March 2018

Optimal Latin Hypercube Designs for Computer Experiments Based on Multiple Objectives

Ruizhe Hou
University of South Florida, houruizhe1210@gmail.com

Optimal Latin Hypercube Designs for Computer Experiments Based on Multiple Objectives

by

Ruizhe Hou

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Statistics
Department of Mathematics & Statistics
College of Statistics
University of South Florida

Major Professor: Lu Lu, Ph. D.
Kandethody Ramachandran, Ph. D.
Dan Shen, Ph. D.

Date of Approval:
March 22, 2018

Keywords: Gaussian process, Screening design, Space-filling design, Pareto front search algorithms

DEDICATION

To my parents.

ACKNOWLEDGMENTS

I am grateful to everyone who has taught, inspired and helped me during the past two years of my graduate study at the USF. I want to express my sincere gratitude to my advisor, Dr. Lu Lu, for her advice, guidance, support, patience and encouragement during my thesis work. I also want to thank Dr. Kandethody Ramachandran and Dr. Dan Shen for serving on my committee and their valuable comments and advice during the process. Finally, many thanks to all my friends and fellow graduate students for their support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Latin hypercube designs (LHDs) have broad applications in constructing computer experiments and sampling for Monte-Carlo integration due to its nice property of having projections evenly distributed on the univariate distribution of each input variable. The LHDs have been combined with some commonly used computer experimental design criteria to achieve enhanced design performance. For example, the Maximin-LHDs were developed to improve its space-filling property in the full dimension of all input variables. The MaxPro-LHDs were proposed in recent years to obtain nicer projections in any subspace of input variables. This thesis integrates both space-filling and projection characteristics for LHDs and develops new algorithms for constructing optimal LHDs that achieve nice properties on both criteria based on using the Pareto front optimization approach. The new LHDs are evaluated through case studies and compared with traditional methods to demonstrate their improved performance.

Keywords: Design of computer experiments; Gaussian process; Latin hypercube design; Screening design; Space-filling design; Multiple objective optimization; Pareto front search algorithms.

CHAPTER ONE:

INTRODUCTION

Computer experiments (Sacks et al., 1989) have been broadly used in scientific and engineering research in lieu of the more expensive or practically infeasible or prohibited physical experiments (Verlet, 1967). In a computer experiment, computer code is used to simulate a complicated process of interest with the generated data being used for studying the relationship between the input variables and the output of the process. For example, computer simulations are often constructed to emulate complex physical systems using discrete event simulation models (Law et al., 2007), finite element models (van Rietbergen et al., 1995), etc. Computer models are extensively used for climate simulations (Washington et al., 2000) as physical experiments on an Earth-sized object is impossible. Computer experiments are also commonly used in computational physics (Pang, 1999), computational chemistry (Jensen, 2017), computational biology (Waterman, 1995) and other similar disciplines to avoid the complicated large-scale physical experiments. In addition, the computer experiments have become more popular in generating informative data for developing precise simpler surrogate models (Bandler et al., 2004) to replace the complicated and time-consuming computer simulation models to support timely decision-making.

A major difference between the computer experiments and the physical experiments is that a computer experiment involves a deterministic process where identical data are produced from running the same simulation code at the same set of inputs for multiple times, while a physical experiment involves a stochastic process which generates varied data due to the natural variability of the process with many unknown noise factors and unavoidable measurement errors. Due to this fundamental difference in the

produced experimental data, the general principles for designing physical experiments such as the use of replication, randomization, and blocking techniques (Montgomery, 2017; Wu & Hamada, 2010) are no longer suitable for designing computer experiments. Instead, different design characteristics are considered for producing informative data.

Generally, there have been two categories of design criteria considered for quantifying the design performance for a computer experiment, namely the geometric distance-based criteria and the model-based criteria. The distance-based criteria focus on maximally spreading the geometric locations of the observations in the design space. The space-filling designs are often used to spread the design locations evenly throughout the design region based on the consideration of covering as much design space as possible to avoid missing any interesting features. The model-based criteria evaluate the design performance based on some specified statistical models such as minimizing the variance of the estimated model parameters or minimizing the bias with respect to some known ground truth about the underlying mechanism of the process. A well-known example is the D-efficiency (Myers et al., 2016), which measures the precision of the estimated response surface models. When Gaussian Process (GP) models (Rasmussen, 2004) are used for analyzing computer experiment data, the integrated mean squared prediction error (Sheiner et al., 1981) or entropy criteria (Jones et al., 1990) are often used to measure the predictive properties of the estimated GP models. Since the model-based design criteria rely on some good understanding of the underlying process and choosing an appropriate model, this thesis focuses on the more flexible space-filling designs when constructing optimal computer experiments.

Space-filling designs generally offer a good coverage across the entire experimental region and great flexibility in fitting a variety of different possible models. However, the "space-filling" property also involves several different aspects which can be captured by different design criteria. For examples, "spacing-filling" could mean to maximally spread the design points by avoiding any two points to be too close together, which is based on a consideration that two close design points will provide similar information and waste the valuable design resource. Hence, the *maximin* distance criterion (Johnson et al.,

1990) was developed to maximize the minimum distance between any pair of design points. On the other hand, "space-filling" could also mean to cover every possible region in the design space with at least one design point based on a belief that any interesting features of the true underlying relationship are likely to exist in any design region and hence it is important to avoid missing out any large region of the design space. This has led to the development of the *minimax* distance criterion (Johnson et al., 1990), which minimizes the maximum distance of design locations to any arbitrary point in the design space. Many experiments have verified that space-filling designs can produce a good performance for the generated computer experiments (Johnson et al., 1990; Pronzato et al., 2012 & Husslage et al., 2011).

Both maximin and minimax distance designs ensure good space-filling property in the full dimensions of all p input variables. However, they do not guarantee good design properties when projected into lower dimensions if some of the input variables turn out not to have a ignificant impact on the response. The Latin hypercube design (LHD) was developed to ensure a uniform marginal distribution for each individual variable, i.e. the design is uniformly spread when projected into the univariate space of each input variable. However, a randomly selected LHD is not guaranteed to have good space-filling property across the full dimension design space. For example, a LHD with n points evenly distributed along the $45°$ diagonal line in the unit square $[0,1]^2$ design space with two input variables apparently does not cover the design space well (all design points located on a line leaving the rest of the unit square unexplored). To combine the merits of space-filling for the full dimensional design space and for the projected univariate subspace, the hybrid designs such as the Maximin-LHD and the minimax-LHD (Husslage et al., 2011) were developed to achieve desirable features in both the full and each single dimension.

Based on the effect of sparsity principle (Montgomery, 2017), in many experiments, only a subset of all the input variables may actually affect the response. Therefore, it is particularly important for the designed computer experiment to have good space-filling property when it is projected into the space of the subset of useful variables. However, often we do not know which variables might be useful before

conducting the experiment. The maximum projection (MaxPro) design (Joseph et al., 2015) was developed to optimize the projections namely having projections as maximin distance designs, across all the subspaces of 2 to $p-1$ input variables. Later on, MaxPro-LHD (Joseph, 2016) was also developed to create LHDs with good projections for all subspaces including single and full dimension spaces.

Despite Maximin-LHD, minimax-LHD, and MaxPro-LHD are substantial enhancements from the general LHDs. These designs still provide different space-filling properties in general. For examples, the Maximin-LHD or minimax-LHD ensures good space-filling property for the full and each single-dimensional design space, with the former focusing on spreading any pair of design points as far as possible and hence pushing more points towards the edge of the design space, while the latter focusing on not to miss any design region and hence having more runs located around the center of the design region. On the other hand, the MaxPro-LHD operates on the 2 to $p-1$ subspaces of the input variables excluding the single and full dimensional spaces. This thesis improves all these existing designs by combining multiple criteria when seeking optimal computer experiments.

The Pareto front (PF) method has been adapted for design optimization based on multiple criteria since Lu et al. (2011) and has been applied for designing physical experiments with varied design structures and randomization constraints while considering a variety of different design criteria (Lu et al. 2012; Lu & Anderson-Cook, 2012 & 2014; Lu, Robinson, Anderson-Cook, 2012; Lu, Johnson, Anderson-Cook, 2014). The method uses a two-phase process to guide structured design optimization and selection. In the first phase, the Pareto front (Marler et al., 2004) consisting of the set of contending (non-dominated) designs based on considering interested multiple design criteria is found through a search algorithm. Since the selection of the PF of optimal design does not rely on any subjective choices on how to scale, combine and prioritize the different design criteria, it offers an objective set of superior choices from which to select the final design. In the second subjective phase, different desirability functions (Derringer et al., 1980) with varied weighting, scaling and functional forms could be explored to understand the trade-offs between most promising designs and their robustness to different subjective

choices to facilitate understanding and making an informed evidence-based decision on selecting the optimal design of computer experiment for the goals of the particular experiment.

This thesis develops new search algorithms for the finding the Pareto optimal LHDs based on considering all of the maximin, minimax, and MaxPro criteria, which provide a set of good candidates with desired space-filling properties on all dimension of projections. Two different search algorithms are developed including a column-wise exchange algorithm (Li et al., 1997) paired with simulated annealing (Kirkpatrick et al., 1983) and a customized genetic algorithm (Mitchell, 1998) for multiple objective optimization. The performance of the newly developed search algorithms is evaluated and compared through case studies. An R package named MOLHD (Multiple Objective Latin Hypercube Designs) was developed based on the new search algorithms to generate optimal LHDs based on multiple objectives for supporting improved computer experiment design for broad practitioners. The package is published at ***The Comprehensive R Archive Network*** (CRAN) (Hou et al., 2017, *URL* *http://CRAN.R-project.org/package=MOLHD*).

In addition, the graphical tools from Lu et al. (2011), Lu and Anderson-Cook (2012), and Lu et al. (2013) were used to evaluate and compare options and guide the further selection of the most promising designs from the identified Pareto front. The selected designs are compared with the optimal designs selected using existing methods (focusing on a single criterion) based on different measures of design performance in a few case studies.

The remaining of the thesis is organized as follows. Chapter 2 discusses more details on the design criteria for evaluating different aspects of performance of computer experiments. Chapter 3 provides background on the Pareto front optimization approach and more details on the new developments of the numerical search algorithms. Chapter 4 illustrates the new method and algorithms and demonstrates the performance of developed optimal LHDs through case studies. Chapter 5 includes the conclusions.

CHAPTER TWO:

DESIGN CRITERIA

Computer experiments use computer models that are designed to simulate what would happen in a situation if a physical experiment was performed. They are very useful when the physical experiments are very hard or cost prohibitive to implement in real life to study interesting effects such as earthquake performance, the climate change, a nuclear explosion process, etc. Using computer-based models will make these experiments possible or not harmful, and hence computer experiments have become more and more popular these days. Design of computer experiments aims to determine at what input settings to run computer models to obtain data in order to build a better understanding of how input factors drive changes on the output/response of the process. Moreover, computer experiments have been used more often to build surrogate models as approximates to the more complicated and time-consuming computer models to support timely decision-making.

The challenge of running a good computer experiment focuses on how to collect most informative data using fewer resources. Typically, in running computer experiments, using fewer runs means saving time and cost because every single run of a complex computer model can sometimes take days or weeks to complete. To accomplish this goal, several design criteria have been proposed to evaluate the design performance on different aspects. Some criteria are model dependent while others are measured based on the geometric location of design points in the design space which does not rely on appropriately specified models.

In this section, we will focus our discussion on the space-filling criteria that are independent of model assumptions, which offers more flexibility of collecting informative and relevant data to allow further exploration of a variety of models in the analysis stage. Particularly, we will discuss the three most popular space-filling designs that are often used for computer experimental design including the Maximin-LHD, minimax-LHD, and MaxPro-LHD designs, and provide more details on how to construct customized computer-generated design based on a chosen criterion. In the next chapter, we will introduce our new method for generating optimal designs based on combining all three or any pair of these criteria.

The geometric distance-based criteria were first introduced by Johnson et al. (1990). Johnson and his colleagues developed two types of space-filling designs referred to as the minimax and maximin distance designs. They had shown that such designs can achieve quite general asymptotically optimum (and dual) characteristics under the G- and D-criteria.

To define notation, let *T* denote the design space which is formed by the collection of all possible design locations. Let $d(\cdot,\cdot)$ be the Euclidean distance function defined on $T \times T$. Below we will introduce the four categories of commonly used space-filling designs in computer experiments.

2.1 Latin Hypercube Design

The Latin hypercube design (LHD) method was first introduced by McKay (McKay et al., 1979). It has been a commonly used design for generating a near-random sample of parameter values from any model with a multidimensional distribution. For any two-dimensional square grid that contains a set of samples, it is called a Latin square, if and only if for each row and each column, there is only one sample. Latin hypercube is developed from this concept that is suited for any number of dimensions.

For simplicity, assume our design space is the two-dimensional unit square $[0,1]^2$. In order to obtain an LHD containing n_s samples, we can split each axis $[0,1]$ into n_s equally sized intervals $[0, {}^{1}/_{n_s}), \cdots, [{}^{(n_s-1)}/_{n_s}, 1]$. This will partition the unit square into n_s^2 cells of equal size. We want to fill in these cells with integers $1, 2, \cdots, n_s$ (corresponding to the run numbers) so as to form a Latin square,

which means that each integer appears in each row and in each column exactly once of this grid of cells. For simplicity, we choose the centers of the selected cells containing the n_s integers as the design points. Then the resulting n_s points form an LHD of size n_s, with each row and each column contain only one design point. With this sampling structure, the LHD has a uniform distribution of the design points when projected into the single space of each input variable. Hence, the LHD ensures all univariate projections to be space-filling in the single marginal space. However, the LHD does not guarantee evenly spread design points over the entire full dimensional unit square (e.g. a LHD can locate all design points along the diagonal of the unit square, which is apprently not space-filling over the full dimension).

In addition to the nice univariate projection property, the LHDs are also advantageous in its ease of generating the design matrix for design construction. Particularly, in an LHD matrix of $n \times p$ dimension with n runs for p input variables, each column of the deisgn matrix can be formed by a permutation of the integers $1,2,\cdots,n$, which indicates the locaiton of the deisgn points for the corresponding input variable. Because of the nice projection and design construction features of LHDs, we will constraint ourselves to consider only LHDs when constructing optimal computer experiments with superimposed space-filling criteria.

Example 2.1.1 Illustration on how to construct a Latin hypercube design.

Suppose we want a Latin hypercube design with 5 runs for 4 input variables. The design matrix with integer entries, denoted by LHD (5,4), contains 5 rows corresponding to the cell locations of the 5 runs and 4 columns corresponding to the 4 input variables. Each column of the design matrix is a permutation of the integers $1,2,\cdots,5$. An example of LHD (5,4) is given as follows:

$$\begin{matrix} 1 & 4 & 1 & 1 \\ 2 & 2 & 3 & 4 \\ 3 & 5 & 4 & 5 \\ 4 & 1 & 2 & 2 \\ 5 & 3 & 5 & 3 \end{matrix}$$

Note the integers indicate the location of the cells containing the sample runs over the 5×5 grid of cells partitioning the design space $[0,1]^4$. Since in the thesis, all runs are located at the center of each Latin square cell, then the Latin hypercube design matrix with the actual design locations are converted from the generated integer $LHD(5,4)$ design matrix by $LHD(5,4) \times 0.2 - 0.1$, which is given by:

0.1	0.7	0.1	0.1
0.3	0.3	0.5	0.7
0.5	0.9	0.7	0.9
0.7	0.1	0.3	0.3
0.9	0.5	0.9	0.5

2.2 Minimax-LHD

For subset S of T, $card(S) = n, n$ is fixed, define S^* as a "minimax distance design" (Johnson et al., 1990) if

$$\min_{S} \max_{t \in T} d(t, S) = \max_{t \in T} d(t, S^*) = d^* \tag{2.1}$$

where $d(t, S) = \min_{s \in S} d(t, s)$.

Understanding the minimax distance design can be made easier by connecting to the popular franchise-store placement problem. Imaging the locations of the collection of franchise stores as a design S^* in the possible location space, T. Then for any potential customer that lives in the interested location space, we want to locate the set of franchise stores to minimize the distance to the farthest customer from any of the stores to achieve maximum customer satisfaction on the location of the stores. Similarly, by locating the design points being not too far from any random point within the design space, we can ensure good precision for any prediction at any arbitrary location in the design space. In this case, we consider the minimax distance design has filled the design space to produce good prediction throughout.

We combine minimax distance design with Latin hypercube designs as the minimax-LHDs, which means we choose S and S^* in the class of Latin hypercube designs, and T always refers to a hypercube design space $[0,1]^p$ with infinite sites, where p is the number of variables.

For computing the minimax-LHD criterion, it is necessary to let a finite location set T^* replace the usually infinite set T. We propose a method as following. Let T^* be a subset of T with k^p points, located on the center of square girds, where the range of each variable is divided into k equally probable intervals. Then the criterion of each design (S) becomes $\min_{S} \max_{t \in T^*} d(t, S)$, which will be more accurate if k is chosen larger. The accuracy as the largest error of using finite set to compute the minimax-LHD criterion can be easily computed.

The process for searching the minimax-LHD is described as follows:

1) Start with a random LHD
2) Compute its minimax-LHD criterion using formula (2.1) (using a finite set T^* to replace T if necessary)
3) Get a new LHD by randomly column-wise exchange the current LHD (see Example 3.2.2 for illustration of column-wise exchange)
4) Compute the new minimax-LHD criterion using formula (using the same set T^* as step 2, this step has computational shortcuts based on the procedure of calculating the design criterion on step 2)
5) A. If the new criterion is smaller (better), set the new LHD as the current.
   B. If the new criterion is greater (worse), the current LHD remains unchanged.
6) Go to step 3 until the terminal conditions are met. And get the latest current LHD as the minimax-LHD.

Finding the near-optimal minimax-LHD with the desired accuracy is very computationally challenging since the number of grid points will increase exponentially as the number of variables increases. Even if we use a finite set T^* to replace T, the set T^* can still be a huge set when the number of variables becomes great. Therefore, when dealing with a great number of variables, we have to reduce our desired accuracy by using T^* with fewer elements. Sometimes even that is very time-consuming.

Example 2.2.1 Illustration on how to generate the replacement set T^*.

(1) If we set $k = 2$, when there are 2 variables, then the finite set T^* is:

$\{(0.25, 0.25), (0.25, 0.75), (0.75, 0.25), (0.75, 0.75)\}$

(2) If we set $k = 4$, when there are 2 variables, then the finite T^* is:

$\{(0.125, 0.125), (0.125, 0.375), (0.125, 0.625), (0.125,0.875), (0.375, 0.125), (0.375, 0.375), (0.375, 0.625),$

$(0.375,0.875), (0.625,0.125), (0.625,0.375), (0.625,0.625), (0.625,0.875), (0.875,0.125),$

$(0.875,0.375), (0.875,0.625), (0.875, 0.875)\}$

2.3 Maximin-LHD

For subset S of T, $card(S) = n, n$ is fixed, define S°as a "maximin distance design" (Johnson et al., 1990) if

$$\max_{S} \min_{s,s' \in S} d(s, s') = \min_{s,s' \in S^\circ} d(s, s') = d^\circ \tag{2.2}$$

The *index* I° denotes the number of pairs of points in S° that share the distance d°. Let $S^{\circ\circ}$ represent the design with smallest I° among all S° designs that have been found. Then in that case, $S^{\circ\circ}$ is the maximin distance design.

From (2.2), we can see that it is an opposite way to solve the franchise store placement problem compared to (2.1) based on another viewpoint. This time, the holder of the franchise emphasizes that the store operator the largest 'exclusive territory' that can be managed. "It extends up to a distance of at least $\frac{1}{2} d^\circ$ by the triangle inequality" (Johnson et al., 1990). In addition, this distance d° should be obtained least often in the final optimal maximin distance design, which means it has the smallest I°. This design here focuses on non-redundancy in choosing samples.

We combine maximin distance design with Latin hypercube designs as the maximin-LHDs, which means that we choose S and S° in the class of Latin hypercube designs. And when S and S° are Latin Hypercube designs, Morris & Mitchell (1995) proposed an alternative and simplified criterion. For subset S of T, $card(S) = n, n$ is fixed (Morris et al., 1995), if S° satisfies

$$\min_{S, x_i, x_j \in S} \{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{d^k(x_i, x_j)}\}^{1/k} = \{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{d^k(x_i^\circ, x_j^\circ)}\}^{1/k} \quad x_i^\circ, x_j^\circ \in S^\circ \tag{2.3}$$

then S° is also the maximin distance design, where *n* is the size of set *S* and also is the total number of runs, and $k > 0$ is chosen large enough to achieve the maximum distance between any two design points.

The above criterion calculates the same thing as the original maximin distance criterion, but it is easier to compute and avoid counting the number of pairs of points in S° that share the distance d°. Hence, we use this criterion in implements.

The process for searching the maximin-LHD is described as follows:

1) Start with a random LHD
2) Compute its maximin-LHD criterion using formula (2.3)
3) Get a new LHD by randomly column-wise exchange the current LHD
4) Compute the new maximin-LHD criterion using formula (this step has some computational shortcuts based on the procedure of calculating the design criterion on step 2)
5) A. If the new criterion is smaller, set the new LHD as current.
   B. If the new criterion is greater, the current LHD remains unchanged.
6) Go to step 3 until the terminal conditions are met. And get the latest current LHD as the maximin-LHD.

Finding the near-optimal maximin-LHD can be hard but it is not that time-consuming as finding the near-optimal minimax-LHD.

2.4 MaxPro-LHD

The MaxPro criterion was developed by Joseph et al. (2015). It is known to generate good space-filling designs when it is necessary to project into a lower dimensional space formed by a subset of active input variables. For subset S of T, $card(S) = n, n$ is fixed (Joseph et al., 2015), define S' as a maximum projection design if

$$\min_{S,\, x_{il},\, x_{jl} \in S} \left\{ \frac{1}{\binom{n}{2}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{\prod_{l=1}^{p} (x_{il} - x_{jl})^2} \right\}^{1/p} = \left\{ \frac{1}{\binom{n}{2}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{\prod_{l=1}^{p} (x'_{il} - x'_{jl})^2} \right\}^{1/p} \; x'_{il}, x'_{jl} \in S' \qquad (2.4)$$

where *n* is the size of set *S*, also is the total number of runs, *p* is the number of variables.

In (2.4), since for any l, if $x_{il} = x_{jl}$ for $i \neq j$, then the criterion will become infinite and the minimum is desired. Therefore, the maximum projection design must have n distinct levels for each factor which shares the similar property as Latin hypercube designs.

We also combine the maximum projection design with Latin hypercube designs by choosing *S* and S' in the class of Latin hypercube designs.

The process for searching the MaxPro-LHD is described as follows:

1) Start with a random LHD
2) Compute its MaxPro-LHD criterion using formula (2.4)
3) Get a new LHD by randomly column-wise exchange the current LHD
4) Compute the new MaxPro-LHD criterion using formula (this step has some computational shortcuts based on the procedure of calculating the design criterion on step 2)
5) A. If the new criterion is smaller, set the new LHD as current.
   B. If the new criterion is greater, the current LHD remains unchanged.
6) Go to step 3 until the terminal conditions are met. And get the latest current LHD as the MaxPro-LHD.

Finding the near-optimal MaxPro-LHD is comparably time-consuming as finding the near-optimal maximin-LHD. Both of them are not hard as finding the near-optimal minimax-LHD.

CHAPTER THREE:

OPTIMAL LHDS BASED ON MULTIPLE CRITERIA

3.1 Pareto Front Optimization

In this section, we provide some basic ideas on the Pareto front approach. Pareto Front optimization is a method to find a set of superior solutions based on considering multiple objectives. For design optimization, if we are considering multiple different objectives, balancing trade-offs between different design objectives can always be very necessary. The Pareto front approach can help users recognize the existing trade-offs between different objectives, and moreover, it also provides a set of contending solutions, which correspond to different weights and scaling among interested criteria (Lu et al., 2011).

To illustrate the Pareto front approach, suppose our goal is to solve a multiple criteria design optimization problem, which is to maximize C ($C \geq 2$) criteria simultaneously for simplicity. Let $\xi \in \boldsymbol{\Omega}$ denote a design matrix with N rows as the number of design points and k columns as the number of variables. $\boldsymbol{\Omega}$ denotes the candidate set of all possible matrices with N rows and k columns. Let vector $\boldsymbol{y} = \boldsymbol{F}(\xi) = (f_1(\xi), f_2(\xi), \cdots, f_C(\xi))'$ be the computed criteria values to each of the C criteria. The region that contains all obtainable different criteria vectors is the "criterion space".

In this case, a solution matrix ξ_1 can be said to Pareto dominate (Lu et al., 2011) matrix ξ_2 if and only if $f_i(\xi_1) \geq f_i(\xi_2)$, for any $i \in \{1,2,\cdots,C\}$, and there exists at least one $j \in \{1,2,\cdots,C\}$ that $f_j(\xi_1) > f_j(\xi_2)$. Similar to the matrices solutions, the corresponding criteria vector $\boldsymbol{F}(\xi_1)$ is also said to Pareto dominate $\boldsymbol{F}(\xi_2)$. A Pareto optimal design or matrix is a design that no other design dominates it, and

hence its criteria vector is not dominated by any other criteria vector. The set of all the Pareto optimal designs is called the Pareto optimal solutions, and the set of corresponding criteria vectors is call the Pareto Front (PF). In the criterion space as talked above, each criteria vector represents a specific point.

The Pareto front (PF) and the Pareto optimal solutions set can often be found through search algorithm. Therefore, we construct the search algorithm based on the definition above (see next section 3.2 for more details).

Once the decision-maker obtains the Pareto optimal solutions, he/she will always need a single solution, or a small subset of it for implements. Since this selection procedure always requires quantified preferences of different objectives, offering a more suitable set to decision-makers can be a very effective way. Utopia point method proposed by Lu (Lu et al., 2011) is a common way of selecting more suitable solutions. Call a point as the Utopia point, $\boldsymbol{F^0}$, in the criterion space if

$$\boldsymbol{F^0} = \underset{\xi}{Max}\{f_i(\xi)|\xi \in \boldsymbol{\Omega}\} \quad \text{for all} \quad i\epsilon\{1,2,\cdots,C\} \tag{3.1}$$

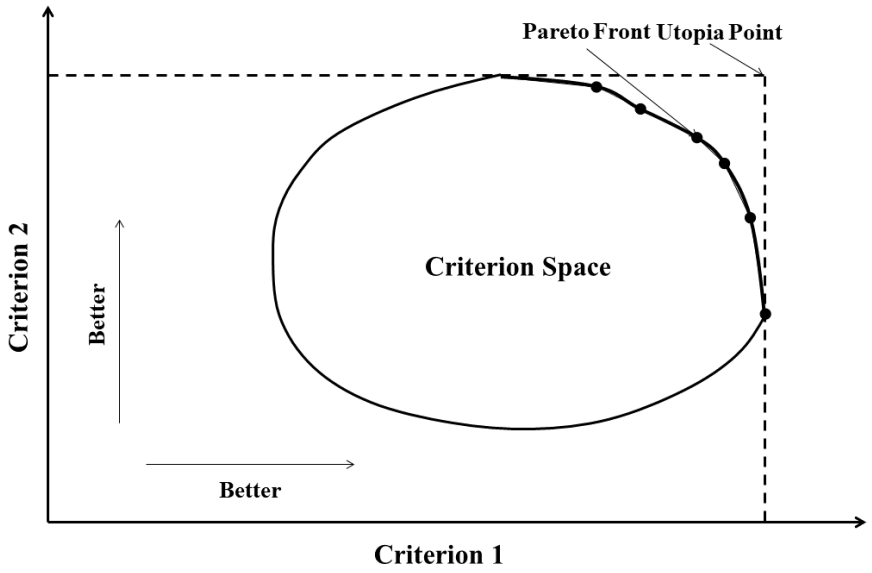


Figure 1. "Illustration of the Pareto front and the Utopia point in a two-criterion maximization problem." (Lu et al., 2011)

If there is a solution $\xi^0 \in \boldsymbol{\Omega}$ that satisfies $\boldsymbol{F}(\xi^0) = \boldsymbol{F^0}$, then ξ^0 is called a Utopia solution. But unfortunately, for multiple objectives optimization problem, the Utopia solution ξ^0 usually does not exist and hence the Utopia point is a criteria vector that usually cannot be attained. But we can use the Utopia point as an ideal standard in choosing one or a set of better solutions from the Pareto optimal solutions found in the search algorithm. Figure 1 (Lu et al., 2011) give an example of the Utopia point within a two-objective maximization problem. From this figure, we can see that the Utopia point is at the maximum of both the two criteria. The Pareto front points are the closest points to the Utopia point, and form as a "boundary" of the criterion space. This "boundary" clearly indicates that there is no way to make one criterion better without decreasing the other criterion. Utopia point selecting method usually prefers to choose the point on the "boundary" that achieves the minimum distance to the Utopia point, unless there are specific preferences towards one or some of the criteria. In that case, there will be different weights for each criterion in choosing the optimal solution. A reasonable scaling of each criterion is important when implementing this method. A frequently used scaling method is to scale the maximum value on the Pareto front to 1 and the minimum value on the Pareto front to 0 for each criterion.

3.2. Pareto Front Search Algorithms

For searching the Pareto front and Pareto optimal designs, we have two different search algorithms. First is column-wise exchange algorithm. Second is a genetic algorithm. Both algorithms need to set a random Latin hypercube design as the start.

3.2.1 Generating Latin Hypercube Design

Let us recall section 2.1. Suppose that a Latin hypercube design containing n_s samples is desired. For simplicity, suppose the design space has been transformed into $[0, 1]^p$. We divide the k^{th} axis into n_s intervals. Each of the intervals has equally sampling probability, $1/n_s$. The division points for k^{th} axis are as follows

$$\frac{1}{n_S}, \cdots, \frac{n_s-1}{n_s}.$$

We can number the n_s parts with integer $1, 2, \cdots, n_s$. In such a way, we can get n_s^p hypercube cells. We select n_s cells so as that for each variable among the *p* variables, each integer appears exactly once. In each selected cell, select the center point of the cell. The resulting n_s points form an LHD of size n_s. This center-located LHDs are used in our thesis.

3.2.2 Column-wise Exchange Algorithm

This algorithm is column-wise exchange algorithm coupled with simulated annealing algorithm. Column-wise exchange, in this case, means to randomly select two runs from a Latin hypercube design. and switch their values. Since randomly swapping two entries in a permutation will stay result in another permutation. In such way, after column-wise exchange, the design will still be a Latin hypercube design. Simulated annealing (Kirkpatrick et al., 1983) is a commonly used method to approximate the global optimum of any given function. In this case, it means that after column-wise exchanging the current design, if the criterion difference $\Delta C = (C_{new} - C_{current})$ is positive (suppose we want to maximize the criterion, change to negative if we want to minimize the criterion), we will accept the new design as the current design and keep on searching; otherwise, the acceptance probability is $P = exp(\Delta C/T)$ ($P = exp(-\Delta C/T)$ if we want to minimize the criterion), where T is a global time-varying parameter called the temperature and it is set to gradually approach zero. When T becomes closer to 0, the acceptance probability P will be close to 0 as ΔC is negative. As T becomes sufficiently small, the system will then "increasingly favor moves that increase criterion" and "avoid those that decrease criterion". When T equals to 0, this algorithm works the same as the greedy search algorithm, which only makes "the increasing criterion transitions". Theoritical proof (Granville et al., 1994) has comfirmed that simulated annealing search algorithm will end up with the global optimal solution with probability approaching 1.

Example 3.2.1.1 Illustration of column-wise exchange.

A column-wise exchange of a 4 runs 4 variables LHD. Switch the values of the 2nd variable within the 2nd and 3rd runs.

$$\begin{matrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{matrix} \rightarrow \begin{matrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 2 & 2 \\ 3 & 2 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{matrix}$$

The Pareto front always contains more than one criteria vector. Every criteria vector could be seen as the best for a weighted sum of these criteria. It suggests us that we should give each criterion a relative weight and use the weighted sum of them as the search direction. To avoid the significant difference between each criterion, we need to find the criteria region and scale each criterion between 0 and 1.

First, we begin with a random Latin hypercube design. Next, we get a new design by column-wise exchanging it. Then, we calculate their criterion difference (ΔC) by using some calculation shortcuts and use simulated annealing as above to decide whether we accept the new design as the current design. Finally, the searching procedure terminates as the temperature goes low and we get the optimal design with its criterion value. After we have found the criteria range, the criteria values can be transformed into $[0, 1]$.

After the single criterion region search, we set a sufficient weight set W. Each element of W is a weight vector (w_1, w_2) (or (w_1, w_2, w_3), depend on how many criteria are concerned), where $\sum_{i=1}^{2} w_i = 1$, $w_i \in [0,1]$, $i = 1,2$ (or $\sum_{i=1}^{3} w_i = 1$, $w_i \in [0,1]$, $i = 1,2,3$). Let $C^* = \sum_{i=1}^{2} w_i C_i$ (or $C^* = \sum_{i=1}^{3} w_i C_i$) be the new criterion, where C_i is the scaled criterion. For each weight vector in set W, using the same algorithm as above to search the optimal design under the new criterion C^*. This time we set the initial design as the initial Pareto Front design, its criteria vector as the initial Pareto Front. While searching, we check every new design's criteria vector with the current Pareto Front. If it is not dominated by all the criteria vectors on the current Pareto Front, add it to current Pareto Front, delete all the criteria vectors that are dominated by it, and update the Pareto optimal designs set in the same way. If

it is dominated by one or some of the criteria vectors on the current Pareto Front, then the current Pareto Front and its designs set remain unchanged. By doing so, after the searching procedure is finished for each weight vector, we will get a set of Pareto Front with its corresponding designs set. The final Pareto Front values with its corresponding designs will be the combination of all the results.

There are two challenges to this algorithm. One is the hard-computing problem of the minimax-LHD criterion. The other is the time-consuming problem when dealing with a large size of Pareto Front. The first one is a difficult problem to solve for a long time. The second problem can be overcome by using a genetic algorithm with some loss of accuracy. The pseudo-code for the column-wise exchange algorithm is provided in Appendix A. Algorithm functions can be found in the R package "**MOLHD**" (Hou & Lu, 2017).

3.2.3 Genetic Algorithm

Genetic algorithms (Mitchell, 1998) are popular in generating solutions (called individuals, creatures, or phenotypes) with high quality for optimization problems or search problems. It relies on some operators including selection, crossover, and mutation, which are inspired by biology.

Generally, in genetic algorithms, there is a "population" set, containing potential solutions to an optimization problem, evolving toward better solutions. Each candidate solution should have some attributes that can be mutated and changed. Traditionally, solutions are represented by a binary string of 0s and 1s. There are also many other available encodings. For example, in our case, the type of encodings is a matrix, with each column as a permutation of a set of finite integers.

Evolution usually begins with a randomly generated population set of individuals and is an iterative process. The population in each iteration is named as a generation. In every iteration, several "children" solutions are generated from crossover between randomly selected parent solutions from the current population, and the fitness of each individual in the generation is assessed. All of the parent and children solutions compete for surving to the next generation. Fitness is usually the value of an objective

function of the optimization problem which quantifies the suitability of the solutions. The more suitable solutions will more likely survive to the next generation. Usually, the iteration goes on until the maximum number of generations is achieved, or some satisfactory fitness level is reached.

In our problem, the encoding type of solutions should be a designs matrix representing the Latin hypercube design with n runs and p input variables. The initial population is generated randomly, allowing the entire range of possible Latin hypercube designs. The population size depends on the specific problem, but typically will contain several hundreds or thousands of possible designs. During every successive generation, there is a portion of the existing population selected through a fitness-based selection process, where fitter solutions (as measured by fitness function) are more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. We choose to use the classic roulette wheel selection (De Jong, 1975). It means that if f_i is the fitness by evaluating individual i in the population, its probability of being selected is $p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$, where N is the total number of individuals in the population.

After selection, we get a "parents" set. Based on this "parents" set, the next step is producing a "children" set of solutions using two operators as crossover and mutation.

The specific implementation of crossover and mutation operators, in this case, is as follows. Suppose we are searching for a Latin hypercube design with n runs and p variables. For the crossover, we randomly sample two "parent" designs from the "parents" set. We set a random integer k_i $(1 < k_i < n, i = 1,2,\cdots,p)$ as the cut-off point for each variable. To generate a "child", for the i^{th} variable, we remain the values in runs 1 to k_i from the first "parent". Next, we continue it with the values in runs $k_i + 1$ to n from the second "parent". Finally, we use a "repair" function to replace the repeated values in each variable with the missing values in a random order and get the "child" solution. For mutation operator, we randomly sample one "parent" design from the "parents" set. The method is to randomly switch two values for each variable. Since we only have two operators and all the children

solutions are generated by either crossover or mutation, the crossover proportion P_c and the mutation proportion P_m are chosen to satisfy $P_c + P_m = 1$. We chose the crossover rate to be between 0.6 to 1 based on our experiences and experiments.

The NSGA-II algorithm which is proposed by Deb et al. (2002), is a fast elitist non-dominated sorting genetic algorithm for multi-objective optimization. It provides a useful fitness function for multi-objective genetic search algorithms and can help find the Pareto Front criteria vectors in a simple way. The basic idea of this algorithm is that it gives every solution a rank as the fitness function. Rank 1 means this solution is currently the Pareto optimal solution, rank 2 means this solution is the Pareto optimal solution excluding solutions ranked as 1, etc. Since we always want to maximize the fitness value while evolving, we use the multiplicative inverse of the rank as the fitness function. The population evolves directly towards to the better solutions which are the Pareto optimal solutions.

The GA search algorithm stops when a prespecified convergence condition is met. In our problem, since our goal is to identify the Pareto front of all superior solutions, the search algorithm is set to terminate when the identified Pareto front no longer changes much. Note as a heuristic search algorithm, the GA algorithm does not guarantee to find the absolute optimal solutions, i.e. the Pareto front for our application. Generally, the longer the search algorithm runs, the better approximation to the true Pareto front can be identified. However, in real applications, we need to balance between the accuracy of the approximation and the computing time of the searching process. We recommend monitoring the change of the identified Pareto front for signaling the termination of the searching process (Lu et al. 2013; Lu & Anderson-Cook, 2013). The pseudo-code for detailing the newly developed Pareto front GA search algorithm is provided in Appendix B.

CHAPTER FOUR:

CASE STUDIES

Gaussian process (Gelman et al., 2014) models are very often used for analyzing computer experimental data. The Gaussian process can be seen as a machine learning algorithm with extreme focus on the small size of data. Because data from computer experiments are always very finite and Gaussian Process method has nice properties when fitting a model (or function) using very limited data. Hence, it becomes a popular method to fit a surface model for computer experiments.

In this chapter, we will simulate data from different Gaussian process models. The R package named "mlegp" (Dancik, 2013) will be used to fit Gaussian process models. The package obtains maximum likelihood estimates of Gaussian processes (GPs) for univariate and multi-dimensional outputs. It can help finding maximum likelihood estimates of Gaussian processes parameters for a vector (or matrix) of one (or more) response.

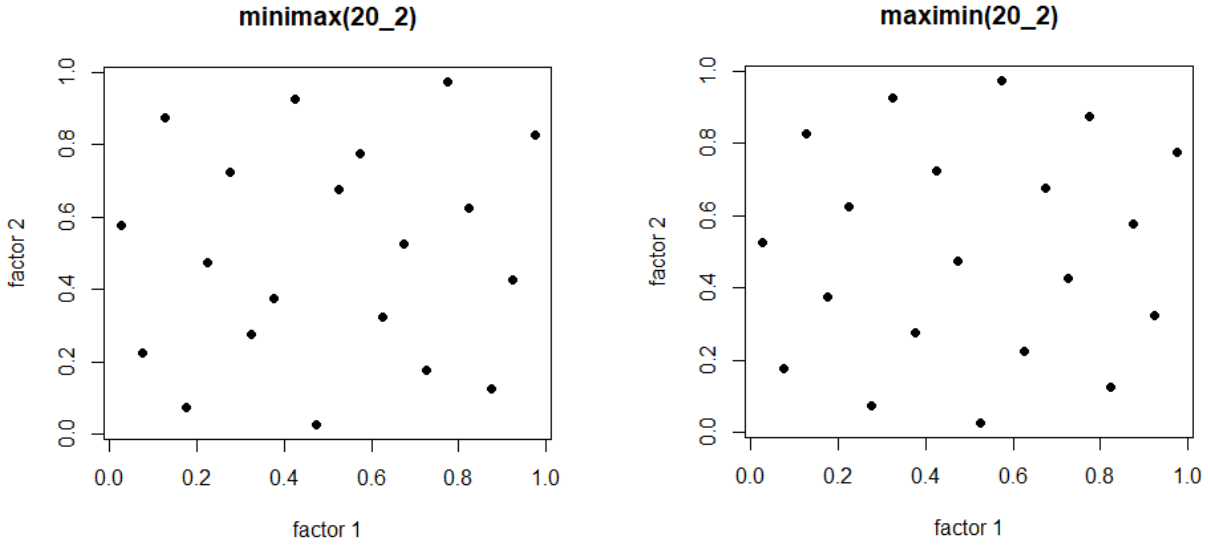
4.1 20 Run Design with 2 Input Variables

First, we start with a low dimension problem of constructing a computer experiment of 20 runs to explore a two-dimensional input space (with two input variables). For a 2-dimensional design, there is no need to consider maximin projection criterion because Latin Hypercube design has already ensured a uniform distribution in 1-dimension. Therefore, we only consider maximin distance and minimax distance criteria. Note that the two criteria tend to favor slightly different space-filling features. The maximin distance design tries hard to avoid nearby runs that could provide similar information. The minimax

distance design, on the other hand, makes sure that any arbitrary point in the design space is not extremely far from the closest design points and hence is likely to be predicted at a reasonable precision level.

Figure 2(a) and Figure 2(b) show some examples of the minimax-LHD and maximin-LHD for 20 runs with 2 variables. For the minimax-LHD in Figure 2(a), the minimax distance is 0.19. However, the minimax-LHDs are not unique. For our example with 20 runs and 2 variables, we found totally 11 non-isomorphic optimal minimax-LHDs with the same minimax distance at 0.19. Figure 2(b) shows an example of the maximin-LHD with 20 runs and 2 variables. Its maximin distance criterion measured by Eqn. (2.3) is 4.75, which is equivalent to 0.212 as the maximin distance in its original scale. This design is the only unique maximin-LHD we have found.

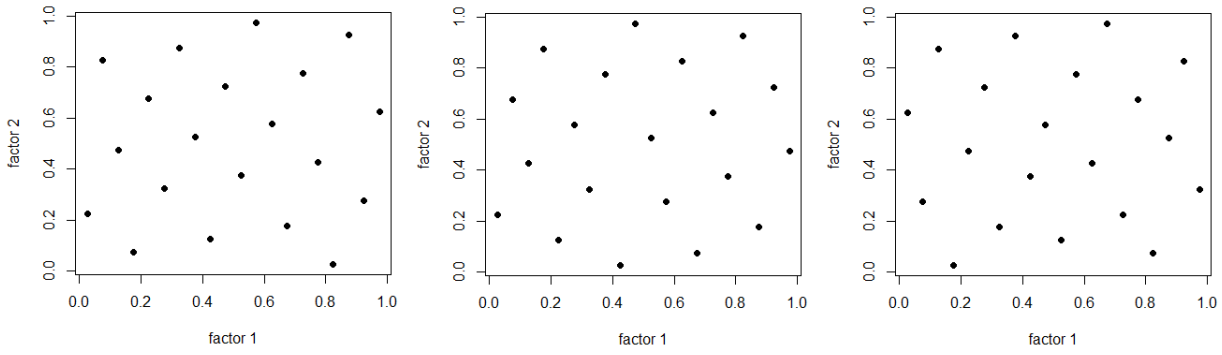
We can see from Figure 2(a) that some points in the minimax-LHD are more spread while others can be very close to each other. There are generally more design points located around the center of the entire design space (8 points inside the inner quarter square $[0.25,0.75]^2$). The maximin-LHD in Figure 2(b) has pairwise design points located more spread across the design region, and there are generally more points located near the corner or the edge of the design space (14 points outside the inner quarter square).



(a) Minimax-LHD (b) Maximin-LHD

Figure 2: (a) a sample of optimal Minimax-LHD and (b) a sample of optimal Maximin-LHD for 20 runs and 2 input variables.

Next, we use the two Pareto front search algorithms we developed in Chapter 3 to seek the optimal LHD considering both the maximin and minimax criteria. Both algorithms found 7 non-isomorphic LHDs as the Pareto optimal solutions. The minimax-LHD and the maximin-LHD found through the single criterion searches were included on the Pareto front. This is reassuring that our search algorithms did a reasonable job of exploring the complete design space and did not miss the extreme cases that are often easier to miss during the search process. The five designs excluding the minimax-LHD and the maximin-LHD are shown in Figure 3 (a)-(e).



(a) Design 1 (b) Design 2 (c) Design 3

(d) Design 4 (e) Design 5

Figure 3: (a)-(e) The total 5 Pareto optimal Latin hypercube designs for 20 runs and 2 variables considering maximin and minimax.

In the second stage of the Pareto front approach, the total seven designs (including the minimax-LHD and the maximin-LHD that have been found before) on the Pareto front are further evaluated using the additive desirability function for combining the maximin and minimax distances. Since the original

criteria values are not on a comparable scale, both criteria are changed to the expected scale between 0 and 1 for a fair comparison, i.e. the worst criteria value for designs on the Pareto front is scaled to 0 and the best value is scaled to 1. Due to the ambiguity of different weights for merging the two design criteria, a variety of different weight combinations through all possible range are explored and optimal designs based on the combined desirability for all possible weight combinations are summarized in the "mixture plot" (Lu et al. 2011) shown in Figure 4. We can see that design 4 is the absolute dominating choice for being optimal for a large region of weights between 0.1 and 0.7 for the maximin criterion, which indicates the strong robustness of design 4 being the optimal choice for a large proportion of possible user priorities between the two criteria. Design 3 is only optimal for a very narrow range of weights around maximin being weighted for 75%. Design 2 is optimal when the maximin criterion is weighted between 75% to 85%. Designs 1 and 5 are optimal when the maximin and minimax are valued less than 15% of the total weight, respectively. Considering that if a design criterion is included in the decision-making process after a careful consideration of its characteristics and value, it should be given at least a reasonable amount of weight, say no less than 20%. Hence, design 4 is the only most robust choice across the sensible range of possible weights.

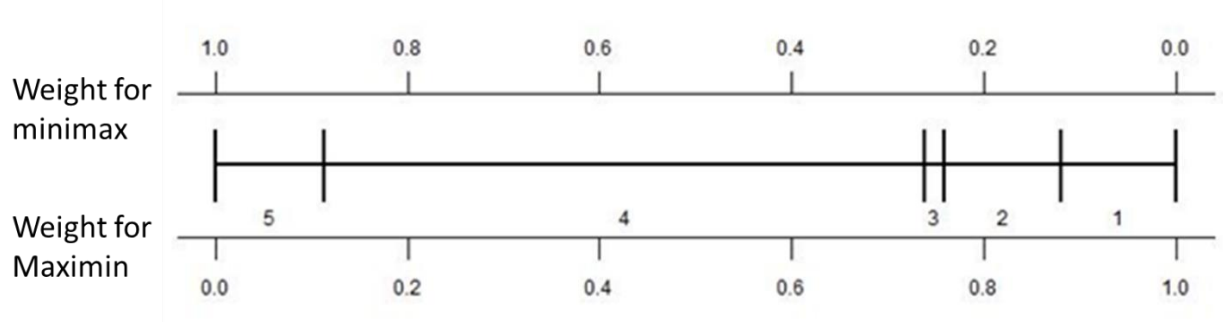
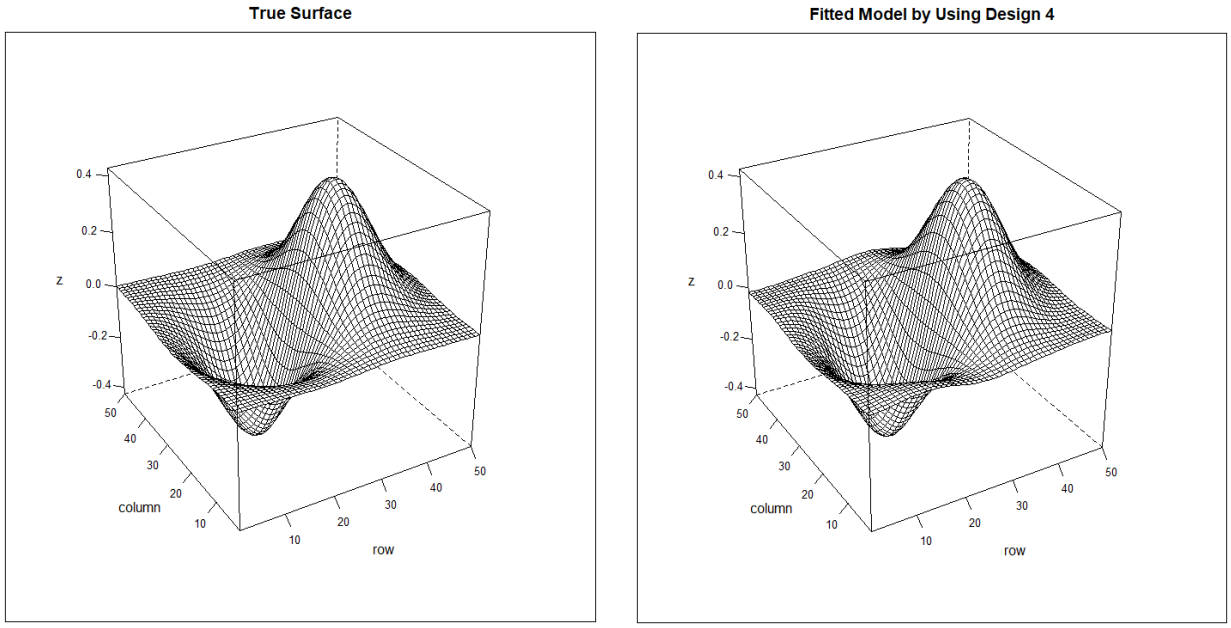


Figure 4: Mixture plots of the selected optimal designs based on using the additive desirability function.

Next, we examine actual performances of designs found on the Pareto front searching through simulation studies. The data of 20 runs are generated from a response model (Gramacy et al., 2009) of the form $z(x) = x_1 exp(-x_1^2 - x_2^2)$, $x_1, x_2 \in [-2,2]^2$. The true response model is shown in Figure 5 (a). By using package "mlegp", we fit a Gaussian process model with a constant mean to the 20 total observations. The fitted surface estimated by using design 4 is shown in Figure 5 (b). We can see from the plots, the estimated surface model is very similar to the true surface model. Note that we have only used 20 experimental data and the estimated model seems pretty good. This reassures that at least one of the Pareto optimal designs has nice performance.



(a) True response surface model of $z(x) = x_1 exp(-x_1^2 - x_2^2)$, $x_1, x_2 \in [-2,2]$

(b) Fitted model estimated by using design 4.

Figure 5. (a) The true response surface model of $z(x) = x_1 exp(-x_1^2 - x_2^2)$, $x_1, x_2 \in [-2,2]$ and (b) the fitted model estimated by using design 4.

We use the mean squared prediction error of the 100×100 grid points in the two-dimensional space to evaluate the design performance. Figure 6 shows the fraction of design space (FDS) plot for comparing the 7 designs on the Pareto front. Note each curve in the FDS plot represents a single design which shows for what fraction of the design space (the x-axis) the squared prediction error is at or below the value on the y-axis. Intuitively a design with small squared prediction error through the design space is deal which corresponds to a low flat curve in the FDS plot. By comparing the curves in Figure 6, we can see that Design 4, which is the final optimal design selected from the Pareto front approach performs consistently better than all the other designs. In addition, the maximin-LHD performs almost the worst among all 7 designs. The minimax-LHD performs similarly with a couple of other designs on the Pareto front, which are all better than the minimax-LHD but are consistently worse than Design 4.

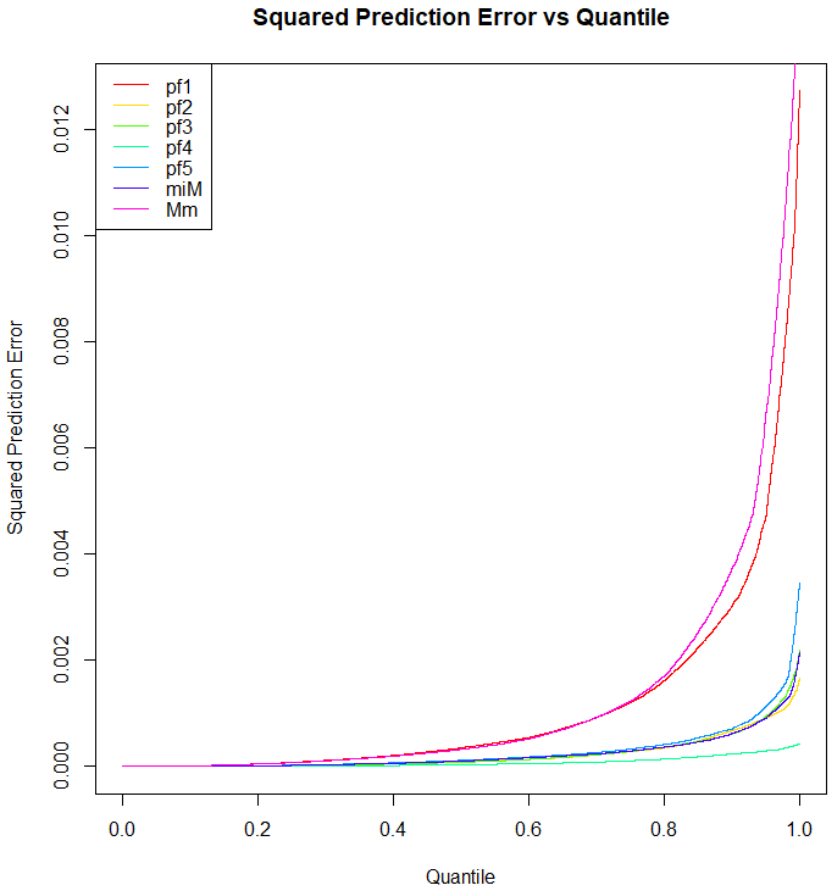
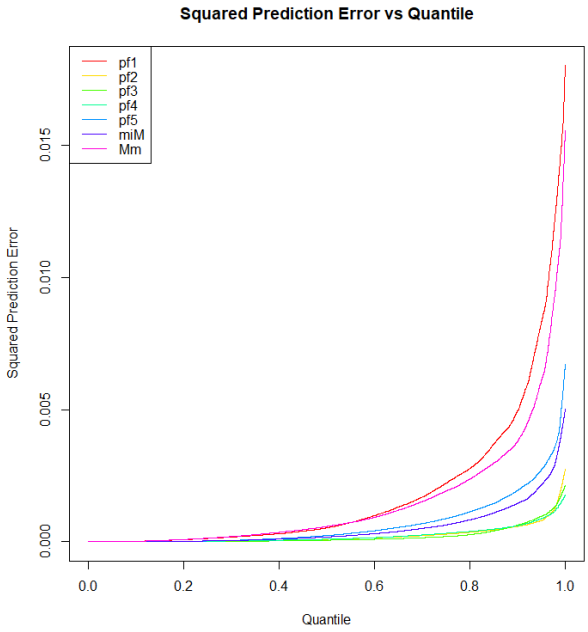
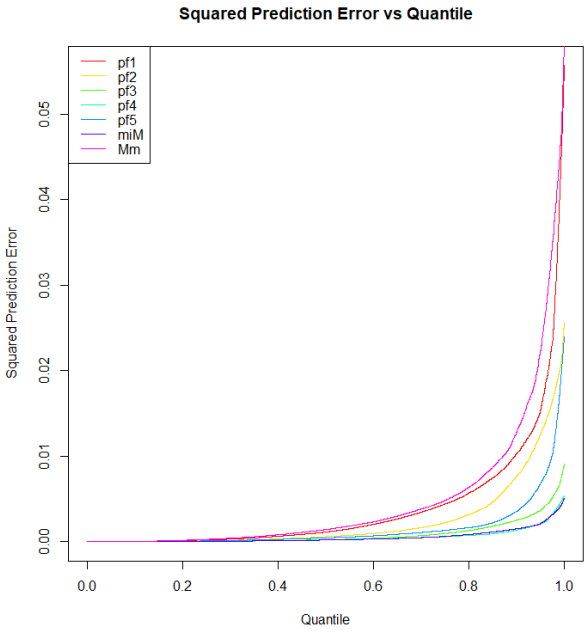


Figure 6. The fraction of design space (FDS) plot of the quantile squared prediction error over the design space for comparing the 7 Pareto optimal designs.

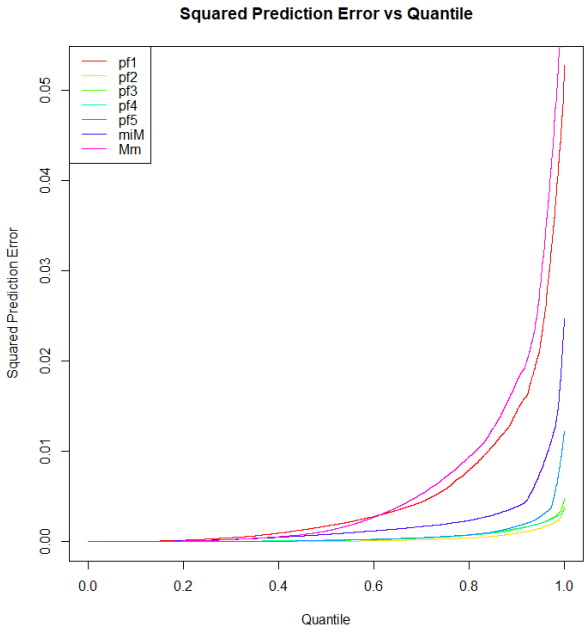
A few more alternative models are explored for comparing the Pareto optimal design performance, which are summarized in Figure 7 (a) – (d). Their corresponding true response surface models are shown in Figure 8 (a) – (d).



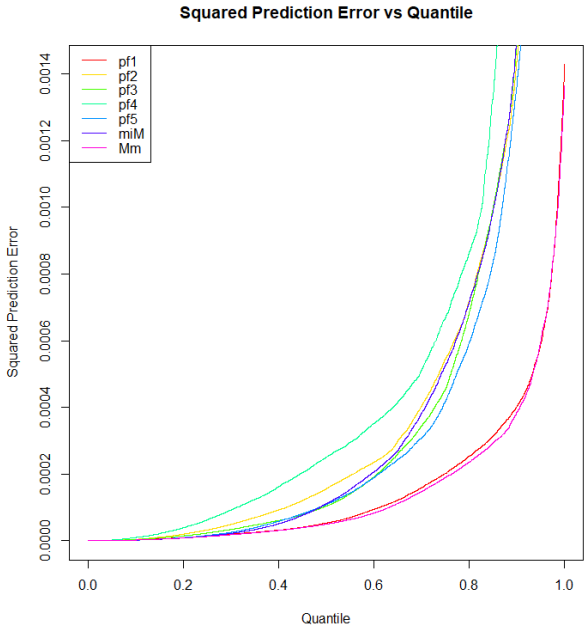
(a) Results for model $z(x) = x_2 exp(-x_1^2 - x_2^2)$, $x_1, x_2 \in [-2,2]$



(b) Results for model $z(x) = (2x_1 + x_2)exp(-x_1^2 - x_2^2)$, $x_1, x_2 \in [-2,2]$

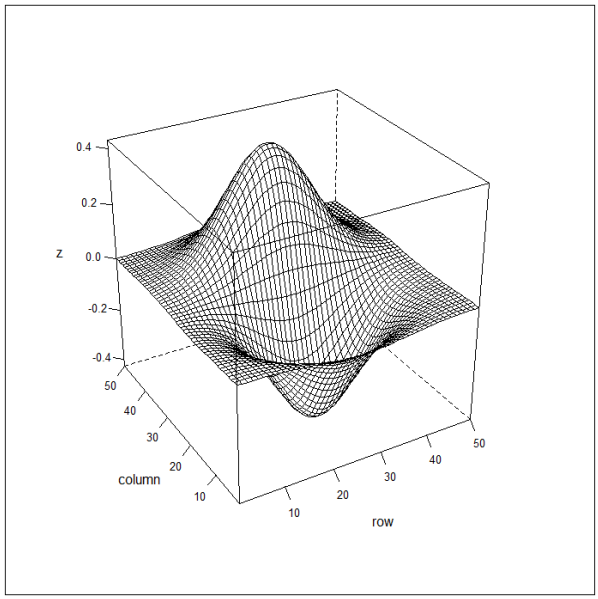


(c) Results for model $z(x) = (2x_1 - x_2)exp(-x_1^2 - x_2^2)$, $x_1, x_2 \in [-2,2]$

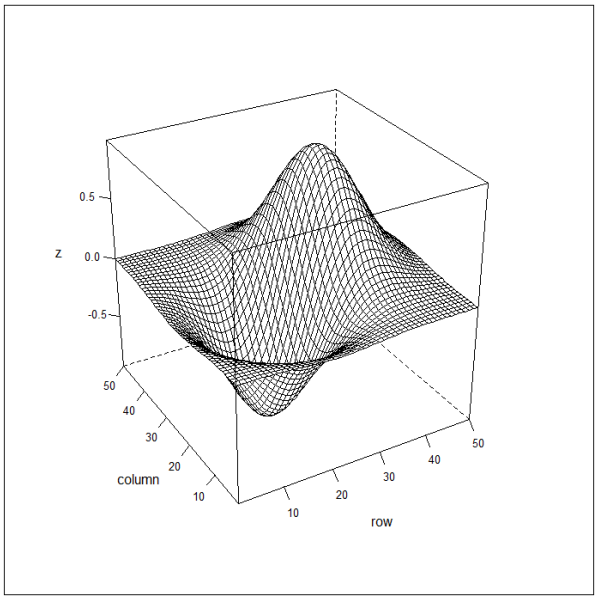


(d) Results for model $z(x) = x_1 x_2 exp(-x_1^2 - x_2^2)$, $x_1, x_2 \in [-2,2]$

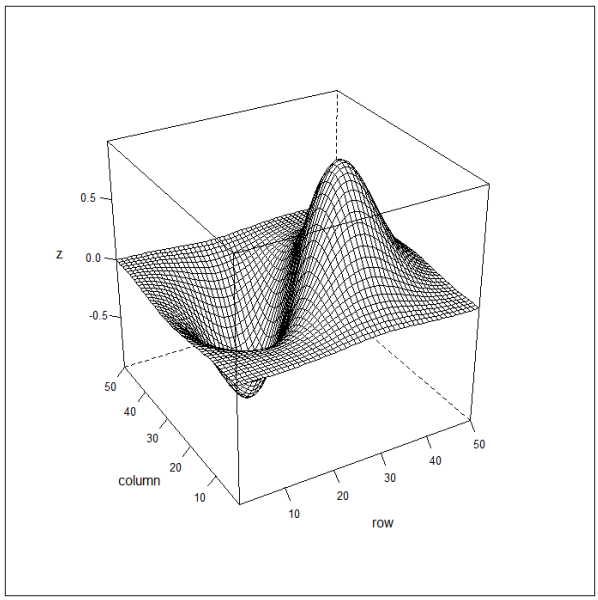
Figure 7. (a)-(d) Different designs' performance results for 4 different models.



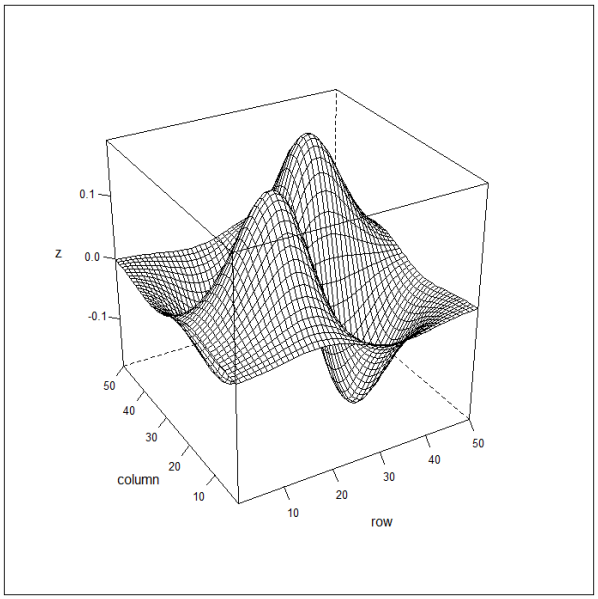
(a) Response surface for model $z(x) = x_2 exp(-x_1^2 - x_2^2)$, $x_1, x_2 \in [-2,2]$



(b) Response surface for model $z(x) = (2x_1 + x_2) exp(-x_1^2 - x_2^2)$, $x_1, x_2 \in [-2,2]$



(c) Response surface for model $z(x) = (2x_1 - x_2) exp(-x_1^2 - x_2^2)$, $x_1, x_2 \in [-2,2]$



(d) Response surface for model $z(x) = x_1 x_2 exp(-x_1^2 - x_2^2)$, $x_1, x_2 \in [-2,2]$

Figure 8. (a)-(d) The true response surfaces plots for 4 different models.

Out of the four additional scenarios we have explored, Design 4 is optimal or near optimal for three cases except the last scenario (shown in Figure 8(d)), which has much more complicated response

surface with more uneven and rapid changes of shape close to the edge of the design space in contrast to the relatively simple shape near the center. Hence, this model favors the maximin-LHD (with similar maximin distance value as Design 1) which has more runs located close to the corner and edge of the design space. The minimax criterion is not favored in this case due to the relatively simpler shape around the center of the design space. As a result, Design 4 has only average performance because it balances between the minimax and maximin criteria and hence lost its advantage as maximin-LHD is the absolute optimal under this model.

In summary, Design 4, which is the optimal design selected based on balancing multiple space-filling criteria, has shown the optimal or near performance across most of the simulation scenarios except one case where a single criterion has dominantly best performance. Also, the single criterion optimal designs based on either the maximin-LHD or minimax-LHD is rarely optimal (actually among the worst for four out of five cases) except for the last case with exceptionally complicated response surface near the edge of the design space.

4.2 100 Runs Design with 6 Input Variables

We have explored a scenario with two input variables, in which case good projection onto subspaces does not add extra value as there are only two dimensions (a single & the full dimension of two variables). Next, we explore another scenario in a higher dimension of with 6 input variables. In this case, when a subset of input variables is active, good space-filling over the subspace of active variables is desirable. Hence, we consider the projection property in design selection in addition to the maximin and minimax distance criteria.

We consider a design contains 100 runs to explore the design space $[0,1]^6$. This time we shall also consider the maximum projection criterion to ensure good projection property over all possible subspaces. Considering for this high dimensional design space, the computation of minimax criterion is not only time-consuming but also suffers from low precision due to the use of sampling strategy in

approximation of the large continuous design space, we choose to focus on only the maximin-LHD and MaxPro-LHD criteria which measures the space-filling across all dimensions of full and sub design spaces. Note when time allows and the minimax criterion is of particular interest, the developed MOLHD package can be used to select the optimal design considering all three criteria. However, the computing time for constructing a design of 100 runs while optimizing all three criteria can be quite long in order to fully populate the rich Pareto front for a high dimensional design of this large size.

Using the two Pareto front search algorithms we have developed, a set of 60 Pareto optimal designs was found when simultaneously optimizing both maximin-LHD and MaxPro-LHD criteria.

In the second subjective phase of the Pareto front approach for design selection, using the similar process as we have described in Section 4.1, the 60 designs are evaluated using the additive desirability function for combining the maximin-LHD and MaxPro-LHD criterion. The original criteria values are not on a comparable scale, so both two criteria are changed to the expected scale between 0 and 1 for a fair comparison, i.e. the worst criteria value for designs on the on the Pareto front is scaled to 0 and the best value is scaled to 1. Due to the ambiguity of different weights for merging the two design criteria, a variety of different weight combinations through all possible range are explored and optimal designs based on the combined desirability for all possible weight combinations are summarized in the "mixture plot" (using the same graphical tools as in Section 4.1) shown in Figure 9. We can see that designs 18 and 20 are the more dominating choices when both criteria are valued more evenly. Particularly, design 18 is optimal when the MaxPro criterion is slightly more emphasized, and design 20 is favored when the

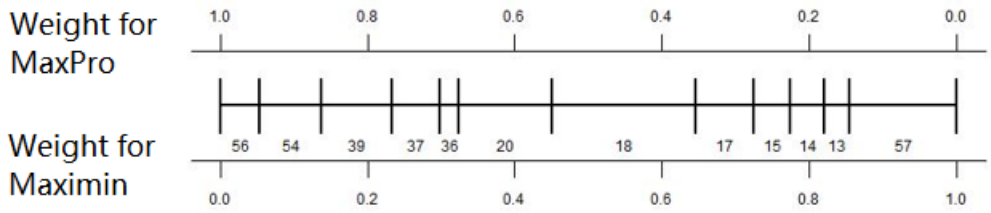


Figure 9. Mixture plots of the selected optimal designs based on using the additive desirability function.

maximin criterion is emphasized a bit more heavily. When maximin is considered more important, design 17 or 15 will be considered as optimal, while if MaxPro is valued more, then design 37 or 39 is the best choice. Design 57, 56 or 54 will not be considered unless one of the criteria is given more dominant weight. It worth noting that there are 48 designs are not selected as optimal for any weight combination and hence are no longer considered for design construction. Also, designs 18, 20, 17, 39 and 57 are more robust to weight uncertainty while designs such 36, 15, 14 and 13 are least robust which are optimal for only a highly narrow range of weights. Design 56 (or 54) and 57 are optimal when the MaxPro and maximin are valued more than 90% of the total weight, respectively.

Next, we examine the performance of more competing designs through simulation studies. Some data are simulated over the design space $[0,1]^6$ from a response model (Gramacy et al., 2009) of the form

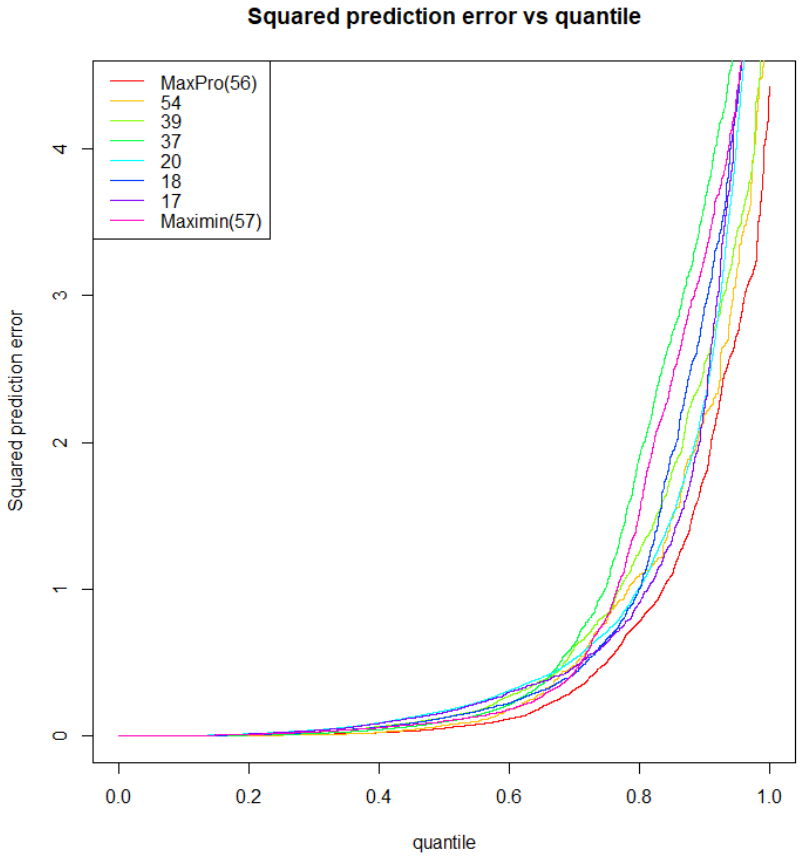


Figure 10. The fraction of design space (FDS) plot of the quantile squared prediction error over the design space for comparing the 8 of total 60 Pareto optimal designs.

$z(x) = exp(sin(0.9 \times (x_1 + 0.48)\text{^}10)) + x_2 \times x_3 + x_4$, $x_i \in [0, 1]$, $i = 1, 2, \cdots, 4$. Note that variable x_5 and x_6 are not active in this model. Hence, when generating the data for 6-dimension design, we can treat the model as from $z(x) = exp(sin(0.9 \times (x_1 + 0.48)\text{^}10)) + x_2 \times x_3 + x_4 + 0 \times x_5 + 0 \times x_6$, $x_i \in [0, 1]$, $i = 1, 2, \cdots, 6$. By using the R package "mlegp", we fit a Gaussian process model to these 100 data from our design points.

Similar to Section 4.1, we compute the mean squared prediction error of 5^6 grid points in the six-dimensional space to evaluate the design performance. Figure 10 shows the fraction of design space (FDS) plot for comparing 8 most promising designs from Figure 9 out of the 60 designs on the Pareto front. Note each curve in the FDS plot represents a single design which shows for what fraction of the design space (the x-axis) the squared prediction error is at or below the value on the y-axis. Intuitively a design with small squared prediction error through the design space is optimal which corresponds to a low flat curve in the FDS plot. By comparing the 8 curves in Figure 10, we can find that the selected 8 designs have competitive performance. However, design 56 has consistently best performance (smallest squared prediction error through the entire design space). This is not surprising as the true model spans only a subspace of 4 input variables and hence the best design emphasizing the projection property should have the best performance compared to other designs with more balanced performance between the full and subspaces.

A few more alternative models with the different sparsity of the input variables are explored to verify the design performance. Table 1 shows six samples of alternative models with varied complexity level. Their corresponding FDS plots are shown in Figure 11 (a)-(f).

From Table 1 & Figure 11 (a) - (f), with the 6 alternative models we have explored, the designs (such as design 57) emphasize MaxPro will perform better for a true model with fewer active variables and the designs emphasize full dimension performance (i.e. maximin criterion such as design 56) will behave better for a true model with all variables active. While for models of mediate sparsity levels, the

most promising designs identified from the Pareto front with more balanced performance across all dimensions (such as designs 37, 20, 18 & 17) tend to have near-optimal performance.

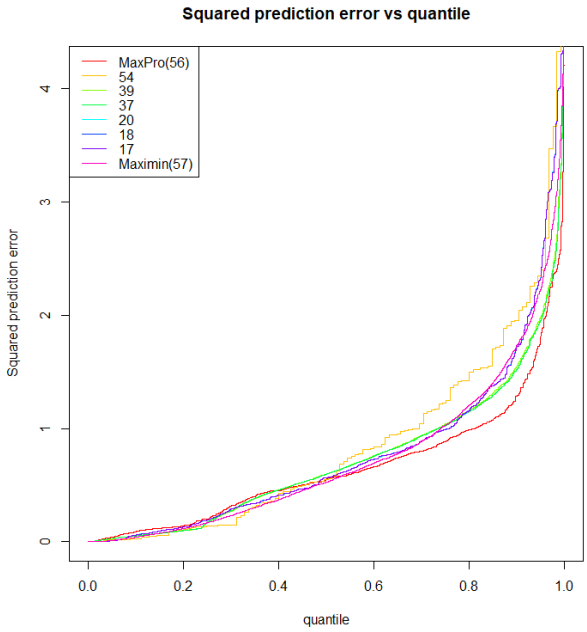
In summary, even though the best performance design can vary for different forms of the true model, however, the Pareto front approach provides an effective mechanism for finding the most promising designs for a variety of different weighting choices. All single criterion optimal designs are also included on the Pareto front, which bounds the performance of all Pareto optimal designs. The graphical summaries offer quantitative tools for evaluating and comparing design performance for different user priorities. Note the method can adapt easily if subjective matter expertise or information on the dimension and the shape of the response surface is available prior to conducting the experiment, which can be leveraged for improved decision-making.

Table 1. Six samples of alternative models with varied sparsity level.

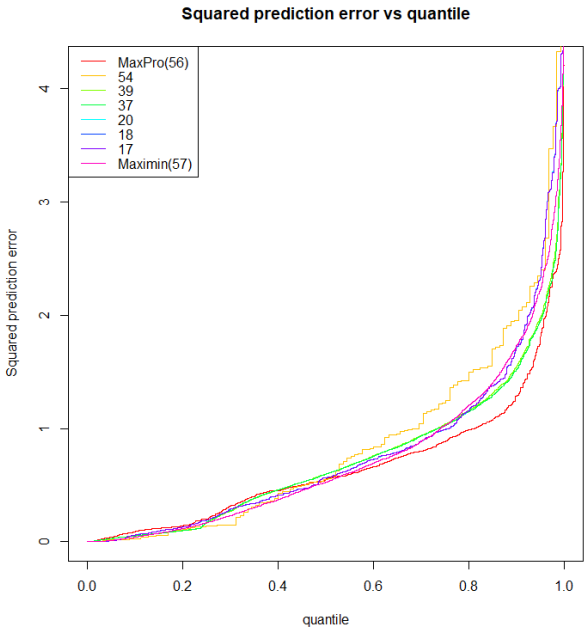
True Model	**Design Performance**
$z(x) = exp(sin(0.9 \times (x_1 \times 3 + 0.48)\text{^}10))$	Design 56, which is the MaxPro-LHD, has the best performance. The order of the performance of the 8 designs from best to worst is 56,37,39,57,20,18,17,54.
$z(x) = exp(sin(0.9 \times (x_1 + 0.48)\text{^}10)) + x_2$	Design 37 has the best performance. The order of the performance of the 8 designs from best to worst is 37,39,54,56,18,17,20,57.

Table 1 (Continued)

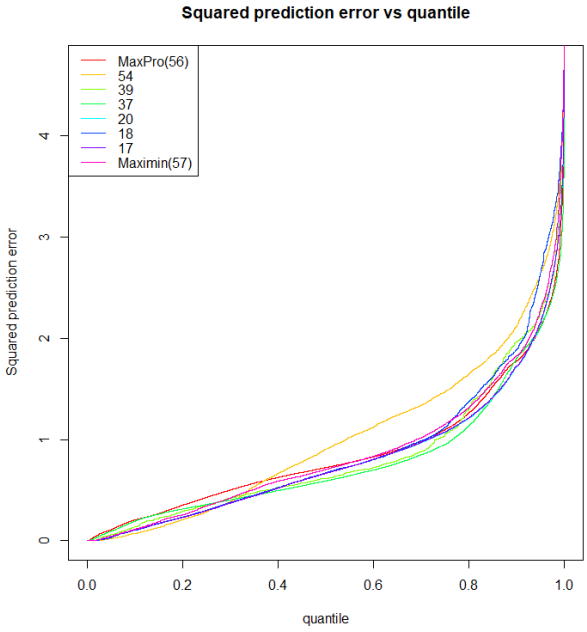
True Model	Design Performance
$z(x) = exp(sin(0.9 \times (x_1 + x_2 + x_3 + 0.48)\text{^}10))$	Design 37 has the best performance. The order of the performance of the 8 designs from best to worst is 37,20,17,39,56,57,18,54.
$z(x) = exp(sin(0.9 \times (x_2 + 0.48)\text{^}10)) + x_1 \times x_3 + x_4$	Design 20 has the best performance. The order of the performance of the 8 designs from best to worst is 20,17,18,39,37,56,57,54.
$z(x) = exp(sin(0.9 \times (x_1 + x_2 + x_3 + x_4 + x_5 + 0.48)\text{^}10))$	Design 37 has the best performance. The order of the performance of the 8 designs from best to worst is 37,54,17,18,20,57,39,56.
$z(x) = exp(sin(0.9 \times (x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + 0.48)\text{^}10))$	Design 57, which is the Maximin-LHD, has the best performance. The order of the performance of the 8 designs from best to worst is 57,20,18,17,54,56,38



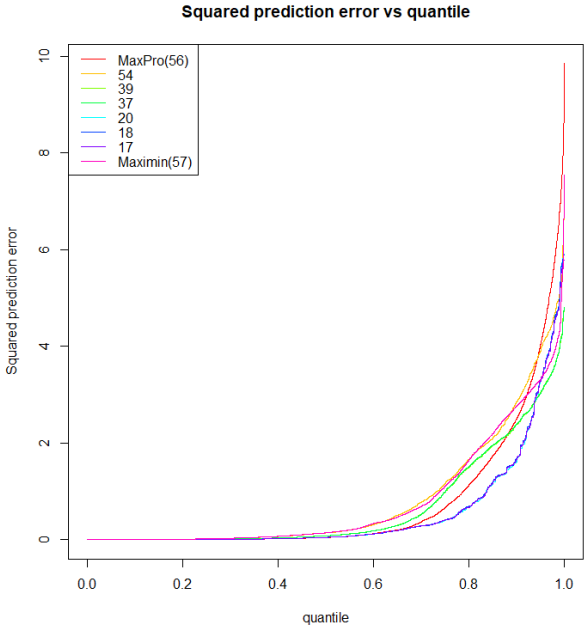
(a) FDS plot for model 1 in Table 1.



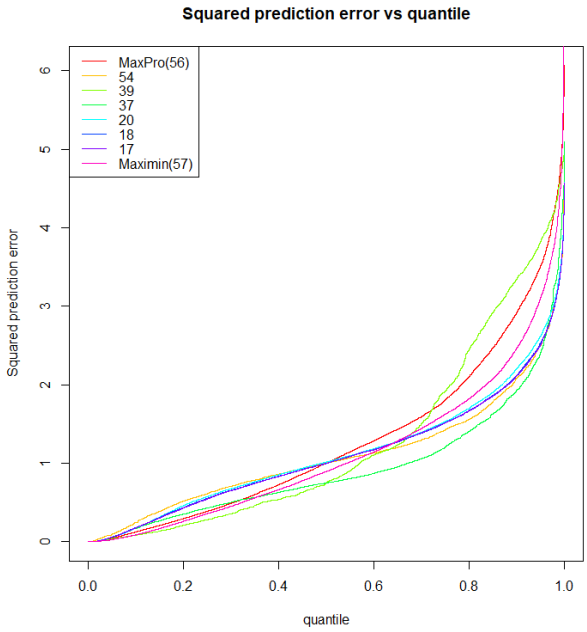
(b) FDS plot for model 2 in Table 1.



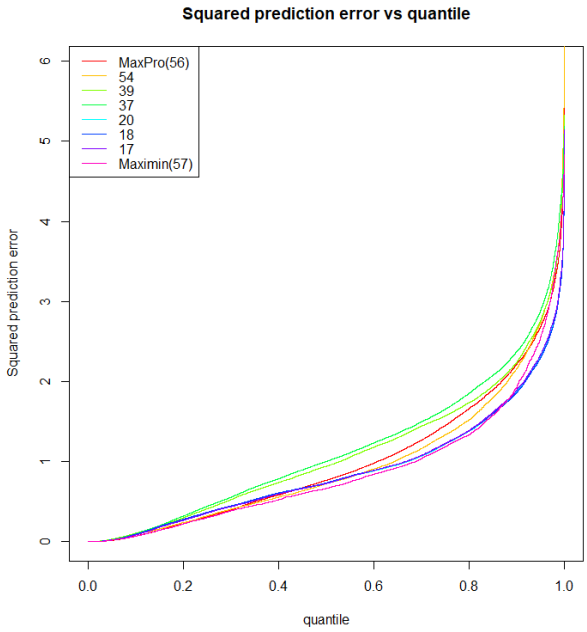
(c) FDS plot for model 3 in Table 1.



(d) FDS plot for model 4 in Table 1.



(e) FDS plot for model 5 in Table 1.



(f) FDS plot for model 6 in Table 1.

Figure 11. (a)-(f) The fraction of design space (FDS) plot of the quantile squared prediction error over the design space for comparing the 8 Pareto optimal designs for 6 additive models in Table 1.

CHAPTER FIVE:

CONCLUSION AND DISCUSSION

When designing computer experiments based on the geometry characteristics of the design locations, two aspects of the design performance are often of interest. One is space-filling property; the other is projection property. The former ensures all design locations are evenly spread through the design space and there is no large region that has not been covered by the experiment. The latter ensures space-filling property in any possible subspace when only a subset of the input variables actively affecting the response of the process. For the space-filling property, there are two criteria often used for capturing different aspects of "space-filling", i.e. to keep design points from being cluttered to lose efficiency, measured by the maximin criterion, and to ensure nowhere in the design space has poor prediction from being far from the experimental runs, measured by the minimax criterion. The projection property is measured by the MaxPro criteria, which calculates the average maximin distances over all subspaces from 2 to $p - 1$ dimensions.

The current practice has been focusing on building one of these design criteria into a search algorithm for optimizing the performance of Latin hypercube design. However, the fundamental differences in the emphasized criterion often lead to suboptimal designs that have the best performance on one aspect but relatively poor performance on other aspects. Simultaneously considering multiple design criteria in design construction and selection is desired to generate better designs with improved and more balanced performance on all aspects of the design characteristics.

This thesis has made innovative contributions on developing new computer algorithms for generating the Pareto optimal Latin hypercube designs based on simultaneously considering multiple space-filing and projection criteria and then utilizing the DMRCS structured multiple objective decision-making to guide the strategic selection of the best computer experiment that matches with the study goals. The newly developed computer search algorithms including both the column-wise exchange algorithm and the genetic algorithm provide effective mechanisms to eliminating noncontending choices and efficiently reducing the candidate solution space to allow experimenters quickly identify the most promising and sensible design choices and make tailored decision based on informative comparison and understanding of the trade-offs between options and the impacts of subjective user priorities. Case studies were conducted to illustrate the algorithms and the method and demonstrate the performance of the generated Pareto optimal designs. The column-wise exchange algorithm has been developed into R package and disseminated to the public through publication at the official software package website at CRAN (*URL* *http://CRAN.R-project.org/package=MOLHD*).

REFERENCES

1. Ba, S., Myers, W. R., & Brenneman, W. A. (2015). Optimal sliced Latin hypercube designs. *Technometrics*, *57*(4), 479-487.
2. Bandler, J. W., Cheng, Q. S., Dakroury, S. A., Mohamed, A. S., Bakr, M. H., Madsen, K., & Sondergaard, J. (2004). Space mapping: the state of the art. *IEEE Transactions on Microwave theory and techniques*, *52*(1), 337-361.
3. Chen, V. C., Tsui, K. L., Barton, R. R., & Meckesheimer, M. (2006). A review on design, modeling and applications of computer experiments. *IIE transactions*, *38*(4), 273-291.
4. Dancik, G. M. (2013). Mlegp: maximum likelihood estimates of gaussian processes. *URL http://cran. r-project. org/web/packages/mlegp/index. html*.
5. De Jong, K. A. (1975). Analysis of the behavior of a class of genetic adaptive systems.
6. Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, *6*(2), 182-197.
7. Derringer, G., & Suich, R. (1980). Simultaneous optimization of several response variables. *Journal of quality technology*, *12*(4), 214-219.
8. Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2014). *Bayesian data analysis* (Vol. 2). Boca Raton, FL: CRC press.
9. Gramacy, R. B., & Lee, H. K. (2009). Adaptive design and analysis of supercomputer experiments. *Technometrics*, *51*(2), 130-145.
10. Gramacy, R. B., & Lee, H. K. H. (2008). Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, *103*(483), 1119-1130.
11. Granville, V., Krivánek, M., & Rasson, J. P. (1994). Simulated annealing: A proof of convergence. *IEEE transactions on pattern analysis and machine intelligence*, *16*(6), 652-656.
12. Hou, R. & Lu, L. (2017). MOLHD: Multiple Objective Latin Hypercube Designs. R package version 0.1, *URL http://CRAN.R-project.org/package=MOLHD.*
13. Husslage, B. G., Rennen, G., van Dam, E. R., & den Hertog, D. (2011). Space-filling Latin hypercube designs for computer experiments. *Optimization and Engineering*, *12*(4), 611-630.
14. Jensen, F. (2017). Introduction to computational chemistry. *John wiley & sons.*
15. Johnson, M. E., Moore, L. M., & Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of statistical planning and inference*, *26*(2), 131-148.
16. Jones, L. K., & Byrne, C. L. (1990). General entropy criteria for inverse problems, with applications to data compression, pattern classification, and cluster analysis. *IEEE transactions on Information Theory*, *36*(1), 23-30.
17. Joseph, V. R. (2016). Space-filling designs for computer experiments: A review. *Quality Engineering*, *28*(1), 28-35.
18. Joseph, V. R., Gul, E., & Ba, S. (2015). Maximum projection designs for computer experiments. *Biometrika*, *102*(2), 371-380.
19. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.

20. Koehler, J. R., & Owen, A. B. (1996). 9 Computer experiments. *Handbook of statistics*, *13*, 261-308.
21. Law, A. M., Kelton, W. D., & Kelton, W. D. (2007). *Simulation modeling and analysis* (Vol. 3). New York: McGraw-Hill.
22. Li, W. W., & Jeff Wu, C. F. (1997). Columnwise-pairwise algorithms with applications to the construction of supersaturated designs. *Technometrics*, *39*(2), 171-179.
23. Lu, L., & Anderson-Cook, C. M. (2012). Rethinking the optimal response surface design for a first-order model with two-factor interactions, when protecting against curvature. *Quality Engineering*, *24*(3), 404-422.
24. Lu, L., Anderson-Cook, C. M., & Robinson, T. J. (2011). Optimization of designed experiments based on multiple criteria utilizing a Pareto frontier. *Technometrics*, *53*(4), 353-365.
25. Lu, L., Chapman, J. L., & Anderson-Cook, C. M. (2013). A case study on selecting a best allocation of new data for improving the estimation precision of system and subsystem reliability using Pareto fronts. *Technometrics*, *55*(4), 473-487.
26. Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, *26*(6), 369-395.
27. McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, *21*(2), 239-245.
28. Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.
29. Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.
30. Morris, M. D., & Mitchell, T. J. (1995). Exploratory designs for computational experiments. *Journal of statistical planning and inference*, *43*(3), 381-402.
31. Myers, R. H., Montgomery, D. C., & Anderson-Cook, C. M. (2016). Response surface methodology: Process and product optimization using designed experiments, 4th edition. *New Jersey: John Wiley & Sons, Inc*.
32. Pang, T. (1999). An introduction to computational physics.
33. Park, J. S. (1994). Optimal Latin-hypercube designs for computer experiments. *Journal of statistical planning and inference*, *39*(1), 95-111.
34. Pronzato, L., & Müller, W. G. (2012). Design of computer experiments: space filling and beyond. *Statistics and Computing*, *22*(3), 681-701.
35. Rasmussen, C. E. (2004). Gaussian processes in machine learning. In *Advanced lectures on machine learning* (pp. 63-71). Springer, Berlin, Heidelberg.
36. Sacks, J., Welch, W. J., Mitchell, T. J., & Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical science*, 409-423.
37. Sheiner, L. B., & Beal, S. L. (1981). Some suggestions for measuring predictive performance. *Journal of pharmacokinetics and biopharmaceutics*, *9*(4), 503-512.
38. Silvestrini, R. T., Montgomery, D. C., & Jones, B. (2013). Comparing computer experiments for the Gaussian process model using integrated prediction variance. *Quality Engineering*, *25*(2), 164-174.
39. Simpson, T. W., Lin, D. K., & Chen, W. (2001). Sampling strategies for computer experiments: design and analysis. *International Journal of Reliability and Applications*, *2*(3), 209-240.
40. van Rietbergen, B., Weinans, H., Huiskes, R., & Odgaard, A. (1995). A new method to determine trabecular bone elastic properties and loading using micromechanical finite-element models. *Journal of biomechanics*, *28*(1), 69-81.
41. Verlet, L. (1967). Computer" experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Physical review*, *159*(1), 98.
42. Washington, W. M., Weatherly, J. W., Meehl, G. A., Semtner Jr, A. J., Bettge, T. W., Craig, A. P., ... & Zhang, Y. (2000). Parallel climate model (PCM) control and transient simulations. *Climate Dynamics*, *16*(10-11), 755-774.

43. Waterman, M. S. (1995). *Introduction to computational biology: maps, sequences and genomes*. CRC Press.
44. Wu, C.F.J. and Hamada, S.M. (2009) *Experiments: Planning, Analysis, and Optimization*, Wiley, 2nd Edition.

APPENDICES

A. Pseudo code for the newly developed algorithms: Column-wise exchange algorithm

Input: Desired runs ***n*** with desired variables ***p***,

Number of random starts ***nstarts***

Maximum iteration times ***maxiter***

A set of weight vectors ***wtset***

Output: Pareto optimal designs with Pareto front values

pfdes = desmat0 = a random LHD

pfvals = desval0 = evaluate function (desmat0)

for each weight vector

desmat = desmat0

bestvals = desval0

i = 0

count = 0

temp = T

$c \in (0, 1)$

while (i < maxiter) and (count < times)

newdesmat = column-wise exchange (desmat)

newbestval = evaluate function (newdesmat)

criterion_difference = weighted-sum (newbestval) - weighted-sum (desval0)

if criterion_difference < 0

desmat = newdesmat; bestvals = newbestval; count = 0

else

if (temp > 0) and {random(0, 1) < exp(- criterion_difference / temp)}

desmat = newdesmat; bestvals = newbestval; count = 0

pfvals, pfdes = check(newbestval,newdesmat,pfvals,pfdes)

"check" function updates Pareto front values and Pareto optimal designs

i = i+1

count = count+1

temp = temp*c

return (pfvals, pfdes)

B. Pseudo code for genetic algorithm

Input: Desired runs ***n*** with desired variables ***p***,

Maximum population size ***maxpopsize***,

Maximum solutions to evaluate in a generation ***maxsoleval***,

Number of generations ***ngen***,

Proportion of crossover children ***pc***

Output: Pareto front values with Pareto optimal designs

pop = column bind (a set of random LHD (***n***, ***p***) with size equal to ***maxsoleval***)

popvals = evaluate function (pop)

popinfo = NSGA-II sort and rank function (popvals)

(NSGA-II sort and rank function returns a list that contains 2 objects. First is a sorted version of criteria vectors by which rank of Pareto front it is on; second is an ordered vector of numbers, which represent how many designs on different ranks of Pareto front.)

pop = sort pop in order of popinfo[[1]]

popvals = popinfo[[1]]

front1 = popvals[1:popinfo[[2]][1],]

frontDesigns = pop designs corresponding to front1

for g in 1: ***ngen***

nOFF = ***maxsoleval***

csize = nOFF * ***pc***

msize = nOFF – csize

for i in 1: csize

select two designs from pop using selection operator and generate a child using crossover operator

for j in 1: msize

select one design from pop using selection operator and generate a child using mutation operator

OFF = column bind (crossover children, mutation children)

OFFvals = evaluate function (OFF)

CPvals = row bind (popvals, OFFvals)

CP = column bind (pop, OFF)

CPinfo = NSGA-II sort and rank function (CPvals)

popvals = CPinfo[[1]] [1:***maxpopsize***,]

pop = CP designs corresponding to popvals

front1 = popvals[1:CPinfo[[2]][1],]

frontDesigns = CP designs corresponding to front1

return (front1, frontDesigns)

ABOUT THE AUTHOR

Ruizhe Hou is currently a graduate student majored in statistics at the University of South Florida. He has received his bachelor's degree in Science majored in Mathematics & Applied Mathematics in 2016, from Nankai University, Tianjin, P.R. China. During his graduate study towards the master's degree, he has received the M.V. Johns Jr. Scholarship in Spring 2018.