ALGORITHMIC SEARCH OF BASELINE MINIMUM ABERRATION DESIGNS

by

Pei Li

B.Sc, Simon Fraser University 2009

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the

Department of Statistics and Actuarial Science

Faculty of Science

© Pei Li 2011

Simon Fraser University

Summer 2011

All rights reserved.
However, in accordance with the *Copyright Act of Canada*, this work may be reproduced, without authorization, under the conditions for "Fair Dealing." Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.

APPROVAL

Name: Pei Li

Degree: Master of Science

Title of Thesis: Algorithmic Search of Baseline Minimum Aberration Designs

Examining Committee:

Chair: Dr. Steve Thompson
Chair

_____

Dr. Boxin Tang
Senior Supervisor

_____

Dr. Tim Swartz
Internal Examiner

_____

Dr. Derek Bingham
External Examiner

Date Defended/Approved: _____

Abstract

In many practical applications, it is natural to introduce baseline parameterization if there is a control or baseline level of each factor. In this project, we search for baseline minimum aberration designs of 20 runs. For the number of factors that is thirteen or less, we apply the method of complete search. When the number of factors is greater than thirteen, we propose a new search algorithm, which provides tremendous savings in computing time as compared with the complete search procedure. The performance of the algorithm is examined, and it is found that our algorithm performs almost as well as the complete search.

Keywords: Baseline Parameterization; Minimum Aberration; Search Algorithm.

To My Dear Family

Acknowledgments

There are many people who have helped and supported me during my enjoyable years at Simon Fraser University. I cannot include everyone in this limited space, but thank you all for shedding light on my life journey.

It has been my great fortune to have Professor Boxin Tang to be my supervisor over the past two years. I appreciate his endless support and encouragement. I would not have reached this far without him.

I am very grateful to my examining committee members, Professors Tim Swartz and Derek Bingham, for their insightful comments and suggestions about the project.

Special thanks go to Professor X. Joan Hu for going beyond the call of duty to help me out when I needed it.

I am very thankful to all my friends in the department. In particular, I wish to thank Yu Xia, Jing Cai, Hua Zheng, Zhiwei Tang, Ryan Lekivetz, Jervyn Ang, Yingying Chen, Harsha Perera, Alisha Albert-Green, Huijing Wang, and Yan Zhang.

Lastly, I would like to express my deepest gratitude to my dear parents and my dear love Jian Gao for their support, guidance, understanding, and complete love.

Contents

Approval ii

Abstract iii

Dedication iv

Acknowledgments v

Contents vi

List of Tables viii

List of Figures ix

1 Introduction 1

1.1 Fractional factorial designs 1

1.2 Baseline parameterization 4

1.3 Overview of the remaining chapters 5

2 Minimum Aberration Criteria 6

2.1 Minimum aberration under orthogonal parameterization 6

2.2 Minimum K-aberration under baseline parameterization 8

3 Complete Search 11

3.1 Method of Complete Search 11

3.2 Searching for baseline MA designs of 20 runs 12

3.2.1 An illustrative example 13
3.2.2 Search results 13

4 Algorithmic Search **15**
4.1 The basic idea 15
4.2 Using an example to illustrate the algorithm 16
4.3 The new algorithm and search results 20

5 Performance of Our Algorithm **23**
5.1 Using histograms of K_3 to compare the accuracy 24
5.2 Influence of starting designs 27
5.3 Performance in broader context 33

6 Concluding Remarks **36**

Bibliography **37**

Appendices

A Baseline MA designs for m $\leq$ 13 (complete search) **38**

B Baseline optimal designs for m $\geq$14 (algorithmic search) **44**

List of Tables

1.1 Full factorial design under orthogonal parameterization 2
1.2 Fractional factorial design under orthogonal parameterization 3
1.3 Example of two factors 4

3.1 The sequence $K_2, K_3 \ldots$ for m = 13 13

List of Figures

4.1 Image of ocean floor 16

5.1 Histogram of K_3 [1] 24
5.2 Histogram of K_3 [2] 25
5.3 Histogram of K_3 [3] 26
5.4 Histogram of K_3 [4] 26
5.5 Histogram of K_3 [5] 27
5.6 Histogram of K_3 with 100 starting designs [1] 31
5.7 Histogram of K_3 with 100 starting designs [2] 33
5.8 Histogram of Positions 35

Chapter 1

Introduction

1.1 Fractional factorial designs

Investigators often perform experiments by varying the inputs of a process or a system in order to identify the reasons of changes in the response. For instance, in an agricultural experiment, the purpose is to study how the temperature, the light level, and the moisture level of soil affect the productivity of cucumbers. Experimenters may be interested in finding out:

1: If a low level in temperature can increase the productivity,

2: If a high level in light can increase the productivity, and

3: If a low temperature level and a high light level together can increase the productivity.

To answer these questions, experimenters conduct a series of tests to identify how they can change the level combinations of these factors to achieve an increase in the cucumber yield. The effect of a single factor, such as the temperature, is called the main effect. The effect of two factors together, such as the temperature and the light level, is called the two-factor interaction effect. Likewise the effect of k factors together is called the k-factor interaction effect.

Full factorial designs are widely used when an experiment involves more than one factor, since they allow estimation of all main effects and all interaction effects. Each of the factors can have multiple levels, such as high, medium, and low. In many practical situations, two-level factorial designs are most common, in which each factor consists of exactly two levels. We denote a two-level full factorial design as a 2^m design, where m is the number of factors involved. A full factorial design considers all the 2^m level combinations of factors. In the above cucumber yield example, suppose that each factor has two levels, high (1) and low (-1). Table 1.1 provides the following 2^3 design, where each row corresponds to an experiment and each column represents the level settings for a factor. In each experiment, we make a single observation that measures the number of cucumbers produced.

Table 1.1: Full factorial design under an orthogonal parameterization

Factor / Experiment	Temperature (F1)	Light (F2)	Moisture (F3)	F1*F2	F1*F3	F2*F3	F1*F2*F3
1	1	1	1	1	1	1	1
2	1	1	-1	1	-1	-1	-1
3	1	-1	1	-1	1	-1	-1
4	1	-1	-1	-1	-1	1	1
5	-1	1	1	-1	-1	1	-1
6	-1	1	-1	-1	1	-1	1
7	-1	-1	1	1	-1	-1	1
8	-1	-1	-1	1	1	1	-1

Each column is also a contrast since the sum of 1's and -1's in each column is zero. The first three columns give the main effect contrasts for the three factors. Every two columns are orthogonal contrasts. For every two columns, if we calculate the product

of the 1's and -1's for each row, then the sum of all these N products equals zero, where N is the number of rows. The last four columns give the interaction contrasts. This type of parameterization using the symbols 1's and -1's is called an orthogonal parameterization, as all effect contrasts are mutually orthogonal. We usually use 1's to represent the high level, and -1's the low level. However, the symbols are interchangeable if one would like to use 1's to represent the low level instead of the high level.

In this example, investigators need to perform eight experiments in order to consider all the possible level combinations of the three factors. As the number of factors increases, the number of required experiments grows rapidly. For ten factors, a full factorial design requires 2^{10}= 1024 experiments. In many practical situations, the experimenter may want to study a large number of factors that have possible effects on the output response. In an air bag test experiment, it is unrealistic to crash 1024 cars since it involves a large investment of money and time. In these situations, fractional factorial designs provide a way to address the problem. For the cucumber yield example, we can construct a fractional factorial design as listed in Table 1.2.

Table 1.2: Fractional factorial design under an orthogonal parameterization

Factor / Experiment	Temperature (F1)	Light (F2)	Moisture (F3=F1xF2)
1	1	1	1
2	1	-1	-1
3	-1	1	-1
4	-1	-1	1

We first write down a full factorial design for the first two factors, then produce the row products of 1's and -1's to represent the levels for the third factor. If we compare this design to the full factorial design, we find that it is a fraction that contains the first, fourth, sixth and seventh rows of the full factorial design. Thus, this design is

often called a half-replicate design. Since F3 is a product of F1 and F2, we call it a generator of the design. In general, a fractional factorial design is referred to as a 2^{m-p} design, where p is the number of generators.

1.2 Baseline parameterization

Instead of using an orthogonal parameterization, it is natural to introduce a baseline parameterization if there is a control or baseline level for each factor. In our cucumber yield example, if the investigators are instead interested in possible improvements in productivity by changing the current settings of the temperature, the light level, and the moisture level of soil, it is reasonable to have these current settings to represent the baseline levels, with the new settings representing the test levels. Similarly, in a dose response study, the standard dose may naturally constitute the baseline level, and the high-dose or low-dose may constitute the test levels. Under baseline parameterization, we denote the baseline level as 0, and the test level as 1 for each factor.

As an example to illustrate the difference between orthogonal parameterization and baseline parameterization, suppose that a design has two factors, each of two levels, low and high.

Table 1.3: Example of two factors

F1	*F2*	*Mean Response*
Low	**Low**	μ_1
Low	**High**	μ_2
High	**Low**	μ_3
High	**High**	μ_4

Under an orthogonal parameterization, the grand mean is

$$\mu = (\mu_1 + \mu_2 + \mu_3 + \mu_4)/4.$$

The main effects of factors one and two are

$$\beta_1 = (\mu_3 + \mu_4)/2 - (\mu_1 + \mu_2)/2 \text{ and } \beta_2 = (\mu_2 + \mu_4)/2 - (\mu_1 + \mu_3)/2.$$

The two-factor interaction is given by

$$\beta_{12} = (\mu_1 + \mu_4)/2 - (\mu_2 + \mu_3)/2.$$

Under the baseline parameterization, assume that the low levels are the baseline levels. Then the baseline effect is

$$\theta_0 = \mu_1.$$

The main effects of factors one and two are

$$\theta_1 = \mu_3 - \mu_1 \text{ and } \theta_2 = \mu_2 - \mu_1.$$

The two-factor interaction is

$$\theta_{12} = \mu_4 - \mu_2 - \mu_3 + \mu_1.$$

1.3 Overview of the remaining chapters

In the next chapter, we will first review the minimum aberration criteria under orthogonal parameterizations, then introduce the minimum K-aberration criterion under the baseline parameterization proposed by Mukerjee and Tang (2011). In Chapter 3, we will apply the method of complete search to obtain the baseline minimum aberration designs of 20 runs for the number of factors that is no greater than thirteen. In Chapter 4, we will introduce our new search algorithm, and then apply it to obtain the baseline optimal designs when the number of factors exceeds thirteen. Chapter 5 examines the performance of our search algorithm. The last chapter gives a brief summary of the project and discusses possible future work.

Chapter 2

Minimum Aberration Criteria

2.1 Minimum aberration under orthogonal parameterization

Fries and Hunter (1980) provided the minimum aberration criterion for selecting fractional factorial designs. As discussed in Chen, Sun & Wu (1993), this criterion is based on the hierarchical assumptions:

1: Lower order effects are more important than higher order effects, and

2: Effects of the same order are equally important.

To address the problem of selecting 2^{m-p} designs when some two-factor interactions are important, Ke and Tang (2003) introduced a minimum N-aberration criterion with the idea of minimizing contamination or bias. Based on this idea, Mukerjee and Tang (2011) proposed a new minimum aberration criterion under baseline parameterization. We will review the minimum N-aberration criterion and then introduce the new minimum aberration criterion in the next section.

Suppose that we want to estimate all the main effects and some important two-factor interactions. The other interactions may be nonnegligible and we assume that the hierarchical assumption applies to them. Ke and Tang (2003) proposed the minimum N-aberration criterion to search for designs that minimize the bias caused by

such nonnegligible interaction effects.

Under this scenario, the fitted model is given by

$$Y = \beta_0 I + W_1\gamma_1 + \epsilon,$$

where Y denotes a vector of n observations, β_0 is the grand mean, I is an n $\times$ 1 vector of 1's, W_1 is the model matrix for all main effects and important two-factor interactions, γ_1 is the vector of main effects and two-factor interactions, and ϵ is the uncorrelated random errors with mean zero and constant variance. Our true model is given by

$$Y = \beta_0 I + W_1\gamma_1 + W_2\gamma_2 + X_3\beta_3 + \ldots + X_m\beta_m + \epsilon,$$

where W_2 is the model matrix for the remaining two-factor interactions, X_j is the model matrix for j-factor interactions, and β_j is the vector of j-factor interactions.

By applying least square estimation for γ_1 to the fitted model, we have the expectation of the estimator,

$$\begin{aligned} E(\hat{\gamma}_1) &= E((W_1'W_1)^{-1}W_1'Y) \\ &= 1/nW_1'E(Y) \\ &= 1/nW_1'(\beta_0 I + W_1\gamma_1 + W_2\gamma_2 + X_3\beta_3 + ... + X_m\beta_m) \\ &= \gamma_1 + 1/nW_1'W_2\gamma_2 + W_1'X_3\beta_3 + ... + 1/nX_1'X_m\beta_m \\ &= \gamma_1 + (P_2\gamma_2 + P_3\beta_3 + ... + P_m\beta_m) \end{aligned}$$

where $P_2 = n^{-1}W_1'W_2$ and $P_j = n^{-1}W_1'X_j$ for $j \geq 3$. This derivation shows that $P_2\gamma_2 + P_3\beta_3 + \ldots + P_m\beta_m$ is the bias or contamination of the estimation of γ_1. Since γ_2 and β_3, ..., β_m are unknown, one way of minimizing the bias is to minimize the size of matrices P_2, ..., P_m. The size can be measured by defining $N_j = \|P_j\|^2 = trace(P_j'P_j) = \sum_{i,k} p_{i,k}^2$, where $p_{i,k}$ is the element in the ith row, k^{th} column in the matrix P_j. Therefore, we want to sequentially minimize N_2, ..., N_m, where N_j can be interpreted as the number of j-factor interactions aliased with the main effects and important two-factor interactions in the fitted model. Ke and Tang (2003) call this criterion, the minimum N-aberration criterion.

2.2 Minimum K-aberration under baseline parameterization

In the spirit of minimizing contamination, Mukerjee and Tang (2011) proposed a minimum K-aberration criterion for selecting 2^{m-p} designs under baseline parameterization. Consider a setup of 2^m factorials, where m is the number of factors $F_1, \ldots, F_m$. Each factor has two levels, the baseline level 0 and the test level 1. Let $j_1, \ldots, j_m$ represent a typical treatment combination taking the values of zeros and ones, and $\tau(j_1 \ldots j_m)$ represent the effect of that combination. Then the baseline effect is $\theta_0 = \tau(00...0)$,

the main effects of F_1 and F_2 are

$\theta_1 = \tau(10...0) - \tau(00...0)$ and $\theta_2 = \tau(01...0) - \tau(00...0)$,

and the two-factor interaction F_1 F_2 is

$\theta_{12} = \tau(11...0) - \tau(10...0) - \tau(01...0) + \tau(00...0)$.

Or equivalently, we have

$\tau(00...0) = \theta_0$,

$\tau(10...0) = \theta_0 + \theta_1$,

$\tau(01...0) = \theta_0 + \theta_2$, and

$\tau(11...0) = \theta_0 + \theta_1 + \theta_2 + \theta_{12}$.

In the same manner, for more than two factors, the general form is given by

$$\tau(j_1...j_m) = \theta_0 + \Sigma_{s=1}^{m} \Sigma_s (\Pi_{w=1}^{s} j_{g_w}) \theta_{g_1} ... \theta_{g_s}, \quad (1)$$

where $\sum_s$ denotes sum over the subsets $(g_1 \ldots g_s)$ of the collection of the $\binom{m}{s}$ s-tuples, $1 \leq g_1 < \ldots < g_s \leq m$.

If we assume the absence of all interactions, then equation (1) reduces to

$$\tau(j_1...j_m) = \theta_0 + j_1\theta_1 + \ldots + j_m\theta_m, \quad (2)$$

for every $j_1, \ldots, j_m$, where $\theta_1, \ldots, \theta_m$ represent the main effects.

Equation (2) can be viewed as a linear model

$$E(Y) = [I_N Z]\theta \quad (3)$$

if we let Y denote the vector of observations, and Z = $(Z_1, \ldots, Z_m)$' is an N × m array with the $(a, b)^{th}$ element Z_{ab}, $1 \leq a \leq N$, $1 \leq b \leq m$, $\theta = (\theta_0, \theta_1, \ldots, \theta_m)$'. With this main effect model (3), Mukerjee and Tang (2011) presented the following result.

Proposition 1. *If Z forms an orthogonal array of strength two, then the associated design is universally optimal among all N-run designs for inference on* $\theta_1, \ldots, \theta_m$.

If all interaction effects are negligible, the least square estimator $(\hat{\theta}_1, \ldots, \hat{\theta}_m)'$ is the best linear unbiased estimator of $(\theta_1, \ldots, \theta_m)'$. If some of the interactions are non-negligible, $(\hat{\theta}_1, \ldots, \hat{\theta}_m)'$ becomes biased, and we want to minimize the bias. Mukerjee and Tang (2011) obtained an expression of the bias as stated in Lemma 1.

Lemma 1. *If no assumption is made about the absence of interactions then the bias or contamination of* $(\hat{\theta}_1, \ldots, \hat{\theta}_m)'$ *as an estimator of* $(\theta_1, \ldots, \theta_m)'$ *is given by* $\sum_{s=2}^{m} B_s \theta^{(s)}$, *where for example* $\sum_{s=3}$ *denotes sum over the subsets* (g_1, g_2, g_3) *of the collection of the* $\binom{m}{3}$ *3-tuples,* $1 \leq g_1 < g_2 < g_3 \leq m$, B_s *is the alias matrix associated with the s-factor interactions, and* $\theta^{(s)}$ *is the vector of all the s-factor interaction effects.*

Thus, applying the method of minimizing bias under the hierarchical assumptions, we are led to sequentially minimizing $K_2, \ldots, K_m$, where $K_s = trace(B_s{}'B_s)$. We name this minimum aberration criterion as the Minimum K-aberration criterion.

Mukerjee and Tang (2011) provided a detailed expression for K_s, as in the following lemma.

Lemma 2. *(a) For* $2 \leq s \leq m - 1$, $K_s = (4/N^2)(sT_{1s} + T_{2s})$, *where*

$T_{1s} = \sum_s [\alpha(g_1 \ldots g_s)]^2$,

$T_{2s} = \sum_{s+1} \{[2\alpha(g_1 \ldots g_{s+1}) - \alpha(g_1 \ldots g_s)]^2 + [2\alpha(g_1 \ldots g_{s+1}) - \alpha(g_1 \ldots g_{s-1} g_{s+1})]^2 + \ldots$
$+ [2\alpha(g_1 \ldots g_{s+1}) - \alpha(g_1 g_3 \ldots g_{s+1})]^2 + [2\alpha(g_1 \ldots g_{s+1}) - \alpha(g_2 g_3 \ldots g_{s+1})]^2\}$.

(b) $K_m = (4m/N^2)[\alpha(12 \ldots m)]^2$,

where $\alpha(g_1 \ldots g_s)$ *is defined as "the number of times the vector (1 1 ... 1) occurs as a row in the* $N \times s$ *subarray given by the* $g_1^{th} \ldots g_s^{th}$ *columns of Z"*

Mukerjee and Tang (2011) showed that the minimization of K_2 is equivalent to that of A_3, the number of defining words of length three in the defining relation for the minimum aberration criterion under orthogonal parameterization. Thus, symbol interchanging under baseline parameterization does not change the value of K_2.

As an example, lets find K_3 and K_4 for the orthogonal array below, for which N = 8 and m = 4.

$$\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{array}$$

$$\begin{aligned} T_{13} &= \textstyle\sum_3 [\alpha(g_1 g_2 g_3)]^2 \\ &= [\alpha(1,2,3)]^2 + [\alpha(1,2,4)]^2 + [\alpha(1,3,4)]^2 + [\alpha(2,3,4)]^2 \\ &= 1^2 + 1^2 + 1^2 + 1^2 = 4 \\ T_{2s} &= \textstyle\sum_4 \{[2\alpha(g_1 \ldots g_4) - \alpha(g_1 g_2 g_3)]^2 + [2\alpha(g_1 \ldots g_4) - \alpha(g_1 g_2 g_4)]^2 + \ldots \\ &\quad + [2\alpha(g_1 \ldots g_4) - \alpha(g_1 g_3 g_4)]^2 + [2\alpha(g_1 \ldots g_4) - \alpha(g_2 g_3 g_4)]^2\} \\ &= (2 \times 1 - 1)^2 \times 4 = 4 \end{aligned}$$

$$K_3 = (4/8^2)(3T_{13} + T_{23}) = (1/16)(3 \times 4 + 4) = 1$$

$$K_4 = (4 \times 4/8^2)[\alpha(1,2,3,4)]^2 = 1/4 \times 1 = 1/4$$

We have reviewed the minimum K-aberration criterion and explained how the associated $K_2, \ldots, K_m$ can be calculated under the baseline parameterization. In the next chapter, we will discuss how to search for baseline minimum aberration designs using a complete search algorithm.

Chapter 3

Complete Search

3.1 Method of Complete Search

Under baseline parameterization, the control level and test level are not interchangeable, so the isomorphic designs under orthogonal parameterization are no longer isomorphic. Mukerjee and Tang (2011) stated the following definitions of isomorphism.

Definition 1. *Two designs are isomorphic if one design can be obtained from the other by row and column permutations.*

We use OA (N, m, 2, 2) to denote a two-symbol orthogonal array of strength two with N runs, m factors, where strength 2 means that the four level combinations (0, 0), (0, 1), (1, 0), and (1, 1) occur with the same frequency for every two columns.

Definition 2. *Two orthogonal arrays are combinatorially isomorphic if one can be obtained from the other by row and columns permutations as well as renaming of symbols in one or more columns.*

A baseline minimum aberration (MA) design is defined as the design that sequentially minimizes $K_2, K_3 \ldots$. Thus, if we collect all the nonisomorphic baseline designs, we can sequentially compare their $K_2, K_3 \ldots$, and obtain the MA design. For any given N and m, suppose we are able to list all the combinatorially nonisomorphic OA (N, m, 2, 2)'s with symbols 0 and 1, and say there are p such designs. Each of these designs can generate 2^m designs by interchanging symbols in none, or any one or more of the m columns. Some of these designs are isomorphic to each other,

but the collection of all the 2^m designs contains all the nonisomorphic designs under baseline parameterization. Since symbol interchanging does not change the K_2 value as discussed in Section 2.2, we can first obtain the K_2s for all the p designs instead of recording the entire sequence $K_2, K_3 \ldots$ at once. We only keep the designs with minimum K_2, and say there are q such designs. For each of these q designs, we generate all 2^m designs and record their $K_3, K_4 \ldots$, then find the MA design that sequentially minimizes the sequence $K_3, K_4 \ldots$. In this way, we avoid calculating the entire sequence $K_3, K_4 \ldots$ for the designs that do not have minimum K_2, which cannot be the MA designs. The complete search method, as introduced in Mukerjee and Tang (2011), has the following steps:

STEP I List all the combinatorially nonisomorphic OA (N, m, 2, 2)'s for given N and m using the symbols 1 and 0, and denote the total number of such arrays as p.

STEP II Compute K_2 for each of the p arrays. Find the arrays with minimum K_2, and suppose there are q such arrays.

STEP III For each of the q arrays, generate the $\binom{m}{0}$ array with no interchange of symbols of 0 and 1, $\binom{m}{1}$ arrays with interchanging symbols in any one of the m columns, $\binom{m}{2}$ arrays with interchanging symbols in any two columns, ..., and $\binom{m}{m}$ array with interchanging symbols in all m columns. Therefore, we generate 2^m arrays for each of the q arrays, thus q$\times 2^m$ arrays in total.

STEP IV For each of the q$\times 2^m$ arrays, compute the sequence $K_3, K_4 \ldots$. Find the arrays that sequentially minimize $K_3, K_4 \ldots$ and hence the MA designs under baseline parameterization.

3.2 Searching for baseline MA designs of 20 runs

Mukerjee & Tang (2011) provided the tables for baseline MA designs of 12 and 16 runs, which allow the experimenter to study up to 15 factors. In this section, we use the method in Section 3.1 to search for baseline MA designs of 20 runs. This allows investigators to include up to 19 factors in their experiments.

3.2.1 An illustrative example

For N =20 and m = 13, there are 730 combinatorially nonisomorphic OA (N, m, 2, 2)'s. After calculating the K_2s for all these arrays, we obtain five arrays with minimum K_2 = 50.94. For each of these arrays, we generate 2^{13} designs by interchanging symbols and record the sequence $K_3, K_4 \ldots$. After sequentially comparing the sequences, we find one optimal design for each of the five arrays, and the corresponding $K_2, K_3 \ldots$ are listed below. The table reveals that the last three designs have a smaller K_3 value (90.47) than the first two designs. All the last three designs sequentially minimize the sequence $K_2, K_3 \ldots$. Thus, these three designs are the MA designs under baseline parameterization.

Table 3.1: The sequence $K_2, K_3 \ldots$ for m = 13

K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}	K_{12}	K_{13}
50.94	90.96	97.35	68.9	33.28	10.92	2.34	0.26	0	0	0	0
50.94	90.96	97.35	68.9	33.28	10.92	2.34	0.26	0	0	0	0
50.94	**90.47**	**95.93**	**65.76**	**28.6**	**7.15**	**0.78**	**0**	**0**	**0**	**0**	**0**
50.94	**90.47**	**95.93**	**65.76**	**28.6**	**7.15**	**0.78**	**0**	**0**	**0**	**0**	**0**
50.94	**90.47**	**95.93**	**65.76**	**28.6**	**7.15**	**0.78**	**0**	**0**	**0**	**0**	**0**

3.2.2 Search results

Using the complete search algorithm, we have obtained the optimal two-level 20-run fractional factorial designs for m between 5 and 13 under baseline parameterization. The results are given in Appendix A. For m greater than 13, we can still apply the algorithm and perform a complete search of optimal designs. However, it becomes very time consuming since it involves an enormous amount of evaluations and comparisons of the sequence $K_2, K_3 \ldots$. The software used is R. For m = 13, there are five combinatorially nonisomorphic designs with minimum K_2, and it takes three days to

complete the search for each of these five designs. For m = 14, there are 328 combinatorially nonisomorphic designs, and among them, we have found 30 designs that have minimum K_2. If it also requires three days for each design, we will need at least 90 days to complete the entire search. The computations behind are as follows. For each of the 30 designs, we generate 2^{14} (=16,384) designs by interchanging symbols. Therefore, we need to evaluate 328 K_2s and 30 $\times$ 16,384 (=491,520) K_3s. If we want to compute $K_4, K_5, \ldots, K_{14}$, it will require additional 491,520 evaluations each time to obtain one more element in the sequence. The major cause of the rapid growth in the amount of evaluations is that the number of 2^m designs generated by symbol permutations doubles every time m is increased by one. In the next chapter, we will propose a fast algorithm to increase the computational efficiency.

Chapter 4

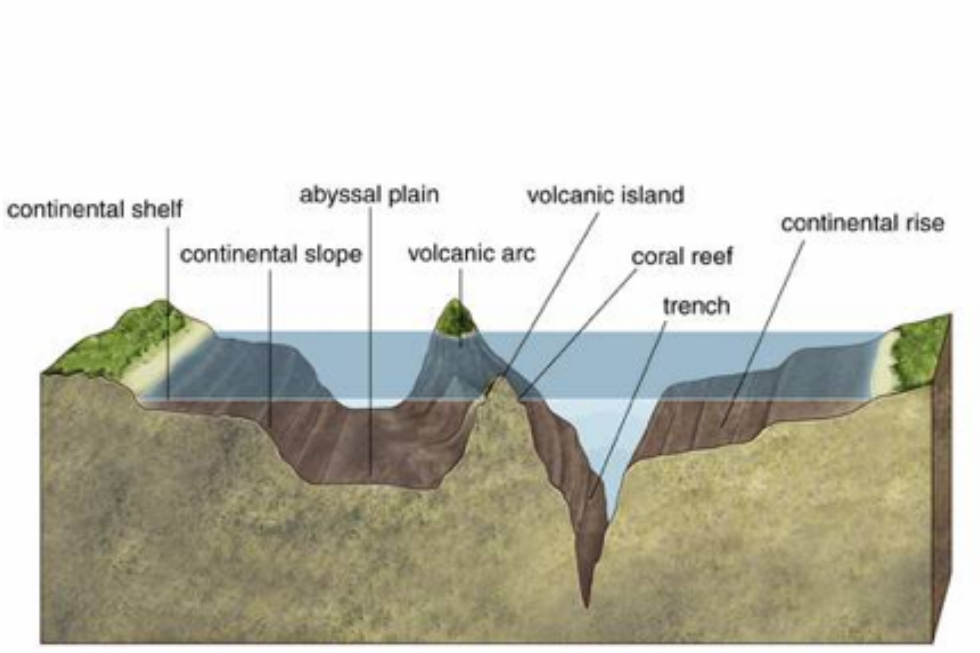
Algorithmic Search

In the previous chapter, we obtained the table for MA designs under baseline parameterization for N = 20 and m $\leq$ 13. When m $\geq$ 14, the complete search algorithm requires a very large amount of computation. We will propose a new algorithm in this chapter in order to reduce the time required to obtain the optimal designs.

4.1 The basic idea

Suppose that a scientific team wishes to explore the deepest point in an ocean. The scientists start with measuring the depth of the sea at one selected location (L_0), and additional six locations ($L_1, \ldots, L_6$) that are evenly spread out in the neighbourhood of L_0 and have the same distance, say 20 km away from L_0. After measuring all the depths, the scientists move to the location that gives the largest depth measure, and again measure the depth for its neighbourhoods. Every time they detect a deeper point, they are likely becoming closer to the deepest point. They would keep moving to new locations until they cannot find a deeper neighbour within the radius of 20 km, and the final location is the deepest point found (L_f). However, if solely considering the neighbours that are only 20 km away, they may fail to detect a deeper neighbour as the result of being trapped into an abyssal plain or an ocean trench that is not the deepest point.

Figure 4.1: Image of ocean floor



To reduce the chance of being trapped, they can scan a larger area by measuring the neighbours that are 40 km away from L_f. If they cannot find a deeper neighbour within the radius of 40 km, then that location is likely the deepest point in the ocean with the starting location L_0. One can always increase the size of scanned area until the entire ocean is covered. Instead of having only a single starting location, they could choose many starting locations that are evenly spread out in the ocean, so that they have a better chance to find the deepest point in the entire ocean.

4.2 Using an example to illustrate the algorithm

The new algorithm uses an idea similar to the method applied to explore the deepest point in an ocean. As a simple example, suppose we are searching for the optimal

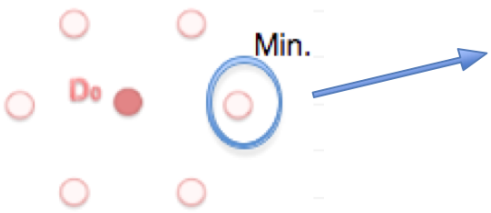
20-run two-level fractional factorial designs with six factors (i.e. m=6) under baseline parameterization. Following the Steps I & II in the previous chapter, we find that there are 13 designs that have a minimum K_2 among all the 75 combinatorially nonisomorphic designs. Choose any design with minimum K_2, and denote it as D_0.

D_0

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	0	1	1	1	0	1
[2,]	0	1	1	0	1	0
[3,]	0	1	0	1	1	0
[4,]	0	1	0	0	1	1
[5,]	0	1	0	0	0	0
[6,]	0	0	1	1	0	0
[7,]	0	0	1	0	1	1
[8,]	0	0	1	0	0	0
[9,]	0	0	0	1	1	1
[10,]	0	0	0	1	0	1
[11,]	1	1	1	1	0	0
[12,]	1	1	1	0	1	1
[13,]	1	1	1	0	0	1
[14,]	1	1	0	1	1	0
[15,]	1	1	0	1	0	1
[16,]	1	0	1	1	1	1
[17,]	1	0	1	1	1	0
[18,]	1	0	0	0	1	0
[19,]	1	0	0	0	0	1
[20,]	1	0	0	0	0	0

$$(K_2, K_3, K_4, K_5, K_6) = (8.10, 5.26, 2.08, 0.50, 0.06)$$

Find all the six neighbours around D_0 by interchanging symbols in one and only one of the six columns. Compute the sequence $K_3, K_4 \ldots$ for both D_0 and its six neighbours, and put them in the following table:



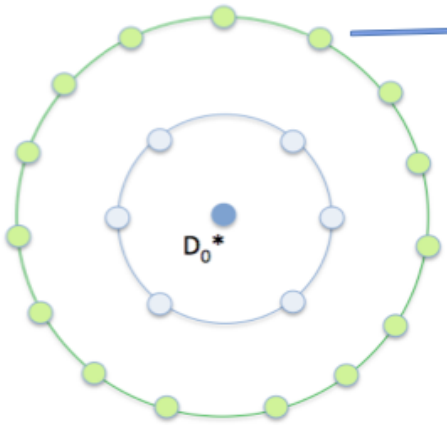
	K_2	K_3	K_4	K_5	K_6
D_1	8.1	3.78	0.72	0.06	0
D_2	8.1	4.74	1.9	0.5	0.06
D_3	8.1	4.1	1.02	0.12	0
D_4	8.1	5.02	2	0.5	0.06
D_5	8.1	4.94	1.64	0.24	0
D_6	8.1	4.7	1.42	0.18	0

The above table shows that D_1 gives a smaller K_3 than all the other six designs, and hence D_1 is a better design. If we treat D_1 as our new D_0, denoted by D_0*, and obtain its six new neighbours, we may expect to find a better design than D_0* with an even smaller K_3. In our case, by treating D_1 as our new D_0, denoted by D_0*, we found these six new neighbours:

	K_2	K_3	K_4	K_5	K_6
D_1	8.1	4.42	1.16	0.12	0
D_2	8.1	4.7	1.42	0.18	0
D_3	8.1	4.66	1.42	0.18	0
D_4	8.1	4.9	1.64	0.24	0
D_5	8.1	4.22	1.4	0.36	0.06
D_6	8.1	4.06	1.02	0.12	0

D_6 minimizes K_3 (=4.06), $K_4 \ldots$ among the six neighbours, but D_0* has an even smaller K_3 (=3.78). Thus, D_0* is currently the optimal design compared to the neighbours that are one unit away.

Similar to the idea of avoiding the ocean trench, we would also like to check the neighbours that are two units away from D_0*. That is, compute the sequence $K_3, K_4 \ldots$ for the 15 (i.e. $\binom{m}{2}$) designs that are generated by interchanging symbols in two of the six columns. If none of these 15 designs is better than D_0*, we then find the optimal design within the distance of two units. If there is a better design, we replace D_0 with it and repeat the search starting from scanning the neighbours that are one unit away. In our example, the sequence $K_2, K_3 \ldots$ for the 15 neighbours are listed below:



	K_2	K_3	K_4	K_5	K_6
D_1	8.1	4.74	1.9	0.5	0.06
D_2	8.1	4.1	1.02	0.12	0
D_3	8.1	5.02	2	0.5	0.06
D_4	8.1	4.94	1.64	0.24	0
D_5	8.1	4.7	1.42	0.18	0
D_6	8.1	5.06	2	0.5	0.06
D_7	8.1	4.62	1.5	0.36	0.06
D_8	8.1	4.7	1.42	0.18	0
D_9	8.1	4.46	1.16	0.12	0
D_{10}	8.1	4.58	1.5	0.36	0.06
D_{11}	8.1	4.5	1.34	0.18	0
D_{12}	8.1	5.02	2	0.5	0.06
D_{13}	8.1	4.22	1.08	0.12	0
D_{14}	8.1	4.58	1.5	0.36	0.06
D_{15}	8.1	4.5	1.34	0.18	0

D_2 minimizes K_3 (=4.1), $K_4 \ldots$ among the 15 neighbours, but D_0* still has a smaller K_3 (=3.78). Thus, D_0* is the optimal design compared to the neighbours that are one and two units away, and it is given below:

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	1	1	1	1	0	1
[2,]	1	1	1	0	1	0
[3,]	1	1	0	1	1	0
[4,]	1	1	0	0	1	1
[5,]	1	1	0	0	0	0
[6,]	1	0	1	1	0	0
[7,]	1	0	1	0	1	1
[8,]	1	0	1	0	0	0
[9,]	1	0	0	1	1	1
[10,]	1	0	0	1	0	1
[11,]	0	1	1	1	0	0
[12,]	0	1	1	0	1	1
[13,]	0	1	1	0	0	1
[14,]	0	1	0	1	1	0
[15,]	0	1	0	1	0	1
[16,]	0	0	1	1	1	1
[17,]	0	0	1	1	1	0
[18,]	0	0	0	0	1	0
[19,]	0	0	0	0	0	1
[20,]	0	0	0	0	0	0

Moreover, if we add an extra step to the neighbour searching algorithm, we may find an even better design than the above one. We can achieve this by searching for the optimal designs using all the other twelve combinatorially nonisomorphic minimum K_2 designs. The corresponding sequences $K_2, K_3 \ldots$ are shown below:

	K_2	K_3	K_4	K_5	K_6
D_1	8.1	3.58	0.66	0.06	0
D_2	8.1	3.82	0.74	0.06	0
D_3	8.1	3.72	0.54	0	0
D_4	8.1	3.92	0.94	0.12	0
D_5	8.1	3.98	0.8	0.06	0
D_6	8.1	4.02	0.8	0.06	0
D_7	8.1	3.92	0.6	0	0
D_8	8.1	4.22	0.88	0.06	0
D_9	8.1	4.28	0.94	0.06	0
D_{10}	8.1	3.8	0.78	0.06	0
D_{11}	8.1	4	0.86	0.06	0
D_{12}	8.1	3.86	0.94	0.12	0
Existing D_0*	8.1	3.78	0.72	0.06	0

D_1 sequentially minimizes $K_2, K_3 \ldots$ among all the 13 designs, so D_1 is the optimal design among the optimal designs given by all the 13 minimum K_2 designs. This gives the same result as that from applying the complete search algorithm.

4.3 The new algorithm and search results

STEP A List all the combinatorially nonisomorphic OA (N, m, 2, 2)'s for given N and m using the symbols 1 and 0, and denote the total number of such arrays as p.

STEP B Compute K_2 for each of the p arrays. Find the arrays with minimum K_2, and suppose there are q such arrays. Denote these arrays as D_{0i}, where i equals 1, 2, ..., q.

STEP C For each D_{0i}, generate $\binom{m}{0}$ arrays with no interchange of symbols of 0 and 1, and $\binom{m}{1}$ arrays with interchanging symbols in any one of the m columns. Denote

these arrays as D_{0ij}, where j equals 0, 1, 2, ..., m. Compute the sequence $K_3, K_4 \ldots$ for $D_{0i0}, D_{0i1}, \ldots, D_{0im}$. Find the D_{0ij} that sequentially minimizes $K_3, K_4 \ldots$ among all the (m + 1) designs, and use it to replace the original D_{0i}. If there are more than one D_{0ij} that sequentially minimizes $K_3, K_4 \ldots$, then randomly choose one to replace the original D_{0i}.

STEP D Repeat STEP C until the sequence $K_3, K_4 \ldots$ for D_{0ij} is the same as that for D_{0i}. Denote the D_{0ij} as D_{0i}* when the repeat stops.

STEP E For each D_{0i}*, generate H = $\binom{m}{2}$ arrays with interchanging symbols in any two columns. Denote these arrays as D_{0ih}, where h equals 1, 2, ..., H. Compute the sequence $K_3, K_4 \ldots$ for $D_{0i0}, D_{0i1}, \ldots, D_{0iH}$. Find the D_{0ih} that sequentially minimizes $K_3, K_4 \ldots$ among all the (H + 1) designs. If the sequence $K_3, K_4 \ldots$ for D_{0ih} is the same as that for D_{0i}*, then D_{0i}* is the MA design found by the algorithm starting with D_{0i}. If not, use D_{0ih} to replace the original D_{0i}, and restart from STEP C. If there are more than one D_{0ih} that sequentially minimizes $K_3, K_4 \ldots$, then randomly choose one to replace the original D_{0i}.

STEP F Compare the sequence $K_3, K_4 \ldots$ for D_{01}*, ..., D_{0q}*. The D_{0i}* that sequentially minimizes $K_3, K_4 \ldots$ is the best found by the algorithm under the baseline parameterization.

Using the above algorithm, we have searched designs of 20 runs for $14 \leq$ m $\leq$17, and the results are in Appendix B. For m = 18, 19, the baseline minimum aberration designs are also given in Appendix B, and they are based on the theoretical results (Mukerjee and Tang 2011).

The method of complete search requires a comparison of 2^m designs for a given D_0. In contrast, the algorithm developed in this chapter only requires an evaluation of m designs for Step C and of $\binom{m}{2}$ designs if Step E is necessary. For designs of 20 runs with $14 \leq$ m ≤ 17, the number of iterations for Step C is in the range of one to five, and that for Step E is one or two. This results in tremendous savings in computing time as compared with the complete search.

Obviously, there is no guarantee that the algorithm will produce the baseline minimum aberration design as our algorithm is a version of local search algorithms (Aarts and Lenstra 2003). However, as will be demonstrated in Chapter 5, our algorithm

preforms quite well, at least for the cases we have examined.

Chapter 5

Performance of Our Algorithm

Recall that for both the complete search algorithm and our new algorithm, we start with searching for the designs with minimum K_2. The difference between these two algorithms occurs right after that step. For each design with minimum K_2, the complete search algorithm compares all the 2^m designs generated by symbol interchange, while our algorithm only compares the neighbours of each given design – the total number of the neighbours compared is often much smaller than 2^m. This chapter will assess how close the results are from these two algorithms.

To start the new algorithm, we need to first select a single design among the 2^m designs for a minimum K_2 design. If we select a design that is close to the global optimum, we would be able to find the global optimum with a small number of iterations. If we select a design that is very far away from the global optimum, we might reach the global optimum after a large number of iterations, or be trapped in a local optimum and never reach the global optimum. We are interested in finding out how different the results are if we use different starting designs for any given minimum K_2 design.

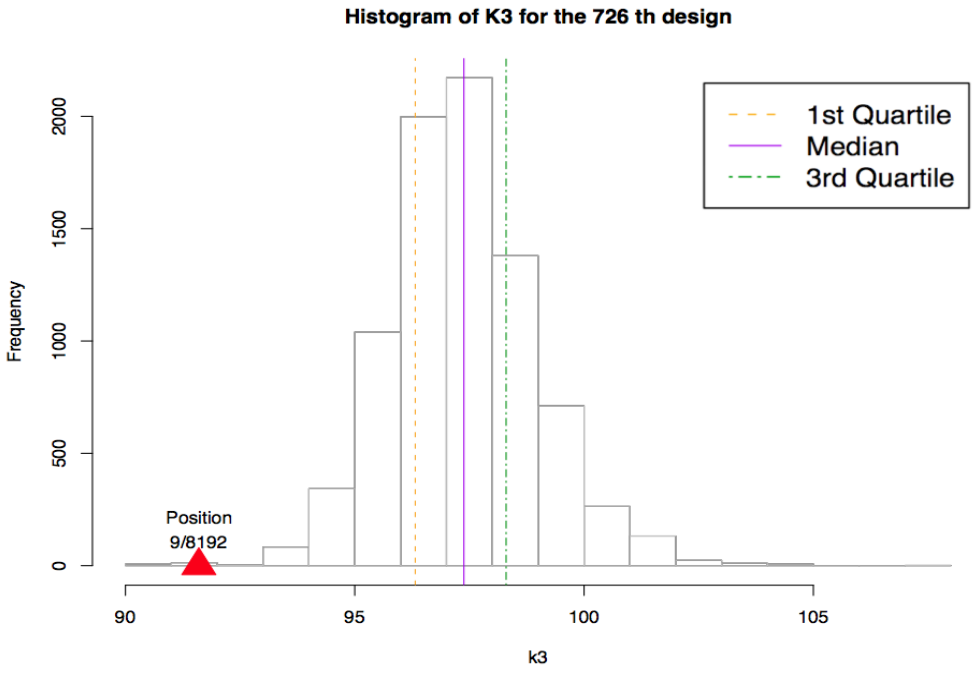
Moreover, it is also of interest to test how well our new algorithm works in a broader context. We want to know whether or not the algorithm of neighbour searching is able to lead to the best design if we just start with any one of the 20-run designs, not necessarily the minimum K_2 design.

5.1 Using histograms of K_3 to compare the accuracy

In the previous chapter, we obtained the minimum aberration designs for m = 5, 6, ..., and 13, using the complete search algorithm. In this section, we will compare these results with those obtained using our algorithm. Since almost all of the designs generated from the minimum K_2 designs can be distinguished from the comparison solely based on K_3 values, for convenience, we will only examine K_3 values instead of the entire sequence $K_3, K_4 \ldots$.

For m=13, there are five designs (the 726-730th designs) with minimum K_2 out of all the 730 combinatorially nonisomorphic designs. For each of the five designs, we can generate 2^{13} (=8192) designs by symbol permutations. We present the histogram of 8192 K_3 values for each of the five minimum K_2 designs.

Figure 5.1: Histogram of K_3 [1]



The histogram above is for the 726th design. The vertical lines, from the left to the right, are the first quartile, median, and the third quartile. The triangle mark indicates the position of the optimal design found using our algorithm, and we denote its corresponding K_3 value as K_3*. The histograms use K_3 values of all $2^m = 8192$ designs. If K_3* equals the minimum K_3 in the histogram, the position of the optimal design is one. This means that the optimal design found by our algorithm is also the optimal design found using complete search algorithm for a given minimum K_2 design. The triangle mark in the histogram above shows that the optimal design found by our algorithm is the 9th best design among all the 8192 designs generated by the 726th design. Likewise, we can also obtain the histograms for the 727-730th designs as shown below. We have also investigated the histograms for other m values. All these studies show that although the algorithm may not find the best design, it always produces very good designs.

Figure 5.2: Histogram of K_3 [2]

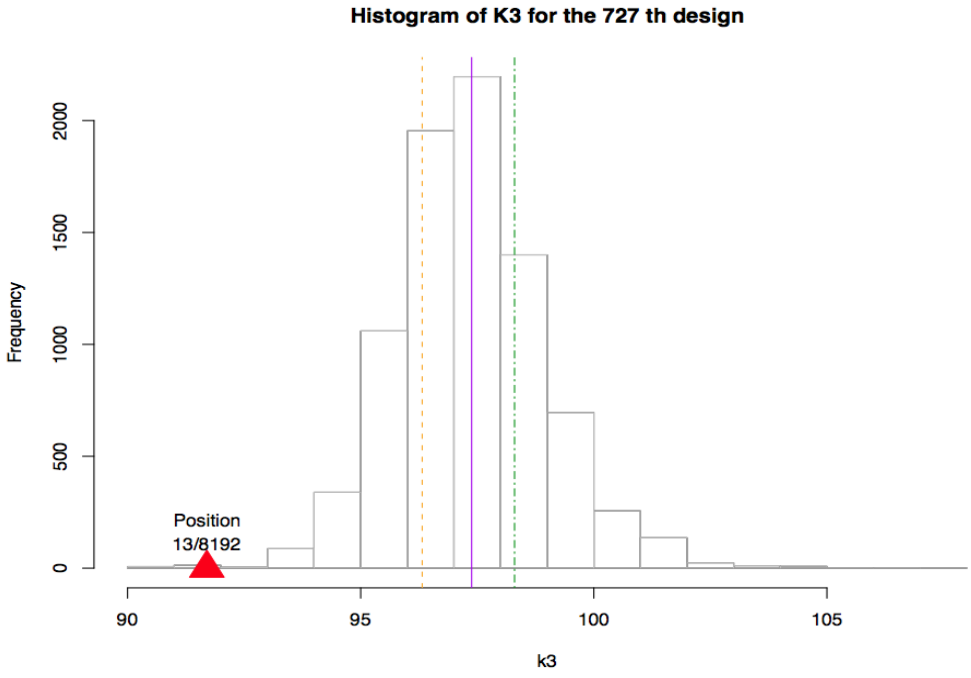


Figure 5.3: Histogram of K_3 [3]

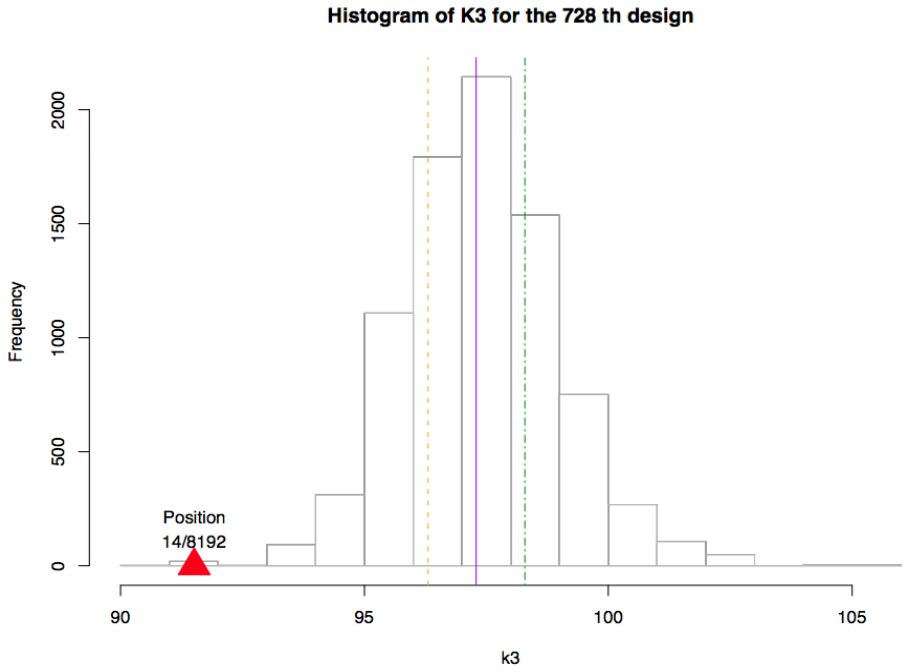


Figure 5.4: Histogram of K_3 [4]

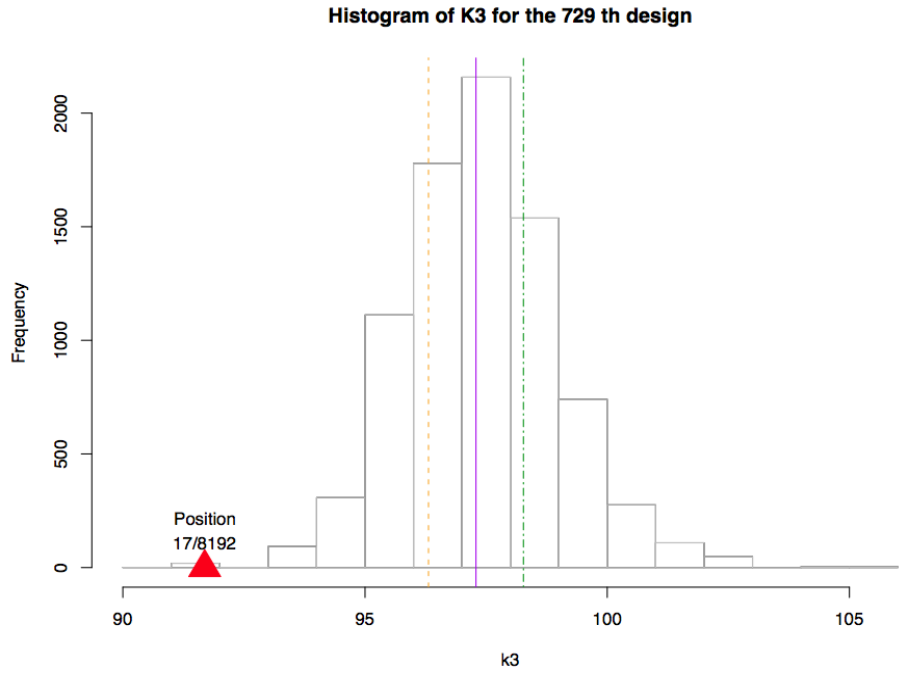
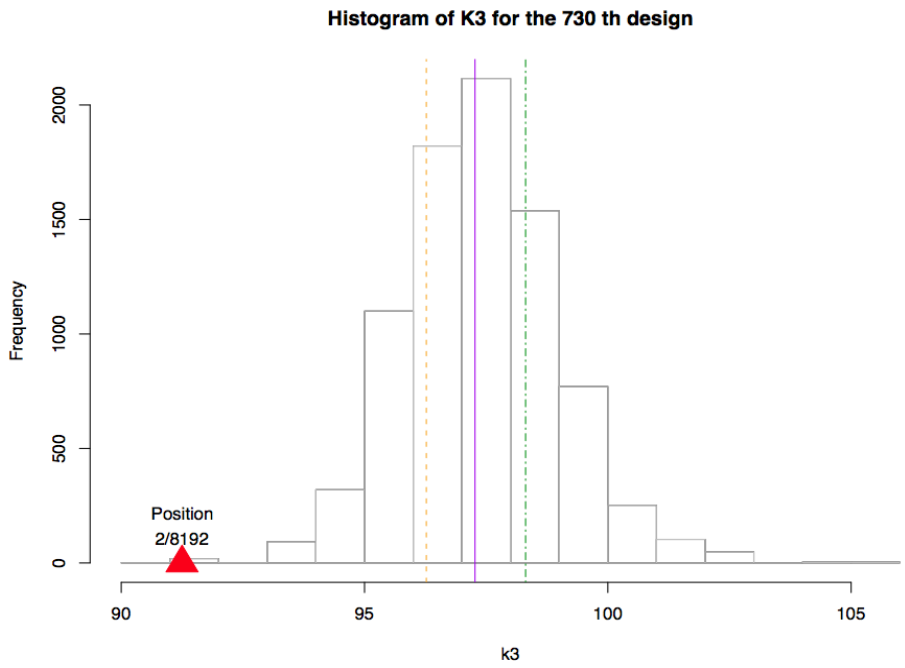


Figure 5.5: Histogram of K_3 [5]



5.2 Influence of starting designs

We start the search with listing all the combinatorially nonisomorphic designs. Among all these designs, we find those with minimum K_2. Using each of the minimum K_2 designs as a starting point, we perform the neighbour searching algorithm. Each minimum K_2 design used is just one of the 2^m designs generated by symbol interchange. Some of these 2^m designs are isomorphic; some are not, but they all share the same K_2 value. If we select two of these 2^m designs as the starting points, the paths of searching for the optimal designs will be different, and may lead to different designs. For instance, suppose that we are given m=5 and the minimum K_2 design listed below, which is also the global optimal design.

1	1	1	1	0
1	1	1	0	1
1	1	0	1	1
1	1	0	0	1
1	1	0	0	0
1	0	1	1	0
1	0	1	0	1
1	0	1	0	0
1	0	0	1	1
1	0	0	1	0
0	1	1	1	0
0	1	1	0	1
0	1	1	0	0
0	1	0	1	1
0	1	0	1	0
0	0	1	1	1
0	0	1	1	1
0	0	0	0	1
0	0	0	0	0
0	0	0	0	0

If we use this global optimal design as our starting point, we can definitely find the global optimum with (K_2, K_3, K_4, K_5) being (5.3, 1.69, 0.15, 0). If we interchange the symbols for all the columns of the global optimal design, and use the resulting design as the starting point, our algorithm leads to a local optimum of (K_2, K_3, K_4, K_5) being (5.3, 1.79, 0.36, 0.05).

1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0
1	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	0	0	0	0
0	0	0	0	0
0	0	1	1	0
0	0	1	1	1
0	0	1	1	1

From the local optimum given above, we found that the symbols for first two columns were changed back to the first two columns in the global optimum. However, we have no chance to interchange the symbols for all the other three columns since we were trapped in the local optimum. That is, symbol interchanging in any one or any two of the remaining three columns cannot result in a better design than the local optimum. If we generalize our current algorithm and allow having any three columns interchange symbols at once, we would then reach the global optimum. However, the generalization will require more computations, which decreases the computation efficiency. Instead of further generalizing our algorithm, we now investigate to what degree the current algorithm could be affected by different starting designs.

For any given m that is smaller than or equal to 13, where the complete search results are available, we can test the influence of the starting points. As an example, if we choose m=13. In this case, then we have five combinatorially nonisomorphic designs with minimum K_2. We randomly choose two of these five designs, the 726th and 728th designs. For each design, we also randomly choose a number of columns to interchange symbols from 0 to 13, and assume that the number is 4. List all the $\binom{13}{4} = 715$ combinations, and randomly choose one of these combinations, and

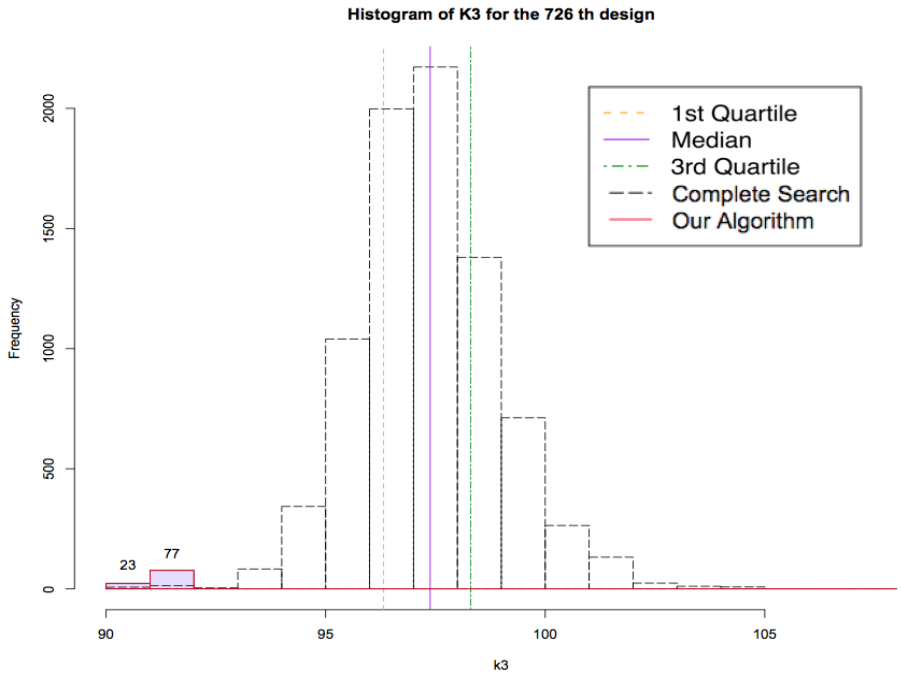
assume that it is (2, 6, 10,13). Next we interchange the symbols in columns 2, 6, 10, and 13 of the original minimum K_2 design. Use this design as our starting point, we perform the neighbour searching algorithm. For each original minimum K_2 design, we repeat the above steps 100 times so that we have 100 different starting points. Using each starting points, we obtain one optimal design and its corresponding $K_2, K_3 \ldots$. As in the previous section, we can record K_3 for each optimal design, and find its position by comparing that K_3 value with all the 2^{13} K_3 Values. Since we have 100 optimal designs with different starting points for each original minimum K_2 design, we obtain 100 corresponding positions. We give the histograms for each of the two combinatorially nonisomorphic designs chosen.

D726	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]	[,13]
[1,]	1	1	1	0	1	1	0	0	0	0	0	0	0
[2,]	1	1	0	1	1	0	1	0	1	1	0	1	0
[3,]	1	1	0	1	1	0	0	0	0	0	1	1	1
[4,]	1	1	0	0	0	1	0	1	0	1	1	1	0
[5,]	1	1	0	0	0	0	1	1	1	0	0	0	1
[6,]	1	0	1	1	0	1	1	0	0	0	1	1	1
[7,]	1	0	1	1	0	0	0	1	1	0	1	0	0
[8,]	1	0	1	0	1	0	1	1	0	1	1	0	1
[9,]	1	0	1	0	0	1	1	0	1	1	0	1	0
[10,]	1	0	0	1	1	1	0	1	1	1	0	0	1
[11,]	0	1	1	1	0	1	0	1	1	0	0	1	1
[12,]	0	1	1	1	0	0	1	0	0	1	0	0	1
[13,]	0	1	1	0	1	1	0	0	1	1	1	0	1
[14,]	0	1	1	0	1	0	1	1	1	0	1	1	0
[15,]	0	1	0	1	0	1	1	1	0	1	1	0	0
[16,]	0	0	1	1	1	0	0	1	0	1	0	1	0
[17,]	0	0	0	1	1	1	1	0	1	0	1	0	0
[18,]	0	0	0	0	1	1	1	1	0	0	0	1	1
[19,]	0	0	0	0	0	0	0	0	1	1	1	1	1
[20,]	0	0	0	0	0	0	0	0	0	0	0	0	0

Position	K_3	Count
1	90.96	2
2	90.98	15
3	91	6
4	91.08	4
5	91.6	30
6	91.7	43
	Total	100

For the first minimum K_2 design, the table shows that, two out of 100 times, we found the optimal design given by the complete search algorithm. In the worst case, we were able to find the 6th best design.

Figure 5.6: Histogram of K_3 with 100 starting designs [1]



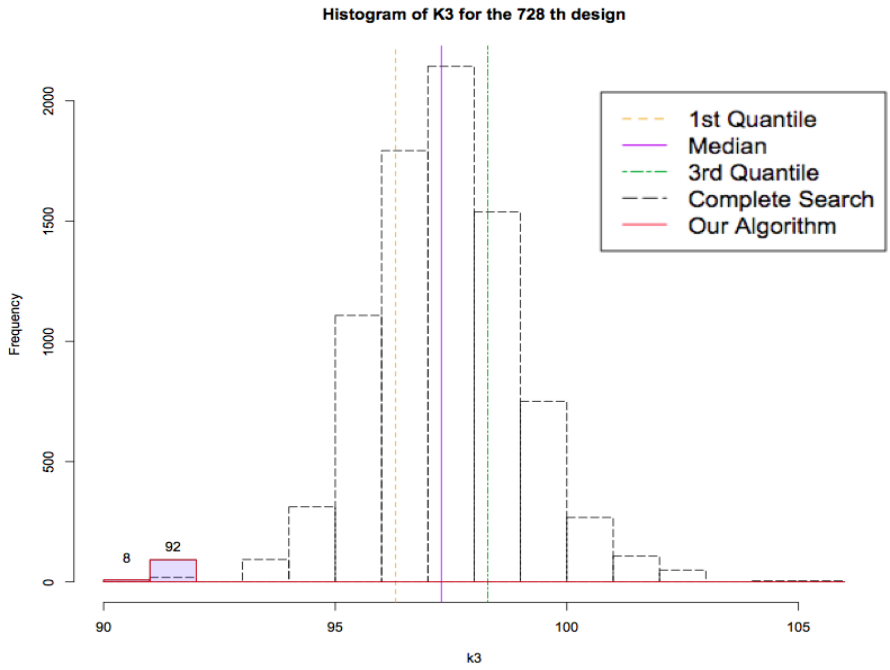
The histogram shows that 23 out of 100 times, K_3 is no larger than 91. All the 100 designs found by the algorithm are within the first percentile.

D728	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]	[,13]
[1,]	1	1	1	0	1	1	0	0	1	0	1	0	1
[2,]	1	1	0	1	1	0	0	1	1	1	0	0	1
[3,]	1	1	0	1	1	0	0	0	0	1	1	1	0
[4,]	1	1	0	0	0	1	1	0	1	1	0	1	0
[5,]	1	1	0	0	0	0	1	1	0	0	1	0	1
[6,]	1	0	1	1	0	1	0	0	1	0	0	1	1
[7,]	1	0	1	1	0	0	1	1	0	1	0	1	1
[8,]	1	0	1	0	1	0	1	1	1	0	1	1	0
[9,]	1	0	1	0	0	1	0	1	0	1	1	0	0
[10,]	1	0	0	1	1	1	1	0	0	0	0	0	0
[11,]	0	1	1	1	0	1	1	0	0	1	1	0	1
[12,]	0	1	1	1	0	0	0	1	1	0	0	0	0
[13,]	0	1	1	0	1	1	0	1	0	1	0	1	0
[14,]	0	1	1	0	1	0	1	0	0	0	0	1	1
[15,]	0	1	0	1	0	1	1	1	1	0	1	1	0
[16,]	0	0	1	1	1	0	1	0	1	1	1	0	0
[17,]	0	0	0	1	1	1	0	1	0	0	1	1	1
[18,]	0	0	0	0	1	1	1	1	1	1	0	0	1
[19,]	0	0	0	0	0	0	0	0	1	1	1	1	1
[20,]	0	0	0	0	0	0	0	0	0	0	0	0	0

Position	K3	Count
1	90.47	8
2	91.25	17
3	91.45	28
4	91.51	39
5	91.69	8
	Total	100

For the second minimum K_2 design, the table shows that, eight out of 100 times, we found the optimal design given by the complete search algorithm. In the worst case, we were able to find the 5th best design.

Figure 5.7: Histogram of K_3 with 100 starting designs [2]



The histogram shows that 8 out of 100 times, K_3 is no larger than 91. All the 100 designs found by the algorithm are within the first percentile.

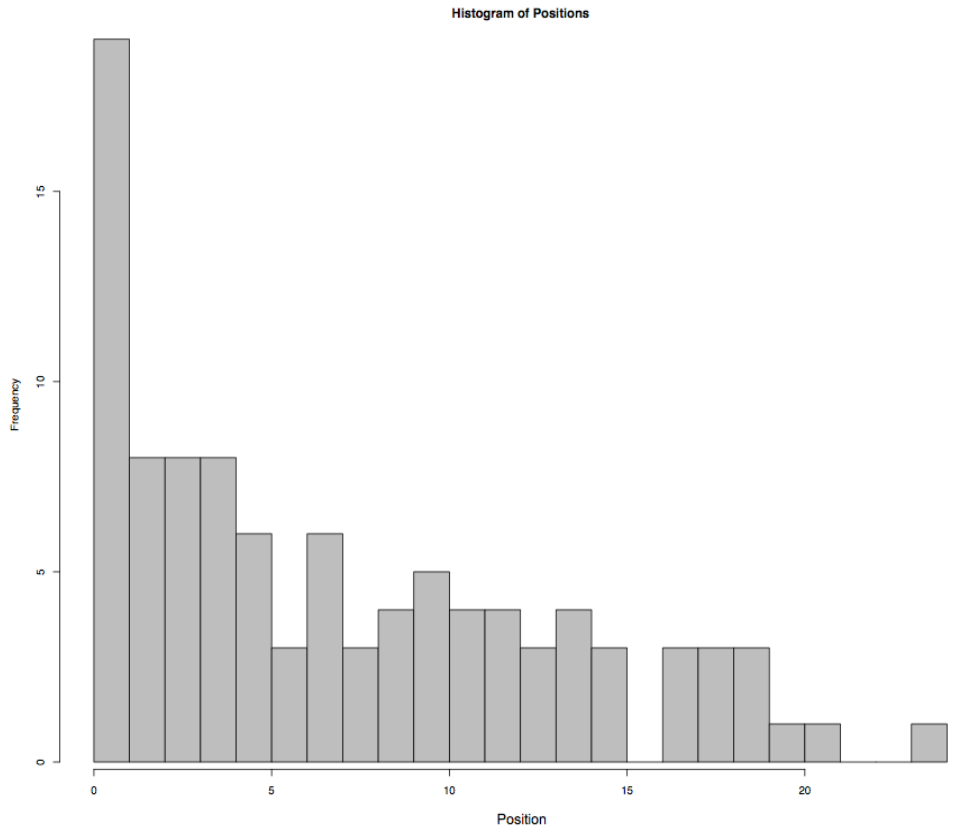
These two cases provide strong evidence that the results given by our algorithm are very robust to the starting design. Our algorithm always finds very good designs, regardless of the starting design.

5.3 Performance in broader context

For m=10, there exists 2389 combinatorially nonisomorphic designs. For each of these designs, we can generate all 2^{10} designs and record the $K_2, K_3 \ldots$. Then sort the K-series by sequentially comparing $K_2, K_3 \ldots$. The one that sequentially minimizes $K_2, K_3 \ldots$ sits in position one on the top, while the one that sequentially maximizes $K_2, K_3 \ldots$ sits in position 2^{10} on the bottom. Next using the same selected design, find

the optimal design by applying the new algorithm, and record its $K_2, K_3 \ldots$. Then sequentially compare this record with the records in the sorted K-series list starting from the top to bottom. The position for the first identical record measures the performance of our algorithm. For example, if the position is 1, then our algorithm leads to the global optimum; if the position is 7, then we find the 7th best design among all the 2^{10} designs. In order to assess the overall performance, we randomly choose 100 combinatorially nonisomorphic designs, and record the position for the optimal design obtained using each of the 100 designs, so that there are in total 100 positions recorded. By graphing the histogram for these positions as in the picture below, we see that, 19 out of 100 times, we found the global optimum. In the worst case, we were able to find the 24th best design. The median is at position 6, so we can find designs better than the 7th optimal design for over 50% of the times. These statistics provide strong evidence that our new algorithm also works very well in a broader context.

Figure 5.8: Histogram of Positions



Chapter 6

Concluding Remarks

We have introduced baseline parameterization, and shown how different it is from orthogonal parameterization. We have also reviewed the minimum aberration criteria under both orthogonal parameterization and baseline parameterization. Since Mukerjee & Tang (2011) only provided the tables for baseline minimum aberration designs of 12 and 16 runs, we have focused our efforts on searching for baseline optimal designs of 20 runs. When the number of factors is 13 or less, complete search is feasible. When the number of factors exceeds 13, the complete search procedure requires a huge amount of computations. We have proposed a new search algorithm to reduce the computation time and obtained the results for m$\geq$14. The performance of our search algorithm has been examined and we have found that it performs very well. A simple way to improve the algorithm is to consider neighbourhoods of size larger than two. A more sophisticated algorithm can be developed by allowing variable size neighbourhoods to be considered. This can be done as follows. Suppose that we consider neighbourhoods of size k or less. Each size is given a probability of being chosen, and denote them as $p_1, \ldots, p_k$. Let $p_1 > \ldots > p_k$ and the sum of the probabilities is one. Instead of continuously examining size one neighbours before extending the size to two, we randomly select a size before each iteration using the given $p_1, \ldots, p_k$. This search method will be looked into in the future.

Bibliography

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

[2] Chen, J., Sun, D. X., and Wu, C. F. J. (1993), A Catalogue of Two-Level and Three-Level Fractional Factorial Designs With Small Runs, *International Statistical Review,* 61, 131 – 145.

[3] Fries, A., and Hunter, W. G. (1980), Minimum Aberration 2^{k-p} Designs, *Technometrics,* 22, 601 – 608.

[4] Ke, W., and Tang, B. (2003), Selecting 2^{m-p} Designs Using a Minimum Aberration Criterion When Some Two-factor Interactions Are Important, *Technometrics,* 45, 352 – 360.

[5] Mukerjee, R., and Tang, B. (2011), Optimal Fractions of Two-level Factorials under a Baseline Parameterization, submitted to *Biometrika.*

Appendix A

Baseline MA designs for m≤13 (complete search)

m=5	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	1	1	1	1	1
[2,]	1	1	1	0	0
[3,]	1	1	0	1	0
[4,]	1	1	0	0	1
[5,]	1	1	0	0	0
[6,]	1	0	1	1	0
[7,]	1	0	1	0	1
[8,]	1	0	1	0	0
[9,]	1	0	0	1	1
[10,]	1	0	0	1	1
[11,]	0	1	1	1	0
[12,]	0	1	1	0	1
[13,]	0	1	1	0	1
[14,]	0	1	0	1	1
[15,]	0	1	0	1	0
[16,]	0	0	1	1	1
[17,]	0	0	1	1	0
[18,]	0	0	0	0	1
[19,]	0	0	0	0	0
[20,]	0	0	0	0	0

m=6	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	1	1	1	1	0	1
[2,]	1	1	1	0	1	0
[3,]	1	1	0	1	1	0
[4,]	1	1	0	0	1	1
[5,]	1	1	0	0	0	0
[6,]	1	0	1	1	0	0

[7,]	1	0	1	0	1	0
[8,]	1	0	1	0	0	1
[9,]	1	0	0	1	1	1
[10,]	1	0	0	1	0	1
[11,]	0	1	1	1	0	0
[12,]	0	1	1	0	1	1
[13,]	0	1	1	0	0	1
[14,]	0	1	0	1	1	0
[15,]	0	1	0	1	0	1
[16,]	0	0	1	1	1	1
[17,]	0	0	1	1	1	0
[18,]	0	0	0	0	1	1
[19,]	0	0	0	0	0	0
[20,]	0	0	0	0	0	0

m=7

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	1	1	1	1	0	1	0
[2,]	1	1	1	0	1	0	0
[3,]	1	1	0	1	1	0	1
[4,]	1	1	0	0	1	1	0
[5,]	1	1	0	0	0	0	1
[6,]	1	0	1	1	0	0	0
[7,]	1	0	1	0	1	0	1
[8,]	1	0	1	0	0	1	1
[9,]	1	0	0	1	1	1	0
[10,]	1	0	0	1	0	1	1
[11,]	0	1	1	1	0	0	1
[12,]	0	1	1	0	1	1	1
[13,]	0	1	1	0	0	1	0
[14,]	0	1	0	1	1	0	0
[15,]	0	1	0	1	0	1	1
[16,]	0	0	1	1	1	1	0
[17,]	0	0	1	1	1	0	1
[18,]	0	0	0	0	1	1	1
[19,]	0	0	0	0	0	0	0
[20,]	0	0	0	0	0	0	0

m=8

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
[1,]	1	1	1	1	0	1	0	0
[2,]	1	1	1	0	1	0	0	0
[3,]	1	1	0	1	1	0	1	0
[4,]	1	1	0	0	1	1	0	1
[5,]	1	1	0	0	0	0	1	1

[6,]	1	0	1	1	0	0	0	1
[7,]	1	0	1	0	1	0	1	1
[8,]	1	0	1	0	0	1	1	0
[9,]	1	0	0	1	1	1	0	0
[10,]	1	0	0	1	0	1	1	1
[11,]	0	1	1	1	0	0	1	1
[12,]	0	1	1	0	1	1	1	0
[13,]	0	1	1	0	0	1	0	1
[14,]	0	1	0	1	1	0	0	1
[15,]	0	1	0	1	0	1	1	0
[16,]	0	0	1	1	1	1	0	1
[17,]	0	0	1	1	1	0	1	0
[18,]	0	0	0	0	1	1	1	1
[19,]	0	0	0	0	0	0	0	0
[20,]	0	0	0	0	0	0	0	0

m=9	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]
[1,]	1	1	1	1	0	1	0	0	0
[2,]	1	1	1	0	1	0	0	0	1
[3,]	1	1	0	1	1	0	1	0	0
[4,]	1	1	0	0	1	1	0	1	0
[5,]	1	1	0	0	0	0	1	1	1
[6,]	1	0	1	1	0	0	0	1	1
[7,]	1	0	1	0	1	0	1	1	0
[8,]	1	0	1	0	0	1	1	0	1
[9,]	1	0	0	1	1	1	0	0	1
[10,]	1	0	0	1	0	1	1	1	0
[11,]	0	1	1	1	0	0	1	1	0
[12,]	0	1	1	0	1	1	1	0	0
[13,]	0	1	1	0	0	1	0	1	1
[14,]	0	1	0	1	1	0	0	1	1
[15,]	0	1	0	1	0	1	1	0	1
[16,]	0	0	1	1	1	1	0	1	0
[17,]	0	0	1	1	1	0	1	0	1
[18,]	0	0	0	0	1	1	1	1	1
[19,]	0	0	0	0	0	0	0	0	0
[20,]	0	0	0	0	0	0	0	0	0

m=10	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]
[1,]	1	1	1	1	0	1	0	0	0	0
[2,]	1	1	1	0	1	0	0	0	1	1
[3,]	1	1	0	1	1	0	1	0	0	1
[4,]	1	1	0	0	1	1	0	1	0	1

[5,]	1	1	0	0	0	0	1	1	1	0
[6,]	1	0	1	1	0	0	0	1	1	1
[7,]	1	0	1	0	1	0	1	1	0	0
[8,]	1	0	1	0	0	1	1	0	1	1
[9,]	1	0	0	1	1	1	0	0	1	0
[10,]	1	0	0	1	0	1	1	1	0	0
[11,]	0	1	1	1	0	0	1	1	0	1
[12,]	0	1	1	0	1	1	1	0	0	0
[13,]	0	1	1	0	0	1	0	1	1	0
[14,]	0	1	0	1	1	0	0	1	1	0
[15,]	0	1	0	1	0	1	1	0	1	1
[16,]	0	0	1	1	1	1	0	1	0	1
[17,]	0	0	1	1	1	0	1	0	1	0
[18,]	0	0	0	0	1	1	1	1	1	1
[19,]	0	0	0	0	0	0	0	0	0	1
[20,]	0	0	0	0	0	0	0	0	0	0

m=11	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]
[1,]	0	1	1	1	0	0	0	1	0	1	1
[2,]	0	1	0	0	0	1	1	1	0	0	1
[3,]	0	1	0	0	1	1	0	1	1	1	0
[4,]	0	1	0	1	1	1	0	0	1	0	1
[5,]	0	1	0	1	1	0	1	0	0	1	0
[6,]	0	0	1	0	0	0	0	0	1	0	0
[7,]	0	0	1	0	1	0	1	1	1	1	1
[8,]	0	0	1	1	0	1	1	0	1	1	1
[9,]	0	0	1	1	1	1	1	1	0	0	0
[10,]	0	0	0	0	0	0	0	0	0	0	0
[11,]	1	1	1	0	1	1	1	0	1	0	0
[12,]	1	1	1	0	1	0	0	1	0	0	1
[13,]	1	1	1	1	0	1	0	0	0	1	0
[14,]	1	1	1	1	0	0	1	1	1	0	0
[15,]	1	1	0	0	0	0	1	0	1	1	1
[16,]	1	0	1	0	1	1	0	0	0	1	1
[17,]	1	0	0	0	0	1	1	1	0	1	0
[18,]	1	0	0	1	0	1	0	1	1	0	1
[19,]	1	0	0	1	1	0	0	1	1	1	0
[20,]	1	0	0	1	1	0	1	0	0	0	1

m=12	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	1	0	1	1	1	1	1	0	0	1	0	0
[2,]	1	0	0	0	1	0	0	1	0	0	1	0
[3,]	1	0	0	0	1	1	0	1	1	1	0	1

[4,]	1	0	0	1	0	0	1	0	1	1	1	1
[5,]	1	0	0	1	0	1	1	1	0	0	0	1
[6,]	1	1	1	0	0	0	0	1	1	1	0	0
[7,]	1	1	1	0	0	1	0	0	0	1	1	1
[8,]	1	1	1	1	1	0	0	0	1	0	0	1
[9,]	1	1	1	1	0	0	1	1	0	0	1	0
[10,]	1	1	0	0	1	1	1	0	1	0	1	0
[11,]	0	0	1	0	0	0	1	0	1	0	0	1
[12,]	0	0	1	0	0	1	1	1	1	1	1	0
[13,]	0	0	1	1	1	0	0	1	0	1	1	1
[14,]	0	0	1	1	1	1	0	0	1	0	1	0
[15,]	0	0	0	0	0	0	0	0	0	0	0	0
[16,]	0	1	1	0	1	1	1	1	0	0	0	1
[17,]	0	1	0	0	1	0	1	0	0	1	1	1
[18,]	0	1	0	1	1	0	1	1	1	1	0	0
[19,]	0	1	0	1	0	1	0	1	1	0	1	1
[20,]	0	1	0	1	0	1	0	0	0	1	0	0

m=13	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]	[,13]
[1,]	1	0	0	1	1	1	0	1	0	0	1	0	1
[2,]	1	0	1	0	1	0	0	0	0	1	0	0	1
[3,]	1	0	1	0	1	0	0	1	1	1	1	1	0
[4,]	1	0	1	1	0	1	1	1	0	1	0	1	0
[5,]	1	0	1	1	0	0	1	0	1	0	1	0	1
[6,]	1	1	0	0	0	1	0	1	0	0	0	1	1
[7,]	1	1	0	0	0	0	1	0	1	1	0	1	1
[8,]	1	1	0	1	1	0	1	0	0	0	1	1	0
[9,]	1	1	0	1	0	1	0	0	1	1	1	0	0
[10,]	1	1	1	0	1	1	1	1	1	0	0	0	0
[11,]	0	0	0	0	0	1	1	1	1	1	1	0	1
[12,]	0	0	0	0	0	0	0	0	0	0	0	0	0
[13,]	0	0	0	1	1	1	0	0	1	1	0	1	0
[14,]	0	0	0	1	1	0	1	1	1	0	0	1	1
[15,]	0	0	1	0	0	1	1	0	0	0	1	1	0
[16,]	0	1	0	0	1	0	1	1	0	1	1	0	0
[17,]	0	1	1	0	1	1	0	0	1	0	1	1	1
[18,]	0	1	1	1	1	1	1	0	0	1	0	0	1
[19,]	0	1	1	1	0	0	0	1	0	1	1	1	1
[20,]	0	1	1	1	0	0	0	1	1	0	0	0	0

	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}	K_{12}	K_{13}
m=5	5.3	1.54	0.25	0.05								
6	8.1	3.58	0.66	0.06	0							
7	11.55	6.75	1.75	0.21	0	0						
8	15.68	11.52	4	0.64	0	0	0					
9	20.52	18.36	8.1	1.62	0	0	0	0				
10	26.1	29.88	18.72	6.3	0.9	0	0	0	0			
11	33.65	46	35.66	16.4	4.18	0.44	0	0	0	0		
12	41.52	64.8	58.92	33.12	10.56	1.44	0	0	0	0	0	
13	50.94	90.47	95.93	65.76	28.6	7.15	0.78	0	0	0	0	0

Appendix B

Baseline optimal designs for m≥14 (algorithmic search)

m=14	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	0	1	1	0	0	0	0	1	0	0	1
2	1	1	0	1	1	0	1	0	0	0	0	1	1	0
3	1	1	0	1	1	0	0	0	1	1	1	0	0	0
4	1	1	0	0	0	1	1	1	0	1	0	1	0	0
5	1	1	0	0	0	0	1	1	1	0	1	0	1	1
6	1	0	1	1	0	1	1	0	0	1	0	0	0	1
7	1	0	1	1	0	0	0	1	1	1	0	0	1	0
8	1	0	1	0	1	0	0	1	1	0	0	1	0	1
9	1	0	1	0	0	1	1	0	1	0	1	1	1	0
10	1	0	0	1	1	1	0	1	0	1	1	1	1	1
11	0	1	1	1	0	1	0	1	0	0	1	0	1	0
12	0	1	1	1	0	0	1	0	1	1	1	1	0	1
13	0	1	1	0	1	1	0	0	1	1	0	1	1	0
14	0	1	1	0	1	0	1	1	0	1	0	0	1	1
15	0	1	0	1	0	1	0	1	1	0	0	1	0	1
16	0	0	1	1	1	0	1	1	0	0	1	1	0	0
17	0	0	0	1	1	1	1	0	1	0	0	0	1	1
18	0	0	0	0	1	1	1	1	1	1	1	0	0	0
19	0	0	0	0	0	0	0	0	0	1	1	1	1	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0

m=15	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	0	1	1	0	0	0	0	1	0	0	1	0
2	1	1	0	1	1	0	1	0	0	0	0	1	1	0	1
3	1	1	0	1	1	0	0	0	1	1	1	0	1	0	0
4	1	1	0	0	0	1	1	1	0	1	0	1	0	0	0
5	1	1	0	0	0	0	1	1	1	0	1	0	0	1	1
6	1	0	1	1	0	1	1	0	0	1	0	0	1	1	0
7	1	0	1	1	0	0	0	1	1	1	0	0	0	0	1

8	1	0	1	0	1	0	0	1	1	0	0	1	1	1	0
9	1	0	1	0	0	1	1	0	1	0	1	1	1	0	1
10	1	0	0	1	1	1	0	1	0	1	1	1	0	1	1
11	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1
12	0	1	1	1	0	0	1	0	1	1	1	1	0	1	0
13	0	1	1	0	1	1	0	0	1	1	0	1	0	0	1
14	0	1	1	0	1	0	1	1	0	1	0	0	1	1	1
15	0	1	0	1	0	1	0	1	1	0	0	1	1	1	0
16	0	0	1	1	1	0	1	1	0	0	1	1	0	0	0
17	0	0	0	1	1	1	1	0	1	0	0	0	0	1	1
18	0	0	0	0	1	1	1	1	1	1	1	0	1	0	0
19	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

m=16	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	0	1	1	0	0	0	0	0	0	1	1	1	1
2	1	1	0	1	1	0	1	0	0	0	0	1	1	0	0	0
3	1	1	0	1	1	0	0	0	1	1	1	0	0	1	1	0
4	1	1	0	0	0	1	1	1	0	1	1	0	1	0	1	0
5	1	1	0	0	0	0	1	1	1	0	1	1	0	1	0	1
6	1	0	1	1	0	1	1	0	0	0	1	1	0	0	1	1
7	1	0	1	1	0	0	0	1	1	1	0	1	1	0	1	0
8	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0	1
9	1	0	1	0	0	1	1	0	1	1	0	0	0	1	0	0
10	1	0	0	1	1	1	0	1	0	1	0	1	0	1	0	1
11	0	1	1	1	0	1	0	0	1	0	1	1	1	1	0	0
12	0	1	1	1	0	0	1	1	0	1	0	0	1	1	0	1
13	0	1	1	0	1	1	0	1	0	1	1	1	0	0	0	0
14	0	1	1	0	1	0	1	0	1	1	0	1	0	0	1	1
15	0	1	0	1	0	1	0	1	1	0	0	0	0	0	1	1
16	0	0	1	1	1	0	1	1	0	0	1	0	0	1	1	0
17	0	0	0	1	1	1	1	0	1	1	1	0	1	0	0	1
18	0	0	0	0	1	1	1	1	1	0	0	1	1	1	1	0
19	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

m=17	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	1	1	0	1	1	0	0	0	0	0	0	1	1	0	1	1
2	1	1	0	1	1	0	1	0	0	0	1	1	0	1	1	0	1
3	1	1	0	1	1	0	0	0	1	1	0	1	1	0	0	1	0
4	1	1	0	0	0	1	1	1	0	1	0	1	0	0	1	1	0
5	1	1	0	0	0	0	1	1	1	0	1	0	1	1	0	0	0

6	1	0	1	1	0	1	1	0	0	0	1	0	1	0	1	1	0
7	1	0	1	1	0	0	0	1	1	0	0	1	1	0	1	0	1
8	1	0	1	0	1	0	0	1	1	1	1	0	0	1	1	1	0
9	1	0	1	0	0	1	1	0	1	1	0	1	0	1	0	0	1
10	1	0	0	1	1	1	0	1	0	1	1	0	0	0	0	0	1
11	0	1	1	1	0	1	0	1	0	1	1	1	1	1	0	0	0
12	0	1	1	1	0	0	1	0	1	1	1	0	0	0	0	1	1
13	0	1	1	0	1	1	0	0	1	0	1	1	0	0	1	0	0
14	0	1	1	0	1	0	1	1	0	1	0	0	1	0	1	0	1
15	0	1	0	1	0	1	0	1	1	0	0	0	0	1	1	1	1
16	0	0	1	1	1	0	1	1	0	0	0	1	0	1	0	1	0
17	0	0	0	1	1	1	1	0	1	1	0	0	1	1	1	0	0
18	0	0	0	0	1	1	1	1	1	0	1	1	1	0	0	1	1
19	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9
m=14	61.22	122.04	147.22	119.86	68.6	28	8.12	1.54
15	72.39	160.07	212.62	187.19	112.5	45.75	12.15	1.95
16	84.48	205.28	297.92	282.72	179.2	73.6	17.76	1.92
17	98	260	413	432.9	309.4	149.6	46.75	8.5

	K_{10}	K_{11}	K_{12}	K_{13}	K_{14}	K_{15}	K_{16}	K_{17}
m=14	0.14	0	0	0	0			
15	0.15	0	0	0	0	0		
16	0	0	0	0	0	0	0	
17	0.68	0	0	0	0	0	0	0

m=18	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1
2	1	1	0	1	1	0	1	0	0	0	0	1	1	1	0	0	0	1
3	1	1	0	1	1	0	0	0	1	1	1	0	0	1	1	1	0	0
4	1	1	0	0	0	1	1	1	0	1	1	0	1	0	0	1	0	1
5	1	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0
6	1	0	1	1	0	1	1	0	0	0	1	1	0	1	0	1	1	0
7	1	0	1	1	0	0	0	1	1	1	0	1	0	0	0	1	0	1
8	1	0	1	0	1	0	0	1	1	0	1	0	1	1	0	0	1	1
9	1	0	1	0	0	1	1	0	1	1	0	0	1	1	1	0	0	0
10	1	0	0	1	1	1	0	1	0	1	0	1	1	0	1	0	1	0
11	0	1	1	1	0	1	0	0	1	0	1	1	1	0	1	0	0	1

12	0	1	1	1	0	0	1	1	0	1	0	0	0	1	1	0	1	1	
13	0	1	1	0	1	1	0	1	0	1	1	1	0	1	0	0	0	0	
14	0	1	1	0	1	0	1	0	1	1	0	1	1	0	0	1	1	0	
15	0	1	0	1	0	1	0	1	1	0	0	0	1	1	0	1	1	0	
16	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	1	0	0	
17	0	0	0	1	1	1	1	0	1	1	1	0	0	0	0	0	1	1	
18	0	0	0	0	1	1	1	1	1	0	0	1	0	1	1	1	0	1	
19	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

m=19	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1
2	1	1	0	1	1	0	1	0	0	0	0	1	1	1	0	0	0	1	1
3	1	1	0	1	1	0	0	0	1	1	1	0	0	1	1	1	0	0	0
4	1	1	0	0	0	1	1	1	0	1	1	0	1	0	0	1	0	1	0
5	1	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1
6	1	0	1	1	0	1	1	0	0	0	1	1	0	1	0	1	1	0	0
7	1	0	1	1	0	0	0	1	1	1	0	1	0	0	0	1	0	1	1
8	1	0	1	0	1	0	0	1	1	0	1	0	1	1	0	0	1	1	0
9	1	0	1	0	0	1	1	0	1	1	0	0	1	1	1	0	0	0	1
10	1	0	0	1	1	1	0	1	0	1	0	1	1	0	1	0	1	0	0
11	0	1	1	1	0	1	0	0	1	0	1	1	1	0	1	0	0	1	0
12	0	1	1	1	0	0	1	1	0	1	0	0	0	1	1	0	1	1	0
13	0	1	1	0	1	1	0	1	0	1	1	1	0	1	0	0	0	0	1
14	0	1	1	0	1	0	1	0	1	1	0	1	1	0	0	1	1	0	0
15	0	1	0	1	0	1	0	1	1	0	0	0	1	1	0	1	1	0	1
16	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	1	0	0	1
17	0	0	0	1	1	1	1	0	1	1	1	0	0	0	0	0	1	1	1
18	0	0	0	0	1	1	1	1	1	0	0	1	0	1	1	1	0	1	0
19	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0