Tuning Differential Evolution Algorithm for Constructing Uniform Projection Designs

Samuel Onyambu and Hongquan Xu

Department of Statistics and Data Science,

University of California, Los Angeles, CA 90095, USA

June 24, 2026

Abstract

Space-filling designs are extensively used in computer experiments to analyze complex systems. Among these, uniform projection designs stand out for their desirable low-dimensional projection properties and robustness against other criteria. However, no efficient algorithm currently exists for generating such designs. This study explores the construction of uniform projection designs using a differential evolution (DE) algorithm. DE, an evolutionary algorithm, is known for its simplicity, robustness, and effectiveness in solving complex optimization problems, though its performance is highly sensitive to several hyperparameters. Our goal is to investigate the structure of the hyperparameter space, evaluate the contribution of each hyperparameter, and provide guidelines for optimal hyperparameter settings across various scenarios. To achieve this, we conduct a comprehensive comparison of different experimental designs and surrogate models.

Keywords: Computer experiment, experimental design, hyperparameter optimization, kriging model, metaheuristic algorithm, space-filling design.

1 Introduction

Experimental design construction is a fundamental aspect of research and data-driven inquiry, aimed at organizing experimental runs to extract maximum information while minimizing resource use. By strategically selecting input combinations, well-constructed designs ensure that researchers can identify key factors, estimate model parameters, and predict responses accurately (Montgomery 2017). The primary goal is to balance efficiency and comprehensiveness, whether in exploring high-dimensional spaces, optimizing processes, or assessing system robustness. Central to this process is the notion of space-filling, where design points are distributed to capture variations across the entire experimental domain, providing a solid foundation for modeling and inference (Santner, Williams, and Notz 2018).

Over the past few decades, computer experiments have become ubiquitous across engineering disciplines and the physical sciences. Space-filling designs play a fundamental role in these experiments, as they ensure a well-balanced exploration of the input space, preventing excessive sampling in certain regions while avoiding sparse coverage in others (Kamath 2022). Broadly speaking, a space-filling design refers to any design that distributes its points uniformly across the design domain. The uniformity of a design can be assessed based on criteria such as distance (Johnson, Moore, and Ylvisaker 1990), orthogonality (Owen 1994), or discrepancy (Fang et al. 2000).

Joseph (2016) provided a comprehensive overview of how space-filling designs help address optimization challenges associated with computationally demanding computer models. By minimizing model error and enhancing predictive accuracy, these designs significantly improve the reliability of simulation models, making them indispensable in applications such as kriging and optimization (Joseph et al. 2019). For an in-depth introduction to computer experiments and space-filling designs, see Fang, Li, and Sudjianto (2006) and Santner, Williams, and Notz (2018). For recent advancements in this field, refer to Chen and Tang (2024), Li, Tian, and Liu (2024), Shi and Xu (2024), Tian and Xu (2022), Tian and Xu (2024), Vázquez and Xu (2024), Wang, Liu, and Xiao (2024), Yin, Wang, and Xu (2023), and Yuan et al. (2025), among others.

In the pursuit of high-quality space-filling designs, uniform projection designs (UPDs) have emerged as a powerful approach for achieving uniformity across all low-dimensional projections of the design space (Sun, Wang, and Xu 2019). Introduced by Sun, Wang, and Xu (2019), UPDs are a specialized class of space-filling designs that exhibit strong performance across various design criteria while maintaining excellent space-filling properties in

multiple dimensions. These designs are particularly valuable in high-dimensional settings, where interactions among subsets of factors often hold critical importance. Wang, Sun, and Xu (2022) investigated the relationship between UPDs and maximin distance designs, establishing novel lower and upper bounds for UPDs. Additionally, Sun and Tang (2023) explored the connection between UPDs and strong orthogonal arrays of strength 2+, demonstrating that such arrays are either optimal or nearly optimal under the uniform projection criterion. Despite these advancements, existing algorithms for generating UPDs remain relatively underdeveloped, presenting a significant avenue for further research. To address this gap, we explore the application of Differential Evolution (DE), a powerful metaheuristic optimization algorithm, for constructing UPDs.

Metaheuristic algorithms, including simulated annealing, genetic algorithms, particle swarm optimization, and threshold accepting, have been employed in the construction of space-filling designs. These methods have been instrumental in generating designs such as maximin distance Latin hypercube designs (Chen et al. 2013; Liefvendahl and Stocki 2006; Morris and Mitchell 1995), uniform designs (Fang et al. 2000), and maximum projection designs (Joseph, Gul, and Ba 2015; Wang, Liu, and Xiao 2024). Recently, Stokes, Wong, and Xu (2024) introduced a novel Differential Evolution (DE) algorithm for constructing order-of-addition designs, demonstrating its superior efficiency compared to other metaheuristic approaches. Building on their insights, we adapt and extend this DE algorithm to facilitate the construction of UPDs.

The performance of the DE algorithm is significantly influenced by its hyperparameters, which dictate the learning process of the optimization (Price, Storn, and Lampinen 2006). An inappropriate setting of these hyperparameters can lead to suboptimal performance of the DE algorithm. While the variant DE algorithm proposed by Stokes, Wong, and Xu (2024) encompasses several hyperparameters, their effects remain largely unexamined. Therefore, we aim to conduct a comprehensive study of the hyperparameters to elucidate their impacts on the algorithm's performance, providing insights that could enhance its effectiveness.

The challenge of identifying the optimal hyperparameter configurations for any learning process is widely recognized, and researchers have sought to address it through two primary approaches: the model-based and model-free frameworks. Model-based hyperparameter optimization refines hyperparameters by approximating the underlying learning algorithm, whereas model-free methods tackle the optimization problem without imposing parametric assumptions. Foundational contributions to model-based hyperparameter optimization include the works of Falkner, Klein, and Hutter (2018) and Hutter, Hoos, and Leyton-Brown

(2011). In contrast, model-free approaches encompass a diverse array of techniques, such as manual search, grid search, random search, genetic algorithms, and orthogonal array tuning methods (Liashchynskyi and Liashchynskyi 2019).

We take an integrated approach based on experimental design and analysis. Our approach leverages various types of factorial designs and space-filling designs to investigate the effectiveness of DE in constructing UPDs. We also employ different types of surrogate models such as second-order models and kriging models to evaluate the performance of different designs, enabling us to visualize the response surface of the DE algorithm's hyperparameters. This framework comprises the data generation, modeling, and analysis procedures. The insights gained from our analysis provide a general guideline for optimal hyperparameter settings necessary for generating good UPDs. Specifically, we aim to address four objectives: (i) identifying useful design types in understanding the surface structure extended by the DE hyperparameters, (ii) determining the most effective surrogate models, (iii) determining efficient hyperparameter settings, and (iv) developing an efficient algorithm for the construction of UPDs. While we focus on tuning the DE algorithm, this methodology is versatile and can be applied to hyperparameter tuning in other optimization algorithms.

Our approach is quite different from the Lujan-Moreno et al. (2018) method which used a 2^k factorial design with the response surface method (RSM) and ridge regression for screening to select the important factors in the data. While Lujan-Moreno et al. (2018) focused on factor screening and selection with the traditional RSM approach, we emphasize on the comparisons of different types of designs and surrogate models in approximating the surface structure of the DE algorithm.

The paper is organized as follows. In Section 2, we review the DE algorithm proposed by Stokes, Wong, and Xu (2024) and its hyperparameters. We describe various experimental designs and surrogate models used in the study, respectively, in Sections 3 and 4. In Section 5, we describe the uniform projection criterion proposed by Sun, Wang, and Xu (2019) and the data generating process. Section 6 presents the results and Section 7 addresses factor importance and optimal hyperparameter settings. Finally, Section 8 concludes the paper.

2 Differential Evolution Algorithm

Originating from the pioneering work of Storn and Price (1997), DE has emerged as a powerful heuristic optimization algorithm. It is an example of metaheuristic algorithm which draws its inspiration from the mechanisms of biological evolution.

Metaheuristic algorithms are a class of optimization techniques designed to find optimal or near-optimal solutions to complex problems, particularly when traditional optimization methods are impractical or inefficient. These algorithms are generally inspired by natural or physical processes and do not guarantee an exact solution but are highly effective for solving large-scale, non-linear, multi-dimensional, and combinatorial problems. They are flexible, adaptable, and can be applied to a wide variety of optimization problems. They typically balance exploration and exploitation to achieve efficient convergence (Weise 2009).

Metaheuristic algorithms can be broadly classified into three categories based on their inspiration: (1) evolutionary algorithms, (2) swarm intelligence, and (3) physical phenomena-based algorithms. Evolutionary algorithms, such as genetic algorithms and DE, simulate the process of natural selection to evolve solutions. These methods are especially effective in maintaining solution diversity through operators like mutation and crossover, which prevent premature convergence (Storn and Price 1997). Swarm intelligence techniques, such as particle swarm optimization and ant colony optimization, mimic the collective behavior of groups in nature, such as bird flocks or ant colonies. These approaches excel in leveraging communication among agents to quickly explore vast solution spaces (Dorigo 2007). Lastly, physical phenomena-based algorithms, such as simulated annealing, draw from thermodynamics principles, allowing for probabilistic escape from local optima to find better solutions (Aarts and Korst 1989; Kirkpatrick, Gelatt Jr, and Vecchi 1983).

Among these, DE stands out due to its simplicity, efficiency, and robustness across diverse problem landscapes. Unlike other metaheuristics, DE uses differential mutations, which introduce solution diversity and improve exploration capabilities while maintaining computational efficiency. This makes DE particularly suited for handling non-linear, non-convex optimization problems. Furthermore, its ability to balance exploration and exploitation ensures convergence to high-quality solutions even in noisy or complex landscapes, a feature critical for problems with large and intricate design spaces (Das and Suganthan 2010; Yang 2020). By combining the strengths of population-based search and vectorized mutation strategies, DE demonstrates its utility in applications ranging from engineering optimization to machine learning.

To simulate survival-of-the-fittest dynamics, DE treats each candidate or agent as a chromosome made up of several genes and implements mutation and crossover procedures that allow beneficial genes to persist into future generations (Storn and Price 1997). DE operates on the principle of population-based search, where a set of candidate solutions evolves over successive generations towards optimal or near-optimal solutions. At its core,

DE employs mutation, crossover, and selection operators to iteratively improve the quality of solutions. The algorithm's efficacy stems from its robustness, simplicity, and ability to handle non-linear, non-convex, and noisy optimization landscapes.

Without loss of generality, we assume that we want to minimize a real-valued objective function ϕ over an m-dimension space Ω. It has five steps

1. **Genetic Representation**: Let $\pi_1, \ldots, \pi_N$ be the initial population, where each agent $\pi_i = (\pi_{i1}, \ldots, \pi_{im})$ is randomly chosen from Ω.

2. **Mutation**: Mutation expands the search space of the current population. For each $i = 1, \ldots, N$, mutation produces a potential donor ν_i in Ω by adding the weighted difference of two agents to a third, all randomly chosen and distinct from the target (π_i), that is,
$$\nu_i = \pi_a + w(\pi_b - \pi_c), \tag{1}$$
where $a,\ b,\ c$ are randomly chosen three distinct numbers from $1, \ldots, N$, and they are all different from i.

3. **Crossover**: Crossover blends the current generation of agents with the population of potential donors in order to form candidates for the next generation known as trial agents. For each $i = 1, \ldots, N$, one of the m variables of ν_i is randomly selected to directly enter the trial agent u_i. In this way, one variable is forced to change so that each u_i will certainly differ from its original target π_i. Next, with probability pCR, more variables are taken from u_i and placed in the trial agent. Whichever variables do not take their value from the donor inherit their original value from π_i. Assuming j_0 is a random number from $1, \ldots, m$, this process can be written as follows: for $j = 1, \ldots, m$,
$$u_{ij} = \begin{cases} \nu_{ij} & \text{with probability } pCR \text{ or if } j = j_0, \\ \pi_{ij} & \text{otherwise} \end{cases}$$

4. **Selection**: Selection creates the next generation of agents by comparing each target to its respective trial agent. The trial agent is adopted if it leads to an improvement and is discarded otherwise. For minimization problems, this process is given by,
$$\pi_i = \begin{cases} u_i & \text{if } \phi(u_i) < \phi(\pi_i) \\ \pi_i & \text{otherwise} \end{cases}$$

5. **Repeat**: Repeat steps 2 through 4 over many generations until a specified stopping condition is satisfied.

Though quite simplistic, its ability to balance exploration and exploitation makes it ideal for solving non-linear and multimodal problems.

Since experimental designs lie on a discrete and constrained space, we leverage the modified DE by Stokes, Wong, and Xu (2024). In the construction of UPD, each agent π_i is an $n \times m$ design matrix where each variable π_{ij} represents a column of length n. To ensure that the resulting design is feasible, Stokes, Wong, and Xu (2024) modified the mutation step by randomly selecting an agent and performing swap mutations. Inspired by particle swarm optimization (Chen et al. 2013), Stokes, Wong, and Xu (2024) incorporated the influence of both the global best solution and the personal best solution. In the context of DE, the personal best solution refers to the current agent itself. To be specific, let π_g be the global best agent where $g = \arg\min_i \phi(\pi_i)$. Stokes, Wong, and Xu (2024) modified the mutation step as follows. For each $i = 1, \ldots, N$,

(i) randomly choose an agent as a potential donor ν_i:

$$\nu_i = \begin{cases} \pi_g & \text{with probability } pGBest, \\ \pi_i & \text{with probability } pSelf, \text{ where } pSelf \leq 1 - pGBest, \\ \pi_r & \text{otherwise, where } r \text{ is a random number from } 1, \ldots, N. \end{cases}$$

(ii) randomly swaps two elements in column ν_{ij} with probability $pMut$ for $j = 1, \ldots, m$, independently.

This modification maintains the feasibility of the solution by preserving its design structure while introducing variation to explore the search space. The mutation step is controlled by the chance of choosing the global best ($pGBest$) or the agent itself ($pSelf$) and the mutation rate $pMut$, which is the probability of swapping two elements in a column. The swap mutation is computationally efficient and effective at diversifying the population, reducing the risk of premature convergence. On the other hand, crossover involves exchanging one column from the potential donor with a column from the agent. This is controlled by the probability of crossover. This resulted in an algorithm characterized by the following hyperparameters:

- NP - The size of the population (N),

- $itermax$ - the maximum number of iterations/generations used,
- pCR - Probability of crossover,
- $pMut$ - Probability of mutation,
- $pGBest$ - Probability of using the global best for mutation,
- $pSelf$ - Probability of using the current agent for mutation.

Stokes, Wong, and Xu (2024) proposed three different choices for selecting an agent to be mutated in order to obtain the potential donor, leading to three distinct variants. Specifically, DE1 utilizes the global best agent, DE2 employs the current agent, and DE3 selects a random agent. Additionally, they introduced a hybrid version, DE4, which combines all three strategies, selecting the global best, current agent, and random agent with probabilities of 50%, 25%, and 25%, respectively. Their findings demonstrated that DE1 and DE4 outperform DE2 and DE3.

Regarding the hypermarameters, the first two, NP and $itermax$, determine the budget size, whereas the other four hyperparameters affect the evolution process. The hyperparameters, $pGBest$ and $pSelf$, determine the probability of using the global best and the current agent in the mutation process, respectively. There is a constraint between these two hyperparameters, that is, $pGBest + pSelf \leq 1$. The question that arises is how these six hyperparameters interact with each other. Also whether we can do better than the proposed fixed probabilities (Stokes, Wong, and Xu 2024) to obtain better settings for the DE algorithm for design generation.

In this study, we shall consider values between $[10, 100]$ for NP, $[500, 1500]$ for $itermax$, and $[0, 1]$ for pCR, $pMut$ and $pGBest$. We fix $pSelf = (1 - pGBest)/2$ so that there is an equal chance for selecting the agent itself or a random agent as the donor when the global best is not selected.

3 Experimental Designs for Hyperparameter Settings

Different design strategies address diverse experimental objectives and constraints. For instance, factorial and fractional factorial designs are widely used to study the main effects and interactions of factors systematically, while response surface designs, such as central composite designs, support optimization and curvature estimation (Myers, Montgomery,

and Anderson-Cook 2016). In other cases, space-filling designs such as Latin hypercube designs, maximin distance designs, and maximum projection designs are preferred for high-dimensional and computational experiments (Joseph 2016). Each design has its own unique strengths and drawbacks. The choice would depend on the experimental goals, computational resources, and the nature of the underlying system being studied. Through thoughtful design construction, researchers can ensure that their experiments are not only scientifically rigorous but also cost-effective and impactful.

Various designs can be used to set the DE hyperparameters before the optimization process. As the functions optimized by DE are often complex with many local minima, one has to carefully choose the initial points for the optimization process. These initial points can be determined using any of the methods discussed below.

3.1 Factorial designs

Factorial designs are a research method for studying the effects of multiple independent variables on a response variable, formalized by R. A. Fisher in the early 20th century (Fisher 1935). Full factorial designs consist of a grid of all possible level combinations to achieve uniform coverage across the design space. These designs typically involve factors at two or three levels, resulting in 2^m or 3^m observations for m factors. They allow for the analysis of main effects and interactions (Montgomery 2017), but can require larger sample sizes for adequate power (Tabachnick and Fidell 2019).

Fractional factorial designs were introduced by Finney (1945) to mitigate the need for larger samples. These designs use a fraction of runs based on the effect hierarchy ordering principle (Wu and Hamada 2011), focusing on main effects and lower-order interactions. They are expressed as 2^{m-p} or 3^{m-p}, and defined by setting some factors as products of others. Selecting defining relations for fractional designs is essential, with criteria such as maximum resolution and minimum aberration guiding this process (Wu and Hamada 2011).

Central composite designs (CCDs) were introduced by Box and Wilson (1951) as an extension of factorial designs. They were developed as a way to efficiently fit quadratic response surfaces and identify optimal process settings in industrial experiments. CCDs are full or fractional factorial designs that are augmented with two additional sets of sampling points described as “center” and “axial or star” points (Box and Wilson 1951). The center point is defined by all factors being set at their center level. The

CCD uses $2m$ axial points, each of which is defined by all but one factor being at their center level and the level of the remaining factor is denoted by α, which is generally chosen to be between 1 and $\sqrt{m}$ (Montgomery 2017).

Orthogonal array composite designs (OACDs), introduced by Xu, Jaynes, and Ding (2014), are a class of composite designs based on a two-level factorial design and a three-level orthogonal array (OA). An OA of n runs, m columns, s levels, and strength t, denoted by $\mathrm{OA}(n, s^m, t)$, is an $n \times m$ matrix in which all s^t level-combinations appear equally often in every $n \times t$ submatrix (Wu and Hamada 2011). For example, a 2^m factorial design can be viewed as $\mathrm{OA}(n, 2^m, t)$ with $n = 2^m$ and $t = m$. Similarly, a three-level OA can be written as $\mathrm{OA}(n, 3^m, t)$. Thus, an OACD is a composite design which consists of a two-level factorial design as its factorial points, a three-level OA as its additional points, plus any number of center points (Luna et al. 2022; Zhou and Xu 2017).

3.2 Space-filling designs

While the previously discussed designs utilize sampling points that are at the boundaries of the design space, space-filling designs generate samples that are dispersed throughout the multidimensional design space and not just at the boundary of the design space. These designs are important in sampling a surface as they could capture important regions thereby minimizing the bias between the true structure of the surface and the estimated surface from the sampled points. There are of various types depending on the approach used to construct them, e.g., sampling-based – Latin Hypercube designs, and distance-based – maximin distance designs.

Latin hypercube designs (LHDs) – Based on McKay (1992)'s Latin hypercube sampling, it divides the range of each factor into bins of equal size, where n also corresponds to the number of samples to be generated resulting in a total of n^m combinations where m is the number of factors being considered. The n samples are then randomly generated such that for all one-dimensional projections, there will be only one sample in each bin.

Maximin distance designs – Introduced by Johnson, Moore, and Ylvisaker (1990), this design aims at spreading out the design points in the design space by maximizing the minimum distance between any two design points. It thus tends to place a large

proportion of points at the corners and on the boundaries of the design space. Mathematically, this can be formulated as follows. Suppose we want to construct an n-run design in m factors. Let the design region be $\mathcal{X}$ and let the design be $D = \{\boldsymbol{x}_1 \ldots, \boldsymbol{x}_n\}$, where each design point $\boldsymbol{x}_i$ is in $\mathcal{X}$. The maximin distance design optimizes the function below:

$$\max_{D} \min_{i \neq j} d(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

where $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \{\sum_{k=1}^{m}(x_{ik} - x_{jk})^2\}^{1/2}$ is the Euclidean distance between the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Maximin LHDs – Unlike LHDs, maximin distance designs do not have good projection properties for each factor. Morris and Mitchell (1995) proposed to overcome this problem by searching for the maximin distance design within the class of LHDs. They also proposed to use the following $\phi_p(D)$ criterion to achieve the maximin distance:

$$\min_{D} \phi_p(D) = \left\{ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{d^p(\boldsymbol{x}_i, \boldsymbol{x}_j)} \right\}^{1/p} \tag{2}$$

where $p > 0$ is chosen large enough, say $p = 15$.

Maximum projection (MaxPro) designs – Although maximin LHDs ensure good space-filling in m dimensions and uniform projections in each dimension, their projection properties in two to $m - 1$ dimensions are not known. By the effect sparsity principle (Wu and Hamada 2011), only a few factors are expected to be important. To curb this, Joseph, Gul, and Ba (2015) proposed a different criterion:

$$\min_{D} \psi(D) = \left\{ \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{\prod_{k=1}^{m} (x_{ik} - x_{jk})^2} \right\}^{1/m}. \tag{3}$$

They showed that the design minimizing $\psi(D)$ tends to maximize its projection capability in all subspaces of factors, and thus named these designs as maximum projection designs.

4 Surrogate Models

We consider three types of surrogate models to describe the response surface of the DE hyperparameters, namely, linear model, kriging model and heterogeneous Gaussian Process (HetGP). We describe each briefly in the following.

4.1 Linear Model (lm)

Linear models are widely used to model the relationship between the response variable and the inputs (i.e., factors). Here we adopt a second-order model which is commonly used to model data from physcical experiments. For m quantitative factors, denoted by $x_1, \ldots, x_m$, a second-order model is defined as

$$y = \beta_0 + \sum_{i=1}^{m} \beta_i x_i + \sum_{i=1}^{m} \beta_{ii} x_i^2 + \sum_{i<j} \beta_{ij} x_i x_j + \epsilon \tag{4}$$

where $\beta_0, \beta_i, \beta_{ii}$, and β_{ij} are the intercept, linear, quadratic, and bilinear (or interaction) terms, respectively, and ϵ is the error term. This model is simple and provides a straightforward way to model and understand relationships between the response variable and the factors.

4.2 Kriging Model (km)

Proposed by South African geostatistician Krige (1951), kriging is one of the methods used to interpolate intermediate values, whereby these intermediate values are modeled using Gaussian Process (GP) which is governed by prior co-variances. It provides a probabilistic prediction of the output variable, as well as an estimate of the uncertainty of the prediction (Chevalier, Picheny, and Ginsbourger 2014). The kriging predictors interpolating the observations are assumed to be noise-free (Roustant, Ginsbourger, and Deville 2012). Intermediate interpolated values obtained by kriging are the best linear unbiased predictors.

Kriging models are widely used to model deterministic functions in computer experiments. The kriging model consists of two parts: a trend and a GP. The trend part is often modeled as a regression on some fixed basis functions. In the specific case where the basis functions reduce to a constant function, it is referred to as ordinary kriging (Roustant, Ginsbourger, and Deville 2012). The general form is as given below

$$Y(\boldsymbol{x}) = \sum_{i=1}^{k} \beta_i f_i(\boldsymbol{x}) + Z(\boldsymbol{x}), \tag{5}$$

where $f_1, \ldots, f_k$ are k basis functions, $\beta_1, \ldots, \beta_k$ are corresponding regression coefficients, and $Z(\boldsymbol{x})$ is a stationary GP with zero mean and covariance function ψ. The covariance function ψ completely defines the behavior of the GP $Z(\boldsymbol{x})$. It is defined as

$$\psi\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \operatorname{Cov}\left(Z\left(\boldsymbol{x}_i\right), Z\left(\boldsymbol{x}_j\right)\right) = \sigma^2 \prod_{l=1}^{m} K\left(h_l; \theta_l\right), \tag{6}$$

where σ^2 is the scale parameter called the process variance, $h_l = |x_{i,l} - x_{j,l}|$, $x_{i,l}$ and $x_{j,l}$ are the lth elements of the i^{th} run $\boldsymbol{x}_i$ and the j^{th} run $\boldsymbol{x}_j$, and $K(h;\theta)$ is the correlation function. The parameters θ_l chosen for the correlation function $K(h_l;\theta_l)$ must be positive. Otherwise the correlation function will not be feasible. These parameters are chosen to be physically interpretable in the same unit as the corresponding variables. They are often referred to as the *characteristic length-scales* by Rasmussen and Williams (2006). Popular correlation functions include Gaussian, Matérn, and power-exponential family correlation functions. The Matérn function with parameter $\nu = 5/2$ is often chosen as the default when fitting kriging models. It is defined as:

$$K(h;\theta) = \left(1 + \frac{\sqrt{5}h}{\theta} + \frac{5h^2}{3\theta^2}\right) \exp\left(-\frac{\sqrt{5}h}{\theta}\right). \tag{7}$$

The unknown parameters can be estimated via MLE or cross validation. We use the `km` function in the DiceKriging package in R (Roustant, Ginsbourger, and Deville 2012) to fit the kriging model.

4.3 Heteroskedastic Gaussian Process (HetGP)

HetGP is useful for modeling simulation experiments exhibiting input-dependent noise. It employs a mean zero GP and shifts all of the modeling effort to the covariance structure (Binois, Gramacy, and Ludkovski 2018). Specifically, the HetGP model is given by

$$y_i = y(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, r(\boldsymbol{x}_i)), \tag{8}$$

where $f(\boldsymbol{x}_i)$ is a GP with covariance or kernel $k(\cdot,\cdot)$ and $r(\boldsymbol{x}_i)$ is the variance of ϵ_i which depends on $\boldsymbol{x}_i$. The kernel $k(\cdot,\cdot)$ is positive definite, with parameterized families such as the Gaussian or Matérn being typical choices. If $r(\boldsymbol{x}_i) = \tau^2$ is a constant, then the process is homoskedastic. In matrix notation, the modeling framework just described is equivalent to writing

$$Y(\boldsymbol{x}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_n + \boldsymbol{\Sigma}_n),$$

where $\mathbf{K}_n$ is the $n \times n$ matrix with (i,j) coordinate $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and $\boldsymbol{\Sigma}_n = \text{Diag}(r(\boldsymbol{x}_1), \ldots, r(\boldsymbol{x}_n))$ is the variance matrix of the vector of independent noise ε_i.

Given the kernel function $k(\cdot,\cdot)$ and data $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$, multivariate normal conditional identities provide a predictive distribution at site $\boldsymbol{x}$: $Y(\boldsymbol{x}) \mid \boldsymbol{y}$, which is Gaussian

with parameters

$$\mu(\boldsymbol{x}) = \mathbb{E}(Y(\boldsymbol{x}) \mid \boldsymbol{y}) = \mathbf{k}(\boldsymbol{x})^\top \left(\mathbf{K}_n + \boldsymbol{\Sigma}_n\right)^{-1} \boldsymbol{y},$$
$$\sigma^2(\boldsymbol{x}) = \mathbb{V}\,\mathrm{ar}(Y(\boldsymbol{x}) \mid \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{x}) + r(\boldsymbol{x}) - \mathbf{k}(\boldsymbol{x})^\top \left(\mathbf{K}_n + \boldsymbol{\Sigma}_n\right)^{-1} \mathbf{k}(\boldsymbol{x}),$$

where $\mathbf{k}(\boldsymbol{x}) = \left(k\left(\boldsymbol{x}, \boldsymbol{x}_1\right), \ldots, k\left(\boldsymbol{x}, \boldsymbol{x}_n\right)\right)^\top$. We use the `mleHetGP` function in the hetGP package in R (Binois and Gramacy 2021) to fit this model with the default Gaussian kernel.

5 The Data Generation Process

We aim to obtain optimal DE hyperparameter settings that can be used to generate UPDs.

5.1 Objective Function: Uniform Projection Criterion

Proposed by Sun, Wang, and Xu (2019), the uniform projection criterion solely focuses on two-dimensional projections. This is due to two factor interactions being more important than three-factor or higher-order interactions. The motivating idea was that although designs with low discrepancy have good uniformity in the full-dimensional space, they can have bad projections in lower dimensional spaces, which is undesirable when only a few factors are active. Thus designs with better projection properties are preferred. Sun, Wang, and Xu (2019) argued that UPDs scatter points uniformly in all dimensions and have good space-filling properties in terms of distance, uniformity and orthogonality.

The uniform projection criterion is defined using the centered L_2-discrepancy. For an $n \times m$ design $D = (x_{ik})$ with s levels from $\{0, 1, \ldots, s-1\}$, its (squared) centered L_2-discrepancy is defined as

$$\begin{aligned}\mathrm{CD}(D) =& \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \prod_{k=1}^{m} \left(1 + \frac{1}{2}\left|z_{ik}\right| + \frac{1}{2}\left|z_{jk}\right| - \frac{1}{2}\left|z_{ik} - z_{jk}\right|\right) \\ & - \frac{2}{n} \sum_{i=1}^{n} \prod_{k=1}^{m} \left(1 + \frac{1}{2}\left|z_{ik}\right| - \frac{1}{2}\left|z_{ik}\right|^2\right) + \left(\frac{13}{12}\right)^m,\end{aligned}$$

where $z_{ik} = \left(2x_{ik} - s + 1\right)/(2s)$. Then the uniform projection criterion is to minimize

$$\phi(D) = \frac{2}{m(m-1)} \sum_{|u|=2} \mathrm{CD}\left(D_u\right), \tag{9}$$

where u is a subset of $\{1, 2, \ldots, m\}$, $|u|$ denotes the cardinality of u and D_u is the projected design of D onto dimensions indexed by the elements of u. The $\phi(D)$ value is the average centered L_2-discrepancy values of all two-dimensional projections of D.

The uniform projection criterion defined in (9) requires $O(n^2m^2)$ operations. Sun, Wang, and Xu (2019) derived an alternative formula for computing the $\phi(D)$ value which requires only $O(n^2m)$ operations. They showed that $\phi(D)$ is a function of the pairwise L_1-distances between design points for balanced designs (i.e., each level appears equally often in any factor). Specifically, for a balanced design $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ with n runs, m factors, and s levels, they showed

$$\phi(D) = \frac{g(D)}{4m(m-1)n^2s^2} + C(m,s), \tag{10}$$

where

$$g(D) = \sum_{i=1}^{n}\sum_{j=1}^{n} d_1^2(\boldsymbol{x}_i, \boldsymbol{x}_j) - \frac{2}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{n} d_1(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)^2, \tag{11}$$

$d_1(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{m} |x_{ik} - x_{jk}|$ is the L_1-distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and

$$C(m,s) = \frac{4(5m-2)s^4 + 30(3m-5)s^2 + 15m + 33}{720(m-1)s^4} + \frac{1+(-1)^s}{64s^4}.$$

We compute the $\phi(D)$ value based on (10) and (11) for its computational efficiency. Because the $\phi(D)$ values are typically small, to ease the presentation, we report the $1000 \times \phi(D)$ values throughout the paper.

We implement the DE algorithm and the uniform projection criterion in the package `UniPro`. The following code generates an $n \times m$ UPD with s levels

```
> UniPro(n, m, s, NP, itermax, pMut, pCR, pGBest, seed)
```

where NP, $itermax$, $pMut$, pCR and $pGBest$ are DE hyperparameters described in Section 2, and *seed* is an optional seed for random number generators that ensures reproducibility.

As the task of design generation is quite complex, we consider 3 design sizes. A UPD of size 30×3 is considered as a small and easy task, 50×5 as a medium task and 70×7 as a large and difficult task. We only consider the construction of designs with $s = n$ so that the resulting UPD is an LHD.

5.2 Training and testing data

Designs discussed in Section 3 are used to determine the parameter settings for the DE algorithm hyperparameters. Specifically, we construct five designs: a CCD with 43 runs (`ccd3_43`), an OACD with 50 runs (`oacd3_50`), a 50-run random LHD (`lhd_50`), a 50-run maximin LHD (`maximin_50`), and a 50-run maxpro LHD (`maxpro_50`). The CCD consists of

a 2^5 full factorial, 10 axial points and 1 center point. The OACD consists of a 2^5 full factorial and an OA$(18, 3^5, 2)$. The maximin and maxpro LHDs are generated using R packages SLHD and MaxPro, respectively. All designs consider five factors, each corresponding to one hyperparameter. Each run represents a unique combination of the hyperparameter values. The CCD and OACD have 3 levels while the rest have 50 levels. For the CCD, we choose $\alpha = 1$ so that it has 3 levels because we have a cuboidal experimental region. The levels are linearly interpolated within the minimum and maximum factor values for each hyperparameter. To avoid the influence of extreme cases in model fitting and prediction, we restrict pCR, $pMut$ and $pGBest$ to $[0.05, 0.95]$. We use the full range $[0, 1]$ for each hyperparameter in a later optimization stage.

Given the target design size ($n \times m$) and a setting of the five hyperparameters, we run the `UniPro` function to generate an $n \times m$ UPD and the resulting $\phi(D)$ value defined in (9). To account for the stochastic nature of the DE algorithm, for each setting, the algorithm is replicated ten times, producing ten $\phi(D)$ values. These $\phi(D)$ values are then averaged to obtain the observed response value for the setting, and the standard deviation is also computed to quantify variability. Thus we obtain five training datasets for each target size.

The same procedure is taken to generate the testing datasets with the exception that the designs used for the hyperparameter setting combinations being a 3^5 full factorial design (`full_243`), a random LHD with 243 runs (`lhd_243`), a combination of these two (`full_243+lhd_243`), and a 4^5 full factorial design (`full_1024`). Testing with these datasets is deemed sufficient because the random LHD exhibits maximum space-filling properties in each dimension, while the 3^5 and 4^5 factorial designs ensure uniform coverage of the entire 5-dimensional input space.

Density plots of the response for the testing and training data are presented in Figure 1 for the target size 50×5. All of the distributions are skewed to the right. For a 50×5 UPD all the training designs lead to similar minimum $\phi(.)$ values, around 0.17, whereas different types of designs lead to different maximum $\phi(.)$ values. Indeed, all space-filling designs have maximum $\phi(.)$ values around 0.28, while the factorial designs and the combined design have a maximum $\phi(.)$ values around 0.34. The narrower range of the $\phi(.)$ values suggests that the space-filling designs do not explore the entire space of hyperparameters.

5.3 Model evaluation

For each training dataset, we fit the three models described in Section 4 and test on the four testing datasets. Designs are evaluated by considering their ability to collect informative

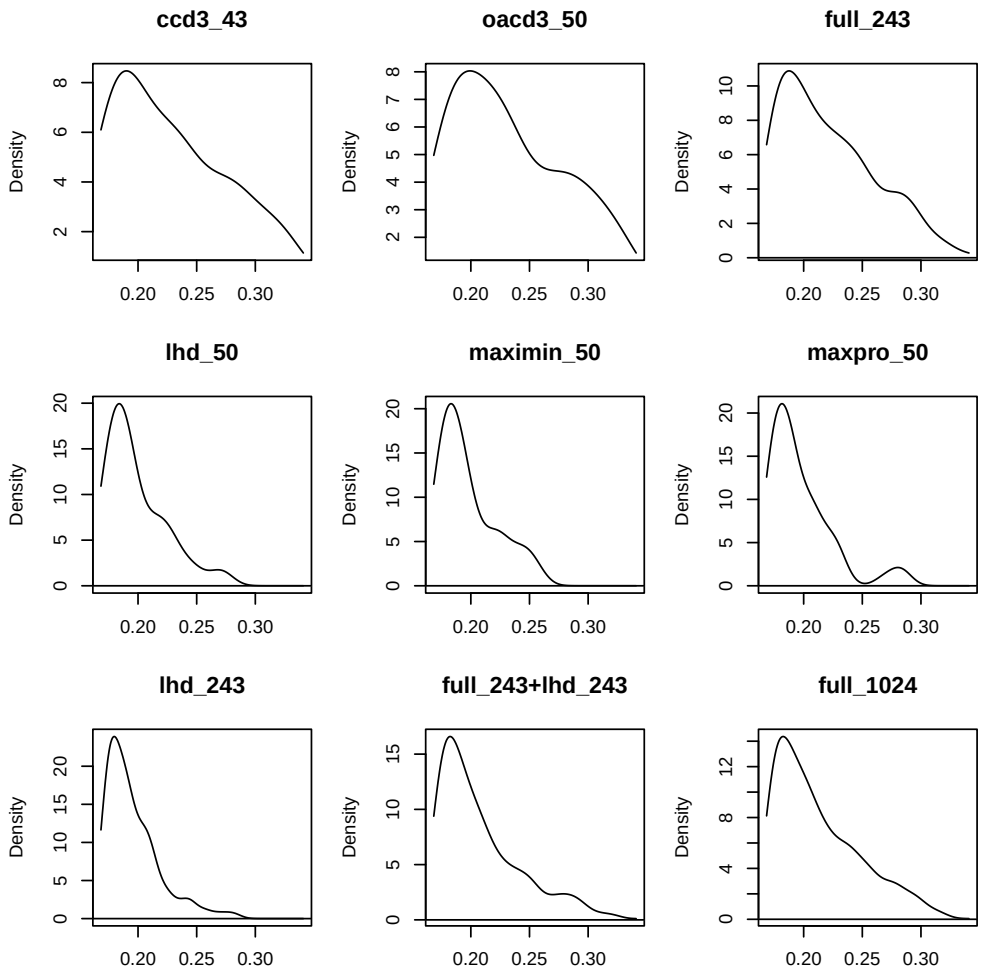


Figure 1: Density plots of the $\phi(D)$ values with target size 50×5

data for building a statistical model that specifies the relationship between the response and the hyperparameters, which is measured by the test root mean squared error (RMSE). For a testing dataset with N runs, the RMSE is calculated as

$$\text{RMSE} = \left\{ \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \right\}^{1/2},$$

where y_i and $\hat{y}_i$ are the ith observed and predicted response, respectively. We use four testing datasets with N varying from 243 to 1024 as described in Section 5.2. Since each y_i is an average of ten $\phi(D)$ values obtained by running the DE algorithm ten times, the RMSE value could be viewed as an average of ten replications. We use RMSE as the evaluation metric because it directly measures the deviation between the predicted and observed response values. Although the objective function value $\phi(D)$ is central to the optimization process, it is not appropriate for evaluating predictive accuracy. In this setting, the goal here is not optimization but prediction — specifically, how well the model can generalize to unseen hyperparameter configurations. RMSE provides a natural and interpretable measure of this predictive accuracy. The correlation between the observed and predicted responses is also reported. A small RMSE value indicates a good fit while a large correlation is preferable.

6 Results and Analysis

Table 1 and Figure 2 compare designs and model evaluations based on correlation and RMSEs for a target size of 30×3. We begin by analyzing the results for testing the two 243-run datasets, as shown in Table 1(a)(b) and Figure 2(a)(b). One striking observation is that the performance of the training data set depends on the nature of the testing data set. The composite designs, CCD and OACD, seem to be better when tested on the 3^5 full factorial design (FFD) while the space-filling designs (random LHD, maximin LHD, and maxpro LHD) do better when tested on the 243-run random LHD. Figure 2(a)(b) appears to suggest that when the design structure of the training samples is similar to that of the testing samples, the predicted RMSE is better. As this does not give a general idea as to which designs might perform better in general, we invoke the 486-run combined design (3^5 FFD + 243-run LHD) and the 4^5 FFD as the testing dataset.

When tested on the 486-run combined design, the OACD performs slightly better than the space-filling designs although the difference appears to be small; see Table 1(c) and Figure 2(c). When tested on the 4^5 FFD, the OACD performs better than the space-filling

Table 1: Comparison of designs and model evaluations with target size 30×3

	(a) Testing on the 3^5 FFD						**(b) Testing on the 243 LHD**					
	correlation			RMSE			correlation			RMSE		
Design	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP
ccd3_43	0.88	0.88	0.88	1.51	1.59	1.62	0.65	0.03	0.60	1.62	3.75	1.51
oacd3_50	0.88	0.94	0.93	1.62	1.25	1.32	0.68	0.63	0.61	1.94	2.30	2.22
lhd_50	0.41	0.36	0.34	3.19	3.30	3.27	0.28	0.20	0.19	1.08	1.14	1.14
maximin_50	0.71	0.73	0.74	2.86	3.19	3.03	0.64	0.63	0.64	0.66	0.66	0.65
maxpro_50	0.71	0.62	0.66	2.35	2.97	2.72	0.69	0.71	0.74	0.74	0.69	0.61
	(c) Testing on the 3^5 FFD+243 LHD						**(d) Testing on the 4^5 FFD**					
	correlation			RMSE			correlation			RMSE		
Design	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP
ccd3_43	0.84	0.60	0.83	1.57	2.88	1.57	0.84	0.66	0.85	1.55	2.83	1.50
oacd3_50	0.82	0.83	0.83	1.79	1.85	1.83	0.85	0.87	0.87	1.68	1.68	1.68
lhd_50	0.42	0.35	0.36	2.38	2.47	2.45	0.45	0.39	0.37	2.47	2.57	2.56
maximin_50	0.69	0.63	0.68	2.08	2.31	2.19	0.74	0.75	0.76	2.16	2.38	2.27
maxpro_50	0.74	0.60	0.68	1.75	2.15	1.98	0.74	0.67	0.71	1.80	2.22	2.03

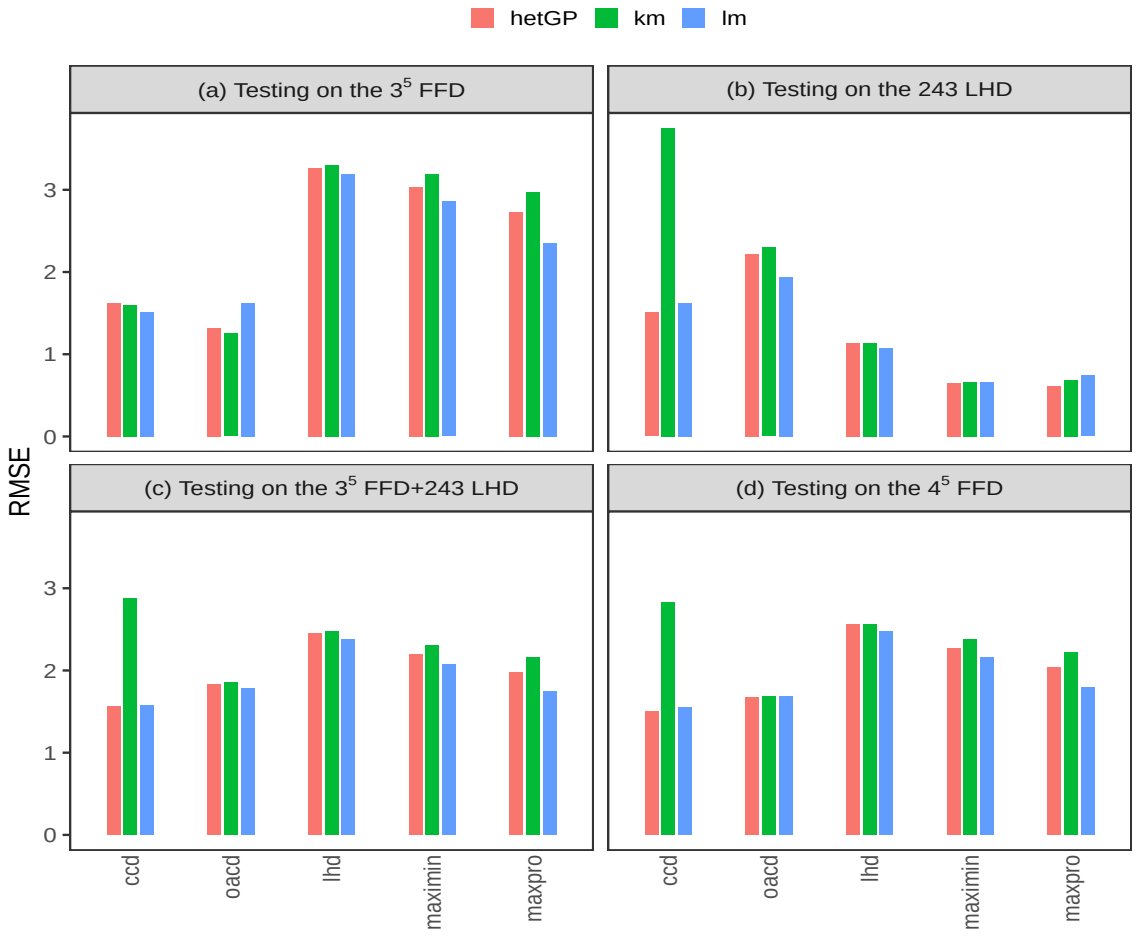


Figure 2: Comparison of RMSE with target size 30×3

designs as it has a more similar structure to the 4^5 FFD; see Table 1(d) and Figure 2(d). The CCD does not perform well with the kriging model. The 50-run random LHD performed the worst in terms of correlation regardless of the testing data. The correlation is strikingly low whereas the RMSE is high. This might be due to randomness, but it does show the weakness of the random LHD.

One bizarre observation from Table 1(b) is the correlation of 0.03 when using the CCD as the training design and testing it on the 243-run random LHD. This value is strikingly lower than any other values given in Table 1. No apparent reason could be deduced as to why this is so. Multiple replications indicated that this is not an error. From all the results, we can deduce the robustness of OACD over CCD. This gives a reason to use OACD for the hyperparameter initialization.

With regards to the models, there seems to be no striking observation to be made as to whether one fitting method performs better than the other two, with exception for one 30×3 case when the kriging model fitting to the CCD training data had a much higher RMSE value than the other cases.

The three models have quite different assumptions. The linear model assumes a polynomial trend and independent random errors with homoskedastic variance. The kriging model assumes a stationary covariance structure, that is, the covariance between two points in the DE hyperparameter surface structure depends only on the distance or spatial lag between those points, and not on their specific locations within the domain. The HetGP model assumes a heteroskedastic variance-covariance structure. For the DE algorithm, the homoskedastic and stationary assumptions are questionable. Due to this, the HetGP model is preferred to the linear model and the kriging model. However, the linear model is easy to interpret and fits as well as the HetGP model, and from the results, there is no striking difference between the two. Therefore, we use the linear model to determine factor importance and optimal hyperparameter settings.

Tables 6-7 and Figures 7-8 in the Appendix show results when the target design sizes are 50×5 and 70×7, respectively. Looking at the results, apart from the random LHD with 50 runs, previously stated observations are upheld.

A natural question is why composite designs perform better than the space-filling designs. We perceive that the hyperparameters at the boundaries lead to some extreme cases in this experiment and the composite designs do capture this phenomena while the space-filling designs do not. Figure 3 presents the histograms of the distances from design points to the design center for all the designs, where each column is rescaled to $[-1, 1]$ and the Euclidean

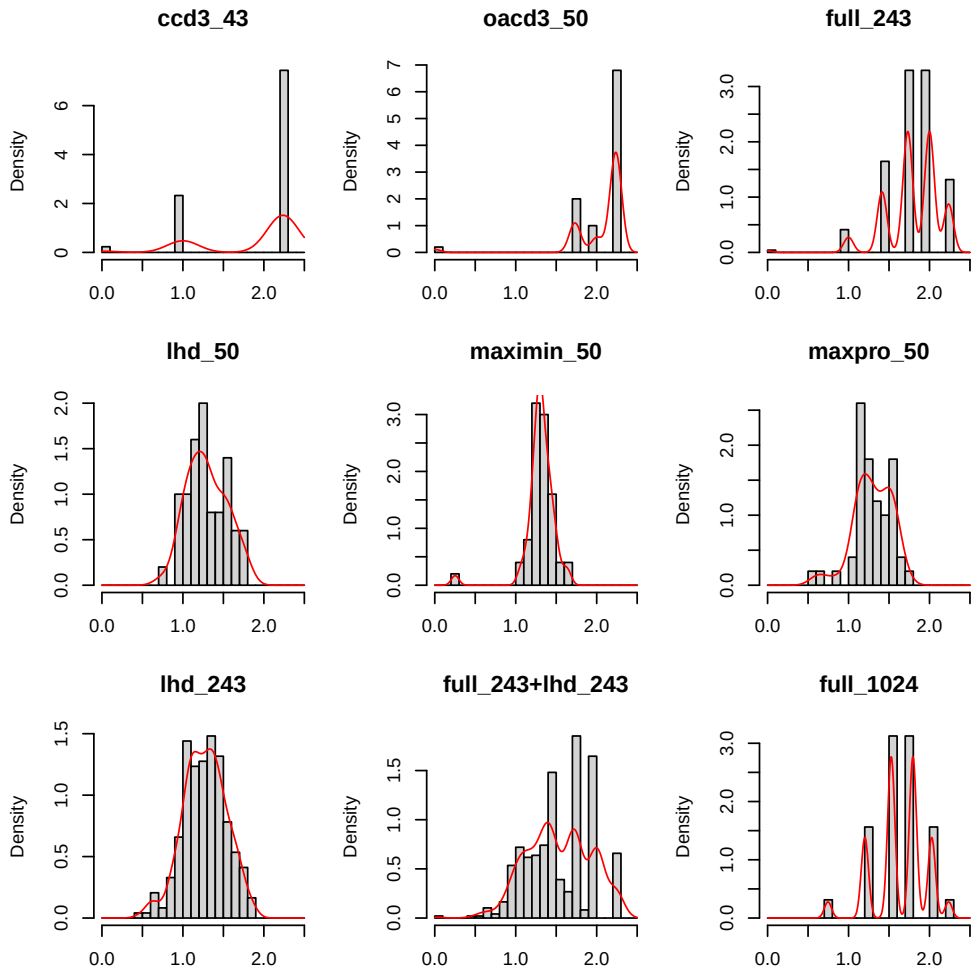


Figure 3: Histogram of the distances from design points to the design center

distance from each point to the center of the design is calculated. This gives an insight as to why the composite designs, OACD and CCD, tend to perform better than the space-filling designs. This is because the composite designs tend to capture information lying at the boundaries compared to the space-filling designs which tend to capture the information lying at the center of the design. This is confirmed by the notion that three of the hyperparameters tend to be optimized around their highest level as discussed in the next section.

7 Factor Importance and Optimal Settings

The results obtained call for a deeper look into the model and how each factor is involved in the surface approximation. This enables us to have a better picture of the surface generated

by the DE hyperparameters.

Table 2 shows the estimated coefficients of the second-order models based on the four different training datasets: CCD, OACD, maximin LHD, and maxpro LHD, for the target size 30×3. Looking at the various models above, the model obtained using the maxpro LHD as the training data is the worst performing model. It has the lowest adjusted R^2 of only 0.54. This model does not capture important main effects. For example, it indicates that the number of iterations ($itermax$) for optimization is not significant. Yet this is well known to be important. On the other hand, the model obtained by using OACD as the training dataset performs the best. It has an adjusted R^2 of 0.87 and captures important main effects and interactions. In addition, CCD might be a little worse than OACD because of the fewer number of points (43) used for training compared to the other models which used 50 points. The model from the CCD does not identify any of the quadratic effects to be significant while the other models do. The maxpro LHD has a relatively small adjusted R^2 value because it prioritizes space-filling whereas the CCD and OACD emphasize the parameter estimations.

Looking at the corresponding p-values from the models obtained, it is evident that all five hyperparameters are important. The main effects of three hyperparameters (NP, $itermax$ and $pGBest$) are very significant, whereas the several interactions involving one of the other two hyperparameters ($pMut$ and pCR) are also very significant. The probability of using the global best ($pGBest$) is quite important because its main effect is highly significant in all the models in Table 2. It can also be inferred that the maximum number of iterations ($itermax$) and the population size (NP) are important. The linear models indicate that to minimize the objective function, it is ideal to use a larger $pGBest$, increase the population size (NP) and increase the number of iterations ($itermax$). The population size and the number of iterations could be constrained by the available budget.

With regards to interactions, all significant interaction terms involve either $pMut$ or $pGBest$. The interaction of $pMut$ and $pGBest$ is negative. On the other hand, the interaction between $pMut$ and pCR is positive. As this is a minimization problem, this positive interaction indicates that one of the variables should be set at low while the other high. We use contour plots to visualize the two parameters when the other three are held at their highest level.

Figure 4 shows the contour plots of $pMut$ and pCR based on the CCD and OACD training data set while fixing $NP = 100$, $itermax = 1500$ and $pGBest$=0.95. From the contour plot for target size 30×3, we note that the response value could be minimized by taking values along the off diagonal, either a smaller $pMut$ and larger pCR or a larger $pMut$

Table 2: Comparison of second-order models from four training datasets

	CCD	OACD	maximin	maxpro
(Intercept)	0.3815^{***}	0.3876^{***}	0.3878^{***}	0.3814^{***}
NP	-0.0134^{***}	-0.0143^{***}	-0.0042^{***}	-0.0055^{*}
pMut	0.0020	0.0028	0.0043^{***}	0.0045^{*}
pGBest	-0.0204^{***}	-0.0224^{***}	-0.0047^{***}	-0.0082^{***}
pCR	−0.0022	−0.0030	0.0004	0.0007
itermax	-0.0146^{***}	-0.0152^{***}	-0.0046^{***}	−0.0021
NP^2	0.0119	0.0080	0.0022	−0.0004
$pMut^2$	0.0150	0.0174^{*}	0.0083^{***}	0.0104^{*}
$pGBest^2$	0.0083	0.0192^{*}	0.0035	0.0159^{***}
pCR^2	0.0071	0.0074	−0.0008	0.0049
$itermax^2$	0.0090	−0.0081	0.0011	0.0057
NP:pMut	0.0058	0.0064^{*}	−0.0002	−0.0037
NP:pGBest	−0.0041	−0.0038	0.0017	0.0006
NP:pCR	0.0002	−0.0007	−0.0006	−0.0002
NP:itermax	0.0049	0.0033	0.0023	0.0052
pMut:pGBest	-0.0080^{*}	-0.0093^{***}	-0.0089^{***}	-0.0139^{**}
pMut:pCR	0.0203^{***}	0.0197^{***}	0.0042^{**}	0.0018
pMut:itermax	0.0032	0.0025	-0.0061^{***}	0.0004
pGBest:pCR	0.0034	0.0036	0.0000	0.0051
pGBest:itermax	0.0057	0.0068^{**}	0.0076^{***}	0.0058
pCR:itermax	0.0037	0.0021	0.0018	0.0040
R^2	0.9034	0.9235	0.8970	0.7297
Adj. R^2	0.8157	0.8708	0.8260	0.5433
Num. obs.	43	50	50	50

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

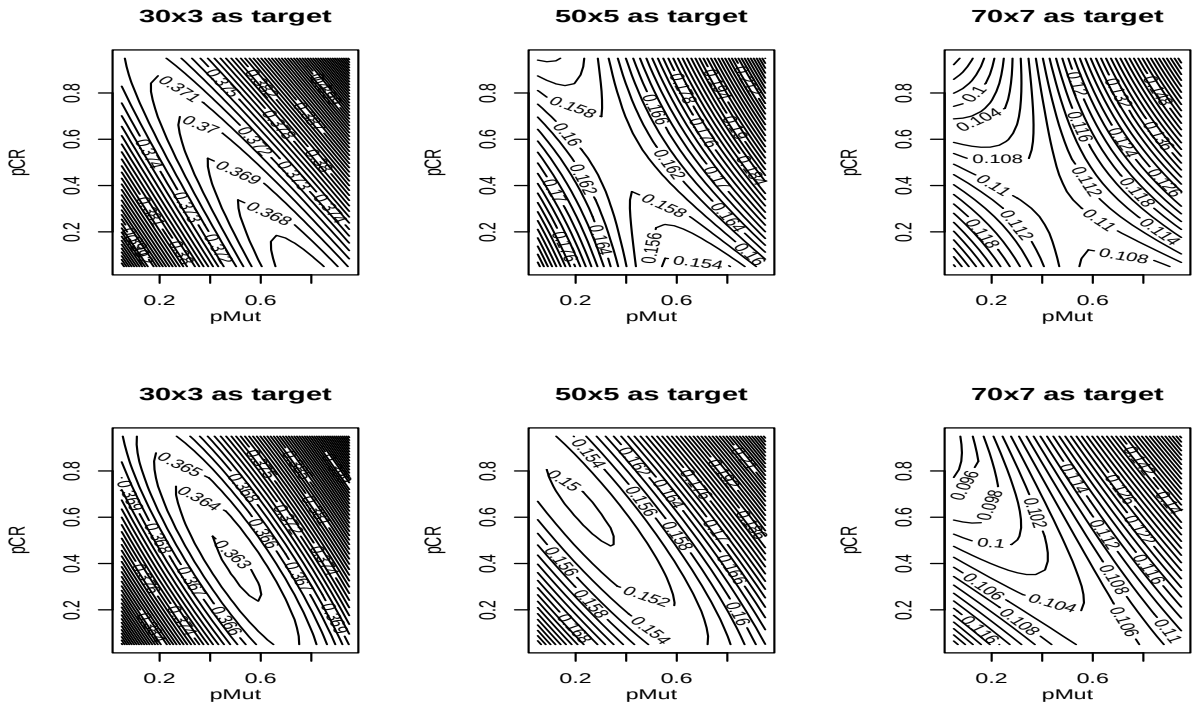


Figure 4: Contour plots of $pMut$ and pCR while fixing other hyperparameters at the highest levels. Top row uses CCD as the training data; bottom row uses OACD as training data.

Table 3: Optimal hyperparameter settings and DE performace via grid search

	NP	itermax	pMut	pCR	pGBest	Average $\phi(\cdot)$	SD $\phi(\cdot)$
30×3	100	1167	0.95	0.05	0.65	0.3863	0.0040
50×5	100	1500	0.35	0.95	0.95	0.1688	0.0016
70×7	100	1500	0.35	0.35	0.95	0.1063	0.0009

Table 4: Optimal hyperparameter settings and DE performace via random search

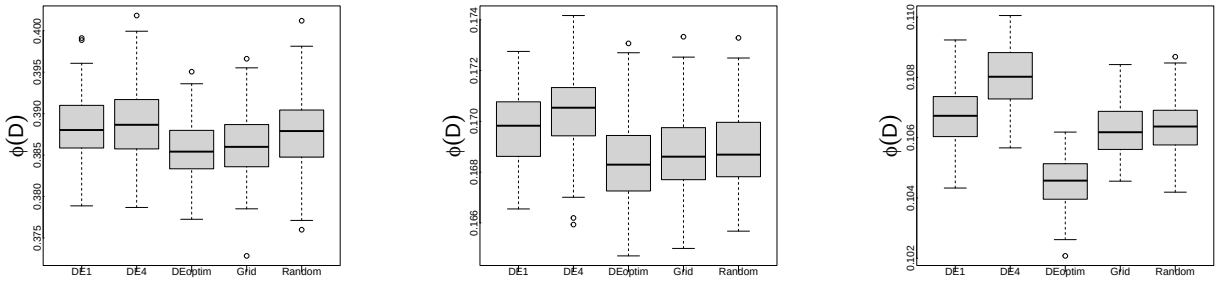
	NP	itermax	pMut	pCR	pGBest	Average $\phi(\cdot)$	SD $\phi(\cdot)$
30×3	95	917	0.63	0.19	0.72	0.3878	0.0046
50×5	98	1485	0.40	0.38	0.88	0.1688	0.0014
70×7	84	1408	0.11	0.92	0.86	0.1064	0.0009

and a smaller pCR. This is also the case for target sizes 50×5 and 70×7. It appears that different target sizes require different optimal settings for $pMut$ and pCR. Here, we take a simple approach to searching the optimal setting by varying $pMut$ from $\{0, 0.1, \ldots, 1.0\}$ while fixing $pCR = 1 - pMut$ and $pGBest = 1$. We look for the best combination of $pMut$ and pCR that gives the best objective value with a limited budget of 100 iterations for each combination. Then we use the best combination and run the DE algorithm with $itermax = 1500$ iterations to obtain the final objective value. We refer to this approach as DEoptim.

We compare the DEoptim approach with the two DE variants, DE1 and DE4, described by Stokes, Wong, and Xu (2024) and the grid and random searches. DE1 uses only the global best solution ($pGBest = 1$) while DE4 is a hybrid approach with $pGBest = 0.5$. Both DE1 and DE4 utilize fixed values of $pMut = 0.1$, $pCR = 0.5$, $NP = 100$ and $itermax = 1500$. On the other hand, for grid search, the best settings are chosen based on the 4^5 FFD while for the random search, a 1024-run random LHD is used instead. Tables 3 and 4 show the optimal hyperparameter settings and the corresponding performances from the grid search and the random search, respectively.

Figure 5 presents the performance of the DE algorithm under various hyperparameter settings, where the DE algorithm was executed 100 times to construct 100 UPDs for each setting and the vertical axis is the $\phi(D)$ values of the UPDs constructed. The results demonstrate that DEoptim produces significantly better UPDs, especially for the large target size

70×7, compared to the two variants, DE1 and DE4, and grid and random strategies, confirming the effectiveness of hyperparameter tuning.



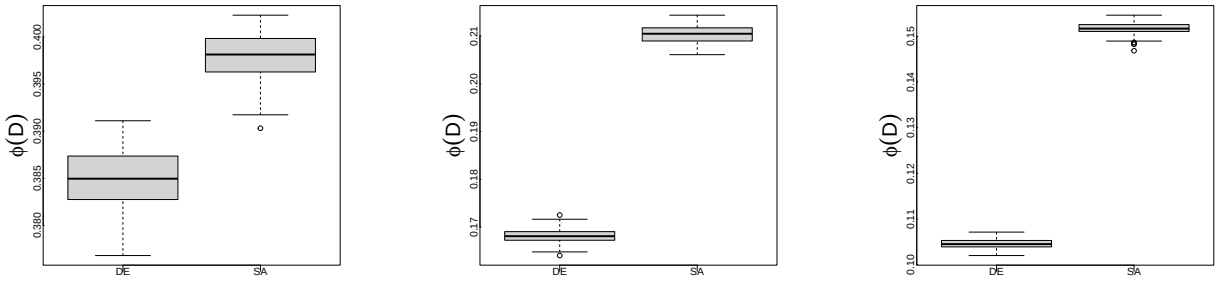
(a) 30×3 as target (b) 50×5 as target (c) 70×7 as target

Figure 5: Performance of the DE algorithm under various setttings: DE1, DE4, DEoptim, Grid, Random

To assess the effectiveness of DE in the UPD construction, we compare it against Simulated Annealing (SA), a classical optimization method commonly used for design construction. Unlike DE, which adapts its search through population-based mutation and crossover, SA relies on a fixed cooling schedule and probabilistic acceptance of worse solutions to escape local optima. We adopt the SA algorithm in R package `NMOF` with a neighbor function that swaps two elements in a randomly selected column. For a fair comparison, we set both the number of iterations and the number of temperatures as 1500 and repeat the SA algorithm 100 times to get the best design. With these settings, the SA and DE algorithms have the same complexity. The comparison results indicate that DE finds better designs with better objective values compared to SA. As shown in Figure 6, DE consistently outperforms SA across all three target sizes 30×3, 50×5, and 70×7 with significantly lower $\phi(D)$ values across 100 replicated runs. The performance gap becomes especially pronounced as the dimensionality increases, where SA tends to suffer from premature convergence and limited search efficiency. These results demonstrate that DE offers more reliable convergence and superior design quality compared to SA, particularly in high-dimensional and complex search spaces. We perceive that the population-based nature of DE enables it to efficiently explore the search space while avoiding stagnation in local minima, making it a more reliable and computationally efficient method.

The DE algorithm can be tuned to construct other types of space-filling designs following

the same strategy. In Appendix B, we compare DE and SA for constructing maximin LHDs and reach the same conclusion that DE is superior to SA.



(a) 30×3 as target (b) 50×5 as target (c) 70×7 as target

Figure 6: Comparison of the DE and SA algorithms for constructing UPDs

8 Conclusions

This paper compared small designs in exploring the response surface of the DE algorithm hyperparameters. Five numerical hyperparameters were considered: population size, the number of maximum iterations, probability of mutation, probability of crossover, and probability of using the global best design for mutation. Various composite designs and space-filling designs for selecting combinations of these hyperparameters were also examined. The performance of a design was evaluated via building a second-order model, a kriging model, and a heterogeneous GP model. The performance was measured in terms of testing RMSEs and correlation, and the comparison was made based on data simulated using the uniform projection criterion. Under the settings considered, the results demonstrate that the OACD is superior to space-filling designs for exploring the response surface of the DE algorithm hyperparameters. Furthermore, the second-order model proves to be a simple yet effective alternative to more complex models like kriging and heterogeneous GP in this context. The importance of tuning the DE algorithm is highlighted, and a simple strategy was proposed for determining optimal hyperparameter settings for constructing UPDs with different target sizes.

Regarding the optimal settings of the five hyperparameters, we recommend set the population size and the maximum number of iterations at high levels. A population size of

Table 5: Recommended $pMut$ setting

n \ m	**3**	**5**	**7**	**10**	**15**	**20**
15	1.0	1.0	1.0	0.4	0.1	0.1
20	1.0	1.0	0.6	0.1	0.1	0.1
30	1.0	0.4	0.2	0.1	0.1	0.1
50	0.9	0.3	0.2	0.1	0.1	0.1
70	0.7	0.3	0.2	0.1	0.1	0.1
100	0.4	0.2	0.2	0.1	0.1	0.1

100 is typically large enough based on our experience whereas the maximum number of iterations can be set to 1500 or higher for larger designs, depending on the budget. The choice of $pGBest = 1$ is recommended so that the global best is always chosen as the donor. The choices of $pMut$ and pCR need more attention as they interact with each other. The contour plots in Figure 4 shows that the best $\phi(D)$ values tend to concentrate along the diagonal in the graph of $pMut$ versus pCR. This diagonal corresponds to the region where $pMut + pCR \approx 1$, indicating a balanced trade-off between mutation and crossover.

To determine the optimal settings for $pMut$ and pCR, we performed a grid search of $pMut$ from $\{0, 0.1, \ldots, 1\}$ under the constraint $pMut + pCR = 1$, and ran the DE algorithm 100 times for each setting while fixing $NP = 100$, $itermax = 1500$ and $pGBest = 1$. Table 5 shows the recommended $pMut$ setting for a wide range of combinations of n and m.

The results show that when both the number of design points (n) and factors (m) are small, high mutation rates ($pMut = 1$) tend to yield the best solutions. In these lower-dimensional settings, aggressive mutation with minimal crossover appears effective, likely because the design space is less complex and the risk of disrupting valuable structure is minimal. The DE algorithm benefits from broad exploration, and excessive refinement is unnecessary. This is especially evident for $m = 3$ or 5 and $n = 15$ to 30, where the proportion of best results associated with $pMut = 1$ is consistently high.

However, as the dimensionality increases, particularly for $m \geq 10$ or $n \geq 50$, the advantage of using a high mutation rate quickly diminishes. In these settings, the best outcomes are rarely associated with $pMut = 1$ and instead favor lower values such as $pMut = 0.1$, indicating the growing importance of crossover in maintaining useful structure. High mutation introduces too much randomness, making it difficult for the algorithm to preserve

and build upon promising designs. These patterns support the recommendation that high mutation is suitable for small problems, but for more complex designs, a lower mutation rate combined with strong crossover ($pCR = 1 - pMut$) provides a more effective balance between exploration and exploitation. While we restricted $pMut + pCR = 1$, it may have some benefits by relaxing the constraint to $0.9 \leq pMut + pCR \leq 1.1$; however, the benefits are often small compared to the extra computational costs. It is aslo important to avoid $pMut = 0$, which prevents mutation and leads to degenerate behavior.

To summarize, we recommend the following settings for $pMut$ and pCR:

- For simple or low-dimensional problems ($m < 10$), use Table 5 to guide the choice of $pMut$ and set $pCR = 1 - pMut$.
- For high-dimensional problems ($10 \leq m \leq 20$), use a small mutation rate $pMut = 0.1$ and large crossover rate $pCR = 0.9$.
- For very high-dimensional problems ($m > 20$), use $pMut = 1/m$ and $pCR = 1$, which ensures strong crossover and one swap mutation on average.

Finally, we compared our tuned DE algorithm with the SA algorithm for constructing designs. The results confirm that DE is the preferred optimization method for generating UPDs. The DE framework provides a robust approach for constructing high-quality designs, making it a valuable tool for experimental design and response surface modeling. To facilitate accessibility and further research, an R package, UniPro, has been developed and is publicly available at https://github.com/oonyambu/UniPro.

References

Aarts, E. and J. Korst (1989). *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*. John Wiley & Sons, Inc.

Ba, S., W. R. Myers, and W. A. Brenneman (2015). "Optimal sliced Latin hypercube designs". In: *Technometrics* 57.4, pp. 479–487.

Binois, M. and R. B. Gramacy (2021). "hetgp: Heteroskedastic Gaussian process modeling and sequential design in R". In: *Journal of Statistical Software* 98, pp. 1–44.

Binois, M., R. B. Gramacy, and M. Ludkovski (2018). "Practical heteroscedastic gaussian process modeling for large simulation experiments". In: *Journal of Computational and Graphical Statistics* 27.4, pp. 808–821.

Box, G. E. P. and K. B. Wilson (1951). "On experimental attainment of optimum conditions". In: *Journal of the Royal Statistical Society, Series B* 13.1, pp. 1–45.

Chen, G. and B. Tang (2024). "Selecting strong orthogonal arrays by linear allowable level permutations". In: *Electronic Journal of Statistics* 18.2, pp. 3573–3589.

Chen, R.-B. et al. (2013). "Optimizing Latin hypercube designs by particle swarm". In: *Statistics and computing* 23, pp. 663–676.

Chevalier, C., V. Picheny, and D. Ginsbourger (2014). "KrigInv: An efficient and user-friendly implementation of batch-sequential inversion strategies based on kriging". In: *Computational statistics & data analysis* 71, pp. 1021–1034.

Das, S. and P. N. Suganthan (2010). "Differential evolution: A survey of the state-of-the-art". In: *IEEE transactions on evolutionary computation* 15.1, pp. 4–31.

Dorigo, M. (2007). "Ant colony optimization". In: *Scholarpedia* 2.3, p. 1461.

Falkner, S., A. Klein, and F. Hutter (2018). "BOHB: Robust and efficient hyperparameter optimization at scale". In: *International conference on machine learning*, pp. 1437–1446.

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

Fang, K.-T. et al. (2000). "Uniform design: theory and application". In: *Technometrics* 42.3, pp. 237–248.

Finney, D. J. (1945). "The Fractional Replication of Factorial Arrangements". In: *Annals of Eugenics* 12, pp. 291–301.

Fisher, R. A. (1935). *The Design of Experiments*. Oliver and Boyd.

Hutter, F., H. H. Hoos, and K. Leyton-Brown (2011). "Sequential model-based optimization for general algorithm configuration". In: *Learning and Intelligent Optimization: 5th International Conference*. Springer, pp. 507–523.

Johnson, M. E., L. M. Moore, and D. Ylvisaker (1990). "Minimax and maximin distance designs". In: *Journal of statistical planning and inference* 26.2, pp. 131–148.

Joseph, V. R. (2016). "Space-filling designs for computer experiments: A review". In: *Quality Engineering* 28.1, pp. 28–35.

Joseph, V. R., E. Gul, and S. Ba (2015). "Maximum projection designs for computer experiments". In: *Biometrika* 102.2, pp. 371–380.

Joseph, V. R. et al. (2019). "Space-filling designs for robustness experiments". In: *Technometrics* 61.1, pp. 24–37.

Kamath, C. (2022). "Intelligent sampling for surrogate modeling, hyperparameter optimization, and data analysis". In: *Machine Learning with Applications* 9, p. 100373.

Kirkpatrick, S., C. D. Gelatt Jr, and M. P. Vecchi (1983). "Optimization by simulated annealing". In: *Science* 220.4598, pp. 671–680.

Krige, D. G. (1951). "A statistical approach to some basic mine valuation problems on the Witwatersrand". In: *Journal of the Southern African Institute of Mining and Metallurgy* 52.6, pp. 119–139.

Li, W., Y. Tian, and M.-Q. Liu (2024). "Construction of orthogonal maximin distance designs". In: *Journal of Quality Technology* 56.4, pp. 312–326.

Liashchynskyi, P. and P. Liashchynskyi (2019). "Grid search, random search, genetic algorithm: a big comparison for NAS". In: *arXiv preprint arXiv:1912.06059*.

Liefvendahl, M. and R. Stocki (2006). "A study on algorithms for optimization of Latin hypercubes". In: *Journal of Statistical Planning and Inference* 136.9, pp. 3231–3247.

Lujan-Moreno, G. A. et al. (2018). "Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study". In: *Expert Systems with Applications* 109, pp. 195–205.

Luna, J. et al. (2022). "Orthogonal array composite designs for drug combination experiments with applications for tuberculosis". In: *Statistics in medicine* 41.17, pp. 3380–3397.

McKay, M. D. (1992). "Latin hypercube sampling as a tool in uncertainty analysis of computer models". In: *Proceedings of the 24th conference on Winter simulation*, pp. 557–564.

Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.

Morris, M. D. and T. J. Mitchell (1995). "Exploratory designs for computational experiments". In: *Journal of Statistical Planning and Inference* 43.3, pp. 381–402.

Myers, R. H., D. C. Montgomery, and C. M. Anderson-Cook (2016). *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons.

Owen, A. B. (1994). "Controlling correlations in Latin hypercube samples". In: *Journal of the American Statistical Association* 89.428, pp. 1517–1522.

Price, K., R. M. Storn, and J. A. Lampinen (2006). *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media.

Rasmussen, C and C Williams (2006). *Gaussian processes for machine learning*. MIT press: Cambridge, MA.

Roustant, O., D. Ginsbourger, and Y. Deville (2012). "DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization". In: *Journal of Statistical Software* 51, pp. 1–55.

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

Shi, C. and H. Xu (2024). "A Projection Space-Filling Criterion and Related Optimality Results". In: *Journal of the American Statistical Association* 119.548, pp. 2658–2669.

Stokes, Z., W. K. Wong, and H. Xu (2024). "Metaheuristic Solutions to Order-of-Addition Design Problems". In: *Journal of Computational and Graphical Statistics* 33.3, pp. 1006–1016.

Storn, R. and K. Price (1997). "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of Global Optimization* 11, pp. 341–359.

Sun, C.-Y. and B. Tang (2023). "Uniform Projection Designs and Strong Orthogonal Arrays". In: *Journal of the American Statistical Association* 118.541, pp. 417–423.

Sun, F., Y. Wang, and H. Xu (2019). "Uniform projection designs". In: *Annals of Statistics* 47.1, pp. 641–661.

Tabachnick, B. G. and L. S. Fidell (2019). *Using Multivariate Statistics*. Pearson.

Tian, Y. and H. Xu (2022). "A minimum aberration-type criterion for selecting space-filling designs". In: *Biometrika* 109.2, pp. 489–501.

— (2024). "Stratification pattern enumerator and its applications". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 86.2, pp. 364–385.

Vázquez, A. R. and H. Xu (2024). "An integer programming algorithm for constructing maximin distance designs from good lattice point sets". In: *Statistica Sinica* 34, pp. 1347–1366.

Wang, Y., S. Liu, and Q. Xiao (2024). "Construction of orthogonal-MaxPro Latin hypercube designs". In: *Journal of Quality Technology* 56.4, pp. 342–354.

Wang, Y., F. Sun, and H. Xu (2022). "On design orthogonality, maximin distance, and projection uniformity for computer experiments". In: *Journal of the American Statistical Association* 117.537, pp. 375–385.

Weise, T. (2009). "Global optimization algorithms-theory and application". In: *Self-Published Thomas Weise* 361, p. 153.

Wu, C. J. and M. S. Hamada (2011). *Experiments: planning, analysis, and optimization*. John Wiley & Sons.

Xu, H., J. Jaynes, and X. Ding (2014). "Combining two-level and three-level orthogonal arrays for factor screening and response surface exploration". In: *Statistica Sinica* 24.1, pp. 269–289.

Yang, X.-S. (2020). *Nature-inspired optimization algorithms*. Academic Press.

Yin, Y., L. Wang, and H. Xu (2023). "Construction of maximin L_1-distance Latin hypercube designs". In: *Electronic Journal of Statistics* 17.2, pp. 3942–3968.

Yuan, R. et al. (2025). "A construction method for maximin L_1-distance Latin hypercube designs". In: *Statistica Sinica* 35.1, pp. 249–272.

Zhou, Y.-D. and H. Xu (2017). "Composite designs based on orthogonal arrays and definitive screening designs". In: *Journal of the American Statistical Association* 112.520, pp. 1675–1683.

Appendix

Appendix A: Additional tables and figures for target sizes 50×5 and 70×7

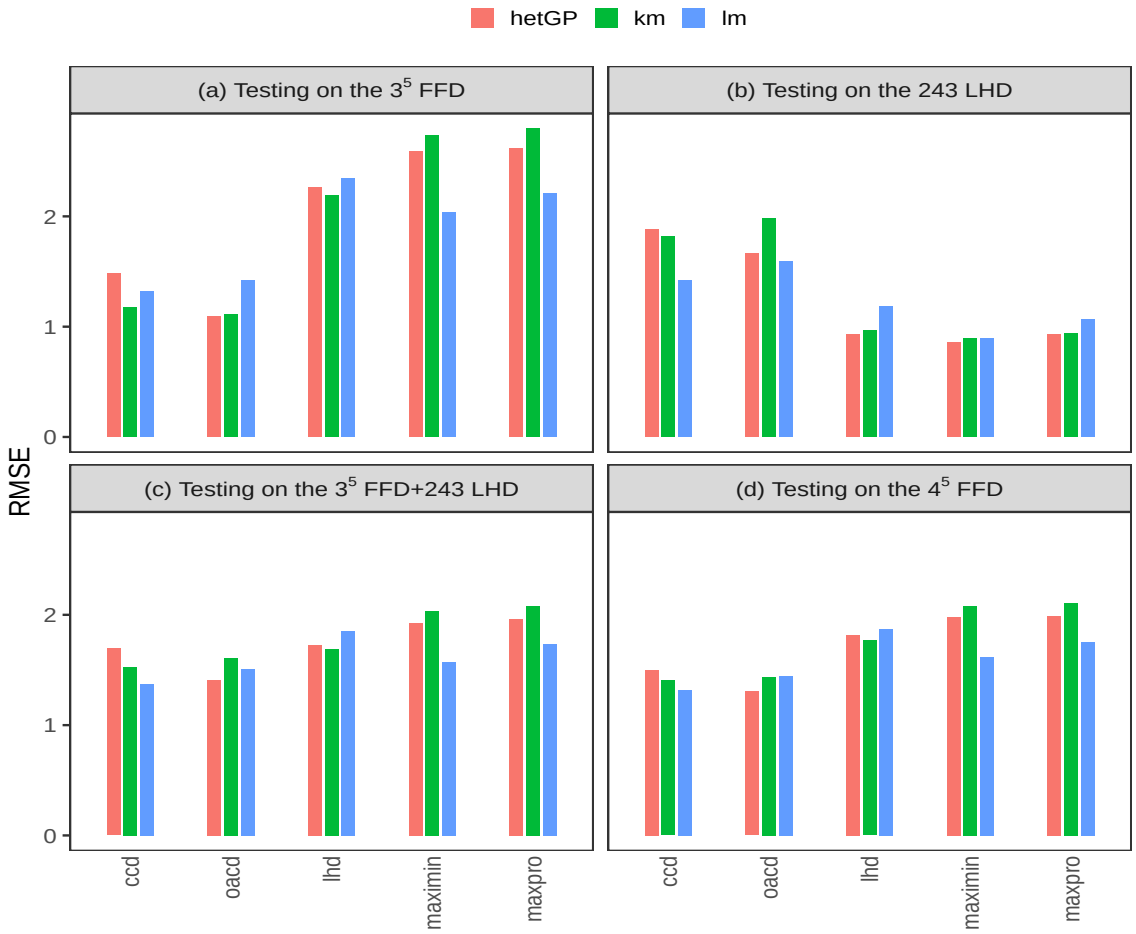


Figure 7: Comparison of RMSE with target size 50×5

Table 6: Comparison of designs and model evaluations with target size 50×5

	(a) Testing on the 3^5 FFD						**(b) Testing on the 243 LHD**					
	correlation			RMSE			correlation			RMSE		
Design	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP
ccd3_43	0.94	0.96	0.93	1.32	1.17	1.49	0.83	0.82	0.80	1.42	1.81	1.88
oacd3_50	0.94	0.96	0.96	1.42	1.11	1.09	0.83	0.84	0.86	1.59	1.98	1.66
lhd_50	0.83	0.85	0.83	2.35	2.19	2.26	0.89	0.92	0.93	1.18	0.97	0.92
maximin_50	0.87	0.84	0.81	2.03	2.73	2.58	0.92	0.92	0.93	0.90	0.89	0.85
maxpro_50	0.85	0.81	0.82	2.21	2.79	2.62	0.90	0.92	0.92	1.06	0.94	0.93
	(c) Testing on the 3^5 FFD+243 LHD						**(d) Testing on the 4^5 FFD**					
	correlation			RMSE			correlation			RMSE		
Design	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP
ccd3_43	0.92	0.92	0.89	1.37	1.53	1.70	0.93	0.93	0.92	1.32	1.40	1.50
oacd3_50	0.91	0.92	0.93	1.51	1.61	1.41	0.92	0.94	0.94	1.45	1.44	1.30
lhd_50	0.86	0.88	0.88	1.86	1.69	1.73	0.87	0.87	0.87	1.87	1.77	1.82
maximin_50	0.90	0.85	0.85	1.57	2.03	1.92	0.90	0.87	0.86	1.61	2.08	1.98
maxpro_50	0.88	0.83	0.85	1.73	2.08	1.97	0.88	0.85	0.86	1.75	2.11	1.99

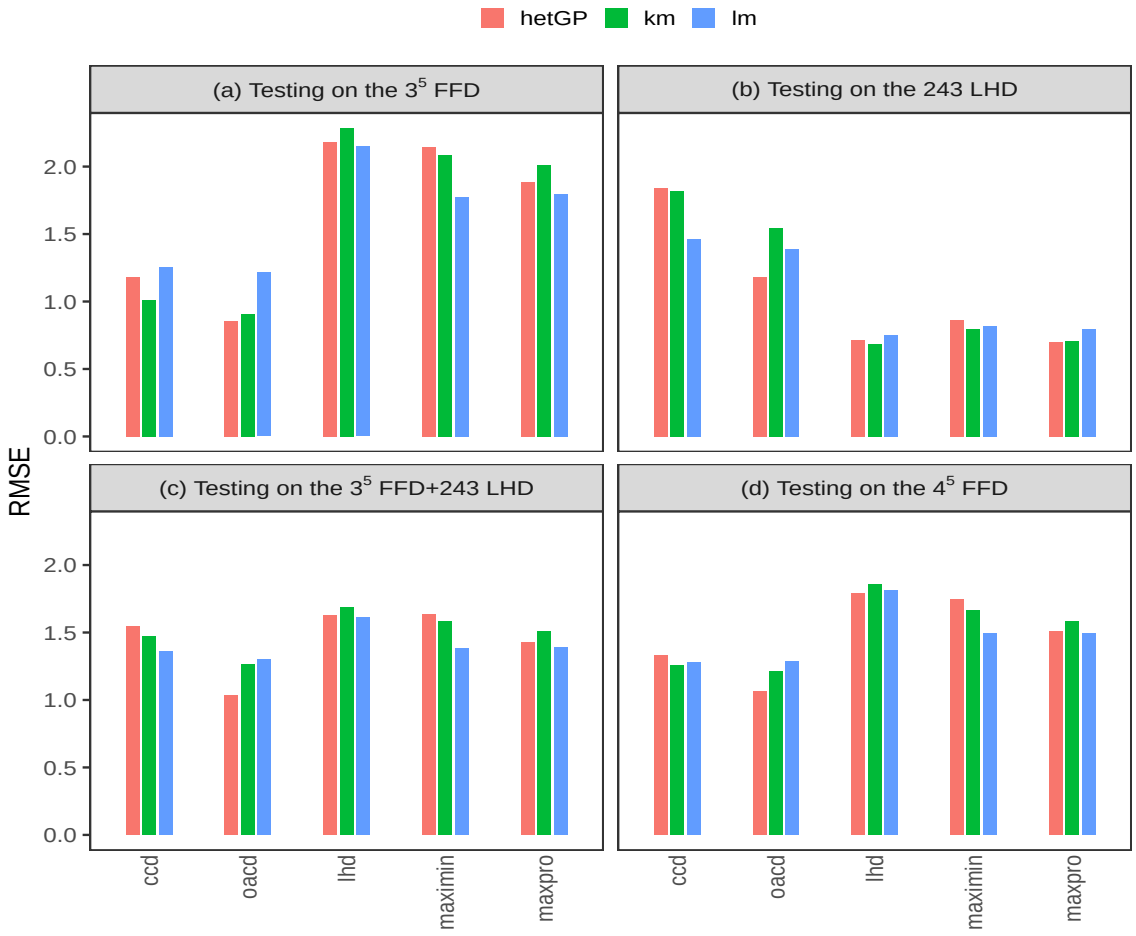


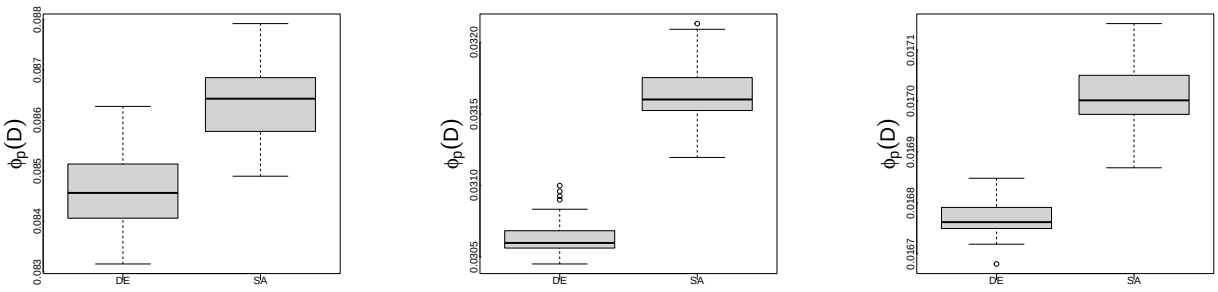
Figure 8: Comparison of RMSE with target size 70×7

Table 7: Comparison of designs and model evaluations with target size 70×7

	(a) Testing on the 3^5 FFD						**(b) Testing on the 243 LHD**					
	correlation			RMSE			correlation			RMSE		
Design	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP
ccd3_43	0.93	0.96	0.94	1.25	1.01	1.18	0.81	0.84	0.83	1.46	1.82	1.84
oacd3_50	0.94	0.97	0.97	1.22	0.91	0.86	0.83	0.88	0.92	1.39	1.54	1.18
lhd_50	0.83	0.84	0.82	2.15	2.28	2.18	0.94	0.95	0.95	0.75	0.69	0.72
maximin_50	0.87	0.87	0.81	1.77	2.09	2.14	0.94	0.94	0.93	0.82	0.80	0.86
maxpro_50	0.88	0.88	0.87	1.80	2.01	1.89	0.94	0.95	0.95	0.80	0.71	0.70
	(c) Testing on the 3^5 FFD+243 LHD						**(d) Testing on the 4^5 FFD**					
	correlation			RMSE			correlation			RMSE		
Design	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP	lm	km	hetGP
ccd3_43	0.90	0.91	0.90	1.36	1.47	1.54	0.92	0.94	0.93	1.28	1.26	1.33
oacd3_50	0.91	0.93	0.95	1.30	1.27	1.03	0.92	0.94	0.95	1.29	1.21	1.06
lhd_50	0.87	0.87	0.87	1.61	1.69	1.62	0.86	0.87	0.86	1.81	1.85	1.79
maximin_50	0.90	0.88	0.86	1.38	1.58	1.63	0.89	0.89	0.86	1.49	1.67	1.75
maxpro_50	0.90	0.89	0.90	1.39	1.51	1.43	0.90	0.90	0.90	1.49	1.58	1.51

Appendix B: Comparsion of DE and SA for constructing maximin LHDs

The DE algorithm can be tuned to construct other types of space-filling designs following the same strategy. For illustration, by using the $\phi_p(D)$ criterion 2 as the objective function, we tune the DE algorithm to construct maximin LHDs and compare it with the SA algorithm implemented in the R package SLHD. The `maximinSLHD` function employs the SA algorithm outlined in Morris and Mitchell (1995) and constructs optimal sliced maximin LHDs (Ba, Myers, and Brenneman 2015). To have fair comparisons, we use 15 random starts for the maximinSLHD function and set the slicing parameter $t = 1$. This gives better results than using just 1 random start and the timing is almost similar to that used with the DE approach. For each target size, we run both the DE and SA algorithms 100 times to get 100 maximin LHDs. Figure 9 compares the DE and SA algorithms for constructing maximin LHDs with target sizes 30×30, 50×5, and 70×7. The results clearly demonstrate that our tuned DE algorithm is superior to the SA algorithm.



(a) 30×3 as target (b) 50×5 as target (c) 70×7 as target

Figure 9: Comparison of the DE and SA algorithms for constructing maximin LHDs under the ϕ_p criterion (2)