2003

Construction of Efficient Fractional Factorial Mixed-Level Designs

Yong Guo

THE FLORIDA STATE UNIVERSITY
COLLEGE OF ENGINEERING

CONSTRUCTION OF EFFICIENT FRACTIONAL FACTORIAL MIXED-LEVEL DESIGNS

By
YONG GUO

A Thesis submitted to the
Department of Industrial Engineering
in partial fulfillment of the
requirements for the degree of
Master of Science

Degree Awarded
Fall Semester, 2003

The members of the Committee approve the thesis of Yong Guo defended on November 21, 2003.

James R. Simpson
Professor Directing Thesis

Samuel A. Awoniyi
Committee Member

Joseph J. Pignatiello, Jr.
Committee Member

Approved:

Ben Wang, Chair, Department of Industrial Engineering

Ching-Jen Chen, Dean, FAMU-FSU College of Engineering

The Office of Graduate Studies has verified and approved the above named committee members.

AKNOWLEDGEMENTS

I wish to thank my mother for giving birth to me and supporting all of my goals. Dr. James Simpson, not only for his skill in research, but also because he would not accept anything less than my best when conducting research. Dr. Samuel Awoniyi taught me critical thinking skills that helped me solve industrial engineering problems. Dr. Joseph Pigniatello showed me the beauty of mathematics and how I can apply it to different situations. I am also grateful for the support I have received from the Quality Lab especially Lisa Hughes, Francisco Ortiz, Marcus Perry, and Rupert Giroux. My friend Todd Kramer helped me edit this thesis and is becoming a great Chinese scholar. Trisha Shaw from the English Department at Florida State University for editing several chapters of my thesis. Lastly, to all of my friends in the Industrial Engineering Department who guided me along this journey.

TABLE OF CONTENTS

List of Tables

List of Figures

ABSTRACT

Mixed-level factorial designs are experimental designs whose factors have different numbers of levels. These designs are very useful in experiments involving both qualitative and quantitative factors. One design approach is to run all possible combinations of the factor levels. However, as the number of factors or factor levels increases, the number of experiments increases dramatically. As a result, research has focused on developing orthogonal or near-orthogonal fractional factorial designs. The property of design balance, that the same number of runs is performed for each factor level, has been maintained in currently proposed designs. In some cases, maintaining balance requires too many experimental runs. The objective of this thesis is to develop fractional mixed-level factorial designs with economical run size that have desirable properties associated with near-balance and near-orthogonality. Two criteria are developed to assess the degree of near-balance for comparing and constructing designs. A modified J_2-optimality criterion is used for comparing design near-orthogonality. These criteria are combined to assess different design alternatives. A genetic algorithm is then used to build designs with the most desirable combination of near-balance and near-orthogonality.

CHAPTER 1

INTRODUCTION

1-1 Statement of the Problem

Traditional two-level factorial designs are widely used in industrial research and development. However, in some situations, factors with more than two levels are more desirable, especially when those factors are qualitative factors. In fact, qualitative factors are the main motivation of research on mixed-level designs. As a result, designs with mixed-level factors have been used more often in designed experiments in modern industries, especially when only limited resources are allowed.

Two of the more desirable properties of factorial designs are balance and orthogonality. Balance requires that each level of a factor is run the same number of times in an experiment. Orthogonal designs are column pairwise linearly independent, useful for assessing factor significance. Construction of mixed-level designs with efficient run sizes and desirable balance and orthogonality properties is the primary focus of research in this area. Several algorithms have been developed by different authors to generate balanced orthogonal mixed-level designs. Additionally, near-orthogonal designs have been generated as the alternatives to strictly orthogonal designs, when orthogonal designs are either difficult or impossible to produce.

Wang & Wu (1991) proposed an approach for constructing orthogonal designs based upon difference matrices. Their construction method essentially applies the generalized Kronecker sum and uses the technique of adding columns. DeCock & Stufken (2000) proposed an algorithm for constructing orthogonal mixed-level designs through searching some existing two-level orthogonal designs. Xu (2002) proposed an algorithm to construct orthogonal and near-orthogonal designs based on the concept of J_2-optimality. The J_2 criterion is equivalent to the other orthogonality criteria, such as the

(*M, S*) criterion (Eccleston and Hedayat, 1974), the A_2-optimality (Xu, 2002), and the *ave*(s^2) criterion (Xu, 2002). However, the J_2 criterion is more appropriate for situations where factor levels are not denoted by contrasts, because J_2-optimality does not require computing the value of $\left|X^T X\right|$, where X is the contrast matrix of the design. Xu (2002) also shows that a design is J_2-optimal and orthogonal if J_2 equals its lower bound.

Unfortunately, as the number of factors or factor levels increases, the number of runs increases dramatically. Maintaining the balance property requires too many runs in some situations. For example, consider a design with four factors, one with 2 levels, one with 3 levels, one with 4 levels and the last with 5 levels. To generate a balanced design, at least 60 runs are needed. Suppose an engineer only has resources for 30 tests and the test objective is screening. The intent of this thesis is to develop a mechanism for creating mixed-level designs with desirable properties capable of meeting resource requirements. In this thesis, efficient near-balanced, near-orthogonal mixed-level designs will be constructed by using the genetic algorithm.

1-2 Research Objective and Methodology

This thesis focuses on the construction of efficient mixed-level factorial designs with desirable balance and orthogonality properties. In order to evaluate these balance and orthogonality properties, some criteria must be proposed and used. These criteria can then be combined into an objective function to be optimized, and a genetic algorithm approach will be developed to build these desirable, efficient designs. Currently, no existing criterion is available to evaluate the balance property of the design matrix. Therefore, a new optimality criterion, called the balance coefficient, will be defined and formulated. Some investigations on the balance coefficient will be performed and analyzed. The J_2-optimality is used to measure the orthogonality property of the design. Essentially, J_2-optimality is consistent with the orthogonality property. Note that the lower bound for the J_2-optimality, defined by Xu (2002), can only be used in the balanced design situations. This lower bound is not applicable to unbalanced design situations. As such, this research proposes a modified J_2 criterion that can be used to measure the degree of orthogonality of unbalanced design matrices.

In terms of the balance coefficient and J_2-optimality, this thesis will employ genetic algorithms to generate efficient mixed-level designs. The basic purpose of the genetic algorithm is to choose a subset of rows from a matrix of all possible combinations. For example, suppose a design involves three factors: factor A with two levels, factor B with three levels and factor C with four levels. The full factorial design, with 24 runs, is shown in Figure 1.1a. Desiring to reduce the experimental runs, eight runs are chosen and combined into a reduced sized design, which is shown in Figure 1.1b. The function of the genetic algorithm is to select the eight runs that optimize the objective function and combine these runs into a suitable mixed-level design.

1-3 Thesis Research Scope

This thesis will focus on experimental design construction issues only. Using the concept of the least common multiple and degrees of freedom necessary for low order models, the number of runs for efficient designs will be suggested. This thesis will propose a method to construct efficient mixed-level factorial designs by applying genetic algorithms. A MATLAB code will be developed to achieve this goal. A new balance criterion using alternative definitions and associated formulations will be developed. A J_2-optimality criterion for unbalanced designs will be developed and combined with the balance coefficient to serve as the objective function for the genetic algorithm. This thesis will also investigate the relationship between these two criteria and the design sizes. The analysis of the mixed-level designs will not be discussed in this thesis.

Run	Factor A	Factor B	Factor C
1	1	1	1
2	1	2	1
3	1	3	1
4	1	1	2
5	1	2	2
6	1	3	2
7	1	1	3
8	1	2	3
9	1	3	3
10	1	1	4
11	1	2	4
12	1	3	4
13	2	1	1
14	2	2	1
15	2	3	1
16	2	1	2
17	2	2	2
18	2	3	2
19	2	1	3
20	2	2	3
21	2	3	3
22	2	1	4
23	2	2	4
24	2	3	4

a) A full factorial design

Run	Factor A	Factor B	Factor C
1	1	1	1
5	1	2	2
9	1	3	3
10	1	1	4
14	2	2	1
18	2	3	2
19	2	1	3
23	2	2	4

b) An efficient design

Figure 1.1 The idea of the efficient mixed-level design construction method.

CHAPTER 2

LITERATURE REVIEW

Mixed-level designs have drawn a considerable amount of attention in terms of construction. This chapter is intended to provide an overview of experimental design concepts and a through discussion of mixed-level research performed to date. The basic concepts of the design of experiments will be introduced in Section 2-1 so as to provide the background knowledge of this research. This thesis focuses on the construction of mixed-level designs, and two existing orthogonal mixed-level design construction methods will be analyzed and summarized in Sections 2-2 and 2-3. One method is based upon difference matrices and another method is based upon J_2-optimality. In Section 2-4, the principle of the genetic algorithms will be briefly introduced. In order to illustrate how genetic algorithms are used in the design of experiments, a construction method of supersaturated designs by genetic algorithms will be introduced in Section 2-5.

2-1 General Concepts of Design of Experiments

An experiment is a test or series of tests conducted under controlled conditions made to demonstrate a known truth, examine the validity of a hypothesis, or determine the efficacy of something previously untried. In an experiment, one or more input process variables are changed deliberately in order to observe the effect the changes have on one or more response variables. Experiments are performed a number of times in order to evaluate the output response variables under the different input process variable conditions. The design of experiments is an efficient method for planning experiments so that the data obtained can be analyzed to yield valid and objective conclusions. The method for conducting designed experiments begins with determining the objectives of an experiment and selecting the process factors for the study. A designed experiment

requires establishing a detailed experimental plan in advance of conducting the experiment, which results in a streamlined approach in the data collection stage. Appropriately choosing experimental designs maximizes the amount of information that can be obtained for a given amount of experimental effort.

2-1-1 Process Models for Designed Experiment

Experimental designs are used to investigate industrial systems or processes. A typical process model is given in Figure 2.1. Purposeful changes are made to the controllable input factors of a process so as to observe and identify the reasons for changes that may be observed in the output responses. The noise factors are considered as random factors that cannot be controlled. Experimental data are used to derive a statistical empirical model linking the outputs and inputs. These empirical models generally contain first and second-order terms. For more information regarding the statistical empirical model, see Montgomery, 2001.

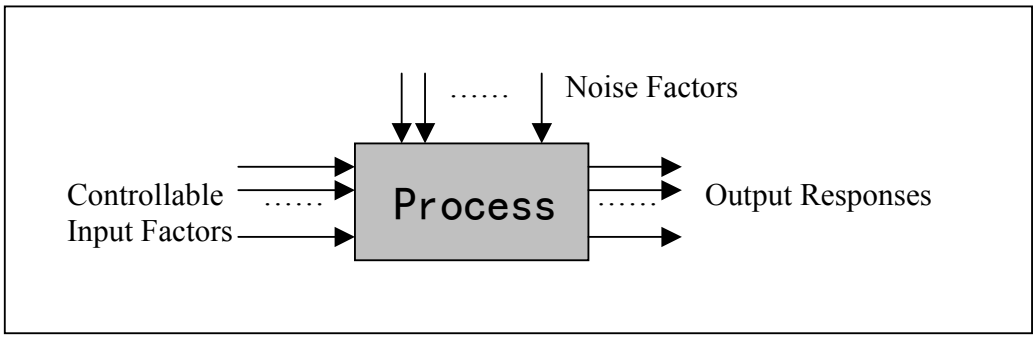

Figure 2.1 The components of a process.

2-1-2 Factorial Designs

Many experiments involve the study of the effects of two or more factors. Full factorial designs are test matrices that contain all possible combinations of the levels of

the factors. For example, if factor A has a levels and factor B has b levels, then the two-factor full factorial design contains ab combinations. Table 2.1 shows another example, a full factorial design with three factors: one with two levels, one with three and one with five. The shorthand notation for this design is ($2^1 3^1 5^1$), which displays the factor levels as the base numbers and the number of factors with that many levels as the exponent.

Table 2.1 Full Factorial Design - $2^1 3^1 5^1$

Run	Factor A	Factor B	Factor C
1	1	1	1
2	2	1	1
3	1	2	1
4	2	2	1
5	1	3	1
6	2	3	1
7	1	1	2
8	2	1	2
9	1	2	2
10	2	2	2
11	1	3	2
12	2	3	2
13	1	1	3
14	2	1	3
15	1	2	3
16	2	2	3
17	1	3	3
18	2	3	3
19	1	1	4
20	2	1	4
21	1	2	4
22	2	2	4
23	1	3	4
24	2	3	4
25	1	1	5
26	2	1	5
27	1	2	5
28	2	2	5
29	1	3	5
30	2	3	5

One purpose of factorial designs is to study the effects of these factors on the response. The main effect of a factor is defined to be the change in response produced by a change in the level of the factor. The term main effect is used because it refers to the primary factors of interest in the experiment. A main effect reflects the individual effect of each factor.

One-factor-at-a-time testing is an extensively used experimentation strategy. This method consists of selecting a starting point setting for each factor, then successively varying the settings of each factor over its range, with the other factors held constant (Montgomery, 2001). Compared with one-factor-at-a-time experiments, factorial designs are more efficient. Factorial designs allow the effects of a factor to be estimated at several levels of the other factors because the difference in response between the levels of one factor may not be the same at all levels of the other factor. Therefore, factorial designs are necessary when interactions may be present.

A specific case of general factorial designs is the 2^k factorial design. That is, these designs have k factors, each at only two levels. These levels may be either quantitative or qualitative. Normally "+" is used to represent the high level and "–" is used to represent the low level in the 2-level factorial designs. A complete replicate of such a design requires 2^k observations and is called a 2^k factorial design. Table 2.2 gives an example for k=2 in three replicates.

Table 2.2 Treatments Combinations for 2 Factors with 2 Levels Each

Factor		**Treatment**	**Replicate**		
A	B	**Combination**	I	II	III
–	–	A low, B low	28	25	27
+	–	A high, B low	36	32	32
–	+	A low, B high	18	19	23
+	+	A high, B high	31	30	29

The interaction effect AB is defined as the average change in response between the effect of A at high level of B and the effect of A at the low level of B. The methods used for generating 2^k factorial designs are straightforward. Each column represents a factor. The levels for the first column follow "– + – + …– +". The levels for the second column follow the pattern of "– – + +…– – + +". For the n^{th} column, the pattern will be "–…– +…+" and the number for minus signs and plus signs is n for each.

2-1-3 Fractional Factorial Designs

If the experimenter can reasonably assume that certain high-order interactions are negligible, information on the main effects and low-order interactions may be obtained by running only a fraction of the complete factorial experiment. The concept of fractions used in factorial designs is creative. The use of fractional factorial designs is especially popular in screening experiments. Screening experiments are experiments in which many factors are considered, and the objective is to identify those factors with large effects. Screening experiments are usually performed in the earlier stages of a project when it is likely that many of the factors initially considered have little or no effect on the response. The factors that are identified as significant are then investigated more thoroughly in subsequent experiments.

Since the concept of the fractional factorial designs is that only a fraction of the complete factorial experiment is used, it raises the questions of how to choose appropriate fractions and how to perform experiment augmentation. Experiment augmentation denotes expansion of factorial designs into larger size designs. The approach used to generate fractions is based on design generators. Design generators are used to generate a particular fraction. Consider a situation in which three factors, each at two levels are of interest, but the experimenters cannot afford to run all 2^3=8 treatment combinations. They can, however, afford four runs. This restriction suggests the use of a one-half fraction of a 2^3 design. Because the design contains 2^{3-1}=4 treatment combinations, a one-half fraction of the 2^3 design is often called a 2^{3-1} design.

Table 2.3 Plus and Minus Signs for the 2^3 Fractional Designs

Factorial Effect							
I	A	B	C	AB	AC	BC	ABC
+	+	–	–	–	–	+	+
+	–	+	–	–	+	–	+
+	–	–	+	+	–	–	+
+	+	+	+	+	+	+	+
+	+	+	–	–	–	–	–
+	+	–	+	+	+	–	–
+	–	+	+	+	–	+	–
+	–	–	–	–	+	+	–

From Table 2.3 it can be seen that the 2^{3-1} design in the first four runs is formed by selecting only those treatment combinations with a "+" sign in the ABC column. In this situation, ABC is called the generator of this particular fraction. Furthermore, the identity column I is also always plus, so I=ABC is called the defining relation for the design including only the first four runs. Generally, the defining relation for a fractional factorial will always be the set of all columns that are equal to the identity column I. ABC in this example is a word in the defining relation. More highly fractionated designs have more than one word in the defining relation.

It is important to consider the resolution of fractional factorial design when designing a fraction. In general, the resolution of a 2-level fractional factorial design is equal to the smallest word (in terms of number of letters) in the defining relation. Usually, fractional factorial designs with highest resolution are preferred. The higher the resolution, the less restrictive the assumptions that are required regarding which interactions are negligible to obtain a unique interpretation of the data. Resolution III designs are designs in which no main effects are aliased with any other main effect, but main effects are aliased with 2-factor interactions and 2-factor interactions may be aliased with each other. Resolution IV designs are designs in which no main effect is aliased with any other main effect or with any 2-factor interaction, but 2-factor interactions are aliased with each other. Resolution V designs are designs in which no main effect or 2-factor

interaction is aliased with any other main effect or 2-factor interaction, but 2-factor interactions are aliased with 3-factor interactions (Montgomery, 2001). Construction of 2^{n-k} fractional factorial designs can be accomplished relatively easily using a basic design and a list of design generations. For additional discussion of construction of fractional factorial designs, see Montgomery (2001).

2-1-4 Mixed-Level Designs

Mixed-level designs are factorial designs whose factors have varying levels. Montgomery (2001) mentions that the 2^k factorial designs are the cornerstone of industrial experimentation. However, in some situations it is necessary to include factors that have more than two levels. It may be undesirable to reduce the factor levels to two, particularly with qualitative factors, because choosing only two levels can require not considering potentially influential factor settings. Usually, mixed-level designs are used when both quantitative and qualitative factors are involved in the experiment. The example given in Table 2.1 shows a simple mixed-level design. Two of the more desirable properties of factorial designs are balance and orthogonality. In Sections 2-1-4-1 and 2-1-4-2, these two properties will be briefly introduced. Some notation for mixed-level designs will then be discussed in Section 2-1-4-3.

2-1-4-1 Orthogonality property. An orthogonal design of strength 2 is an experimental design in which all pairwise columns are orthogonal. This thesis will not consider the designs whose orthogonal strength are larger than 2, because only orthogonal designs with strength 2 are currently used in experimental designs. Two columns are orthogonal if each of their level combinations appears equally often. Table 2.4 shows an orthogonal pair.

Table 2.4 An Orthogonal Pair

1	1
2	2
1	3
2	1
1	2
2	3

Normally the degree of orthogonality is measured by orthogonal optimality criteria. Some optimality criteria are (*M, S*) criterion (Eccleston and Hedayat, 1974), A_2-optimality (Xu, 2002), $ave(s^2)$ criterion (Xu, 2002) and J_2-optimality (Xu, 2002).

2-1-4-2 Balance property. In addition to orthogonality, balance is another property commonly considered in experimental designs. By definition, a balanced design is a design in which all the columns are balanced. A column is balanced if each factor level contains the same number of runs. An example is shown in Table 2.5.

Table 2.5 A Unbalanced Design

A	B	C
1	1	1
2	2	2
1	3	3
2	4	1
1	1	2
2	2	3
1	3	1
2	4	2

In this example, factor C has three levels and these three levels do not appear with the same frequency. Therefore, this design is not a balanced design. Regarding the balance property of design matrices, there is no criterion currently available to measure this property. So it is necessary to develop a proper criterion to evaluate and compare the design matrices in terms of the balance property. The balance criterion is discussed in Chapter 3.

2-1-4-3 Notation for mixed-level designs. In order to measure the balance and orthogonality properties, factor levels have to be coded. There are several forms of notation with regard to factor levels. At least two notational forms are

commonly used. In 2-level factorial designs, the two levels are denoted by using plus and minus signs. However, in 3 or higher-level designs, the "1, 2, 3…" representation is used. In this case, each number is just a label, so it does not matter which number represents that level. In mixed-level design situations, factor levels are coded by natural numbers. But for the 2-level design particularly, the plus-minus sign notation other than "1 2" should be used in order to avoid confusion in the sign interpretation for the interaction effect estimate. Regarding orthogonal mixed-level designs, there are different notations commonly used. One commonly used notation for orthogonal design is $OA(n, l_1^{k_1} l_2^{k_2} \cdots l_r^{k_r})$, which denotes an orthogonal design of size n and strength 2 having k_i columns with l_j levels. In this thesis the notation $EA(n, l_1^{k_1} l_2^{k_2} \cdots l_r^{k_r})$ is used to denote a near-balanced, near-orthogonal, efficient mixed-level design.

2-1-4-4 Construction of mixed-level designs. The concept of orthogonal designs dates back to Rao (1947). Construction methods include combinatorial, geometrical, algebraic, coding theoretic, and algorithmic approaches. State-of-art construction of orthogonal designs has been described by Hedayat, Sloane, and Stufken (1999).

Nguyen (1996) proposed an interchange algorithm for constructing supersaturated designs. His program can be used to construct two-level orthogonal designs and the largest one constructed and published is $OA(20, 2^{19})$.

DeCock and Stufken (2000) proposed an algorithm for constructing mixed-level orthogonal designs through searching some existing two-level orthogonal designs. Their purpose is to construct mixed-level orthogonal designs with as many two-level columns as possible, and their algorithm succeeded in constructing several new large mixed-level orthogonal designs. However, their algorithm fails to produce some orthogonal designs.

Nguyen (1996) proposed an algorithm for constructing near-orthogonal designs by adding two-columns to existing orthogonal designs. Wang and Wu (1992) systematically studied near-orthogonal designs and proposed some general combinatorial construction methods, such as the construction method based on the difference matrix.

Ma et al. (2000) proposed two algorithms for constructing near-orthogonal designs by minimizing some combinatorial criteria via the thresholding accepting technique.

Xu (2002) proposed an algorithm for constructing orthogonal and near-orthogonal designs based upon the J_2-optimality. His algorithm is easier to achieve by program. The method works with columns of more than two levels and appropriate weights are also considered for different columns.

2-2 Construction of Orthogonal Designs Based on Difference Matrices

Wang and Wu (1991) proposed a method to construct orthogonal mixed-level designs based upon the existing ideas or their extensions, and synthesized all these ideas in a single approach. The construction method employs two mathematical concepts: difference matrices and Kronecker sums.

2-2-1 General Ideas

Let G be an additive group of p elements denoted by $\{0, 1, \ldots, p\text{-}1\}$, where the sum of any two elements x and y is $x+y \pmod p$ for prime p, and is defined as in a Galois field for prime power p. A $\lambda p \times r$ matrix with elements from G denoted by $D_{\lambda p,r;p}$ is called a difference matrix if, among the differences modulus p of the corresponding elements of any of its two columns, each element of G occurs exactly λ times. For $p=2$, $D_{n,n;2}$ is a Hadamard matrix of order n. For two matrices $A=[a_{ij}]$ of order $n \times r$ and B of order $m \times s$, both with entries from G, their Kronecker sum is defined to be

$$A * B = \left\lfloor B^{a_{ij}} \right\rfloor_{1 \le i \le n, 1 \le j \le r},$$

where

$$B^k = (B + kJ) \bmod p$$

is obtained from adding $k \pmod p$ to the elements of B, and J is an $m \times s$ matrix of 1's.

The generalized Kronecker sum of A and B is defined as

$$A \otimes B = [A_1 * B_1, \cdots, A_n * B_n].$$

If A is an orthogonal design with the partition, then

$$A = [OA(N, p_1^{s_1}), \cdots, OA(N, p_n^{s_n})]$$

and

$$B = [D_{M,k_1;p_1}, \cdots, D_{M,K_n;p_n}],$$

where $D_{M,k_i;p_i}$ is a difference matrix, and N and M are both multiples of the p_i's. Then their generalized Kronecker sum,

$$A \otimes B = [OA(N, p_1^{s_1}) * D_{M,k_1;p_1}, \cdots, OA(N, p_n^{s_n}) * D_{M,k_n;p_n}]$$

is an orthogonal design $OA(NM, p_1^{k_1 s_1} \cdots p_r^{k_r s_n})$.

2-2-2 General Construction Procedures

Based on the principle of this construction method, the general procedures to generate mixed-level designs can be summarized as follows. First construct the orthogonal design $A \otimes B$. Then let

$$L = 0_N * OA(N, q_1^{r_1} \cdots q_m^{r_m})$$

be a matrix consisting of N copies of an orthogonal design, with $OA(N, q_1^{r_1} \cdots q_m^{r_m})$ as its rows, and 0_N is the $N \times 1$ vector of zeros. By adding the columns of L to $A \otimes B$, the resulting matrix $[A \otimes B, L]$ is an orthogonal design $OA(NM, p_1^{k_1 s_1} \cdots p_r^{k_r s_n} q_1^{r_1} \cdots q_m^{r_m})$.

The proof of the orthogonality between $A \otimes B$ is given in Wang and Wu (1991). The variety number types of orthogonal designs generated by this method are limited. Consider a difference matrix $D_{6,6;3}$, which is

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 2 & 1 & 1 & 0 & 2 \\ 0 & 0 & 2 & 1 & 2 & 1 \\ 0 & 2 & 0 & 2 & 1 & 1 \\ 0 & 1 & 1 & 2 & 2 & 0 \end{bmatrix}.$$

An orthogonal design can be constructed, namely

$$OA(3,3^1)=\begin{bmatrix}0\\1\\2\end{bmatrix}.$$

By the construction procedures just mentioned, it can be shown that

$$OA(3,3^1)*D_{6,6;3}=\begin{bmatrix}D_{6,6;3}\\ D_{6,6;3}+1 \pmod 3\\ D_{6,6;3}+2 \pmod 3\end{bmatrix}.$$

This resulting design is an $OA(18,3^6)$, which is displayed below,

$$\begin{bmatrix}
Run & A & B & C & D & E & F\\
1 & 0 & 0 & 0 & 0 & 0 & 0\\
2 & 1 & 1 & 1 & 1 & 1 & 1\\
3 & 2 & 2 & 2 & 2 & 2 & 2\\
4 & 0 & 0 & 1 & 1 & 2 & 2\\
5 & 1 & 1 & 2 & 2 & 0 & 0\\
6 & 2 & 2 & 0 & 0 & 1 & 1\\
7 & 0 & 1 & 0 & 2 & 1 & 2\\
8 & 1 & 2 & 1 & 0 & 2 & 0\\
9 & 2 & 0 & 2 & 1 & 0 & 1\\
10 & 0 & 2 & 2 & 1 & 1 & 0\\
11 & 1 & 0 & 0 & 2 & 2 & 1\\
12 & 2 & 1 & 1 & 0 & 0 & 2\\
13 & 0 & 1 & 2 & 0 & 2 & 1\\
14 & 1 & 2 & 0 & 1 & 0 & 2\\
15 & 2 & 0 & 1 & 2 & 1 & 0\\
16 & 0 & 2 & 1 & 2 & 0 & 1\\
17 & 1 & 0 & 2 & 0 & 1 & 2\\
18 & 2 & 1 & 0 & 1 & 2 & 0
\end{bmatrix}.$$

The constructed design will be used to build orthogonal mixed-level designs. Let L_6 denote a six-run orthogonal design, and L_6 is chosen to be

$$L_6=\begin{bmatrix}0 & 1 & 2 & 3 & 4 & 5\end{bmatrix}^T,$$

then

$$\left[OA(3,3^1)*D_{6,6;3},0_3*L_6\right]$$

is an orthogonal design $OA(18,6^1 3^6)$, which is

$$\begin{bmatrix} Run & G & A & B & C & D & E & F \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 3 & 0 & 2 & 2 & 2 & 2 & 2 & 2 \\ 4 & 1 & 0 & 0 & 1 & 1 & 2 & 2 \\ 5 & 1 & 1 & 1 & 2 & 2 & 0 & 0 \\ 6 & 1 & 2 & 2 & 0 & 0 & 1 & 1 \\ 7 & 2 & 0 & 1 & 0 & 2 & 1 & 2 \\ 8 & 2 & 1 & 2 & 1 & 0 & 2 & 0 \\ 9 & 2 & 2 & 0 & 2 & 1 & 0 & 1 \\ 10 & 3 & 0 & 2 & 2 & 1 & 1 & 0 \\ 11 & 3 & 1 & 0 & 0 & 2 & 2 & 1 \\ 12 & 3 & 2 & 1 & 1 & 0 & 0 & 2 \\ 13 & 4 & 0 & 1 & 2 & 0 & 2 & 1 \\ 14 & 4 & 1 & 2 & 0 & 1 & 0 & 2 \\ 15 & 4 & 2 & 0 & 1 & 2 & 1 & 0 \\ 16 & 5 & 0 & 2 & 1 & 2 & 0 & 1 \\ 17 & 5 & 1 & 0 & 2 & 0 & 1 & 2 \\ 18 & 5 & 2 & 1 & 0 & 1 & 2 & 0 \end{bmatrix}$$

2-2-3 Generate Orthogonal Mixed-level Designs by $D_{m,m;2}$

In order to see the limitation of this mixed-level design construction method, a series of orthogonal mixed-level designs are being constructed by using $D_{m,m;2}$. In this simple example, $OA(2,2^1)=(0,1)^T$ and $D_{m,m;2}$ are used to construct orthogonal mixed-level designs. Then by the construction procedures,

$$\mathrm{O}A(2,2^1) * D_{m,m;2} = \begin{bmatrix} D_{m,m;2} \\ D_{m,m;2}+1 \end{bmatrix} = OA(2m,2^m).$$

If $L = 0_2 * L_m$ is selected, the generated designs are expressed as

$$[OA(2,2^1) * D_{m,m;2}, 0_2 * L_m].$$

By choosing different *m*, the possible designs generated by the $D_{m,m;2}$ and L_m are shown in Table 2.6.

Table 2.6 Possible Designs Generated by the $D_{m,m;2}$ and L_m

m	Possible L_m	Final designs
3,5, 7,11	$OA(3, 3^1)$; $OA(5, 5^1)$; $OA(7, 7^1)$; $OA(11, (11)^1)$	$OA(6, 3^1 2^3)$; $OA(10, 5^1 2^5)$; $OA(14, 7^1 2^7)$; $OA(22, (11)^1 2^{11})$
4	$OA(4, 4^1)$ or $OA(4, 2^2)$	$OA(8, 2^4 4^1)$ or $OA(8, 2^6)$
6	$OA(6, 6^1)$ or $OA(6, 2^1 3^1)$	$OA(12, 2^6 6^1)$ or $OA(12, 2^7 3^1)$
8	$OA(8, 8^1)$ or $OA(8, 2^3)$ or $OA(8, 2^2 4^1)$	$OA(16, 2^8 8^1)$or $OA(16, 2^{11})$or $OA(16, 2^{10} 4^1)$
9	$OA(9, 9^1)$ or $OA(9, 3^2)$	$OA(18, 2^9 9^1)$or $OA(18, 2^9 3^2)$
10	$OA(10, (10)^1)$ or $OA(10, 2^1 5^1)$	$OA(20, 2^{10}(10)^1)$ or $OA(20, 2^{11} 5^1)$
12	$OA(12, (12)^1)$ or $OA(12, 2^1 6^1)$ or $OA(12, 2^2 3^1)$ or $OA(12, 4^1 3^1)$	$OA(24, 2^{12}(12)^1)$ or $OA(24, 2^{13} 6^1)$ or OA$(24, 2^{14} 3^1)$ or $OA(24, 2^{12} 3^1 4^1)$

There are several designs generated by this method. All these designs are orthogonal designs and can be used in different situations. However, this construction method reveals its limitation. That is, the number of possible designs that can be generated by this method is limited by the availability of difference matrices.

2-3 Construction of Orthogonal Designs Based upon J_2–Optimality

J_2-optimality is used to measure the degree of orthogonality of mixed-level designs. Based upon this criterion, Xu (2002) proposed a method to generate balanced orthogonal designs. His method can efficiently construct various designs with good balance and orthogonality properties.

2-3-1 Definition of J_2-Optimality

Before discussing this mixed-level design construction method, the definition of J_2-optimality is first introduced. Consider a $n \times m$ matrix $D = [x_{ik}]$, with weights $w_k > 0$, and s_k levels for each factor. The weights indicate the relative importance of each factor and if $w_k = 1$ are chosen, $\delta_{i,j}(d)$ is the number of coincidences between the i^{th} and j^{th} rows. For $1 \le i, j \le n$,

$$\delta_{i,j}(d) = \sum_{k=1}^{m} w_k \delta(x_{ik}, x_{jk}),$$

where $\delta(x, y)$=1 if $x = y$ and 0 otherwise.

Then the J_2-optimality of this design matrix d is defined as

$$J_2(d) = \sum_{1 \le i < j \le n} \left[\delta_{i,j}(d)\right]^2 .$$

Table 2.7 gives an example of an orthogonal design, OA (12, $3^1 2^4$), with J_2 value 330. The corresponding coincidence matrix $\delta_{i,j}(d)$ of this matrix is given in Table 2.8.

Table 2.7 *OA* (12,$3^1 2^4$)

Run	1	2	3	4	5
1	0	0	1	0	1
2	0	1	0	0	1
3	0	0	1	1	0
4	0	1	0	1	0
5	1	0	0	0	0
6	1	1	1	0	0
7	1	0	1	1	1
8	1	1	0	1	1
9	2	0	0	1	0
10	2	1	1	0	0
11	2	0	0	0	1
12	2	1	1	1	1

Table 2.8 Coincidence Matrix of *OA* (12,$3^1 2^4$)

5	3	3	1	2	2	3	1	1	2	3	2
3	5	1	3	2	2	1	3	1	2	3	2
3	1	5	3	2	2	3	1	3	2	1	2
1	3	3	5	2	2	1	3	3	2	1	2
2	2	2	2	5	3	2	2	3	2	3	0
2	2	2	2	3	5	2	2	1	4	1	2
3	1	3	1	2	2	5	3	2	1	2	3
1	3	1	3	2	2	3	5	2	1	2	3
1	1	3	3	3	1	2	2	5	2	3	2
2	2	2	2	2	4	1	1	2	5	2	3
3	3	1	1	3	1	2	2	3	2	5	2
2	2	2	2	0	2	3	3	2	3	2	5

This coincidence matrix is symmetric. The elements on the diagonal will always equal k. In fact, J_2 is the sum of squares of the elements above the diagonal. It is easy to see that J_2=330 for this example.

2-3-2 Lower bound of J_2-optimality for balanced designs

There exists a lower bound for the J_2-optimality. When the J_2-optimality of a balanced design matrix reaches this lower bound, the design matrix is orthogonal. This conclusion was proved by Xu (2002). The lower bound, $L(n)$, is given as follows,

$$L(n) = 2^{-1}\left[\left(\sum_{k=1}^{m} n s_k^{-1} w_k\right)^2 + \left(\sum_{k=1}^{m}(s_k - 1)\left(n s_k^{-1} w_k\right)^2\right) - n\left(\sum_{k=1}^{m} w_k\right)^2\right].$$

Recall the orthogonal design OA (12, $3^1 2^4$) discussed in Section 2-3-1. It is easy to verify the lower bound is also 330. Therefore, this design is orthogonal, because the J_2 value is equal to the lower bound.

2-3-3 Construction Algorithm Based on J_2-optimality

Xu (2002) provides an algorithm to construct orthogonal designs based on J_2-optimality. The basic idea of the algorithm is to add columns sequentially to an existing design. The general construction algorithm is stated as follows. Considering the change in the J_2 value if a column is added to d or if two levels are switched in a column. A column c=$(c_1, c_2, \ldots, c_n)^T$ is added to d and let d_+ be the new $n \times (m+1)$ design. Then for $1 \leq i, j \leq n$

$$\delta_{i,j}(d_+) = \delta_{i,j}(d) + w_k \delta_{i,j}(c),$$

where

$$\delta_{i,j}(c) = \delta(c_i, c_j),$$

and

$$J_2(d_+) = J_2(d) + 2w_k \sum_{1 \leq i < j \leq n} \delta_{i,j}(d)\delta_{i,j}(c) + 2^{-1} n w_k^2 \left(n s_k^{-1} - 1\right).$$

Then all $\delta_{i,j}(c)$ are unchanged except that $\delta_{a,j}(c) = \delta_{j,a}(c)$ and $\delta_{b,j}(c) = \delta_{j,b}(c)$ are switched for $j \neq a,b$. Hence, $J_2(d_+)$ is reduced by $2w_k\Delta(a,b)$, where

$$\Delta(a,b) = \sum_{1\le j\neq a,b\le n} \left[\delta_{a,j}(d) - \delta_{b,j}(d)\right] \left[\delta_{a,j}(c) - \delta_{b,j}(c)\right].$$

The following cited procedure to construct orthogonal mixed-level designs is summarized by Xu in 2002.

1. For k=1,…,n, compute the lower bound $L(k)$ according to (2).
2. Specify an initial design d with two columns: (0, …, 0, 1, …, 1, …, s_1-1, …, s_1-1) and (0, …, s_2-1, 0,…, s_2-1, …, 0, …, s_2-1). Compute $\delta_{i,j}(d)$ and $J_2(d)$. If $J_2(d) = L(2)$, let $n_0 = 2$ and $T = T_1$; otherwise, let $n_0 = 0$ and $T = T_2$.
3. For $k = 3,...,n$, do the following:
   (a) Randomly generate a balanced s_k-level column c. Compute $J_2(d_+)$ by definition. If $J_2(d_+) = L(k)$, go to (d).
   (b) For all pairs of rows a and b with distinct levels, compute $\Delta(a,b)$ as in (5). Choose a pair of rows with largest $\Delta(a,b)$ and exchange the symbols in rows a and b of column c. Reduce $J_2(d_+)$ by $2w_k\Delta(a,b)$. If $J_2(d_+) = L(k)$, go to (d); otherwise, repeat (b) until no further improvement is made.
   (c) Repeat (a) and (b) T times and choose a column c that produces the smallest $J_2(d_+)$.
   (d) Add column c as the k^{th} column of d, let If $J_2(d) = J_2(d_+)$ and update $\delta_{i,j}(d)$ by (3). If $J_2(d_+) = L(k)$, let n_0=k; otherwise, let T=T_2.

2-4 Genetic Algorithms

The foundation of genetic algorithms was developed by John Holland in the 1960s. By the middle of 1980s, genetic algorithms had provided solutions to complex problems in optimization, machine learning, programming, and job scheduling. Instead of using gradient information, a genetic algorithm can be used to perform a direct search.

Genetic algorithms are attractive not only because they are relatively easy to implement, but also mathematically they do not require differentiable objective functions.

As Gen and Cheng (2000) mentions, genetic algorithms are evolutionary search strategies based on simplified rules, biological population genetics and theories of evolution. A genetic algorithm maintains a population of candidate solutions for a problem, and then selects those candidates most fit to solve the problem. After the selection process, the most fit candidate solutions are combined and altered by reproduction operators to produce new solutions for the next generation. The process continues, with each generation evolving more fit solutions until an acceptable solution has evolved. Figure 2.2 shows a flowchart of the working principle of a genetic algorithm.

For any genetic algorithm, the construction of the genetic representation of a chromosome is required. In general, a genetic algorithm has five basic components:

1. A genetic representation of solutions to the problem
2. A way to create an initial population of solutions
3. An evaluation function rating solutions in terms of their fitness
4. Genetic operators that alter the genetic composition of children during reproduction, and
5. Values for the parameters of genetic algorithms.

2-4-1 Encode Chromosomes

Implementation of genetic algorithms are based upon the chromosomes. A chromosome represents a potential solution to the problem of interest and traditionally is represented by a string of genes that are either binary encoded or real number encoded. In a real number representation, genes are encoded with real number while in a binary representation; genes are encoded 0 or 1. Besides binary encoding and real number encoding, integer or literal permutation encoding and general data structure encoding are also used for genetic algorithms.

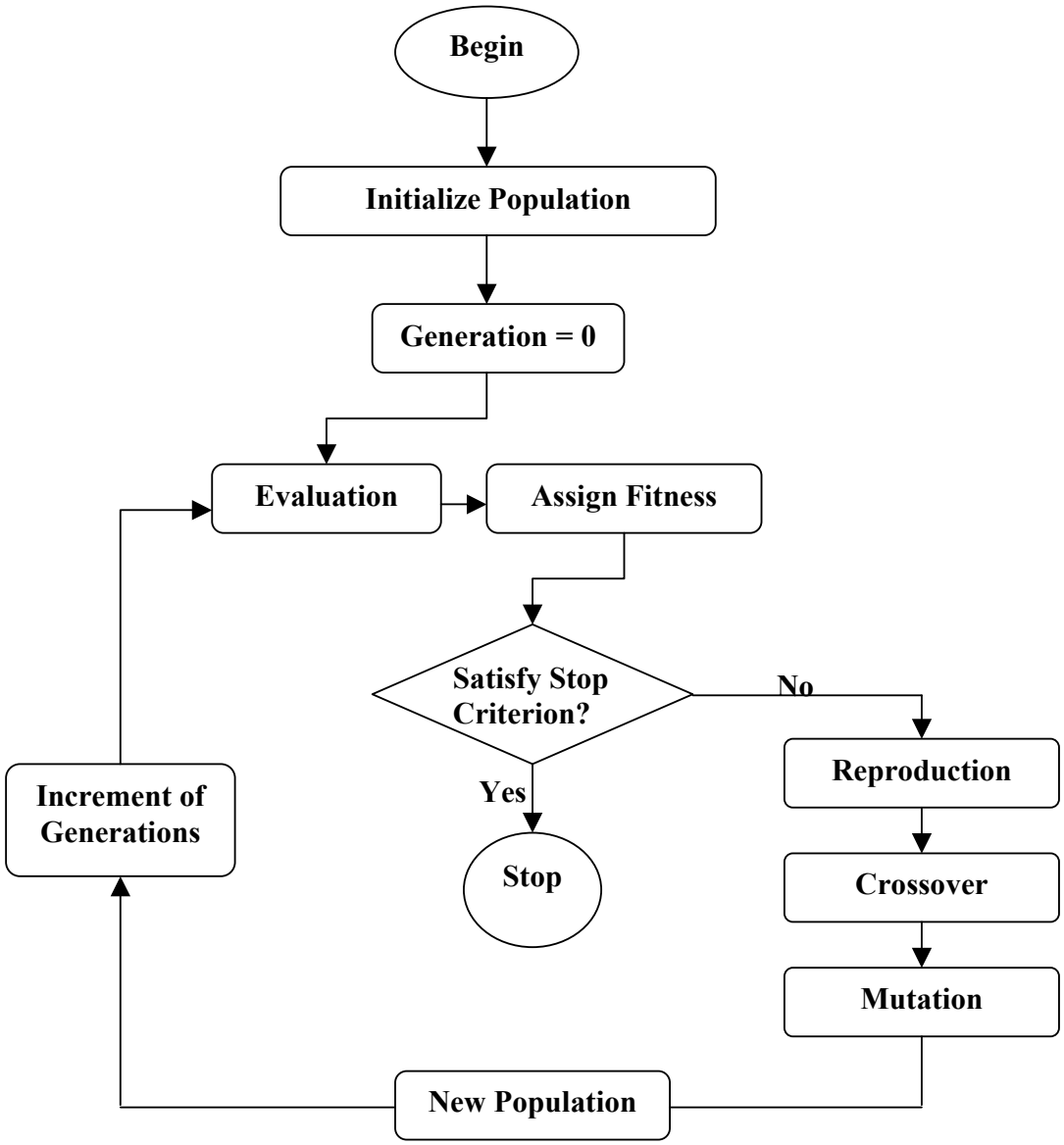

Figure 2.2 A flowchart of the working principle of a genetic algorithm.

2-4-2 Genetic Operations

Typically there are two types of search behaviors: random search and local search. Random search explores the entire solution space and is capable of achieving an escape from a local optimum. Local search exploits the best solution and is capable of climbing upward toward a local optimum. The two types of search capabilities form the mutually complementary components of optimization. An ideal search should possess both types simultaneously. It is almost impossible to design such a search method using traditional techniques. Genetic algorithms are a class of general-purpose search methods combining elements of directed and stochastic searches, which can make a good balance between exploration and exploitation of the search space. In conventional genetic algorithms, the crossover operator is used as the principal operator and the performance of a genetic system is heavily dependent on it. The mutation operator that produces spontaneous random changes in various chromosomes is used as a background operator. In essence, genetic operators perform a random search and cannot guarantee to yield improved offspring because of its heuristic sense. Commonly used genetic operations are crossover and mutation.

2-4-2-1 Crossover operation and crossover rate. The crossover operation is one of the most commonly used forms of recombination. A typical crossover involves two chromosomes, called parents. These two distinct chromosomes are selected randomly from the population and broken at the same location. The new chromosomes, called offspring, are obtained by combining opposite portions of the parent chromosomes. The parent chromosomes can also be broken in more than one location.

The crossover rate refers to the percent of the parent population X that will undergo a crossover operation. A single point crossover technique is employed throughout our experimentation since the chromosome string length is relatively short. While a high crossover rate typically causes good points to be discarded, a low crossover rate places too much emphasis on parents and may stagnate the search.

2-4-2-2 Mutation operation and mutation rate. Mutation is used to alter the genetic material of a very small number of individuals in a random fashion, enhancing the diversity of the population and expanding the volume of the search space.

After a crossover is performed, mutation takes place. This operation is intended to prevent all solutions in a population from converging to a local optimum of a solved problem. Mutation randomly changes the new offspring chromosome. For binary encoding we can switch a few randomly chosen genes from 1 to 0 or from 0 to 1.

Mutation refers to the altering of one or more genes in a chromosome. Mutation types are characterized by how many genes are altered and the degree to which they are changed. As Ortiz (2003) mentions, two types appropriate for real-valued coding are uniform and Gaussian mutation. Uniform mutation changes only one gene according to a uniform distribution. The Gaussian type changes all the genes by a small amount according to normally distributed disturbance.

A chromosome altered by Gaussian mutation results in a point located in the neighborhood of its parent. Gaussian mutation alters the vector X by adding a randomly generated vector $m = (m_1, m_2, ..., m_k)$ consisting of mild, normally distributed perturbations. The new design space point becomes $X^* = X + m$.

Bashir (2003) first applies genetic algorithms to construct supersaturated designs. In his dissertation, supersaturated designs are constructed using different approaches to create the initial population. The principle of his method will be introduced in the following section.

2-5 Construction of Supersaturated Designs by Genetic Algorithms

A supersaturated design is a fractional factorial design in which the number of factors m exceeds the number of runs n. A design is called supersaturated when $m \geq n - 1$.

Because of the special features of the supersaturated designs, there exist all kinds of construction methods. However, the designs may not be the best in terms of the criteria used for generation. Bashir (2003) proposed a construction method for improved supersaturated designs by the criterion of $E(s^2)$.

The $E(s^2)$ optimality criterion has been the primary criterion used for comparing different supersaturated designs. A smaller value is preferred and a value of zero represents orthogonality. The $E(s^2)$ criterion was used in his research for comparing different supersaturated designs, and it was considered as the fitness function in a genetic algorithm approach. Regarding the approach, supersaturated designs are constructed based on the $E(s^2)$ criterion using different approaches to create the initial population for genetic algorithm. The basic idea is to find a reasonable initial population of candidate columns which will be used by the genetic algorithm to select the best columns using the $E(s^2)$ criterion.

The operating matrices include the Hadamard matrix, a general balanced matrix containing all balanced combinations of the minus and plus signs, and a combined matrix formed from a Hadamard and balanced incomplete block designs.

The genetic algorithm was executed by first constructing an initial solution which consists of certain candidate columns; each candidate column is considered a gene. This chromosome length is equal to the number of factors in the design. Each gene in the chromosome will represent a column in the supersaturated design. In order to construct a chromosome, binary coding is used. The 0 value indicates that the column is not selected, and the 1 value indicates that the column is selected. Then the fitness function $E(s^2)$ is calculated for that solution. Using the crossover and the mutation operations, the algorithms then perform reproduction of a new solution. The fitness function for the new design is calculated and compared to the previous one. This operation is repeated until no further improvement in the fitness function is achieved.

2-5-1 The $E(S^2)$ Criterion

Bashir (2003) constructs supersaturated designs based upon the $E(s^2)$ criterion, which is used to measure orthogonality of design matrices. Let s_{ij} denote the dot product of columns i and j in the design matrix X, then the criterion is given by

$$E(s^2) = \frac{\sum s_{ij}^2}{\binom{m}{2}}.$$

The $E(s^2)$ criterion gives an intuitive measure of the degree of non-orthogonality, the smaller the better. Tang and Wu (1997) derived a lower bound for $E(s^2)$, meaning optimal with respect to $E(s^2)$. The lower bound of $E(s^2)$, which is a function of n and m, is defined as

$$E(s^2) \geq \frac{n^2(m-n+1)}{(m-1)(n-1)} \qquad (m \geq n),$$

where m is the number of columns of the design and n is the number of runs.

2-5-2 An Example

The following example is given in order to understand his method. Consider the Hadamard matrix H_{12} below. From this matrix, a supersaturated design with $\frac{N}{2} = 6$ runs for a maximum of $N - 2 = 10$ factors could be produced. A process contains eight factors is required to be studied and investigated by using a supersaturated design approach. The following matrix shows the Hadamard matrix H_{12}, consisting of 12 rows and 12 columns. The H_{12} matrix is orthogonal and all its entries are minus one and plus one. It is noted that the first column and the first row of this matrix is ones.

By arbitrarily choosing any column as the branching column of matrix H_{12}, the total of 12 runs can be split into two groups. Suppose the last column is selected as the branching column, so group I has the sign of one in the branching column of H_{12} and group II has the minus one in the branching column.

$$H_{12} = \begin{array}{c} \\ \left[\begin{array}{cccccccccccc:c} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \text{GI} \\ 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & \text{GII} \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & \text{GI} \\ 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & \text{GII} \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & \text{GII} \\ 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & \text{GII} \\ 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & \text{GI} \\ 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & \text{GI} \\ 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & \text{GI} \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & \text{GII} \\ 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & \text{GI} \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & \text{GII} \end{array}\right] \end{array} \begin{array}{c} \text{Group} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}$$

Deleting the first column and the branching columns from GI, results in a matrix with 6 rows and 10 columns. This matrix is a supersaturated design that examines $N-2=10$ factors in $\frac{N}{2}=6$ runs.

$$X = \begin{array}{c} \begin{array}{cccccccccc} \textit{1} & \textit{2} & \textit{3} & \textit{4} & \textit{5} & \textit{6} & \textit{7} & \textit{8} & \textit{9} & \textit{10} \end{array} \\ \left[\begin{array}{cccccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 \\ -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \end{array}\right] \end{array}$$

In the above matrix, there are 6 rows and 10 candidate columns to construct a supersaturated design for 8 factors. Any combination consists of 8 factors out of the 10

candidate columns could be a candidate solution. There will be 45 candidate solutions in this case. As an example, one solution could be formed by selecting the first eight columns of the matrix. The value of s_{ij} for the design matrix, for example the value of s_{42} is equal to 2 and is calculated as follows:

$$s_{42} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \end{pmatrix} \bullet \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = 2$$

The value of $E(s^2)$ for every design is calculated and the lowest value is picked to be the best design among the others. Figure 2.3 shows the optimal solution based on $E(s^2)$ for the X matrix.

		m	=	8							112
				2		E(S2)	=	4.00			
		n	=	6							
	1	2	3	4	5	6	7	8	9	10	m
	1	1	1	1	1	1	1	1	0	0	8
	1	2	3	4	5	6	7	8	9	10	
	Sij										E(S2)
1	0	2	2	2	2	-2	2	2	-2	-2	28
2	0	0	2	2	-2	2	2	-2	2	-2	24
3	0	0	0	-2	2	2	-2	2	2	-2	20
4	0	0	0	0	2	2	2	-2	-2	2	16
5	0	0	0	0	0	2	-2	2	-2	2	12
6	0	0	0	0	0	0	-2	-2	2	2	8
7	0	0	0	0	0	0	0	2	2	2	4
8	0	0	0	0	0	0	0	0	2	2	0
9	0	0	0	0	0	0	0	0	0	2	0
10	0	0	0	0	0	0	0	0	0	0	0

Figure 2.3 Optimal solution output.

This supersaturated design construction method provides optimal designs most of the time. The genetic algorithm chooses the best columns from the candidate columns to

build an optimal supersaturated design based on $E(s^2)$. However, in this thesis, genetic algorithms will be used to choose rows from full design matrices based on different criteria.

CHAPTER 3

METHODOLOGY

This thesis concentrates on the construction of efficient mixed-level designs with desirable balance and orthogonality properties. A new optimality criterion, called the balance coefficient, is proposed in Section 3-1. This balance coefficient is defined using two alternate formulations. These definitions provide for a numerical evaluation of design matrices in terms of their balance property. In Section 3-2, the J_2-optimality criterion is modified to the standardized J_2-optimality, which is independent of design sizes. Based on these balance and orthogonality criteria, in Section 3-3, a genetic algorithm is developed to construct efficient mixed-level designs.

3-1 Balance Coefficient for Factorial Designs

Balance property is an important property for factorial designs. For a balanced design, there is consistency in the variance of the difference between two treatment combinations. As such, it is necessary to develop a new criterion called the balance coefficient, to assess the degree of balance for design matrices. Before introducing the definition of the balance coefficient, some related notation is given as follows.

Let n represent the number of rows and m represent the number of columns for design matrices. For each column j, w_j is the corresponding weight, which shows the relative importance of this column compared to the others. The column j contains l_j levels and l_{ij} are the frequency of the i^{th} level in that column. The standardized number of levels for a specific level is defined as $f_{ij} = \frac{l_{ij}}{n}$.

For a column j, the number of each level is shown by a pattern of $[l_{1j}\ l_{2j}\ \ldots l_{l_j j}]$. For example, for a 3-level, 10-run column, [3 4 3] means this column contains three 1^{st} levels, four 2^{nd} levels, and three 3^{rd} levels.

The balance situation of a design can be represented by these l_{ij}, which denote the numbers of i^{th} level in factor j. Two alternate formulations of balance coefficient will be given in Sections 3-1-2 and 3-1-3. Before the statement of these definitions, the graphical representation of $[l_{1j}\ l_{2j}\ \ldots\ l_{l_j j}]$ will be first introduced in Section 3-1-1.

3-1-1 Graphical Representation

The pattern of $[l_{1j}\ l_{2j}\ \ldots\ l_{l_j j}]$ indicates the balance situation of a column. When the l_{ij} equal each other, that column is balanced. Because the summation of l_{ij} is the number of runs, which is a constant, these l_{ij} can be represented by a coordinate plane. For example, a twelve-run column has three levels. These three levels are represented by the notation (1, 2, 3). Then the number of levels can be expressed by an equation

$$l_{11} + l_{21} + l_{31} = 12\ .$$

This equation has two degrees of freedom, and represents a coordinate plane in three-dimensional space, which is shown in Figure 3.1.

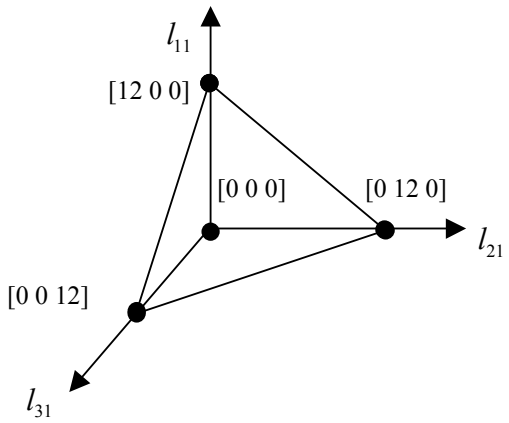

Figure 3.1 Graphical representation of a 3-level, 10-run column.

All the feasible solutions of $l_{11} + l_{21} + l_{31} = 12$ are discretely scattered on this plane, which is shown in Figure 3.2. The point in the center, [4 4 4], represents the most balanced situation. In general, the column with k levels can be represented by

$$\sum_{i=1}^{k} l_{ij} = n,$$

which denotes a hyperplane.

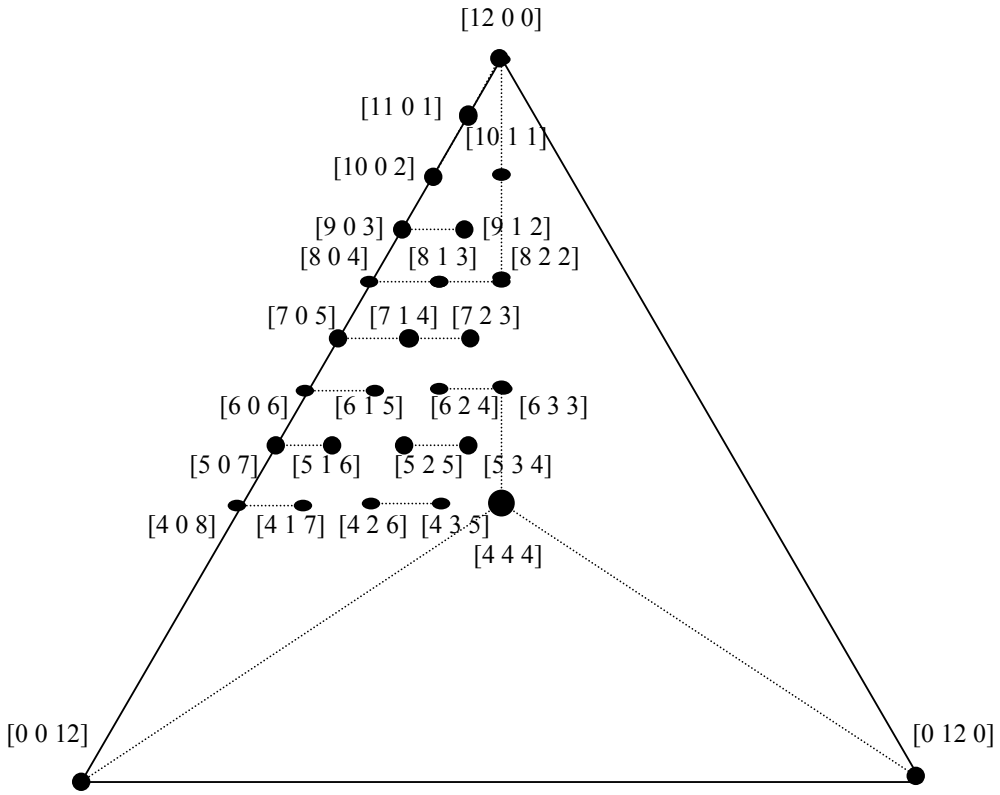

Figure 3.2 Feasible solutions of $l_{11}+l_{21}+l_{31}=12$.

Based on the nature of l_{ij}, a balance coefficient needs to be defined. The objective of this new optimality criterion is to provide numerical evaluation for design matrices in terms of balance property. The definition is given in two forms.

3-1-2 Formulation of Balance Coefficient - Form I

In form I, the motivation behind the definition of the balance coefficient is a simple optimization problem. The balance coefficient of design matrices will be derived from this optimization problem. The problem can be formulated as,

$$\text{Max} \quad G = \prod_{k=1}^{K} X_k$$

$$\text{Subject to} \quad \sum_{k=1}^{K} X_k = C \quad ,$$

where C is a constant. It can be shown that when

$$X_i = X_j = \frac{C}{K} \text{ for all } i \text{ and } j,$$

$$\max(G) = \left(\frac{C}{K}\right)^K .$$

To apply this conclusion to experimental designs, F_j are defined as the balance coefficient for column j, and

$$F_j = \prod_{i=1}^{l_j} l_{ij} ,$$

where l_{ij} is the number of the i^{th} level. When l_{ij} are equal to each other, and

$$l_{ij} = \frac{n}{l_j},$$

then the F_j reach their maximum values, which are

$$\max(F_j) = \left(\frac{n}{l_j}\right)^{l_j} .$$

The balance coefficient for design matrix d, $F(d)$, is defined as the combination of the balanced coefficient of each column, F_j,

$$F(d) = \sum_{j=1}^{m} w_j F_j = \sum_{j=1}^{m} \left(\prod_{i=1}^{l_j} l_{ij} \right) w_j ,$$

where w_j are the weights for the corresponding columns. This balance coefficient depends on the number of runs. To avoid this situation, a standardized balance coefficient is defined by using a standardized number of levels.

The balance coefficient is standardized when the number of levels is standardized. The notation f_{ij} is used instead of l_{ij}. In this case, the definitions of the standardized balance coefficient for a specific column and for a design matrix are given as follows:

$$\hat{F}_j = \prod_{i=1}^{l_j} f_{ij} = \frac{\left(\prod_{i=1}^{l_j} l_{ij} \right)}{n^{l_j}},$$

$$\hat{F}(d) = \sum_{j=1}^{m} w_j \hat{F}_j = \sum_{j=1}^{m} \frac{\left(\prod_{i=1}^{l_j} l_{ij} \right) w_j}{n^{l_j}}.$$

When

$$f_{ij} = \frac{1}{l_j},$$

$$\max(\hat{F}_j) = \frac{\left(\frac{n}{l_j} \right)^{l_j}}{n^{l_j}} = \frac{1}{l_j^{l_j}}.$$

Consider the unbalanced design matrix in Figure 3.3 with seven factors: A, B, C, D, E, F and G. Factor G has 4 levels and all other factors have 3 levels. This design contains 10 experimental runs, so n=10, and m=7. The numbers of levels and the balance coefficient for each factor are shown in Table 3.1.

Run	*G*	*A*	*B*	*C*	*D*	*E*	*F*
1	1	1	1	1	1	1	1
2	1	2	2	2	2	2	2
3	1	3	3	3	3	3	3
4	2	1	1	2	2	3	3
5	2	2	2	3	3	1	1
6	2	3	3	1	1	2	2
7	3	1	2	1	2	2	3
8	3	2	3	2	1	3	1
9	3	3	1	3	2	1	2
10	4	1	3	3	2	2	1

Figure 3.3 An unbalanced design.

Table 3.1 Computation of Balance Coefficient for the Design Shown in Figure 3.3

Factor	Number of Level	Balance Coefficient
G	[3 3 3 1]	$\hat{F}_1 = \frac{3 \cdot 3 \cdot 3 \cdot 1}{10^4} = 0.0027$
A	[4 3 3]	$\hat{F}_2 = \frac{4 \cdot 3 \cdot 3}{10^3} = 0.036$
B	[3 3 4]	$\hat{F}_3 = \frac{3 \cdot 3 \cdot 4}{10^3} = 0.036$
C	[3 3 4]	$\hat{F}_4 = \frac{3 \cdot 3 \cdot 4}{10^3} = 0.036$
D	[3 4 3]	$\hat{F}_5 = \frac{3 \cdot 4 \cdot 3}{10^3} = 0.036$
E	[3 4 3]	$\hat{F}_6 = \frac{3 \cdot 4 \cdot 3}{10^3} = 0.036$
F	[4 3 3]	$\hat{F}_7 = \frac{4 \cdot 3 \cdot 3}{10^3} = 0.036$

The balance coefficient of this design is

$$\hat{F}(d) = \prod_{j=1}^{7} w_j \hat{F}_j = 0.0035,$$

where $w_j = \frac{1}{7}$ are used, meaning all factors will be treated equally.

From this example, the balance coefficient under form I is consistent with the degree of balance. For a ten-run, three-level column, the [3 3 4] pattern is the most balanced situation and has the maximum balance coefficient. Compare the patterns [1 1 8], [2 2 6], [3 3 4], and [4 4 2]. Pattern [4 4 2] has a larger balance coefficient than [2 2 6]. The balance coefficient treats each level equally. For example, [3 3 4] for level (1, 2, 3) has the same balance coefficient as patter [3 4 3]. The balance coefficient is also applicable in a subset of all levels. For example, if the number of the 1^{st} level is fixed at two, in terms of the balance coefficient, [2 1 7] < [2 2 6] < [2 3 5] < [2 4 4].

In form I, the balance coefficient with maximum value is the most balanced situation. By contrast, in form II, the smaller balanced coefficient will denote a more balanced situation.

3-1-3 Formulation of Balance Coefficient - Form II

In form II, the definition of balance coefficient employs the concept of the distance function. Consider a distance function

$$H_j = \sum_{i=1}^{l_j} \left(l_{ij} - T \right)^2 ,$$

where $T = \dfrac{n}{l_j}$ is a fixed value.

The balance coefficient under this definition becomes

$$H_j = \sum_{i=1}^{l_j} \left(l_{ij} - \frac{n}{l_j} \right)^2$$

and

$$H = \sum_{j=1}^{m} H_j = \sum_{j=1}^{m} w_j \sum_{i=1}^{l_j} \left(l_{ij} - \frac{n}{l_j} \right)^2 .$$

If f_{ij} are used instead of l_{ij}, then standardized H_j and H can be given by

$$\hat{H}_j = \sum_{i=1}^{l_j} \left(f_{ij} - \frac{1}{l_j} \right)^2 \text{, and}$$

$$\hat{H} = \sum_{j=1}^{m} w_j \hat{H}_j = \sum_{j=1}^{m} w_j \sum_{i=1}^{l_j} \left(f_{ij} - \frac{1}{l_j} \right)^2 . \qquad (3.1)$$

When the design is balanced,

$$f_{ij} = \frac{1}{l_j} \text{ and } l_{ij} = \frac{n}{l_j} \text{ for all } i \text{ in } j ,$$

the distance function reaches its minimum value, which is

$$\hat{H}^* = 0 .$$

It should be understood that some designs may never reach zero balance coefficients, because in some cases $l_{ij} = \frac{n}{l_j}$ are not integer values. The balance coefficient is maximized when

$$l_{ij} = \begin{cases} n & i = k \\ 0 & i \neq k \end{cases},$$

where k represents an arbitrary integer number in between 1 and l_j.

Although both formulations can be used to define the balance property of design matrices, form II is preferred in this thesis. First, designs with smaller balance coefficients are more balanced in form II, but less balanced in form I. This feature is consistent with the J_2-optimality, which is used to evaluate the orthogonality of design matrices. In addition, the balance coefficient has lower polynomial degrees in form II than that in form I. Moreover, the distance function of form II has a nicer geometric interpretation.

3-2 Standardized J_2-Optimality

Besides the balance property, orthogonality is another desirable property when constructing mixed-level designs. The concept of J_2-optimality is used to optimize the orthogonality of design matrices. Consider a $n \times m$ matrix $D = [\, x_{ik} \,]$, with weights

w_k>0,and s_k levels for each factor. The weights indicate the relative importance of each factor and if w_k=1 are chosen, $\delta_{i,j}(d)$ is the number of coincidences between the i^{th} and j^{th} rows. The J_2-optimality is

$$J_2(d) = \sum_{1 \le i < j \le n} \left[\sum_{k=1}^{m} w_k \delta(x_{ik}, x_{jk}) \right]^2,$$

where $\delta(x, y) = 1$ if x=y and 0 otherwise.

For a fixed number of runs n, a smaller J_2 value is desired. However in order to compare the J_2 values of designs with different numbers of runs, it is necessary to derive the standardized J_2-optimality.

The standardized coincidence is defined as follows. For any two rows, x_i and x_j,

$$\hat{\delta}(x_i, x_j) = \frac{\delta(x_i, x_j)}{m} = \frac{\sum_{k=1}^{m} \delta(x_{ik}, x_{jk})}{m},$$

where m is length of the row. The interpretation of this standardized coincidence value is the degree of similarity between these two rows.

Instead of using the sum of squares of the coincidences, the standardized J_2-optimality is defined as the averaged coincidence, which is

$$\hat{J}_2(d) = \overline{\hat{\delta}(x_i, x_j)} \quad \text{for all } 1 \le i, j \le n. \tag{3.2}$$

Figure 3.4 shows an example, which is a 2^{3-1} design.

d =

1	1	1
2	1	2
1	2	2
2	2	1

Figure 3.4 A 2^{3-1} design.

This design is a balanced orthogonal design. By definition, $J_2(d) = 6$ and $\hat{J}_2(d) = 0.1111$.

3-3 Construction of Mixed-level Designs by Genetic Algorithms

The basic idea of the thesis is to choose several rows from a full design matrix and combine these rows into a small size design matrix based upon some criteria. Genetic algorithms are employed to perform this task. The criteria used in the objective function of the GA are the balance coefficient and J_2-optimality.

3-3-1 Encoding the Problem

Implementation of genetic algorithms is based on the chromosomes. In this thesis, each chromosome represents a design. Each gene in the chromosome denotes an experimental run. Both binary encoding and real-value encoding can be used. For example, consider a design with three factors A, B, and C: A with 2 levels, B with 3 levels, and C with 4 levels. The full design is shown in Table 3.2.

Table 3.2 Full Size $2^1 3^1 4^1$ Factorial Design

Identity	A	B	C
1	1	1	1
2	2	1	1
3	1	2	1
4	2	2	1
5	1	3	1
6	2	3	1
7	1	1	2
8	2	1	2
9	1	2	2
10	2	2	2
11	1	3	2
12	2	3	2
13	1	1	3
14	2	1	3
15	1	2	3
16	2	2	3

Table 3.2 Continued

Identity	A	B	C
17	1	3	3
18	2	3	3
19	1	1	4
20	2	1	4
21	1	2	4
22	2	2	4
23	1	3	4
24	2	3	4

The full factorial contains 24 design points. Assume a small design with 12 runs is desired. In the case of binary encoding of chromosomes, "1" indicates this design point is chosen and "0" otherwise. For the GA, each chromosome will contain 24 genes. Table 3.3 shows examples of three binary encoded chromosomes.

Table 3.3 Chromosome Representation Using Binary Encoding

Chromosome 1	1	0	1	0	1	1	1	0	0	0	1	0	1	0	0	0	0	1	1	0	0	1	1	1
Chromosome 2	0	1	0	1	1	1	0	0	1	0	0	0	1	1	0	1	0	0	1	1	1	1	0	0
Chromosome 3	1	0	1	0	1	1	0	1	0	0	0	1	0	1	1	0	1	0	0	1	0	1	0	1

For the case of real-value encoded chromosomes, each chromosome string contains only 12 genes. These genes represent the identity numbers of the rows in the full design. Table 3.4 shows three examples encoded by the real-value scheme.

Table 3.4 Chromosome Representation Using Real-value Encoding

Chromosome 1	1	3	4	5	6	10	12	17	18	21	22	23
Chromosome 2	2	4	5	6	9	13	14	16	19	20	21	22
Chromosome 3	1	3	5	6	8	12	14	15	17	20	22	24

Both encoding schemes can be used in this research. In order to use binary encoded chromosomes, the number of "1's" in each chromosome has to equal the desirable design sizes. Besides, binary encoded chromosomes contain more genes. These longer chromosomes slow the program convergence speed. In the case that the real-valued scheme is used to form chromosomes, two undesirable situations may occur. First, the order of genes in chromosomes may not be ascending. As a result, it is necessary to sort the chromosomes before performing genetic operations. In this way, the genetic algorithms can easily recognize new chromosomes which helps increase the efficiency of the genetic algorithm. Second, some chromosomes may contain identical genes after several generations. In order to generate designs without replicate rows, a correction process will be inserted to ensure that the genes in the chromosomes are unique.

3-3-2 Chromosome Evaluation Using An Objective Function

Several chromosomes constitute a population. The genetic algorithm maintains a population to perform genetic operations on selected chromosomes. Chromosomes are selected based on the value of their objective function. The objective function measures the fitness value of each chromosome. The objective function, Z, used in this thesis research is a linear combination of the balance coefficient and J_2-optimality, such that

$$Z = \hat{J}_2(d) + a\hat{H}(d) .$$

where a is a coefficient used to adjust the relative weight of the two criteria.

Equation 3.2 and 3.1 are used in Z such that

$$Z = \sum_{1 \le i < j \le n} \left[\sum_{k=1}^{m} w_k \delta(x_{ik}, x_{jk}) \right]^2 + a \sum_{j=1}^{m} w_j \sum_{i=1}^{l_j} \left(f_{ij} - \frac{1}{l_j} \right)^2 ,$$

where smaller values are desired.

Based on this objective function, chromosomes with better fitness values are generated and the population is being updated in each generation. The process continues until the stopping criteria are satisfied. The stopping criterion used in this thesis is the maximum generations. The genetic algorithm to construct mixed-level designs will be developed in the following section.

3-3-3 Genetic Algorithm for Mixed-level Designs

Based upon the principle of genetic algorithms and their application in design of experiments, a genetic algorithm is developed to construct mixed-level designs. The general outline of this genetic algorithm is given as follows.

MIXED_LEVEL_GA

- •Input parameters
- •Generate full design matrix
- •Initialize chromosomes
- •While (generational counter < stop criterion)
  - –Save the best
  - –Fitness assignments to population
  - –Select chromosomes from population
  - –Crossover of selected chromosomes
  - –Mutate offspring
  - –Calculate fitness value for offspring
  - –Insert best offspring in population replacing worst parents
  - –Increment generational counter
- •Output

Figure 3.5 The GA to construct mixed-level designs.

The Figure 3.5 outline will be discussed in this section. The program used for coding is MATLAB 6.1. A Genetic Algorithm MATLAB Toolbox is also used. This Toolbox was developed by the Department of Automatic Control and System Engineering of the University of Sheffield in 1994. For more information regarding this GA MATLAB

Toolbox, see its User's Guide (Chipperfield, 1994). The details of the functions in this toolbox will not be discussed in this thesis, but the application of these functions in this genetic algorithm is explained as follows.

Since this GA program is tuned to meet various experimental design requirements, there are some parameters that need to be defined as input parameters. Generally, the input parameters of the GA program have the format of

(a,na,b,nb,c,nc,d,nd,e,ne,f,nf,g,ng,h,nh,i,ni,j,nj,m),

where *a, b, c, d, e, f, g, h, i, j* are factor levels; *na, nb, nc, nd, ne, nf, ng, nh, ni,* and *nj* represent the numbers of factors corresponding to these levels; and *m* is the desirable run sizes.

When input parameters are given, the full design matrix, containing all the possible level combinations, is already determined. There are several ways to generate the full design matrices. One way consists of generating unrepeated rows. The function *full_design_matrix* in Appendix accomplishes this outcome. The first row has ones in all the entries. Then a new row is generated randomly. This new random row is compared with the previous (*i*-1) rows. If the new row is different from all the existing rows, then name this new row as the i^{th} row; otherwise, regenerate a new row and repeat the process until an eligible row is generated.

Based upon the generated full design matrix, chromosomes have to be initialized. Generation of initial chromosomes uses the function *crtbp(Nind, Lind, Base)* in the Genetic Algorithm Toolbox. In this function, *Nind*×*Lind* is the dimension of the population. *Nind* specifies the number of chromosomes in the population and *Lind* defines the length of the chromosomes. The function *crtbp* is only used to generate binary or integer initial populations. The parameter *Base* is the number of rows in the full design matrix.

Each chromosome represents a design matrix. The chromosome fitness is evaluated by an objective function. The objective function is used to provide a measure of the balance and orthogonality properties of experimental designs represented by the chromosomes. Because minimization of *Z* is desired, the most fit chromosomes will have the lowest numerical *Z* value of the associated objective function.

In every generation, a certain proportion of chromosomes with the best fitness will be selected for future use. These selected chromosomes will be used to perform genetic operations. Selection is the process of choosing a specific number of chromosomes for genetic operations. The commonly used operations are crossover and mutation, which have been discussed in Section 2-4-2. In this research, the function used for crossover is *recombin*. Crossover produces new chromosomes that have some parts of both parents' genetic material. For this research using real-value representations, mutation is achieved by either perturbing the gene values or randomly selecting of new values within the allowable range. Function *mut* is used for binary and integer mutation in the GA Toolbox.

The chromosomes generated by genetic operations will replace old ones. To maintain the size of the original population, the new chromosomes have to be reinserted into the old population. The Genetic Algorithm Toolbox provides a function for reinserting chromosomes into the population after genetic operations, *reins*.

It is necessary to specify the stop criterion for the generation loop. Commonly used stop criteria are the error-based stop criterion and the iteration-based stop criterion. An error-based stop criterion makes the loop stop when the error is less than the required value. An iteration-based stop criterion makes the loop stop when all the required iterations finish. In this program, an iteration-based stop criterion in use is the maximum generations.

The outputs of the algorithm are a population of chromosomes with the best fitness value, representing several mixed-level designs with desirable balance and orthogonality properties. Chapter 4 discusses the performance of the genetic algorithm for this research. This mixed-level design construction method will be evaluated and validated by generating several balanced orthogonal mixed-level designs. These designs have the zero balance coefficient and their J_2-optimality equal the lower bounds. Because the stop criterion used in the genetic algorithm is the maximum number of generations, it is necessary to find a proper number for the maximum generations.

CHAPTER 4

RESULTS AND ANALYSIS

The mixed-level design construction method developed in this thesis is very efficient. Designs with different numbers of runs are investigated in terms of the standardized balance coefficient and the standardized J_2-optimality in Section 4-1. Several balanced orthogonal designs are generated in Section 4-2 so as to validate this construction method. The performance of the program is analyzed in this section as well. As the important results, some efficient designs are generated and analyzed in Section 4-3.

4-1 Investigation of Balance Coefficient and J_2-optimality for Designs with Different Numbers of Runs

It is desirable to compare the designs with different numbers of runs in terms of the balance coefficient and J_2-optimality. Both criteria are standardized and independent of design sizes. Firstly, the simulation analysis is performed. Then the theoretical analysis is conducted for comparison and further study. The design used for analysis in this section is a mixed-level design with one 3-level factor, one 5-level factor and one 7-level factor.

4-1-1 Analysis of Simulation Results

4-1-1-1 Appropriate maximum generations. It is necessary to choose a proper bound for maximum number of generations, which is used for stopping the generation loop. In order to analyze the variance of the program performance, the sample size of 100 is chosen. Each simulation is performed 100 times to investigate the program

stability in terms of the standardized balance coefficient $\hat{H}$ and the standardized J_2-optimality, $\hat{J}_2$. The statistics used to measure the stability are the mean and the standard deviation. Different numbers of generations, 100, 500, 1000, 5000 and 10,000, are investigated. The corresponding statistics are showed in Table 4.1.

Table 4.1 Statistics for *EA* (20, $3^1 5^1 7^1$) with Sample=100

Generations	100	500	1000	5000	10,000
Mean ($\hat{H}$)	0.0040	0.0031	0.0030	0.0021	0.0020
Std ($\hat{H}$)	0.0017	0.0013	0.0012	0.0011	0.0011
Mean ($\hat{J}_2$)	0.0767	0.0750	0.0743	0.0734	0.0729
Std ($\hat{J}_2$)	0.0017	0.0015	0.0013	0.0013	0.0011

The sample means of both criteria decrease when using more generations. The standard deviations of these two criteria also decrease by using more generations. The more generations are in use, the better the algorithm performs. The generations of 10,000 will be used in the thesis under the consideration of efficiency and reliability.

4-1-1-2 Simulation and analysis. Consider a three-factor mixed-level design. Notation *EA*(*n*, $3^1 5^1 7^1$) displays the factor levels as the base numbers and the number of factors with that many levels as the exponent. *n* is the number of runs. Designs with different run sizes are generated and the corresponding balance coefficient and standardized J_2-optimality are listed in Table 4.2 and plotted in Figure 4.1. Table 4.3 collects some statistics of the standardized balance coefficient and J_2-optimality. Note the sample correlation coefficient between the standardized balance coefficient and the standardized J_2-optimality is 0.2402, which does not show any strong correlation between the standardized balance coefficient and the standardized J_2-optimality. Observing Figure 4.1, the standardized J_2-optimality is "normally" plotted except first 4 points. However, if the first four points are considered as outliers and removed, the sample correlation coefficient, –0.58683, is negative and relatively strong, which is shown in Figure 4.2.

This negative correlation shows that when standardized J_2-optimality increases, the balance coefficient decreases. Figure 4.3 shows an enlarged plot of the standardized balance coefficients. This plot shows a periodic, decreasing trend.

Table 4.2 $\hat{H}$ and $\hat{J}_2$ for Different Number of Runs

***EA* (*n*, $3^1 5^1 7^1$)**					
n	$\hat{H}$	$\hat{J}_2$	n	$\hat{H}$	$\hat{J}_2$
15	0.0050	0.0974	28	0.0008	0.0800
16	0.0029	0.0954	29	0.0025	0.0826
17	0.0039	0.0997	30	0.0020	0.0817
18	0.0030	0.1024	31	0.0018	0.0829
19	0.0027	0.0721	32	0.0012	0.0820
20	0.0029	0.0708	33	0.0020	0.0838
21	0.0036	0.0720	34	0.0030	0.0846
22	0.0019	0.0750	35	0.0013	0.0859
23	0.0021	0.0755	36	0.0009	0.0857
24	0.0015	0.0765	37	0.0023	0.0864
25	0.0013	0.0774	38	0.0008	0.0871
26	0.0024	0.0779	39	0.0006	0.0865
27	0.0028	0.0788	40	0.0004	0.0872

Table 4.3 Statistics of $\hat{H}$ and $\hat{J}_2$

	$\hat{H}$	$\hat{J}_2$
Max	0.0050	0.1024
Min	0.0004	0.0708
Mean	0.002138	0.083358
Std	0.001404	0.008314
Correlation Coefficient	0.2402	
Correlation Coefficient (19 - 40)	-0.58683	

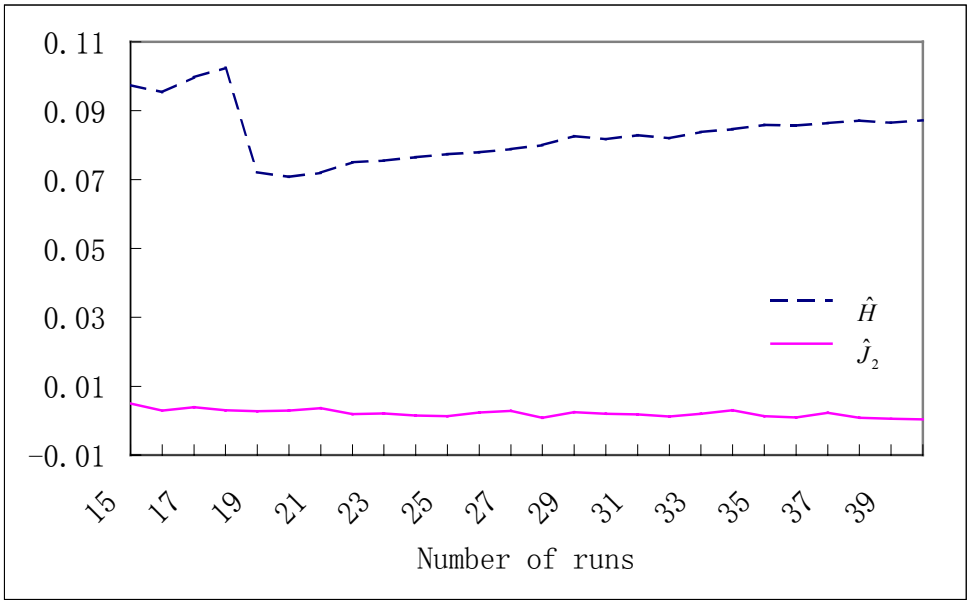

Figure 4.1 Plot of $\hat{H}$ and $\hat{J}_2$ vs. number of runs.

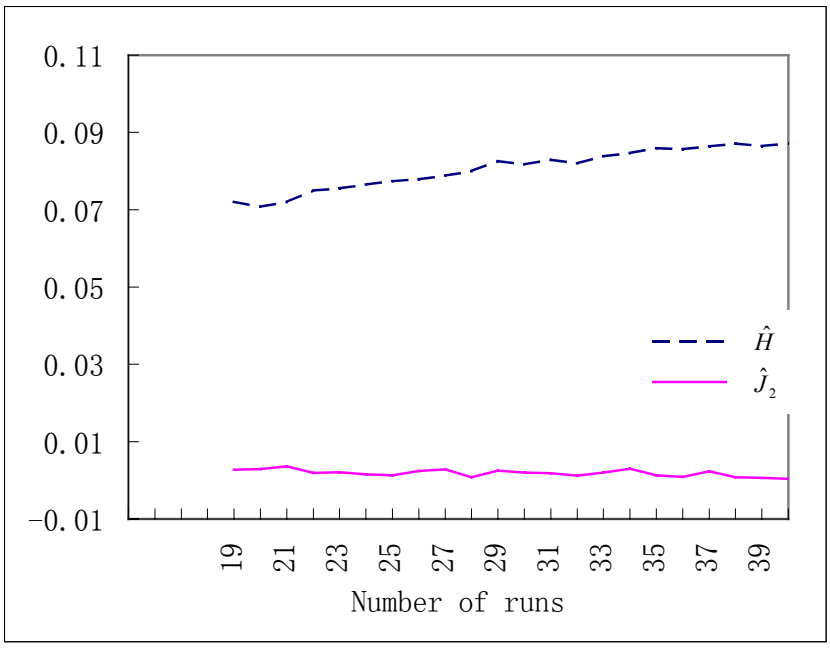

Figure 4.2 Adjusted plot of $\hat{H}$ and $\hat{J}_2$ vs. number of runs.

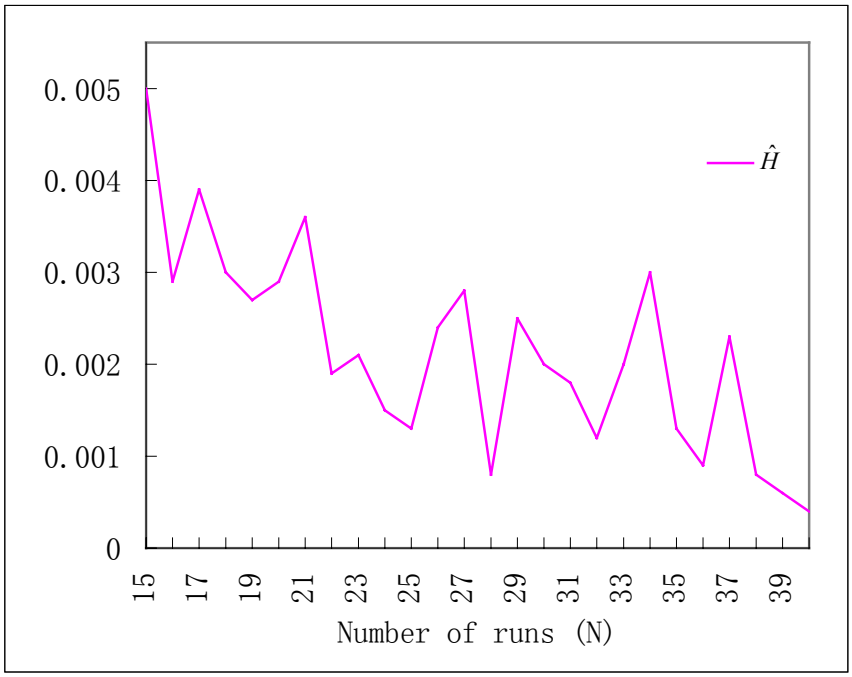

Figure 4.3 Enlarged adjusted plot of $\hat{H}$ vs. number of runs.

In order to clarify the nature of the standardized balance coefficient and J_2-optimality as criteria for varying design sizes, a theoretical analysis is needed.

4-1-2 Analysis of Theoretical Results

4-1-2-1 Standardized balance coefficient. By definition, the balance coefficient,

$$\hat{H}_j = \sum_{i=1}^{l_j} \left(f_{ij} - \frac{1}{l_j} \right)^2 ,$$

$$\hat{H} = \sum_{j=1}^{m} w_j \hat{H}_j = \sum_{j=1}^{m} w_j \sum_{i=1}^{l_j} \left(f_{ij} - \frac{1}{l_j} \right)^2$$

can be calculated if f_{ij} are known. For a specific number of runs, the most balanced situation can be easily determined. Table 4.4 shows the calculation of balance coefficient with different n. These balance coefficient values are then plotted in Figure 4.4.

Table 4.4 Computation of the Ideal Standardized Balance Coefficient

n	Column1	$\hat{H}_1$	Column 2	$\hat{H}_2$	Column 3	$\hat{H}_3$	$\sum \hat{H}_j$	$\hat{H}$
20	[6 7 7]	0.0025	[4 4 4 4 4]	0	[2 3 3 3 3 3 3]	0.0021	0.0046	0.0015
21	[7 7 7]	0	[4 4 4 4 5]	0.0018	[3 3 3 3 3 3 3]	0	0.0018	0.0006
22	[7 7 8]	0.0014	[4 4 4 5 5]	0.0025	[3 3 3 3 3 3 4]	0.0018	0.0056	0.0019
23	[7 8 8]	0.0013	[4 4 5 5 5]	0.0023	[3 3 3 3 3 4 4]	0.0027	0.0062	0.0020
24	[8 8 8]	0	[4 5 5 5 5]	0.0014	[3 3 3 3 4 4 4]	0.0030	0.0044	0.0015
25	[8 8 9]	0.0011	[5 5 5 5 5]	0	[3 3 3 4 4 4 4]	0.0027	0.0038	0.0013
26	[8 9 9]	0.0010	[5 5 5 5 6]	0.0012	[3 3 4 4 4 4 4]	0.0021	0.0043	0.0014
27	[9 9 9]	0	[5 5 5 6 6]	0.0016	[3 4 4 4 4 4 4]	0.0012	0.0028	0.0009

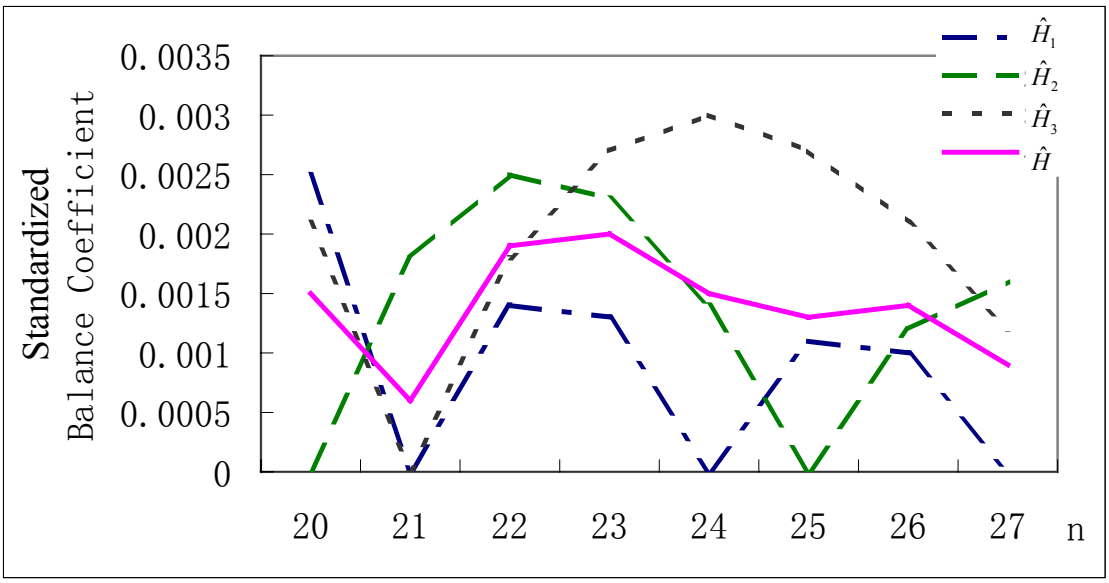

Figure 4.4 The optimal standardized balance coefficient vs. number of runs.

From Table 4.4 and Figure 4.4, the conclusions match those drawn from the simulation analysis. The trend of periodically decreasing $\hat{H}$ can still be observed. Both the simulation and theoretical results show that the standardized balance coefficient of the design decreases periodically. The mathematic model and corresponding proof is given as follows.

A three-level column has n runs, and

$$f_{11} = \frac{l_{11}}{n},$$

$$f_{21} = \frac{l_{21}}{n},$$

and

$$f_{31} = \frac{l_{31}}{n}.$$

Suppose three more experimental runs are added, to each level is added one more run, then the situation changes to

$$f'_{11} = \frac{l_{11}+1}{n+3},$$

$$f'_{21} = \frac{l_{21}+1}{n+3},$$

and

$$f'_{31} = \frac{l_{31}+1}{n+3}.$$

Here $f'_{11}+f'_{21}+f'_{31}=1$ still holds. Then the objective is to prove that the balance coefficient with n+3 runs will be less than or equal to the balance coefficient with n runs. The following inequality is desired to be proved

$$\sum_{i=1}^{3}\left(\frac{l_{i1}+1}{n+3}-\frac{1}{3}\right)^2 \le \sum_{i=1}^{3}\left(\frac{l_{i1}}{n}-\frac{1}{3}\right)^2.$$

The proof will use two facts:

Fact I. Given $a<b, c<d$,

$$\frac{a+c}{b+d} > \frac{a}{b},$$

if

$$b > a\frac{d}{c}$$

Fact II. Cauchy Inequality

$$\sum_{i=1}^{n}(a_i b_i)^2 \le \left(\sum_{i=1}^{n} a_i^2\right)\left(\sum_{i=1}^{n} b_i^2\right).$$

Proof

Let $c = \frac{1}{3}$, then the left hand side can be decomposed as

$$\sum_{i=1}^{3}\left(\frac{l_{i1}+1}{n+3}-c\right)^2 = \sum_{i=1}^{3}\left(\frac{l_{i1}+1}{n+3}\right)^2 - 2c\sum_{i=1}^{3}\left(\frac{l_{i1}+1}{n+3}\right)+3c^2.$$

Because

$$\sum_{i=1}^{3}\left(\frac{l_{i1}+1}{n+3}\right)=1,$$

$$\sum_{i=1}^{3}\left(\frac{l_{i1}+1}{n+3}-c\right)^2 = \sum_{i=1}^{3}\left(\frac{l_{i1}+1}{n+3}\right)^2 - 2c+3c^2.$$

From this simplified quantity, the original problem is independent of the value of *c*. Assuming c=0, then

$$\sum_{i=1}^{3}\left(\frac{l_{i1}+1}{n+3}\right)^2 = \frac{\sum_{i=1}^{3}l_{i1}^{\ 2}+2\sum_{i=1}^{3}l_{i1}+3}{(n+3)^2}.$$

Because of

$$\sum_{i=1}^{3}l_{i1}=n,$$

then

$$\sum_{i=1}^{3}\left(\frac{l_{i1}+1}{n+3}\right)^2 = \frac{\sum_{i=1}^{3}l_{i1}^{\ 2}+2n+3}{n^2+6n+9}.$$

At this moment, the original proof is actually

$$\frac{\sum_{i=1}^{3}l_{i1}^{\ 2}+2n+3}{n^2+6n+9} \le \frac{\sum_{i=1}^{3}l_{i1}^{\ 2}}{n^2}.$$

According to Fact I, this inequality will hold if

$$\sum_{i=1}^{3}l_{i1}^{\ 2} \ge \frac{n^2}{3}$$

or

$$\sum_{i=1}^{3} l_{i1}^{\,2} \geq \frac{\left(\sum_{i=1}^{3} l_{i1}\right)^2}{3}$$

This is the special case of the well-known Cauchy-Inequality in the three variables case, where a_i=l_{i1} and b_i=1. The equality holds when l_{i1} equal each other. The proof is then obtained. The generalized inequality is stated as Lemma 1.

Lemma 1. For a $n \times 1$ column, if l_j more runs are added. Then following inequality exists

$$\sum_{i=1}^{l_j}\left(\frac{l_{i1}+1}{n+l_j}-\frac{1}{l_j}\right)^2 \leq \sum_{i=1}^{l_j}\left(\frac{l_{i1}}{n}-\frac{1}{l_j}\right)^2,$$

and the equality holds if and only if l_{i1} equal each other.

By this lemma, the conclusion is that a column will be more or equally balanced if the number of added runs equals the number of levels.

Lemma 2. For a design, suppose k more runs are added. If

$$k = LCM(l_1, l_2, ..., l_m),$$

the design with these k more runs becomes more or equally balanced. Notation *LCM* denotes the least common multiple. Therefore,

$$\hat{H}(d_2) \leq \hat{H}(d_1),$$

where d_2 is the design after adding k more runs.

The proof of Lemma 2 will use the Lemma 1 and will not be stated in this thesis. After analyzing the standardized balance coefficient, the increasing trend of the standardized J_2-optimality will be analyzed.

4-1-2-2 Standardized J_2-optimality. The standardized J_2-optimality for a design matrix will increase if more runs are available. Recalling Table 4.2 and Figure 4.2, standardized J_2-optimality shows a increasing trend. That is, the standardized J_2-

optimality increases as the number of runs increases. With more runs incorporated, the similarity of these runs will decrease. For example, consider two runs of a design, which are shown in Figure 4.5a. Assuming two more runs are added, the design is shown in Figure 4.5b.

a) b)

Figure 4.5 Design matrices with two runs and four runs.

The average similarity will decrease after these two more runs are added. In order to validate the conclusion, further simulation with 100,000 generations is performed and the results are displayed in Table 4.5 and plotted in Figure 4.6.

Table 4.5 $\hat{J}_2$ with Different Number of Runs (Validation with More Generations)

***EA* (n, $3^1 5^1 7^1$)**				
n	$\hat{J}_2$		n	$\hat{J}_2$
14	0.0574		40	0.0872
15	0.0593		41	0.0882
16	0.0630		42	0.0878
29	0.0813		49	0.0905
30	0.0812		50	0.0909
31	0.0822		51	0.0913
Balanced design			105	0.0983

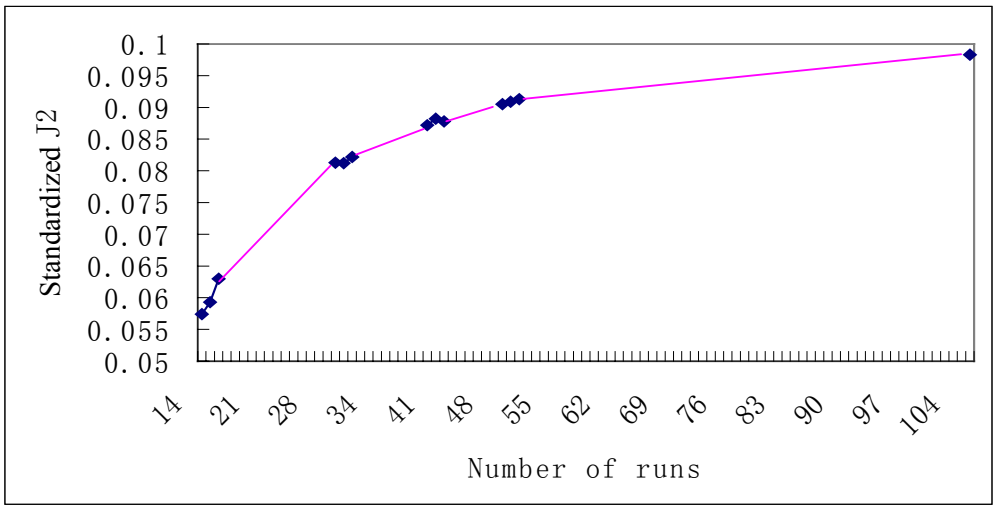

Figure 4.6 Increasing $\hat{J}_2$ trend with 100,000 generations.

The conclusion is that with more runs, the standardized J_2-optimality values will increase. Note that with 105 runs, the standardized J_2-value has the biggest value. By the discussion in this section, the number of runs affects both balance and orthogonality properties. Additional experimental runs will increase the degree of orthogonality for design matrices, but the balance may decrease to some degree with these more runs. As such, design sizes should be determined with a certain degree of care.

4-2 Generation of the Balanced Orthogonal Mixed-level Designs and Performance of the Program

If the number of runs for mixed-level designs is chosen properly, it is possible to generate balanced, orthogonal designs. For a balanced mixed-level design, the number of runs is found in relation to the number of factors and the number of levels of each factor. In Section 4-2-1, a method will be developed to find this relation by using the concept of least common multiple.

4-2-1 Determine Number of Runs by the Concept of Least Common Multiple

The least common multiple of t numbers of $a_1, a_2, ..., a_t$, denoted by $LCM(a_1, a_2, ..., a_t)$, is the smallest number n for which there exist positive integers n_{a_1}, $n_{a_2} \ldots n_{a_t}$ such that

$$n_{a_1} a_1 = n_{a_2} a_2 = ... = n_{a_t} a_t = n .$$

Then the least common multiple is expressed as

$$LCM(a_1, a_2, ..., a_t) = n_{a_1} a_1 = n_{a_2} a_2 = ... = n_{a_t} a_t = n .$$

Regarding mixed-level designs, l_j is taken to be a_t where l_j are numbers of levels of the factor j. For a mixed-level design with two 2-level factors and six 3-level factors, the least common multiple notation is as follows:

$$LCM(2,2,3,3,3,3,3,3) .$$

The number of runs is also in relation to the degrees of freedom necessary for lower order models. Using the least common multiple to find the run size, let

$$n = f(LCM(l_1, ..., l_m), k_1, ..., k_m) .$$

where k_j is the number of l_j-level factors. An appropriate function for f is still being investigated.

Suppose the number of runs is given properly, it is possible to generate balanced mixed-level designs. For a balanced orthogonal design, the balance coefficient is zero and J_2-optimality equals its lower bound. However, because the genetic algorithm is a heuristic method, GA does not guarantee the optimal solution. Therefore, it is necessary to study the performance of this algorithm by generating balanced orthogonal designs.

4-2-2 Case Study 1: $OA(4, 2^3)$

In this case study, consider an experimental design, which contains three two-level factors. The full design matrix has 8 runs. In order to test the performance of the algorithm, a small size design with only 4 runs is generated and shown in Figure 4.7.

Under the condition of balance, the J_2-optimality reaches its lower bound. This design is a balanced orthogonal design. Figure 4.8 shows the convergence of the best fitness value of the chromosomes in each generation. Convergence occurs at fourth generation.

$$D = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$$

Figure 4.7 $OA(4, 2^3)$ generated by GA.

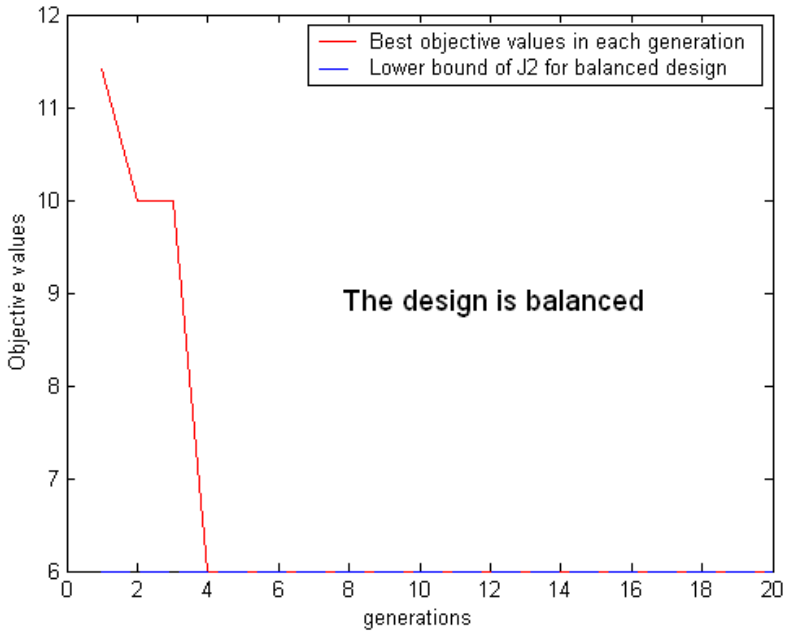

Figure 4.8 Plot of the best value vs generations for $OA(4, 2^3)$.

4-2-3 Case Study 2: $OA(8, 2^2 4)$

This study is a simple mixed-level design. Three factors are considered: two two-level factors and one four-level factor. The number of runs of the full design is 16. The algorithm generates a small size balanced orthogonal design with 8 runs, which is shown

in Figure 4.9. Figure 4.10 shows the plot of the best fitness values in each generation, and convergence occurs at around seventieth generation.

$$D = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 4 \\ 2 & 1 & 3 \\ 2 & 1 & 4 \\ 2 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix}$$

Figure 4.9 $OA(8, 2^2 4)$ generated by GA.

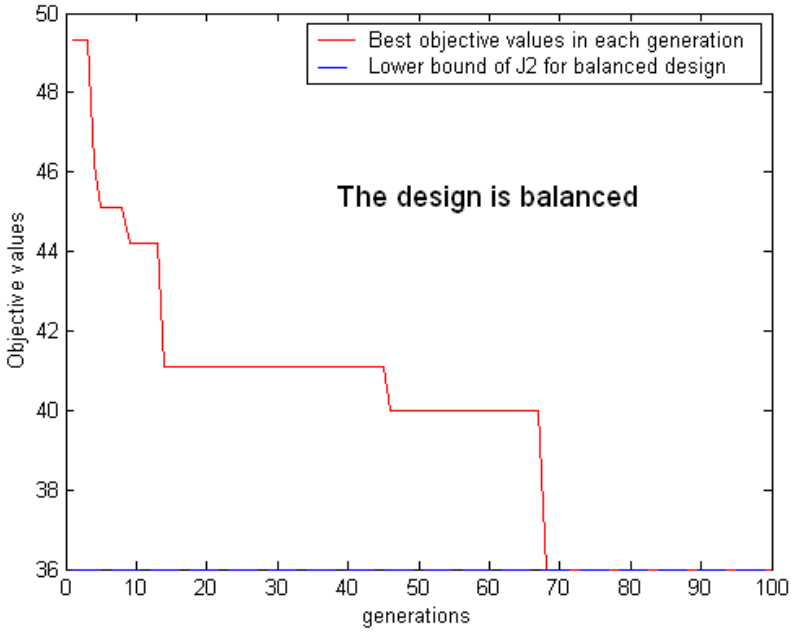

Figure 4.10 Plot of the best value vs generations for $OA(8, 2^2 4)$.

4-2-4 Case Study 3: $OA(12, 2^4 3)$

In case 3, a design considers five factors: four two-level factors and one three-level factor. This example shows a little more complicated situation. The number of runs

for the full design is 48. The small balanced orthogonal designs with 12 runs are desired. Figure 4.12 shows the converging process. Although the total number of generations is 300,000, convergence occurs before 50,000th generation. Two small size balanced orthogonal designs are generated and shown in Figure 4.11. Compare these two designs with the orthogonal design generated by Xu (2002), which is shown in Figure 4.13. These three designs are actually the different fractions of the full design with the same balance and orthogonality properties. The research of the comparison of these designs is still being conducted.

However, sometimes pre-convergence will happen. In this situation, genetic algorithm converges before it should be. In fact, genetic algorithm reduces its efficiency when it goes close to the optimal solution. Figure 4.14 is an example of pre-convergence. The fitness value improves dramatically at first 10,000 generations, and also improves between 30,000 and 40,000, and then it stops improvement. By adjusting the parameters of GA or the objective functions, pre-convergence could be mitigated, but it cannot be prevented.

$$D_1 = \begin{bmatrix} 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 2 & 2 \\ 1 & 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 & 1 \\ 2 & 2 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 1 & 1 & 2 & 1 \\ 3 & 1 & 1 & 1 & 2 \\ 3 & 2 & 2 & 1 & 2 \\ 3 & 2 & 2 & 2 & 1 \end{bmatrix} \quad D_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 2 \\ 1 & 2 & 2 & 2 & 1 \\ 2 & 1 & 1 & 2 & 1 \\ 2 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 1 & 1 & 2 & 2 \\ 3 & 1 & 2 & 2 & 1 \\ 3 & 2 & 1 & 1 & 2 \\ 3 & 2 & 2 & 1 & 1 \end{bmatrix}$$

Figure 4.11 $OA(12, 2^4 3)$ generated by GA.

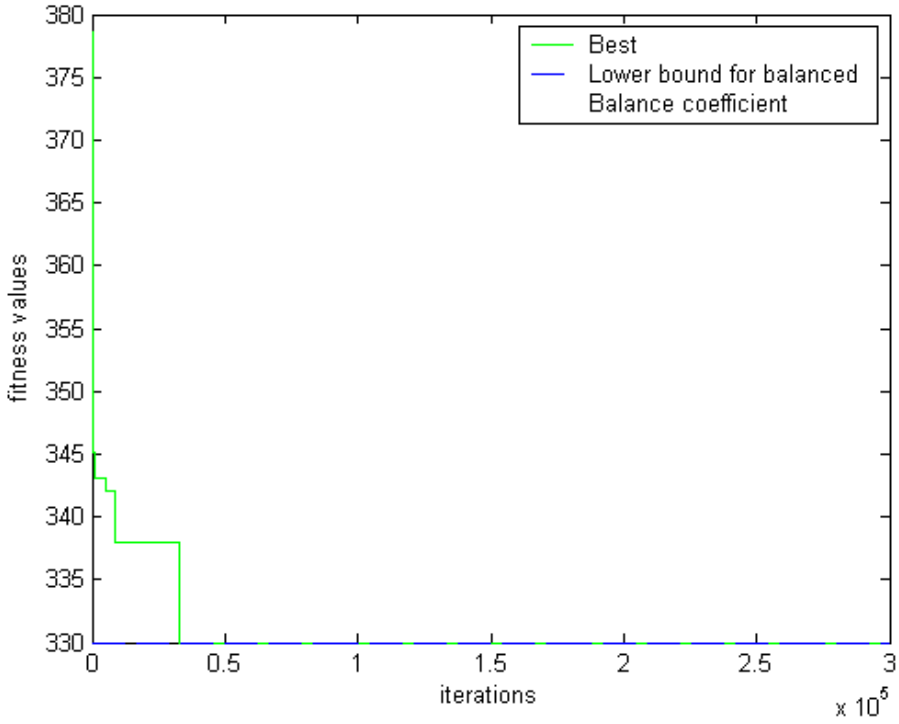

Figure 4.12 Plot of the best value vs. generations for $OA(12, 2^4 3)$.

$$D = \begin{bmatrix} 1 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 1 & 2 \\ 1 & 1 & 2 & 2 & 1 \\ 1 & 2 & 1 & 2 & 1 \\ 2 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 1 & 1 \\ 2 & 1 & 2 & 2 & 2 \\ 2 & 2 & 1 & 2 & 2 \\ 3 & 1 & 1 & 2 & 1 \\ 3 & 2 & 2 & 1 & 1 \\ 3 & 1 & 1 & 1 & 2 \\ 3 & 2 & 2 & 2 & 2 \end{bmatrix}$$

Figure 4.13 $OA(12, 2^4 3)$ generated by Xu, 2002.

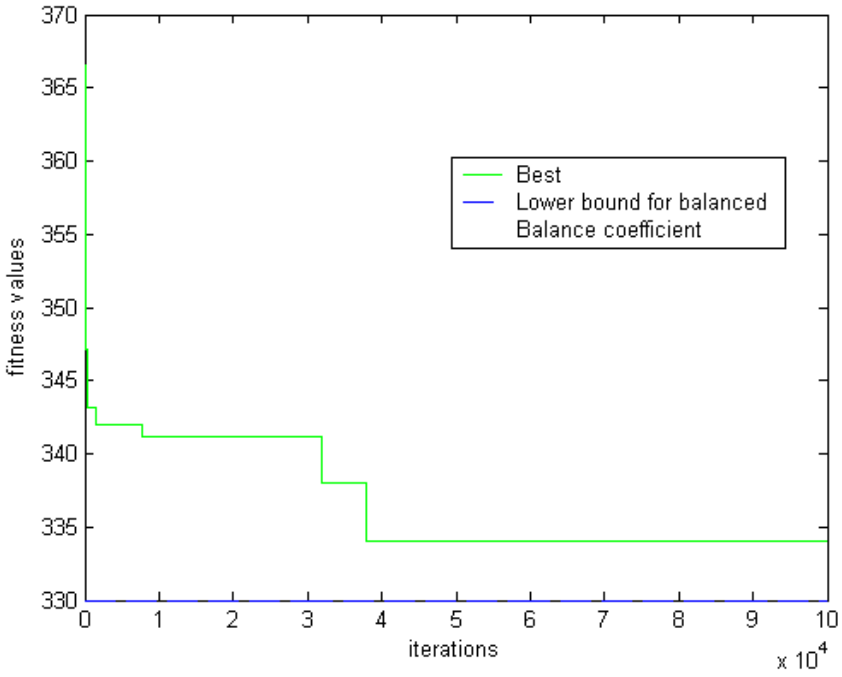

Figure 4.14 A pre-convergence case $OA(12, 2^4 3)$.

4-3 Generation of the Efficient Mixed-level Designs

Although balanced orthogonal mixed-level designs can be generated, the major contribution of this thesis is to construct efficient mixed-level designs with near balance and orthogonality properties. In this section, several efficient designs will be generated. All these designs involve three factors: one with three levels, one with five levels and one with seven levels. The full design matrix for these three factors needs 105 runs. In fact, 105 is the smallest number of runs to make all these three factors balanced. It is impossible to generate any balanced design with less than 105 runs. In this situation, efficient designs with near-balance and near-orthogonality properties will be considered. Three case studies will be conducted: $EA(15, 3^1 5^1 7^1)$, $EA(21, 3^1 5^1 7^1)$ and $EA(30, 3^1 5^1 7^1)$. By choosing these three numbers: 15, 21 and 30, at least two factors can be balanced. For each case study, the value of standardized balance coefficient and J_2-optimality are given as references.

4-3-1 Case study 1: *EA* (15, $3^1 5^1 7^1$)

In the first case study, the efficient designs with fifteen runs are of interest. The generated design is shown in Figure 4.15. Table 4.6 shows the corresponding statistics of this design. Both 3-level and 5-level factors are balanced, but the 7-level factor is not balanced. The balance coefficient of this design matrix is 0.0013, which is the possibly most balanced situation. No design is more balanced than this design in 15 runs.

Table 4.6 Statistics of Generated *EA* (15, $3^1 5^1 7^1$)

Column 1	Column 2	Column 3
[5 5 5]	[3 3 3 3 3]	[2 3 2 2 2 2 2]
$\hat{H}$	J_2	
0.0013	54	

D =

1	1	7
1	2	2
1	3	5
1	4	4
1	5	3
2	1	2
2	2	5
2	3	6
2	4	7
2	5	1
3	1	4
3	2	1
3	3	2
3	4	3
3	5	6

Figure 4.15 *EA* (15, $3^1 5^1 7^1$) generated by GA.

4-3-2 Case study 2: *EA* (21, 3^1, 5^1, 7^1)

In this case study, twenty-one design points are considered. Two simulations are performed and two designs, D_1 and D_2, are generated and compared, which are shown in Figure 4.16. Table 4.7 shows the statistics for these two designs. The 3-level factor and 7-level factor for both designs are balanced, but the 5-level factor is not balanced for either situation, even though D_1 is more balanced than D_2. The pattern of 5-level factor in D_1 is [4 4 4 5 4] and the pattern in D_2 is [3 4 5 5 4]. In terms of J_2-optimality, D_2 is more orthogonal than D_1.

4-3-3 Case study 3: *EA* (30, $3^1 5^1 7^1$)

Case study 3 considers the design in 30 runs. The generated design is shown in Figure 4.17. Both 3-level factor and 5-level factor are balanced. However, the 7-level factor is unbalanced. This factor has the pattern of [4 3 3 5 5 5 5], which can still be improved. Improvement of the design could be achieved by reducing both the balance coefficient and J_2-optimality.

This efficient mixed-level design construction method shows its efficiency when balanced orthogonal designs do not exist. The algorithm optimizes both balance and orthogonality properties, although the optimal solution is not guaranteed. To generate designs with better balance and orthogonality properties, use more generations. The genetic algorithm uses the combination of balance coefficient and J_2-optimality as its objective function. Therefore, mixed-level designs with same fitness values but different balance coefficient and different J_2-optimality may simultaneously exist. Furthermore, even though some designs have exactly same balance coefficient and J_2-optimality values, these designs are still different designs. Further study to compare these designs is still being investigated.

Table 4.7 Statistics of Generated *EA* (21, $3^1 5^1 7^1$)

Column 1	Column 2	Column 3
[7 7 7]	[3 3 3 3 3 3 3]	[4 4 4 5 4]
$\hat{H}$	J_2	
0.0006	132	

Column 1	Column 2	Column 3
[7 7 7]	[3 3 3 3 3 3 3]	[3 4 5 5 4]
$\hat{H}$	J_2	
0.0021	131	

$D_1 =$

1	1	3
1	2	5
1	3	4
1	4	4
1	5	5
1	6	2
1	7	3
2	1	4
2	2	2
2	3	1
2	4	3
2	5	4
2	6	1
2	7	5
3	1	2
3	2	1
3	3	3
3	4	2
3	5	1
3	6	5
3	7	4

$D_2 =$

1	1	1
1	2	4
1	3	3
1	4	5
1	5	4
1	6	3
1	7	2
2	1	2
2	2	1
2	3	5
2	4	2
2	5	5
2	6	4
2	7	3
3	1	4
3	2	3
3	3	4
3	4	1
3	5	3
3	6	2
3	7	5

Figure 4.16 *EA* (21, $3^1 5^1 7^1$) generated by GA.

Table 4.8 Statistics of Generated *EA* (30, $3^1 5^1 7^1$)

Column 1	Column 2	Column 3
[10 10 10]	[6 6 6 6 6]	[4 3 3 5 5 5 5]
$\hat{H}$	J_2	
0.0020	318	

D =

1	1	4
1	1	7
1	2	3
1	2	5
1	3	1
1	3	6
1	4	5
1	4	7
1	5	2
1	5	6
2	1	4
2	1	7
2	2	3
2	2	6
2	3	2
2	3	7
2	4	4
2	4	6
2	5	1
2	5	5
3	1	1
3	1	6
3	2	1
3	2	7
3	3	4
3	3	5
3	4	2
3	4	5
3	5	3
3	5	4

Figure 4.17 *EA* (30, $3^1 5^1 7^1$) generated by GA.

CHAPTER 5

CONCLUSIONS AND FUTURE RESEARCH

5-1 Conclusions

Mixed-level factorial designs are the necessary alternatives to the traditional two-level factorial designs when qualitative factors are present. Mixed-level designs are employed when factors with more than two levels are involved. Usually, only balanced orthogonal and near-orthogonal designs are constructed. Based on the genetic algorithm, this thesis proposes a construction method to generate efficient mixed-level designs with desirable balance and orthogonality properties.

In order to evaluate the balance property of a design, the criterion of balance coefficient is proposed in this thesis. Two alternative formulations of the balance coefficient are provided, form I and form II. The formulation in form I is derived from an optimization problem. The balance coefficient under this definition measures the degree of unbalance of mixed-level designs. A design is balanced when the balance coefficient is maximized. By contrast, under the formulation in form II, a design is balanced when the balance coefficient is minimized. In form II, the balance coefficient is defined by applying the distance function and is used in this thesis. The balance coefficient is standardized so as to compare the balance property of designs with different numbers of runs. Besides the balance coefficient, the J_2-optimality criterion is also standardized and independent of the number of runs as well. J_2-optimality measures the degree of orthogonality for design matrices. This thesis investigates the standardized balance coefficient and the standardized J_2-optimality of designs with a varied number of runs. If additional runs are added, the degree of orthogonality decreases but the balance property increases periodically. For a column, assume the additional experiment runs are available. If the number of added runs equals the multiple of the number of level of that factor, the

column is either more or equally balanced. However, if the number of added runs is not equal to the multiple of the number of levels of the factor, the column is either more or less balanced, depending on the current number of runs and the added number of runs. For a design, this conclusion holds if the added number of runs equals the least common multiple of the factor levels. Taking into account both the balance property and the orthogonality property, the genetic algorithm is applied in constructing efficient mixed-level designs. Basically, this construction algorithm performs very well. Balanced orthogonal mixed-level designs can be generated by this construction algorithm, which shows the capability of this method. The major contribution of this thesis is the construction of efficient mixed-level designs associated with optimal near-balance and near-orthogonality properties. The efficient mixed-level fractional designs are the solution to the excessive run requirements associated with many balanced mixed-level designs.

5-2 Future Research Suggestions

There are many aspects of this research that may be further investigated. For a balanced mixed-level design, the run size is determined in relation to the number of factors and the numbers of levels of the factors. The concept of the least common multiple is helpful in finding this number. The number of runs is also determined in relation to the degrees of freedom necessary for lower order models. Therefore, an explicit formula able to generate this number is a topic worth investigating.

The balance coefficient is defined using two alternate formulations and only form II is investigated to generate efficient mixed-level designs in this research. Further comparison and assessment of these definitions is expected. For a column in a design matrix, the plot of the standardized balance coefficients shows a periodic decreasing pattern when the number of runs increases. This thesis provides a mathematic proof that, when the number of added runs is the multiple of the factor levels, this column will be more or equally balanced. This proof is based upon a specific case, in which the number of levels is three. The generalized conclusion is stated in the thesis and its corresponding proof is part of the future work. For a complete design, the situation is more complicated.

This thesis only considers the added number of runs being the least common multiple of the levels of the factors. Further research regarding the nature of this pattern is suggested.

This thesis defines standardized J_2-optimality so as to compare designs with different numbers of runs. The standardized J_2-optimality increases when more runs are added. In this thesis, this situation is interpreted, but the strict mathematical proof is still of interest. The lower bound of the J_2-optimality of balanced orthogonal mixed-level designs cannot be extended to unbalanced situations. However, because the lower bound is useful in determining the stopping criterion of the genetic algorithm, a general lower bound of J_2-optimality would be desirable.

Because designs with equal balance and orthogonality properties are not unique, it is desired to perform research on the criteria to compare these designs. The concept of the minimum aberration, proposed by Xu (2001), is valuable and could be used in finding the minimum abberation mixed-level designs.

For mixed-level designs with certain number of runs, it is impossible to generate balanced designs. In this situation, the balance coefficients of these designs, which reach their possibly most balanced situations, are not zero. In order to judge if a design is the most balanced design easily, it is necessary to perform more studies. Similarly, it is also desirable to judge the most orthogonal situation for mixed-level designs.

Further research may improve the efficiency of the genetic algorithms. Regarding the genetic algorithm, the problem of pre-mature convergence still needs to be improved. Genetic algorithms are not necessarily the only algorithms which can be used to perform this task. It is possible to use a new algorithm instead of genetic algorithms. Or, one could use genetic algorithm to get close to the optimal solution, and then use other algorithms to find the optimal solution.

Lastly, application and analysis of efficient mixed-level designs is a main area of research that holds great promise. This aspect of experimentation will be an important part of the future work.

APPENDIX

MATLAB SOURCE CODES

Main function

```
% function for balance mixed-level factorial design using GA
function Binary_GA(a, na, b, nb, c, nc, d, nd, e, ne, f, nf, g, ng, h, nh, i, ni, j, nj, small_size)


% original design is a full design matrix, which is determined by the factors and their levels
original_design=full_design_matrix(a, na, b, nb, c, nc, d, nd, e, ne, f, nf, g, ng, h, nh, i, ni, j, nj);

% lower bound for the small size design
l_b=lower_bound2(a, na, b, nb, c, nc, d, nd, e, ne, f, nf, g, ng, h, nh, i, ni, j, nj, small_size);

% optimum value of balance coefficient for a given design
o_b_c=optimum_balance_coef(original_design);


[m, n]=size(original_design); %m is the rows of the full design n is the number of factors

Lind=small_size;  %the size of the reduced design

Nind=2*small_size; %the number of chromosomes

% Creates initial population
% Population will be a Nind X Lind matrix
% Nind chromosomes with Lind genes aka decision variables

% [Chrom Lind BaseV] = crtbp(Nind, Lind, Base)

[Chrom Lind BaseV] = crtbp(Nind, Lind, m);

% Makes each gene vary from 1 to m
Chrom=Chrom+1;

% Maximum number of generations

MAXGEN = 1000;

% Reset counters
Best = NaN*ones(MAXGEN, 1);  % best in current population
gen = 0;     % Initialize generation

%Sort Chromosome so each row is in ascending order
Chrom = sort(Chrom, 2);

% Corrects each chromosome so there are no repeat rows
Chrom=correction(Chrom, BaseV);

% Calculate objective function for population
ObjVal = j2optimalitybc(Chrom, original_design);

% Generational loop
```

```
while gen < MAXGEN,

    % Save the best
    Best(gen+1,1) = min(ObjVal);

    % Fitness assignment to whole population
    FitnV = ranking(ObjVal);

    % Select individuals from population
    SelCh = select('sus', Chrom, FitnV);

    % Recombine selected individuals
    if gen<= 0.3*MAXGEN
        crorate=0.5;
    elseif gen > 0.3*MAXGEN
        crorate=1;
    end
    SelCh=recombin('xovsp', SelCh,crorate, 1);

    % Mutate offspring
    if gen<= 0.3*MAXGEN
        murate=0.4;
    elseif gen > 0.3*MAXGEN
        murate=0.001;
    end
    SelCh = mut(SelCh, murate, BaseV)+1;

    % New operation work on offspring
    if gen > 0.4*MAXGEN
        SelCh = my_operation(SelCh,Lind);
    end

    %Sort Chromosome so each row is in ascending order
    SelCh = sort(SelCh,2);

    % Corrects each chromosome so there are no repeat rows
    SelCh=correction(SelCh,BaseV);

    % Calculate objective function for offsprings
    ObjVOff = j2optimalitybc(SelCh,original_design);

    % Insert best offspring in population replacing worst parents
    [Chrom, ObjVal] = reins(Chrom, SelCh, 1, [1 .9], ObjVal, ObjVOff);

    % Increment generational counter
    gen=gen+1;

    % compute lower bound and balanced coefficient
    lb(gen)=l_b;
    obc(gen)=o_b_c;
end
% End of GA

% Find best solution in final population
i = find(ObjVal==Best(MAXGEN,1));
Best_Solution=Chrom(i,:);

% Plots the best per generation
figure(1)
plot(Best,'g-');
hold on
plot(lb,'b--');
plot(obc,'r-.');
hold off
title('Plot of the Best value vs iterations');
legend('Best','Lower bound for balanced','Balance coefficient');
xlabel('iterations');
ylabel('fitness values');
```

```
% output the solution design
[mm,nn]=size(original_design); % nn - number of columns for design
[r,k]=size(Best_Solution); % r - number of best solutions
                          % k - runs of small size design
solution_design(1:k,1:nn,1:r)=0;
for w=1:r
    for p=1:k
        solution_design(p,1:nn,w)=original_design(Best_Solution(w,p),1:nn);
    end
end

for tt=1:r
    solution_designs(:,:,tt)=sortrows(solution_design(:,:,tt));
end
solution_designs

if optimum_balance_coef(solution_design)==balance_coef(solution_design)
    text(2,20,'balance');
else
    text(2,20,'not balance, see values for references');
end
```

Random row generating

```
function rir=a_random_integer_row(a,na,b,nb,c,nc,d,nd,e,ne,f,nf,g,ng,h,nh,i,ni,j,nj)
n=na+nb+nc+nd+ne+nf+ng+nh+ni+nj;
for t=1:n
    if na>0 & t<=na
        new_row(1,t)=ceil(a*rand);
    elseif nb>0 & t>na & t<=na+nb
        new_row(1,t)=ceil(b*rand);
    elseif nc>0 & t>na+nb & t<=na+nb+nc
        new_row(1,t)=ceil(c*rand);
    elseif nd>0 & t>na+nb+nc & t<=na+nb+nc+nd
        new_row(1,t)=ceil(d*rand);
    elseif ne>0 & t>na+nb+nc+nd & t<=na+nb+nc+nd+ne
        new_row(1,t)=ceil(e*rand);
    elseif nf>0 & t>na+nb+nc+nd+ne & t<=na+nb+nc+nd+ne+nf
        new_row(1,t)=ceil(f*rand);
    elseif ng>0 & t>na+nb+nc+nd+ne+nf & t<=na+nb+nc+nd+ne+nf+ng
        new_row(1,t)=ceil(g*rand);
    elseif nh>0 & t>na+nb+nc+nd+ne+nf+ng & t<=na+nb+nc+nd+ne+nf+ng+nh
        new_row(1,t)=ceil(h*rand);
    elseif ni>0 & t>na+nb+nc+nd+ne+nf+ng+nh & t<=na+nb+nc+nd+ne+nf+ng+nh+ni
        new_row(1,t)=ceil(i*rand);
    elseif nj>0 & t>na+nb+nc+nd+ne+nf+ng+nh+ni & t<=na+nb+nc+nd+ne+nf+ng+nh+ni+nj
        new_row(1,t)=ceil(j*rand);
    end
end
rir=new_row;
```

Balance Coefficient

```
function b_c=balance_coef(a_design_matrix)
% n rows
% m columns
[n,m]=size(a_design_matrix);

% find the max level for each factor or column
for column_id = 1:m
    max_level(column_id)=max(a_design_matrix(:,column_id));
end


% number of level vv in column ww
for ww=1:m
    for vv=1:max_level(ww)
        number_of_each_level(vv,ww)=count(a_design_matrix(:,ww),vv);
    end
end

% balance coefficient for column j

ff(1:m)=0;
fff=0;

for jj=1:m
    for ii=1:max_level(jj)
        f(ii,jj)=number_of_each_level(ii,jj)/n;
    end
end

for j=1:m
    for i=1:max_level(j)
        ff(j)=ff(j)+f(i,j)^2;
    end
    fff=fff+ff(j);
end

b_c=fff;
```

Correction

```
function Chrom=correction(Chrom,BaseV)


[row, col]=size(Chrom);

for i=1:row
    for j=1:col-1
        if (Chrom(i,j)==Chrom(i,j+1))
            if (Chrom(i,j+1)==BaseV(:,j))
                Chrom(i,j+1)=BaseV(:,j);
                return
            end
            Chrom(i,j+1)=(Chrom(i,j+1)+1);
        end
    end
end
```

Count

```
function ct=count(matrix,number)
[m,n]=size(matrix);
ct=0;
for j=1:m
    for i=1:n
        if matrix(j,i)==number
            ct=ct+1;
        end
    end
end
```

Delta computation

```
function dta=deltaij(xi,xj)
dta=0;
n=length(xi);
for k=1:n
    if xi(k)==xj(k)
        dta=dta+1;
    end
end
```

Full design matrix generation

```
% na=0 and a=1 mean there is no this factor
function fdm=full_design_matrix(a,na,b,nb,c,nc,d,nd,e,ne,f,nf,g,ng,h,nh,i,ni,j,nj)
n=na+nb+nc+nd+ne+nf+ng+nh+ni+nj;
m=a*b*c*d*e*f*g*h*i*j;
for q=1:n
    fdm(1,q)=1;
end
p=2;
while (p <= m)
    new_row=a_random_integer_row(a,na,b,nb,c,nc,d,nd,e,ne,f,nf,g,ng,h,nh,i,ni,j,nj);
    repeat_value=0;
    for r=1:p-1
        if isequal(new_row,fdm(r,:))==0     %new_row(1,:)~=fdm(r,:)
            repeat_value=repeat_value+1;
        else
            repeat_value=repeat_value;
        end
    end
    if repeat_value==p-1
        fdm(p,:)=new_row;
        p=p+1;
    elseif repeat_value < p-1
        p=p;
    end
end
fdm=sortrows(fdm);
```

Objective function

```
% a sub function used to calculate the j2 optimality value
% oritinal_design is the full size design
% will return j2 optimality value and the delta matrix


function j2p=j2optimalitybc(Chrom,original_design)

[mm,nn]=size(original_design);
[r,k]=size(Chrom);
d(1:k,1:nn,1:r)=0;
for w=1:r
    for p=1:k
        d(p,1:nn,w)=original_design(Chrom(w,p),1:nn);
    end
end

for t=1:r
    [n,m]=size(d(:,:,t));
    j2p(t)=0;
    for i=1:n
        for j=1:n
            if i < j
                j2p(t)=j2p(t)+(deltaij(d(i,:,t),d(j,:,t)))^2;
            end
        end
    end
end

a=10;  % a is the adjusting coefficient for balance property related to j2optimality

for tt=1:r
    j2p(tt)=j2p(tt)+a*(balance_coef(d(:,:,tt))-optimum_balance_coef(d(:,:,tt)));
%     j2p(tt)=balance_coef(d(:,:,tt));
end

j2p=j2p';
```

Lower bound of J_2-optimality of balanced design matrix

```
function ln=lower_bound2(a,na,b,nb,c,nc,d,nd,e,ne,f,nf,g,ng,h,nh,i,ni,j,nj,expected_number_runs)

% Number of column
m=na+nb+nc+nd+ne+nf+ng+nh+ni+nj;
% Number of rows for full design
n=a*b*c*d*e*f*g*h*i*j;
% Number of rows for small design
nn=expected_number_runs;

a_design_matrix=full_design_matrix(a,na,b,nb,c,nc,d,nd,e,ne,f,nf,g,ng,h,nh,i,ni,j,nj);

% find the max level for each factor or column
for column_id = 1:m
    max_level(column_id)=max(a_design_matrix(:,column_id));
end

sumone=0;
for i=1:m
    sumone=sumone+nn/max_level(i);
end


sumtwo=0;
for j=1:m
    sumtwo=sumtwo+((max_level(j)-1)*(nn/max_level(j))^2);
end

sumthree=0;
for k=1:m
    sumthree=sumthree+1;
end

ln=(sumone*sumone+sumtwo-nn*sumthree*sumthree)/2;
```

New operation

```
function newchrom=my_operation(oldchrom,Lind)
[m,n]=size(oldchrom);
newchrom=oldchrom;
p=ceil(m*rand);
q=ceil(n*rand);
if rand < 1
    newchrom(p,q)=ceil(Lind*rand);
end
```

Optimum balanced coefficient

```
function o_b_c=optimum_balance_coef(a_design_matrix)
% n rows
% m columns
[n,m]=size(a_design_matrix);
% lower_n=expected_number_runs;

% find the max level for each factor or column
for column_id = 1:m
    max_level(column_id)=max(a_design_matrix(:,column_id));
end

%
% % number of level vv in column ww
% for ww=1:m
%     for vv=1:max_level(ww)
%         number_of_each_level(vv,ww)=count(a_design_matrix(:,ww),vv);
%     end
% end

% balance coefficient for column j

ff(1:m)=0;
fff=0;

% for jj=1:m
%     for ii=1:max_level(jj)
%         f(ii,jj)=number_of_each_level(ii,jj)/n;
%     end
% end

for j=1:m
    for i=1:max_level(j)
        ff(j)=ff(j)+(1/max_level(j))^2;
    end
    fff=fff+ff(j);
end

o_b_c=fff;
```

REFERENCES

Addelman, S. (1962), "Orthogonal Main-Effect Plans for Asymmetrical Factorial Experiments" *Technometrices*, 4, 21-46.

Ankenman, B.E. (1999). "Design of Experiments with Two- and Four-Level Factors", *Journal of Quality Technology*, 31, 363-375.

Bashir, A. (2003), "A New Approach for Constructing and Analyzing Supersaturated Designs for Experiments Involving Numerous Factors", Dissertation (Unpublished).

Borkowski, J.J. (2003). "Using a Genetic Algorithm to Genetic Small Exact Response Surface Designs" *Journal of Probability and Statistical Science*, 1,65-88.

Chipperfield, Fleming, Pohlheim and Fonseca (1994). Genetic Algorithm TOOLBOX For Use with MATLAB User's Guide, Version 1.2, http://www.shef.ac.uk/~gaipp/ga-toolbox/manual.pdf

Cox, D.R. and Reid, N. (2000), *The Theory of the Design of Experiments*, Chapman & Hall/CRC, New York

DeCock, D and Stufken, J (2000), "On Finding Mixed Orthogonal Designs of Strength 2 with Many 2-level Factors", *Statistics & Probability Letters*, 50, 383-388.

Fang, K., Lin, D and Ma, C. (2000), "On the Construction of Multi-level Supersaturated Designs" *Journal of Statistical Planning and Inference*, 86, 239-252.

Fries, A. and Hunter, W.G. (1980), "Minimum aberration 2^{k-p} designs*", Technometrics*, 22, 601-608.

Gen, M and Cheng, R. (2000). *Genetic Algorithm & Engineering Optimization*, John Wiley & Sons, Inc. New York, NY.

Hedayat, A.S., Sloane, N.J.A. and Stufken, J. (1999), *Orthogonal Designs Theory and Applications*. Springer-Verlag, New York.

Montgomery, D.C.(2001). *Design and Analysis of Experiments*, 5th ed., John Wiley & Sons, Inc. New York, NY.

Myers, R.H. and Montgomery, D.C. (2002). *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 2nd ed., John Wiely & Sons, Inc. New York, NY.

Nguyen, N. (1996), "A note on the Construction of Near-Orthogonal Designs with Mixed Levels and Economic Run Size" *Technometrices*, 38, 279-283.

Ortiz.J.F., Simpson, J. R. and Pignatiello, J.J. (2003), "A Genetic Algorithm Approach to Multiple Response Optimization", submitted to *Journal of Quality Technology*.

Rao, C. R. (1947), "Factorial Experiments Derivable from Combinatorial Arrangements of Designs"*, Journal of Royal Statistical Society*, Supplement, 9, 128-139.

Sloane, N. J. A., *A Library of Orthogonal Designs*, www.research.att.com/~njas/oadir/index.html

Wang,J.C., and Wu,C.F.J. (1991), "An Approach to the Construction of Asymmetrical Orthogonal Designs " *Journal of the American Statistical Association*.,86, 450-456.

Wang,J.C., and Wu,C.F.J. (1992), "Nearly Orthogonal Designs with Mixed Levels and Small Runs" *Technometrices*, 34, 409-422.

Wang. J.C. (1996), "Mixed Difference Matrices and the Construction of Orthogonal Designs"*, Statistics & Probability Letters*, 28, 121-126.

Wu, C.F.J., and Hamada, M.S. (2000), *Experiments: Planning, Analysis and Parameter Design Optimization*, New York: Wiley.

Wu, C.F.J., and Zhang, R. (1993) "Minimum aberration designs with two-level and four-level factors", *Biometrika*, 80, 203-209.

Xu, H (2001), "Generalized Minimum Aberration for Asymmetrical Fractional Factorial Designs" *The Annals of Statistics*, 29, 1066-1077

Xu, H. (2002), “An Algorithm for Constructing Orthogonal and Nearly-Orthogonal Designs with Mixed Levels and Small Runs” *Technometrices*, 44,356-368.

BIOGRAPHICAL SKETCH

Yong Guo began exploring the field of industrial engineering during his sophomore year of college. He graduated from Xi'an Jiao Tong University in four years with Bachelor of Science degrees in both Industrial Engineering and Mechanical Engineering. After gaining experience in industry for one and one-half years, he entered the master's degree program in industrial engineering at the FAMU-FSU College of Engineering with a specialty in quality engineering.