least n .

Input: p , n , t , and $\mathcal{L}^{(i)}$, $\mathbf{s}^{(i)}$, and $\mathcal{Y}^{(i)}$ for $i = 1, \dots, t$.

- 1: **for** $i = 1$ to t **do**
 - 2: Choose $\mathbf{u}^{(i)} \in \arg \min_{\mathbf{u} \in \{0,1\}^p: m(\mathcal{D}(\mathcal{L}^{(i)}, \mathbf{s}^{(i)}, \mathcal{Y}^{(i)}, \mathbf{u})) \geq n} m(\mathcal{D}(\mathcal{L}^{(i)}, \mathbf{s}^{(i)}, \mathcal{Y}^{(i)}, \mathbf{u}))$, and set $m^{(i)} \leftarrow m(\mathcal{D}(\mathcal{L}^{(i)}, \mathbf{s}^{(i)}, \mathcal{Y}^{(i)}, \mathbf{u}^{(i)}))$.
 - 3: **end for**
 - 4: Choose $\hat{i} \in \arg \min_{i=1, \dots, t} m^{(i)}$.
 - 5: **return** $\mathcal{L}^{(\hat{i})}$, $\mathbf{s}^{(\hat{i})}$, $\mathcal{Y}^{(\hat{i})}$, $\mathbf{u}^{(\hat{i})}$, and $m^{(\hat{i})}$.
-

size. Algorithms 6, 9, and 10, presented in the following subsections, address $2 \leq p \leq 5$, $6 \leq p \leq 7$, and $p \geq 8$, respectively.

3.3.1 Construction for $2 \leq p \leq 5$

For $2 \leq p \leq 5$, we enumerate all p -dimensional standard ILs and search over the admissible span vectors to construct an ILMDM. The procedure is described in Algorithm 6.

For each candidate lattice $\mathcal{L}_i$, the span vectors are examined in increasing lexicographic order, and $\hat{\rho}$ records the largest separation distance attained among the designs examined so far. We use $\succ_{\text{lex}}$ and $\min_{\text{lex}}$ to denote the strict lexicographic order and the corresponding minimum, respectively. By Theorem 5, $\rho(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{0}_p)) \leq d^+(\mathcal{Y}_k)$ for every k . Hence, a value of s_k can be discarded whenever no $\gamma \in \mathcal{F}(\mathcal{G}_k, s_k)$ satisfies $d^+(\gamma) \geq \hat{\rho}$. This screening step is implemented in Line 10 for finite input spaces and in Line 11 for continuous input spaces. In addition, once a span vector $\mathbf{s}$ satisfies the run-size requirement, larger values of s_p are skipped because adding more levels within the same coordinate range cannot increase the attainable separation distance characterized in Theorem 5. This pruning step is implemented in Line 15.

We next present three examples illustrating different aspects of Algorithm 6.

3.3 Design construction

Algorithm 6 Construct an ILMDM for $2 \leq p \leq 5$.

Input: $p, n, \mathcal{G}, \mathcal{F}$, and the list of all p -dimensional standard ILs $\{\mathcal{L}_i\}_{i=1}^N$.

- 1: Initialize $\hat{\rho} \leftarrow 0$.
 - 2: For each k , set $\mathcal{S}_k \leftarrow \{2, \dots, m(\mathcal{G}_k)\}$ if $\mathcal{G}_k$ is finite and $\mathcal{S}_k \leftarrow \{2, 3, \dots\}$ if $\mathcal{G}_k = [0, w_k]$.
 - 3: **for** $i = 1$ to N **do**
 - 4: Set $\tilde{\mathbf{s}} \leftarrow \mathbf{0}_p$.
 - 5: **while** $\{\mathbf{z} \in \prod_{k=1}^p \mathcal{S}_k : m(\mathcal{L}_i \cap \prod_{k=1}^p [0, z_k - 1]) \geq n, \mathbf{z} \succ_{\text{lex}} \tilde{\mathbf{s}}\} \neq \emptyset$ **do**
 - 6: Set $\mathbf{s} \leftarrow \min_{\text{lex}} \{\mathbf{z} \in \prod_{k=1}^p \mathcal{S}_k : m(\mathcal{L}_i \cap \prod_{k=1}^p [0, z_k - 1]) \geq n, \mathbf{z} \succ_{\text{lex}} \tilde{\mathbf{s}}\}$.
 - 7: Run Algorithm 4 to obtain $\tilde{\rho}$ and $\tilde{\mathcal{Y}}$, where $\tilde{\rho} = \rho(\mathcal{D}(\mathcal{L}_i, \mathbf{s}, \tilde{\mathcal{Y}}, \mathbf{0}_p))$.
 - 8: **if** $\tilde{\rho} > \hat{\rho}$ **then**
 - 9: Set $\hat{\rho} \leftarrow \tilde{\rho}$, $t \leftarrow 1$, $\hat{\mathcal{L}}^{(1)} \leftarrow \mathcal{L}_i$, $\hat{\mathbf{s}}^{(1)} \leftarrow \mathbf{s}$, and $\hat{\mathcal{Y}}^{(1)} \leftarrow \tilde{\mathcal{Y}}$.
 - 10: For each k with finite $\mathcal{G}_k$, set $\mathcal{S}_k \leftarrow \{z \in \mathcal{S}_k : \max_{\gamma \in \mathcal{F}(\mathcal{G}_k, z)} d^+(\gamma) \geq \hat{\rho}\}$.
 - 11: For each k with $\mathcal{G}_k = [0, w_k]$, retain $z \in \mathcal{S}_k$ if $z = 2$, if $2w_k/(z - 1) \geq \hat{\rho}$ for odd z , or if $2w_k/(z - 2) > \hat{\rho}$ for even $z > 2$.
 - 12: **else if** $\tilde{\rho} = \hat{\rho}$ **then**
 - 13: Set $t \leftarrow t + 1$, $\hat{\mathcal{L}}^{(t)} \leftarrow \mathcal{L}_i$, $\hat{\mathbf{s}}^{(t)} \leftarrow \mathbf{s}$, and $\hat{\mathcal{Y}}^{(t)} \leftarrow \tilde{\mathcal{Y}}$.
 - 14: **end if**
 - 15: Set $\tilde{\mathbf{s}} \leftarrow \mathbf{s}$ and $\tilde{s}_p \leftarrow +\infty$.
 - 16: **end while**
 - 17: **end for**
 - 18: Apply Algorithm 5 to $\{(\hat{\mathcal{L}}^{(j)}, \hat{\mathbf{s}}^{(j)}, \hat{\mathcal{Y}}^{(j)}) : j = 1, \dots, t\}$ to obtain $\hat{\mathcal{L}}, \hat{\mathbf{s}}, \hat{\mathcal{Y}}, \hat{\mathbf{u}}$, and $\hat{m}$.
 - 19: **return** $\mathcal{D}(\hat{\mathcal{L}}, \hat{\mathbf{s}}, \hat{\mathcal{Y}}, \hat{\mathbf{u}})$, $\hat{\rho}$, and $\hat{m}$.
-

Example 1. Suppose $p = 3$, $n = 62$, and $\mathcal{G} = \{0, 0.12, 0.24, 0.36, 0.48, 0.60, 0.72, 0.84, 1\} \times [0, 1]^2$. The collections $\mathcal{F}(\mathcal{G}_1, s)$ for $s \in \{2, \dots, 9\}$ are reported in Table 2. At Line 2, we initialize $\hat{\rho} = 0$, $\mathcal{S}_1 = \{2, \dots, 9\}$, and $\mathcal{S}_2 = \mathcal{S}_3 = \{2, 3, \dots\}$. For the first three-dimensional standard IL, $\mathcal{L}(\mathbf{I}_3)$, the first admissible span vector is $\mathbf{s} = (2, 2, 16)$. Algorithm 4 yields $\mathcal{Y} = \{0, 1\}^2 \times \{0, 1/15, \dots, 1\}$ with separation distance $\rho = 0.067$, which updates $\hat{\rho}$ and the admissible span sets. Table 3 reports this candidate and all subsequent candidates that improve upon or attain the current best separation distance, together with the corresponding updates to $\mathcal{S}_1, \dots, \mathcal{S}_3$. After completing the loop at Line 17, the design with $\mathbf{u} = \mathbf{0}_p$ is shown

in Figure 1(c). At Line 18, Algorithm 5 identifies $\mathbf{u} = (0, 0, 1)$, which gives the smallest run size that is at least n and yields the 62-run design shown in Figure 1(d).

Example 2. Applying Algorithm 6 to all settings with $2 \leq p \leq 5$, $4 \leq n \leq 1000$, and $\mathcal{G} = \{0, 0.01, 0.02, \dots, 1\} \times [0, 1]^{p-1}$ shows that most resulting designs use only level sets that attain the largest value of d^* within their respective collections $\mathcal{F}(\mathcal{G}_k, s_k)$. For $p = 2, 3, 4$, and 5, among the 997 resulting designs for each value of p , only 134, 33, 66, and 16 designs, respectively, contain at least one $\mathcal{Y}_k$ that does not attain the largest value of d^* .

Example 3. Suppose that $p = 3$, $n = 10$, and $\mathcal{G} = [0, 1]^3$. The ILMDC constructed by He (2019) is $\mathcal{D}(\mathcal{L}(\mathbf{G}_{3,2}), (2, 3, 3), \{0, 1\} \times \{0, 0.5, 1\}^2, (0, 0, 0))$, which has 10 runs and attains $\rho = 0.7071$. In contrast, Algorithm 6 yields the ILMDM $\mathcal{D}(\mathcal{L}(\mathbf{G}_{3,3}), (4, 3, 3), \{0, 0.25, 0.75, 1\} \times \{0, 0.5, 1\}^2, (0, 0, 0))$, which has the same run size but attains a larger separation distance of $\rho = 0.75$.

Example 3 illustrates the additional flexibility of ILMDMs in choosing coordinate levels, whereas ILMDCs use fixed, equally spaced levels. Algorithm 6 exhaustively examines all standard ILs together with the span vectors retained by the exact screening rules. Because the number of candidate ILs grows rapidly with p , we restrict the use of this algorithm to $2 \leq p \leq 5$.

3.3.2 Construction for $6 \leq p \leq 7$

We next consider higher-dimensional settings with $6 \leq p \leq 7$. Specifically, we use two strategies to accelerate the search. First, during the execution of Algorithm 6, a larger value of the current best separation distance $\hat{\rho}$ leads to more aggressive pruning of candidate

3.3 Design construction

Table 3: Updates to $\hat{\rho}$, $\mathcal{L}$, $\mathbf{s}$, $\mathcal{Y}$, and $\prod_{k=1}^3 \mathcal{S}_k$ in Algorithm 6 for Example 1.

$\hat{\rho}$	$\mathcal{L}$	$\mathbf{s}$	$\mathcal{Y}$	$\prod_{k=1}^3 \mathcal{S}_k$
0.067	$\mathcal{L}(\mathbf{G}_{3,1})$	(2, 2, 16)	$\{0, 1\}^2 \times \Delta_{1/15}$	$\llbracket 9 \rrbracket \times \llbracket 31 \rrbracket^2$
0.100	$\mathcal{L}(\mathbf{G}_{3,1})$	(2, 3, 11)	$\{0, 1\} \times \Delta_{1/2} \times \Delta_{1/10}$	$\llbracket 9 \rrbracket \times \llbracket 21 \rrbracket^2$
0.143	$\mathcal{L}(\mathbf{G}_{3,1})$	(2, 4, 8)	$\{0, 1\} \times \Delta_{1/3} \times \Delta_{1/7}$	$\llbracket 9 \rrbracket \times \llbracket 15 \rrbracket^2$
0.167	$\mathcal{L}(\mathbf{G}_{3,1})$	(2, 5, 7)	$\{0, 1\} \times \Delta_{1/4} \times \Delta_{1/6}$	$\llbracket 9 \rrbracket \times \llbracket 13 \rrbracket^2$
0.200	$\mathcal{L}(\mathbf{G}_{3,1})$	(2, 6, 6)	$\{0, 1\} \times \Delta_{1/5}^2$	$\llbracket 9 \rrbracket \times \llbracket 11 \rrbracket^2$
	$\mathcal{L}(\mathbf{G}_{3,1})$	(3, 4, 6)	$\{0, 0.48, 1\} \times \Delta_{1/3} \times \Delta_{1/5}$	
0.250	$\mathcal{L}(\mathbf{G}_{3,1})$	(3, 5, 5)	$\{0, 0.48, 1\} \times \Delta_{1/4}^2$	$\llbracket 7 \rrbracket \times \llbracket 9 \rrbracket^2$
0.280	$\mathcal{L}(\mathbf{G}_{3,1})$	(4, 4, 4)	$\{0, 0.36, 0.72, 1\} \times \Delta_{1/3}^2$	$\llbracket 7 \rrbracket \times \llbracket 8 \rrbracket^2$
	$\mathcal{L}(\mathbf{G}_{3,3})$	(4, 4, 8)	$\{0, 0.36, 0.72, 1\} \times \Delta_{1/3} \times \Delta_{1/7}$	
	$\mathcal{L}(\mathbf{G}_{3,3})$	(4, 5, 7)	$\{0, 0.36, 0.72, 1\} \times \Delta_{1/4} \times \Delta_{1/6}$	
	$\mathcal{L}(\mathbf{G}_{3,3})$	(4, 6, 6)	$\{0, 0.36, 0.72, 1\} \times \Delta_{1/5}^2$	
	$\mathcal{L}(\mathbf{G}_{3,3})$	(4, 7, 5)	$\{0, 0.36, 0.72, 1\} \times \Delta_{1/6} \times \Delta_{1/4}$	
	$\mathcal{L}(\mathbf{G}_{3,3})$	(4, 8, 4)	$\{0, 0.36, 0.72, 1\} \times \Delta_{1/7} \times \Delta_{1/3}$	
0.299	$\mathcal{L}(\mathbf{G}_{3,5})$	(4, 4, 8)	$\{0, 0.36, 0.72, 1\} \times \Delta_{1/3} \times \Delta_{1/7}$	$\llbracket 6 \rrbracket \times \llbracket 8 \rrbracket^2$
	$\mathcal{L}(\mathbf{G}_{3,6})$	(4, 8, 4)	$\{0, 0.36, 0.72, 1\} \times \Delta_{1/7} \times \Delta_{1/3}$	
	$\mathcal{L}(\mathbf{G}_{3,2})$	(4, 4, 8)	$\{0, 0.36, 0.72, 1\} \times \Delta_{1/3} \times \Delta_{1/7}$	
0.300	$\mathcal{L}(\mathbf{G}_{3,2})$	(4, 5, 7)	$\{0, 0.36, 0.72, 1\} \times \Delta_{1/4} \times \Delta_{1/6}$	$\llbracket 6 \rrbracket \times \llbracket 8 \rrbracket^2$
	$\mathcal{L}(\mathbf{G}_{3,2})$	(4, 6, 6)	$\{0, 0.36, 0.72, 1\} \times \Delta_{1/5}^2$	
0.347	$\mathcal{L}(\mathbf{G}_{3,2})$	(5, 5, 5)	$\{0, 0.24, 0.48, 0.72, 1\} \times \Delta_{1/4}^2$	$\llbracket 6 \rrbracket^3$

Here, $\Delta_x := \{0, x, 2x, \dots, 1\}$ and $\llbracket a \rrbracket := \{2, 3, \dots, a\}$. The matrices $\mathbf{G}_{3,1}$, $\mathbf{G}_{3,2}$, and $\mathbf{G}_{3,3}$ are given in (3.4), while $\mathbf{G}_{3,5}$ and $\mathbf{G}_{3,6}$ are given in the Supplementary Material.

3.3 Design construction

Algorithm 7 Construct ILMDMs for the three ILs defined in (3.4).

Input: $p, n, \mathcal{G}$.

- 1: Initialize the candidate-result set $\mathcal{C} \leftarrow \emptyset$.
 - 2: **for** $i = 1$ to 3 **do**
 - 3: Set $\mathbf{s} \leftarrow \mathbf{2}_p$.
 - 4: **while** $m(\mathcal{L}(\mathbf{G}_{p,i}) \cap \prod_{j=1}^p [0, s_j - 1]) < n$ and $s_k < m(\mathcal{G}_k)$ for at least one k **do**
 - 5: For $k = 1, \dots, p$, set $\mathcal{Y}_k$ to an element of $\mathcal{F}(\mathcal{G}_k, s_k)$ that maximizes d^* .
 - 6: Set $s_k \leftarrow s_k + 1$ for every k such that $d^*(\mathcal{Y}_k) = \max_{j=1}^p d^*(\mathcal{Y}_j)$.
 - 7: **end while**
 - 8: Run Algorithm 4 to obtain $\tilde{\mathcal{Y}}$, which maximizes $\rho[\mathcal{D}\{\mathcal{L}(\mathbf{G}_{p,i}), \mathbf{s}, \mathcal{Y}, \mathbf{0}_p\}]$.
 - 9: Add $(\rho[\mathcal{D}\{\mathcal{L}(\mathbf{G}_{p,i}), \mathbf{s}, \tilde{\mathcal{Y}}, \mathbf{0}_p\}], \mathcal{L}(\mathbf{G}_{p,i}), \mathbf{s}, \tilde{\mathcal{Y}})$ to $\mathcal{C}$.
 - 10: **end for**
 - 11: Return $\mathcal{C}$.
-

span vectors $\mathbf{s}$. Consequently, identifying a near-optimal design early in the search can substantially reduce the number of configurations that need to be examined. Algorithm 7 uses the following three explicitly defined ILs to initialize the search. Let

$$\mathbf{G}_{p,1} = \mathbf{I}_p, \quad \mathbf{G}_{p,2} = \begin{pmatrix} \mathbf{I}_{p-1} & \mathbf{1}_{(p-1) \times 1} \\ \mathbf{0}_{1 \times (p-1)} & 2 \end{pmatrix}, \quad \mathbf{G}_{p,3} = \begin{pmatrix} 1 & \mathbf{1}_{1 \times (p-1)} \\ \mathbf{0}_{(p-1) \times 1} & 2\mathbf{I}_{p-1} \end{pmatrix}. \quad (3.4)$$

The lattice $\mathcal{L}(\mathbf{G}_{p,1}) = \mathbb{Z}^p$ is the integer lattice. The lattice $\mathcal{L}(\mathbf{G}_{p,2})$ is the index-two sublattice of $\mathbb{Z}^p$ consisting of points $\mathbf{x}$ for which $\sum_{k=1}^p x_k$ is even. The third lattice satisfies $\mathcal{L}(\mathbf{G}_{p,3}) = \mathbb{E}^p \cup (\mathbb{E}^p \oplus \mathbf{1}_p)$.

As a second strategy, we restrict the search to candidate ILs with favorable separation properties. For each admissible $(q, r, \mathbf{s})$, we propose to construct one such candidate IL with a separation distance larger than the current value of $\hat{\rho}$ using Algorithm 8. The algorithm initializes $\mathcal{H}$ with $\mathbf{0}_p$ and subsequently assigns vectors in $\{0, 1\}^p$ to either $\mathcal{H}$ or its complement $\bar{\mathcal{H}} = \{0, 1\}^p \setminus \mathcal{H}$. Since $r(\mathcal{L}) = r$, the r standard basis vectors $\mathbf{e}_k$ corresponding to the largest values of $d^*(\mathcal{Y}_k)$ are assigned to $\mathcal{H}$, while the remaining $p - r$

Algorithm 8 Construct an admissible candidate IL for the given $(q, r, \mathbf{s})$ and separation-distance threshold $\hat{\rho}$.

Input: $p, \mathcal{Y}, q, r, \mathbf{s}$, and $\hat{\rho}$.

- 1: Initialize $\mathcal{H}$ as the set of all binary sums of the r standard basis vectors $\mathbf{e}_k$ corresponding to the largest values of $d^*(\mathcal{Y}_k)$, and let $\bar{\mathcal{H}}$ contain the remaining $p - r$ standard basis vectors. Set $J \leftarrow 2^{p-q}$.
 - 2: **while** $m(\mathcal{H}) < 2^q$ and $\sum_{x_k=1} d^*(\mathcal{Y}_k)^2 \geq \hat{\rho}^2$ for all $\mathbf{x} \in \mathcal{H} \setminus \{\mathbf{0}_p\}$ **do**
 - 3: Choose $\bar{\mathbf{x}} \in \{0, 1\}^p \setminus (\mathcal{H} \cup \bar{\mathcal{H}})$ to minimize $\sum_{x_k=1} d^*(\mathcal{Y}_k)^2$.
 - 4: Set $E \leftarrow 1$, $\mathcal{H}_0 \leftarrow \mathcal{H}$, and $\mathcal{H}_j \leftarrow \emptyset$ for $j = 1, \dots, J - 1$.
 - 5: **for** each $\mathbf{x} \in \bar{\mathcal{H}} \cup \{\bar{\mathbf{x}}\}$ in increasing order of $\sum_{x_k=1} d^*(\mathcal{Y}_k)^2$ **do**
 - 6: Set $j \leftarrow 1$.
 - 7: **while** $j < J$ and $\mathcal{H}_j \neq \emptyset$ **do**
 - 8: Set $\tilde{\mathcal{H}}_j \leftarrow [\bigcup_{\mathbf{z} \in \mathbb{Z}^p} (\mathcal{H}_j \oplus \mathbf{x} \oplus 2\mathbf{z})] \cap \{0, 1\}^p$, $\hat{\mathcal{H}} \leftarrow [\bigcup_{\tilde{\mathbf{x}} \in \tilde{\mathcal{H}}_j} \bigcup_{\mathbf{z} \in \mathbb{Z}^p} (\mathcal{H}_0 \oplus \tilde{\mathbf{x}} \oplus 2\mathbf{z})] \cap \{0, 1\}^p$, and $\hat{\mathcal{H}}_0 \leftarrow \mathcal{H}_0 \cup \hat{\mathcal{H}}$, $\hat{\mathcal{H}}_k \leftarrow [\bigcup_{\hat{\mathbf{x}} \in \hat{\mathcal{H}}} \bigcup_{\mathbf{z} \in \mathbb{Z}^p} (\mathcal{H}_k \oplus \hat{\mathbf{x}} \oplus 2\mathbf{z})] \cap \{0, 1\}^p$, $k = 1, \dots, J - 1$.
 - 9: If $\hat{\mathcal{H}}_0, \dots, \hat{\mathcal{H}}_{J-1}$ are pairwise disjoint, set $\mathcal{H}_k \leftarrow \hat{\mathcal{H}}_k$ for $k = 0, \dots, J - 1$ and $\mathcal{H} \leftarrow \mathcal{H}_0$, and **break**; otherwise, set $j \leftarrow j + 1$.
 - 10: **end while**
 - 11: **if** $j = J$ **then**
 - 12: Set $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\mathbf{h} + \mathbf{x}) \bmod 2 : \mathbf{h} \in \mathcal{H}\}$, set $\bar{\mathcal{H}} \leftarrow \bar{\mathcal{H}} \setminus \mathcal{H}$, $E \leftarrow 0$, and **break**.
 - 13: **else if** $\mathcal{H}_j = \emptyset$ **then**
 - 14: Set $\mathcal{H}_j \leftarrow \{\mathbf{x}\}$.
 - 15: **end if**
 - 16: **end for**
 - 17: If $E = 1$, set $\mathcal{H} \leftarrow \mathcal{H}_0$ and $\bar{\mathcal{H}} \leftarrow \bar{\mathcal{H}} \cup \{\bar{\mathbf{x}}\}$.
 - 18: **end while**
 - 19: If $m(\mathcal{H}) \neq 2^q$ or $\sum_{x_k=1} d^*(\mathcal{Y}_k)^2 < \hat{\rho}^2$ for some $\mathbf{x} \in \mathcal{H} \setminus \{\mathbf{0}_p\}$, **return** that no admissible $\mathcal{L}$ exists.
 - 20: **return** $\mathcal{L} = \bigcup_{\mathbf{z} \in \mathbb{Z}^p} (\mathcal{H} \oplus 2\mathbf{z})$.
-

standard basis vectors are assigned to $\bar{\mathcal{H}}$. When $q(\mathcal{L}) = q$, $\mathbb{Z}^p$ is partitioned into 2^{p-q} disjoint cosets $\mathcal{L}_0, \dots, \mathcal{L}_{2^{p-q}-1}$, where $\mathcal{L}_0 = \mathcal{L}$ and each $\mathcal{L}_j$, $j > 0$, is a translate of $\mathcal{L}$ of the form $\mathcal{L}_j = \mathcal{L} \oplus \mathbf{z}_j$ for some representative vector $\mathbf{z}_j \in \{0, 1\}^p$. Under the lattice constraints,

3.3 Design construction

if $\mathbf{x} \in \mathcal{L}_0$ and $\mathbf{z} \in \mathcal{L}_j$, then $\mathbf{x} + \mathbf{z} \in \mathcal{L}_j$; if $\mathbf{x}, \mathbf{z} \in \mathcal{L}_j$, then $\mathbf{x} + \mathbf{z} \in \mathcal{L}_0$. Line 8 enforces these relations, and Lines 7–10 use them to determine feasible coset assignments. Subject to these constraints, the remaining vectors $\mathbf{x} \in \{0, 1\}^p$ are considered in increasing order of $\sum_{x_k=1} d^*(\mathcal{Y}_k)^2$. Whenever feasible, a vector is assigned to $\bar{\mathcal{H}}$ so that vectors with smaller values of this quantity are excluded from $\mathcal{H}$, thereby favoring a candidate lattice $\mathcal{L}$ with a larger separation distance. Among candidate lattices with the same separation distance, this ordering also tends to favor a larger second-smallest value of this quantity over $\mathcal{H} \setminus \{\mathbf{0}_p\}$.

The procedure is deterministic and terminates in one of two cases. If the lattice constraints require a nonzero vector $\mathbf{x}$ satisfying $\sum_{x_k=1} d^*(\mathcal{Y}_k)^2 < \hat{\rho}^2$ to belong to $\mathcal{H}$, then no admissible lattice can attain a separation distance of at least $\hat{\rho}$, and the current configuration is discarded. Otherwise, the construction continues until $\mathcal{H}$ contains 2^q elements, ensuring that $q(\mathcal{L}) = q$ and yielding a valid candidate lattice $\mathcal{L}$.

Algorithm 9 provides the main procedure for constructing ILMDMs for $6 \leq p \leq 7$. It combines the two strategies described above. The T th largest separation distance serves as the retention threshold. We recommend setting $T = 1$ to retain only designs attaining the largest separation distance.

The case $q = r = p$, corresponding to $\mathcal{L}(\mathbf{G}_{p,1}) = \mathbb{Z}^p$, is already considered in Algorithm 7. The subsequent search therefore considers only $q < p$ and examines $r = q - 1, \dots, 0$ in decreasing order. Line 9 uses the run-size bounds in Theorem 2 to screen candidate $(q, \mathbf{s})$ configurations. Line 14 further applies an early-stopping rule motivated by Theorem 3, which shows that the maximum attainable run size over the corresponding lattice class is nondecreasing in r . Because the candidate lattice constructed for a given r need not

3.3 Design construction

Algorithm 9 Construct an ILMDM for $6 \leq p \leq 7$.

Input: $p, n, \mathcal{G}$, the collections $\mathcal{F}(\mathcal{G}_k, s_k)$, and T .

- 1: For each k , set $\mathcal{S}_k \leftarrow \{2, \dots, m(\mathcal{G}_k)\}$ if $\mathcal{G}_k$ is finite and $\mathcal{S}_k \leftarrow \{2, 3, \dots\}$ if $\mathcal{G}_k = [0, w_k]$.
 - 2: Use Algorithm 7 to obtain the candidate set $\mathcal{C}$.
 - 3: Set $\hat{\rho}$ to the smallest separation distance in $\mathcal{C}$ if $m(\mathcal{C}) < T$ and to the T th largest separation distance otherwise. In the latter case, remove candidates with separation distances smaller than $\hat{\rho}$.
 - 4: For each k with finite $\mathcal{G}_k$, set $\mathcal{S}_k \leftarrow \{z \in \mathcal{S}_k : \max_{\gamma \in \mathcal{F}(\mathcal{G}_k, z)} d^+(\gamma) \geq \hat{\rho}\}$.
 - 5: For each k with $\mathcal{G}_k = [0, w_k]$, retain $z \in \mathcal{S}_k$ if $z = 2$, if $2w_k/(z-1) \geq \hat{\rho}$ for odd z , or if $2w_k/(z-2) > \hat{\rho}$ for even $z > 2$.
 - 6: **for** $q = p-1$ down to 1 **do**
 - 7: Set $\tilde{s} \leftarrow \mathbf{0}_p$.
 - 8: **while** $\{z \in \prod_{k=1}^p \mathcal{S}_k : z \succ_{\text{lex}} \tilde{s}, z_p \geq 2 \lceil 2^{p-q-1} n \{ \prod_{k=1}^{p-1} 2 \lceil z_k/2 \rceil \}^{-1} \rceil - 1\} \neq \emptyset$ **do**
 - 9: Set $\mathbf{s} \leftarrow \min_{\text{lex}} \{z \in \prod_{k=1}^p \mathcal{S}_k : z \succ_{\text{lex}} \tilde{s}, z_p \geq 2 \lceil 2^{p-q-1} n \{ \prod_{k=1}^{p-1} 2 \lceil z_k/2 \rceil \}^{-1} \rceil - 1\}$,
and set $E \leftarrow 0$.
 - 10: For each k , choose $\mathcal{Y}_k \in \mathcal{F}(\mathcal{G}_k, s_k)$ to maximize $d^*(\mathcal{Y}_k)$ subject to $d^+(\mathcal{Y}_k) \geq \hat{\rho}$.
 - 11: **for** $r = q-1$ down to 0 **do**
 - 12: Use Algorithm 8 to construct an admissible candidate lattice $\mathcal{L}$ for the current
($q, r, \mathbf{s}, \mathcal{Y}$) and separation-distance threshold $\hat{\rho}$.
 - 13: If no such $\mathcal{L}$ is returned, **continue** to the next value of r ; otherwise, set $E \leftarrow 1$.
 - 14: If $m(\mathcal{D}(\mathcal{L}, \mathbf{s}, \mathcal{Y}, \mathbf{0}_p)) < n$, **break** the loop over r .
 - 15: Run Algorithm 4 to obtain $\tilde{\mathcal{Y}}$ and set $\tilde{\rho} \leftarrow \rho(\mathcal{D}(\mathcal{L}, \mathbf{s}, \tilde{\mathcal{Y}}, \mathbf{0}_p))$.
 - 16: **if** $\tilde{\rho} \geq \hat{\rho}$ **then**
 - 17: Add $(\tilde{\rho}, \mathcal{L}, \mathbf{s}, \tilde{\mathcal{Y}})$ to $\mathcal{C}$ and update $\hat{\rho}$, $\mathcal{C}$, and $\mathcal{S}_1, \dots, \mathcal{S}_p$ according to Lines 3–5.
 - 18: **end if**
 - 19: **end for**
 - 20: Set $\tilde{s} \leftarrow \mathbf{s}$; if $E = 0$, set $\tilde{s}_p \leftarrow +\infty$.
 - 21: **end while**
 - 22: **end for**
 - 23: Set $\hat{\rho} \leftarrow \max\{\rho : (\rho, \mathcal{L}, \mathbf{s}, \mathcal{Y}) \in \mathcal{C}\}$ and retain only candidates in $\mathcal{C}$ with separation distance $\hat{\rho}$.
 - 24: Apply Algorithm 5 to the retained candidates to obtain $\hat{\mathcal{L}}, \hat{\mathbf{s}}, \hat{\mathcal{Y}}, \hat{\mathbf{u}}$, and $\hat{m}$.
 - 25: **return** $\mathcal{D}(\hat{\mathcal{L}}, \hat{\mathbf{s}}, \hat{\mathcal{Y}}, \hat{\mathbf{u}})$, $\hat{\rho}$, and $\hat{m}$.
-

3.3 Design construction

attain this class-level maximum, the resulting pruning step is heuristic. Together, the restricted lattice construction and the early-stopping rule make Algorithm 9 nonexhaustive. Consequently, the algorithm does not guarantee identification of the ILDM with the largest separation distance over the full admissible search space.

The number of admissible (q, r) configurations and associated lattice structures grows combinatorially with p (He, 2017), making Algorithm 9 increasingly computationally expensive as p increases. For $p > 7$, the resulting computational and memory requirements become prohibitive, motivating a different construction strategy.

3.3.3 Construction for $p > 7$

For $p > 7$, Algorithm 10 constructs ILMDMs by sequentially augmenting seven-dimensional candidate designs, thereby avoiding a direct search over all admissible $(q, r, \mathbf{s})$ configurations in the full p -dimensional space.

The coordinates are first ordered according to their prespecified weights so that $w_1 \geq \dots \geq w_p$, after which Algorithm 9 is applied to the first seven coordinates to obtain a collection of initial designs. We recommend setting $T = 20$ to retain a sufficiently rich collection of initial designs, and we let $\hat{T}$ denote the number of designs retained. Because fewer than T initial designs may be available and ties at the retention threshold are retained, $\hat{T}$ need not equal T . For each retained initial design, the remaining coordinates $k = 8, \dots, p$ are added sequentially in decreasing order of w_k . Motivated by the empirical findings of He (2019), we set $s_k = 2$ and $\mathcal{Y}_k = \{0, w_k\}$ for each subsequently added coordinate, thereby further reducing the search space.

Algorithm 10 Construct an ILMDM for $p > 7$.

Input: $p, n, \mathcal{G}$, the collections $\mathcal{F}(\mathcal{G}_k, s_k)$, and T .

- 1: Permute the coordinates of $\mathbf{w}$ and $\mathcal{G}$ so that $w_1 \geq \dots \geq w_p$.
 - 2: Execute Lines 1–22 of Alg. 9 with $p = 7$ and retain $(\mathcal{L}^{(i,7)}, \mathbf{s}^{(i,7)}, \mathcal{Y}^{(i,7)})$, $i = 1, \dots, \hat{T}$.
 - 3: Set $\hat{\rho} \leftarrow 0$ and $t \leftarrow 0$.
 - 4: **for** $i = 1$ to $\hat{T}$ **do**
 - 5: **for** $j = 7$ to $p - 1$ **do**
 - 6: Set $\mathcal{H}^{(i,j)} \leftarrow \mathcal{L}^{(i,j)} \cap \{0, 1\}^j$, $\mathcal{X} \leftarrow \mathcal{H}^{(i,j)} \setminus \{\mathbf{0}_j\}$, $\mathcal{H}_0 \leftarrow \{\mathbf{0}_j\}$, and $\mathcal{H}_1 \leftarrow \emptyset$.
 - 7: **while** $\mathcal{X} \neq \emptyset$ **do**
 - 8: Choose $\mathbf{x} \in \mathcal{X}$ that minimizes $\sum_{k=1} d^*(\mathcal{Y}_k^{(i,j)})^2$.
 - 9: Set $\mathcal{H}_0 \leftarrow \mathcal{H}_0 \cup \{(\mathbf{y} + \mathbf{x}) \bmod 2 : \mathbf{y} \in \mathcal{H}_1\}$, $\mathcal{H}_1 \leftarrow \{(\mathbf{y} + \mathbf{x}) \bmod 2 : \mathbf{y} \in \mathcal{H}_0\}$, and $\mathcal{X} \leftarrow \mathcal{H}^{(i,j)} \setminus (\mathcal{H}_0 \cup \mathcal{H}_1)$.
 - 10: **end while**
 - 11: Set $\mathcal{L}_0 \leftarrow \bigcup_{z \in \mathbb{Z}^j} (\mathcal{H}_0 \oplus 2z)$ and $\mathcal{L}_1 \leftarrow \bigcup_{z \in \mathbb{Z}^j} (\mathcal{H}_1 \oplus 2z)$.
 - 12: Set $\mathcal{L}^{(i,j+1)} \leftarrow \{(\mathbf{y}, 2z) : \mathbf{y} \in \mathcal{L}_0, z \in \mathbb{Z}\} \cup \{(\mathbf{y}, 2z + 1) : \mathbf{y} \in \mathcal{L}_1, z \in \mathbb{Z}\}$, $\mathbf{s}^{(i,j+1)} \leftarrow (\mathbf{s}^{(i,j)}, 2)$, and $\mathcal{Y}^{(i,j+1)} \leftarrow \mathcal{Y}^{(i,j)} \times \{0, w_{j+1}\}$.
 - 13: **end for**
 - 14: Run Algorithm 4 with $(\mathcal{G}, \mathcal{L}^{(i,p)}, \mathbf{s}^{(i,p)})$ to obtain $\tilde{\rho}$ and $\tilde{\mathcal{Y}}$.
 - 15: **if** $\tilde{\rho} > \hat{\rho}$ **then**
 - 16: Set $\hat{\rho} \leftarrow \tilde{\rho}$, $t \leftarrow 1$, and $(\hat{\mathcal{L}}^{(1)}, \hat{\mathbf{s}}^{(1)}, \hat{\mathcal{Y}}^{(1)}) \leftarrow (\mathcal{L}^{(i,p)}, \mathbf{s}^{(i,p)}, \tilde{\mathcal{Y}})$.
 - 17: **else if** $\tilde{\rho} = \hat{\rho}$ **then**
 - 18: Set $t \leftarrow t + 1$ and $(\hat{\mathcal{L}}^{(t)}, \hat{\mathbf{s}}^{(t)}, \hat{\mathcal{Y}}^{(t)}) \leftarrow (\mathcal{L}^{(i,p)}, \mathbf{s}^{(i,p)}, \tilde{\mathcal{Y}})$.
 - 19: **end if**
 - 20: **end for**
 - 21: Apply Algorithm 5 to the t retained candidates to obtain $\hat{\mathcal{L}}, \hat{\mathbf{s}}, \hat{\mathcal{Y}}, \hat{\mathbf{u}}$, and $\hat{m}$.
 - 22: Apply the inverse permutation from Line 1 to $\hat{\mathcal{L}}, \hat{\mathbf{s}}, \hat{\mathcal{Y}}$, and $\hat{\mathbf{u}}$.
 - 23: **return** $\mathcal{D}(\hat{\mathcal{L}}, \hat{\mathbf{s}}, \hat{\mathcal{Y}}, \hat{\mathbf{u}})$, $\hat{\rho}$, and $\hat{m}$.
-

At each augmentation step, the finite local point set $\mathcal{H}^{(i,j)} = \mathcal{L}^{(i,j)} \cap \{0, 1\}^j$ is partitioned into two subsets, $\mathcal{H}_0$ and $\mathcal{H}_1$, subject to the lattice closure conditions. Starting with $\mathbf{0}_j \in \mathcal{H}_0$, the remaining points are considered in increasing order of $\sum_{k=1} d^*(\mathcal{Y}_k^{(i,j)})^2$, and the point selected at each iteration is assigned to $\mathcal{H}_1$, as specified in Lines 6–9. Under coordinatewise addition modulo two, the sum of two points from the same subset belongs to $\mathcal{H}_0$, whereas

3.3 Design construction

the sum of two points from different subsets belongs to $\mathcal{H}_1$. By Theorem 5, assigning a point to $\mathcal{H}_1$ adds w_{j+1}^2 to its squared-distance contribution through the newly added coordinate. This ordering therefore prioritizes points with smaller distance contributions and tends to yield a larger separation distance.

The partition induces the decomposition $\mathcal{L}^{(i,j)} = \mathcal{L}_0 \cup \mathcal{L}_1$, where $\mathcal{L}_0$ is the periodic extension of $\mathcal{H}_0$ and $\mathcal{L}_1$ is the complementary coset. In Lines 11–12, even and odd indices in the newly added coordinate are assigned to $\mathcal{L}_0$ and $\mathcal{L}_1$, respectively, corresponding to the levels 0 and w_{j+1} . The resulting $\mathcal{L}^{(i,j+1)}$ remains a standard IL, with $\mathbf{s}^{(i,j+1)} = (\mathbf{s}^{(i,j)}, 2)$ and $\mathcal{Y}^{(i,j+1)} = \mathcal{Y}^{(i,j)} \times \{0, w_{j+1}\}$. Moreover, because each point in $\mathcal{L}^{(i,j)}$ belongs to exactly one of $\mathcal{L}_0$ and $\mathcal{L}_1$, the augmentation preserves the run size at zero shift. Hence, $m(\mathcal{D}(\mathcal{L}^{(i,j+1)}, \mathbf{s}^{(i,j+1)}, \mathcal{Y}^{(i,j+1)}, \mathbf{0}_{j+1})) = m(\mathcal{D}(\mathcal{L}^{(i,j)}, \mathbf{s}^{(i,j)}, \mathcal{Y}^{(i,j)}, \mathbf{0}_j))$.

After all coordinates have been added, Algorithm 4 is applied to optimize the level configuration for each augmented candidate. Because Algorithm 10 starts from candidates produced by Algorithm 9, which is nonexhaustive, and restricts each added coordinate to $s_k = 2$ with $\mathcal{Y}_k = \{0, w_k\}$, it searches only a subset of the admissible ILDM class and therefore does not guarantee identification of a design with the largest separation distance over the full search space.

3.3.4 Computational performance and implementation considerations

The three algorithms balance computational cost against the attained separation distance, with representative computing times reported in Table 4. For $2 \leq p \leq 5$, Algorithm 6 exhaustively searches the specified class of IL-based designs and therefore attains the largest

3.3 Design construction

Table 4: Computing times in seconds for the three algorithms at selected (p, n) settings.

(p, n)	Alg. 6	Alg. 9	(p, n)	Alg. 9	Alg. 10
(3,150)	0.220	0.015	(7,300)	54.4	–
(3,2000)	0.543	0.037	(7,500)	498	–
(5,200)	15.4	1.50	(8,500)	13,840	509
(5,500)	48.9	1.48	(9,300)	13,447	504
(5,1000)	61.3	6.10	(15,100)	13,215	61.5

separation distance within that class. For $\mathcal{G} = [0, 1]^{p-1} \times \{0, 0.1, \dots, 1\}$, Algorithms 6 and 9 attain the same separation distances for $p = 2, \dots, 5$ and $2 \leq n \leq 500$, although Algorithm 9 is faster. Their agreement is not universal. For $\mathcal{G} = [0, 1]^4 \times \{r/100 : r \text{ is prime and } 2 \leq r \leq 97\}$, and target run size $n = 272$, Algorithm 6 returns $m = 272$ runs with separation distance 0.5000, whereas Algorithm 9 returns $m = 288$ runs with separation distance 0.4714. Their computing times are 50.4 and 2.65 seconds, respectively. We therefore recommend Algorithm 6 for $2 \leq p \leq 5$, where its exact search remains computationally feasible, and use Algorithm 9 for $6 \leq p \leq 7$.

For $p > 7$, Algorithm 10 substantially reduces the computational cost, although it can attain a smaller separation distance. For example, for $p = 9$, $n = 20$, and $\mathcal{G} = [0, 1]^8 \times \{0, 0.1, \dots, 1\}$, both algorithms return $m = 32$ runs. Algorithm 9 attains separation distance 1.7321 in 1.31 seconds, whereas Algorithm 10 attains separation distance 1.4142 in 0.391 seconds. We therefore recommend Algorithm 10 for $p > 7$ when computational efficiency is the primary consideration, while Algorithm 9 may be preferred when separation

distance has priority and the additional computational cost is acceptable.

Compared with the ILMDC construction of He (2019), the proposed algorithms require two additional considerations. First, the level configuration $\mathcal{Y}$ is treated as an additional design variable and is optimized as part of the construction, whereas the coordinate levels in an ILMDC are fixed and equally spaced once the span vector $\mathbf{s}$ is specified. For each ordinal variable, the selected levels must belong to its prespecified finite set of allowable levels, which also constrains the admissible values of s_k . Second, after maximizing the separation distance, an additional step selects the shift that yields the smallest run size that is at least the target value n , a step that is not required in the ILMDC construction of He (2019).

4. Numerical comparisons

4.1 Separation distance evaluation

We compare ILMDM with two groups of benchmark designs. The maximin distance benchmarks include ILMDC (He, 2019), the method of Sun et al. (2019), referred to here as the heuristic maximin design (HMD), Construction 2 of Stinstra et al. (2003), referred to here as the constrained maximin design (CMD), the sliced maximin Latin hypercube design (SLHD) (Ba et al., 2015), and the maximin Latin hypercube design (MLHD) implemented in the `SFDesign` package (Wang and Joseph, 2025). We also consider space-filling designs for mixed inputs, including the MaxPro design for quantitative and qualitative variables (MaxProQQ) (Joseph et al., 2020), MCD (Deng et al., 2015), the fast flexible filling design (FFF) (Lekivetz and Jones, 2015), and the fractional factorial design (FFA) (Wu and

4.1 Separation distance evaluation

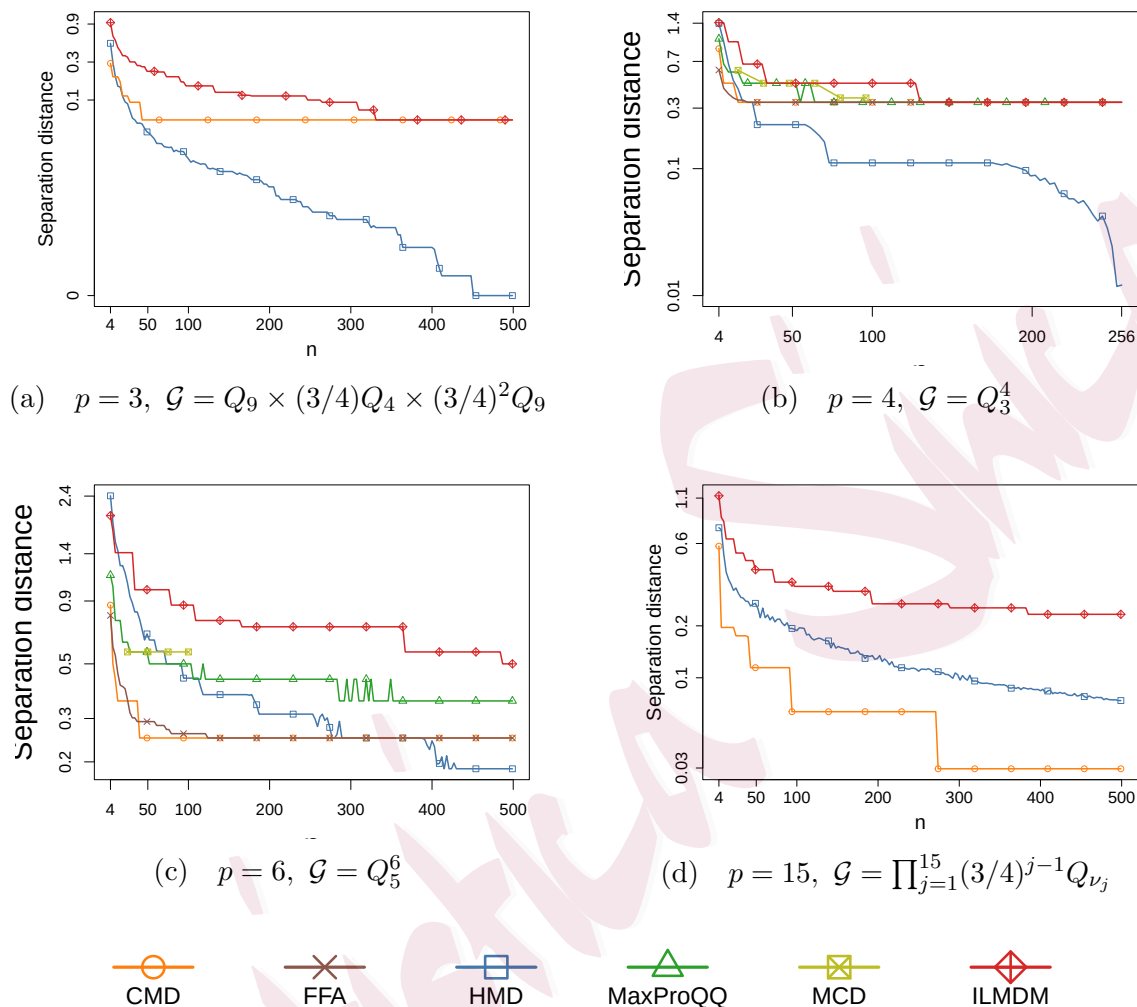
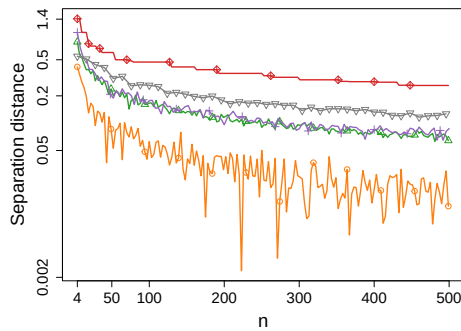


Figure 2: Separation distances of six types of designs for cases with only ordinal inputs. For panel (d), $\nu = (2, 1, 4, 1, 1, 6, 1, 1, 1, 1, 5, 2, 7, 1, 1)$.

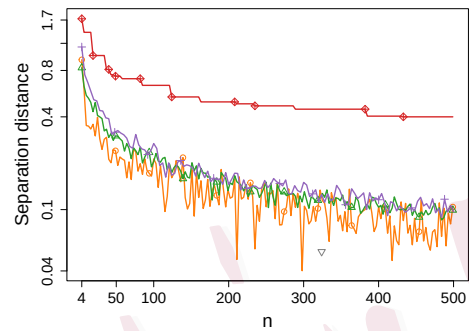
(Hamada, 2011).

The benchmark methods have different admissible input spaces and construction requirements. MLHD and ILMDC are restricted to continuous inputs. For HMD, FFA, MCD, and SLHD, ordinal variables are treated as categorical variables. MCD requires suitable orthogonal arrays and is therefore used only for combinations of run sizes and factor levels for which appropriate orthogonal arrays are available, with implementation details provided in

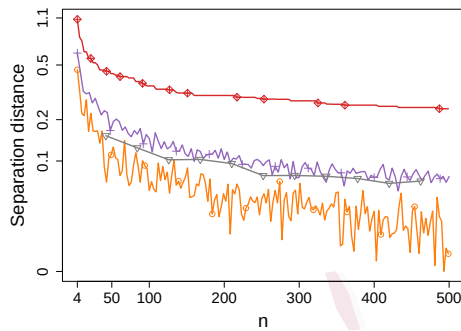
4.1 Separation distance evaluation



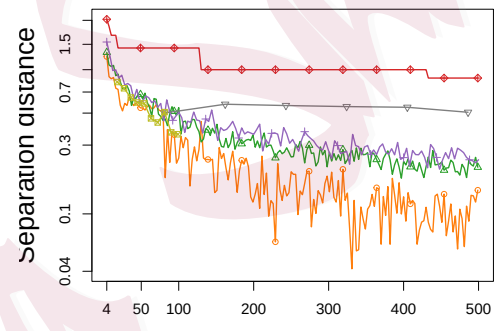
(a) $p = 4, \mathcal{G} = [0, 1]^3 \times Q_3$



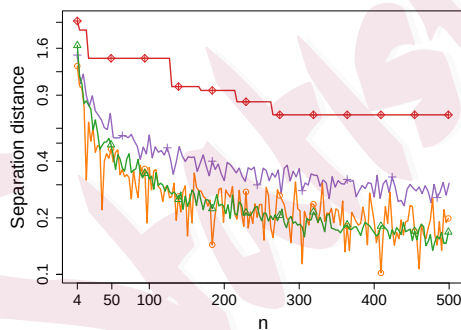
(b) $p = 5, \mathcal{G} = [0, 1]^2 \times Q_1 \times Q_6 \times Q_{10}$



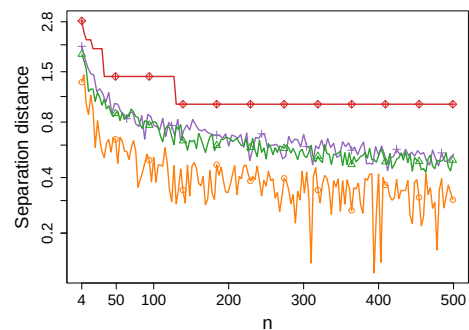
(c) $p = 5, \mathcal{G} = [0, 1] \times (3/4)Q_7 \times [0, (3/4)^2] \times [0, (3/4)^3] \times (3/4)^4Q_8$



(d) $p = 8, \mathcal{G} = [0, 1]^4 \times Q_2^4$



(e) $p = 8, \mathcal{G} = [0, 1]^4 \times Q_1 \times Q_4 \times Q_5 \times Q_9$



(f) $p = 12, \mathcal{G} = [0, 1]^3 \times Q_1^3 \times Q_2 \times \prod_{i=4}^8 Q_i$

CMD

FFF

MaxProQQ

MCD

SLHD

ILMDM

Figure 3: Separation distances for six types of designs with mixed continuous and ordinal variables.

4.1 Separation distance evaluation

the Supplementary Material. For SLHD, the joint ordinal profiles are treated as the levels of a single slicing factor (Ba et al., 2015), and thus the run size must be divisible by the number of such profiles. CMD incorporates restrictions on admissible levels as equality constraints. FFA is obtained by random subsampling from the full factorial array. FFA, MaxProQQ, MCD, MLHD, and SLHD are restricted to equal weights across variables.

We compare the separation distances of the methods across various p and n under three categories of scenarios, namely those with only ordinal inputs, those with both continuous and ordinal inputs, and those with only continuous inputs. Figure 2 presents four scenarios involving only ordinal inputs and based on ten ordinal input sets: $Q_1 = \{0, 1\}$, $Q_2 = \Delta_{0.5}$, $Q_3 = \Delta_{1/3}$, $Q_4 = \{0, 0.1, 0.3, 0.6, 1\}$, $Q_5 = \Delta_{0.25}$, $Q_6 = \{0, 0.2, 0.3, 0.5, 0.7, 1\}$, $Q_7 = \Delta_{0.2}$, $Q_8 = \{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1\}$, $Q_9 = \Delta_{0.1}$, and $Q_{10} = \{0, 1\} \cup \{z/100 : z \text{ is a prime}, z < 100\}$, where $\Delta_x = \{0, x, 2x, \dots, 1\}$. Figure 3 presents six scenarios involving both continuous and ordinal inputs, while Figure 4 presents the scenarios involving only continuous inputs. Because the separation distances vary substantially across methods, the results are displayed on a logarithmic scale.

Across the settings considered in Figures 2–4, ILMDM attains the largest separation distance among the compared designs in most cases. In scenarios with only continuous inputs, ILMDC yields the same separation distance as ILMDM in most settings, whereas ILMDM attains a slightly larger value in a few cases. For scenarios with only ordinal inputs, MCD matches ILMDM at $n = 48$ and 64 when $p = 4$, but attains smaller separation distances at the remaining MCD-admissible run sizes considered for $p = 4$ and 6 . MaxProQQ and CMD also attain separation distances comparable to those of ILMDM in scenarios with

4.1 Separation distance evaluation

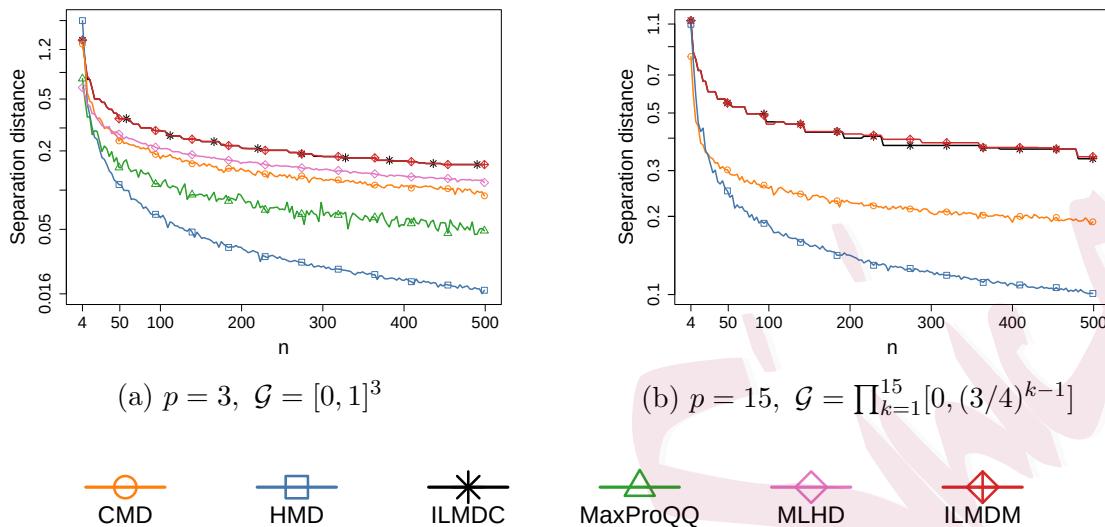


Figure 4: Separation distances for six types of designs in scenarios with only continuous inputs.

only ordinal inputs when p is small. In the remaining scenarios, ILMDM generally attains larger separation distances than the benchmark designs considered.

We examine the effect of coordinate weights by comparing designs constructed under different weight specifications. Table 5 reports the span vectors $\mathbf{s}$ for designs obtained at several run sizes n with $p = 3, \mathcal{G} = Q_4 \times [0, 1]^2$, and weight ratios given by permutations of $1 : 2 : 3$. Variables with larger weights tend to receive more levels, subject to the admissible level sets of the ordinal variables and the run-size constraints. For example, when the weight ratio is $3 : 1 : 2$ and $n = 30$, the first variable receives only five levels because Q_4 contains only five admissible levels. Thus, the table illustrates how the construction allocates levels according to the prespecified coordinate weights while respecting the admissible level sets.

4.2 Predictive performance evaluation

Table 5: Vectors $\mathbf{s}$ for different weight ratios and run sizes.

n	1:2:3	1:3:2	2:1:3	2:3:1	3:1:2	3:2:1
10	(2, 3, 4)	(2, 4, 3)	(3, 2, 4)	(3, 4, 2)	(5, 2, 4)	(5, 4, 2)
30	(3, 5, 7)	(3, 7, 5)	(5, 3, 7)	(5, 7, 3)	(5, 4, 6)	(5, 6, 4)
60	(3, 5, 8)	(3, 8, 5)	(5, 3, 8)	(5, 8, 3)	(5, 3, 8)	(5, 8, 3)
100	(3, 5, 13)	(3, 13, 5)	(5, 3, 13)	(5, 13, 3)	(5, 4, 10)	(5, 10, 4)

4.2 Predictive performance evaluation

4.2.1 Branin function

To assess the predictive performance of geometrically well-separated ILMDMs for surrogate modeling with mixed continuous and discrete inputs, we compare ILMDM with competing designs on two test problems. The first test problem uses a modified Branin function. Following Pelamatti et al. (2021), the function involves two continuous variables x_1 and x_2 and two numerically coded binary variables x_3 and x_4 , each taking values in $\{0, 1\}$. The mixed Branin function is defined as

$$f(x_1, x_2, x_3, x_4) = \begin{cases} h(x_1, x_2), & x_3 = 0, x_4 = 0, \\ 0.4 h(x_1, x_2) + 1.1, & x_3 = 0, x_4 = 1, \\ -0.75 h(x_1, x_2) + 5.2, & x_3 = 1, x_4 = 0, \\ -0.5 h(x_1, x_2) - 2.1, & x_3 = 1, x_4 = 1, \end{cases}$$

where $h(x_1, x_2) = \{(15x_2 - 5(15x_1 - 5)^2 / (4\pi^2) + 5(15x_1 - 5) / \pi - 6)^2 + 10(1 - 1 / (8\pi)) \cos(15x_1 - 5) - 44.8104\} / 51.9496$. The target run sizes are $n = 40, 60$, and 80 . The four binary profiles

4.2 Predictive performance evaluation

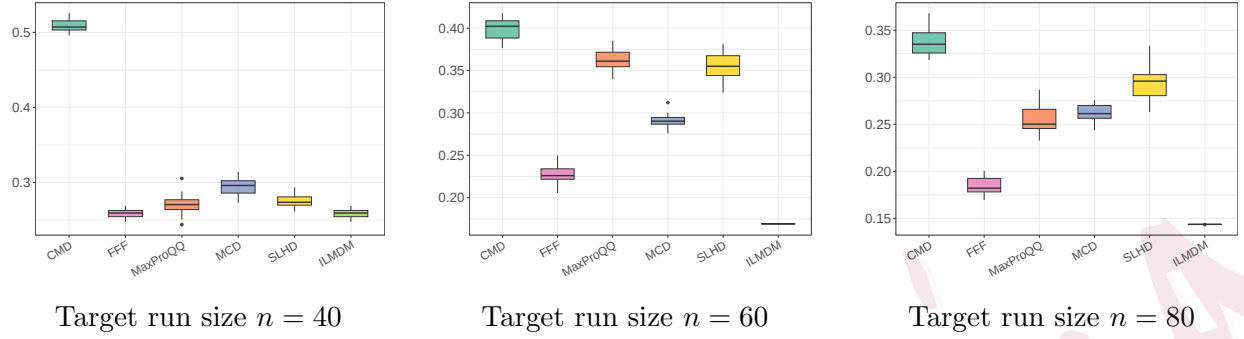


Figure 5: Boxplots of RMSE values for six design methods for the modified Branin function.

induce scaling and sign changes in the otherwise smooth continuous response surfaces, resulting in nonstationarity across the binary profiles.

For each method and target run size, we fit a Gaussian process emulator. We use **RobustGaSP** (Gu et al., 2019), with a mean function that is linear in all four inputs and a nugget fixed at zero. To accommodate the mixed inputs, continuous variables enter the correlation function through squared distances, whereas binary variables enter through indicators of level mismatch. Writing $\mathbf{x} = (x_1, x_2)$ and $\mathbf{z} = (x_3, x_4)$, we use the correlation function

$$R_B\{(\mathbf{x}, \mathbf{z}), (\mathbf{x}', \mathbf{z}')\} = \exp\left\{-\theta \sum_{j=1}^2 (x_j - x'_j)^2 - \theta \sum_{h=1}^2 I(z_h \neq z'_h)\right\}.$$

More general Gaussian process models for computer experiments with qualitative and quantitative inputs have been developed by Qian et al. (2008) and Zhou et al. (2011).

The fitted emulators are evaluated on the same 20 independently generated test sets across all methods at each target run size. Each test set contains 1,000 points, with 250 continuous Latin hypercube points for each of the four binary profiles. Figure 5 therefore summarizes variation across test sets rather than variation across independently generated training designs. Under this emulator and test protocol, ILMDM and FFF have the same

empirical distribution of RMSE values at $n = 40$, whereas ILMDM attains the lowest RMSE values at $n = 60$ and 80 among CMD, FFF, MaxProQQ, MCD, and ILMDM.

4.2.2 GKLS functions

Next, we use the Gaviano–Kvasov–Lera–Sergeyev (GKLS) test function generator developed by Gaviano et al. (2003) to evaluate predictive performance. Each GKLS function is constructed by adding local polynomial perturbations to a convex quadratic function. To construct a mixed-input test problem, we restrict the third coordinate to $Q_6 = \{0, 0.2, 0.3, 0.5, 0.7, 1\}$ while retaining continuous domains for the first two coordinates. Thus, $p = 3$ and $\mathcal{G} = [0, 1]^2 \times Q_6$. The proposed ILMDM is compared with CMD, FFF, MaxProQQ, and SLHD. We consider 20 GKLS functions for target run sizes ranging from $n = 10p = 30$ to $n = 500$. For each function, the same 100-point test set is used across all methods. SLHD is included only for the 14 target run sizes that permit equal slice sizes. For each function and design, we fit a Gaussian process emulator using **RobustGaSP** with a constant mean, a zero nugget, and the anisotropic Gaussian correlation function $R_G\{(\mathbf{x}, z), (\mathbf{x}', z')\} = \exp\{-\sum_{j=1}^2 \theta_j(x_j - x'_j)^2 - \eta(z - z')^2\}$.

Figure 6 reports the average RMSE over the 20 functions. ILMDM attains the lowest average RMSE at most of the evaluated run sizes, while FFF and MaxProQQ attain comparable average RMSE values at several run sizes between 290 and 350. CMD generally has larger average RMSE values, as does SLHD at the run sizes for which it is available.

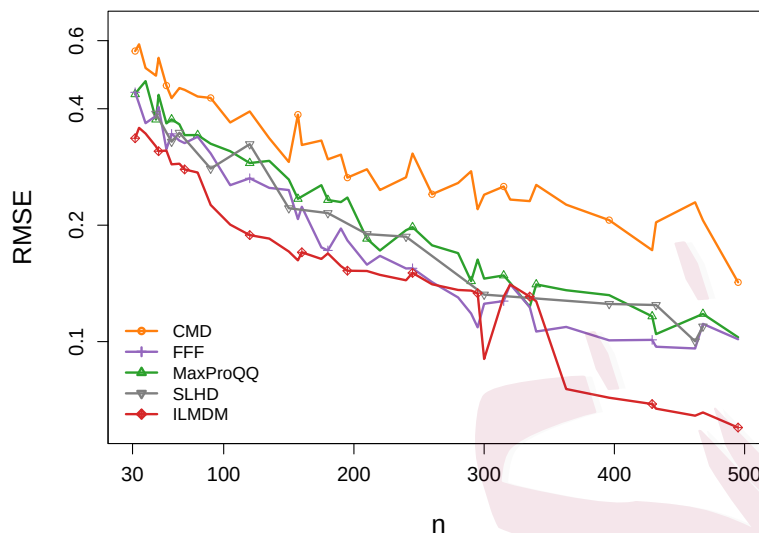


Figure 6: Average RMSE values for the design methods over 20 GKLS test functions at different run sizes.

5. Conclusion

This paper develops maximin distance designs for continuous, ordinal, and binary variables by incorporating prespecified ordinal level sets directly into the admissible design space, ensuring that the resulting designs satisfy discrete level constraints without post hoc discretization. Theoretical characterizations of run size, separation distance, and admissible level configurations support early screening and computationally tractable constructions across a broad range of run sizes and dimensions. In numerical studies, ILMDMs generally attain larger separation distances than the competing designs, with lower or comparable Gaussian process emulation errors, demonstrating the method's effectiveness and flexibility for mixed-input computer experiments. Extensions to multilevel nominal factors, constrained input spaces, and sequential experiments are directions for future research.

Supplementary Materials

The supplementary materials include proofs, a list of three-dimensional standard ILs, the source code for the R package used to generate the proposed designs, and the scripts used to reproduce the numerical studies.

Acknowledgements

The authors thank the three referees and the associate editor for their insightful comments. Xu He was supported by the National Natural Science Foundation of China (Grant Nos. 12288201 and 12471247), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA0500400), and the CAS Project for Young Scientists in Basic Research (Grant No. YSBR-149).

References

- Ba, S., W. R. Myers, and W. A. Brenneman (2015). Optimal sliced Latin hypercube designs. *Technometrics* 57(4), 479–487.
- Conway, J. H. and N. J. A. Sloane (2013). *Sphere packings, lattices and groups*, Volume 290. Springer.
- Deng, X., Y. Hung, and C. D. Lin (2015). Design for computer experiments with qualitative and quantitative factors. *Stat. Sin.* 25(4), 1567–1581.
- Gaviano, M., D. E. Kvasov, D. Lera, and Y. D. Sergeyev (2003). Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Trans. Math. Softw.* 29(4), 469–480.

REFERENCES

- Gu, M., J. Palomo, and J. O. Berger (2019). RobustGaSP: Robust gaussian stochastic process emulation in R. *The R Journal* 11(1), 112–136.
- Haaland, B. and P. Z. G. Qian (2011). Accurate emulators for large-scale computer experiments. *Ann. Statist.* 39(6), 2974–3002.
- Haaland, B., W. Wang, and V. Maheshwari (2018). A framework for controlling sources of inaccuracy in Gaussian process emulation of deterministic computer experiments. *SIAM/ASA J. Uncertain. Quantif.* 6(2), 497–521.
- He, X. (2017). Interleaved lattice-based minimax distance designs. *Biometrika* 104(3), 713–725.
- He, X. (2019). Interleaved lattice-based maximin distance designs. *Biometrika* 106(2), 453–464.
- Johnson, M., L. Moore, and D. Ylvisaker (1990). Minimax and maximin distance designs. *J. Stat. Plan. Infer.* 26(2), 131–148.
- Joseph, V. R., E. Gul, and S. Ba (2015, 03). Maximum projection designs for computer experiments. *Biometrika* 102(2), 371–380.
- Joseph, V. R., E. Gul, and S. Ba (2020). Designing computer experiments with multiple types of factors: The MaxPro approach. *J. Qual. Technol.* 52(4), 343–354.
- Lekivetz, R. and B. Jones (2015). Fast flexible space-filling designs for nonrectangular regions. *Qual. Reliab. Eng. Int.* 31(5), 829–837.
- Li, W., M. Liu, and B. Tang (2021). A method of constructing maximin distance designs. *Biometrika* 108(4), 845–855.
- Morris, M. D. and T. J. Mitchell (1995). Exploratory designs for computational experiments. *J. Stat. Plan. Infer.* 43(3), 381–402.
- Mu, W. and S. Xiong (2017). On algorithmic construction of maximin distance designs. *Commun. Stat.-Simul.*

REFERENCES

- Comput.* 46(10), 7972–7985.
- Pelamatti, J., L. Brevault, M. Balesdent, E.-G. Talbi, and Y. Guerin (2021). Mixed variable gaussian process-based surrogate modeling techniques: Application to aerospace design. *J. Aerosp. Inf. Syst.* 18(11), 813–837.
- Qian, P. Z. G., H. Wu, and C. F. J. Wu (2008). Gaussian process models for computer experiments with qualitative and quantitative factors. *Technometrics* 50(3), 383–396.
- Santner, T. J., B. J. Williams, W. I. Notz, and B. J. Williams (2003). *The design and analysis of computer experiments*, Volume 1. Springer.
- Sobol', I. M. (1993). Sensitivity estimates for nonlinear mathematical models. *Math. Model. Comput. Exp.* 1, 407.
- Stinstra, E., D. den Hertog, P. Stehouwer, and A. Vestjens (2003). Constrained maximin designs for computer experiments. *Technometrics* 45(4), 340–346.
- Sun, F., R. B. Gramacy, B. Haaland, S. Lu, and Y. Hwang (2019). Synthesizing simulation and field data of solar irradiance. *Stat. Anal. Data Min.* 12(4), 311–324.
- Trosset, M. W. et al. (1999). Approximate maximin distance designs. In *Proceedings of the Section on Physical and Engineering Sciences*, pp. 223–227.
- Wang, L., Q. Xiao, and H. Xu (2018). Optimal maximin L_1 -distance Latin hypercube designs based on good lattice point designs. *Ann. Statist.* 46(6B), 3741–3766.
- Wang, S. and R. Joseph (2025). *SFDesign: Space-Filling Designs*. R package version 0.1.2.
- Wu, C. J. and M. S. Hamada (2011). *Experiments: planning, analysis, and optimization*. John Wiley & Sons.
- Xiao, Q. and H. Xu (2017). Construction of maximin distance Latin squares and related Latin hypercube designs. *Biometrika* 104(2), 455–464.
- Yuan, R., Y. Yin, H. Xu, and M. Liu (2025). A construction method for maximin L_1 -distance Latin hypercube

REFERENCES

designs. *Stat. Sin.* 35(1), 1–24.

Zhou, Q., P. Z. G. Qian, and S. Zhou (2011). A simple approach to emulation for computer models with qualitative and quantitative factors. *Technometrics* 53(3), 266–273.

School of Mathematics, Statistics and Mechanics, Beijing University of Technology, Pingleyuan No. 100, Chaoyang District, Beijing, 100124, China.

E-mail: huilan@bjut.edu.cn; ORCID: orcid.org/0000-0003-0020-0048

State Key Laboratory of Mathematical Sciences (SKLMS), Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Haidian District, Beijing 100190, China.

E-mail: hexu@amss.ac.cn; ORCID: orcid.org/0000-0002-3187-7046