Construction of Orthogonal Designs and Baseline Designs

by

Ruwan Chamara Karunanayaka

M.Sc., Sam Houston State University, USA, 2014
B.Sc.(Hons.), University of Kelaniya, Sri Lanka, 2009

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
Department of Statistics and Actuarial Science
Faculty of Science

SIMON FRASER UNIVERSITY
Summer 2018

Approval

Name:	**Ruwan Chamara Karunanayaka**
Degree:	**Doctor of Philosophy (Statistics)**
Title:	**Construction of Orthogonal Designs and Baseline Designs**
Examining Committee:	**Chair:** Jinko Graham Professor
	Boxin Tang Senior Supervisor Professor
	Tim Swartz Supervisor Professor
	Jiguo Cao Internal Examiner Professor
	William J. Welch External Examiner Professor Department of Statistics University of British Columbia
Date Defended:	**July 23, 2018**

Abstract

In this thesis, we study the construction of designs for computer experiments and for screening experiments.

We consider the existence and construction of orthogonal designs, which are a useful class of designs for computer experiments. We first establish a non-existence result on orthogonal designs, generalizing an early result on orthogonal Latin hypercubes, and then present some construction results. By computer search, we obtain a collection of orthogonal designs with small run sizes. Using these results and existing methods in the literature, we create a comprehensive catalogue of orthogonal designs for up to 100 runs.

In the rest of the thesis, we study designs for screening experiments. We propose two classes of compromise designs for estimation of main effects using two-level fractional factorial designs under baseline parameterization. Previous work in the area indicates that orthogonal arrays are more efficient than one-factor-at-a-time designs whereas the latter are better than the former in terms of minimizing the bias due to non-negligible interactions.

Using efficiency criteria, we examine a class of compromise designs, which are obtained by adding runs to one-factor-at-a-time designs. A theoretical result is established for the case of adding one run. For adding two or more runs, we develop a complete search algorithm for finding optimal compromise designs. We also investigate another class of compromise designs, which are constructed from orthogonal arrays by changing some ones to zeros in design matrices. We then use a method of complete search for small run sizes to obtain optimal compromise designs. When the complete search is not feasible, we propose an efficient, though incomplete, search algorithm.

Keywords: Computer experiment; Design catalogue; Efficiency criterion; Latin hypercube; Minimum aberration; One-factor-at-a-time design; Orthogonal array; Rotation method; Search algorithm

Dedication

To my loving wife, Hiranya, and my daughter, Sanaya.

Acknowledgements

I would like to express my special appreciation and thanks to all the people who contributed in some way to the work described in this thesis.

First and foremost, I thank my senior supervisor, Dr. Boxin Tang, for believing in me. He contributed to a rewarding graduate school experience and his guidance gave me intellectual freedom in my work, while helping me find my limits. His encouragement in my writing, and his facility in funding were invaluable. I'll always appreciate the ways he helped me understand and negotiate this new language and culture. He has been a tremendous mentor in so many ways.

I also thank my supervisor, Dr. Tim Swartz, for his unconditional support and unfailing assistance in funding my research.

I also need to express my gratitude to my examining committee, Dr. William Welch and Dr. Jiguo Cao for their valuable contributions.

I would like to thank the faculty and staff of SFU's Department of Statistics and Actuarial Science for their help and support during this time.

I am grateful to all the professors in my graduate studies, especially Dr. Ananda Manage, Graduate Chair at Sam Houston State University. He helped introduce me to the world of graduate studies and make me comfortable here. Without his help, I may never have entered the PhD program at SFU. I also want to express my deep appreciation to Dr. Mallawa Arachchi of the University of Kelaniya, Sri Lanka, who set me on my path.

Finally, I would like to acknowledge my family and my friends. Most importantly, I owe a debt of gratitude to my wife, who crossed oceans and continents to accompany me, for her love and unyielding support. I am grateful to my dad for his encouragement right from the start of my studies. I would like to thank Dr. Jack Davis who helped me with proofreading some of my work. Thanks, too, to my Sri Lankan friends at SFU, Bhagya, Pulindu, Lasantha, Rajitha, Dilshani, Thilini, Himahansi, Nethangi and Nadheera, who made my time there a lot more fun and also, a special thanks goes to Dr. Harsha Perera for helping me with LaTex.

Table of Contents

List of Tables

Chapter 1

Introduction

1.1 Orthogonal Designs for Computer Experiments

Historically, the design of physical experiments was considered the gold standard of data collection for exploring a cause and effect relationships. Computer experiments have become popular and powerful tools in the last two decades due to the increase of computer power. Two key elements, the existence of mathematical theory and numerical methods, allow one to conduct computer experiments. They are useful when physical experiments are infeasible, or in some cases, impossible to perform. Computer experiments are used in scientific and technological developments in diverse areas such as engineering, meteorology, cosmology, nuclear physics, neuroprosthetics and many more. See, for example, Sacks, Welch, Mitchell and Wynn (1989) [31].

The main feature of a computer experiment is that it produces the same output with the same inputs, i.e., a *deterministic* answer, as opposed to a physical experiment. Therefore, none of the traditional principles of randomization, replication and blocking are relevant in computer experiments. In general, the functional form of the true relationship between the inputs and the outputs is unknown and com-

plicated. Therefore, designs for computer experiments should allow one to employ various types of modelling methods and should provide information about all portions of the experimental region. There are various types of designs that spread the points evenly throughout the region, and these are referred to as *space-filling* designs.

Latin hypercube designs, first introduced by McKay, Beckman and Conover (1979) [22], are considered to be the most commonly used class of space-filling designs for computer experiments. These designs have one-dimensional uniformity and do not have repeated runs. Optimality criteria such as orthogonality and maximum distance were used to obtain good designs. Iman and Conover (1982) [14], Owen (1994) [29], Tang (1998) [41], and Ye (1998) [44] introduced orthogonal and nearly orthogonal Latin hypercubes and provided some construction and computational results. These designs have zero or very low correlations in all two-dimensional projections and are useful in fitting data using main effect models.

Bingham, Sitter and Tang (2009) [3] introduced and studied orthogonal and nearly orthogonal designs for computer experiments by relaxing the condition that the number of levels is the same as the run size. This is a very rich class of designs, which include orthogonal Latin hypercubes at one extreme and two-level orthogonal arrays at the other. Sun, Pang and Liu (2011) [37] constructed orthogonal and nearly-orthogonal designs for computer experiments by rotating groups of factors of orthogonal arrays. Georgiou, Stylianou, Drosou and Koukouvinos (2014) [12] obtained 3-orthogonal U-type and non U-type designs with run sizes up to 100 by using known combinatorial structures. Investigation of orthogonal designs was further pursued by Liu and Liu (2015) [21], and Sun and Tang (2017) [39].

1.2 Baseline Designs for Screening Experiments

Factorial designs with two-level factors are useful in both theory and practice of physical experiments. They can be used to study the effects of multiple input variables. Experimental variables are called factors, and they can be quantitative or qualitative. Values of these factors are referred to as levels. A level combination of all factors is called a treatment or a run. A two-level full factorial design with m factors consists of 2^m runs. In practice, full factorial designs for large m are not economical. Therefore, subsets of full factorial designs are commonly used and are referred to as fractional factorial designs. As opposed to the factorial design, one can conduct one-factor-at-a-time design to investigate several factors one at a time.

Consider a full factorial design with m factors of two levels ± 1. Then, there are a total of $2^m - 1$ factorial effects, and these effects are the contrasts of treatment means. This type of parameterization using the symbols 1's and -1's is called an *orthogonal parameterization*, as all effect contrasts are mutually orthogonal. When there is a control or baseline level for each factor, it is quite natural to introduce a *baseline parameterization*. Under the baseline parameterization, 0 and 1 are the baseline and test level, respectively. The factorial effects are still contrasts of the treatment means, but these contrasts are not mutually orthogonal, unlike those under the orthogonal parameterization. Kerr (2006) [16], Banerjee and Mukerjee (2008) [2] and Stallings and Morgan (2015) [33] pointed out the increasing importance of the baseline parameterization.

Consider selecting an experimental design using the principle of "purpose" of the experiment [32]. The purpose can be defined in terms of optimizing a particular quantity. Then, we can ask, what are the settings of inputs that we should

observe the response to optimize this quantity? This approach to the selection of experimental design is referred to as an *optimal design*. The popular optimality criteria A, D, and E are to minimize the trace, the determinant and the maximum eigenvalue of the covariance matrix of the least square estimates of the parameters, respectively. In screening experiments, the intercept term is of little interest, and therefore we can simply ignore the first row and first column in the covariance matrix. Now, we can redefine the A, D, and E optimality criteria by A_s, D_s and E_s optimality criteria for the resulting matrix.

Fries and Hunter (1980) [10] introduced the minimum aberration criterion for selecting regular fractional factorial designs. Using one of the fundamental principles for factorial effects, the hierarchical ordering principle, Mukerjee and Tang (2012) [27] proposed a new minimum aberration criterion under the baseline parameterization for estimating main effects when interaction effects are non-negligible, and they also showed that orthogonal arrays of strength two are universally optimal for estimating main effects. Mukerjee and Tang (2012) [27] also considered one-factor-at-a-time designs for estimating baseline main effects and these designs allow unbiased estimation of main effects without any assumption about the absence of interactions.

1.3 Outline of the Thesis

An outline of the remainder of the thesis is as follows. Chapter 2 is devoted to new results on the existence and construction of orthogonal designs for computer experiments. Lin, Bingham, Sitter and Tang (2010) [19] showed that an orthogonal Latin hypercube does not exist when the run size is even but not a multiple of 4. A general non-existence result is obtained for orthogonal designs, which includes the aforementioned result as a special case. Based on the ideas in Lin, Mukerjee and

Tang (2009) [20] and Sun and Tang (2017) [38], we present some further results on the constructions of large orthogonal designs using an orthogonal array and a small orthogonal design.

A computer search for small orthogonal and nearly orthogonal designs is carried out for up to 18 runs. This computer search algorithm is very much similar to Lin's [18] adapted algorithm and we try to improve it further by applying a simulated annealing algorithm similar to that in Morris and Mitchell (1995) [26]. Using these and existing orthogonal designs combined with the construction methods in this chapter and other available methods, a comprehensive catalogue of orthogonal designs is created for up to 100 runs.

In Chapter 3, we introduce a first class of compromise designs under the baseline parameterization obtained by adding runs to one-factor-at-a-time designs. Because such designs contain one-factor-at-a-time designs, they should perform well in terms of minimizing the bias. Using efficiency criteria, we then find optimal designs from this class of compromise designs. The resulting optimal designs provide attractive alternatives to minimum aberration designs and one-factor-at-a-time designs. For the case of adding one run, optimal compromise designs are provided by a theoretical result. For the case of adding more than one run, we develop a complete search algorithm, which allows us to find optimal compromise designs for adding up to four runs to one-factor-at-a-time designs with $m \leq 20$ factors.

In Chapter 4, we propose a second class of compromise designs obtained from orthogonal arrays by changing some ones to zeros. These designs should perform better than orthogonal arrays in terms of minimizing the bias, because such designs move towards the one-factor-at-a-time designs. We then select optimal designs from the second class of compromise designs using both efficiency, A_s and

bias criterion, K_2. We use a complete search procedure to obtain such designs starting from changing a single one to zero in each column in minimum aberration designs. However, as the number of runs and the number of factors increase, the complete search procedure is not feasible. Therefore, we develop an efficient incomplete search algorithm to obtain optimal designs. In addition to the computer search procedures, we propose a systematic method to obtain compromise designs with some nice properties and these designs are actually the so-called balanced arrays. We examine the performance of our incomplete search algorithm by comparing its results with those from the complete search procedure.

Chapter 5 summarizes the thesis with a discussion of future research directions.

Chapter 2

On the Existence and Construction of Orthogonal and Nearly Orthogonal Designs

2.1 Introduction

Computer experiments are very useful statistical tools for investigating complex systems in science and engineering. Designing such experiments plays a vital role in subsequent statistical data analysis and model building. One attractive class of designs for computer experiments is given by orthogonal Latin hypercubes. Latin hypercubes were introduced by McKay, Beckman and Conover (1979) [22] and orthogonal Latin hypercubes were first considered by Ye (1998) [44]. Further results on orthogonal Latin hypercubes were provided in Lin, Mukerjee and Tang (2009) [20], Sun, Liu and Lin (2009) [36], and Georgiou and Efthimiou (2014) [11].

Generalizing orthogonal Latin hypercubes, Bingham, Sitter and Tang (2009) [3] introduced and studied orthogonal designs for computer experiments. Orthogonal designs do not require the number of levels to be the same as the run size, as in the case of Latin hypercubes. This is a very rich class of designs, which include orthogonal Latin hypercubes at one extreme and two-level orthogonal arrays at the other. Investigation of orthogonal designs was further pursued by Sun, Pang

and Liu (2011) [37], Georgiou, Stylianou, Drosou and Koukouvinos (2014) [12], Liu and Liu (2015) [21], and Sun and Tang (2017) [39].

Consider a design D of n runs with m factors of s levels. For convenience of studying orthogonality, the s levels are taken to be equally spaced and centred at zero. One convenient choice is given by $\{-(s-1)/2, -(s-3)/2, \ldots, (s-3)/2, (s-1)/2\}$. For example, when $s = 2$, the two levels are -0.5 and 0.5. When $s = 3$, the three levels are $-1, 0, 1$. This thesis considers designs where the s levels are equally replicated for each factor. Because of this, we must have $n = \lambda s$ for some integer $\lambda \geq 1$. Such designs are denoted by $\mathrm{D}(n, s^m)$. When $\lambda = 1$, a $\mathrm{D}(n, s^m)$ becomes a Latin hypercube, which we denote by $\mathrm{LH}(n, m)$. A $\mathrm{D}(n, s^m)$ is said to be orthogonal if $d_i^T d_j = 0$ for any two distinct columns d_i and d_j, in which case we use $\mathrm{OD}(n, s^m)$ to represent the design. Similarly, we use $\mathrm{OLH}(n, m)$ to denote an orthogonal Latin hypercube.

Nearly orthogonal designs are useful for finding good space filling designs for a large number of factors. Bingham, Sitter and Tang (2009) [3] proposed two measures to assess the near orthogonality of design D, namely, the maximum correlation $\rho_M(D)$ and the average squared correlation $\rho^2_{ave}(D)$, where $\rho_M(D) = \max_{i<j} |\rho_{ij}(D)|$, $\rho^2_{ave}(D) = \sum_{i<j} \rho^2_{ij}(D)/[k(k-1)/2]$ and $\rho_{ij}(D) = d_i^T d_j/[(d_i^T d_i)\cdot(d_j^T d_j)]^{1/2}$. Since the mean of the levels of each and every column of orthogonal designs is zero, $\rho_{ij}(D)$ is simply the correlation coefficient between the i^{th} and j^{th} column. Designs with small values of $\rho_M(D)$ and $\rho^2_{ave}(D)$ are considered as nearly orthogonal designs. Orthogonal designs have values zero for both of these measurements.

We also need the concept of orthogonal arrays in this thesis. An $n \times m$ matrix is said to be an orthogonal array of strength two with n runs for m factors of s levels, which are taken as $0, 1, \ldots, s-1$ unless stated otherwise, if the s^2 level

combinations occur with the same frequency in every submatrix of two columns. Such an array is denoted by $\mathrm{OA}(n, s^m, 2)$.

2.2 Theoretical Results

Lin, Bingham, Sitter and Tang (2010 [19], Theorem 2) showed that an $\mathrm{OLH}(n, m)$ with $m \geq 2$ does not exist if $n > 3$ is even but not a multiple of 4. This result can be generalized to orthogonal designs.

Theorem 1. *An OD$(\lambda s, s^m)$ with $m \geq 2$ does not exist if λ is odd and $s = 4k+2$ for any integer $k \geq 0$.*

When $\lambda = 1$, Theorem 1 reduces to the result of Lin, Bingham, Sitter and Tang (2010) [19]. Though the idea of proving Theorem 1 is similar to that of Lin, Bingham, Sitter and Tang (2010) [19], it is more challenging this time. We thus give a full proof.

Proof. Suppose an $\mathrm{OD}(\lambda s, s^m)$ with $m \geq 2$ exists for an odd λ and an $s = 4k+2$ for some integer $k \geq 0$. Let $(a_1, \ldots, a_n)^T$ and $(b_1, \ldots, b_n)^T$ be its two columns, which have levels given by $\pm(2j-1)/2$ where $j = 1, \ldots, s/2$ and $n = \lambda s$. As these two columns are orthogonal, we first have

$$0 = \sum_{i=1}^{n} a_i b_i = \sum_{j=1}^{s/2} \sum_{|a_i|=(2j-1)/2} a_i b_i,$$

which gives

$$\sum_{j=1}^{s/2} (2j-1) \left[\sum_{a_i=(2j-1)/2} b_i - \sum_{a_i=-(2j-1)/2} b_i \right] = 0. \tag{2.1}$$

Since $\sum_{i=1}^{n} b_i = 0$, we must have that

$$\sum_{j=1}^{s/2} \left[\sum_{a_i=(2j-1)/2} b_i + \sum_{a_i=-(2j-1)/2} b_i \right] = 0. \tag{2.2}$$

Combining Equations (2.1) and (2.2), we obtain

$$\sum_{j=1}^{s/2}\left[(j-1)\sum_{a_i=(2j-1)/2}(2b_i)-j\sum_{a_i=-(2j-1)/2}(2b_i)\right]=0. \tag{2.3}$$

Since $s=4k+2$, $(2b_i)$ is odd for all i, and the number of a_i's equal to $(2j-1)/2$ is λ, which is odd, the sum $\sum_{a_i=(2j-1)/2}(2b_i)$ must be odd. Similarly, $\sum_{a_i=-(2j-1)/2}(2b_i)$ must be also odd. Since $j-1$ and j have different parity,

$$(j-1)\sum_{a_i=(2j-1)/2}(2b_i)-j\sum_{a_i=-(2j-1)/2}(2b_i)$$

has to be odd. Since $s=4k+2$, $s/2=2k+1$ and so is odd. This means that the left hand side of equation (2.3) is a sum of an odd number of odd numbers, an obvious contradiction. The proof is completed. Q.E.D.

Example 1. (a) For $s=2$, Theorem 1 produces a well known result that an $OD(2\lambda,2^m)$ with $m\geq 2$ does not exist for any odd λ.
(b) Let $s=6$. Then Theorem 1 says that an OD$(6\lambda,6^m)$ does not exist for any odd λ and any $m\geq 2$. In particular, an OD$(18,6^m)$, an OD$(30,6^m)$ and an OD$(42,6^m)$ do not exist for any $m\geq 2$.
(c) For $s=10$, we have that an OD$(30,10^m)$, an OD$(50,10^m)$ and an OD$(70,10^m)$ do not exist for any $m\geq 2$.

Steinberg and Lin (2006) [34] proposed an elegant method for constructing orthogonal Latin hypercubes by rotating an orthogonal array in groups of factors. A general version of this method was presented in Pang, Liu and Lin (2009) [30]. The method was made more powerful by Lin, Mukerjee and Tang (2009) [20] by coupling an orthogonal array with a small orthogonal Latin hypercube, which allows orthogonal Latin hypercubes with many more columns to be constructed. Based

on the ideas in Lin, Mukerjee and Tang (2009) [20] and those in Sun and Tang (2017) [39] [38], we now present the constructions of three classes of orthogonal designs.

Let A be an OA$(n, s^{m_1}, 2)$ and B be an OD(s, p^{m_2}). If we replace the uth level in each column of A by the uth row of B, we obtain a design D_1, which must be an OD$(n, p^{m_1 m_2})$. Write $D_1 = (C_1, \ldots, C_{m_1})$ where C_i is the ith group of m_2 factors resulting from replacing the levels of the ith column of A by the rows of B. Any two columns of D_1 from two different groups C_{i_1} and C_{i_2} must be an OA$(n, p^2, 2)$. Let $c_{i1}, \ldots, c_{im_2}$ be the columns of C_i. We list the c_{ij}'s in the following order:

$$c_{11}, c_{21}, \ldots, c_{m_1 1}; c_{12}, c_{22}, \ldots, c_{m_1 2}; \ldots; c_{1m_2}, c_{2m_2}, \ldots, c_{m_1 m_2}.$$

We now take two columns at a time from the above list to obtain $q = [m_1 m_2/2]$ sets of two columns, where $[x]$ denotes the largest integer not exceeding x. Since each set of two columns come from different groups C_{i_1} and C_{i_2}, it is an OA$(n, p^2, 2)$. Note that if m_1 and m_2 are both odd, the last column $c_{m_1 m_2}$ is left unselected and all columns get selected otherwise.

Let these q sets of two columns be $C^{(1)}, C^{(2)}, \ldots, C^{(q)}$, which are all OA$(n, p^2, 2)$'s. Then

$$D_2 = (C^{(1)}R, \ldots, C^{(q)}R), \ R = \begin{pmatrix} p & -1 \\ 1 & p \end{pmatrix}$$

is an OD$(n, (p^2)^m)$, an orthogonal design with $m = 2q = 2[m_1 m_2/2]$ factors, each with p^2 levels. Note that all the OA$(n, p^2, 2)$'s referred to in this paragraph have their p levels given by $j - (p-1)/2$ for $j = 0, 1, \ldots, p-1$.

We next use the idea in Sun and Tang (2017) [38] to construct an orthogonal design with p^3 levels. For this purpose, we need to require p to be a prime power.

For simplicity of presentation, we only give details for the case where p is a prime. One can refer to Sun and Tang (2017) [39] for how to deal with the general case. For each $C^{(j)}$, let $B_j = C^{(j)} + (p-1)/2$ so that B_j is an OA$(n, p^2, 2)$ with levels $0, 1, \ldots, p-1$. Now define

$$D^{(j)} = \begin{bmatrix} B_j & B_j \\ B_j & 1+B_j \\ \vdots & \vdots \\ B_j & p-1+B_j \end{bmatrix},$$

where for example $1 + B_j$ means that 1 is added to all entries of B_j, and all calculations are modulo p. Let $E_j = D^{(j)} - (p-1)/2$. Then

$$D_3 = (E_1R, E_2R, \ldots, E_qR)$$

is an OD$(np, (p^3)^{2m})$, where

$$R = \begin{bmatrix} p^2 & -p & -1 & 0 \\ p & p^2 & 0 & 1 \\ 1 & 0 & p^2 & -p \\ 0 & -1 & p & p^2 \end{bmatrix}.$$

We summarize the above in a theorem.

Theorem 2. *Given an OA(n, s^{m_1}) and an OD(s, p^{m_2}), we have that*

(i) design D_1 above is an OD$(n, p^{m_1 m_2})$,

(ii) design D_2 above is an OD$(n, (p^2)^{2[m_1 m_2/2]})$,

(iii) design D_3 above is an OD$(np, (p^3)^{4[m_1 m_2/2]})$.

The result in Theorem 2(i) was mentioned but only in the text of Sun and Tang (2017) [39]. This is a useful result worth highlighting, which is what we are doing. Theorem 2(iii) can be thought of as an application of the general rotation method for Latin hypercubes in Sun and Tang (2017) [38] to the construction of orthogonal designs. The result in Theorem 2(ii) further extends a result in Sun and Tang (2017 [39], Propostion 3), which has already extended the original result of Lin, Mukerjee and Tang (2009) [20]. In Lin, Mukerjee and Tang (2009) [20], an even m_1 is required and whereas in Sun and Tang (2017) [39], an even $m_1 m_2$ is required.

Example 2. Take A to be an OA$(98, 7^{15}, 2)$ from Addelman-Kempthorne construction, and B to be an OD$(7, 7^3)$, which is an OLH$(7, 3)$ available in Lin, Mukerjee and Tang (2009) [20]. Then Theorem 1(i) gives an OD$(98, 7^{45})$. As $m_1 = 15$ and $m_2 = 3$, both odd, Theorem 2(ii) gives an OD$(98, 49^{44})$. To use the methods in Lin, Mukerjee and Tang (2009) [20] and Sun and Tang (2017) [39], we have to make one of m_1 and m_2 even. Deleting one column from A makes $m_1 = 14$. From this new A, we obtain an OD$(98, 49^{42})$. Alternatively, we can delete one column from B in order to have an even m_2 which is 2. This will give an OD$(98, 49^{30})$.

The current example is meant to be an illustration of Theorem 2. One can in fact obtain an OD$(98, 49^{48})$ by applying the method of Bingham, Sitter and Tang (2009) [3] to the OLH$(49, 24)$.

Example 3. Assuming the existence of a Hadamard matrix of order 4μ, let A be an OA$(4\mu, 2^{4\mu-2}, 2)$. Take B to be a trivial OD$(2, 2^1)$. Then Theorem 2(iii) produces a family of orthogonal designs with 8 levels, given by OD$(8\mu, 8^{8\mu-4})$. Taking $\mu = 1, 2, 3, 4, 5$, we obtain an OD$(8, 8^4)$, an OD$(16, 8^{12})$, an OD$(24, 8^{20})$, an OD$(32, 8^{28})$ and an OD$(40, 8^{36})$.

2.3 Computer Search and Design Catalogue

2.3.1 Orthogonal Designs

Lin (2008) [18] conducted a computer search of small orthogonal Latin hypercubes. Using a similar algorithm, we conduct a computer search of small orthogonal designs for up to 20 runs. The basic idea is to construct orthogonal designs by sequentially selecting columns. The details of the algorithm are omitted here and the interested reader is referred to the original presentation of Lin (2008) [18]. Instead, we focus on presenting the final products, the orthogonal designs obtained by the algorithm.

n	6	10	12	12	15	15	18
s	3	5	3	6	3	5	3
m	3	6	9	5	7	8	10

Table 2.1: A summary of small OD(n, s^m)s from computer search.

Table 2.1 is a summary of the results. To the best of our knowledge, all the orthogonal designs in Table 2.1 are new, because of which we give a full display of these designs in Tables 2.2, 2.3 and 2.4.

The existing literature has results for two cases in Table 2.1. Stylianou, Drosou, Georgiou, and Koukouvinos (2015) [35] obtained an OD$(12, 3^6)$ with six factors while ours in Table 2.1 gives an OD$(12, 3^9)$ which has nine factors. An OA$(18, 3^7, 2)$ with levels $-1, 0, 1$ is an OD$(18, 3^7)$. We have obtained an OD$(18, 3^{10})$. For all other parameter values of n and s in Table 2.1, there have been no orthogonal designs available thus far. Since orthogonal Latin hypercubes of $n = 6, 10$ and 18 runs

do not exist, the orthogonal designs of these run sizes in Table 2.1 thus provide attractive solutions if orthogonality is required of a design.

n											
6			12								
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0
0	-1	1	-1	-1	-1	0	1	1	1	0	0
1	0	0	0	-1	1	1	-1	0	1	-1	-1
-1	1	1	-1	0	1	1	1	-1	0	1	1
0	1	-1	1	-1	1	-1	1	1	-1	0	0
1	0	0	0	1	-1	1	1	0	-1	-1	-1
			1	0	-1	1	-1	1	0	1	1
			-1	1	1	0	-1	1	-1	0	0
			0	1	0	-1	0	0	1	-1	1
			1	0	0	0	0	-1	0	-1	1
			1	0	0	0	0	-1	0	1	-1
			0	1	0	-1	0	0	1	1	-1

Table 2.2: OD$(6, 3^3)$ and OD$(12, 3^9)$

The rotation method in Sun, Pang and Liu (2011) [37] allows construction of an OD$(8, 4^6)$, an OD$(12, 4^{10})$, an OD$(16, 4^{14})$, an OD$(18, 9^6)$ and an OD$(20, 4^{18})$. Using the OLH$(7, 3)$ in Lin (2008) [18] and the OD$(10, 5^6)$ in Table 2.4 in combination with the method of Bingham, Sitter and Tang (2009) [3], an OD$(14, 7^6)$ and an OD$(20, 5^{12})$ can be constructed. These theoretical results are hard to beat. Our algorithm can be applied to these parameter values but we have not found designs with more factors than these theoretically constructed designs, which is expected.

Many larger orthogonal designs can be constructed if the small orthogonal designs in Tables 2.2, 2.3 and 2.4 are used in conjunction with the method of Bingham, Sitter and Tang (2009) [3]. To illustrate, we give two examples.

n																				
15							18													
1	−1	−1	−1	−1	−1	−1	0	−1	−1	−1	−1	−1	−1	−1	1	1
1	−1	−1	0	0	0	1	−1	−1	−1	−1	−1	0	0	1	−1	−1
1	−1	0	0	1	1	0	0	−1	−1	0	1	0	0	−1	−1	−1
−1	−1	1	1	−1	−1	−1	1	−1	−1	0	1	1	1	1	1	1
0	−1	1	1	1	1	0	0	−1	1	1	−1	0	1	0	1	0
0	0	1	0	−1	0	1	0	−1	1	1	0	1	−1	0	−1	1
0	1	−1	1	−1	1	0	−1	0	0	1	0	1	0	0	1	−1
−1	0	−1	1	0	0	0	−1	0	0	1	1	−1	1	0	−1	0
−1	0	0	−1	−1	1	1	−1	0	1	0	1	−1	−1	−1	1	0
1	1	1	0	0	−1	1	−1	1	0	−1	1	1	−1	1	0	1
0	1	−1	1	1	−1	0	0	1	0	−1	0	1	1	−1	1	−1
0	1	0	−1	0	1	−1	1	0	1	−1	0	0	1	−1	−1	1
−1	0	0	−1	1	−1	1	1	0	1	−1	1	−1	0	1	0	−1
−1	0	0	−1	1	0	−1	1	0	1	0	−1	0	−1	1	0	−1
1	1	1	0	0	0	−1	0	1	0	0	−1	1	0	−1	−1	0
							1	1	−1	1	0	−1	0	0	0	0
							−1	1	0	0	−1	−1	1	1	0	1
							1	1	−1	1	0	0	−1	0	0	0

$n = 12$				
−2.5	−2.5	−2.5	−0.5	0.5
2.5	−0.5	0.5	−1.5	−1.5
0.5	1.5	−1.5	−2.5	2.5
−1.5	1.5	−0.5	2.5	−2.5
−0.5	−2.5	2.5	0.5	−0.5
2.5	0.5	0.5	1.5	−0.5
0.5	−1.5	−2.5	2.5	1.5
−2.5	2.5	2.5	0.5	2.5
−0.5	2.5	−1.5	−0.5	−2.5
−1.5	−1.5	1.5	−2.5	−1.5
1.5	−0.5	1.5	1.5	1.5
1.5	0.5	−0.5	−1.5	0.5

Table 2.3: $\mathrm{OD}(15, 3^7)$, $\mathrm{OD}(18, 3^{10})$ and $\mathrm{OD}(12, 6^5)$

n																
10						15										
0	−2	−2	−2	−1	0	0	−1	−2	0	−2	−1	1	2
1	−2	−1	2	0	−1	1	−1	1	−1	2	−2	−1	−2
−1	−1	0	1	1	1	−1	1	1	2	−1	−2	−1	0
−1	−1	1	−1	0	1	−1	1	0	2	−1	2	−2	−2
0	0	2	−2	2	−2	1	2	0	2	0	−2	1	1
−2	1	0	1	1	2	−2	−2	1	1	1	2	1	2
2	1	1	−1	−2	2	1	0	−1	−1	−2	1	−2	0
−2	2	−1	0	−2	−2	−1	1	−1	−2	2	−1	−1	1
1	0	2	2	−1	−1	2	−2	−2	1	0	0	2	−2
2	2	−2	0	2	0	2	2	2	0	1	2	2	0
						−2	−1	−1	1	2	0	0	−1
						0	0	0	−2	−1	1	0	−1
						0	2	−2	−1	1	1	0	1
						−2	0	2	−2	−2	−1	2	−1
						2	−2	2	0	0	0	−2	2

Table 2.4: $\mathrm{OD}(10, 5^6)$ and $\mathrm{OD}(15, 5^8)$

Example 4. Let D be the OD$(10, 5^6)$ in Table 2.4. Let H be a Hadamard matrix of order k. Then according to Bingham, Sitter and Tang (2009) [3], design $H \otimes D$ is an OD$(10k, 5^{6k})$. Taking $k = 2, 4, 8$ and 12, we obtain an OD$(20, 5^{12})$, an OD$(40, 5^{24})$, an OD$(80, 5^{48})$ and an OD$(120, 5^{72})$.

Example 5. Again, let H be a Hadamard matrix of order k. If we take D be the OD$(12, 3^9)$ in Table 2.2, then design $H \otimes D$ is an OD$(12k, 3^{9k})$. Taking $k = 2, 4, 8$, we obtain an OD$(24, 3^{18})$, an OD$(48, 3^{36})$ and an OD$(96, 3^{72})$.

n	s	m	Construction Method	n	s	m	Construction Method
6	3	3	Table 2.2	52	4	50	SPL
8	4	6	SPL	52	13	24	Lin & BST
10	5	6	Table 2.4	54	9	24	SPL
12	3	9	Table 2.2	54	27	12	Theorem 2(iii)
12	6	5	Table 2.3	56	4	54	SPL
12	4	10	SPL	56	7	24	Lin & BST
14	7	6	Lin & BST	56	8	52	Theorem 2(iii)
15	3	7	Table 2.3	60	3	28	Table 2.3 & BST
15	5	8	Table 2.4	60	5	32	Table 2.4 & BST
16	4	14	SPL	60	15	24	Lin & BST
16	8	12	Theorem 2(iii)	64	4	62	SPL
18	3	10	Table 2.3	64	8	60	Theorem 2(iii)
18	9	10	Lin & BST	64	16	48	ST
20	4	18	SPL	64	16	48	SL & BST
20	5	12	Table 2.4 & BST	64	32	32	SLL & BST
22	11	14	Lin & BST	66	33	32	SLL & BST
24	3	18	Table 2.2 & BST	68	4	66	SPL
24	4	22	SPL	68	17	24	Lin & BST
24	6	10	Table 2.3 & BST	72	3	40	Table 2.3 & BST
24	8	20	Theorem 2(iii)	72	4	70	SPL
24	12	12	Lin & BST	72	8	68	Theorem 2(iii)
25	5	12	Theorem 2(i)	72	9	40	Lin & BST
26	13	12	Lin & BST	72	36	6	SPL
27	9	12	SPL	75	5	16	Theorem 2(i)
28	4	26	SPL	75	25	16	Theorem 2(ii)
28	7	12	Lin & BST	76	4	74	SPL
30	3	14	Table 2.3 & BST	76	19	24	Lin & BST
30	5	16	Table 2.4 & BST	80	4	78	SPL
30	15	12	Lin & BST	80	5	48	Table 2.4 & BST
32	4	30	SPL	80	8	76	Theorem 2(iii)
32	8	28	Theorem 2(iii)	80	20	24	Lin & BST
32	16	24	SL & BST	81	9	50	Theorem 2(i)
34	17	12	Lin & BST	81	27	24	Theorem 2(iii)
36	3	20	Table 2.3 & BST	84	4	82	SPL
36	4	34	SPL	84	21	24	Lin & BST
36	9	20	Lin & BST	88	4	86	SPL
38	19	12	Lin & BST	88	8	84	Theorem 2(iii)
40	4	38	SPL	90	9	20	SPL & BST
40	5	24	Table 2.4 & BST	92	4	90	SPL
40	8	36	Theorem 2(iii)	92	23	24	Lin & BST
40	20	12	Lin & BST	96	3	72	Table 2.3 & BST
42	21	12	Lin & BST	96	4	94	SPL
44	4	42	SPL	96	6	40	Table 2.3 & BST
44	11	28	Lin & BST	96	8	92	Theorem 2(iii)
45	9	10	SPL	96	12	48	Lin & BST
46	23	12	Lin & BST	96	48	24	LBST & BST
48	3	36	Table 2.2 & BST	98	7	48	Theorem 2(i) & BST
48	4	46	SPL	98	49	48	LMT & BST
48	6	20	Table 2.3 & BST	100	4	98	SPL
48	8	36	Theorem 2(iii)	100	5	48	Theorem 2(i) & BST
48	12	24	Lin & BST	100	25	48	LMT & BST
49	7	24	Theorem 2(i)				
50	5	24	Theorem 2(i) & BST				
50	25	24	LMT & BST				

Table 2.5: A catalogue of OD(n, s^m) for $n \leq 100$

Using our computer search results and existing designs, in conjunction with our Theorem 2 and other existing methods in the literature, we now create a comprehensive catalogue of orthogonal designs for up to 100 runs. This is given in Table 2.5, in which the following abbreviations are used to save space: BST - Bingham , Sitter and Tang (2009) [3] , SPL - Sun, Pang and Liu (2011) [37] , ST - Sun and Tang (2017) [39], Lin - Lin (2008) [18], SL - Steinberg and Lin (2006) [34], LMT - Lin, Mukerjee, and Tang (2009) [20], SLL - Sun, Liu and Lin (2009) [36], LBST - Lin, Bingham, Sitter and Tang (2010) [19].

2.3.2 Nearly Orthogonal Designs

We use a computer search algorithm to obtain nearly orthogonal designs. The search algorithm is very much similar to the Lin's [18] adapted algorithm and we try to improve it further by applying a simulated annealing algorithm similar to that in Morris and Mitchell (1995) [26]. We use a linearly decreasing annealing temperature scheme to maximize our chance of finding a global optimum. In early iterations the probability of selecting a candidate that is evaluated to be worse than our comparison solution is relatively high. This permits the exploration of the whole solution space and to find the general region of the global optimum. In later iterations, when annealing temperature, Q, is small, the acceptance chance of a worse candidate solution is small as well. The movement between solutions becomes predominantly towards the nearest optimum. We selected this initial temperature Q_0 as a tuning parameter through trial-and-error.

In the first phase, we obtain an initial solution by sequentially adding columns. For each column, we evaluated up to $T1$ random permutation of d_k and choose the best candidate column under the ρ^2_{ave} optimality criterion. If $\rho^2_{ave} = 0$ then the orthogonal solution was obtained. This procedure is continued until the max-

imum number of orthogonal columns were achieved by the sequentially adding columns. After achieving the maximum number of orthogonal columns, we search for the next best candidate columns under the ρ^2_{ave} optimality criterion. If there are multiple candidate columns that are equally optimal, we select the first one. In the second phase, we search for improvements on this initial solution by way of simulated annealing. A step by step description of this algorithm as follows:

Step1: Let D_{OLD} be the initial design we found from Phase 1. This is the comparison solution until something replaces it. Let $\rho^2_{ave}(OLD)$ be the average squared correlation of D_{OLD}.

Step2: Among the columns of D_{OLD} that are not already part of orthogonal solution, select a column j, $k \leq j \leq n-1$ and swap all possible pairs within that column. If $\rho^2_{ave}(OLD) = 0$ then go to Step 1. Otherwise move on to Step 3.

Step3: The matrix with the swapped elements in that column, D_{NEW} is the candidate matrix and $\rho^2_{ave}(NEW)$ is the candidate solution.

Step4 (a): If $\rho^2_{ave}(NEW) \leq \rho^2_{ave}(OLD)$, accept the candidate solution as the comparison solution, and assign $\rho^2_{ave}(NEW) \rightarrow \rho^2_{ave}(OLD)$.

Step4 (b): If $\rho^2_{ave}(NEW) > \rho^2_{ave}(OLD)$, accept the candidate solution with probability $e^{-\frac{\{\rho^2_{ave}(NEW)-\rho^2_{ave}(OLD)\}}{Q_i}}$, where Q_i is i^{th} annealing temperature within Phase 2.

Step 5: If a new candidate solution has been selected, and the changed column j is part of an orthogonal solution, mark the column as orthogonal so that it is not selected to change in future iterations.

The algorithm is run until no improvement. Nearly orthogonal designs allow accommodation of more factors than orthogonal designs, thus providing a class of very useful designs for computer experiments. We have applied our algorithm to obtain a collection of nearly orthogonal designs for up to 18 runs. Tables 2.6 and 2.7 summarize the smallest ρ_{ave} value. Full display of the designs are given in Appendix A.

	$n=6$	$n=8$	$n=9$	$n=10$	$n=12$		
m	$s=3$	$s=4$	$s=3$	$s=5$	$s=3$	$s=4$	$s=6$
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0.144	0	0	0	0	0	0
5	0.209	0.078	0.053	0	0	0	0
6		0.100	0.086	0	0	0	0.007
7		0.134	0.096	0.027	0	0	0.011
8			0.109	0.045	0	0.033	0.018
9				0.074	0	0.044	0.029
10					0.070	0.061	0.034
11					0.099	0.084	0.049

Table 2.6: Best value for ρ_{ave} for designs $n = 6, 8, 9, 10$ and 12

	$n=14$	$n=15$		$n=16$		$n=18$	
m	$s=7$	$s=3$	$s=5$	$s=4$	$s=8$	$s=3$	$s=9$
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0.017	0	0	0	0.011	0	0.018
7	0.023	0	0	0	0.012	0	0.023
8	0.025	0.038	0	0	0.019	0	0.025
9	0.029	0.058	0.016	0.012	0.027	0	0.028
10	0.030	0.061	0.025	0.022	0.029	0	0.029
11	0.031	0.063	0.031	0.029	0.030	0.025	0.030
12	0.033	0.071	0.032	0.033	0.031	0.036	0.030
13	0.035	0.076	0.041	0.041	0.032	0.043	0.031
14		0.076	0.046	0.046	0.033	0.051	0.032
15				0.053	0.035	0.057	0.032
16						0.058	0.033
17						0.065	0.033

Table 2.7: Best value for ρ_{ave} for designs $n = 14, 15, 16$ and 18

Chapter 3

Compromise Designs Under Baseline Parametrization

3.1 Introduction

Two-level factorial designs are widely used in scientific and technological investigations to study the effects of a large number of factors on a response variable. Theory and practice of these designs mostly employ the familiar orthogonal parameterization, where main effects and interaction effects are defined using a set of orthogonal contrasts (Mee 2009 [23]; Wu and Hamada 2009 [42]; Cheng 2014 [6]). However, recent research (Kerr 2006 [16]; Banerjee and Mukerjee 2008 [2]; Stallings and Morgan 2015 [33]) points to the increasing importance of a less common but quite natural non-orthogonal baseline parameterization. Under the baseline parameterization, factorial effects are defined with reference to the baseline level. This chapter of the thesis considers two-level factorial designs under the baseline parameterization.

Mukerjee and Tang (2012) [27] showed that orthogonal arrays of strength two are universally optimal for estimating main effects. However, if significant interactions exist, the no-interaction model becomes biased. Mukerjee and Tang (2012) [27] proposed a minimum aberration criterion, which can be used to min-

imize the bias in the main effect estimates among all orthogonal arrays. Further work along this line has been carried out by Li, Miller and Tang (2014) [17], Miller and Tang (2016) [25] and Mukerjee and Tang (2016) [28]. In addition to minimum aberration designs, Mukerjee and Tang (2012) [27] also considered one-factor-at-a-time (OFAT) designs for estimating baseline main effects. Unlike orthogonal arrays, OFAT designs allow unbiased estimation of main effects without any assumption on the absence of interactions.

3.2 Introducing Compromise Designs

Let Z be a baseline design of n runs for m factors of two levels 0 and 1 with 0 denoting the baseline level and 1 the test level. Thus, design matrix Z is an $n \times m$ matrix with entries 0 and 1. We first consider a model that contains only an intercept and the main effects of the m factors. Under the baseline parameterization, the main effect of a factor is defined as the difference in the mean response when this factor changes from the baseline level to the test level while all other factors are held at the baseline level. Let a_n denote the vector of n ones. Then in matrix form, the main effect model is given by

$$Y = X\theta + \epsilon, \tag{3.1}$$

Y is the vector of the observed responses, $X = [a_n, Z]$ the model matrix, θ consists of the intercept and all the main effects, and ϵ is the vector of the random errors assumed to be uncorrelated with a constant variance σ^2. The least square estimate of θ is $\hat{\theta} = (X^T X)^{-1} X^T Y$ and its variance-covariance matrix is given by $\sigma^2 (X^T X)^{-1}$. For screening experiments, which is the situation we are considering, the intercept is of little interest and our focus should be on estimating the main

effects. Up to a constant σ^2, the variance-covariance matrix of the main effect estimates is $(X^TX)^{-1}_{(-1,-1)}$, which is the resulting matrix from deleting the first row and first column of $(X^TX)^{-1}$. Mukerjee and Tang (2012) [27] showed that design Z is universally optimal and thus A_s, D_s and E_s optimal for estimating the main effects among all n-run designs if Z is an orthogonal array of strength 2. The A_s, D_s and E_s optimality criteria are defined as the ones that minimize the trace, the determinant and the maximum eigenvalue of $(X^TX)^{-1}_{(-1,-1)}$, respectively.

If the interactions cannot be ignored, the true model becomes

$$Y = X\theta + X_2\theta_2 + \cdots + X_m\theta_m + \epsilon, \tag{3.2}$$

where θ_j is the vector of all j-factor interactions and X_j the corresponding model matrix. Then the expected value of $\hat{\theta}$ under the true model is given by $E(\hat{\theta}) = \theta + (X^TX)^{-1}X^TX_2\theta_2 + \cdots + (X^TX)^{-1}X^TX_m\theta_m$. It is clear that $(X^TX)^{-1}X^TX_j\theta_j$ is the contribution to the bias in $\hat{\theta}$ due to the j-factor interactions. When the intercept is of little interest, our focus should be on the bias in the estimates of the main effects. Now let C_j be the matrix obtained by removing the first row of $(X^TX)^{-1}X^TX_j$ which corresponds to the bias in the intercept estimate. Under the effect hierarchy principle that lower order effects are more important than higher order effects, one should sequentially minimize some size measures of $C_2, C_3, \ldots, C_m$. One such measure is given by $K_j = tr(C_jC_j^T)$. Then the minimum aberration criterion is to sequentially minimize $K_2, K_3, \ldots, K_m$.

The baseline parameterization has an attractive property that there exist designs that allow unbiased estimation of the main effects even in the presence of some or all interactions. These are given by one-factor-at-a-time (OFAT) designs. The basic OFAT design consists of treatment combinations $(0, 0, \ldots, 0)$,

$(1, 0, \ldots, 0)$, $\ldots$,$(0, 0, \ldots, 1)$, which give a design of $n = m + 1$ runs. When $n > m+1$, the class of all OFAT designs is obtained if all n runs are selected from the basic OFAT design. Let $f_0, f_1, \ldots, f_m$ denote, respectively, the frequencies of the treatment combinations $(0, 0, \ldots, 0)$, $(1, 0, \ldots, 0)$, $\ldots$,$(0, 0, \ldots, 1)$ occuring in a general OFAT design. Mukerjee and Tang (2012) [27] showed that the A_s-optimal OFAT design can be obtained by minimizing $(mf_0^{-1} + \sum_{i=1}^{m} f_i^{-1})$ subject to $f_0 + \sum_{i=1}^{m} f_i = n$.

Example 1. For $n = 8$ and $m = 6$, the A_s optimal OFAT design, say Z_{OFAT}, is given by $f_0 = 2, f_1 = f_2 = \cdots = f_6 = 1$. Let us compare Z_{OFAT} with the minimum aberration design Z_{MA} using the A_s criterion and a bias criterion K_2, the leading term in the minimum aberration criterion. The two design matrices are displayed below:

$$Z_{OFAT} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad Z_{MA} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Table 3.1 provides the values of A_s and K_2 for the two designs. Clearly, Z_{MA} is better in minimizing the variance while Z_{OFAT} is better in minimizing the bias. Given the sharp contrasts between Z_{MA} and Z_{OFAT}, it would be very useful to have some designs available that are not optimal under either criterion but perform well under both.

Z_{MA}	$A_s = 3.0$	$K_2 = 10.5$
Z_{OFAT}	$A_s = 9.0$	$K_2 = 0.0$

Table 3.1: The A_s and K_2 values of Z_{OFAT} and Z_{MA} for $n = 8$ and $m = 6$

Instead of restricting to the runs in the basic OFAT design, we propose the consideration of a class of compromise designs obtained by adding any $(n-m-1)$ runs to the basic OFAT design. As all such designs contain the basic OFAT design, they should perform well under the K_2 bias criterion. Out of this class of designs, optimal compromise designs can then be selected using efficiency criteria. All three A_s, D_s and E_s criteria will be considered in this thesis.

The next section examines how to construct such optimal compromise designs. A theoretical result is established showing that the design obtained by adding the all-ones run is optimal under all three A_s, D_s and E_s criteria. For the case of adding more than one run, it appears quite difficult to obtain such theoretical results. Instead, we resort to computer to search for optimal compromise designs. The naive approach of considering all possible ways of adding the extra runs is very time-consuming and only allows us to find some optimal designs for adding two runs. By making use of the symmetry of the basic OFAT design, we develop a complete search algorithm that is quite powerful and enables us to find all optimal compromise designs for adding up to four runs and up to $m = 20$ factors.

3.3 Finding Optimal Compromise Designs

3.3.1 Adding one run to basic OFAT design

This section is devoted to finding the optimal compromise design for the case of adding one run to the basic OFAT design. Then the design matrix Z consists of $m + 1$ treatment combinations $(0, 0, \ldots, 0)$, $(1, 0, \ldots, 0)$, $\ldots$,$(0, 0, \ldots, 1)$ and an

extra run $u^T = (u_1, u_2, \ldots, u_m)$, where $u_i = 0, 1$, for $i = 1, 2, \ldots, m$. The design matrix is thus given by

$$Z = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ u_1 & u_2 & u_3 & \cdots & u_m \end{bmatrix}.$$

As the model matrix is $X = [a_n, Z]$, we obtain

$$X^T X = \begin{pmatrix} n & a_m^T + u^T \\ a_m + u & I_m + uu^T \end{pmatrix},$$

where I_m is the identity matrix of order m and a_m is the vector of m ones. Since we are interested only in the main effects, not the intercept term, we need the variance-covariance matrix, say M, of the main effect estimates, which can calculated as follows

$$M = (X^T X)^{-1}_{(-1,-1)} = \left[I_m + uu^T - n^{-1}(a_m + u)(a_m + u)^T \right]^{-1}. \tag{3.3}$$

The A_s, D_s and E_s criteria select designs by minimizing the trace, the determinant and the maximum eigenvalue of matrix M, respectively. We have the following general result.

Theorem 3. *Design Z given by adding the all-ones run $u^T = (1, 1, \ldots, 1)$ to the basic OFAT design is optimal under all three A_s, D_s and E_s criteria.*

Proof. Equation (3.3) becomes $M = A^{-1}$ if we define

$$A = I + uu^T - n^{-1}(a+u)(a+u)^T \tag{3.4}$$

where we write I for I_m and a for a_m for simplicity in the proof. The m eigenvalues $\tau_1, \ldots, \tau_m$ of matrix A in (3.4) can be obtained via $\tau_i = 1 + \lambda_i$ with $i = 1, \ldots, m$ from the m eigenvalues $\lambda_1, \ldots, \lambda_m$ of matrix

$$B = uu^T - n^{-1}(a+u)(a+u)^T. \tag{3.5}$$

As B has rank at most 2, it has at least $m-2$ eigenvalues equal to 0. Let these be $\lambda_3 = \cdots = \lambda_m = 0$. To find the other two eigenvalues λ_1 and λ_2, we differentiate three cases.

Case (i). For $u = (0, \ldots, 0)^T$, matrix B in (3.5) becomes $-n^{-1}aa^T$, which has rank 1. Therefore the other two eigenvalues of B are $\lambda_1 = 0$ and $\lambda_2 = -n^{-1}\|a\|^2 = -n^{-1}m$, where $\|a\|^2$ denotes the squared length of vector a.

Case (ii). For $u = (1, \ldots, 1)^T = a$, we obtain $B = aa^T - n^{-1}(2a)(2a)^T = (1 - 4n^{-1})aa^T$, which also has rank 1. Therefore, $\lambda_1 = 0$ and $\lambda_2 = (1 - 4n^{-1})m$.

Case (iii). In this case, u consist of k ones and $m-k$ zeros with $1 \leq k \leq m-1$. Note that cases (i) and (ii) correspond to $k = 0$ and $k = m$, respectively. When $1 \leq k \leq m-1$, u and $a+u$ are independent, and thus B in (3.5) has rank 2. Obviously, u is not an eigenvector. So, an eigenvector of B must have form $x = u + \alpha v$ for some real α where $v = a + u$. We will find λ_1 and λ_2 from $Bx = \lambda x$. As $B = uu^T - n^{-1}vv^T$ and $x = u + \alpha v$, we obtain

$$Bx = (\|u\|^2 + \alpha u^T v)u - n^{-1}(v^T u + \alpha\|v\|^2)v = \lambda(u + \alpha v).$$

Therefore $\lambda = \|u\|^2 + \alpha u^T v$ and $\alpha\lambda = -n^{-1}(v^T u + \alpha\|v\|^2)$. Combining these two equations for α and λ and noting that $\|u\|^2 = k$, $\|v\|^2 = 3k + m$ and $u^T v = 2k$,

we obtain

$$\alpha^2 + b\alpha + g = 0, \tag{3.6}$$

where $g = n^{-1}$ and $b = (k + 3gk + mg)/2k$. Let α_1 and α_2 be the two solutions to equation (3.6). Then the two eigenvalues λ_1 and λ_2 of B are therefore given by

$$\lambda_1 = k + 2k\alpha_1, \quad \lambda_2 = k + 2k\alpha_2. \tag{3.7}$$

In addition, we must have

$$\alpha_1 + \alpha_2 = -b, \qquad \alpha_1\alpha_2 = g. \tag{3.8}$$

We thus obtain that $\det(A) = \prod_{i=1}^{m} \tau_i = \prod_{i=1}^{m}(1 + \lambda_i) = (1 + \lambda_1)(1 + \lambda_2)$ as $\lambda_i = 0$ for $i = 3, \ldots, m$. By (3.7) and (3.8), we obtain

$$\det(A) = gk^2 + (1 - 3g - mg)k + 1 - gm. \tag{3.9}$$

Although the above expression for $\det(A)$ is obtained for $1 \leq k \leq m - 1$, we can easily verify that it also holds for cases (i) $k = 0$ and (ii) $k = m$. As a real-valued function of k, the expression in (3.9) is a quadratic function with a positive coefficient for the k^2 term and thus minimized at $-(1-3g-mg)/(2g) = 1/2$. This shows that $\det(A)$ is maximized at $k = m$ when k is only allowed to take values $0, 1, \ldots, m$. Thus $k = m$ minimizes $\det(M)$. The D_s optimality is now established.

We next prove the A_s optimality. The A_s criterion minimizes

$$\text{trace}(M) = \sum_{i=1}^{m} \frac{1}{\tau_i} = \sum_{i=1}^{m} \frac{1}{1 + \lambda_i} = m-2+\frac{1}{1 + \lambda_1}+\frac{1}{1 + \lambda_2} = m-2+\frac{2 + \lambda_1 + \lambda_2}{(1 + \lambda_1)(1 + \lambda_2)}.$$

Using (3.7), (3.8) and (3.9), we have

$$\text{trace}(M) - (m-2) = h(k) = \frac{p(k)}{f(k)} \tag{3.10}$$

where $f(k) = gk^2 + (1-3g-mg)k + 1 - gm$ and $p(k) = (1-3g)k + 2 - mg$. Again, although (10) is obtained for case (iii), one can verify directly that it holds for cases (i) and (ii) as well. Treating $h(k)$ as a real-valued function, we obtain $h'(k) = \frac{w(k)}{f^2(k)}$, where $w(k) = c_0k^2 + c_1k + c_2$ with $c_0 = -g(1-3g)$, $c_1 = -2(2-mg)g$ and $c_2 = (2-mg)mg - (1-3g)$. Since $w(k)$ is a quadratic function of k, the following three facts (a) $c_0 < 0$, (b) $w(k)$ is maximized at $-c_1/(2c_0) = -(2-mg)/(1-3g) < 0$, and $w(0) = (3m+2)/(m+2)^2 > 0$ imply that $w(k) > 0$ for $0 \le k < k^*$ for some $k^* > 0$ and $w(k) < 0$ for $k > k^*$. This shows that $h(k)$ is increasing in $0 \le k \le k^*$ and decreasing in $k > k^*$. Because of this property, $h(k)$ is minimized at $k = m$ out of the allowable k values $k = 0, 1, \ldots, m$ if $h(m) < h(0)$, which amounts to $(m+2-4mg)/(m+1-4mg) < (2-mg)/(1-mg)$. The latter inequality can be verified easily as $m > 1$. The A_s is now established.

The E_s criterion finds optimal designs by minimizing $\max\{1/(1+\lambda_1, \ldots, 1/(1+\lambda_m)\}$, which is equivalent to maximizing $\min(\lambda_1, \ldots, \lambda_m)$. Clearly, $\min(\lambda_1, \ldots, \lambda_m)$ is equal to $-mg$ for case (i), 0 for case (ii) and $\lambda_1 = k + 2k\alpha_1$ for case (iii) where α_1 is the smaller root of equation (3.6), which is given by $\alpha_1 = -b/2 - (b^2 - 4g)^{1/2}/2$ as $b > 0$. Therefore

$$\min(\lambda_1, \ldots, \lambda_m) = k\left[1 - (b + (b^2 - 4g)^{1/2}\right] \text{ for } 1 \le k \le m-1. \tag{3.11}$$

Since $b = (k + 3gk + mg)/2k$ depends on k, we make this explicit by introducing $b_k = (k + 3gk + mg)/2k$, which is a strictly decreasing function of k for $k \ge 1$. Thus $b_k > b_m$, which implies that $b_k + (b_k^2 - 4g)^{1/2} > b_m + (b_m^2 - 4g)^{1/2} = 1$ for

$1 \le k \le m-1$. By (11), we have that $\min(\lambda_1, \ldots, \lambda_m) < 0$ for $1 \le k \le m-1$. Recall that $\min(\lambda_1, \ldots, \lambda_m) = -mg < 0$ for case (i) ($k = 0$) and $\min(\lambda_1, \ldots, \lambda_m) = 0$ for case (ii) ($k = m$). Thus 0 is the maximum value of $\min(\lambda_1, \ldots, \lambda_m)$, which is attained at $k = m$. This completes the proof for the E_s optimality. Q.E.D.

Table 3.2 provides an illustration of Theorem 3, in which for $m = 6$, we evaluate the performances of all designs from adding one run to the basic OFAT design. As can be seen from the table, none of the A_s, D_s and E_s criteria is a strictly decreasing function of k. This partly explains why the proof of Theorem 3 is nontrivial.

k	A_s	D_s	E_s
0	9.00	4.00	4.00
1	11.50	4.00	6.92
2	9.00	2.00	4.56
3	7.13	1.00	2.76
4	6.14	0.57	1.83
5	5.59	0.36	1.31
6	5.25	0.25	1.00

Table 3.2: The A_s, D_s and E_s values of the designs from adding a run consisting of k ones and $m - k$ zeros to the basic OFAT design for $k = 0, 1, \ldots, m = 6$.

We next compare in Table 3.3 three designs: the minimum aberration (MA) design Z_{MA}, the optimal compromise design Z_C and the optimal OFAT design Z_{OFAT} in terms of both the efficiency and the bias. For simplicity, we only calculate A_s for the efficiency and K_2 for the bias. Comparisons are made for designs of $n = m+2$ runs with $m = 2, 6, 10, 14, 18$ factors, in which cases, the MA designs are available. We know that MA designs are the best in terms of efficiency while OFAT

designs are the best in minimizing the bias. In contrast, our optimal compromise designs Z_C, though not optimal under either criterion, are quite competitive under both criteria.

	A_s value			K_2 value		
m	Z_{MA}	Z_C	Z_{OFAT}	Z_{MA}	Z_C	Z_{OFAT}
2	2.00	2.00	3.00	0.50	0.50	0.00
6	3.00	5.25	9.00	10.50	3.16	0.00
10	3.33	9.13	15.00	32.50	5.32	0.00
14	3.57	13.09	21.00	54.13	7.38	0.00
18	3.60	17.06	27.00	112.50	9.41	0.00

Table 3.3: Comparisons of A_s and K_2 values of three designs Z_{MA}, Z_C and Z_{OFAT} for $m = 2, 6, 10, 14$ and 18 factors.

A by-product of Theorem 3 is a set of simple formulas for the A_s, D_s, E_s values and the leading bias term K_2 of the optimal design. This is given in a corollary.

Corollary 1. *The optimal design in Theorem 3 has*

$$D_s = \frac{m+2}{m^2-m+2}, \quad A_s = m-1+D_s, \quad E_s = 1, \quad K_2 = \frac{m^4(m-1)}{2(m^2-m+2)^2}.$$

3.3.2 Adding two or more runs to basic OFAT design

In view of the lengthy proof of Theorem 3, it appears that a theoretical derivation of optimal designs for the general case is extremely difficult if possible at all. Instead, we devote our effort to computer search of optimal compromise designs for adding $p \geq 2$ runs to the basic OFAT design for m factors.

A naive approach is to consider all possible ways of adding p runs to the basic OFAT design. With m factors, there are 2^m possible runs to choose from for just adding one run. For adding p runs, one would have to examine $(2^m)^p$ choices if the search is to be complete. This number 2^{mp} is 2^{30} which is already more than one billion even for $m = 15$ and $p = 2$, and becomes astronomically large for $m = 20$ and $p = 3$. We have in fact tried out this approach and only obtained some optimal designs for $p = 2$.

A much more efficient search is to work with columns instead of rows by taking advantage of the symmetry of the basic OFAT design. When adding p runs to the basic OFAT design of m factors, the design matrix has two parts: the top part being the $m + 1$ runs given by the basic OFAT design and the bottom part being the extra p runs, which form a $p \times m$ matrix with entries 0 and 1. Let U denote this matrix consisting of the added p runs. Because of the symmetry of the basic OFAT design, the whole design remains essentially the same when the columns of U are arbitrarily permuted. This observation is the key to our development of an efficient search.

Any column of U is a binary vector of length p. There are in total 2^p such binary vectors and let them be $\omega_1, \ldots, \omega_{2^p}$. Let x_j be the number of times ω_j occurs in U for $j = 1, \ldots, 2^p$. Then two different U's can be obtained from each other by column permutation, if their sets of x's values are the same. This means that we only need to consider U's with different sets of x's values when searching for optimal compromise designs. Note that $\sum_{j=1}^{2^p} x_j = m$. Given a set of solutions $x_1, \ldots, x_{2^p}$ satisfying $\sum_{j=1}^{2^p} x_j = m$, one such U can be generated by collecting in any order ω_j repeated x_j times for every $j = 1, \ldots, 2^p$.

The following is a detailed description of our computational search algorithm for finding optimal compromise designs when adding p runs to the basic OFAT design.

Step 1. Systematically generate all nonnegative integer solutions to equation $\sum_{j=1}^{2^p} x_j = m$. This is done as follows. First let $x_1 = 0, 1, \ldots, m$. Given $x_1, \ldots, x_j$, let $x_{j+1} = 0, 1, \ldots, m - \sum_{i=1}^{j} x_i$, where $j = 1, \ldots, 2^p - 1$.

Step 2. For each solution $(x_1, \ldots, x_{2^p})$, generate matrix U by collecting $\omega_1, \ldots, \omega_{2^p}$ with ω_j repeated x_j times. Then stack Z_0 on top of U to get design matrix Z, where Z_0 is the matrix of the basic OFAT design. Appending a column of all ones to Z, we obtain the model matrix X.

Step 3. Evaluate the design using all the three criteria A_s, D_s and E_s.

With this efficient, yet still complete, search algorithm, we are able to obtain optimal compromise designs for all $m \leq 20$ factors and adding $p = 2, 3, 4$ runs. We have used all three A_s, D_s and E_s criteria in our search. While the same optimal designs are found under all three criteria in some cases, there are also many cases where the three criteria give different designs. It is remarkable that in all cases, A_s, D_s and E_s optimal designs only depend on a small subset of ω vectors. A run-down of the results for $p = 2, 3, 4$ is given below.

m	4	5	6	7	8	9	10	11	12
U	$\omega_1^2\omega_2^2$	$\omega_1^2\omega_2^3$	$\omega_1^3\omega_2^3$	$\omega_1^3\omega_2^4$	$\omega_1^4\omega_2^4$	$\omega_1^4\omega_2^5$	$\omega_1^4\omega_2^6$	$\omega_1^5\omega_2^6$	$\omega_1^5\omega_2^7$
m	13	14	15	16	17	18	19	20	
U	$\omega_1^6\omega_2^7$	$\omega_2^7\omega_3^7$	$\omega_2^7\omega_3^8$	$\omega_2^8\omega_3^8$	$\omega_2^8\omega_3^9$	$\omega_2^9\omega_3^9$	$\omega_2^9\omega_3^{10}$	$\omega_2^{10}\omega_3^{10}$	

Table 3.4: A_s optimal compromise designs for adding $p = 2$ runs.

m	4	5	6	7	8	9
U	$\omega_1^2\omega_2^2$	$\omega_1^3\omega_2^2$	$\omega_1^4\omega_2^2$	$\omega_1^4\omega_2^3$	$\omega_1^4\omega_2^4$	$\omega_1^5\omega_2^4$
m	10	11	12	13	14	15
U	$\omega_1^5\omega_2^5$	$\omega_1^6\omega_2^4\omega_3$	$\omega_1^6\omega_2^3\omega_3^3$	$\omega_1^6\omega_2^4\omega_3^3$	$\omega_1^7\omega_2^3\omega_3^4$	$\omega_1^7\omega_2^4\omega_3^4$
m	16	17	18	19	20	
U	$\omega_1^7\omega_2^5\omega_3^4$	$\omega_1^7\omega_2^5\omega_3^5$	$\omega_1^8\omega_2^5\omega_3^5$	$\omega_1^8\omega_2^5\omega_3^6$	$\omega_1^9\omega_2^5\omega_3^6$	

Table 3.5: D_s optimal compromise designs for adding $p = 2$ runs.

m	4	5	6	7	8	9	10	11	12
U	ω_1^4	ω_1^5	ω_1^6	ω_1^7	ω_1^8	ω_1^9	ω_1^{10}	ω_1^{11}	ω_1^{12}
m	13	14	15	16	17	18	19	20	
U	ω_1^{13}	ω_1^{14}	ω_1^{15}	ω_1^{16}	ω_1^{17}	ω_1^{18}	ω_1^{19}	ω_1^{20}	

Table 3.6: E_s optimal compromise designs for adding $p = 2$ runs.

In Tables 3.4, 3.5 and 3.6 , we present A_s, D_s and E_s, respectively, optimal designs from adding $p = 2$ runs to the basic OFAT design. The optimal designs depend on three ω vectors and let them be

$$\omega_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \omega_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \omega_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

In Tables 3.4, 3.5 and 3.6, a shortcut notation is to used to present the two extra runs where, for example, entry $\omega_1^2\omega_2^3$ means that matrix U is obtained by repeating ω_1 twice and ω_2 three times, that is,

$$U = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

In Tables 3.7 and 3.8, we present A_s, D_s and E_s optimal designs for adding $p = 3$ runs. This time, the optimal designs can be determined using four ω vectors as given below

$$\omega_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad \omega_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad \omega_3 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \quad \omega_4 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

m	4	5	6	7	8	9
U	$\omega_1\omega_2\omega_3^2$	$\omega_1\omega_2^2\omega_3^2$	$\omega_1^2\omega_2^2\omega_3^2$	$\omega_1^2\omega_2^2\omega_3^3$	$\omega_1^2\omega_2^3\omega_3^3$	$\omega_1^3\omega_2^3\omega_3^3$
m	10	11	12	13	14	15
U	$\omega_1^3\omega_2^3\omega_3^4$	$\omega_1^3\omega_2^3\omega_3^5$	$\omega_1^4\omega_2^4\omega_3^4$	$\omega_1^4\omega_2^4\omega_3^5$	$\omega_1^4\omega_2^5\omega_3^5$	$\omega_1^5\omega_2^5\omega_3^5$
m	16	17	18	19	20	
U	$\omega_1^5\omega_2^5\omega_3^6$	$\omega_1^5\omega_2^6\omega_3^6$	$\omega_1^6\omega_2^6\omega_3^6$	$\omega_1^6\omega_2^6\omega_3^7$	$\omega_1^6\omega_2^7\omega_3^7$	

Table 3.7: A_s and D_s optimal compromise designs for adding $p = 3$ runs.

m	4	5	6	7	8	9	10	11	12
U	ω_4^4	ω_4^5	ω_4^6	ω_4^7	ω_4^8	ω_4^9	ω_4^{10}	ω_4^{11}	ω_4^{12}
m	13	14	15	16	17	18	19	20	
U	ω_4^{13}	ω_4^{14}	ω_4^{15}	ω_4^{16}	ω_4^{17}	ω_4^{18}	ω_4^{19}	ω_4^{20}	

Table 3.8: E_s optimal compromise designs for adding $p = 3$ runs.

The A_s, D_s and E_s optimal designs for adding four runs can be found in Tables 3.9, 3.10 and 3.11, respectively, where the following 11 ω vectors are needed.

ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9	ω_{10}	ω_{11}
1	1	1	0	1	1	1	0	0	0	1
1	1	0	1	1	0	0	1	1	0	1
1	0	1	1	0	1	0	1	0	1	1
0	1	1	1	0	0	1	0	1	1	1

m	4	5	6	7
U	$\omega_1\omega_2\omega_3\omega_4$	$\omega_1\omega_2\omega_3\omega_4^2$	$\omega_1\omega_2\omega_3\omega_8\omega_9\omega_{10}$	$\omega_1\omega_2\omega_3\omega_8\omega_9\omega_{10}^2$
m	8	9	10	11
U	$\omega_1^2\omega_8^2\omega_9^2\omega_{10}^2$	$\omega_1\omega_2\omega_3\omega_8^2\omega_9^2\omega_{10}^2$	$\omega_1\omega_5\omega_6\omega_7^2\omega_8\omega_9^2\omega_{10}^2$	$\omega_5\omega_6^2\omega_7^2\omega_8^2\omega_9^2\omega_{10}^2$
m	12	13	14	15
U	$\omega_5^2\omega_6^2\omega_7^2\omega_8^2\omega_9^2\omega_{10}^2$	$\omega_5^3\omega_6^2\omega_7^2\omega_8^2\omega_9^2\omega_{10}^2$	$\omega_5^3\omega_6^3\omega_7^2\omega_8^2\omega_9^2\omega_{10}^2$	$\omega_5^3\omega_6^3\omega_7^3\omega_8^2\omega_9^2\omega_{10}^2$
m	16	17	18	19
U	$\omega_5^3\omega_6^3\omega_7^3\omega_8^3\omega_9^2\omega_{10}^2$	$\omega_5^3\omega_6^3\omega_7^3\omega_8^3\omega_9^3\omega_{10}^2$	$\omega_5^3\omega_6^3\omega_7^3\omega_8^3\omega_9^3\omega_{10}^3$	$\omega_5^4\omega_6^3\omega_7^3\omega_8^3\omega_9^3\omega_{10}^3$
m	20			
U	$\omega_5^4\omega_6^4\omega_7^3\omega_8^3\omega_9^3\omega_{10}^3$			

Table 3.9: A_s optimal compromise designs for adding $p = 4$ runs.

m	4	5	6	7
U	$\omega_1\omega_2\omega_3\omega_4$	$\omega_1^2\omega_2\omega_3\omega_4$	$\omega_1^2\omega_2^2\omega_3\omega_4$	$\omega_1^2\omega_2^2\omega_3^2\omega_4$
m	8	9	10	11
U	$\omega_1^2\omega_2^2\omega_3^2\omega_4^2$	$\omega_1^3\omega_2^2\omega_3^2\omega_4^2$	$\omega_1^2\omega_2^2\omega_3^2\omega_4\omega_8\omega_9\omega_{10}$	$\omega_1^3\omega_2^2\omega_3^2\omega_4^2\omega_8\omega_9$
m	12	13	14	15
U	$\omega_1^3\omega_2^3\omega_3^2\omega_4^2\omega_8\omega_9$	$\omega_1^3\omega_2^3\omega_3^3\omega_4^2\omega_8\omega_9$	$\omega_1^3\omega_2^3\omega_3^3\omega_4^3\omega_8\omega_9$	$\omega_1^4\omega_2^3\omega_3^3\omega_4^3\omega_8\omega_9$
m	16	17	18	19
U	$\omega_1^4\omega_2^4\omega_3^3\omega_4^3\omega_8\omega_9$	$\omega_1^4\omega_2^4\omega_3^4\omega_4^3\omega_8\omega_9$	$\omega_1^4\omega_2^4\omega_3^4\omega_4^4\omega_8\omega_9$	$\omega_1^5\omega_2^4\omega_3^4\omega_4^4\omega_8\omega_9$
m	20			
U	$\omega_1^5\omega_2^5\omega_3^4\omega_4^4\omega_8\omega_9$			

Table 3.10: D_s optimal compromise designs for adding $p = 4$ runs.

m	4	5	6	7	8	9	10	11	12
U	ω_{11}^{4}	ω_{11}^{5}	ω_{11}^{6}	ω_{11}^{7}	ω_{11}^{8}	ω_{11}^{9}	ω_{11}^{10}	ω_{11}^{11}	ω_{11}^{12}
m	13	14	15	16	17	18	19	20	
U	ω_{11}^{13}	ω_{11}^{14}	ω_{11}^{15}	ω_{11}^{16}	ω_{11}^{17}	ω_{11}^{18}	ω_{11}^{19}	ω_{11}^{20}	

Table 3.11: E_s optimal compromise designs for adding $p = 4$ runs.

While there is no apparent pattern in the A_s and D_s optimal designs. E_s optimal designs can all be obtained by adding copies of all-ones run to the basic OFAT design. This may well hold in general and deserves further investigation.

Chapter 4

Second Class of Compromise Designs Under Baseline Parameterization

Orthogonal arrays and one-factor-at-a-time (OFAT) designs perform well under the criteria of minimizing variance and bias, respectively, but not under both criteria. Compromise designs provide middle ground between orthogonal arrays and OFAT designs. Chapter 3 has considered a class of compromise designs by adding runs to the basic OFAT designs using efficiency criteria. This chapter investigates another class of compromise designs obtained from orthogonal arrays by changing some ones to zeros. To obtain the best designs, we use a method of complete search for smaller run sizes, and an incomplete search algorithm when the complete search is not feasible.

4.1 Introduction

Two-level factorial designs are a useful class of screening designs in both theory and practice. The orthogonal parameterization is the most often used method of analyzing two-level factorial designs. The baseline parameterization is less common but quite natural in situations where there is a null state or baseline level for

each factor (Banerjee and Mukerjee, 2008 [2]). Mukerjee and Tang (2012) [27] studied two-level factorial designs under the baseline parameterization. They showed that orthogonal arrays of strength two are universally optimal for estimating main effects and proposed a minimum aberration criterion when interactions exist in addition to the main effects. The minimum aberration criterion can be used to minimize the bias in the main effect estimation among all orthogonal arrays. OFAT designs allow unbiased estimation of main effects even in the presence of some or all interactions.

Even though minimum aberration (MA) designs are universally optimal, they are not optimal under the bias criterion. On the other hand, basic OFAT designs are better at minimizing the bias, but not the variance. Therefore it would be very useful to have some designs available that are not optimal under either criterion, but perform well under both. Karunanayaka and Tang (2017) [15] proposed a class of compromise designs obtained by adding runs to the basic OFAT designs using efficiency criteria.

In this chapter, we propose another class of compromise designs obtained from MA designs by changing some ones to zeros. These designs should perform well in terms of minimizing the variance if the number of such changes is small. The best designs from this second class of compromise designs are then selected using both the efficiency criterion, A_s and the bias criterion, K_2. We use a complete search procedure to obtain such designs starting from changing a single one to zero in each column in MA designs. However, the complete search procedure is feasible only for small designs. Therefore, an efficient incomplete search algorithm is developed to handle larger designs. The performance of our incomplete search algorithm is examined by comparing its results with those from the complete search procedure.

4.2 A New Class of Compromise Designs

The first class of compromise designs was studied by Karunanayaka and Tang (2017) [15] - see Chapter 3. Such designs are obtained by adding runs to the basic OFAT designs. When adding runs to the basic OFAT design, the resulting design would be biased, but it will perform well in terms of minimizing variance. This trade-off between minimizing the bias and variance is the foundation to building a second class of compromise designs. Starting from the MA designs, which are the best in terms of minimizing variance, but not in minimizing the bias, we are looking for a class of compromise designs which may not be optimal under the variance but may improve the bias. This second class of compromise designs is obtained by changing some ones to zeros in MA designs. First, consider the n run, m factor orthogonal array of strength two. Then, there are $n/2$ ones and zeros for each factor. Let p be the number of ones to be changed to zeros in all m columns, where $1 < p < n/2$. For a given m and p, we can obtain a class of compromise designs and, one such design, Z_{C_2}, can be obtained by taking $p = 1$. Let's consider the three designs which are A_s optimal, bias optimal and compromise design of second class for $n = 8$ and $m = 6$.

$$Z_{MA} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad Z_{OFAT} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$Z_{C_2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

The following table summarizes the values of A_s and K_2 for the three designs. Compared with Z_{MA}, design Z_{C_2} has a significant decrease of bias while it loses only a bit of the efficiency. This makes a lot of sense because this design is not far from the MA design.

Z_{MA}	$A_s = 3.0$	$K_2 = 10.5$
Z_{C_2}	$A_s = 3.3$	$K_2 = 6.24$
Z_{OFAT}	$A_s = 9.0$	$K_2 = 0.0$

Table 4.1: The A_s and K_2 values of Z_{OFAT}, Z_{C_2} and Z_{MA} for $n = 8$ and $m = 6$

We can obtain the second class of compromise designs for different values of p, but the question is how to find the optimal design out of all the designs in the class.

4.3 Finding Optimal Designs from the Second Class of Compromise Designs

4.3.1 Method of Complete Search

We can obtain the second class of compromise designs by changing some ones to zeros in MA designs. The changes can be done in many ways, which will produce many classes of compromise designs. One such class of compromise designs of interest is given by changing p ones to zeros in all m columns. We start the search

procedure for the case $p = 1$. Since the resulting designs are not very far from the orthogonal arrays, they should perform well under the efficiency criteria. However, the bias can be improved significantly due to those changes, because the number of zeros increases in interaction matrices. We continue the search procedure for the cases $1 < p < n/2$ and the following example shows the results for $n = 8$ and $m = 6$.

	$n = 8, m = 6$	A_s	K_2
	MA	3	10.5
C_2: $p = 1$	A_s-criterion	3.3	6.24
	K_2-criterion	3.3	6.24
C_2: $p = 2$	A_s-criterion	5.25	4.5
	K_2-criterion	10	3
C_2: $p = 3$	A_s-criterion	9	0
	K_2-criterion	9	0
	$OFAT$	9	0
	C_1	5.25	3.16

Table 4.2: The A_s and K_2 values of the MA , C_2, $OFAT$ and C_1 designs for $n = 8$ and $m = 6$

This table compares the results of MA, $OFAT$, first class of compromise (C_1) and second class of compromise (C_2) designs using the both bias and efficiency criteria. We first obtain the best design using each criterion and then use that design to calculate the value of the other criterion. For an example, consider the class of compromise designs for the case of $p = 2$. First, we search for a design that minimizes the variance. We use the A_s criterion and find that the best design has $A_s = 5.25$ and its bias $K_2 = 4.5$. Then we obtain the bias-best design using K_2 criterion and find that the best design has $K_2 = 3$ and $A_s = 10$.

We are able to produce compromise designs using both bias and efficiency criteria for the cases of $n = 8$ and all m, and for $n = 12$ up to $m = 5$ factors. The results are presented in Appendix B. We can still use the complete search procedure to obtain a few more optimal designs, but it becomes very time consuming since it involves an enormous amount of evaluations and at some point this search procedure has to be terminated, because the number of evaluations is given by $\binom{n/2}{p}^m$, where $1 \leq p < n/2$. In fact, the complete search procedure is not feasible even for the smaller run sizes, for an example, $n = 12$ and $m = 8$ there are 2.56×10^{10} possibilities for the case of $p = 3$. This leads us to use an algorithmic search procedure. In the next section, we will propose an efficient incomplete search algorithm.

4.3.2 A systematic method of construction

Mukerjee and Tang (2012) [27] proved that the MA designs are the best in minimizing variance under the baseline parameterization. Compromise designs corresponding to $p = 1$ also have some good properties in minimizing the variance, because they are not very far away from the MA designs. Motivated by an apparent pattern in the optimal designs found by complete search, we introduce a systematic method of construction.

Step 1. Let $1 + Z_{MA}$ be the matrix obtained by adding 1 (mod 2), to all the entries of design matrix of Z_{MA},

Step 2. Replace the row consisting of all ones by zeros.

Even though a theoretical result has not yet been obtained, this method provides all optimal designs given by complete search. In addition, we can show that

the design from this construction has

$$A_s = \frac{4m}{n}\left[1 + \frac{4}{(n^2 - 4m)}\right]. \tag{4.1}$$

Example 1: We give a systematic construction of the variance-best design for $n = 8$ and $m = 6$. From an $OA(8, 6, 2, 2)$, say Z_{MA}, we can obtain $1 + Z_{MA}$, which is the matrix obtained by adding 1 (mod 2) all the entries of Z_{MA}. Then we replace all the ones by zeros in the resulting matrix and which gives the best design, $Z^*_{C_2}$, under A_s criterion. Using the equation (4.1), we have $A_s = 3.3$.

$$1 + Z_{MA} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}, \qquad Z^*_{C_2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

4.3.3 Algorithmic Search

The idea of our search algorithm is a version of the local search algorithms from Aarts and Lenstra (2003) [1]. Suppose that we are searching for the bias-best design for the case of $p = 1$ in the MA design of $n = 8$ and $m = 6$. We can select a random initial design by changing a single one to zero in all six columns or pick the design that is obtained from the systematic method. Let's denote this as D_0. First, let's obtain all the fifteen (i.e. 5×3) neighbours around D_0 by interchanging a one and a zero in one and only one of the six columns and, compute the K_2 for the resulting designs. This process will continue for all the other columns. Then we compare the results and obtain the design that has the smallest K_2, say D_1. If D_1 is better than D_0, we then use D_1 as our new D_0, denoted by D_0^*. For this

new D_0^*, we then obtain all its new neighbours and compare them with D_0^*. We will continue this process until there is no improvement. Moreover, if we can start the process with different random designs and continue the same process, we may be able to find a better design. A step by step description of this algorithm as follows:

1. Choose a random design and compute K_2.

2. Generate new arrays by interchanging the zeros and ones in each column of the initial design and compute the K_2 for these arrays. If the initial design has a smaller K_2 value than any of these designs, we choose a different initial design. Otherwise, replace the initial design by the current best deign.

3. Continue Step 2 until there is no improvement.

4. We can continue Steps 1-3 by starting several initial designs and see if we can find a better design.

Remark 1. Though not considered in this thesis, our algorithm is obviously applicable to run sizes for which orthogonal arrays do not exist. For example, it can be applied to designs of $n = 10$ runs with each column consisting of four ones and six zeros.

4.4 Performance of Our Algorithm

The complete search procedure produces results for all the cases of m in run sizes $n = 8$ and for $n = 12$ up to $m \leq 5$ factors. Using our algorithm, we are able to obtain the optimal designs for every m in run sizes up to $n = 20$. The only way that we can check the performance of our search algorithm is to compare its results with those of the complete search. It can be observed that, in most cases,

the search algorithm performs as well as the complete search. The following tables make comparison of the results from the complete search algorithm and incomplete search algorithm for only the cases where two algorithms produce different results for both A_s and K_2 criteria. All the other results are presented in Appendix B.

n	m	p	Complete:K_2	Incomplete:K_2
8	3	1	0.778	0.582
8	6	1	6.24	6.694
8	7	1	9.333	12.611
12	2	1	0.202	0
12	3	1	0.812	0.736
12	4	1	1.709	1.506
12	5	1	3.074	2.845

Table 4.3: Compare K_2 values for both complete and incomplete search algorithms

n	m	p	Complete:A_s	Incomplete:A_s
8	6	1	3.3	3.563
8	7	1	3.889	6.111

Table 4.4: Compare A_s values for both complete and incomplete search algorithms

By looking at these numbers, we conclude that the incomplete search algorithm performance is very satisfactory. Surprisingly, it produces better results than the complete search in some cases. The reason behind this is that in complete search we replace ones by zeros and in the search algorithm we swap ones and zeros. So there are other designs that are not obtainable from the complete search. It is also worth to mentioning the impact of the initial design. The search procedure starts from the random design and our best design under the bias criterion, K_2 will depends on that. Therefore, we cannot guarantee that the best design can be obtained from the incomplete search algorithm all the time.

Chapter 5

Conclusions and Future Research

In Chapter 2, we studied the construction of orthogonal and nearly orthogonal designs. In addition to what has been done, an interesting theoretical problem is the construction of the best nearly orthogonal design when orthogonal designs do not exist, which happens when $s = 4k+2$ and λ is odd as shown in Theorem 1. In this case, it is possible to obtain a lower bound on a measure of nonorthogonality, which can then be used to assess the quality of a given design. Lin (2008) [18] considered this problem for Latin hypercubes. It would be interesting to examine if such a lower bound could be obtained in the general case.

Besides computer experiments, the orthogonal designs given in this thesis can also be used as fractional factorial designs in screening experiments. Miller and Sitter (2001) [24] examined how to identify interactions using a design of 24 runs for 12 factors obtained by folding over a 12 run Plackett-Burman design. Folding over the $\mathrm{OD}(12, 3^9)$ gives an $\mathrm{OD}(24, 3^9)$, which also has the property that main effects are all orthogonal to interactions. This design may not be as efficient in terms of estimating linear main effects as the two-level design of 24 runs considered by Miller and Sitter (2001) [24], but it has three levels, allowing detection of curvature and thus the possibility of response surface exploration (Cheng and Wu (2001) [7]). This would be an interesting problem to look at in the future.

In Chapter 3, we proposed and studied a new class of baseline designs, which are obtained by adding runs to the basic OFAT design. Such designs are quite competitive under both the efficiency criterion and the bias criterion, and thus offer an attractive class of alternatives to the efficiency-best MA designs and the bias-best OFAT designs. A theoretical result shows that the design obtained by adding a run of all ones is A_s, D_s and E_s optimal. In view of this result, it is possible to establish the type 1 optimality of this design using the results of Cheng (1980) [4] and Cheng (2014) [5]. Our computer search results of optimal designs for adding two or more runs exhibit some apparent patterns, which strongly hint that there may exist some theoretical optimality results for the general case of adding $p \geq 2$ runs. These are interesting further research topics, which we hope to address in the future.

In Chapter 4, we have investigated a second class of compromise designs starting from MA designs. The search for the best designs is done using both complete and incomplete search algorithms. Performance of our incomplete search algorithm is studied by making comparison with the available cases of complete search results. We have seen that our algorithm gives similar results to the complete search algorithm. However, this might not be the case in general. In the current incomplete search algorithm, we select a neighbourhood of size one. One possible way to improve the algorithm is to consider using larger neighbourhoods in our local search algorithm.

Bibliography

[1] E. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. Princeton University Press: New Jersey, 2003.

[2] T. Banerjee and R. Mukerjee. Optimal factorial designs for cDNA microarray experiments. *The Annals of Applied Statistics*, 2:366–385, 2008.

[3] D. Bingham, R. R. Sitter, and B. Tang. Orthogonal and nearly orthogonal designs for computer experiments. *Biometrika*, 966:51–65, 2009.

[4] C. S. Cheng. Optimality of some weighing and 2^n fractional factorial designs. *The Annals of Applied Statistics*, 8:436–446, 1980.

[5] C. S. Cheng. Optimal biased weighing designs and two-level main-effect plans. *Journal of Statistical Theory and Practice*, 8:83–89, 2014.

[6] C. S. Cheng. *Theory of Factorial Design: Single- and Multi-Stratum Experiments*. CRC Press, 2014.

[7] S. W. Cheng and C. F. J. Wu. Factor screening and response surface exploration. *Statistica Sinica*, 11:553–604, 2001.

[8] A. Dean, M. Morris, J. Stufken, and D. Bingham. *Handbook of Design and Analysis of Experiments*. CRC Press, 2015.

[9] K. Fang, R. Li, and A. Sudjianto. *Design and Modeling for Computer Experiments*. CRC Press, 2006.

[10] A. Fries and W. G. Hunter. Minimum aberration 2^{k-p} designs. *Technometrics*, 22:601–608, 1980.

[11] S. D. Georgiou and I. Efthimiou. Some classes of orthogonal latin hypercube designs. *Statistica Sinica*, 24:101–120, 2014.

[12] S. D. Georgiou, S. Stylianou, K. Drosou, and C. Koukouvinos. Construction of orthogonal and nearly orthogonal designs for computer experiments. *Biometrika*, 101:741–747, 2014.

[13] A. V. Geramita and J. Seberry. *Orthogonal Designs*. Marcel Dekker, 1979.

[14] R. L. Iman and W. J. Conover. A distribution-free approach to inducing rank correlation among input variables. *Communication in Statistics–Simulation and Computation*, 11:311–334, 1982.

[15] R. C. Karunanayaka and B. Tang. Compromise designs under baseline parametrization. *Journal of Statistical Planning and Inference*, 190:32–38, 2017.

[16] K. F. Kerr. Efficient 2^k factorial designs for blocks of size 2 with microarray applications. *Journal of Quality Technology*, 38:309–318, 2006.

[17] P. Li, A. Miller, and B. Tang. Algorithmic search for baseline minimum aberration designs. *Journal of Statistical Planning and Inference*, 149:172–189, 2014.

[18] C. D. Lin. New developments in designs for computer experiments and physical experiments. *Doctoral dissertation, Department of Statistics and Actuarial Science, Simon Fraser University*, 2008.

[19] C. D. Lin, D. Bingham, R. R. Sitter, and B. Tang. A new and flexible method for constructing designs for computer experiments. *The Annals of Statistics*, 38:1460–1477, 2010.

[20] C. D. Lin, R. Mukerjee, and B. Tang. Construction of orthogonal and nearly orthogonal latin hypercubes. *Biometrika*, 96:243–247, 2009.

[21] H. Liu and M.Q. Liu. Column-orthogonal strong orthogonal arrays and sliced strong orthogonal arrays. *Statistica Sinica*, 25:1713–1734, 2015.

[22] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 1979.

[23] R. Mee. *A comprehensive guide to factorial two-level experimentation*. Springer Science & Business Media, 2006.

[24] A. Miller and R. R. Sitter. Using the folded-over 12-run plackett-burman design to consider interactions. *Technometrics*, 43:44–55, 2001.

[25] A. Miller and B. Tang. Using regular fractions of two-level designs to find baseline designs. *Statistica Sinica*, 26:745–759, 2016.

[26] M. D. Morris and T. J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43:381–402, 1995.

[27] R. Mukerjee and B. Tang. Optimal fractions of two-level factorials under a baseline parameterization. *Biometrika*, 99:71–84, 2012.

[28] R. Mukerjee and B. Tang. Optimal two-level regular designs under baseline parametrization via cosets and minimum moment aberration. *Statistica Sinica*, 26:1001–1019, 2016.

[29] A. Owen. Controlling correlations in latin hypercube samples. *Journal of the American Statistical Association*, 89:1517–1522, 1994.

[30] F. Pang, M. Q. Liu, and D. K. Lin. A construction method for orthogonal latin hypercube designs with prime power levels. *Statistica Sinica*, 96:1721–1728, 2009.

[31] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Designs and analysis of computer experiments. *Statistical Science.*, 4(4):433–435, 1989.

[32] T. J. Sanater, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer Series in Statistics, 2003.

[33] J. W. Stallings and J. P. Morgan. General weighted optimality of designed experiments. *Biometrika*, 102:925–935, 2015.

[34] D. M. Steinberg and D. K. Lin. A construction method for orthogonal latin hypercube designs. *Biometrika*, 93:279–288, 2006.

[35] S. Stylianou, S. K. Drosou, S. D. Georgiou, and C. Koukouvinos. Column-orthogonal and nearly column-orthogonal designs for models with second-order terms. *Journal of Statistical Planning and Inference*, 161:81—90, 2015.

[36] F. Sun, M. Q. Liu, and D. K. Lin. Construction of orthogonal latin hypercube designs. *Biometrika*, 96:971–974, 2009.

[37] F. S. Sun, F. Pang, and M. Q. Liu. Construction of column-orthogonal designs for computer experiments. *Science China Mathematics*, 54:2683–2692, 2011.

[38] F. S. Sun and B. Tang. A general rotation method for orthogonal latin hypercubes. *Biometrika*, 104:465–472, 2017.

[39] F. S. Sun and B. Tang. A method of constructing space-filling orthogonal designs. *Journal of the American Statistical Association*, 92:683–689, 2017.

[40] B. Tang. Orthogonal array-based latin hypercubes. *Journal of the American Statistical Association*, 88:1392–1397, 1993.

[41] B. Tang. Selecting latin hypercube designs using correlation criteria. *Statistics Sinica*, 8:965–977, 1998.

[42] C. F. J. Wu and M. S. Hamada. *Experiments: Planning, Analysis and Optimization*. Wiley, Hoboken, New Jersey, 2009.

[43] H. Xu. An algorithm for constructing orthogonal and nearly-orthogonal arrays with mixed levels and small runs. *Technometrics*, 44:356–368, 2002.

[44] K. Q. Ye. Orthogonal column latin hypercubes and their application in computer experiments. *Journal of the American Statistical Association*, 93:1430–1439, 1998.

Appendix A

Nearly Orthogonal Designs (NOD) up to $n = 18$

n																
6					12											
−1	−1	−1	−1	0	−1	−1	−1	−1	−1	−1	−1	0	0	−1	0
0	−1	1	1	−1	−1	−1	−1	0	1	1	1	0	0	1	−1
1	0	0	−1	−1	0	−1	1	1	−1	0	1	−1	−1	−1	0
−1	1	1	0	1	−1	0	1	1	1	−1	0	1	1	−1	0
0	1	−1	1	0	1	−1	1	−1	1	1	−1	0	0	0	0
1	0	0	0	1	0	1	−1	1	1	0	−1	−1	−1	0	1
					1	0	−1	1	−1	1	0	1	1	0	1
					−1	1	1	0	−1	1	−1	0	0	1	−1
					0	1	0	−1	0	0	1	−1	1	−1	−1
					1	0	0	0	0	−1	0	−1	1	1	1
					1	0	0	0	0	−1	0	1	−1	1	−1
					0	1	0	−1	0	0	1	1	−1	0	1

Table A.1: NOD$(n, 3^m)$ for $n = 6, 12$

n																
8							12									
3	3	1	−1	−3	−3	−3	−3	3	3	−1	−1	−1	−1	−3	−3	−3	−3
1	1	1	−1	−3	1	3	3	−1	1	3	−3	−3	3	−3	−1	−1	1
−1	−1	3	−3	3	1	−3	1	−1	3	−3	−3	3	−1	−1	1	3	−1
3	−3	−3	1	1	3	−1	−1	−3	−3	1	1	3	1	−3	1	1	−3
−3	−1	−3	−3	−1	−1	1	−1	1	−1	1	3	−1	−3	−1	−1	3	3
−1	−3	3	3	−1	−1	1	3	−3	−1	−3	1	−3	−3	1	−1	−1	−3
−3	3	−1	3	1	3	−1	1	1	1	1	3	1	3	3	−3	3	−1
1	1	−1	1	3	−3	3	−3	1	−3	−3	−1	−3	3	1	3	1	1
							−1	−1	−1	−1	−1	3	1	1	−3	−3	3
							−3	−3	3	3	1	−1	−1	3	3	−1	1
							1	3	−3	3	−3	1	−3	3	1	1	−1
							3	3	1	−1	3	1	1	−1	3	−3	3

Table A.2: NOD$(n, 4^m)$ for $n = 8, 12$

n																			
9								12											
1	1	0	−1	1	1	0	1	−5	−5	−5	−1	1	−3	3	−1	3	−1	5	
0	0	0	0	−1	−1	−1	1	5	−1	1	−3	−3	−5	−3	5	−3	−3	3	
1	−1	1	0	0	−1	1	0	1	3	−3	−5	5	5	1	3	−1	−5	−1	
−1	0	1	−1	−1	1	1	0	−3	3	−1	5	−5	3	1	−1	−5	−3	3	
1	1	0	1	−1	0	0	−1	−1	−5	5	1	−1	3	5	5	−1	3	−3	
−1	0	1	1	1	0	−1	−1	5	1	1	3	−1	−3	3	−3	5	−5	−5	
0	−1	−1	−1	0	0	−1	−1	1	−3	−5	5	3	−1	−5	1	−3	3	−5	
0	−1	−1	1	0	1	0	1	−5	5	5	1	5	−5	−1	1	1	1	1	
−1	1	−1	0	1	−1	1	0	−1	5	−3	−1	−5	1	−1	3	5	5	−1	
								−3	−3	3	−5	−3	1	−5	−5	1	−1	−3	
								3	−1	3	3	3	5	−3	−3	3	1	5	
								3	1	−1	−3	1	−1	5	−5	−5	5	1	

Table A.3: NOD$(9, 3^m)$ and NOD$(12, 6^m)$

n												
14												
0	1	−1	−3	3	2	−2	−2	3	1	−1	0	−3
2	3	1	0	−1	0	−1	0	2	−3	3	3	2
−1	2	−1	−3	0	−2	2	3	−2	3	0	3	0
−2	−1	2	2	−1	−1	1	3	3	−2	−2	1	−3
1	−3	0	−2	−2	−3	3	−3	1	0	3	−1	−1
−3	2	−3	3	−3	0	−1	−2	1	2	−1	0	1
1	0	2	0	1	−2	−3	2	2	3	0	−3	3
3	1	−2	1	0	−3	−2	0	−3	−2	−2	−1	−2
−1	−2	1	1	3	−1	1	−3	−1	−1	−3	2	3
3	−3	−2	3	1	3	0	1	0	2	2	2	0
−2	−2	3	−1	−2	2	−3	−1	−3	0	1	1	−1
2	0	0	−2	−3	3	2	1	0	−1	−3	−2	2
0	3	3	2	2	1	3	−1	−2	1	1	−3	−2
−3	−1	−3	−1	2	1	0	2	−1	−3	2	−2	1

Table A.4: NOD$(14, 7^m)$

n														
15														
1	−1	−1	−1	−1	−1	−1	−1	0	1	0	1	1	0	
1	−1	−1	0	0	0	1	1	−1	0	−1	−1	−1	1	
1	−1	0	0	1	1	0	0	1	−1	0	1	0	−1	
−1	−1	1	1	−1	−1	−1	0	0	−1	0	0	−1	0	
0	−1	1	1	1	1	0	−1	−1	1	0	0	1	1	
0	0	1	0	−1	0	1	1	−1	1	0	1	0	−1	
0	1	−1	1	−1	1	0	0	0	−1	1	1	0	1	
−1	0	−1	1	0	0	0	1	1	1	1	−1	1	−1	
−1	0	0	−1	−1	1	1	−1	1	0	−1	−1	0	0	
1	1	1	0	0	−1	1	−1	0	−1	1	−1	1	0	
0	1	−1	1	1	−1	0	−1	0	0	−1	0	−1	−1	
0	1	0	−1	0	1	−1	0	−1	0	1	−1	−1	−1	
−1	0	0	−1	1	−1	1	0	1	0	1	1	−1	1	
−1	0	0	−1	1	0	−1	1	−1	−1	−1	0	1	0	
1	1	1	0	0	0	−1	1	1	1	−1	0	0	1	

Table A.5: NOD$(15, 3^m)$

n													
15													
0	−1	−2	0	−2	−1	1	2	2	−1	−2	−1	1	2
1	−1	1	−1	2	−2	−1	−2	2	1	0	2	2	1
−1	1	1	2	−1	−2	−1	0	−1	2	2	−1	−1	2
−1	1	0	2	−1	2	−2	−2	1	−1	−2	1	−1	0
1	2	0	2	0	−2	1	1	0	−2	1	2	0	−2
−2	−2	1	1	1	2	1	2	2	1	1	1	0	−1
1	0	−1	−1	−2	1	−2	0	0	1	2	−1	2	−2
−1	1	−1	−2	2	−1	−1	1	1	0	−1	−2	−2	−2
2	−2	−2	1	0	0	2	−2	−1	2	0	0	−2	−1
2	2	2	0	1	2	2	0	1	0	0	−2	1	1
−2	−1	−1	1	2	0	0	−1	−2	−2	1	−2	2	0
0	0	0	−2	−1	1	0	−1	0	−2	2	1	−2	2
0	2	−2	−1	1	1	0	1	−2	2	−1	2	1	1
−2	0	2	−2	−2	−1	2	−1	−1	0	−2	0	0	−1
2	−2	2	0	0	0	−2	2	−2	−1	−1	0	−1	0

Table A.6: NOD$(15, 5^m)$

n															
16															
−3	−3	1	3	−3	−1	3	−3	1	−1	−3	1	3	1	3
3	1	1	−3	1	−3	1	−3	3	−1	−1	−1	−3	−3	3
−1	−1	−1	3	3	−3	3	3	−1	−1	1	1	−1	−3	−3
−1	−3	−3	−3	1	−3	−1	−3	−3	1	3	−1	1	3	−1
1	−1	3	1	3	−1	−1	3	−1	−3	−1	−3	−1	3	3
−1	3	3	−1	−1	1	1	−1	1	−3	3	−3	3	−1	−3
1	−3	−3	1	−3	3	−1	1	3	−3	1	−1	−3	1	−1
−1	1	1	3	1	1	−3	−3	−1	3	−3	−3	−3	−1	−3
3	−1	−1	1	3	1	−3	−1	3	1	−1	3	3	1	−3
3	3	1	3	−3	−1	−1	−1	−3	−1	3	3	−1	1	1
−3	3	−1	−1	3	3	3	−1	1	1	1	3	−3	3	1
3	−3	3	−3	−1	3	3	1	−3	3	−1	1	−1	−1	−1
1	3	−3	−1	−3	−3	1	3	1	3	−3	−1	1	3	−1
−3	1	−1	−3	−1	1	−3	1	−3	−3	−3	3	1	−3	1
−3	−1	3	−1	−1	−1	−3	3	3	3	3	1	1	−1	1
1	1	−3	1	1	3	1	1	−1	1	1	−3	3	−3	3

Table A.7: NOD$(16, 4^m)$

n														
16														
−7	−7	−5	3	3	3	−7	1	1	3	−5	−7	5	1	−3
−1	−5	5	5	5	−3	1	3	5	−3	5	5	7	−1	5
3	1	−1	−5	7	−7	−7	5	−7	−5	1	−1	−5	5	1
−5	−5	7	−7	−5	3	5	5	−1	−7	−3	1	−3	−1	−5
−3	5	−5	−7	−5	−1	−3	7	3	7	−1	5	3	−5	5
−1	−1	3	1	7	5	1	−5	−3	1	−5	3	−7	−7	7
−7	3	−3	1	1	−3	7	−1	7	1	5	−5	−7	7	3
1	3	−1	−3	−1	1	−5	−7	5	−7	7	−5	1	−7	−3
3	5	5	3	−7	1	−5	−3	3	−3	−7	−1	3	7	7
5	−3	1	−5	−1	−7	7	−5	−3	5	−3	−7	5	−3	1
7	−1	−7	−1	3	−1	3	−1	7	−1	−7	7	−1	3	−7
−5	−3	−3	5	−7	−5	−1	−7	−7	−1	3	7	−1	1	−1
1	7	7	7	1	−5	−1	3	−1	5	−1	−3	−3	−5	−7
7	−7	3	−1	−3	7	−3	1	1	7	7	1	−5	3	−1
−3	7	1	−3	5	7	3	−3	−5	3	3	3	7	5	−5
5	1	−7	7	−3	5	5	7	−5	−5	1	−3	1	−3	3

Table A.8: NOD$(16, 8^m)$

n																
18																
0	−1	−1	−1	−1	−1	−1	−1	1	1	1	0	−1	0	1	0	−1
−1	−1	−1	−1	−1	0	0	1	−1	−1	−1	0	1	0	0	1	0
0	−1	−1	0	1	0	0	−1	−1	−1	0	−1	−1	1	−1	−1	0
1	−1	−1	0	1	1	1	1	1	1	0	1	0	−1	0	0	0
0	−1	1	1	−1	0	1	0	1	0	−1	−1	−1	0	−1	1	0
0	−1	1	1	0	1	−1	0	−1	1	1	−1	1	1	1	0	1
−1	0	0	1	0	1	0	0	1	−1	−1	1	0	1	1	−1	−1
−1	0	0	1	1	−1	1	0	−1	0	1	0	0	−1	1	1	−1
−1	0	1	0	1	−1	−1	−1	1	0	0	1	1	−1	−1	0	1
−1	1	0	−1	1	1	−1	1	0	1	−1	−1	−1	0	0	0	0
0	1	0	−1	0	1	1	−1	1	−1	1	−1	1	0	0	1	0
1	0	1	−1	0	0	1	−1	−1	1	−1	0	1	0	0	−1	−1
1	0	1	−1	1	−1	0	1	0	−1	0	1	−1	1	1	1	1
1	0	1	0	−1	0	−1	1	0	−1	1	0	0	−1	−1	−1	−1
0	1	0	0	−1	1	0	−1	−1	0	0	1	−1	−1	0	0	1
1	1	−1	1	0	−1	0	0	0	0	−1	−1	0	−1	1	−1	1
−1	1	0	0	−1	−1	1	0	1	1	0	0	1	−1	−1	−1	1
1	1	−1	1	0	0	−1	0	0	0	0	1	1	1	−1	1	−1

Table A.9: NOD$(18, 3^m)$

n																
18																
−1	−1	−1	−3	−3	0	3	0	3	4	−2	−2	−3	−4	3	−3	3
0	−2	0	−1	−3	4	−3	1	1	−4	3	−1	−1	4	4	2	4
−2	−2	2	−4	−1	−3	−1	3	−3	−3	3	−3	−2	−3	−4	1	−1
−1	2	−3	−4	4	3	2	4	2	0	1	0	4	1	1	−2	−4
0	−4	3	−2	4	2	2	−1	2	1	−4	2	−1	2	−3	4	1
1	−3	2	4	0	−1	−4	3	4	0	0	3	−3	0	0	−4	−4
3	3	4	−3	3	−1	−4	−4	0	2	1	−3	0	1	1	−3	1
−4	4	1	2	−2	4	1	−1	−3	2	−1	−1	−4	3	−1	0	−3
3	−4	0	2	0	−2	3	0	−4	4	4	−2	1	2	3	2	−2
−2	1	4	3	2	0	0	4	−4	−1	−3	1	3	−2	2	−2	4
4	0	−2	0	1	2	4	−3	−2	−4	2	4	−2	−1	−2	−4	2
4	1	−2	1	−4	−2	1	2	1	−1	−4	−4	2	4	−3	−1	0
−3	−3	−4	4	3	1	−2	−4	0	−2	−1	−4	1	−3	−1	0	0
2	3	−3	0	2	−4	−1	1	−1	−2	−3	2	−4	−1	4	4	−1
−4	−1	−4	−2	−1	−4	−2	−2	−1	3	0	4	2	3	−2	−1	2
2	0	1	−1	−4	3	−3	−2	−2	1	−2	3	4	−4	0	3	−3
−3	2	3	1	−2	−3	4	−3	4	−3	2	1	3	0	2	1	−2
1	4	−1	3	1	1	0	2	3	3	4	0	0	−2	−4	3	3

Table A.10: NOD$(18, 9^m)$

Appendix B

Search Algorithm Results from Chapter 4

B.1 Results for $n = 8$

	$n = 8, m = 2$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	1.071	0.255
	Algorithm:A_s-criterion	1.071	0.255
	Complete search:K_2-criterion	1.667	0
	Algorithm:K_2-criterion	1.667	0
	Systematic method	1.071	0.255
$p = 2$	Complete search:A_s-criterion	1.5	0.281
	Algorithm:A_s-criterion	1.5	0.281
	Complete search:K_2-criterion	1.5	0
	Algorithm:K_2-criterion	1.5	0
$p = 3$	Complete search:A_s-criterion	2.333	0
	Algorithm:A_s-criterion	2.333	0
	Complete search:K_2-criterion	2.333	0
	Algorithm:K_2-criterion	2.333	0
	MA	1	0.5

	$n = 8, m = 3$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	1.615	0.789
	Algorithm:A_s-criterion	1.615	0.789
	Complete search:K_2-criterion	2.833	0.778
	Algorithm:K_2-criterion	2.242	0.582
	Systematic method	1.615	1.331
$p = 2$	Complete search:A_s-criterion	2.4	2.25
	Algorithm:A_s-criterion	2.4	0.33
	Complete search:K_2-criterion	3	0
	Algorithm:K_2-criterion	3	0
$p = 3$	Complete search:A_s-criterion	3.6	0
	Algorithm:A_s-criterion	3.6	0
	Complete search:K_2-criterion	3.6	0
	Algorithm:K_2-criterion	3.6	0
	MA	1.5	2.25

	$n = 8, m = 4$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	2.167	1.667
	Algorithm:A_s-criterion	2.167	2.146
	Complete search:K_2-criterion	3.667	1.444
	Algorithm:K_2-criterion	3.667	1.444
	Systematic method	2.167	2.146
$p = 2$	Complete search:A_s-criterion	3.333	1.083
	Algorithm:A_s-criterion	3.333	1.083
	Complete search:K_2-criterion	4.5	0.625
	Algorithm:K_2-criterion	4.5	0.625
$p = 3$	Complete search:A_s-criterion	5	0
	Algorithm:A_s-criterion	5	0
	Complete search:K_2-criterion	5	0
	Algorithm:K_2-criterion	5	0
	MA	2	3.75

	$n = 8, m = 5$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	2.727	3.810
	Algorithm:A_s-criterion	2.727	3.810
	Complete search:K_2-criterion	3.444	3.333
	Algorithm:K_2-criterion	3.444	3.333
	Systematic method	2.727	3.810
$p = 2$	Complete search:A_s-criterion	4.286	2.235
	Algorithm:A_s-criterion	4.286	1.653
	Complete search:K_2-criterion	6	1.375
	Algorithm:K_2-criterion	6	1.375
$p = 3$	Complete search:A_s-criterion	6.667	0
	Algorithm:A_s-criterion	6.667	0
	Complete search:K_2-criterion	6.667	0
	Algorithm:K_2-criterion	6.667	0
	MA	2.5	6.5

	$n = 8, m = 6$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	3.3	6.24
	Algorithm:A_s-criterion	3.563	6.313
	Complete search:K_2-criterion	3.3	6.24
	Algorithm:K_2-criterion	4.611	6.694
	Systematic method	3.3	6.24
$p = 2$	Complete search:A_s-criterion	5.25	4.5
	Algorithm: A_s-criterion	5.25	3.656
	Complete search:K_2-criterion	10	3
	Algorithm: K_2-criterion	7.5	3
$p = 3$	Complete search:A_s-criterion	9	0
	Algorithm: A_s-criterion	9	0
	Complete search:K_2-criterion	9	0
	Algorithm: K_2-criterion	9	0
	MA	3	10.5
	$OFAT$	9	0
	C_1	5.25	3.16

	$n = 8, m = 7$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	3.889	9.333
	Algorithm:A_s-criterion	6.111	12.611
	Complete search:K_2-criterion	3.889	9.333
	Algorithm:K_2-criterion	6.111	12.611
	Systematic method	3.889	9.333
$p = 2$	Complete search:A_s-criterion	9	6
	Algorithm:A_s-criterion	9	6
	Complete search:K_2-criterion	9	6
	Algorithm:K_2-criterion	9	6
$p = 3$	Complete search:A_s-criterion	14	0
	Algorithm:A_s-criterion	14	0
	Complete search:K_2-criterion	14	0
	Algorithm:K_2-criterion	14	0
	MA	3.5	15.75

B.2 Results for $n = 12$

	$n = 12, m = 2$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	0.686	0.339
	Algorithm:A_s-criterion	0.686	0.339
	Complete search:K_2-criterion	0.795	0.202
	Algorithm:K_2-criterion	1.4	0
	Systematic method	0.686	0.339
$p = 2$	Complete search:A_s-criterion	0.762	0.163
	Algorithm:A_s-criterion	0.762	0.163
	Complete search:K_2-criterion	1	0
	Algorithm:K_2-criterion	1	0
$p = 3$	Complete search:A_s-criterion	0.9	0.18
	Algorithm:A_s-criterion	0.9	0.18
	Complete search:K_2-criterion	1	0
	Algorithm:K_2-criterion	1	0
$p = 4$	Complete search:A_s-criterion	1.25	0
	Algorithm:A_s-criterion	1.25	0
	Complete search:K_2-criterion	1.25	0
	Algorithm:K_2-criterion	1.25	0
$p = 5$	Complete search:A_s-criterion	2.2	0
	Algorithm:A_s-criterion	2.2	0
	Complete search:K_2-criterion	2.2	0
	Algorithm:K_2-criterion	2.2	0
	MA	0.667	0.5

	$n = 12, m = 3$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	1.030	1.049
	Algorithm:A_s-criterion	1.030	1.049
	Complete search:K_2-criterion	1.264	0.812
	Algorithm:K_2-criterion	1.752	0.736
	Systematic method	1.030	1.049
$p = 2$	Complete search:A_s-criterion	1.167	0.472
	Algorithm:A_s-criterion	1.167	0.472
	Complete search:K_2-criterion	1.45	0.403
	Algorithm:K_2-criterion	1.45	0.403
$p = 3$	Complete search:A_s-criterion	1.364	0.707
	Algorithm:A_s-criterion	1.364	0.669
	Complete search:K_2-criterion	2	0
	Algorithm:K_2-criterion	2	0
$p = 4$	Complete search:A_s-criterion	2	0
	Algorithm:A_s-criterion	2	0
	Complete search:K_2-criterion	2	0
	Algorithm:K_2-criterion	2	0
$p = 5$	Complete search:A_s-criterion	3.333	0
	Algorithm:A_s-criterion	3.333	0
	Complete search:K_2-criterion	3.333	0
	Algorithm:K_2-criterion	3.333	0
	MA	1	1.583

	$n = 12, m = 4$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	1.375	2.293
	Algorithm:A_s-criterion	1.375	2.293
	Complete search:K_2-criterion	1.654	1.709
	Algorithm:K_2-criterion	1.914	1.506
	Systematic method	1.375	2.293
$p = 2$	Complete search:A_s-criterion	1.6	0.907
	Algorithm:A_s-criterion	1.6	0.907
	Complete search:K_2-criterion	1.6	0.907
	Algorithm:K_2-criterion	1.6	0.907
$p = 3$	Complete search:A_s-criterion	1.833	1.729
	Algorithm:A_s-criterion	1.833	1.729
	Complete search:K_2-criterion	2.556	0.469
	Algorithm:K_2-criterion	2.556	0.469
$p = 4$	Complete search:A_s-criterion	2.875	0.289
	Algorithm:A_s-criterion	2.875	0.289
	Complete search:K_2-criterion	3	0
	Algorithm:K_2-criterion	3	0
$p = 5$	Complete search:A_s-criterion	4.5	0
	Algorithm:A_s-criterion	4.5	0
	Complete search:K_2-criterion	4.5	0
	Algorithm:K_2-criterion	4.5	0
	MA	1.333	3.333

	$n = 12, m = 5$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	1.720	4.112
	Algorithm:A_s-criterion	1.720	4.112
	Complete search:K_2-criterion	2.163	3.074
	Algorithm:K_2-criterion	2.389	2.845
	Systematic method	1.720	4.112
$p = 2$	Complete search:A_s-criterion	2.05	3.061
	Algorithm:A_s-criterion	2.05	3.061
	Complete search:K_2-criterion	2.083	1.458
	Algorithm:K_2-criterion	2.083	1.458
$p = 3$	Complete search:A_s-criterion	2.308	3.391
	Algorithm:A_s-criterion	2.467	2.46
	Complete search:K_2-criterion	3.556	0.938
	Algorithm:K_2-criterion	3.556	0.938
$p = 4$	Complete search:A_s-criterion	3.762	0.599
	Algorithm:A_s-criterion	3.762	0.599
	Complete search:K_2-criterion	5	0
	Algorithm:K_2-criterion	5	0
$p = 5$	Complete search:A_s-criterion	5.714	0
	Algorithm:A_s-criterion	5.714	0
	Complete search:K_2-criterion	5.714	0
	Algorithm:K_2-criterion	5.714	0
	MA	1.667	5.833

	$n = 12, m = 6$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	2.067	6.653
	Algorithm:A_s-criterion	2.067	6.887
	Complete search:K_2-criterion	2.802	4.815
	Algorithm:K_2-criterion	2.802	4.815
	Systematic method	2.067	6.653
$p = 2$	Algorithm:A_s-criterion	2.511	4.685
	Algorithm:K_2-criterion	3.4	2.78
$p = 3$	Algorithm:A_s-criterion	3.119	4.418
	Algorithm:K_2-criterion	3.933	2.044
$p = 4$	Algorithm:A_s-criterion	4.667	0.944
	Algorithm:K_2-criterion	7	0.75
$p = 5$	Complete search:A_s-criterion	7	0
	Algorithm:A_s-criterion	7	0
	Complete search:K_2-criterion	7	0
	Algorithm:K_2-criterion	7	0
	MA	2	9.167

	$n=12, m=7$	A_s	K_2
$p=1$	Algorithm:A_s-criterion	2.414	9.692
	Algorithm:K_2-criterion	3.035	7.644
	Systematic method	2.414	9.811
$p=2$	Algorithm:A_s-criterion	2.982	7.962
	Algorithm:K_2-criterion	4.252	4.674
$p=3$	Algorithm:A_s-criterion	3.778	6.074
	Algorithm:K_2-criterion	4.482	3.174
$p=4$	Algorithm:A_s-criterion	5.606	2.898
	Algorithm:K_2-criterion	6.333	1.417
$p=5$	Algorithm:A_s-criterion	8.4	0
	Algorithm:K_2-criterion	8.4	0
	MA	2.333	13.417

	$n=12, m=8$	A_s	K_2
$p=1$	Algorithm:A_s-criterion	2.762	13.755
	Algorithm:K_2-criterion	3.752	12.289
	Systematic method	2.762	13.755
$p=2$	Algorithm:A_s-criterion	3.438	11.016
	Algorithm:K_2-criterion	4.733	7.506
$p=3$	Algorithm:A_s-criterion	4.75	7.858
	Algorithm:K_2-criterion	5.769	3.778
$p=4$	Algorithm:A_s-criterion	6.55	3.646
	Algorithm:K_2-criterion	6.55	2.458
$p=5$	Algorithm:A_s-criterion	10	0
	Algorithm:K_2-criterion	10	0
	MA	2.667	18.667

	$n=12, m=9$	A_s	K_2
$p=1$	Algorithm:A_s-criterion	3.111	18.494
	Algorithm:K_2-criterion	3.856	17.803
	Systematic method	3.111	18.494
$p=2$	Algorithm:A_s-criterion	3.905	14.223
	Algorithm:K_2-criterion	5.599	12.998
$p=3$	Algorithm:A_s-criterion	5.579	11.665
	Algorithm:K_2-criterion	9.781	9.928
$p=4$	Algorithm:A_s-criterion	7.5	4.5
	Algorithm:K_2-criterion	14.25	4
$p=5$	Algorithm:A_s-criterion	12	0
	Algorithm:K_2-criterion	12	0
	MA	3	25

	$n = 12, m = 10$	A_s	K_2
$p = 1$	Algorithm:A_s-criterion	3.843	25.493
	Algorithm:K_2-criterion	4.299	25.662
	Systematic method	3.462	24.172
$p = 2$	Algorithm:A_s-criterion	5.071	19.932
	Algorithm:K_2-criterion	5.993	17.334
$p = 3$	Algorithm:A_s-criterion	6.588	14.427
	Algorithm:K_2-criterion	10.9	13.64
$p = 4$	Algorithm:A_s-criterion	8.464	6.872
	Algorithm:K_2-criterion	15	6
$p = 5$	Algorithm:A_s-criterion	15	0
	Algorithm:K_2-criterion	15	0
	MA	3.33	32.5
	$C1$	9.13	5.32
	$OFAT$	15	0

	$n = 12, m = 11$	A_s	K_2
$p = 1$	Algorithm:A_s-criterion	4.634	33.842
	Algorithm:K_2-criterion	4.634	33.842
	Systematic method	3.813	30.8
$p = 2$	Algorithm:A_s-criterion	5.469	24.657
	Algorithm:K_2-criterion	6.832	26.804
$p = 3$	Algorithm:A_s-criterion	8.148	20.080
	Algorithm:K_2-criterion	8.533	17.489
$p = 4$	Algorithm:A_s-criterion	13.5	10.75
	Algorithm:K_2-criterion	19	12
$p = 5$	Algorithm:A_s-criterion	22	0
	Algorithm:K_2-criterion	22	0
	MA	3.667	41.25

B.3 Results for $n = 16$

	$n = 16, m = 2$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	0.508	0.379
	Algorithm:A_s-criterion	0.508	0.379
	Complete search:K_2-criterion	0.548	0.306
	Algorithm:K_2-criterion	0.508	0.379
	Systematic method	0.508	0.379
$p = 2$	Complete search:A_s-criterion	0.536	0.255
	Algorithm:A_s-criterion	0.536	0.255
	Complete search:K_2-criterion	0.833	0
	Algorithm:K_2-criterion	0.833	0
$p = 3$	Complete search:A_s-criterion	0.591	0.252
	Algorithm:A_s-criterion	0.591	0.252
	Complete search:K_2-criterion	0.733	0
	Algorithm:K_2-criterion	0.733	0
$p = 4$	Complete search:A_s-criterion	0.667	0.125
	Algorithm:A_s-criterion	0.667	0.125
	Complete search:K_2-criterion	0.75	0
	Algorithm:K_2-criterion	0.75	0
$p = 5$	Complete search:A_s-criterion	0.848	0.160
	Algorithm:A_s-criterion	0.848	0.160
	Complete search:K_2-criterion	0.867	0
	Algorithm:K_2-criterion	0.867	0
$p = 6$	Complete search:A_s-criterion	1.167	0
	Algorithm:A_s-criterion	1.167	0
	Complete search:K_2-criterion	1.167	0
	Algorithm:K_2-criterion	1.167	0
$p = 7$	Complete search:A_s-criterion	2.143	0
	Algorithm:A_s-criterion	2.143	0
	Complete search:K_2-criterion	2.143	0
	Algorithm:K_2-criterion	2.143	0
	MA	0.5	0.5

	$n = 16, m = 3$	A_s	K_2
$p = 1$	Complete search:A_s-criterion	0.762	1.418
	Algorithm:A_s-criterion	0.762	1.418
	Complete search:K_2-criterion	0.762	1.418
	Algorithm:K_2-criterion	1.726	0.755
	Systematic method	0.762	1.763
$p = 2$	Complete search:A_s-criterion	0.808	0.790
	Algorithm:A_s-criterion	0.808	0.790
	Complete search:K_2-criterion	1.029	0.743
	Algorithm:K_2-criterion	1.238	0.569
$p = 3$	Complete search:A_s-criterion	0.899	1.132
	Algorithm:A_s-criterion	0.899	0.747
	Complete search:K_2-criterion	0.932	0.336
	Algorithm:K_2-criterion	1.114	0.299
$p = 4$	Complete search:A_s-criterion	1	0.396
	Algorithm:A_s-criterion	1	0.396
	Complete search:K_2-criterion	1.5	0
	Algorithm:K_2-criterion	1.5	0
$p = 5$	Complete search:A_s-criterion	1.302	0.688
	Algorithm:A_s-criterion	1.302	0.541
	Complete search:K_2-criterion	1.429	0
	Algorithm:K_2-criterion	1.429	0
$p = 6$	Complete search:A_s-criterion	1.8	0
	Algorithm:A_s-criterion	1.8	0
	Complete search:K_2-criterion	1.8	0
	Algorithm:K_2-criterion	1.8	0
$p = 7$	Complete search:A_s-criterion	3.231	0
	Algorithm:A_s-criterion	3.231	0
	Complete search:K_2-criterion	3.231	0
	Algorithm:K_2-criterion	3.231	0
	MA	0.75	2.25

	$n=16, m=4$	A_s	K_2
$p=1$	Algorithm:A_s-criterion	1.017	2.690
	Algorithm:K_2-criterion	1.469	1.576
	Systematic method	1.017	2.919
$p=2$	Algorithm:A_s-criterion	1.083	1.786
	Algorithm:K_2-criterion	1.843	1.094
$p=3$	Algorithm:A_s-criterion	1.211	1.481
	Algorithm:K_2-criterion	1.321	0.651
$p=4$	Algorithm:A_s-criterion	1.333	1
	Algorithm:K_2-criterion	1.833	0.375
$p=5$	Algorithm:A_s-criterion	1.767	1.666
	Algorithm:K_2-criterion	2.333	0
$p=6$	Algorithm:A_s-criterion	2.5	0
	Algorithm:K_2-criterion	2.5	0
$p=7$	Algorithm:A_s-criterion	4.333	0
	Algorithm:K_2-criterion	4.333	0
	MA	1	3.75

	$n=16, m=5$	A_s	K_2
$p=1$	Algorithm:A_s-criterion	1.271	4.637
	Algorithm:K_2-criterion	1.943	2.938
	Systematic method	1.271	5.087
$p=2$	Algorithm:A_s-criterion	1.364	3.029
	Algorithm:K_2-criterion	2.756	2.0467
$p=3$	Algorithm:A_s-criterion	1.526	3.539
	Algorithm:K_2-criterion	1.842	1.021
$p=4$	Algorithm:A_s-criterion	1.667	1.792
	Algorithm:K_2-criterion	2.25	0.75
$p=5$	Algorithm:A_s-criterion	2.286	1.721
	Algorithm:K_2-criterion	2.88	0.420
$p=6$	Algorithm:A_s-criterion	3.333	0
	Algorithm:K_2-criterion	3.333	0
$p=7$	Algorithm:A_s-criterion	5.455	0
	Algorithm:K_2-criterion	5.455	0
	MA	1.25	6.5

	$n=16, m=6$	A_s	K_2
$p=1$	Algorithm:A_s-criterion	1.526	7.373
	Algorithm:K_2-criterion	2.149	4.972
	Systematic method	1.526	7.503
$p=2$	Algorithm:A_s-criterion	1.65	4.95
	Algorithm:K_2-criterion	2.359	3.312
$p=3$	Algorithm:A_s-criterion	1.864	3.627
	Algorithm:K_2-criterion	5.454	2.119
$p=4$	Algorithm:A_s-criterion	2	3.125
	Algorithm:K_2-criterion	3	1.125
$p=5$	Algorithm:A_s-criterion	2.768	3.386
	Algorithm:K_2-criterion	3.433	0.877
$p=6$	Algorithm:A_s-criterion	4.233	0.552
	Algorithm:K_2-criterion	4.5	0
$p=7$	Algorithm:A_s-criterion	6.6	0
	Algorithm:K_2-criterion	6.6	0
	MA	1.5	9.75

	$n=16, m=7$	A_s	K_2
$p=1$	Algorithm:A_s-criterion	1.871	11.254
	Algorithm:K_2-criterion	2.511	7.372
	Systematic method	1.781	10.936
$p=2$	Algorithm:A_s-criterion	1.944	7.079
	Algorithm:K_2-criterion	5.230	3.936
$p=3$	Algorithm:A_s-criterion	2.186	6.910
	Algorithm:K_2-criterion	3.267	3.387
$p=4$	Algorithm:A_s-criterion	2.5	4.707
	Algorithm:K_2-criterion	3.917	2.197
$p=5$	Algorithm:A_s-criterion	3.296	3.646
	Algorithm:K_2-criterion	4.361	1.357
$p=6$	Algorithm:A_s-criterion	5.158	2.529
	Algorithm:K_2-criterion	7	0
$p=7$	Algorithm:A_s-criterion	7.778	0
	Algorithm:K_2-criterion	7.778	0
	MA	1.75	14.25

	$n = 16, m = 8$	A_s	K_2
$p = 1$	Algorithm:A_s-criterion	2.096	14.164
	Algorithm:K_2-criterion	2.581	10.438
	Systematic method	2.036	13.754
$p = 2$	Algorithm:A_s-criterion	2.253	10.305
	Algorithm:K_2-criterion	3.256	7.654
$p = 3$	Algorithm:A_s-criterion	2.572	7.508
	Algorithm:K_2-criterion	4.299	4.606
$p = 4$	Algorithm:A_s-criterion	2.667	6.5
	Algorithm:K_2-criterion	5.541	3.709
$p = 5$	Algorithm:A_s-criterion	4.062	3.657
	Algorithm:K_2-criterion	5.533	2.049
$p = 6$	Algorithm:A_s-criterion	6.039	2.823
	Algorithm:K_2-criterion	8.5	0.75
$p = 7$	Algorithm:A_s-criterion	9	0
	Algorithm:K_2-criterion	9	0
	MA	2	17.75

	$n = 16, m = 9$	A_s	K_2
$p = 1$	Algorithm:A_s-criterion	2.291	17.633
	Algorithm:K_2-criterion	3.147	14.916
	Systematic method	2.291	17.619
$p = 2$	Algorithm:A_s-criterion	2.559	14.640
	Algorithm:K_2-criterion	4.361	9.621
$p = 3$	Algorithm:A_s-criterion	2.811	11.934
	Algorithm:K_2-criterion	4.529	7.154
$p = 4$	Algorithm:A_s-criterion	3	8.75
	Algorithm:K_2-criterion	5.032	5.085
$p = 5$	Algorithm:A_s-criterion	4.347	5.415
	Algorithm:K_2-criterion	5.516	3.168
$p = 6$	Algorithm:A_s-criterion	6.933	1.78
	Algorithm:K_2-criterion	7.2	1.427
$p = 7$	Algorithm:A_s-criterion	10.286	0
	Algorithm:K_2-criterion	10.286	0
	MA	2.25	22.5

	$n = 16, m = 10$	A_s	K_2
$p = 1$	Algorithm:A_s-criterion	2.588	23.823
	Algorithm:K_2-criterion	3.491	21.075
	Systematic method	2.546	23.288
$p = 2$	Algorithm:A_s-criterion	2.934	18.669
	Algorithm:K_2-criterion	4.005	14.849
$p = 3$	Algorithm:A_s-criterion	3.199	14.463
	Algorithm:K_2-criterion	3.730	10.910
$p = 4$	Algorithm:A_s-criterion	3.667	10.866
	Algorithm:K_2-criterion	6.212	6.218
$p = 5$	Algorithm:A_s-criterion	4.859	7.909
	Algorithm:K_2-criterion	12.085	5.161
$p = 6$	Algorithm:A_s-criterion	7.869	3.771
	Algorithm:K_2-criterion	11.5	2.25
$p = 7$	Algorithm:A_s-criterion	11.667	0
	Algorithm:K_2-criterion	11.667	0
	MA	2.5	29.25

	$n = 16, m = 11$	A_s	K_2
$p = 1$	Algorithm:A_s-criterion	2.948	30.349
	Algorithm:K_2-criterion	3.881	28.081
	Systematic method	2.802	29.982
$p = 2$	Algorithm:A_s-criterion	3.239	23.859
	Algorithm:K_2-criterion	3.803	21.421
$p = 3$	Algorithm:A_s-criterion	3.653	20.751
	Algorithm:K_2-criterion	4.248	15.525
$p = 4$	Algorithm:A_s-criterion	3.667	15.313
	Algorithm:K_2-criterion	6.735	9.697
$p = 5$	Algorithm:A_s-criterion	6.242	9.556
	Algorithm:K_2-criterion	7.357	6.100
$p = 6$	Algorithm:A_s-criterion	8.808	4.454
	Algorithm:K_2-criterion	14	3.125
$p = 7$	Algorithm:A_s-criterion	13.2	0
	Algorithm:K_2-criterion	13.2	0
	MA	2.75	37.25

	$n = 16, m = 12$	A_s	K_2
$p = 1$	Algorithm:A_s-criterion	3.289	38.526
	Algorithm:K_2-criterion	3.633	37.114
	Systematic method	3.058	36.982
$p = 2$	Algorithm:A_s-criterion	3.583	30.529
	Algorithm:K_2-criterion	4.330	28.462
$p = 3$	Algorithm:A_s-criterion	4.292	26.675
	Algorithm:K_2-criterion	5.885	20.877
$p = 4$	Algorithm:A_s-criterion	4	19.5
	Algorithm:K_2-criterion	12.268	19.167
$p = 5$	Algorithm:A_s-criterion	6.651	11.700
	Algorithm:K_2-criterion	12.712	10.958
$p = 6$	Algorithm:A_s-criterion	9.75	7.734
	Algorithm:K_2-criterion	18.5	5
	MA	3	45.75

	$n = 16, m = 13$	A_s	K_2
$p = 1$	Algorithm:A_s-criterion	3.474	46.299
	Algorithm:K_2-criterion	3.579	45.256
	Systematic method	3.314	45.0299
$p = 2$	Algorithm:A_s-criterion	3.966	38.449
	Algorithm:K_2-criterion	4.504	36.559
$p = 3$	Algorithm:A_s-criterion	4.792	30.365
	Algorithm:K_2-criterion	5.115	27.064
$p = 4$	Algorithm:A_s-criterion	5.048	23.233
	Algorithm:K_2-criterion	8.25	20.518
$p = 5$	Algorithm:A_s-criterion	7.973	15.733
	Algorithm:K_2-criterion	15.107	15.618
$p = 6$	Algorithm:A_s-criterion	10.711	8.749
	Algorithm:K_2-criterion	22.667	6.583
	MA	3.25	55.5

	$n = 16, m = 14$	A_s	K_2
$p = 1$	Algorithm:A_s-criterion	3.911	56.126
	Algorithm:K_2-criterion	4.139	56.355
	Systematic method	3.57	54.129
$p = 2$	Algorithm:A_s-criterion	4.678	48.899
	Algorithm:K_2-criterion	4.722	46.844
$p = 3$	Algorithm:A_s-criterion	5.319	39.001
	Algorithm:K_2-criterion	5.588	35.205
$p = 4$	Algorithm:A_s-criterion	6.239	30.401
	Algorithm:K_2-criterion	7.794	26.264
$p = 5$	Algorithm:A_s-criterion	9.302	20.448
	Algorithm:K_2-criterion	11.611	18.583
$p = 6$	Algorithm:A_s-criterion	14.75	12.531
	Algorithm:K_2-criterion	19.85	9.095
	MA	3.5	66.5
	$C1$	13.09	7.38
	$OFAT$	21	0

	$n = 16, m = 15$	A_s	K_2
$p = 1$	Algorithm:A_s-criterion	4.256	67.449
	Algorithm:K_2-criterion	4.491	69.216
	Systematic method	3.827	64.286
$p = 2$	Algorithm:A_s-criterion	5.127	57.204
	Algorithm:K_2-criterion	5.674	60.079
$p = 3$	Algorithm:A_s-criterion	5.935	46.407
	Algorithm:K_2-criterion	6.369	45.879
$p = 4$	Algorithm:A_s-criterion	7.305	33.674
	Algorithm:K_2-criterion	7.971	35.226
$p = 5$	Algorithm:A_s-criterion	11.803	27.510
	Algorithm:K_2-criterion	11.915	26.433
$p = 6$	Algorithm:A_s-criterion	30	19.25
	Algorithm:K_2-criterion	25.5	15.75
	MA	3.75	78.75