2-1-2007

Generalized construction of trend resistent 2-level split-plot designs

Guillermo Lopez

GENERALIZED CONSTRUCTION OF TREND RESISTANT 2-LEVEL SPLIT-PLOT DESIGNS

BY
GUILLERMO LOPEZ

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE MASTER OF SCIENCE DEGREE IN
INDUSTRIAL AND SYSTEMS ENGINEERING IN
KATE GLEASON COLLEGE OF ENGINEERING OF
ROCHESTER INSTITUTE OF TECHNOLOGY

INDUSTRIAL AND SYSTEMS ENGINEERING DEPARTMENT
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

FEBRUARY 2007

COMMITTEE MEMBERS

DR. ANDRES CARRANO
ASSOCIATE PROFESSOR
INDUSTRIAL AND SYSTEMS ENGINEERING
ROCHESTER INSTITUTE OF TECHNOLOGY

DR. BRIAN THORN
ASSOCIATE PROFESSOR
INDUSTRIAL AND SYSTEMS ENGINEERING
ROCHESTER INSTITUTE OF TECHNOLOGY

KATE GLEASON COLLEGE OF ENGINEERING
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

MASTER OF SCIENCE DEGREE THESIS

The M.S. Degree Thesis of Guillermo T. Lopez Perez
has been examined and approved by the thesis committee
as satisfactory for the thesis requirement for the
Master of Science degree

Dr. Andres Carrano, Ph.D. *Advisor*

Dr. Brian Thorn, Ph.D. *Advisor*

TO MY FATHER,

WHO HAS GIVEN ME

THE KNOWLEDGE AND SUPPORT

TO ACHIEVE MY GOALS.

Abstract

Common experimental practices suggest randomizing the order in which runs are performed. However, there may be situations in which randomization might not produce the most desirable order, especially in the presence of known trends. There has been research done on systematically designing experiments to be robust against trends. However, few studies address the additional dimensions that arise in nested designs such as split-plot designs. Split-plot designs have been used for many years in agricultural applications and are sometimes preferred where there are hard-to-change factors in industrial settings. There currently is no established methodology to produce split-plot designs that are robust to potential two-dimensional trends. The objective of this work is to develop a methodology to design run orders for two-level, split-plot ($2^w \times 2^s$) designs that are robust or nearly robust against a set of trends. Two methods are developed in this work. A fold-over method that uses already established principles is extended for use in split-plot designs. The second method uses an integer linear programming approach to search for an optimal design that is resistant to specific trends. A comparison between the two methods is presented and evaluated with a proposed set of metrics.

Keywords: Split-Plot Experimental Designs, Trend Resistant Run Order, Fold-over, Integer Linear Programming.

Table of Contents

List of Figures

List of Tables

Terminology

SPD = Split-Plot Design

DOE = Design of experiments

w = number of factors at the whole-plot level

s = number of factors at the sub-plot level

α = denotes the numeric representation of the factor

$\mathbf{D}^{\alpha}$ = matrix design for the contrast of factor α, size: $2^w \times 2^s$

$\boldsymbol{\tau}_{\phi}$ = matrix model for the trend, size: $2^w \times 2^s$

ϕ = denotes a specific trend interaction

t = number of trends that the design should be robust against.

N_{α} = the total combinatory possibilities for the sub-plot factor α

i = position row i on the matrix

j = position column j on the matrix

k = the treatment level combination (TLC)

TI = Trend Index measurement: $TI_{\phi}^{\alpha} = \sum_{i=1}^{2^w} \sum_{j=1}^{2^s} D_{i,j}^{\alpha} \times \tau_{i,j,\phi}$

$c_{i,j}$ = is the contrast level for a factors in row i and column j

$\bar{c}_{i,j} = -c_{i,j}$

$V_{\alpha k}$ = the value of the contrast of factor α in permutation k

$$X_{ijk} = \begin{cases} 1, \text{ if TLC } k \text{ is used in row } i \text{ and column } j. \\ 0, \text{ otherwise.} \end{cases}$$

$s_{\phi,\alpha}^{+}$ $\equiv$ positive magnitude the TI of trend ϕ and factor α

$s_{\phi,\alpha}^{-}$ $\equiv$ negative magnitude the TI of trend ϕ and factor α

1 Introduction

Experimenters are advised to randomize experiments so that the order of the runs is not biased; however, there are some situations where it is impossible or simply not advisable to completely randomize an experiment. Such is the case with blocked, Latin-squares, nested, and split-plot designs, among others. These situations are called restricted randomized experiments. These designs impose a restriction that prevents them from being completely randomized. These restrictions require the experiment to be analyzed with other methods to provide a significant conclusion.

Sometimes, randomizing the experiment will not produce the most adequate designs. There is no control over the design when it is done randomly and the sequence of runs might fall in an undesirable order. On occasions, researchers might be concerned with possible trends that might affect the results due to an unsatisfying order. Learning curves, wear and tear, and time-correlated trends are some examples of potentially damaging trends in experiments. When there are suspicions that a potential trend may corrupt an experiment, it might be more convenient if the order of the experimental runs is pre-selected. This will obviously remove the benefits of randomizing the experiments, but will also reduce the potential for a more damaging trend effect in the results. For this purpose there are several techniques and strategies that have been studied for the past decades.

Split-plot designs have been used for many years in industrial applications. Split-plot experiments are used when there are factors that are hard to change or too expensive

to change from one treatment over to another. Factors like temperature settings, machining tool settings, land use or any other factor that requires a large amount of time to change are considered hard to change factors.

In completely randomized experiments, every treatment combination is positioned in the experiment in a random order. A split-plot experiment, on the other hand, will change the hard to change factors less frequently; thus, reducing the time it takes to perform the experiment.

In the case of split-plot designs, the restriction on the randomization produces a two-dimensional experiment that could be represented in a row and column procedure where every treatment in the sub-plots is conducted before changing the treatments in the whole-plots. There may be potential trend effects across both dimensions; therefore, the trends that are considered in this thesis are two-dimensional trends. These trends are composed of a row × column interaction effect between the trends.

This paper will study methods for optimizing a split-plot design to be robust to various two-dimensional trends. As the experiments become larger, the possible designs available grow in exponential fashion. If the computer applications need to calculate every possible outcome, these results can take a large amount of time.

This thesis will concentrate on designing 2^k split-plot experiments such that they are resistant to potential polynomial trends. Furthermore, a fold-over method and an integer linear programming method will be studied as methods for achieving robust split-plot experiments.

2 Problem Statement

Past research on trend resistance has not addressed trends affecting split-plot experiments. Split-plot experiments robust to trends can be similar to block designs, but the research done on block designs has only addressed one-dimensional trends while split-plot designs could potentially be affected by two or more independent trends at the same time. These independent trends can interact with each other to create a two-dimensional trend. Unfortunately, very little work has been done on designs affected by two-dimensional trends.

Many industries today have constraints when performing experiments. Some experiments tend to be very large and costly. Split-plot designs can considerably reduce the time of performing an experiment, which in turn will reduce its cost. Past research suggests that it is better to conduct a split-plot experiment when there are hard-to-change factors (Lucas and Ju, 1992, and Arvidsson and Gremyr, 2003).

The purpose for this thesis is to design a method for developing 2-level split-plot experiments that are resistant to potential trends. The possible trends will be identified, including linear, polynomial, exponential, sinusoidal, and other non-polynomial trends. Using computer applications such as MATLAB and CPLEX, several designs can be examined to understand the influence of these trends in the design.

3 Background

3.1 Design of Experiments

The purpose of experimentation is to understand the relationship between input and output variables. By modifying the inputs and recording the changes in the response, the experimenter can identify what inputs are influential in the response.

Design of Experiments is a set of procedures that are commonly used to conduct these experiments. Many statisticians have developed methods to conduct experiments and statistically identify the factors and interactions that are influential in the responses (Coleman and Montgomery, 1993). One of the more popular methods is called the Analysis of Variance (ANOVA). The ANOVA is used to identify which factors produce the greatest changes in the response.

Montgomery describes the single factor statistical model in the book *Design and Analysis of Experiments (2000)* as:

$$y_{ij} = \mu + \tau_i + \varepsilon_{ij} \begin{cases} i = 1, 2, \ldots, \mathrm{a} \\ i = 1, 2, \ldots, \mathrm{n} \end{cases}$$

where y_{ij} is the *ij*th observation, μ is the overall mean common to all treatments, τ_i is the *i*th treatment effect, ε_{ij} is a random error component that incorporates all other sources of variability in the experiment.

This statistical model assumes that the error term, ε_{ij}, has a normal distribution with constant mean of 0 and constant variance. Because this error term is desired to be

unbiased, the experiment is advised to be designed in a random order. The randomness of the experiment will reduce the noise effects in the response.

There is a consideration with the size. When perfoming an experiment, one of the biggest limitations is cost. In general, the larger the size of the experiment the more costly it can be. Experimenters always try to reduce the size of the experiment. Running a fraction of the total number of runs is a method used to reduce the size of the factorial designs. These designs are called fractional factorial designs. However, this limits and confounds some responses and the experimenter should take into consideration what was lost.

On occasions, an experiment could be constrained by time, size, costs, or other factors. These restrictions produce several situations that require advanced experimental designs. If the experiment is designed appropriately and every restriction, constraint and outside influence is taken into consideration in the experiment, then the experiment might yield more reliable results.

3.2 Trends

Sometimes, randomizing the run order of an experiment might yield an undesirable order, especially in the presence of a trend. A trend is an uncontrolled variable that is highly correlated with the experiment, such as learning curves and the passing of time. This is commonly known as a time trend (Hill, 1960, Steinberg, 1988, John, 1990, etc), but for the purpose of this thesis, the time trend will be addressed simply as a trend. Randomizing will help guard against noise variables. Trends, on the other hand, may not have a constant mean of 0 and some might not have a constant variance. Therefore, they can cause an undesirable effect in the response.

When experimenters anticipate that there may be uncontrolled variables in their experiments, they could choose to ignore them, block them, include them in the experiment, or conduct the experiment in an order in which these variables do not affect the results. To be resistant to a particular trend, the experiment should be designed so that it is orthogonal to the trend.

There are several well-established representations of the trends including that of a vector with the value of the trend at that position or sequence in the experiment: (1, 2, 3, 4, …, i). This model is referred to as a linear trend because it follows a linear pattern. Linear trends can be modeled with the following formula: $y = ax + b$. Daniel and Wilcoxon (1966) used a linear polynomial for a 2^4 factorial experiment. In this work, the trend was modeled as (-15, -13, -11, …, -1, +1, …, +13, +15).

Most experimenters tend to be concerned with linear trends and sometimes with quadratic trends, but typically not higher. If the contrast and the trend are orthogonal, then the contrast is considered trend free.

It is known that a contrast with coefficients $(\alpha_1, \alpha_2, ..., \alpha_n)$ is orthogonal to a trend of the kth-order if the response is not correlated, the variance is the same, and they meet the following condition:

$$\sum_{i=1}^{n} \alpha_i i^k = 0,$$

for n experimental runs, where k is the order of the trend. Considering the following example: a 2^3 factorial experiment, whose run order is $[(1) \quad a \quad b \quad ab \quad c \quad ac \quad bc \quad abc]$ and a linear trend modeled as: $[1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8]$ is present, the run order will not satisfy the rule: $\sum_{i=1}^{n} \alpha_i i^k = 0$

for any factor. In fact, it will be equal to 4, 8 and 16 for factors A, B and C respectively. However, if the run order is changed to $[c \quad a \quad b \quad abc \quad ab \quad bc \quad ac \quad (1)]$, then all factors will satisfy $\sum_{i=1}^{n} \alpha_i i^k = 0$. The latter design is considered to be robust against the trend in question.

Most research in this field considers only polynomial trends. There is a possibility that a trend might be best modeled as a non-polynomial (Atkinson and Donev, 1996). In some cases, such as weather patterns, the trend might be modeled as a sinusoidal model, or an exponential model for elements with a very short half-life or decay function.

3.2.1 Polynomial Trends

A polynomial trend would be modeled by the general equation: $y = a_p x^p + a_{p-1} x^{p-1} + ... + a_2 x^2 + a_1 x^1 + a_0$ where *x* is the position of the design point and *y* is the effect of the trend. Also, *p* denotes the degree of the polynomial.

Experimenters have studied the effects of polynomial trends and the consequences of the degree of the trend. Linear and quadratic models (p=1 and p=2) are of most concern among experimenters. Higher order trends might also be of some concern if found present in the experiment.

There are many other possible trends that could be modeled as a polynomial. Positional trends, learning curves and tool wear are common trends that an experimenter may wish to model.

3.2.2 Sinusoidal Trends

Sometimes, the trend of concern might be difficult to model as a polynomial. Any trend that is suspected to have a cyclical variation is probably best modeled as a sinusoidal model. These models follow the general sinusoidal equation: $y = a\sin\theta + a_0$ where θ is the radian position of the design point and y is the effect of the trend. Experiments, especially those that take an extremely large time to conduct, could have some seasonal changes or uncontrolled temperature fluctuations.

Some possible trends that can be modeled with a sinusoid are seasonal changes, temperature fluctuations and vibrations.

3.2.3 Exponential Trends

Exponential trends may also be of concern. Wear and decay of materials used, radioactive half-life, or population changes that could affect certain experiments might have some effect on the design. These trends can be modeled by variation of the equation: $y = a_p^{p(x+\beta)} + a_0$ where x is the position of the design point and y is the effect of the trend, and p denotes the degree of the model.

3.3 Restricted Randomized Experiments

There are situations in which an experiment cannot be completely randomized. This can be due to limited amounts of space or time. Common restrictions can include experiments that need to be run at the same time, or when the experimental runs use several batches of materials, or when there are different operators or equipment for a set of runs. These experiments are often called Restricted Randomized Experiments. These

types of experiment offer valid results for certain restrictions while sacrificing other information.

By completely randomizing the experiment, the potential effects of noise and other uncontrolled variables are lessened in the experiment. This advantage may be discarded because there are other more important considerations in the experiment. It is common to see an experiment that can be conducted as a completely randomized experiment, but the experimenter chooses to purposely restrict the randomization so as to obtain other valuable information or to arrange for a less complicated experiment.

Common restricted randomized experiments are blocked, Latin-squares, nested and split-plot designs. The blocked designs are used when there are different environmental or operational conditions between groups of runs in the experiment, such as: two operators each conducting a part of the experiment or an experiment that takes so long that parts of the experiment are conducted on different days. It is true that the experiment can be designed by including the operators as factors in the experiment, thus reducing this problem; however, the response due to operators in this situation is either unimportant or fairly obvious. Besides, adding the operators as a new factor, will double the size of the experiment.

Latin-squares designs are used to block multifactor designs in two directions. They are mostly used as a mean to reduce the size of very large experiments with factors that have large number of levels. Latin-squares have very good estimates for the main effects. There are several variations to the Latin-squares such as incomplete Latin-squares and Graeco-Latin-squares.

Nested designs are used when the factors are in a hierarchy, and the levels of some factors are nested under the levels of other factors. For example, if an experimenter wishes to compare different cooking styles from different chefs, each chef uses several factors that will be considered in the experiment. The problem is that the chefs use different ingredients and utensils rather than having all the chefs with the same factors. This difference can be influential in the experiment and cannot be conducted as a completely randomized experiment, because it might not yield a valid response.

Split-plot is a special case of nested designs and originated in agriculture, when experimenters wished to perform experiments on the crops. This caused problems because crops are seasonal and had to be conducted over several years. To avoid this, the land was separated into several plots and the experimental runs were all conducted at the same time. This technique allowed experimenters to run the entire experiment in one season, preventing uncontrolled climatic variables to affect the results. The following section presents a more detailed explanation of split-plot experiments.

3.4 Split Plot Designs:

Split-plot experiments or SPDs are usually conducted when there are "hard to change" factors included in the experiment. Hard to change factors are those that, when compared to the other factors in the experiment, will take a longer time to change from one setting to another or are just too expensive or resource intensive to change in a random matter. Temperature changes, complicated and time consuming machine settings, large batches of materials, plots of croplands, among others are some examples of common hard to change factors used in split-plot designs.

Table 3.1 illustrates a cake baking experiment as an example. The purpose of this experiment is to test how much a cake would rise with different ingredients and at specific oven temperatures. In this example, there are three types of baking mixtures that combine different ingredients at different proportions. The mixtures are prepared and baked at four different temperature levels in the oven. Three replicates are run under every treatment combination of temperature and ingredients. This experiment can be completely randomized by randomly selecting the temperature and the ingredient mix and performing the experiment at these settings. Afterwards, a new temperature and mixture is randomly selected, excluding the ones already performed, and the experiment is run at the new settings.

When hard to change factors are involved, like the temperature in the oven, randomly selecting the temperature will make this experiment too expensive. Changing the temperature of the oven on every run of the experiment will make this experiment too

Table 3.1: Cake baking experiment

	Baking Ingredients								
	Type A			Type B			Type C		
Oven Temperature (℉)	I	II	III	I	II	III	I	II	III
350	X_{111}	X_{112}	X_{113}	X_{121}	X_{122}	X_{123}	X_{131}	X_{132}	X_{133}
400	X_{211}	X_{212}	X_{213}	X_{221}	X_{222}	X_{223}	X_{231}	X_{232}	X_{233}
450	X_{311}	X_{312}	X_{313}	X_{321}	X_{322}	X_{323}	X_{331}	X_{312}	X_{333}
500	X_{411}	X_{412}	X_{413}	X_{421}	X_{422}	X_{423}	X_{431}	X_{432}	X_{433}

Note: X_{ijk} represents data points where *i* is the oven temperature, *j* is the baking ingredient, and *k* is the replicate.

expensive or impractical. The most convenient and logical way to perform such an experiment is to set the oven at one temperature (randomly selected) and perform all the other settings either at the same time inside the oven or one at a time but with the same temperature setting. For example, the temperature of the oven is heated to 350°F and three replicates of three baking ingredients are prepared. Then all nine mixtures are

introduced in the oven and cooked at 350°F. After the cakes are baked, the temperature in the oven is changed to a new setting (i.e. 400°F) and 9 new mixtures are introduced into the oven. This process is repeated until all temperatures have been tested. The temperature factor, with its 9 data points, is referred to as the whole-plots, while the ingredients are referred to as split-plots or sub-plots.

SPDs originated when experimenters had to perform experiments on crops where they could not use the same plot of land for the experimental contrasts. Currently, experiments on croplands are designed as split-plot experiments or Latin-squares.

Split-plots have their limitations. Since the hard-to-change factors and the whole-plots are the same, any condition that changes from one plot to the next may generate as a temperature effect. This arrangement will have the main effects confounded with the plots. There is also an extra error term that is included in the whole plots. An appropriate model for this experiment can be:

$$Y_{ijk} = \mu + R_k + T_i + TR_{ik} + B_j + BR_{jk} + TB_{ij} + TBR_{ijk} + \varepsilon_{ijk}$$

where R is the replicate, T is the temperature, TR is the Replicate × Temperature interaction, B is the baking ingredients, BR is the Replicate × Baking Ingredients interaction, TB is the Temperature × Baking Ingredients interaction, TBR is the three-way interaction, and ε is the error term. The TR interaction is considered the whole-plot error.

3.5 Trends on Two-level Split-Plot Designs

It has come to the attention of some researchers (Edmondson, 1993, Carrano *et al*, 2002, and Carrano and Thorn, 2004) that when trends are present in split-plot

experiments, the results might be impacted by a composite, two-dimensional trend effect. Edmondson called these designs row $\times$ column designs and that these could be affected by row $\times$ column trend interaction. Edmondson (1993) concentrated on Latin-squares while Carrano *et al* (2002, 2007) observed a specific split-plot experiment.

Carrano *et al* (2002) performed a split-plot experiment to determine the effect of a set of process parameters on the roughness of the wood surface when it is sanded. The run order of this experiment was systematically planned to be trend resistant. The input of the experiments were as follows: wood species, grit size, depth of cut, tooling resilience, feed rate, spindle speed, and grain orientation. Each factor had 2 levels except for the wood species, which had 3 levels. The size of this experiment was $3^1 \times 2^6 = 192$ treatment combinations. Changing the parameters randomly between each experimental unit would have been cost intensive and time consuming. It was decided to perform a split plot experiment, where the machine parameters were set to be the whole-plot factors and the wood setup will be the sub-plot factors. In the sub-plot, the wood had $3^1 \times 2^1 = 6$ units (species $\times$ orientation). Then the spindle passed on all 6 units. The machine factors are varied between runs for a total of $2^5 = 32$ experimental runs.

This arrangement of the split-plot allowed the reduction of cost and time when the experiment was performed. There was, however, a concern for the wear of the tool and particular learning curves when setting up each experiment. It was noted that if they were to be resistant to these trends, the experimental runs needed to be ordered appropriately. Carrano et al (2004, 2007) showed the appropriate order for this experiment. They identified a possible trend across the sub-plots. This trend was the wear of the tool, and

they assumed the presence of a trend across the whole-plots. This meant that each sub-plot trend would be different among the other whole-plots.

In this work, the researchers used the available methods in literature on trend resistance to find an optimal run-order. Since there is no other research available that questions the two-dimensional trend, their work establishes the foundation for a general construction for trend-resistant split-plot designs.

The following thesis particularly addresses the generalization of the work Carrano et al (2007) performed by finding appropriate designs for two-level trend resistant split-plot experiments.

4 Literature Review

4.1 DOE and Nested Designs

When performing an experiment, it is desirable to randomize the order in which the experiment is conducted. There are, however, situations in which a complete randomization of the experiment is not feasible or recommended. Costs and time factors can constrain the experiment making it difficult to completely randomize the experiment. Whenever the experimenter is unknowledgeable about the experiment, this could lead to unreliable analysis of the experiment (Coleman and Montgomery, 1993). Techniques for coping with restrictions to randomization include blocked designs, nested designs and split-plot designs. Arvidsson and Gremyr (2003) compared how the results are affected if the experiment is done by completely randomizing the experiment or by deliberately restricting the runs.

Experimenters routinely encounter experiments in which one or more factors are very hard or very costly to change. In these cases, it might be more convenient for the experiment to be analyzed as a split-plot design, which accounts for hard-to-change factors and easy-to-change factors. A split-plot design is a form of restricting the randomization of the experiment. Lucas and Ju (1992) point out that analyzing experiments that contain hard to change factors as if they did not, would not yield the most desirable results. This is because experiments that have hard-to-change factors often include two error terms. They state that the error associated with the hard-to-change

factors, could mask the effects of the easy-to-change factors. Therefore, these experiments should be performed as a split-plot experiment.

4.2 Experimental Designs with Trends

It is possible for experiments to be affected by trend effects. In these cases, when the experimental units are selected at random, the trends can have undesirable effects in the results. Cox (1951) discussed the presence of a possible trend in randomized experiments. It is possible to design the experiment in such a way that the trend effects are orthogonal to the effects of interest in the experiments. Cox (1951, 1952) produced design plans that deal with first order trends. Daniel and Wilcoxon (1966) discuss why experiments should be resistant to trends and suggest a few designs as well. The consequences of ordering the design points in an experiment have been studied. Atkinson and Donev (1996) concluded that very little information is lost when the experiment is designed to be resistant against trends when in fact there is no trend present. Thus, when the experimenter assumes that there is a possible trend that might affect the result, it may be convenient if the experiment is robust to the trend. Whether the trend exists or not, may be irrelevant.

There have been several works that have considered trends when designing experiments. Daniel and Wilcoxon (1966) have developed plans where some main effects are robust against quadratic trends. Cheng and Jacroux (1988) developed an algorithm that generates trend free runs and gives estimates of main and two-factor interaction effects protected against high-degree polynomial trend effects. Jacroux and Ray (1990) give a method that constructs the order of experiments that contain v treatments and n runs over time or space that could contain an unknown trend effect.

John (1990) applies the principle of folding over on the experimental runs to generate runs that are resistant to linear and quadratic time trends. Other works have been performed on different forms of experiments: Cheng and Steingberg (1991), Bailey *et al* (1992), Coster (1993), Jacroux (1994) refer to factorial experiments, Daniel and Wilcoxon (1966) consider fractional factorial designs, while Jacroux (1996), Steinberg (1988), and Edmonson (1993) study the effects of the trend itself.

4.3 Split-plot designs and trends

Experimenters have developed techniques to design experiments that are resistant to possible high-order trend effects; however, little has been done on designing trend resistant split-plot or other nested designs. Goos and Vandebroek (2004) mention that these designs naturally give better protection against trends but they do not address the effects of a trend on these designs.

Blocked designs with trends have received lots of attention in the past twenty years which originated with Bradley and Yeh (1980). They discuss trend resistance in blocked designs and why is it important to be resistant to possible trends. Yeh and Bradley (1983, 1985) have developed methods for constructing trend-free and nearly trend-free block designs. Stufken (1988) studied trend-free block designs, and concluded that there are some criteria that will prevent an experiment from being trend-free. In this paper, Stufken mentions the parameters that a block design should have to be resistant to possible trends. Other works concerning block designs include: Lin and Dean (1991), Chai and Stufken (1999), Lin and Stufken (1999, 2002), and Tack and Vandebroek (2002).

Box and Jones (1992) compare methods of performing experiments when some environmental condition arises that can affect the results. They compare completely randomized, split-plot and strip-block experiments. They have concluded that split plot designs that take these environmental issues into account are very reliable. The environmental conditions used by Box and Jones are known effects and were introduced into the experiment as hard-to-change factors, but trends may have unknown effects on the factors and cannot be modeled like this experiment. Kowalski (2002) is concerned with split-plot experiments robust to parameter designs and uses a form of semi-folding to generate 24 run experiments for this purpose.

Carrano *et al* (2002) performed a split-plot experiment on wood machinery where the settings for the machine were set as the whole-plot factors and the type of wood and grain orientation were set as the sub-plot factors. Suspecting that there could be a possible unknown trend effect due to time and position, the experiment was designed to be robust against two linear trend effects. By combining the technique used by Daniel and Wilcoxon (1966), the principle of folding over and a nonlinear integer program, they developed a feasible design that would be simultaneously resistant to two linear trends, one for the whole-plots and the other for the sub-plots.

For any split-plot design the experimenter may have several factors that are hard to change at a whole-plot level, and some factors that are easy to change, and are located in a sub-plot level. There is also the potential existence of individual trends, some for the hard to change factors and others to the easy to change factors. Edmondson (1993) considered that row-and-column designs are affected by a row $\times$ column interaction effect. Given that split-plot designs are a form of row-and-column designs, it is possible

to assume that they could be affected by a two-dimensional trend. Atkinson and Donev (1996) mentioned that the trend might not be accurately modeled as a polynomial. Depending on the environmental conditions, the trend might be better represented as an exponential or sinusoidal model instead. There has been little work that considers these trends. Steinberg (1988) modeled some time trends as an autoregressive integrated moving average (ARIMA) time series and shows how this can be helpful in factorial experiments. Still, works on exponential and sinusoidal models are missing.

Split-plot experiments are widely used in industrial environments. Protecting against possible trends will enable these experiments to give unbiased results. Most research on similar experiments only considers trends modeled by a one-dimensional array across all runs. The purpose of this work is to propose a method to design split-plot experiments that are resistant to two-dimensional trends. As shown in this section, this has not been addressed in literature before.

5 Methodology

5.1 The Split-Plot Model

In order to simplify calculations when designing a trend resistant split-plot experiment, a matrix format with rows and columns is followed in this work. The rows will represent the whole-plot treatments and the sub-plot treatments will be arranged along the columns. Each cell in the matrix contains the treatment level combination of all the factors in that run of the experiment. Figure 5.1 shows how the runs are arranged in a split-plot experiment. The experimenter will set the whole-plots at the desired level in the *i*th row and run accordingly all the sub-plot runs within that row across the columns. The split-plot design has a $2^w \times 2^s$ size (i.e. number of treatment level combinations) where w is the number of factors at the whole-plot level and s is the number of factors at the sub-plot level. The matrix D^{α} is defined such that it will have 2^w rows and 2^s columns and represents the contrast matrix of factor α. Each unit in D^{α} will represent the level of factor α at that point (i.e. treatment) in the experiment. The way split-plot designs are typically conducted, the first experimental unit will be the treatment level on the 1^{st} column in the 1^{st} row. The second treatment will be on the 2^{nd} column in the 1^{st} row, and so forth until there are no more experiments to be run in that 1^{st} row. This completes all the runs for the whole-plot represented in the first row. The next run will begin in the 1^{st} column on the 2^{nd} row, and so on until all rows and columns have been done. To better illustrate these notations, the following injection-molding experiment will be used through the rest of the paper.

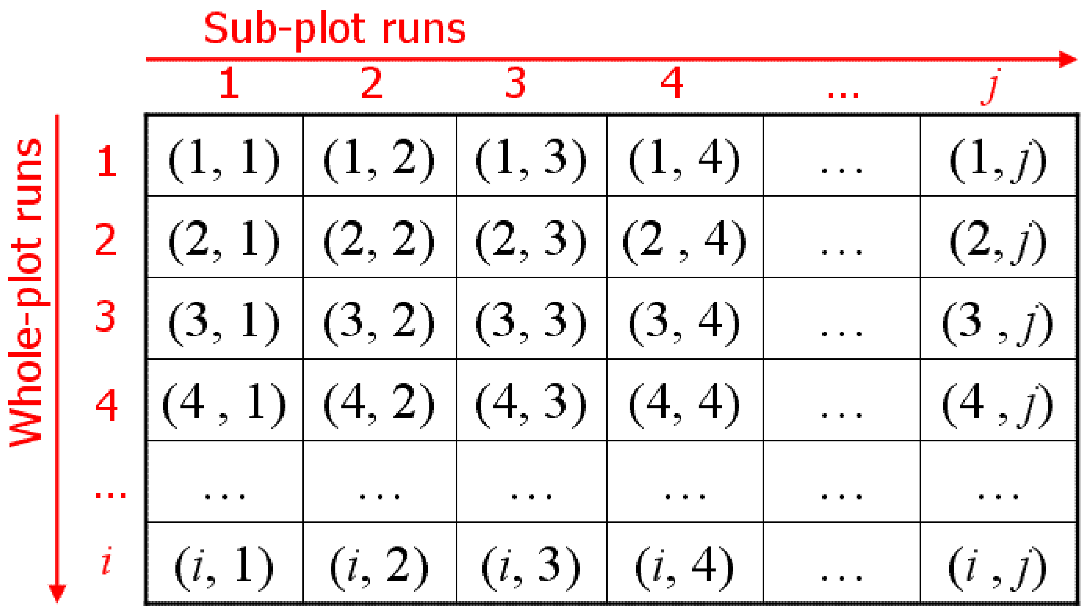


Figure 5.1: Arrangement of the runs in a split-plot design

5.1.1 The Injection Molding Experiment

In an injection-molding experiment, where the factors considered include: A) barrel temperature, B) mold temperature, C) holding pressure, D) back pressure and E) injection speed. This experiment has 5 factors at 2 levels. Typically for this type of experiment, the temperature settings are considered hard to change because every time the setting is changed, the temperature has to be reset and increased or decreased to the desired temperature. These heating and cooling cycles can be very time consuming. If conducted as a 2^5 fully randomized factorial experiment, this could require far more than several days of experimentation. Additionally, this may expose the system to the effects of nuisance factors that cannot be controlled, or even worse, factors that the experimenter might not be aware of. For experiments like these that involve hard-to-change factors, it may be more convenient to analyze using a split-plot experimental approach.

For this example, factors A (barrel temperature) and B (mold temperature) are considered as the whole-plot factors while C (holding pressure), D (back pressure) and E (injection speed) are the sub-plot factors. A random order is selected and shown in Table 5.1. As shown in the table, factors A and B are set at their low level (-1) in the first row. Afterwards, within the same run, the sub-plot factors are changed and run eight times (2^3), which is the total number of treatment level combinations. The first whole plot or row 1, is: c, ce, (1), de, e, d, cd and cde. After performing the first whole-plot run, the whole-plot treatments are set to ab and the runs are: abcde, abc, abe, abcd, abd, abde, ab, and abce. The experiment will continue in the same fashion by changing the hard-to-change factors just when all the treatment combinations of the subplot factors are exhausted.

Table 5.1: Randomly selected run orders for the injection molding experiment

			SUB-PLOT																							
RUN	**WHOLE-PLOT**		1			2			3			4			5			6			7			8		
	A	B	C	D	E	C	D	E	C	D	E	C	D	E	C	D	E	C	D	E	C	D	E	C	D	E
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
2	ab		1	1	1	1	-1	-1	-1	-1	1	1	1	-1	-1	1	-1	-1	1	1	-1	-1	-1	1	-1	1
3	a		1	1	1	-1	-1	-1	1	-1	-1	-1	1	1	-1	1	-1	-1	-1	1	1	-1	1	1	1	-1
4	b		1	1	-1	-1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	1	1	1	-1	1	1	-1	-1	-1

In this example, the design matrix for the whole-plot factor A is:

$$
D^{A} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}
$$

While the design matrix for the sub-plot factor C is:

$$D^{C} = \begin{bmatrix} +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & +1 & -1 & +1 & -1 & -1 & -1 & +1 \\ +1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 \end{bmatrix}$$

and similarly for the other factors.

5.2 Two-dimensional Trends

When looking at possible trends that can affect a split-plot design, the sub-plot factors might have a trend that runs across them. In the same manner, the whole-plot effects could also be affected by another possible trend. These column and row trend effects, if present, may have a composite effect in the experiment.

5.2.1 Trend Presence on the Sub-plot

Figure 5.2 shows how a split-plot arrangement can be affected by a trend. This trend, called the sub-plot trend in this work, could be present while conducting the sub-plot runs. This trend could be present and repeats itself every time the sub-plots are run. When the first sub-plot is run in the experiment, there could be a linear time trend (i.e.: $y = 2x$), as shown in Figure 5.2. This will mean that the first treatment combination in the first run (i.e.: 1,1) will experience an effect of magnitude 2. Similarly the next experimental unit will have a trend effect of magnitude 4, the third will be 6 and so forth until all runs are done. When performing the next run (i.e.: 2,1 2,2 … 2,8), the first will have a trend effect of 2 and then the next one will be 4 and so on.

5.2.2 Trend Presence on the Whole-plots

Just like the sub-plot runs, each whole plot run could be contaminated by a trend referred to as a whole-plot trend. The split-plot arrangement in Figure 5.2 has a whole-plot trend (i.e.: $y = x^2$) as well as a sub-plot trend (i.e.: $y = 2x$). This will mean that the first whole-plot run might be affected differently than the second and third whole-plot treatment combination. While each whole-plot run is affected by the same sub-plot trend, this gets compounded or amplified by the whole-plot trend. This means that the second sub-plot treatment combination in the first whole-plot row might not be affected in the same way as the second sub-plot in the second whole-plot row. This last statement forms the basis for the hypothesis tested in this thesis.

Sub-plot Trend ($y = 2x$)

Whole-plot Trend ($y = x^2$)

	2	4	6	8	…	$2j$
1	(1, 1)	(1, 2)	(1, 3)	(1, 4)	…	$(1, j)$
4	(2, 1)	(2, 2)	(2, 3)	(2 , 4)	…	$(2, j)$
9	(3, 1)	(3, 2)	(3, 3)	(3, 4)	…	$(3 , j)$
16	(4 , 1)	(4, 2)	(4, 3)	(4, 4)	…	$(4 , j)$
…	…	…	…	…	…	…
i^2	$(i, 1)$	$(i, 2)$	$(i, 3)$	$(i, 4)$	…	(i , j)

Figure 5.2: Trends on split-plot designs

5.2.3 Trend Interaction

As mentioned before, the whole-plot trend will have an effect on each sub-plot treatment combination. This interaction is represented in this thesis with $\tau_{i,j}$ which is

defined as the value of the interaction between a whole-plot trend and a sub-plot trend on the ith row and the jth column. This interaction can be modeled mathematically by making some assumptions.

A multiplicative interaction, as shown in Figure 5.3, will multiply the value of each whole-plot and the sub-plot trends. The example assumes that the sub-plot trend is modeled as $y = 2x$ while the whole-plot trend is quadratic modeled as $y = x^2$. The effects of the trend show a multiplicative increase in the effects of the trends. Therefore, $\tau_{1,2}$ has a value of 4 while $\tau_{2,2}$ has a value of 16.

Sub-plot Trend ($y = 2x$)

Whole-plot Trend ($y = x^2$)

	2	4	6	8	…	$2j$
1	1×2	1×4	1×6	1×8	…	$1 \times 2j$
4	4×2	16	24	32	…	$4 \times 2j$
9	18	36	54	72	…	$9 \times 2j$
16	32	64	96	128	…	$16 \times 2j$
…	…	…	…	…	…	…
i^2	$i^2 \times 2$	$i^2 \times 4$	$i^2 \times 6$	$i^2 \times 8$	…	$i^2 \times 2j$

Figure 5.3: Multiplicative interaction between trends.

Figure 5.4 shows an additive interaction between the whole-plot trend and the sub-plot trend. The additive interaction assumes that there is equal spacing between the runs, while in the multiplicative interaction, the difference between $\tau_{2,2}$ and $\tau_{2,3}$ is smaller than the difference between $\tau_{3,2}$ and $\tau_{3,3}$. This is not the case with the additive

interaction. By using the additive interaction it is implied that the spacing between runs has the same interval through time or position.

A subtractive interaction is a form of adding negative numbers; therefore, the additive interaction will work for this interaction. Just like subtraction, division is the multiplication of fractional numbers and they can also be interpreted as a multiplication interaction.

Sub-plot Trend ($y = 2x$)

Whole-plot Trend ($y = x^2$)

	2	4	6	8	...	$2j$
1	$1 + 2$	$1 + 4$	$1 + 6$	$1 + 8$	...	$1 + 2j$
4	$4 + 2$	8	10	12	...	$4 + 2j$
9	11	13	15	17	...	$9 + 2j$
16	18	20	22	24	...	$16 + 2j$
...	...	...	...	...	...	...
i^2	$i^2 + 2$	$i^2 + 4$	$i^2 + 6$	$i^2 + 8$	...	$i^2 + 2j$

Figure 5.4: Additive interaction between trends.

5.2.4 Implications from ignoring two-dimensional trend

Two-dimensional trends assume that there could be a trend that is correlated to the whole-plots. If this assumption is ignored, then it is implied that every whole-plot treatment combination is conducted under the exact same conditions. If this were the case, then there would be no two-dimensional trends in the split-plot and the whole plot factors can be selected randomly. However, there are situations where exact experimental conditions are impossible or impractical to achieve, such as conducting experiments

onsite in a manufacturing plant or outdoors with unpredictable climate conditions. In these cases, trends that might be present on the whole-plots will also be present on the sub-plots by interacting with the sub-plot trends.

As shown in section 5.2.3, each sub-plot could have a different trend due to the multiplication or addition of the whole-plot trend. If this interaction is ignored, a systematic selection of the run order might result in a design that is robust to the one-dimensional sub-plot trend or to the one-dimensional whole plot trend, but it might not be robust to the two-dimensional trend, resulting in a design that may adversely affect the results of the experiment.

5.2.5 Multiple 2D trends

There could be more than one trend that affects the whole-plot and sub-plots. The number of trends that could be modeled is infinite because the parameters in the models for trends can be modified in many ways to incorporate these.

Most of the possible trends are of no concern to the experimenter either because their potential correlation is insignificant compared to the few important trends. The trends that the experiment needs to be robust against are those that are suspected to have the highest influence in the design. It could be either 2 or 20 trends at the discretion of the experimenter. An experimenter might consider that if a trend could affect the experiment, then it may be worthwhile to design against it.

Being robust against the greatest number of potential trends may be the best option. Unfortunately, the more trends the design is supposed to be robust against, the harder it will be to obtain a feasible design. In any case, the design should be resistant to the most important trends.

5.2.6 Example

The injection molding example before might be affected by various trends. Consider that the sub-plot could be affected by the following polynomial trends modeled as $y = x$ and $y = x^2$. Furthermore a possible polynomial trend might be present on the whole-plots. Combined with the column trend, the resulting trend that affects the design could be an interaction between these two trends. This interaction can vary depending on how the whole plot trend interacts with the sub-plot trend.

Consider the example in Table 5.2, where the whole-plots and the sub-plots have linear trends, and the interaction between the linear whole-plot trend and the linear sub-plot trend is the multiplicative between them, where $\tau_{i,j} = i \times j$ and will produce the following linear × linear trend represented by the matrix $\boldsymbol{\tau}$:

Table 5.2: The linear × linear trend that affects the injection molding experiment

		Sub-plot Trend							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Whole-plot Trend	**1**	1	2	3	4	5	6	7	8
	2	2	4	6	8	10	12	14	16
	3	3	6	9	12	15	18	21	24
	4	4	8	12	16	20	24	28	32

$$\boldsymbol{\tau} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \\ 4 & 8 & 12 & 16 & 20 & 24 & 28 & 32 \end{bmatrix}$$

In table 5.3, the sub-plots are now believed to be affected by a quadratic trend and with the same multiplicative interaction between the linear whole-plot trend and the quadratic sub-plot trend. Therefore $\tau_{i,j} = i \times j^2$ resulting in the following linear × quadratic trend $\boldsymbol{\tau}$:

Table 5.3: The linear × quadratic trend that affects the injection molding experiment

		Sub-plot Trend							
		1	**4**	**9**	**16**	**25**	**36**	**42**	**64**
Whole-plot Trend	**1**	1	4	9	16	25	36	42	64
	2	2	8	18	32	50	72	84	128
	3	3	12	27	48	75	108	126	192
	4	4	16	36	64	100	144	168	256

$$\boldsymbol{\tau} = \begin{bmatrix} 1 & 4 & 9 & 16 & 25 & 36 & 42 & 64 \\ 2 & 8 & 18 & 32 & 50 & 72 & 84 & 128 \\ 3 & 12 & 27 & 48 & 75 & 108 & 126 & 192 \\ 4 & 16 & 36 & 64 & 100 & 144 & 168 & 256 \end{bmatrix}$$

5.3 Trend Index

5.3.1 Definition

In this thesis, the robustness of a design against the trends will be measured by calculating the proposed Trend Index (TI). The Trend Index proposed in this work is a measurement of how robust the design is against a specific trend. More specifically, it cumulatively measures if a trend has an adverse effect on a particular design. Draper and Stoneman (1968), and Dickinson (1973) refer to this term as a time count but this is probably because they studied the robustness against time-correlated variables. The Trend Index defines a much broader area of robustness and assumes that the trends do not necessarily have to be time related. Instead, there could be many types of trends such as learning curves, tool wear, and temperature fluctuations as mentioned in section 3.2.

A value of 0 on the TI will define that the selected design is perfectly orthogonal (i.e. robust) to the trend it is exposed to. A departure from 0 means a smaller degree of robustness. This value is relative to the design and the trends used; therefore, the TI from

a specific design and trends should not be compared to other designs and trends of different size in dimensions. The TI is a measurement that compares the robustness of a particular design against one or more trends with other arrangements of the same experiment.

The proposed TI can be used in calculating the robustness of any design of any dimension. This thesis will focus on the TI with two-dimensional trends from hierarchical designs such as split plots.

5.3.2 Notation

Let D^{α} be the design matrix of contrast for factor α and $\boldsymbol{\tau}$ be the matrix that denotes a trend with the same size as D^{α}. The formula used to measure the robustness of the design is:

$$TI^{\alpha} = \sum_{i=1}^{2^w} \sum_{j=1}^{2^s} D_{i,j}^{\alpha} \times \tau_{i,j} .$$

This TI will be used to quantify the robustness of the contrast arrangement of factor α with the trend $\boldsymbol{\tau}$. It is convenient to standardize the TI so that it will always show a positive value. This helps when comparing trend indexes and optimizing the design. To incorporate more than one two-dimensional trend, a set of $\boldsymbol{\tau}_{\phi}$ is defined where ϕ represents a specific trend interaction as defined by the experimenter. With the addition of the new trends, the formula is modified to:

$$TI_{\phi}^{\alpha} = \left| \sum_{i=1}^{2^w} \sum_{j=1}^{2^s} D_{i,j}^{\alpha} \times \tau_{i,j,\phi} \right|$$

where ϕ defines which potential trend the TI is aiming to calculate.

5.3.3 Example

Using the injection molding experiment introduced in section 5.1 and the trends obtained in section 5.2, the experiment needs to be simultaneously robust against the following two trends:

$$\boldsymbol{\tau}_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \\ 4 & 8 & 12 & 16 & 20 & 24 & 28 & 32 \end{bmatrix}, \text{ and}$$

$$\boldsymbol{\tau}_2 = \begin{bmatrix} 1 & 4 & 9 & 16 & 25 & 36 & 42 & 64 \\ 2 & 8 & 18 & 32 & 50 & 72 & 84 & 128 \\ 3 & 12 & 27 & 48 & 75 & 108 & 126 & 192 \\ 4 & 16 & 36 & 64 & 100 & 144 & 168 & 256 \end{bmatrix}$$

Where $\boldsymbol{\tau}_1$ is produced by the following model: $\tau_{i,j} = i \times j$, and $\boldsymbol{\tau}_2$ is produced by the model: $\tau_{i,j} = i \times j^2$.

Table 5.4 gives the trend indexes of all five factors in the run order established in section 5.1.1. Only factor A is resistant to both trends.

Table 5.4: Trend Index (*TI*) for each factor

	$\boldsymbol{\tau}_\phi$	
Factor α	$\boldsymbol{\tau}_1$	$\boldsymbol{\tau}_2$
A	0	0
B	72	408
C	22	102
D	2	2
E	18	142

The random selection did not produce trend resistance on the other factors. For this example the random design $\mathbf{D}^{\mathrm{C}}$ chosen for factor C:

$$
\mathrm{D}^{\mathrm{C}} = \begin{bmatrix} +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & +1 & -1 & +1 & -1 & -1 & -1 & +1 \\ +1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 \end{bmatrix}
$$

in the injection molding experiment yielded the following trend indexes:

$$
TI_1^{\mathrm{C}} = \left| \sum_{i=1}^{4} \sum_{j=1}^{8} D_{i,j}^{\mathrm{C}} \times \tau_{i,j,1} \right| = 22 \text{ for } \boldsymbol{\tau}_1 \text{ and } TI_2^{\mathrm{C}} = \left| \sum_{i=1}^{4} \sum_{j=1}^{8} D_{i,j}^{\mathrm{C}} \times \tau_{i,j,2} \right| = 102 \text{ for } \boldsymbol{\tau}_2 .
$$

Factor C is not robust to either $\boldsymbol{\tau}_1$ or $\boldsymbol{\tau}_2$.

If the design for the injection molding experiment is modified as proposed on table 5.5 then the trend index will change as seen on table 5.6. Every sub-plot factor has received a substantial improvement. The linear sub-plot trend has no effect on factor B. To improve the robustness of factor B will require the reordering of the whole-plot factors, which might aggravate factor A. It is up to the experimenter to decide what design is best for the whole-plot factors. The order of the whole-plot factors will not have any negative effect on the trend index of the sub-plot factors. Therefore run orders for the subplot factors and the whole-plot factors can be designed separately.

Table 5.5: Redesigned injection molding experiment

			SUB-PLOT																					
RUN	WHOLE-PLOT		1			2			3			4			5			6			7			8		
	A	B	C	D	E	C	D	E	C	D	E	C	D	E	C	D	E	C	D	E	C	D	E	C	D	E
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
2	ab		1	1	1	-1	1	1	1	-1	1	-1	-1	1	1	1	-1	-1	1	-1	1	-1	-1	-1	-1	-1
3	a		1	1	1	-1	1	1	1	-1	1	-1	-1	1	1	1	-1	-1	1	-1	1	-1	-1	-1	-1	-1
4	b		-1	-1	-1	1	-1	-1	-1	1	-1	1	1	-1	-1	-1	1	1	-1	1	-1	1	1	1	1	1

Table 5.6: Trend indexes (Ti) for each redesigned factor

	$\boldsymbol{\tau}_{\phi}$	
Factor α	$\boldsymbol{\tau}_1$	$\boldsymbol{\tau}_2$
A	0	0
B	72	408
C	0	0
D	0	0
E	0	0

Comparing the results of both designs, the second design contains four factors that are robust to $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ while the randomly selected design only has one factor robust to both trends. The second design is a better design when it comes to being robust to trends. It is possible that there are more trends involved. For the sake of the example, only two trends were selected. Other trends might have been present and could have been included in the set of $\boldsymbol{\tau}_{\phi}$.

5.4 Other Metrics

The trend index can be used to estimate of the robustness of a design against several trends. If the TI approaches 0, then the design is nearly trend free or nearly-robust design. A nearly-robust design can be defined as a design whose TI is not equal to 0 but it is the closest the design can be under the given conditions. There are some trends and some designs that cannot be orthogonal to each other. In this case the TI will never be 0. Ideally, a design should have a very small number for a TI. The TI can be used with multiple trends, and if this is the case, a summation of the TI of every factor and trends will show the robustness of the design.

The TI, however, cannot be used as a comparison between different designs and different trends. For example, the TI for a $2^2 \times 2^3$ split-plot design calculated for the

trends: $\tau_{i,j} = i \times j$, $\tau_{i,j} = i \times j^2$, and $\tau_{i,j} = i^2 \times j$, cannot be compared to the TI of the same design calculated with different trends such as: $\tau_{i,j} = i \times j^3$, $\tau_{i,j} = i^3 \times j$, and $\tau_{i,j} = i^2 \times j^2$. The TI is intended as a metric used to select the order of the design under specific possible trends. A particular order in a design might have a lower TI than another. This TI is then considered to be more robust than the design with the higher trend.

This thesis will use an additional metric the number of completely robust trends. The purpose for including this metric lies in the fact that it is possible for a design to have a very low TI, but none of the factors and none of the trends are completely robust. A completely robust design is defined as a design whose TI is equal to 0 for the given trends. Some experimenters might be more interested in the number of completely robust trends than in a nearly robust design.

5.5 Methodology

There have been several methods used in the past to generate designs robust to trends; however, few of them can be used on row × column designs. The fold-over method has been widely used to design factorial and fractional factorial experiments robust to trends (John, 1990). However, there is no available extension done on folding a split-plot design to generate robust designs. Carrano *et. al.* (2007) used integer linear programming to generate a feasible order for a specific split-plot experiment. A generalized integer linear program can be used for designing split-plot experiment to be robust against trends.

This thesis develops two methods to design general 2-level split-plot designs to be robust to trends. One method is an extension of the fold-over approach and the other method uses integer linear programming that optimizes the TI. The two methods will be compared and an appropriate method can be used to design the split-plot experiment.

The fold-over method uses the same principle used in factorial and fractional factorial experiments as proposed by Box and Wilson (1951) in literature and used against trends by John (1990), and modifies the algorithm so that it can be used in split-plot designs or any other row × column design. Chapter 6 explains how to perform a fold-over extension to generate a robust 2-level split-plot design.

The integer linear programming method uses mathematical programming to obtain a design with optimal TI. This is addressed in chapter 7. In the work by Carrano *et. al.* (2007), the integer linear program found a feasible solution for their split-plot design. However, this solution was not proven optimal. There may be several other solutions that can outperform their design, especially when considering other metrics.

Both methods will concentrate on designing split-plot experiments that are robust to the following two-dimensional trends: linear × linear (L × L), linear × quadratic (L × Q), linear × cubic (L × C), quadratic × linear (Q × L), quadratic × quadratic (Q × Q), quadratic × cubic (Q × C), cubic × linear (C × L), cubic × quadratic (C × Q), and cubic × cubic (C × C). There will be designs robust or nearly robust to each of the trends individually and to all nine trends simultaneously.

The arrangement of the whole-plot factors are not affected by the order of the sub-plot factors and are not influenced by the two-dimensional trend. The whole-plots can be designed separately and this thesis will only discuss the methods used for arranging the

order of the sub-plot treatment level combinations. The whole-plots can be arranged by the methods well established in the literature such as: Daniel and Wilcoxon (1966), and Cheng and Jacroux (1988), among others.

6 Fold-Over Method

6.1 Definition

The principle of folding over has been used as a technique for generating trend resistant designs (Cheng and Jacroux, 1988, Coster and Cheng, 1988, and John, 1990). It is used in DOE to set up experiments so that they are orthogonal to polynomial trends. The general fold-over of a design starts with N treatment level combinations or points. For a two-level design, each point contains the contrast for a factor in its high (+1) and low (-1) state. A new set of N points is then generated, and the design will have a total $2N$ points, where the $(N + i)$th position has a contrast that is opposite of that in the ith unit. A more detailed explanation on fold-over procedure can be found in John (1990).

6.2 Notation

For this method, selecting the order of the contrast in the first row of a design is very important for generating a design that has the smallest trend index. This first whole-plot row is called the generator row, or generator contrast for a specific factor. In table 5.5, the first row was set to follow Yates order. Higher degree trends would require a different initial order.

Let $c_{i,j}$ be the contrast level for a factor in row i and column j of the design matrix D^{α} and can be either -1 or 1. Let $\bar{c}_{i,j} = -c_{i,j}$. Since the focus of this work is to build robustness on the sub-plot factors, then only the sub-plot factors are required to be folded. Initially the fold-over requires a starting generator. This generator will be the first

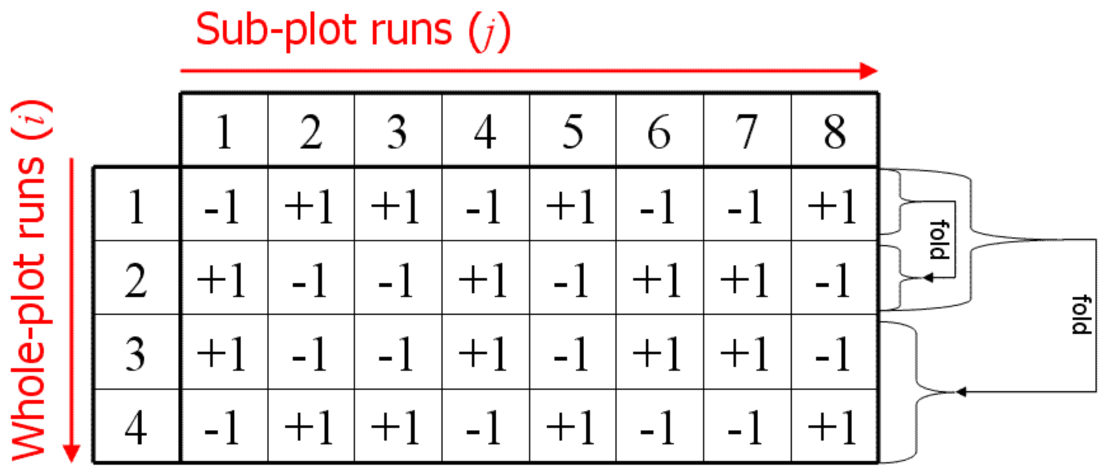
whole-plot row ($c_{1,j}$) in the split-plot design. The following algorithm will fold-over the factor across all whole-plot rows.

Split-plot Fold-over algorithm.

1. For a $2^w \times 2^s$ split-plot design, set $m = 2^s$, n = 1 and $r = 2^w$
2. $c_{1,j}$ for all $j = \{1,2,...,m\}$ is assigned to the generator row.
3. Fold-over $c_{i+n,j} = \bar{c}_{i,j}$ for $i = \{1,..., n\}$ and $j = \{1,2,..., m\}$. Set n = 2n.
4. If $n \geq r$ then design is finished, else go to 3.

The two-level split-plot design can only be folded *w* times. The more the design is folded, the more trends will it be resistant to. However, selection of the generator row is crucial for getting optimal trend indices. This thesis will discuss methods for selecting generators later in the chapter.

Figure 6.1 illustrates how the fold over algorithm works to fold over one factor across all whole-plots. The design in figure 6.1 is a $2^2 \times 2^3$ split-plot. For this design, $m = 2^3 = 8$, n = 1 and $r = 2^2 = 4$. This example will use the generator row ($c_{1,j}$) $[-1 \ \ +1 \ \ +1 \ \ -1 \ \ +1 \ \ -1 \ \ -1 \ \ +1]$ and correspond to the row $i = 1$. Row $i = 2$ is obtained by folding row $i = 1$. Row 2 is now $[+1 \ \ -1 \ \ -1 \ \ +1 \ \ -1 \ \ +1 \ \ +1 \ \ -1]$ which satisfies $c_{i+n,j} = \bar{c}_{i,j}$. Because now n = 2 and $n \leq r$, rows 1 and 2 are folded again into the new rows 3 and 4 respectively. The complete folded design is shown in figure 6.1. Since w = 2 (number of whole-plot factors), the design can only be folded twice.



	1	2	3	4	5	6	7	8
1	-1	+1	+1	-1	+1	-1	-1	+1
2	+1	-1	-1	+1	-1	+1	+1	-1
3	+1	-1	-1	+1	-1	+1	+1	-1
4	-1	+1	+1	-1	+1	-1	-1	+1

Figure 6.1: Fold-over on a $2^2 \times 2^3$ split-plot design

6.3 Adapting the Split-plot Fold-over Method

The fold-over algorithm needs a generator row to start the fold-over. The general fold-over algorithm has a generator that allows using all of the factors at once. This can cause complications when identifying the generator. The maximum number of possible generators is $2^s!$. For example, if there are two sub-plot factors, then the design has a total of $2^s!$ possible generators ($[(1) \quad a \quad b \quad ab], [(1) \quad a \quad ab \quad b]$, … , etc). This number will get very large as more and more sub-plot factors are added to the design. With 3 factors, the number of generators is 40,320. With 4 factors, the number increases to 2.0922×10^{13}. With 5 or more factors, these very large numbers cannot even be handled by some computational devices, and will take a large amount of time to go through every possible generator.

This thesis will concentrate on a simplified method for the previously defined fold-over method. Since every factor has the same possible generators, the method will only fold one factor and use these results on the remaining factors. The total combinatory possibilities (N_α) for the sub-plot factor α is calculated by:

$$N_{\alpha} = \binom{2^{s}}{2^{s-1}} = \frac{2^{s}!}{(2^{s-1}!)(2^{s-1}!)}.$$

For 2 sub-plot factors, there are 6 possible generators. For 3 factors, there are 70 possible combinations. For 4 and 5 the number of possible combinations is 12,870 and 6.01×10^{8} respectively. Every factor has the same number of possible contrast designs in that row, but each factor is constrained by other factors' contrasts. If the generator row selected for factor A in a design with 3 sub-plot factors is $[-1 \quad +1 \quad -1 \quad +1 \quad -1 \quad +1 \quad -1 \quad +1]$, then neither factor B or C can have this contrast arrangement. The generator row for factor B has to be orthogonal to the generator rows for factors A and C.

Rule 1: Let C_A and C_B be the array of contrast of the factor A and B respectively of size 2^s (cardinality) where s is the total number of sub-plot factors. The contrast can be only –1 or +1 and the summation of the terms in the array must equal to 0. Factor A and Factor B can be in the same design if and only if they are orthogonal to each other. To be orthogonal, the factors need to satisfy the following equation: $C_A \times C'_B = 0$.

As shown in Figure 6.2, two sub-plot factors can be in the same design if they are orthogonal to each other. This means that they have to satisfy the following rule: $C_A \times C'_B = 0$, where C_A and C_B are the contrast arrays of both factors. If the design for factor A is generated with $[+1 \quad -1 \quad +1 \quad -1]$, then the corresponding factor B cannot have a generator row of $[-1 \quad +1 \quad -1 \quad +1]$ because:

$$[+1 \quad -1 \quad +1 \quad -1] \times \begin{bmatrix} -1 \\ +1 \\ -1 \\ +1 \end{bmatrix} = -4$$

This makes sense because this design will be: $[a \quad b \quad a \quad b]$ and all treatment combinations are not represented. Instead, if C_A = $[+1 \quad -1 \quad +1 \quad -1]$ and C_B = $[+1 \quad +1 \quad -1 \quad -1]$ then $C_A \times C_B' = 0$ and the experiment will be: $[ab \quad b \quad a \quad (1)]$.

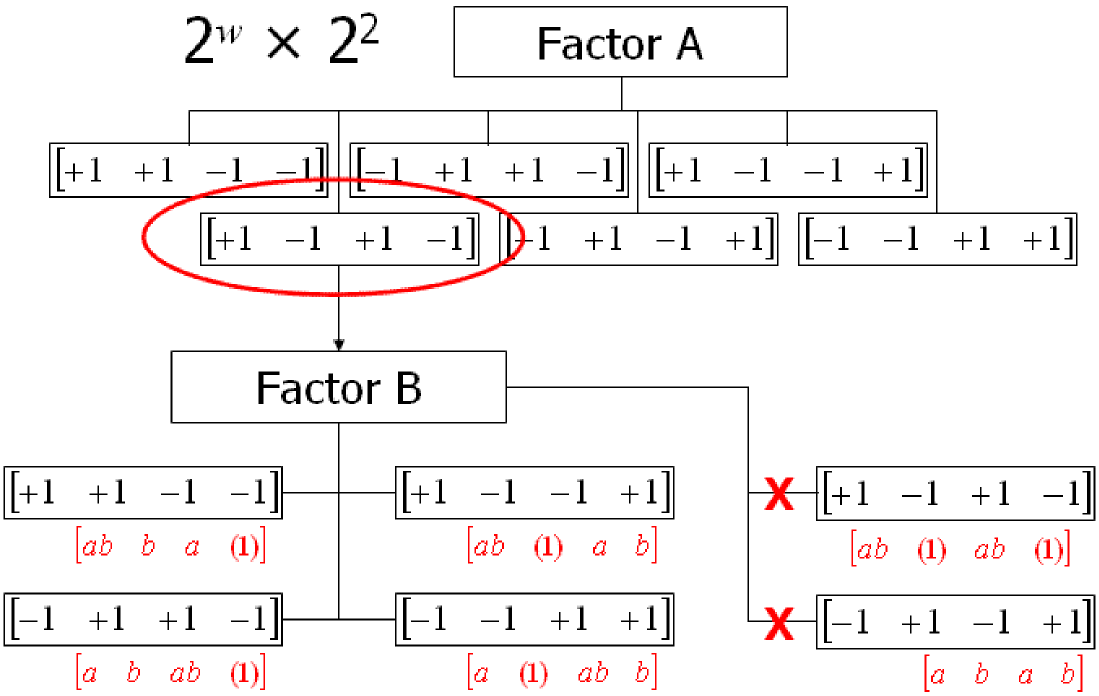


Figure 6.2: Possible combinations for generators in a $2^w \times 2^2$ design.

For selecting all of the generators in the sub-plots, one factor needs to be selected at a time. The generator for the first factor is first selected out of the all the possible generators. Afterwards, the generator for the second factor is selected from the possible generators as long as it is orthogonal to the first factor. The generator for the third and following factors is selected in the same manner and it needs to be orthogonal to the already selected generators. The limitations to this methodology arise when selecting the generator for other factors. Although an exhaustive list is used to select the design for the other factors, some of the generators cannot be selected. No two factors can have the same contrast matrix. If the wrong combination of generators is selected, then the

resulting experiment will be an incomplete experiment. With this method, every generator is folded and compared to the trends, and the best design of the available designs is selected for the factors.

6.4 Setup

Because every factor has $N = \frac{2^s!}{\left(2^{s-1}!\right)\left(2^{s-1}!\right)}$ different possible generators, and all generators are the same for every factor, the method used in this thesis consists of analyzing one factor at a time by folding all of the generators. The designs containing several factors will have a limitation when folding over the design one factor at a time. If a generator row for one factor is selected, the set of generators for the next factor are then reduced. As more factors are selected, limitations are imposed in the remaining factors, which could potentially eliminate an effective robustness to the trends for these remaining factors.

Every possible combination for the generator row is folded and the resulting design is compared to several two-dimensional trends. The two-dimensional trend is coded as whole-plot trend × sub-plot trend. All nine combinations of linear, quadratic, and cubic trends on the sub-plot level and whole-plot level are evaluated. For simplicity, the trends are coded as: L, Q, and C for linear ($y = x$), quadratic ($y = x^2$), and cubic ($y = x^3$) respectively. With this coding, L × Q refers to the two-dimensional trend generated by a linear trend on the whole-plot factors and a quadratic trend on the sub-plot factors.

The number of whole plots does not affect the number of generators which is based on the number of sub-plot factors. This thesis is focused on designs with 4 or less

subplot factors. The number of possible generators would increase exponentially with the number of factors. Each generator for each design is folded to have 2, 3, 4, and 5 whole-plots. Although it is possible to design with more whole-plots without incrementing the number of possible generators, the results should be clear to this extent. As mentioned before, because of the fold-over properties, the more the factors are folded, the more robust they are to higher degree trends.

6.5 Results

Table 6.1 shows how different generator rows used in the fold-over method change the TI of the specific design. This TI is only shown for one factor. The table shows only the fold-over for the following 2-level split-plot designs: $2^2 \times 2^2$, $2^3 \times 2^2$, $2^4 \times 2^2$, and $2^5 \times 2^2$. Appendix B, C, D, and E contain tables for the following 2-level split-plot designs: $2^2 \times 2^3$, $2^3 \times 2^3$, $2^4 \times 2^3$, and $2^5 \times 2^3$ respectively. The TI is calculated using the formula:

$$TI^{\alpha} = \left| \sum_{i=1}^{2^w} \sum_{j=1}^{2^s} D_{i,j}^{\alpha} \times \tau_{i,j} \right|,$$

on each trend using the contrast design matrix $\mathbf{D}^{\alpha}$ generated by folding each generator row. Because there are nine two-dimensional trends that are calculated, the total TI is measured with

$$TI^{\alpha} = \sum_{\phi=1}^{9} \left| \sum_{i=1}^{2^w} \sum_{j=1}^{2^s} D_{i,j}^{\alpha} \times \tau_{i,j,\phi} \right|$$

where ϕ denotes the trends. If the selected generator row for factor A in the $2^2 \times 2^2$ split-plot design is the generator #4 in Table 6.1 ([+1 –1 –1 +1]), it will generate the following design when folded:

$$\mathbf{D}^A = \begin{bmatrix} +1 & -1 & -1 & +1 \\ -1 & +1 & +1 & -1 \\ -1 & +1 & +1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}.$$

This design has a total TI of 1156. On the other hand, if the generator selected is the generator #6 ([–1 –1 +1 +1]), then the resulting design after folding will be:

$$\mathbf{D}^B = \begin{bmatrix} -1 & -1 & +1 & +1 \\ +1 & +1 & -1 & -1 \\ +1 & +1 & -1 & -1 \\ -1 & -1 & +1 & +1 \end{bmatrix},$$

which has a TI of 3604. This value, however, does not indicate the number of completely

Table 6.1: TI results for folded generators for $2^w \times 2^2$ split-plot designs for factor A

	Design for: 2^2 X 2^2										
	Generator row	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	Total
1	+1 +1 -1 -1	0	0	0	16	80	328	120	600	2460	3604
2	+1 -1 +1 -1	0	0	0	8	40	176	60	300	1320	1904
3	-1 +1 +1 -1	0	0	0	0	16	120	0	120	900	1156
4	+1 -1 -1 +1	0	0	0	0	16	120	0	120	900	1156
5	-1 +1 -1 +1	0	0	0	8	40	176	60	300	1320	1904
6	-1 -1 +1 +1	0	0	0	16	80	328	120	600	2460	3604

	Design for: 2^3 X 2^2										
	Generator row	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	Total
7	+1 +1 -1 -1	0	0	0	0	0	0	192	960	3936	5088
8	+1 -1 +1 -1	0	0	0	0	0	0	96	480	2112	2688
9	-1 +1 +1 -1	0	0	0	0	0	0	0	192	1440	1632
10	+1 -1 -1 +1	0	0	0	0	0	0	0	192	1440	1632
11	-1 +1 -1 +1	0	0	0	0	0	0	96	480	2112	2688
12	-1 -1 +1 +1	0	0	0	0	0	0	192	960	3936	5088

	Design for: 2^4 X 2^2										
	Generator row	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	Total
13	+1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
14	+1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
15	-1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
16	+1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
17	-1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
18	-1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0

	Design for: 2^5 X 2^2										
	Generator row	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	Total
19	+1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
20	+1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
21	-1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
22	+1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
23	-1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
24	-1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0

robust trends or which trends are more robust than others. The metric used for the selection of the generator is at the discretion of the experimenter.

It should be noted that, when considering factor B in the previous $2^2 \times 2^2$ design, if A has the generator row [+1 –1 –1 +1], which has the lowest TI, then B is restricted since it can only use the generators, [+1 +1 –1 –1] [+1 –1 +1 –1] [–1 +1 –1 +1] or [–1 –1 +1 +1]. The other two generators, [+1 –1 –1 +1] and [–1 +1 +1 –1], cannot be used by factor B because if used, it will not generate a full split-plot design.

The following greedy selection method can be used to select the generators for every sub-plot factor in the design. Any metric can be used for this selection method.

1. Enumerate the list of possible generators for *s* sub-plot factors. Each generator should refer to a metric, i.e. the total TI.
2. Arrange the factors in decreasing order of importance.
3. Select the generator with the best metric for the first factor.
4. Eliminate from the list of generators all of the generators that are not orthogonal to the last generator selected.
5. Select from the new list of generators the generator with the best metric to be used for the next factor in the sequence.
6. Repeat steps 4 and 5 until all factors are selected.

As observed in Table 6.1, the best selection for factor B under a greedy approach is either [+1 –1 +1 –1] or [–1 +1 –1 +1] because they both have lower TI (1904) than the generators: [+1 +1 –1 –1] or [–1 –1 +1 +1]. In the end, the best design for the $2^2 \times 2^2$ split-plot experiment using the fold-over method will be given by the following design matrices:

$$\mathbf{D}^A = \begin{bmatrix} +1 & -1 & -1 & +1 \\ -1 & +1 & +1 & -1 \\ -1 & +1 & +1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix} \text{ and } \mathbf{D}^B = \begin{bmatrix} -1 & -1 & +1 & +1 \\ +1 & +1 & -1 & -1 \\ +1 & +1 & -1 & -1 \\ -1 & -1 & +1 & +1 \end{bmatrix}.$$

This design will have a total TI for A of 1156 and a total TI for B of 1904, for a total TI of 3060. Furthermore, this design is robust against 5 trends (factor A) and against 3 trends (factor B).

This method prioritizes the factors but does not prioritize the trends because the fold-over method does not target the trends to be robust against. This approach can work on split-plot designs of any size $2^w \times 2^s$, where w is the number of whole-plot factors and s is the number of sub-plots. The limitations lie with the number of sub-plots, which increase the number of generators that have to be listed.

6.6 Analysis

Because of the properties associated with the fold-over method, the design will generally be completely robust to low-degree trends, such as L × L trends. The larger the design, the more times it can be folded, and the more robust the design can be to higher degree trends. The factors, however, have to be selected in order of importance. When one factor is selected to have the best overall trend resistance, the subsequent factors present a reduced resistance to a higher degree trends.

The fold-over method, however, does not distinguish between trends. Table 6.2 shows the trend index results of folding over all of the generators in a design with 2 whole-plots and 3 sub-plots ($2^2 \times 2^3$). There are a total of 70 possible generator outcomes.

Table 6.2: TI results for folded generators for a $2^2 \times 2^3$ split-plot design

	Design for: 2^2 X 2^3										
	Generator row	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	Total
1	+1 +1 +1 +1 -1 -1 -1 -1	0	0	0	64	576	4384	480	4320	32880	42704
2	+1 +1 +1 -1 +1 -1 -1 -1	0	0	0	56	504	3896	420	3780	29220	37876
3	+1 +1 -1 +1 +1 -1 -1 -1	0	0	0	48	448	3600	360	3360	27000	34816
4	+1 -1 +1 +1 +1 -1 -1 -1	0	0	0	40	408	3448	300	3060	25860	33116
5	-1 +1 +1 +1 +1 -1 -1 -1	0	0	0	32	384	3392	240	2880	25440	32368
6	+1 +1 +1 -1 -1 +1 -1 -1	0	0	0	48	416	3168	360	3120	23760	30872
7	+1 +1 -1 +1 -1 +1 -1 -1	0	0	0	40	360	2872	300	2700	21540	27812
8	+1 -1 +1 +1 -1 +1 -1 -1	0	0	0	32	320	2720	240	2400	20400	26112
9	-1 +1 +1 +1 -1 +1 -1 -1	0	0	0	24	296	2664	180	2220	19980	25364
10	+1 +1 -1 -1 +1 +1 -1 -1	0	0	0	32	288	2384	240	2160	17880	22984
11	+1 -1 +1 -1 +1 +1 -1 -1	0	0	0	24	248	2232	180	1860	16740	21284
12	-1 +1 +1 -1 +1 +1 -1 -1	0	0	0	16	224	2176	120	1680	16320	20536
13	+1 -1 -1 +1 +1 +1 -1 -1	0	0	0	16	192	1936	120	1440	14520	18224
14	-1 +1 -1 +1 +1 +1 -1 -1	0	0	0	8	168	1880	60	1260	14100	17476
15	-1 -1 +1 +1 +1 +1 -1 -1	0	0	0	0	128	1728	0	960	12960	15776
16	+1 +1 +1 -1 -1 -1 +1 -1	0	0	0	40	312	2152	300	2340	16140	21284
17	+1 +1 -1 +1 -1 -1 +1 -1	0	0	0	32	256	1856	240	1920	13920	18224
18	+1 -1 +1 +1 -1 -1 +1 -1	0	0	0	24	216	1704	180	1620	12780	16524
19	-1 +1 +1 +1 -1 -1 +1 -1	0	0	0	16	192	1648	120	1440	12360	15776
20	+1 +1 -1 -1 +1 -1 +1 -1	0	0	0	24	184	1368	180	1380	10260	13396
21	+1 -1 +1 -1 +1 -1 +1 -1	0	0	0	16	144	1216	120	1080	9120	11696
22	-1 +1 +1 -1 +1 -1 +1 -1	0	0	0	8	120	1160	60	900	8700	10948
23	+1 -1 -1 +1 +1 -1 +1 -1	0	0	0	8	88	920	60	660	6900	8636
24	-1 +1 -1 +1 +1 -1 +1 -1	0	0	0	0	64	864	0	480	6480	7888
25	-1 -1 +1 +1 +1 -1 +1 -1	0	0	0	8	24	712	60	180	5340	6324
26	+1 +1 -1 -1 -1 +1 +1 -1	0	0	0	16	96	640	120	720	4800	6392
27	+1 -1 +1 -1 -1 +1 +1 -1	0	0	0	8	56	488	60	420	3660	4692
28	-1 +1 +1 -1 -1 +1 +1 -1	0	0	0	0	32	432	0	240	3240	3944
29	+1 -1 -1 +1 -1 +1 +1 -1	0	0	0	0	0	192	0	0	1440	1632
30	-1 +1 -1 +1 -1 +1 +1 -1	0	0	0	8	24	136	60	180	1020	1428
31	-1 -1 +1 +1 -1 +1 +1 -1	0	0	0	16	64	16	120	480	120	816
32	+1 -1 -1 -1 +1 +1 +1 -1	0	0	0	8	72	296	60	540	2220	3196
33	-1 +1 -1 -1 +1 +1 +1 -1	0	0	0	16	96	352	120	720	2640	3944
34	-1 -1 +1 -1 +1 +1 +1 -1	0	0	0	24	136	504	180	1020	3780	5644
35	-1 -1 -1 +1 +1 +1 +1 -1	0	0	0	32	192	800	240	1440	6000	8704
36	+1 +1 +1 -1 -1 -1 -1 +1	0	0	0	32	192	800	240	1440	6000	8704
37	+1 +1 -1 +1 -1 -1 -1 +1	0	0	0	24	136	504	180	1020	3780	5644
38	+1 -1 +1 +1 -1 -1 -1 +1	0	0	0	16	96	352	120	720	2640	3944
39	-1 +1 +1 +1 -1 -1 -1 +1	0	0	0	8	72	296	60	540	2220	3196
40	+1 +1 -1 -1 +1 -1 -1 +1	0	0	0	16	64	16	120	480	120	816
41	+1 -1 +1 -1 +1 -1 -1 +1	0	0	0	8	24	136	60	180	1020	1428
42	-1 +1 +1 -1 +1 -1 -1 +1	0	0	0	0	0	192	0	0	1440	1632
43	+1 -1 -1 +1 +1 -1 -1 +1	0	0	0	0	32	432	0	240	3240	3944
44	-1 +1 -1 +1 +1 -1 -1 +1	0	0	0	8	56	488	60	420	3660	4692
45	-1 -1 +1 +1 +1 -1 -1 +1	0	0	0	16	96	640	120	720	4800	6392
46	+1 +1 -1 -1 -1 +1 -1 +1	0	0	0	8	24	712	60	180	5340	6324
47	+1 -1 +1 -1 -1 +1 -1 +1	0	0	0	0	64	864	0	480	6480	7888
48	-1 +1 +1 -1 -1 +1 -1 +1	0	0	0	8	88	920	60	660	6900	8636
49	+1 -1 -1 +1 -1 +1 -1 +1	0	0	0	8	120	1160	60	900	8700	10948
50	-1 +1 -1 +1 -1 +1 -1 +1	0	0	0	16	144	1216	120	1080	9120	11696
51	-1 -1 +1 +1 -1 +1 -1 +1	0	0	0	24	184	1368	180	1380	10260	13396
52	+1 -1 -1 -1 +1 +1 -1 +1	0	0	0	16	192	1648	120	1440	12360	15776
53	-1 +1 -1 -1 +1 +1 -1 +1	0	0	0	24	216	1704	180	1620	12780	16524
54	-1 -1 +1 -1 +1 +1 -1 +1	0	0	0	32	256	1856	240	1920	13920	18224
55	-1 -1 -1 +1 +1 +1 -1 +1	0	0	0	40	312	2152	300	2340	16140	21284
56	+1 +1 -1 -1 -1 -1 +1 +1	0	0	0	0	128	1728	0	960	12960	15776
57	+1 -1 +1 -1 -1 -1 +1 +1	0	0	0	8	168	1880	60	1260	14100	17476
58	-1 +1 +1 -1 -1 -1 +1 +1	0	0	0	16	192	1936	120	1440	14520	18224
59	+1 -1 -1 +1 -1 -1 +1 +1	0	0	0	16	224	2176	120	1680	16320	20536
60	-1 +1 -1 +1 -1 -1 +1 +1	0	0	0	24	248	2232	180	1860	16740	21284
61	-1 -1 +1 +1 -1 -1 +1 +1	0	0	0	32	288	2384	240	2160	17880	22984
62	+1 -1 -1 -1 +1 -1 +1 +1	0	0	0	24	296	2664	180	2220	19980	25364
63	-1 +1 -1 -1 +1 -1 +1 +1	0	0	0	32	320	2720	240	2400	20400	26112
64	-1 -1 +1 -1 +1 -1 +1 +1	0	0	0	40	360	2872	300	2700	21540	27812
65	-1 -1 -1 +1 +1 -1 +1 +1	0	0	0	48	416	3168	360	3120	23760	30872
66	+1 -1 -1 -1 -1 +1 +1 +1	0	0	0	32	384	3392	240	2880	25440	32368
67	-1 +1 -1 -1 -1 +1 +1 +1	0	0	0	40	408	3448	300	3060	25860	33116
68	-1 -1 +1 -1 -1 +1 +1 +1	0	0	0	48	448	3600	360	3360	27000	34816
69	-1 -1 -1 +1 -1 +1 +1 +1	0	0	0	56	504	3896	420	3780	29220	37876
70	-1 -1 -1 -1 +1 +1 +1 +1	0	0	0	64	576	4384	480	4320	32880	42704

The generators that have the lowest TI are: $[-1 \quad -1 \quad +1 \quad +1 \quad -1 \quad +1 \quad +1 \quad -1]$ (row #31) and $[+1 \quad +1 \quad -1 \quad -1 \quad +1 \quad -1 \quad -1 \quad +1]$ (row #40), each with a total TI value of 816 across all trends. These generators are completely resistant to L × L, L × Q, and L × C. All other trend combinations are nearly trend resistant. If compared to the generator: $[+1 \quad -1 \quad -1 \quad +1 \quad -1 \quad +1 \quad +1 \quad -1]$ (row #29), which has a total TI of 1,632, it is clear that the generators 31 and 40 have better TI. On the other hand, generator 29 is completely robust to more trends: L × L, L × Q, L × C, Q × L, Q × Q, C × L, and C × Q. The Q × C and C × C trends have a respective TI value of 192 and 1440. The generators 31 and 40, however, had a TI value of 16 and 120 on the Q × C and C × C trends respectively. It lies on the priorities of the experimenter to decide which generator to choose. It is possible that the objective of the design is to be completely robust to most trends. On the other hand, a nearly trend resistant design might be more convenient. Unfortunately, the fold-over method cannot target particular trends. This method may not be convenient when the experiment is possibly affected by higher order trends. The trend index does not have to be the only metric used to select appropriate generators. It is possible to include as a metric the number of trends the factor is completely robust against. By using the TI as the metric for selecting the best $2^2 \times 2^3$ design will have the following generators for the three sub-plot factors (denoted here as A, B, and C):

Factor A = $[-1 \quad -1 \quad +1 \quad +1 \quad -1 \quad +1 \quad +1 \quad -1]$(row #31),

Factor B = $[-1 \quad +1 \quad +1 \quad +1 \quad -1 \quad -1 \quad -1 \quad +1]$ (row #39), and

Factor C = $[+1 \quad -1 \quad +1 \quad -1 \quad -1 \quad +1 \quad -1 \quad +1]$ (row #47).

Table 6.3 shows a detailed look at the trend resistance of this design. This design has a total TI of 11,900. This design was chosen by selecting the best TI for factor A,

then the next possible generator for factor B that is orthogonal to factor A, and finally a generator for factor C which is orthogonal to both factors A and B. The table also includes the number of trends that are resisted for each factor. Factor A has the lowest TI value of 816 and is completely resistant to the three trends: L × L, L × Q, and L × C. Factor B has a TI Value of 3196 and is also resistant to the L × L, L × Q, and L × C trends. Factor C has the highest TI value at 7888, but is resistant to five trends, L × L, L × Q, L × C, Q × L, and C × L.

Table 6.3: Fold-over results for a $2^2 \times 2^3$ design selected with the TI metric.

Factor	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	TI	Resisted Trends
A	0	0	0	16	64	16	120	480	120	816	3
B	0	0	0	8	72	296	60	540	2220	3196	3
C	0	0	0	0	64	864	0	480	6480	7888	5
									TOTAL	11900	11

Instead, if the metric used for selecting robust generators is the number of trends to which the design is resistant, then the generators for the factors in the $2^2 \times 2^3$ design will be:

Factor A = $\begin{bmatrix}+1 & -1 & -1 & +1 & -1 & +1 & +1 & -1\end{bmatrix}$ (row #29),

Factor B = $\begin{bmatrix}+1 & -1 & -1 & -1 & +1 & +1 & +1 & -1\end{bmatrix}$ (row #32), and

Factor C = $\begin{bmatrix}-1 & -1 & +1 & +1 & +1 & -1 & +1 & -1\end{bmatrix}$ (row #25).

Similarly factor A is selected, by selecting the generator that has the largest number of completely robust trends (highest number of zeros). In the case of having two possible generators with the same number of robust trends, then the generator with the best TI is selected. The generator for factor B is then selected such that it is orthogonal to the one selected for factor A. The process is repeated for factor C such that its generator is orthogonal to both factor A's and B's generator. Table 6.4 shows the detailed results

for this selection method. Compared to the method that used the TI metric, this method has a slightly lower TI. Above all, the number of robust trends available is increased to 13. The experimenter might prefer this method; then again, it all depends on the trends that the experimenter is more concerned about.

Table 6.4: Fold-over results for a $2^2 \times 2^3$ design selected with the robust trend metric

Factor	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	TI	Resisted Trends
A	0	0	0	0	0	192	0	0	1440	1632	7
B	0	0	0	8	72	296	60	540	2220	3196	3
C	0	0	0	8	24	712	60	180	5340	6324	3
									TOTAL	**11152**	**13**

The fold-over method will naturally make designs robust to trends as they are folded. The more folds are performed in a design, the more robust it will be to higher degree trends. The selection method hides other possible options. The heuristic developed here follows a greedy methodology that does not guarantee an optimal solution. There is no control over which trends the design is robust against. The fold-over method does not target trends. If there is a particular trend that needs to be designed against, it is possible for the fold-over method to generate a design that might be highly correlated to that trend, but robust to many other trends of, perhaps, less interest.

6.7 Split-plot designs with 5 or more sub-plot factors

Designs with 5 sub-plot factors have a total of 601,080,390 possible generators for the fold-over design. To enumerate this amount may not be practical. This number increases exponentially as more sub-plot factors are added into the design. There are 1.8326×10^{18} possible generators for 6 sub-plot factors.

Folding over an array of contrasts will improve the robustness of the design to higher degree trends. A suggested approach for split-plot designs with more than 5 sub-

plot factors is to choose the generators from a design with $s-1$ sub-plot factors and fold those generators into the first row for s sub-plot factors which can later be folded into the full split-plot design. This approach does not guarantee an optimal solution, and there might be a better design available if the design is selected with the proposed method in section 6.5, which selects a design from a full list of generators.

When a split-plot design has more than 5 sub-plot factors, it might be more practical to use another method for systematically selecting the run order of the experiment, such as the integer linear programming approach explained in the next section.

7 Integer Linear Programming Approach

7.1 Definition

Linear Programming (LP) is a tool used to find the optimal solution to a problem. Unfortunately, linear programming is not capable of solving problems with nonlinear equations. In the case of the Trend Index, the objective of the design is to minimize the absolute value of the Trend Index: $|TI|$, which is a nonlinear relationship. However, this nonlinear programming (NLP) problem can be modified into the standard form of Linear Program for easier calculation. A Non-Linear model for developing a trend resistant split-plot design will yield designs without the need of going through all of the possible combinations of designs.

The fold-over method will only explore a fraction of the total population of possible designs. Exploring the total population of designs will be too exhaustive in comparison to a NLP approach. The goal is to make TI = 0, or at best approach 0, for all trends that are considered in the experiment.

7.2 The Model

The method used in this work to calculate D^{α} for all factors (α), uses a combined variable that represents the treatment level combination for all the factors. This reduces the number of constraints in the experiment and the complication that extends from using D^{α}. There are only 2^s different combinations of contrasts in a design with s sub-plot factors. In a design with 2^w rows, the model should arrange the permutations in order to

minimize the Trend Index for that factor. The number of possibilities in which the treatment level combinations (TLC) can be ordered is calculated by: $N = 2^s!$. As s increases, the number of possible solutions will grow exponentially and the model will take longer to solve.

The model needs an initial preset order for the treatment level combinations. This order can be generated by any means as long as the same order is used when interpreting the results. This thesis uses the Yates' order because of its simplicity and ease of use. If there are 3 sub-plot factors, then the treatment level combinations will be: $[(1) \quad a \quad b \quad ab \quad c \quad ac \quad bc \quad abc]$. This can be interpreted in the following way:

$$\begin{aligned}
&[-1 \quad +1 \quad -1 \quad +1 \quad -1 \quad +1 \quad -1 \quad +1] \text{ for factor A} \\
&[-1 \quad -1 \quad +1 \quad +1 \quad -1 \quad -1 \quad +1 \quad +1] \text{ for factor B} \\
&[-1 \quad -1 \quad -1 \quad -1 \quad +1 \quad +1 \quad +1 \quad +1] \text{ for factor C}
\end{aligned}$$

Let V' be the matrix of the contrast of size $s \times 2^s$ for factor α in TLC k; therefore, for 3 sub-plot factors (α) with 8 treatment level combinations (k),

$$\mathbf{V}' = \begin{bmatrix} -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \\ -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 \end{bmatrix}.$$

To avoid the use of negative numbers in the NLP, the representation of the low level of a factor is replaced from a -1 to a 0. The model would use constant $V_{\alpha k}$ as a 0 or 1 which represents the level of the factor α in permutation k whether the contrast is low or high respectively. Figure 7.1 demonstrates how the values of V are arranged according to the treatment level combination.

$$\mathbf{V} = \begin{array}{cl} \xrightarrow{\qquad\qquad k \qquad\qquad} & \\ \begin{array}{cccccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} \end{array} & \\ \left[\begin{array}{cccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right] & \begin{array}{l} \text{Factor A} \\ \text{Factor B} \\ \text{Factor C} \end{array} \; \alpha \downarrow \\ \begin{array}{cccccccc} (1) & a & b & ab & c & ac & bc & abc \end{array} & \end{array}$$

Figure 7.1: Treatment level combinations used in the ILP

The objective function is based on the Trend Index:

$$TI^{\alpha} = \left| \sum_{i=1}^{2^w} \sum_{j=1}^{2^s} D_{i,j}^{\alpha} \times \tau_{i,j} \right|.$$

The NLP needs to select the contrast for the factors in every row and column on the design so as to minimize TI. Let

$$X_{ijk} = \begin{cases} 1, \text{if TLC } k \text{ appears in row } i \text{ and column } j. \\ 0, \text{otherwise.} \end{cases}, \text{ where } k = \begin{bmatrix} 1 & 2 & \ldots & 2^s \end{bmatrix}$$

and each k refers to a different treatment level combination whose factor's value can be found in $V_{\alpha k}$. The TI will be calculated with the following formula:

$$Z = \sum_{\alpha=1}^{s} \left| TI^{\alpha} \right| = \sum_{\alpha=1}^{s} \left| \sum_{i=1}^{2^w} \sum_{j=1}^{2^s} \tau_{i,j} \sum_{k=1}^{2^k} V_{\alpha k} X_{ijk} - \sum_{i=1}^{2^w} \sum_{j=1}^{2^s} \tau_{i,j} \sum_{k=1}^{2^s} (1 - V_{\alpha k}) X_{ijk} \right|.$$

The first term, $\sum_{i=1}^{2^w} \sum_{j=1}^{2^s} \tau_{i,j} \sum_{k=1}^{2^k} V_{\alpha k} X_{ijk}$, is the summation of trend index where factor α is at its high level (V = 1), and the second term, $\sum_{i=1}^{2^w} \sum_{j=1}^{2^s} \tau_{i,j} \sum_{k=1}^{2^s} (1 - V_{\alpha k}) X_{ijk}$, is the summation of the trend index where factor α is at its low level (V = 0).

As far as the model is concerned, the objective function will arrange the contrast combinations along the rows and columns to make the design be robust to a particular trend τ. All the model needs is a set of constraints that will validate the model.

To prevent the same contrast combination from being used twice in the same row, the following constraint will be used:

$$\sum_{j=1}^{2^s} X_{ijk} = 1, \text{ for all } i \text{ and } k .$$

Afterwards, only one permutation can be present in one position (i, j), therefore, the constraint used is:

$$\sum_{k=1}^{2^s} X_{ijk} = 1, \text{ for all } i \text{ and } j .$$

When considering multiple trends, the model will be modified so that each factor can be robust to each trend. The new objective function will be:

$$Z = \sum_{\phi=1}^{t} \sum_{\alpha=1}^{s} \left| TI_{\phi}^{\alpha} \right| = \sum_{\phi=1}^{t} \sum_{\alpha=1}^{s} \left| \sum_{i=1}^{2^w} \sum_{j=1}^{2^s} \tau_{i,j,\phi} \sum_{k=1}^{2^k} V_{\alpha k} X_{ijk} - \sum_{i=1}^{2^w} \sum_{j=1}^{2^s} \tau_{i,j,\phi} \sum_{k=1}^{2^s} (1 - V_{\alpha k}) X_{ijk} \right| ,$$

where $\phi \in [1 \quad 2 \quad ... \quad t]$ are the two-dimensional trends in question. The different trends included in $\tau_{i,j,\phi}$ can be L×L, Q×L, C×Q, etc., where each trend corresponds to one ϕ.

Running this program for multiple trends might not yield a result that is completely robust to all trends. As it is, the objective function will consider all trends to be of equal importance. The best solution might give very low values of TI for every trend. In this case the model would have to weigh all the trends. Therefore, if the linear × linear trend needs to be completely robust (TI = 0), then this trend would need to be weighted more than the other trends.

For this purpose the constant W_{ϕ} is introduced to the objective function. This constant represents the weight that a particular trend has in the design. The experimenter should decide the level of importance of every trend. The trends that have the most

influence on the design should have a larger weight than those with little influence. The purpose is to achieve a TI = 0 on the highly influential trends and return a low value of TI on the trends with low influence. The new objective function is as follows:

$$Z=\sum_{\phi=1}^{t}\sum_{\alpha=1}^{s}\left|W_{\phi}TI_{\phi}^{\alpha}\right|=\sum_{\phi=1}^{t}\sum_{\alpha=1}^{s}\left|\sum_{i=1}^{2^{w}}\sum_{j=1}^{2^{s}}W_{\phi}\tau_{i,j,\phi}\sum_{k=1}^{2^{k}}V_{\alpha k}X_{ijk}-\sum_{i=1}^{2^{w}}\sum_{j=1}^{2^{s}}W_{\phi}\tau_{i,j,\phi}\sum_{k=1}^{2^{s}}(1-V_{\alpha k})X_{ijk}\right|.$$

7.3 The Integer Linear Model

Any objective function involving a positive absolute value of a linear function can be modeled linearly. By introducing new variables expressing the linear function will convert the NLP into an Integer Linear Program (ILP).

The new variables added to the model are:

$s_{\phi,\alpha}^{+}\equiv$ positive magnitude of the TI of trend ϕ and factor α

$s_{\phi,\alpha}^{-}\equiv$ negative magnitude of the TI of trend ϕ and factor α

The new objective function added to the model is:

$$Z=\sum_{\phi=1}^{t}\sum_{\alpha}^{s}s_{\phi,\alpha}^{+}+s_{\phi,\alpha}^{-}\ .$$

The new set of constraints is:

$$\sum_{i=1}^{2^{w}}\sum_{j=1}^{2^{s}}\tau_{i,j,\phi}\sum_{k=1}^{2^{k}}V_{\alpha k}X_{ijk}-\sum_{i=1}^{2^{w}}\sum_{j=1}^{2^{s}}\tau_{i,j,\phi}\sum_{k=1}^{2^{s}}(1-V_{\alpha k})X_{ijk}=s_{\phi,\alpha}^{+}-s_{\phi,\alpha}^{-}\ .$$

After these modifications, the model can be solved by traditional linear programming methods.

When considering the weighted objective function, the constant W_{ϕ} is introduced in the new set of constraints like shown here:

$$\sum_{i=1}^{2^w}\sum_{j=1}^{2^s} W_\phi \tau_{i,j,\phi} \sum_{k=1}^{2^k} V_{\alpha k} X_{ijk} - \sum_{i=1}^{2^w}\sum_{j=1}^{2^s} W_\phi \tau_{i,j,\phi} \sum_{k=1}^{2^s} (1 - V_{\alpha k}) X_{ijk} = s^+_{\phi,\alpha} - s^-_{\phi,\alpha}.$$

The weight constant can also be introduced in the objective function instead of the constraints: $Z = \sum_{\phi=1}^{t}\sum_{\alpha}^{s} W_\phi (s^+_{\phi,\alpha} + s^-_{\phi,\alpha})$. Whether the weight ($W_\phi$) is added to the objective function or the constraints, it would yield the same results. In this thesis, the weight constant is added to the constraints but the experimenter can choose to add it in the objective function if they wish.

The ILP model used in this thesis will be the following:

Minimize:

$$Z = \sum_{\phi=1}^{t}\sum_{\alpha}^{s} (s^+_{\phi,\alpha} + s^-_{\phi,\alpha}) \tag{1}$$

Subject to:

$$\sum_{i=1}^{2^w}\sum_{j=1}^{2^s} W_\phi \tau_{i,j,\phi} \sum_{k=1}^{2^k} V_{\alpha k} X_{ijk} - \sum_{i=1}^{2^w}\sum_{j=1}^{2^s} W_\phi \tau_{i,j,\phi} \sum_{k=1}^{2^s} (1 - V_{\alpha k}) X_{ijk} = s^+_{\phi,\alpha} - s^-_{\phi,\alpha}, \text{ for all } \phi \text{ and } \alpha \tag{2}$$

$$\sum_{k=1}^{2^s} X_{ijk} = 1, \text{ for all } i \text{ and } j \tag{3}$$

$$\sum_{j=1}^{2^s} X_{ijk} = 1, \text{ for all } i \text{ and } k \tag{4}$$

$$X_{ijk} \in \{0,1\}, \text{ for all } i, j, \text{ and } k$$

$$s^+_{\phi,\alpha} \geq 0, \text{ for all } \phi \text{ and } \alpha$$

$$s^-_{\phi,\alpha} \geq 0, \text{ for all } \phi \text{ and } \alpha$$

7.4 Complexity

This model is NP hard and nonlinear. It can be modeled as a linear model with ease, as explained in the previous section. Considering just the NLP, the number of constraints in this model is 2^{s+w+1}. The number of variables for this model is 2^{2s+w}. The ILP model, however, has $2^{s+w+1} + 4(s \times t)$ variables and $2^{2s+w} + (s \times t)$ constraints. This is an exponentially growing problem and therefore NP hard. This means that as the problem becomes larger, the time it takes to solve the problem, would grow exponentially. With large NP hard problems like this, there would be a size that would make it impractical to solve the problem.

The LP model will have a slight increase in the number of constraints and variables. This increase will depend on the number of trends that the model has to be robust against.

7.5 Setup

The ILP model was tested to see how well it could reach an optimal solution. The model was programmed into OPL and then run with several options. The computers that ran the OPL model had a Windows XP operating system with an Intel(R) Pentium(R) 4 CPU 3.20GHz (2 CPUs)processor and 1014 MB RAM.

OPL uses the branch and cut algorithm for solving integer linear programming. It is a hybrid method that combines the branch and bound and the cutting plane methods. First, OPL will solve the problem using the simplex algorithm without the integer constraints. When a non-integer solution is obtained, OPL uses a cutting plane algorithm to find linear constraints that satisfy the feasible integer points. These new constraints are

added to the original problem and the process repeats itself until an integer solution is found.

The run time for each setup was dependant on the number of factors. If the solution reached a TI value of 0, then OPL would stop because there is no better solution than this. This truncates the problem substantially, which also makes it harder to estimate the time it will take to find a solution.

The setup variables that changed during the testing were: the number of sub-plot factors, the number of whole-plot factors, and the two-dimensional trend combinations. The purpose for this testing was to verify the validity of the model, and to demonstrate that the ILP model can be used to find robust designs.

The amount of data inputted into OPL can become large and for this reason the arrangements of the setup were done in such a way to reduce the changes in the data file. Changing the sub-plot factor from 2 to 3 usually required the most amount of change in the data file. The arrangement was done in a way so as to minimize the change of the number of factors.

The trends used for these tests were multiplicative combinations of linear, quadratic and cubic trends for a total of nine two-dimensional trends. For simplicity, the trends were coded as L, Q, and C for linear, quadratic, and cubic respectively. The two-dimensional trend was coded as whole-plot trend × sub-plot trend. As before, with this coding, L×Q referred to the two-dimensional trend generated by a linear trend on the whole-plot factors and a quadratic trend on the sub-plot factors.

For this test, the linear trend was generated by using the formula: $y = x$. The equations $y = x^2$ and $y = x^3$ were used for the quadratic and cubic trends respectively.

The setup designs were tested in two stages. The first tested the ILP model with a single two-dimensional trend and up to three factors in the sub-plot and three factors in the whole-plots. The two-dimensional trend interactions used were: L×L, L×Q, L×C, Q×L, Q×Q, Q×C, C×L, C×Q, and C×C. Table 7.1 shows the different possible tests done for this stage.

Table 7.1: First stage test run for the ILP model

Design #	Design	Trend	Design #	Design	Trend
1	$2^2 \times 2^2$	L×L	**19**	$2^2 \times 2^3$	L×L
2	$2^2 \times 2^2$	L×Q	**20**	$2^2 \times 2^3$	L×Q
3	$2^2 \times 2^2$	L×C	**21**	$2^2 \times 2^3$	L×C
4	$2^2 \times 2^2$	Q×L	**22**	$2^2 \times 2^3$	Q×L
5	$2^2 \times 2^2$	Q×Q	**23**	$2^2 \times 2^3$	Q×Q
6	$2^2 \times 2^2$	Q×C	**24**	$2^2 \times 2^3$	Q×C
7	$2^2 \times 2^2$	C×L	**25**	$2^2 \times 2^3$	C×L
8	$2^2 \times 2^2$	C×Q	**26**	$2^2 \times 2^3$	C×Q
9	$2^2 \times 2^2$	C×C	**27**	$2^2 \times 2^3$	C×C
10	$2^3 \times 2^2$	L×L	**28**	$2^3 \times 2^3$	L×L
11	$2^3 \times 2^2$	L×Q	**29**	$2^3 \times 2^3$	L×Q
12	$2^3 \times 2^2$	L×C	**30**	$2^3 \times 2^3$	L×C
13	$2^3 \times 2^2$	Q×L	**31**	$2^3 \times 2^3$	Q×L
14	$2^3 \times 2^2$	Q×Q	**32**	$2^3 \times 2^3$	Q×Q
15	$2^3 \times 2^2$	Q×C	**33**	$2^3 \times 2^3$	Q×C
16	$2^3 \times 2^2$	C×L	**34**	$2^3 \times 2^3$	C×L
17	$2^3 \times 2^2$	C×Q	**35**	$2^3 \times 2^3$	C×Q
18	$2^3 \times 2^2$	C×C	**36**	$2^3 \times 2^3$	C×C

The second stage considered all nine two-dimensional trends at the same time. This introduced the weighted objective function into the ILP model. For every $2^2 \times 2^2$, $2^2 \times 2^3$, $2^3 \times 2^2$, and $2^3 \times 2^3$ split-plot design, the trends were weighted equally or systematically weighted. The equal weights meant that each two-dimensional trend had the same consideration and that the objective of the experimenter is to have the lowest TI amongst all the trends. This, of course, would yield results for nearly trend-resistant

designs. The weighted trends are organized in a systematic order. Each trend is weighted with a value obtained by the series:

$$W_\phi = \left(\frac{1}{2}\right)^\phi$$

where ϕ serves as the priority of the trend, as shown in table 7.2. On a note, the smallest weights are slightly modified so that the weights sum to 1. These weights mean that the first trend is twice as important of the second trend, which in turn is twice as important as the third trend and so forth.

Table 7.2: The weights on the trends for stage three

Priority	**TREND**	**WEIGHT**
1	L×L	0.500
2	L×Q	0.250
3	Q×L	0.125
4	Q×Q	0.063
5	L×C	0.032
6	C×L	0.016
7	Q×C	0.008
8	C×Q	0.004
9	C×C	0.002
Summation		1.000

7.6 Results

Table 7.3 shows the results of running all 36 designs in the OPL program. During stage one, 27 of the designs reached an optimal solution in less than 10 seconds. On the other hand, the much larger designs, such as the $2^3 \times 2^3$ with the C×Q trend required 29.60 hours to obtain a solution. The major factor that determined the run time was whether the optimal solution for the objective function reached 0 or not. If the best available trend resistant design has a TI = 0, OPL can reach it and truncates the rest of the iterations. The TI cannot be better than 0. It is quite possible that there are situations where there are no trend resistant designs with TI = 0. In these cases the best solution obtained is a nearly

trend-resistant design. The branch and cut algorithm that OPL employs will have to iterate until the solution reaches an optimal TI. Designs larger than $2^3 \times 2^3$ designs ran for more than 24 hours before stopping at an optimum solution.

Table 7.3: Stage 1 results from OPL

Design #	Split-plot Design	Trend	Best TI	Solving Time (seconds)	Iterations	Number of Constraints	Number of Variables
1	$2^2 \times 2^2$	L×L	0	0.03	117	35	69
2	$2^2 \times 2^2$	L×Q	0	0.02	73	35	69
3	$2^2 \times 2^2$	L×C	0	0.01	141	35	69
4	$2^2 \times 2^2$	Q×L	0	0.03	339	35	69
5	$2^2 \times 2^2$	Q×Q	0	0.02	108	35	69
6	$2^2 \times 2^2$	Q×C	2	0.08	1526	35	69
7	$2^2 \times 2^2$	C×L	0	0.03	131	35	69
8	$2^2 \times 2^2$	C×Q	8	0.2	5430	35	69
9	$2^2 \times 2^2$	C×C	200	0.03	127	35	69
10	$2^3 \times 2^2$	L×L	0	0.03	78	67	133
11	$2^3 \times 2^2$	L×Q	0	0.06	416	67	133
12	$2^3 \times 2^2$	L×C	0	0.13	2105	67	133
13	$2^3 \times 2^2$	Q×L	0	0.06	256	67	133
14	$2^3 \times 2^2$	Q×Q	0	0.19	3883	67	133
15	$2^3 \times 2^2$	Q×C	0	0.19	4490	67	133
16	$2^3 \times 2^2$	C×L	0	0.28	7538	67	133
17	$2^3 \times 2^2$	C×Q	0	3.58	127442	67	133
18	$2^3 \times 2^2$	C×C	2	735.98	25375477	67	133
19	$2^2 \times 2^3$	L×L	0	0.14	917	68	263
20	$2^2 \times 2^3$	L×Q	0	0.41	3405	68	263
21	$2^2 \times 2^3$	L×C	0	0.34	5284	68	263
22	$2^2 \times 2^3$	Q×L	0	4.42	132081	68	263
23	$2^2 \times 2^3$	Q×Q	0	4.7	148531	68	263
24	$2^2 \times 2^3$	Q×C	0	5.45	135143	68	263
25	$2^2 \times 2^3$	C×L	0	2492.56	66772562	68	263
26	$2^2 \times 2^3$	C×Q	0	275.94	8822403	68	263
27	$2^2 \times 2^3$	C×C	0	7056.23	225960528	68	263
28	$2^3 \times 2^3$	L×L	0	0.45	2237	156	567
29	$2^3 \times 2^3$	L×Q	0	0.7	4464	156	567
30	$2^3 \times 2^3$	L×C	0	1.41	13633	156	567
31	$2^3 \times 2^3$	Q×L	0	4.53	69828	156	567
32	$2^3 \times 2^3$	Q×Q	0	108.95	1901766	132	519
33	$2^3 \times 2^3$	Q×C	0	13.27	207810	156	567
34	$2^3 \times 2^3$	C×L	0	604.53	11233545	156	567
35	$2^3 \times 2^3$	C×Q	0	106554.28	263040078	156	567
36	$2^3 \times 2^3$	C×C	0	1022.55	17487706	156	567

As shown in Figure 7.2, the time it takes to complete the run increases as both the size of the design increases and the degree of the trend increases. With the same design, the higher the degree of the trends, the longer time or greater number of iterations it required to achieve a result.

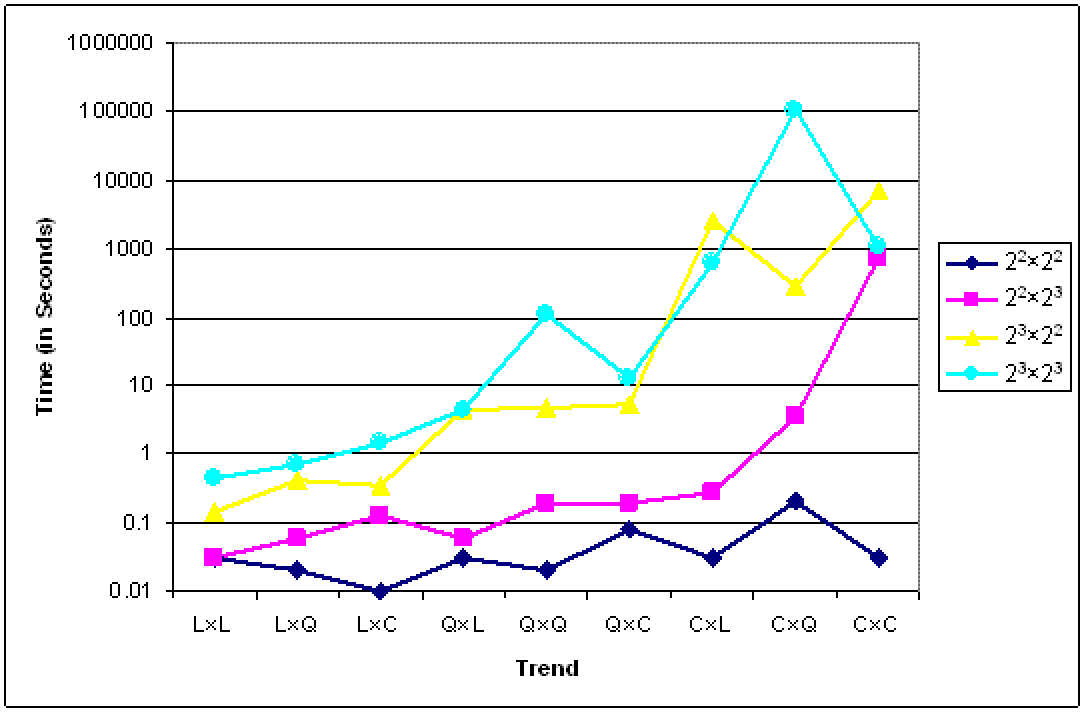


Figure 7.2: Graph of completion time vs. trends in OPL for each design

The best solution in table 7.3 is the TI for the resulting design with the modeled trend. Design #9, $2^2 \times 2^2$ modeled with the two-dimensional trend C × C, yielded a TI = 200. This is the best solution that OPL could obtain from the ILP model. This design's order was selected to be robust to the C × C trend without considering any other trend.

Table 7.4 shows how the designs obtained in stage one compare to the effects of other trends that were not programmed into the ILP model. The resulting designs are robust or nearly robust to the trends used in the model. However, the designs do poorly on the other trends, especially on higher degree trends. If an experiment is believed to be

affected by two or more trends, then it is necessary to include them in the model such as the designs obtained in stage two.

Table 7.4: Trend Index for the designs obtained with the ILP method

#	Design	Factor	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	TI
			TRENDS									
1	$2^2 \times 2^2$	Factor 1:	0	4	24	16	84	376	132	592	2496	8652
		Factor 2:	0	20	108	12	136	648	96	716	3192	
2	$2^2 \times 2^2$	Factor 1:	4	0	44	20	0	220	76	24	1016	2560
		Factor 2:	0	0	0	0	16	120	0	120	900	
3	$2^2 \times 2^2$	Factor 1:	0	0	0	8	8	112	60	60	840	1956
		Factor 2:	12	16	0	60	80	0	240	292	168	
4	$2^2 \times 2^2$	Factor 1:	0	8	60	0	40	300	0	176	1320	2716
		Factor 2:	4	20	76	0	32	120	68	100	392	
5	$2^2 \times 2^2$	Factor 1:	12	20	36	32	0	184	36	340	1992	3832
		Factor 2:	4	4	20	8	0	136	20	116	872	
6	$2^2 \times 2^2$	Factor 1:	6	14	30	26	42	2	90	74	498	1650
		Factor 2:	12	16	0	60	80	0	240	292	168	
7	$2^2 \times 2^2$	Factor 1:	0	0	0	0	16	120	0	120	900	3060
		Factor 2:	0	8	60	0	40	300	0	176	1320	
8	$2^2 \times 2^2$	Factor 1:	8	8	16	32	24	112	92	4	760	2004
		Factor 2:	4	8	16	20	16	64	88	4	728	
9	$2^2 \times 2^2$	Factor 1:	16	36	88	64	116	184	220	288	32	1912
		Factor 2:	12	16	0	60	80	0	240	292	168	
10	$2^3 \times 2^2$	Factor 1:	0	12	372	4	0	1096	36	120	2784	9028
		Factor 2:	0	100	996	40	164	2324	324	560	96	
11	$2^3 \times 2^2$	Factor 1:	4	0	56	60	504	3840	548	3936	28688	42056
		Factor 2:	8	0	388	76	204	824	452	1620	848	
12	$2^3 \times 2^2$	Factor 1:	12	56	0	44	120	1264	132	88	10176	29808
		Factor 2:	12	24	0	44	72	1792	108	1128	14736	
13	$2^3 \times 2^2$	Factor 1:	4	12	76	0	40	408	32	336	3488	43892
		Factor 2:	4	104	1196	0	620	6480	56	2852	28184	
14	$2^3 \times 2^2$	Factor 1:	8	36	76	20	0	1720	20	648	12304	25296
		Factor 2:	8	36	652	76	0	2288	428	552	6424	
15	$2^3 \times 2^2$	Factor 1:	24	108	384	68	232	476	180	0	3708	8176
		Factor 2:	16	60	172	64	220	532	100	0	1832	
16	$2^3 \times 2^2$	Factor 1:	12	136	1272	28	544	5752	0	1756	21492	44524
		Factor 2:	24	124	660	52	152	152	0	836	11532	
17	$2^3 \times 2^2$	Factor 1:	0	0	60	4	36	56	0	0	1500	10080
		Factor 2:	4	96	940	20	208	2620	212	0	4324	
18	$2^3 \times 2^2$	Factor 1:	38	158	698	158	534	1442	566	1478	2	8450
		Factor 2:	0	56	636	48	16	1308	348	964	0	
19	$2^2 \times 2^3$	Factor 1:	0	16	60	48	32	204	612	1220	5880	58636
		Factor 2:	0	56	312	104	16	880	1392	2872	5400	
		Factor 3:	0	68	444	36	552	3888	384	4148	30012	
20	$2^2 \times 2^3$	Factor 1:	12	0	108	64	212	1928	444	2280	18540	75400
		Factor 2:	0	0	0	96	224	576	1488	3792	10848	
		Factor 3:	16	0	152	64	296	2408	188	4668	26996	
21	$2^2 \times 2^3$	Factor 1:	24	36	0	284	592	1004	2832	6924	16344	70920
		Factor 2:	48	64	0	288	96	1728	1392	1808	21984	
		Factor 3:	12	16	0	44	64	712	168	2324	12132	
22	$2^2 \times 2^3$	Factor 1:	8	8	16	0	120	744	556	2692	11956	33964
		Factor 2:	12	52	216	0	144	876	480	296	2532	
		Factor 3:	12	36	156	0	80	144	540	2748	9540	
23	$2^2 \times 2^3$	Factor 1:	8	12	16	68	0	748	440	1068	11392	37608
		Factor 2:	24	56	168	96	0	384	96	2752	13440	
		Factor 3:	4	40	224	156	0	840	1720	1388	2468	

#	Design	Factor	TRENDS									TI
			L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	
24	$2^2 \times 2^3$	Factor 1:	8	16	44	60	60	0	668	604	736	20000
		Factor 2:	8	20	148	180	168	0	2048	3436	8012	
		Factor 3:	20	36	44	120	176	0	788	1356	1244	
25	$2^2 \times 2^3$	Factor 1:	24	8	120	104	224	1888	0	3904	20736	57204
		Factor 2:	0	8	72	16	112	208	0	1192	3816	
		Factor 3:	24	32	24	92	148	1348	0	3664	19440	
26	$2^2 \times 2^3$	Factor 1:	24	36	60	196	128	584	1476	0	10728	22636
		Factor 2:	24	48	108	172	236	88	816	0	6372	
		Factor 3:	4	24	112	36	168	768	52	0	376	
27	$2^2 \times 2^3$	Factor 1:	36	116	360	244	628	1384	1476	2492	0	10116
		Factor 2:	0	28	120	56	76	472	840	548	0	
		Factor 3:	0	16	72	32	64	400	432	224	0	
28	$2^3 \times 2^3$	Factor 1:	0	108	840	148	40	1268	2988	12984	75948	184960
		Factor 2:	0	96	1476	116	164	9112	1920	4344	38940	
		Factor 3:	0	268	2544	96	1172	11544	1488	748	16608	
29	$2^3 \times 2^3$	Factor 1:	16	0	212	652	3244	20224	8188	45252	294448	506400
		Factor 2:	16	0	320	208	16	4160	2200	1152	29312	
		Factor 3:	4	0	776	64	252	6772	1688	7428	79796	
30	$2^3 \times 2^3$	Factor 1:	144	368	0	1376	3832	4712	11628	36332	90036	433888
		Factor 2:	36	76	0	72	704	8556	744	12344	108996	
		Factor 3:	24	132	0	520	2700	8152	5568	29052	107784	
31	$2^3 \times 2^3$	Factor 1:	24	108	1884	0	2708	29676	1176	30684	302316	708760
		Factor 2:	12	288	2808	0	1244	13020	912	12	17940	
		Factor 3:	68	220	1028	0	1608	13944	3544	33920	249616	
32	$2^3 \times 2^3$	Factor 1:	120	316	372	620	0	15664	2676	12440	201048	483832
		Factor 2:	32	124	332	196	0	7304	812	6848	116392	
		Factor 3:	60	172	420	264	0	5016	324	10220	102060	
33	$2^3 \times 2^3$	Factor 1:	80	152	184	528	928	0	3104	5456	15152	200620
		Factor 2:	8	196	1448	288	276	0	4576	14420	71680	
		Factor 3:	32	164	1304	24	268	0	1492	12616	66244	
34	$2^3 \times 2^3$	Factor 1:	36	400	3384	96	2628	26244	0	18916	208908	771280
		Factor 2:	48	684	6360	216	4692	46176	0	25476	277248	
		Factor 3:	24	336	2808	56	2080	18128	0	13104	113232	
35	$2^3 \times 2^3$	Factor 1:	48	24	1020	496	104	11396	4176	0	104604	354300
		Factor 2:	40	216	1564	132	836	8712	320	0	36124	
		Factor 3:	76	144	668	660	504	15084	5068	0	162284	
36	$2^3 \times 2^3$	Factor 1:	12	52	192	16	72	1508	180	364	0	61140
		Factor 2:	108	512	2712	792	3140	12468	4524	11840	0	
		Factor 3:	84	404	2088	500	2180	9368	2028	5996	0	

Table 7.5 shows the results for the designs that ran on stage two where the ILP was run with all 9 two-dimensional trends for each design. The non-weighted models achieved a better overall TI. Unfortunately, these solutions are not completely trend resistant. Most factors are nearly trend resistant, but few factors had a TI = 0. The weighted models contained more completely robust trends. The most important, or higher

weighted, trends received the best considerations and achieved trend resistance. On the other hand, the least important trends managed to be nearly-trend resistant. Although the weighted design obtained the most robust trends, the total TI was larger than those in the designs that are not weighted. For example, factor 1 in the $2^2 \times 2^3$ design has 1 robust trend when it is not weighed, but has 7 robust trends when it is weighted. The trends on the left are weighted higher than those on the right, which means that these are more important to be robust against. As shown, the C × C trend has a TI of 1440 in the weighted design, which is considerably higher than the TI of 16 that is obtained with the design that is not weighted.

Table 7.5: Trend Index for the designs used in stage two

Design #	Design	Trends	Factor	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	TI
1	$2^2 \times 2^2$	Not Weighted	Factor 1:	4	16	52	12	32	36	28	8	536	1376
			Factor 2:	8	16	8	40	80	40	152	280	28	
2	$2^2 \times 2^2$	Weighted	Factor 1:	2	6	14	6	2	78	14	78	802	1702
			Factor 2:	0	0	24	8	16	88	48	72	444	
3	$2^2 \times 2^3$	Not Weighted	Factor 1:	8	8	4	24	0	132	68	4	16	758
			Factor 2:	6	2	18	14	38	34	6	346	30	
4	$2^2 \times 2^3$	Weighted	Factor 1:	0	0	0	0	0	192	0	0	1440	2760
			Factor 2:	0	0	0	0	0	0	24	144	960	
5	$2^3 \times 2^2$	Not Weighted	Factor 1:	0	4	0	16	76	184	0	124	0	1418
			Factor 2:	10	18	34	26	2	94	10	126	34	
			Factor 3:	8	0	52	28	20	224	20	132	176	
6	$2^3 \times 2^2$	Weighted	Factor 1:	0	0	0	0	0	96	96	192	912	4322
			Factor 2:	2	2	2	6	30	6	2	2	1442	
			Factor 3:	0	0	12	4	12	64	36	132	1272	

7.7 Analysis

When modeled to find the design with the minimum TI for a single trend, the ILP model obtained a TI = 0 on most designs and nearly-robust designs on the higher degree trends, such as the C × C trend on the $2^2 \times 2^2$ model with a TI = 200. When considering

multiple trends the weighted model yielded TI = 0 in most factors and trends. The non-weighted models yielded very few TI = 0, although they have the lowest overall TI.

Since the ILP method searches through the possible designs and truncates those designs that have worse solutions than the upper bound, it is safe to assume that using un-weighted trends in the model will yield the arrangement with the lowest TI of the design. No other arrangement for the same design can have a lower TI than the one obtained with un-weighted trends.

The downside for the ILP method is the time it takes to achieve a solution. Currently, only small models can be solved. Larger designs will take weeks to complete. The duration of the program is mostly related to the size of the design and the number of high-degree trends. If the trend has no solution equal to 0, then the algorithm will explore almost every possible choice point in the ILP model. This can be very large. The number of possible choice points can be more than a billion for designs larger than $2^3 \times 2^3$.

8 Conclusion

Two-dimensional trends have rarely been modeled in experimental applications. It has been established that the trends that might be present on row-by-column designs such as split plot designs, have a row-by-column interaction effect on the experiment (Edmondson (1993), Carrano *et. al.* (2007)). Systematically designing an experiment needs to take into account every dimension in which a trend can be present.

Traditionally, when performing experiments on row-by-column designs where a possible trend effect could be present, the design is built to be robust to the trend in one dimension. Any other trends present in other dimensions of the design are ignored. Cox (1979) brought to the attention these row-by-column interaction effects on Latin-squares. Since, very little has been done on this subject.

Carrano *et. al.* (2007) proposed that a more systematic and general design needed to be developed for building robust split-plot designs. One area of research was utilizing the already known fold-over method and applying it on split-plot experiments. This method has been used on traditional one-dimensional designs to effectively reduce the effects of trends. It can be implemented on row-by-column designs to achieve the same results.

One concern, however, on any design possibly affected by a trend, is the robustness to several higher degree trends and trend free factors and factor interactions within the same design. Fold-over methods, although fast, might not yield optimum designs. Fold-over methods will build robustness on low-degree trends and may not

target trends that are actually affecting the design. It is possible that the experimenter wishes to build robustness, to a very high degree trend and is willing to ignore the low-degree trends. For this situation, the fold-over method might not yield the most useful design.

For this reason, there are other systematic methods that are used to design experiments robust to specific trends. However, there are no methods used on split-plot designs. Carrano *et. al.* used a non-linear integer program to find a feasible order that was robust to a set of specific trends present in their split-plot experiment. However, this approach was not generalized for the class of split-plot designs and was not concerned with the development of solutions that are optimal with respect to some metric.

The integer linear programming used in this thesis was derived from the non-linear objective function of: $TI^{\alpha} = \left| \sum_{i=1}^{2^w} \sum_{j=1}^{2^s} D_{i,j}^{\alpha} \times \tau_{i,j} \right|$. Its purpose is to find an optimal design that had the best possible TI on all factors and included trends.

Table 8.1: Comparison of the Fold-over method and the ILP Method by using the Trend Index

Design		Fold-over method	ILP Method (not-weighted)	ILP Method (Weighted)
$2^2 \times 2^2$	Factor 1:	1156	724	1002
	Factor 2:	1904	652	700
	Total:	3060	1376	1702
$2^3 \times 2^2$	Factor 1:	1632	264	1632
	Factor 2:	2688	494	1128
	Total:	4320	758	2760
$2^2 \times 2^3$	Factor 1:	816	404	1296
	Factor 2:	3196	354	1494
	Factor 3:	7888	660	1532
	Total:	11900	1418	4322

Table 8.1 shows the Trend Index obtained by using the different methods used in this thesis. After comparing the results, the integer linear programming method achieved better robustness than the fold-over methods. The best TI was obtained by not weighting

the trends in the LP method. The un-weighted trends however yielded very few trends to which the factors were completely robust. On the other hand, using weighted trends on the LP method yielded far more completely robust trends in the factors. The total TI is understandably higher on the weighted trends. The weighted trends have an extra constraint and consequently, the un-weighted model gives the lower bounds of the design.

Table 8.2: Comparison of the Fold-over method and the LP Method by using the number of resisted trends

Design		Fold-over method	ILP Method (not-weighted)	ILP Method (Weighted)
$2^2 \times 2^2$	Factor 1:	5	none	none
	Factor 2:	3	none	2
	Total:	8	none	2
$2^3 \times 2^2$	Factor 1:	7	1	7
	Factor 2:	6	none	6
	Total:	13	1	13
$2^2 \times 2^3$	Factor 1:	7	4	5
	Factor 2:	3	none	none
	Factor 3:	3	1	2
	Total:	13	5	7

Table 8.2 compares the number of resisted trends obtained from the methods in this thesis. The fold-over method achieves more resisted trends than the ILP method. The weighted ILP method obtained more trends resisted than the un-weighted ILP method. The weighted ILP method targets important trends first and chooses the best design that is orthogonal to those trends.

The best method depends on the final objective of the experimenter. If the experimenter assumes that there are very few trends that need to be designed against, then a weighted ILP model can be used. With the weighted ILP, other less important trends can be added to the model so that they can have a low TI in the design. If the experimenter is worried about low-degree trends affecting the experiment, then the best

method is to fold the design as shown in the previous chapter. This fold-over method produces a design that is orthogonal to low-degree polynomial trends. The larger the design is, the more robust higher degree trends it will have. If the trends are not believed to be polynomial in nature, then the ILP model is a good way to make the design resistant to non-polynomial trends.

8.1 Future Research

The fold-over method is limited to only a fraction of the entire population of possible designs. Furthermore, the method used in this thesis is based on a greedy method of selection that does not guarantee optimality. An improved method of search might be required to guarantee a better design. On the other hand, the ILP does reach an optimal. The disadvantage of the ILP is that the problem is NP hard, and for large designs, the ILP will take longer to reach a solution. The ILP is based on the metric used in the objective function. This thesis looked at the TI of the design in the objective function. Other objective functions with different metrics might achieve a more desired design for the situation.

This thesis focused on making robust 2-level split-plot design. Fractional designs where not taken into account and this work can be advanced into other row × column designs. Split-split-plot designs might require three-dimensional trend resistance. Other designs that might require more work in this area are mixed-level split-plot designs.

The Fold-over method and the ILP method presented in this work generated robust designs; however, the run orders of these designs are completely systematically chosen, and lack the benefits of randomization. The lack of the randomization is outweighed by the need to be robust to trends. Also the D-efficiency of the design, as

discussed by Goos and Vandebroek (2001), might be jeopardized due to the systematic selection of the order without the consideration of the optimality plan.

9 References

Arvidsson, M. and Gremyr, I (2003), "Deliberate choices of restrictions in complete randomization", *Qual. Reliab. Engng. Int.*, Vol. 19, 87-99.

Atkinson, A. C. and Doney, A. N. (1996), "Experimental designs optimally balanced for trend", *Technometrics*, Vol. 38, No. 4, 333-341.

Bailey, R.A., Cheng, C. S. and Kipnis, P. (1992), "Construction of trend-resistant factorial designs", *Statistica Sinica*, Vol. 2, 393-411.

Bingham, D. R., Schoen, E. D. and Sitter, R. R. (2004), "Designing fractional factorial split-plot experiments with few whole-plot factors", *Appl. Statist.*, Vol. 53, Part 2, 325-339.

Bingham, D. and Sitter, R. R. (2003), "Fractional factorial split-plot designs for robust parameter experiments", *Technometrics*, Vol. 45, No. 1, 80-89.

Box, G. and Jones, S. (1992), "Split-plot designs for robust product experimentation", *Journal of applied Statistics*, Vol. 19, No. 1, 3-26.

Box, G. E. P. and Wilson, K. P. (1951), "On the experimental attainment of optimum conditions", *Journal of Royal Statistical Society, Ser. B*, Vol. 13, 1-45.

Bradley, R. A. and Yeh, C. M. (1980), "Trend-free block designs: Theory", *The Annals of Statistics*, Vol. 8, No. 4, 883-893.

Carrano, A. L., Taylor, J. B. and Lemaster, R. (2002), "Parametric characterization of peripheral sanding", *Forest Products Journal*, Vol. 52, No. 9, 1-7.

Carrano, A.L. and Thorn, B.K. (2004), "The construction of trend-free experimental plans on two-level split-plot designs". Industrial Engineering Research Conference, IERC 2004. May 15-19, 2004. Houston, Texas

Carrano, A.L., Thorn, B.K. and Lopez, G. (2007), B.K. "The construction of trend-free experimental plans on split-plot designs", (in press, *Journal of Manufacturing Systems*).

Chai, F. S. and Stufken, J. (1999), "Trend-free block designs for higher order trends", *Utilitas Mathematica*, Vol. 56, 65-78.

Chen, M. H. and Wang, P. C. (2001), “Multi-level factorial designs with minimum number of level changes”, *Communications in Statistics, Theory and Methods*, Vol. 30, No. 5, 875-885.

Cheng, C. S. (1985), “Run orders of factorial designs”, *Proceedings of the Berkeley Conference in Honor of Jerzy Neyman and Jack Kiefer* (Vol. 2), eds. Lucien M LeCam and Richard A. Olshen, Belmont, California: Wadsworth, 619-633.

Cheng, C. S. and Jacroux, M. (1988), “The construction of trend-free run orders of two-level factorial designs”, *Journal of American Statistical Association*, Vol. 83, No.404, 1152-1158.

Cheng, C. S. and Steinberg (1991), “Trend robust two-level factorial designs”, *Biometrika*, Vol. 78, No. 2, 325-336.

Chitturi, P. and John, P. W. M. (2002), “Nesting optimal main effects plans and optimal foldover designs”, *Communications in Statistics, Theory and Methods*, Vol. 31, No. 10, 1777-1794.

Coleman, D. E. and Montgomery, D. C. (1993), “A systematic approach to planning for a designed industrial experiment”, *Technometrics*, Vol. 35, No. 1, 1-12.

Coster, D. C. (1993), “Trend-free run orders of mixed-level fractional factorial designs”, *The Annals of Statistics*, Vol. 21, No. 4, 2072-2086.

Coster, D. C. (1993), “Tables of minimum cost, linear trend-free run sequences for two- and three-level fractional factorial designs”, *Computational Statistics & Data Analysis*, Vol. 16, 325-336.

Coster, D. C. and Cheng, C. S. (1988), “Minimum cost trend-free run orders of fractional factorial designs”, *The Annals of Statistics*, Vol. 16, No. 3, 1188-1205.

Cox, D. R. (1951), “Some systematic experimental designs”, *Biometrika*, Vol. 38, No. ¾, 312-323.

Cox, D. R. (1952), “Some recent work on systematic experimental designs”, *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 14, No. 2, 211-219.

Cox, D. R. (1979), “A remark on systematic Latin Squares”, *Journal of Royal Statistical Society. Series B (Methodological)*, Vol. 41, No. 3, 388-389.

Cui, X. and John, P. W. M. (1998), “Time-trend free run orders with the minimum level changes”, *Communications in Statistics, Theory and Methods*, Vol. 27, No. 1, 55-68.

Daniel, C. and Wilcoxon, F. (1966), “Factorial 2^{p-q} plans robust against linear and quadratic trends”, *Technometrics*, Vol. 8, No. 2, 259-278.

Dickinson, A. W. (1973), "Some run orders requiring a minimum number of factor level changes for the 2^4 and 2^5 main effect plans", *Technometrics*, Vol. 16, No. 1, 31-37.

Draper, N. R. and Stoneman, D. M. (1968), "Factor changes and linear trends in eight-run two-level factorial designs" *Technometrics*, Vol. 10, No. 2, 301-311

Edmonson, R. N. (1993), "Systematic row-and-column designs balanced for low order polynomial interactions between rows and columns", *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 55, No. 3, 707-723.

Gilat, A., *MATLAB: An introduction with Applications*, John Wiley & Sons, INC, Hoboken, NJ, 2005.

Goos, P. and Vandebroek, M. (2001), "Optimal split-plot designs", *Journal of Quality Technology*, Vol. 33, No. 4, 436-450.

Goos, P. and Vandebroek, M. (2004), "Outperforming completely randomized designs", *Journal of Quality Technology*, Vol. 36, No. 1, 12-26.

Hentenryck, P. V., *The OPL: Optimization Programming Language*, MIT Press, Cambridge, MA, 1999.

Hill, H. M. (1960), "Experimental designs to adjust for time trends", *Technometrics*, Vol. 2, No. 1, 67-82.

Jacroux, M. (1994), "On the construction of trend resistant mixed level factorial run orders", *The Annals of Statistics*, Vol. 22, No. 2, 904-916.

Jacroux, M. (1996), "On the construction of trend resistant asymmetrical orthogonal arrays", *Statistica Sinica*, Vol. 6, 289-297.

Jacroux, M. and Ray, R. S. (1990), "On the construction of trend-free run orders of treatments." *Biometrika*, Vol. 77, No. 1, 187-191.

John, P. W. M. (1990), "Time trends and factorial experiments", *Technometrics*, Vol. 32, No. 3, 275-282.

John, P. W. M. (1995), "Optimal foldover designs", *Commun. Statist. – Theory Meth*., Vol. 24, No. 7, 1821-1827.

Kowalski, S. M. (2002), "24 Run split-plot experiments for robust parameter design", *Journal of Quality Technology*, Vol. 34, No. 4, 399-410.

Lin, M. and Dean, A. M. (1991), "Trend-free block designs for varietal and factorial experiments", *The Annals of Statistics*, Vol. 19, No. 3, 1582-1596.

Lin, W. C. and Stufken, J. (1999), “On finding trend-free block designs”, *Journal of Statistical Planning and Inference*, Vol. 78, 57-70.

Lin, W. C. and Stufken, J. (2002), “Strongly linear trend-free block designs and 1-factor of representative graphs”, *Journal of Statistical Planning and Inference*, Vol. 106, 375-385.

Loeppky, L. L. and Sitter, R. R. (2002), “Analyzing unreplicated blocked or split-plot fractional factorial designs”, *Journal of Quality Technology*, Vol. 34, No. 3, 229-243.

Lucas, J. M. and Ju, H. L. (1992), “Split plotting and randomization in Industrial experiments”, *ASQC Quality congress transactions – Nashville*, 374-382.

Montgomery, D. C., *Design and Analysis of Experiments*, John Wiley & Sons, Inc., New York, 2001.

McLeod, R. G and Brewster, J. F. (2004), “The design of blocked fractional factorial split-plot experiments”, *Technometrics*, Vol. 46, No. 2, 135-146.

Robinson, T. J., Borror, C. M. and Myers, R. H. (2004), “Robust parameter design: A review”, *Qual. Reliab. Engng. Int.*, Vol. 20, 81-101.

Steinberg, D. M. (1988), “Factorials experiments with time trends”, *Technometrics*, Vol. 30, No. 3, 259-269.

Steinberg, D. M. and Hunter, W. G. (1984), “Experimental design: Review and comment”, *Technometrics*, Vol. 26, No. 2, 71-97.

Stufken, J. (1988), “On the existence of linear trend-free block designs”, *Commun. Statist. – Theory Meth.*, Vol. 17, No. 11, 3857-3863.

Tack, L. and Vandebroek, M. (2002), “Trend-Resistant and cost-efficient block designs with fixed or random block effects”, *Journal of Quality Technology*, Vol. 34, No. 4, 422-436.

Wang, P. C. (1991), “Symbol changes and trend resistance in orthogonal plans of symmetric factorials”, *Sankhya: The Indian Journal of Statistics*, Vol. 53, No. 3, 297-303.

Wang, P. C. and Chen, M. H. (1998), “Level changes and trend resistance on replacement in asymmetric orthogonal arrays”, *Journal of Statistical Planning and Inference*, Vol. 69, 349-358.

Wang, P.C. and Jan, H. W. (1995), "Designing two-level factorial experiments using orthogonal arrays when the run order is important", *The Statistician*, Vol. 44, No. 3, 379-388.

Yeh, C. M., Bradley, R. A. and Notz, W. I. (1985), "Nearly trend-free block designs", *Journal of the American Statistics Association*, Vol. 80, No. 392, 985-992.

Yeh, C. M. and Bradley, R. A. (1983), "Trend-free block designs: Existence and construction results", Communications *in Statistics, Theory and Methods*, Vol. 12, No. 1, 1-24.

10 Appendices

Appendix A: TI results for folded generators for $2^w \times 2^2$ split-plot designs

	Design for: 2^2 X 2^2										
	Generator row	**L X L**	**L X Q**	**L X C**	**Q X L**	**Q X Q**	**Q X C**	**C X L**	**C X Q**	**C X C**	**Total**
1	+1 +1 -1 -1	0	0	0	16	80	328	120	600	2460	3604
2	+1 -1 +1 -1	0	0	0	8	40	176	60	300	1320	1904
3	-1 +1 +1 -1	0	0	0	0	16	120	0	120	900	1156
4	+1 -1 -1 +1	0	0	0	0	16	120	0	120	900	1156
5	-1 +1 -1 +1	0	0	0	8	40	176	60	300	1320	1904
6	-1 -1 +1 +1	0	0	0	16	80	328	120	600	2460	3604

	Design for: 2^3 X 2^2										
	Generator row	**L X L**	**L X Q**	**L X C**	**Q X L**	**Q X Q**	**Q X C**	**C X L**	**C X Q**	**C X C**	**Total**
7	+1 +1 -1 -1	0	0	0	0	0	0	192	960	3936	5088
8	+1 -1 +1 -1	0	0	0	0	0	0	96	480	2112	2688
9	-1 +1 +1 -1	0	0	0	0	0	0	0	192	1440	1632
10	+1 -1 -1 +1	0	0	0	0	0	0	0	192	1440	1632
11	-1 +1 -1 +1	0	0	0	0	0	0	96	480	2112	2688
12	-1 -1 +1 +1	0	0	0	0	0	0	192	960	3936	5088

	Design for: 2^4 X 2^2										
	Generator row	**L X L**	**L X Q**	**L X C**	**Q X L**	**Q X Q**	**Q X C**	**C X L**	**C X Q**	**C X C**	**Total**
13	+1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
14	+1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
15	-1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
16	+1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
17	-1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
18	-1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0

	Design for: 2^5 X 2^2										
	Generator row	**L X L**	**L X Q**	**L X C**	**Q X L**	**Q X Q**	**Q X C**	**C X L**	**C X Q**	**C X C**	**Total**
19	+1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
20	+1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
21	-1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
22	+1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
23	-1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
24	-1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0

Appendix B: TI results for folded generators for a $2^2 \times 2^3$ split-plot design

	Generator row	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	Total
1	+1 +1 +1 +1 -1 -1 -1 -1	0	0	0	64	576	4384	480	4320	32880	42704
2	+1 +1 +1 -1 +1 -1 -1 -1	0	0	0	56	504	3896	420	3780	29220	37876
3	+1 +1 -1 +1 +1 -1 -1 -1	0	0	0	48	448	3600	360	3360	27000	34816
4	+1 -1 +1 +1 +1 -1 -1 -1	0	0	0	40	408	3448	300	3060	25860	33116
5	-1 +1 +1 +1 +1 -1 -1 -1	0	0	0	32	384	3392	240	2880	25440	32368
6	+1 +1 +1 -1 -1 +1 -1 -1	0	0	0	48	416	3168	360	3120	23760	30872
7	+1 +1 -1 +1 -1 +1 -1 -1	0	0	0	40	360	2872	300	2700	21540	27812
8	+1 -1 +1 +1 -1 +1 -1 -1	0	0	0	32	320	2720	240	2400	20400	26112
9	-1 +1 +1 +1 -1 +1 -1 -1	0	0	0	24	296	2664	180	2220	19980	25364
10	+1 +1 -1 -1 +1 +1 -1 -1	0	0	0	32	288	2384	240	2160	17880	22984
11	+1 -1 +1 -1 +1 +1 -1 -1	0	0	0	24	248	2232	180	1860	16740	21284
12	-1 +1 +1 -1 +1 +1 -1 -1	0	0	0	16	224	2176	120	1680	16320	20536
13	+1 -1 -1 +1 +1 +1 -1 -1	0	0	0	16	192	1936	120	1440	14520	18224
14	-1 +1 -1 +1 +1 +1 -1 -1	0	0	0	8	168	1880	60	1260	14100	17476
15	-1 -1 +1 +1 +1 +1 -1 -1	0	0	0	0	128	1728	0	960	12960	15776
16	+1 +1 +1 -1 -1 -1 +1 -1	0	0	0	40	312	2152	300	2340	16140	21284
17	+1 +1 -1 +1 -1 -1 +1 -1	0	0	0	32	256	1856	240	1920	13920	18224
18	+1 -1 +1 +1 -1 -1 +1 -1	0	0	0	24	216	1704	180	1620	12780	16524
19	-1 +1 +1 +1 -1 -1 +1 -1	0	0	0	16	192	1648	120	1440	12360	15776
20	+1 +1 -1 -1 +1 -1 +1 -1	0	0	0	24	184	1368	180	1380	10260	13396
21	+1 -1 +1 -1 +1 -1 +1 -1	0	0	0	16	144	1216	120	1080	9120	11696
22	-1 +1 +1 -1 +1 -1 +1 -1	0	0	0	8	120	1160	60	900	8700	10948
23	+1 -1 -1 +1 +1 -1 +1 -1	0	0	0	8	88	920	60	660	6900	8636
24	-1 +1 -1 +1 +1 -1 +1 -1	0	0	0	0	64	864	0	480	6480	7888
25	-1 -1 +1 +1 +1 -1 +1 -1	0	0	0	8	24	712	60	180	5340	6324
26	+1 +1 -1 -1 -1 +1 +1 -1	0	0	0	16	96	640	120	720	4800	6392
27	+1 -1 +1 -1 -1 +1 +1 -1	0	0	0	8	56	488	60	420	3660	4692
28	-1 +1 +1 -1 -1 +1 +1 -1	0	0	0	0	32	432	0	240	3240	3944
29	+1 -1 -1 +1 -1 +1 +1 -1	0	0	0	0	0	192	0	0	1440	1632
30	-1 +1 -1 +1 -1 +1 +1 -1	0	0	0	8	24	136	60	180	1020	1428
31	-1 -1 +1 +1 -1 +1 +1 -1	0	0	0	16	64	16	120	480	120	816
32	+1 -1 -1 -1 +1 +1 +1 -1	0	0	0	8	72	296	60	540	2220	3196
33	-1 +1 -1 -1 +1 +1 +1 -1	0	0	0	16	96	352	120	720	2640	3944
34	-1 -1 +1 -1 +1 +1 +1 -1	0	0	0	24	136	504	180	1020	3780	5644
35	-1 -1 -1 +1 +1 +1 +1 -1	0	0	0	32	192	800	240	1440	6000	8704
36	+1 +1 +1 -1 -1 -1 -1 +1	0	0	0	32	192	800	240	1440	6000	8704
37	+1 +1 -1 +1 -1 -1 -1 +1	0	0	0	24	136	504	180	1020	3780	5644
38	+1 -1 +1 +1 -1 -1 -1 +1	0	0	0	16	96	352	120	720	2640	3944
39	-1 +1 +1 +1 -1 -1 -1 +1	0	0	0	8	72	296	60	540	2220	3196
40	+1 +1 -1 -1 +1 -1 -1 +1	0	0	0	16	64	16	120	480	120	816
41	+1 -1 +1 -1 +1 -1 -1 +1	0	0	0	8	24	136	60	180	1020	1428
42	-1 +1 +1 -1 +1 -1 -1 +1	0	0	0	0	0	192	0	0	1440	1632
43	+1 -1 -1 +1 +1 -1 -1 +1	0	0	0	0	32	432	0	240	3240	3944
44	-1 +1 -1 +1 +1 -1 -1 +1	0	0	0	8	56	488	60	420	3660	4692
45	-1 -1 +1 +1 +1 -1 -1 +1	0	0	0	16	96	640	120	720	4800	6392
46	+1 +1 -1 -1 -1 +1 -1 +1	0	0	0	8	24	712	60	180	5340	6324
47	+1 -1 +1 -1 -1 +1 -1 +1	0	0	0	0	64	864	0	480	6480	7888
48	-1 +1 +1 -1 -1 +1 -1 +1	0	0	0	8	88	920	60	660	6900	8636
49	+1 -1 -1 +1 -1 +1 -1 +1	0	0	0	8	120	1160	60	900	8700	10948
50	-1 +1 -1 +1 -1 +1 -1 +1	0	0	0	16	144	1216	120	1080	9120	11696
51	-1 -1 +1 +1 -1 +1 -1 +1	0	0	0	24	184	1368	180	1380	10260	13396
52	+1 -1 -1 -1 +1 +1 -1 +1	0	0	0	16	192	1648	120	1440	12360	15776
53	-1 +1 -1 -1 +1 +1 -1 +1	0	0	0	24	216	1704	180	1620	12780	16524
54	-1 -1 +1 -1 +1 +1 -1 +1	0	0	0	32	256	1856	240	1920	13920	18224
55	-1 -1 -1 +1 +1 +1 -1 +1	0	0	0	40	312	2152	300	2340	16140	21284
56	+1 +1 -1 -1 -1 -1 +1 +1	0	0	0	0	128	1728	0	960	12960	15776
57	+1 -1 +1 -1 -1 -1 +1 +1	0	0	0	8	168	1880	60	1260	14100	17476
58	-1 +1 +1 -1 -1 -1 +1 +1	0	0	0	16	192	1936	120	1440	14520	18224
59	+1 -1 -1 +1 -1 -1 +1 +1	0	0	0	16	224	2176	120	1680	16320	20536
60	-1 +1 -1 +1 -1 -1 +1 +1	0	0	0	24	248	2232	180	1860	16740	21284
61	-1 -1 +1 +1 -1 -1 +1 +1	0	0	0	32	288	2384	240	2160	17880	22984
62	+1 -1 -1 -1 +1 -1 +1 +1	0	0	0	24	296	2664	180	2220	19980	25364
63	-1 +1 -1 -1 +1 -1 +1 +1	0	0	0	32	320	2720	240	2400	20400	26112
64	-1 -1 +1 -1 +1 -1 +1 +1	0	0	0	40	360	2872	300	2700	21540	27812
65	-1 -1 -1 +1 +1 -1 +1 +1	0	0	0	48	416	3168	360	3120	23760	30872
66	+1 -1 -1 -1 -1 +1 +1 +1	0	0	0	32	384	3392	240	2880	25440	32368
67	-1 +1 -1 -1 -1 +1 +1 +1	0	0	0	40	408	3448	300	3060	25860	33116
68	-1 -1 +1 -1 -1 +1 +1 +1	0	0	0	48	448	3600	360	3360	27000	34816
69	-1 -1 -1 +1 -1 +1 +1 +1	0	0	0	56	504	3896	420	3780	29220	37876
70	-1 -1 -1 -1 +1 +1 +1 +1	0	0	0	64	576	4384	480	4320	32880	42704

Appendix C: TI results for folded generators for a $2^3 \times 2^3$ split-plot design

	Generator row	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	Total
1	+1 +1 +1 +1 -1 -1 -1 -1	0	0	0	0	0	0	768	6912	52608	60288
2	+1 +1 +1 -1 +1 -1 -1 -1	0	0	0	0	0	0	672	6048	46752	53472
3	+1 +1 -1 +1 +1 -1 -1 -1	0	0	0	0	0	0	576	5376	43200	49152
4	+1 -1 +1 +1 +1 -1 -1 -1	0	0	0	0	0	0	480	4896	41376	46752
5	-1 +1 +1 +1 +1 -1 -1 -1	0	0	0	0	0	0	384	4608	40704	45696
6	+1 +1 +1 -1 -1 +1 -1 -1	0	0	0	0	0	0	576	4992	38016	43584
7	+1 +1 -1 +1 -1 +1 -1 -1	0	0	0	0	0	0	480	4320	34464	39264
8	+1 -1 +1 +1 -1 +1 -1 -1	0	0	0	0	0	0	384	3840	32640	36864
9	-1 +1 +1 +1 -1 +1 -1 -1	0	0	0	0	0	0	288	3552	31968	35808
10	+1 +1 -1 -1 +1 +1 -1 -1	0	0	0	0	0	0	384	3456	28608	32448
11	+1 -1 +1 -1 +1 +1 -1 -1	0	0	0	0	0	0	288	2976	26784	30048
12	-1 +1 +1 -1 +1 +1 -1 -1	0	0	0	0	0	0	192	2688	26112	28992
13	+1 -1 -1 +1 +1 +1 -1 -1	0	0	0	0	0	0	192	2304	23232	25728
14	-1 +1 -1 +1 +1 +1 -1 -1	0	0	0	0	0	0	96	2016	22560	24672
15	-1 -1 +1 +1 +1 +1 -1 -1	0	0	0	0	0	0	0	1536	20736	22272
16	+1 +1 +1 -1 -1 -1 +1 -1	0	0	0	0	0	0	480	3744	25824	30048
17	+1 +1 -1 +1 -1 -1 +1 -1	0	0	0	0	0	0	384	3072	22272	25728
18	+1 -1 +1 +1 -1 -1 +1 -1	0	0	0	0	0	0	288	2592	20448	23328
19	-1 +1 +1 +1 -1 -1 +1 -1	0	0	0	0	0	0	192	2304	19776	22272
20	+1 +1 -1 -1 +1 -1 +1 -1	0	0	0	0	0	0	288	2208	16416	18912
21	+1 -1 +1 -1 +1 -1 +1 -1	0	0	0	0	0	0	192	1728	14592	16512
22	-1 +1 +1 -1 +1 -1 +1 -1	0	0	0	0	0	0	96	1440	13920	15456
23	+1 -1 -1 +1 +1 -1 +1 -1	0	0	0	0	0	0	96	1056	11040	12192
24	-1 +1 -1 +1 +1 -1 +1 -1	0	0	0	0	0	0	0	768	10368	11136
25	-1 -1 +1 +1 +1 -1 +1 -1	0	0	0	0	0	0	96	288	8544	8928
26	+1 +1 -1 -1 -1 +1 +1 -1	0	0	0	0	0	0	192	1152	7680	9024
27	+1 -1 +1 -1 -1 +1 +1 -1	0	0	0	0	0	0	96	672	5856	6624
28	-1 +1 +1 -1 -1 +1 +1 -1	0	0	0	0	0	0	0	384	5184	5568
29	+1 -1 -1 +1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	2304	2304
30	-1 +1 -1 +1 -1 +1 +1 -1	0	0	0	0	0	0	96	288	1632	2016
31	-1 -1 +1 +1 -1 +1 +1 -1	0	0	0	0	0	0	192	768	192	1152
32	+1 -1 -1 -1 +1 +1 +1 -1	0	0	0	0	0	0	96	864	3552	4512
33	-1 +1 -1 -1 +1 +1 +1 -1	0	0	0	0	0	0	192	1152	4224	5568
34	-1 -1 +1 -1 +1 +1 +1 -1	0	0	0	0	0	0	288	1632	6048	7968
35	-1 -1 -1 +1 +1 +1 +1 -1	0	0	0	0	0	0	384	2304	9600	12288
36	+1 +1 +1 -1 -1 -1 -1 +1	0	0	0	0	0	0	384	2304	9600	12288
37	+1 +1 -1 +1 -1 -1 -1 +1	0	0	0	0	0	0	288	1632	6048	7968
38	+1 -1 +1 +1 -1 -1 -1 +1	0	0	0	0	0	0	192	1152	4224	5568
39	-1 +1 +1 +1 -1 -1 -1 +1	0	0	0	0	0	0	96	864	3552	4512
40	+1 +1 -1 -1 +1 -1 -1 +1	0	0	0	0	0	0	192	768	192	1152
41	+1 -1 +1 -1 +1 -1 -1 +1	0	0	0	0	0	0	96	288	1632	2016
42	-1 +1 +1 -1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	2304	2304
43	+1 -1 -1 +1 +1 -1 -1 +1	0	0	0	0	0	0	0	384	5184	5568
44	-1 +1 -1 +1 +1 -1 -1 +1	0	0	0	0	0	0	96	672	5856	6624
45	-1 -1 +1 +1 +1 -1 -1 +1	0	0	0	0	0	0	192	1152	7680	9024
46	+1 +1 -1 -1 -1 +1 -1 +1	0	0	0	0	0	0	96	288	8544	8928
47	+1 -1 +1 -1 -1 +1 -1 +1	0	0	0	0	0	0	0	768	10368	11136
48	-1 +1 +1 -1 -1 +1 -1 +1	0	0	0	0	0	0	96	1056	11040	12192
49	+1 -1 -1 +1 -1 +1 -1 +1	0	0	0	0	0	0	96	1440	13920	15456
50	-1 +1 -1 +1 -1 +1 -1 +1	0	0	0	0	0	0	192	1728	14592	16512
51	-1 -1 +1 +1 -1 +1 -1 +1	0	0	0	0	0	0	288	2208	16416	18912
52	+1 -1 -1 -1 +1 +1 -1 +1	0	0	0	0	0	0	192	2304	19776	22272
53	-1 +1 -1 -1 +1 +1 -1 +1	0	0	0	0	0	0	288	2592	20448	23328
54	-1 -1 +1 -1 +1 +1 -1 +1	0	0	0	0	0	0	384	3072	22272	25728
55	-1 -1 -1 +1 +1 +1 -1 +1	0	0	0	0	0	0	480	3744	25824	30048
56	+1 +1 -1 -1 -1 -1 +1 +1	0	0	0	0	0	0	0	1536	20736	22272
57	+1 -1 +1 -1 -1 -1 +1 +1	0	0	0	0	0	0	96	2016	22560	24672
58	-1 +1 +1 -1 -1 -1 +1 +1	0	0	0	0	0	0	192	2304	23232	25728
59	+1 -1 -1 +1 -1 -1 +1 +1	0	0	0	0	0	0	192	2688	26112	28992
60	-1 +1 -1 +1 -1 -1 +1 +1	0	0	0	0	0	0	288	2976	26784	30048
61	-1 -1 +1 +1 -1 -1 +1 +1	0	0	0	0	0	0	384	3456	28608	32448
62	+1 -1 -1 -1 +1 -1 +1 +1	0	0	0	0	0	0	288	3552	31968	35808
63	-1 +1 -1 -1 +1 -1 +1 +1	0	0	0	0	0	0	384	3840	32640	36864
64	-1 -1 +1 -1 +1 -1 +1 +1	0	0	0	0	0	0	480	4320	34464	39264
65	-1 -1 -1 +1 +1 -1 +1 +1	0	0	0	0	0	0	576	4992	38016	43584
66	+1 -1 -1 -1 -1 +1 +1 +1	0	0	0	0	0	0	384	4608	40704	45696
67	-1 +1 -1 -1 -1 +1 +1 +1	0	0	0	0	0	0	480	4896	41376	46752
68	-1 -1 +1 -1 -1 +1 +1 +1	0	0	0	0	0	0	576	5376	43200	49152
69	-1 -1 -1 +1 -1 +1 +1 +1	0	0	0	0	0	0	672	6048	46752	53472
70	-1 -1 -1 -1 +1 +1 +1 +1	0	0	0	0	0	0	768	6912	52608	60288

Appendix D: TI results for folded generators for a $2^4 \times 2^3$ split-plot design

	Generator row	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	Total
1	+1 +1 +1 +1 -1 -1 -1 -1	0	0	0	0	0	0	0	0	0	0
2	+1 +1 +1 -1 +1 -1 -1 -1	0	0	0	0	0	0	0	0	0	0
3	+1 +1 -1 +1 +1 -1 -1 -1	0	0	0	0	0	0	0	0	0	0
4	+1 -1 +1 +1 +1 -1 -1 -1	0	0	0	0	0	0	0	0	0	0
5	-1 +1 +1 +1 +1 -1 -1 -1	0	0	0	0	0	0	0	0	0	0
6	+1 +1 +1 -1 -1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
7	+1 +1 -1 +1 -1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
8	+1 -1 +1 +1 -1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
9	-1 +1 +1 +1 -1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
10	+1 +1 -1 -1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
11	+1 -1 +1 -1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
12	-1 +1 +1 -1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
13	+1 -1 -1 +1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
14	-1 +1 -1 +1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
15	-1 -1 +1 +1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
16	+1 +1 +1 -1 -1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
17	+1 +1 -1 +1 -1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
18	+1 -1 +1 +1 -1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
19	-1 +1 +1 +1 -1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
20	+1 +1 -1 -1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
21	+1 -1 +1 -1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
22	-1 +1 +1 -1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
23	+1 -1 -1 +1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
24	-1 +1 -1 +1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
25	-1 -1 +1 +1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
26	+1 +1 -1 -1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
27	+1 -1 +1 -1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
28	-1 +1 +1 -1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
29	+1 -1 -1 +1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
30	-1 +1 -1 +1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
31	-1 -1 +1 +1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
32	+1 -1 -1 -1 +1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
33	-1 +1 -1 -1 +1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
34	-1 -1 +1 -1 +1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
35	-1 -1 -1 +1 +1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
36	+1 +1 +1 -1 -1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
37	+1 +1 -1 +1 -1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
38	+1 -1 +1 +1 -1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
39	-1 +1 +1 +1 -1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
40	+1 +1 -1 -1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
41	+1 -1 +1 -1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
42	-1 +1 +1 -1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
43	+1 -1 -1 +1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
44	-1 +1 -1 +1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
45	-1 -1 +1 +1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
46	+1 +1 -1 -1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
47	+1 -1 +1 -1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
48	-1 +1 +1 -1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
49	+1 -1 -1 +1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
50	-1 +1 -1 +1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
51	-1 -1 +1 +1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
52	+1 -1 -1 -1 +1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
53	-1 +1 -1 -1 +1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
54	-1 -1 +1 -1 +1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
55	-1 -1 -1 +1 +1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
56	+1 +1 -1 -1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
57	+1 -1 +1 -1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
58	-1 +1 +1 -1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
59	+1 -1 -1 +1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
60	-1 +1 -1 +1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
61	-1 -1 +1 +1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
62	+1 -1 -1 -1 +1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
63	-1 +1 -1 -1 +1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
64	-1 -1 +1 -1 +1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
65	-1 -1 -1 +1 +1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
66	+1 -1 -1 -1 -1 +1 +1 +1	0	0	0	0	0	0	0	0	0	0
67	-1 +1 -1 -1 -1 +1 +1 +1	0	0	0	0	0	0	0	0	0	0
68	-1 -1 +1 -1 -1 +1 +1 +1	0	0	0	0	0	0	0	0	0	0
69	-1 -1 -1 +1 -1 +1 +1 +1	0	0	0	0	0	0	0	0	0	0
70	-1 -1 -1 -1 +1 +1 +1 +1	0	0	0	0	0	0	0	0	0	0

Appendix E: TI results for folded generators for a $2^5 \times 2^3$ split-plot design

	Generator row	L X L	L X Q	L X C	Q X L	Q X Q	Q X C	C X L	C X Q	C X C	Total
1	+1 +1 +1 +1 -1 -1 -1 -1	0	0	0	0	0	0	0	0	0	0
2	+1 +1 +1 -1 +1 -1 -1 -1	0	0	0	0	0	0	0	0	0	0
3	+1 +1 -1 +1 +1 -1 -1 -1	0	0	0	0	0	0	0	0	0	0
4	+1 -1 +1 +1 +1 -1 -1 -1	0	0	0	0	0	0	0	0	0	0
5	-1 +1 +1 +1 +1 -1 -1 -1	0	0	0	0	0	0	0	0	0	0
6	+1 +1 +1 -1 -1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
7	+1 +1 -1 +1 -1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
8	+1 -1 +1 +1 -1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
9	-1 +1 +1 +1 -1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
10	+1 +1 -1 -1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
11	+1 -1 +1 -1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
12	-1 +1 +1 -1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
13	+1 -1 -1 +1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
14	-1 +1 -1 +1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
15	-1 -1 +1 +1 +1 +1 -1 -1	0	0	0	0	0	0	0	0	0	0
16	+1 +1 +1 -1 -1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
17	+1 +1 -1 +1 -1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
18	+1 -1 +1 +1 -1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
19	-1 +1 +1 +1 -1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
20	+1 +1 -1 -1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
21	+1 -1 +1 -1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
22	-1 +1 +1 -1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
23	+1 -1 -1 +1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
24	-1 +1 -1 +1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
25	-1 -1 +1 +1 +1 -1 +1 -1	0	0	0	0	0	0	0	0	0	0
26	+1 +1 -1 -1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
27	+1 -1 +1 -1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
28	-1 +1 +1 -1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
29	+1 -1 -1 +1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
30	-1 +1 -1 +1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
31	-1 -1 +1 +1 -1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
32	+1 -1 -1 -1 +1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
33	-1 +1 -1 -1 +1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
34	-1 -1 +1 -1 +1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
35	-1 -1 -1 +1 +1 +1 +1 -1	0	0	0	0	0	0	0	0	0	0
36	+1 +1 +1 -1 -1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
37	+1 +1 -1 +1 -1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
38	+1 -1 +1 +1 -1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
39	-1 +1 +1 +1 -1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
40	+1 +1 -1 -1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
41	+1 -1 +1 -1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
42	-1 +1 +1 -1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
43	+1 -1 -1 +1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
44	-1 +1 -1 +1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
45	-1 -1 +1 +1 +1 -1 -1 +1	0	0	0	0	0	0	0	0	0	0
46	+1 +1 -1 -1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
47	+1 -1 +1 -1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
48	-1 +1 +1 -1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
49	+1 -1 -1 +1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
50	-1 +1 -1 +1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
51	-1 -1 +1 +1 -1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
52	+1 -1 -1 -1 +1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
53	-1 +1 -1 -1 +1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
54	-1 -1 +1 -1 +1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
55	-1 -1 -1 +1 +1 +1 -1 +1	0	0	0	0	0	0	0	0	0	0
56	+1 +1 -1 -1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
57	+1 -1 +1 -1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
58	-1 +1 +1 -1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
59	+1 -1 -1 +1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
60	-1 +1 -1 +1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
61	-1 -1 +1 +1 -1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
62	+1 -1 -1 -1 +1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
63	-1 +1 -1 -1 +1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
64	-1 -1 +1 -1 +1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
65	-1 -1 -1 +1 +1 -1 +1 +1	0	0	0	0	0	0	0	0	0	0
66	+1 -1 -1 -1 -1 +1 +1 +1	0	0	0	0	0	0	0	0	0	0
67	-1 +1 -1 -1 -1 +1 +1 +1	0	0	0	0	0	0	0	0	0	0
68	-1 -1 +1 -1 -1 +1 +1 +1	0	0	0	0	0	0	0	0	0	0
69	-1 -1 -1 +1 -1 +1 +1 +1	0	0	0	0	0	0	0	0	0	0
70	-1 -1 -1 -1 +1 +1 +1 +1	0	0	0	0	0	0	0	0	0	0

Appendix F: Code in MATLAB used to generate the list of fold-over generators.

```
function DD = dgen(n)
%dgen(n) creates all the possible generators for n factors.
%This will generate a design for one factor.
%n = number of factors to be used in the design

tic
%initiate all variables
colmax = 2^n;
k = 1;
count = 0;
countmax = 2^colmax;

%initiate the contrast Array
for i = 1:(colmax+1)
    C(i) = -1;
end
D(1,:) = C(1:colmax);

%generate
for j = 1:2^colmax
    %disp(j)
    for i = 1:colmax
        if C(i) > 1
            C(i+1) = C(i+1) + 2;
            C(i) = -1;
        end
    end
    %sum(C(1:colmax))
    if sum(C(1:colmax))==0            %only select those that are
        D(k,:) = C(1:colmax);
        k = k + 1;
    end
    count = count+1;
    X = (count/countmax)*100;
    %clc
    fprintf('%8.5f percent completed.\n',X)
    C(1) = C(1) + 2;

end

DD = D;
toc
%t = toc;
fprintf('The run time in seconds was: %f seconds',toc);
```

Appendix G: Code in MATLAB used to calculate the TI of the folded generators.

```
function evaltc(A,n,TT)
%evaltc(A,n,TT) evaluates all the timecounts for all the
%   arrangements in A and with n whole plot factors.
%   Each row in A is a new arrangement.
%   TT is the set of Trend matrices
tic

sizeA = size(A);
sizeTT = size(TT);
DD = date;
s = log2(sizeA(2));     %number of sub-plot factors
count = 0;

fid = fopen('trendindex.txt','a');

fprintf(fid,'\n \n');
fprintf(fid,'#################################\n');
fprintf(fid,'            TRENDINDEX\n           ');
fprintf(fid,DD);
fprintf(fid,'\n#################################\n \n');

% fprintf(fid,'DEBUG\n');
% fprintf(fid,'%i ',A');
% eval = A;

% detail the trends
fprintf(fid,'=================================\n');
fprintf(fid,'Trends to be used:\n \n');

% write to file all the trends for reference
for t = 1:sizeTT(3)
     fprintf(fid,'Trend %i:\n',t);
     fprintf(fid,' %i',TT(:,:,t)');
     fprintf(fid,'\n');
end

fprintf(fid,'\n=================================\n');
fprintf(fid,'Setup conditions:\n \n');
fprintf(fid,'Model for %i subplots and %i
wholeplots\n',s,n);
```

```
disp('running TimeIndex')
countmax = sizeA(1);
fprintf(fid,'\n');
for t = 1:sizeTT(3)
    fprintf(fid,' trend %2i',t);
end
for i = 1:sizeA(1)
    design = foldover(A(i,:),n);
    fprintf(fid,'\nDesign %5i: ',i);
    fprintf(fid,'%+i  ',design(1,:));
    for t = 1:sizeTT(3)
        tc = abs(sum(sum(design.*TT(:,:,t))));
        fprintf(fid,' %8d',tc);
     end
    count = count+1;
    X = (count/countmax)*100;
    %clc
    fprintf('%5.2f percent completed.\n',X);
end


fclose(fid);
toc;
%t = toc;
fprintf('The run time in seconds was: %f seconds\n',toc);
```

Appendix H: Code in MATLAB used to create the fold-over design from one generator

```
function FO = foldover(A,n)
%foldover(A,n) generates a design for n factors in the
%whole plot for the initial A contrast vector
%If A is a matrix, foldover function will just use the
%first row as the vector.

row = 2^n;       %max number of rows for the design matrix

FO(1,:) = A(1,:);

k = 1;
i = 1;
while k < row
    %k
    while i <= 2*k-1
        %k
        %value = i - k
        FO(i+1,:) = FO(i-k+1,:)*-1;
        i = i+1;
    end
    k = k*2;
end
```

Appendix I: Model used in OPL to calculate the optimal TI using Integer Linear Programming.

```
int spfactormax = ...;
int wpfactormax = ...;

int column = pow(2,spfactormax);
int row = pow(2,wpfactormax);
int maxsptrend = ...;
int maxwptrend = ...;

range columns 1 .. column;
range rows 1 .. row;
range factor 1 .. spfactormax;
range spt 1..maxsptrend;
range wpt 1..maxwptrend;
range perm 1 .. column;

float weight[spt,wpt] = ...;
int sptrend[spt,columns] = ...;
int wptrend[wpt,rows] = ...;
int value[factor,perm] = ...;

var float+ Spos[spt,wpt,factor];
var float+ Sneg[spt,wpt,factor];
var int+ X[rows,columns,perm] in 0..1;

minimize
sum(f in factor,s in spt,w in wpt)
   (Spos[s,w,f] + Sneg[s,w,f])

subject to
{
  forall(f in factor,s in spt,w in wpt)
    (sum(j in rows,k in columns)
    (sptrend[s,k]*wptrend[w,j]*weight[s,w]*(sum(i in perm)
    (value[f,i]*X[j,k,i]))) - (sum(j in rows,k in columns)
    (sptrend[s,k]*wptrend[w,j]*weight[s,w]*(sum(i in perm)
    ((1-value[f,i])*X[j,k,i])))) =
    (Spos[s,w,f] - Sneg[s,w,f]));
```

Appendix I (continued)

```
//A permutation appears once in every row
  forall(i in perm,j in rows)
    sum(k in columns)
    X[j,k,i] = 1;

//A permutation appears once in every position
  forall(j in rows, k in columns)
    sum(i in perm)
    X[j,k,i] = 1;

};
```

Appendix J: Example of the data file used in OPL for the model.

```
spfactormax = 3;
wpfactormax = 2;
maxsptrend = 3;
maxwptrend = 3;


weight = [//wpt  L   Q   C     spt
               [0.0 0.0 0.0] // L
               [1.0 0.0 0.0] // Q
               [0.0 0.0 0.0] // C
         ];

sptrend = [
            [1 2 3 4 5 6 7 8]                //Linear
            [1 4 9 16 25 36 49 64]           //Quadratic
            [1 8 27 64 125 216 343 512]      //Cubic
          ];
wptrend = [
            [1 2 3 4]                        //Linear
            [1 4 9 16]                       //Quadratic
            [1 8 27 64]                      //Cubic
          ];
value = [
        [0 1 0 1 0 1 0 1]                    //Factor (A)
        [0 0 1 1 0 0 1 1]                    //Factor (B)
        [0 0 0 0 1 1 1 1]                    //Factor (C)
        ];
```