2017

TUNING OPTIMIZATION SOFTWARE PARAMETERS FOR MIXED INTEGER PROGRAMMING PROBLEMS

Toni P. Sorrell
Virginia Commonwealth University

TUNING OPTIMIZATION SOFTWARE PARAMETERS FOR MIXED INTEGER PROGRAMMING PROBLEMS

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Systems Modeling and Analysis, at Virginia Commonwealth University.

by

Toni P. Sorrell

Bachelor of Science in Secondary Education-Math Option,
Pennsylvania State University, 1989
Masters of Interdisciplinary Studies, Virginia Commonwealth University, 2002

Major Directors: J. Paul Brooks, David J. Edwards, Associate Professors
Statistical Science and Operations Research

Virginia Commonwealth University
Richmond, Virginia
July, 2017

Acknowledgments

I would like to thank my advisors, Dr. J. Paul Brooks and Dr. David Edwards for their time and support. I would also like to thank my committee members for all of their valuable comments. Throughout my studies I have had the support of my friends and colleagues in the Statistical Sciences and Operations Research and Mathematics departments. To my classmates and study buddies, thanks for being so patient when I asked questions, sometime several times. Especially Dr. Sudharshana Srinivasan Apte and Dr. Robert Leonard. You both are the best!!! Thank you, Dr. Aimee Ellington, for being so flexible when scheduling my work obligations around my student schedule. I was fortunate to be able to teach with some very wonderful teachers so thank you Dr. Joy Whitenack and Kristina Anthony, I learned so much from both of you. I have been especially fortunate to have a wonderful mentor, Dr. Reuben Farley, throughout my career. He has encouraged my love of learning and teaching. I would not have ever attempted to earn a Master's degree let alone a PhD, without all of your support, encouragement, and friendship. A special thanks to Dr. Jill Hardin Wilson who encouraged me to take my first class in optimization; your class made me work hard and fall in love with modeling and optimization. Thanks to Amy Kimbrough, my late-night pal:) Thank you John Nobel, Mike Davis, Johnny Layne and Carlisle Childress, for all of the support you gave me when using Bach. VCU is a diverse community. Due to that diversity, I have been able to make friends with people from so many different countries and they have enriched my life by sharing with me their culture, thoughts, and ideas.

I have been so blessed to have a wonderful and supportive family. Mom and Dad, I love you both for all that you have done for me throughout my life. I am so proud to be you daughter. To my brother Mike, thanks for all of your help with C and your advice. (When are you moving next door to me? :)

I have been working on my degree since my son Benjamin was five years old, VCU is his home away from home. Thanks for spending a lot of time there with me Ben. I love you and I am very proud of you! You are a smart and loving son who has so many talents, but perhaps the best is that you really are a joyful person. I have a very special husband who has always been willing to accommodate my need to study on the weekends or at night. Calvin, you gave me the gift of time, which has been priceless. Thanks for taking Ben on all of your band trips, parades, and football games. I know that was challenging especially when he was young. But look, you got a drummer out of that deal :) Love you the best!

TABLE OF CONTENTS

List of Tables

List of Figures

Abstract

TUNING OPTIMIZATION SOFTWARE PARAMETERS FOR MIXED INTEGER PROGRAMMING PROBLEMS

By Toni P. Sorrell

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Systems Modeling and Analysis, at Virginia Commonwealth University.

Virginia Commonwealth University, 2017.

Major Directors: J. Paul Brooks, David J. Edwards, Associate Professors
Statistical Science and Operations Research

The tuning of optimization software is of key interest to researchers solving mixed integer programming (MIP) problems. The efficiency of the optimization software can be greatly impacted by the solver's parameter settings and the structure of the MIP. A designed experiment approach is used to fit a statistical model that would suggest settings of the parameters that provided the largest reduction in the primal integral metric. Tuning exemplars of six and 59 factors (parameters) of optimization software, experimentation takes place on three classes of MIPs: survivable fixed telecommunication network design, a formulation of the support vector machine with the ramp loss and L1-norm regularization, and node packing for coding theory graphs. This research presents and demonstrates a framework for tuning a portfolio of MIP

instances to not only obtain good parameter settings used for future instances of the same class of MIPs, but to also gain insights into which parameters and interactions of parameters are significant for that class of MIPs. The framework is used for benchmarking of solvers with tuned parameters on a portfolio of instances. A group screening method provides a way to reduce the number of factors in a design and reduces the time it takes to perform the tuning process. Portfolio benchmarking provides performance information of optimization solvers on a class with instances of a similar structure.

Chapter 2 Introduction

1.1 Motivation

Great strides have been made in the performance of mixed integer programming (MIP) solvers over the past 30 years (Achterberg and Wunderling, 2013; Bixby and Rothberg, 2007). Commercial and open source optimization solvers easily solve many MIPs. However, more complex problems with millions of constraints and variables may be difficult and time consuming to solve for a feasible or optimal solution Bixby and Rothberg, (2007). Applications of MIPs occur in a plethora of industries and research fields such as production planning, supply chain management, management systems of electric power distribution networks, survivable fixed telecommunication network designs, node packing for coding theory problems, and support vector machines (Borghetti, 2013; Hess and Brooks, 2015; Orlowski et al., 2010; Pochet and Wolsey, 2006; Raack et al., 2011). Often practitioners need to solve multiple instances of MIPs repeatedly and over time. For example, electric power distribution networks need to conduct periodic optimization of operating conditions in order to minimize demand on the power network; therefore, solving these instances quickly keeps the network operational (Borghetti, 2013). Parameter settings, the structure of the MIP, and random effects (such as changing the order in which the constraints and variables are added or a change in a random seed for an algorithm) impact the efficiency of MIP solvers (Danna, 2008; Koch and Hendel, 2014; Lodi

and Tramontani, 2013). Determining good parameter settings for a specific solver and instance can reduce computational requirements needed to produce a feasible or optimal solution.

Another important consideration for the practitioner is to determine which optimization software (MIP solvers) provide the best performance for the types of problems they need solved. Hans Mittelmann provides a website with benchmarking information for optimization software (Mittelmann, 2016). MIPLIB2010 Koch et al., (2011), a test-bed library, consists of 361 instances, classifying 62% as easy, 16% as hard, and 22% as not solved (Mittelmann, 2016). Mittelmann's benchmarking work involving MIPs offers a comparison of leading commercial and open source optimization software, using 24% of the MIPLIB2010 test-bed containing only 'easy' instances (Mittelmann, 2016). Commercial optimization software outperforms the open source in terms of the number of instances solved and the time it takes to find an optimal solution. Mittelmann's results rely on using the optimizers tested at default settings. For easy problems, the average scaled time ranges between one and seven seconds for the top three optimizers under default settings. This information would not offer the necessary insights needed for users working with more difficult problems. Comparing solvers under alternative settings potentially increases efficiency because in many cases the optimizer may be more efficient when users tune parameter settings for a specific MIP. When researchers compare a new algorithm with existing algorithms, they create conditions that highlight the strengths of the existing algorithms to ensure that the results reveal a true comparison of best attributes. By doing this, researchers can easily verify if the new algorithm will be an improvement over the previous algorithms in those particular settings. Since solvers are instrumental in creating equitable conditions to compare algorithms, tuning the parameters of solvers for a specific class of problems could achieve better testing environment (Baz et al., 2011). Even though Gurobi

Optimization, a leading optimization software company, suggests that for most situations their default settings perform well, they also discuss the importance of tuning the parameter settings for MIPs (Gurobi Optimization, 2015). Recently, at the 2016 annual meeting of INFORMS, a developer of Gurobi Optimization explained that for certain default parameter settings, they used their best "guess." A "guess" from this expert in the field definitely offers great value to all users of their product, but is potentially missing the possible information that could be gained by conducting a designed experiment. Comparing the optimizers after they have been tuned would offer valuable information to researchers and practitioners alike.

1.2 Problem Description

Finding a feasible solution to a given MIP model can be an extremely hard problem to solve in practice. MIP solvers and their efficiency in solving MIPs are impacted by the solver's parameter settings and the structure of the MIP. The problem faced by users of optimization software is that there are a variety of parameters with different settings, making it computationally intractable to test all possible combinations. One way to approach this problem is to set up a designed experiment in order to reduce computation time. For example, a full factorial designed involving 10 two-level factors would have 1024 experiment runs to complete in order to test all possible combinations of the factors and levels. It is possible to reduce this number by selecting a fraction of the full factorial design. An eighth fractional factorial design with the 10 factors would give the user the ability to identify main effects and two-factor interactions with only 128 experimental runs which is a $\frac{7}{8}$ or 87.5% reduction. Thus, providing a design that uses less computational effort but still is able to identify significant variables that affect the response. Although this research focuses on MIPs, tuning parameters with the methodology used should

provide similar insights on any type of mathematical programming problem that the optimization software is designed to solve. The goal of this research is to use a designed experiment approach to fit a model that will not only obtain good parameter settings, but also provide insights on which parameters are critical for improving the performance of the optimization software. Further, the ability to compare solvers after they have been tuned for a specific class of MIPs could provide valuable information to practitioners on which software provides the greatest advantage for the problems they are working with.

1.3 Common Language

The following definitions provide an understanding of the language used in this paper.

- **Model** – refers to the first order with interactions regression model that is used to make recommendations for parameter settings.
- **MIP, instance** – refers to a mixed integer programming model that is part of a class of instances.
- **Design point, run, setting, treatment** –all refer to a unique set of parameter values to be tested
- **Parameter** – is a feature that can be set to different values to alter the functionality of the optimization software.
- **Factor, variable** – both refer to a single parameter that can be set on optimization software
- **Levels** – the possible number of values that can be assigned to a categorical or ordinal variable

1.4 Dissertation Outline

Chapter 2 provides literature review, background information, fundamental concepts, along with information about the test-beds performance metric. Chapter 3 contains methodology and results for the experiment with six parameters. Chapter 4 contains the methodology and results of the experiment with 59 factors. Three methods used for the additional step of screening out unimportant parameters is also discussed. Chapter 5 contains benchmark results for the three classes of instances for two commercial solvers.

Chapter 3 Literature Review and Background

2.1 Parameter Tuning

Parameter tuning takes place in a variety of areas. In the area of the automatic algorithm configuration problem there are five main areas: numerical optimization, heuristic search methods, model based optimization, experimental design and analysis of variance (ANOVA), and sequential statistical testing (Stützle and López-Ibáñez, 2013). Under numerical optimization techniques, mesh adaptive direct search (MADS) is used to tune parameters to local optima (Audet and Orban, 2006). Each run may take several hours and results are not predictable. This method was tested on a small number (four) of continuous parameters. Yuan et al., (2012) developed a continuous optimization method that was able to deal with the stochastic nature of the tuning parameters for swarm intelligence algorithms. This work dealt only with real and integer-valued parameters and not with categorical parameters. The experiments conducted were for 2 to 5 parameters and a minimum tuning budget of 240 to 480 runs respectively. There are also runs needed for the post-selection budget which would be at most twenty. The post-selection budget is used in the last part of this method where the best setting can be chosen from a pool of high performing settings.

Second, there are heuristic search methods. Meta-genetic algorithm (meta-GA) avoids the short comings of local searches by instituting a global optimization process (Grefenstette, 1986).

This work focused on tuning a limited number of parameters (six) utilizing 50 unique GAs. Due to the fact that GAs are randomized algorithms, Grefenstette (1986) chose the top 20 GAs after the first trial and then tested these for five more trials using different random seeds in order to choose the best performer. More recently, Brain and Addicoat, (2010) used meta-GA on five and eleven parameters when trying to find the lowest energy molecular conformers. Brain and Addicoat,(2010) found that the initial conditions played an important role in obtaining of a good configuration of the parameters, i.e. when the initial configuration was close to global optima the performance was good, but the further the distance the poorer the performance. ParamILS is a program that works on tuning parameters that are numeric(finite) and categorical by using an iterated local search method (Hutter et al., 2007; Hutter et al., 2009). This method is a sequential process capable of escaping local optima in order to perform multiple local searches. In order to reduce computational time needed for tuning, this program has a feature called *adaptive capping*, which adjusts the time limit given to explore a specific configuration of parameters by using the best solution time found. A drawback of adaptive capping is that the performance of a specific parameter configuration is not necessarily best for the entire course of the run (Hutter et al., 2009). Therefore, adaptive capping would be a disadvantage when dealing with instances that exceed the given time limit as it would be harder to gain any insight on the efficiency of the settings. Hutter et al., (2009) look at tuning 63 of CPLEX's parameters for two classes of mixed integer linear programming (MILP) problems in which all instances were solvable within the time limit provided. The use of ParmILS did not guarantee a configuration of parameters that outperforms the default settings but it did beat the default setting for one of the classes of MILPs. Ansótegui et al., (2009) describe a gender-based GA that races GAs in parallel to reduce computational time. By introducing gender separation, they achieve good performance and

robustness compared to other automatic algorithm configuration methods. Five hours of tuning were given to 20 runs with a limited number of parameters. Nannen and Eiben, (2006) developed a parameter calibration and relevance estimation (CRE) method (later referred to as relevance estimation and value calibration (REVAC)) in order to tune parameters of evolutionary algorithms. To estimate the number important parameters, the Shannon entropy is used, and for the experiment conducted in this paper. Nannen and Eiben, (2006) are able to reduce the number of continuous parameters calibrated from 13 to six. Smit and Eiben, (2009) improve REVAC by using sharpening methods and also racing methods (which are examples of sequential statistical testing) developed by Birattari et al., (2002), Balaprakash et al., (2007). Smit and Eiben, (2010) further improved REVAC by applying it to multiple instances within the training set. Baz et al., (2007) studied automated parameter tuning by using a set of similar instances when searching for the best set of parameters and machine learning to improve on the initial set of settings. They limit the number of parameters and the number of possible values each parameter can assume. Fischetti and Monaci, (2014a) developed a "bet and run" approach to improving performance of solvers by taking advantage of the inherent performance variability associated with tree search. Their approach is to initially solve the LP relaxation and then repeatedly change a starting condition and resolve searching a limited number of nodes until an optimal solution is found or a time limit reached. If the time limit is reached then choose the conditions that have given the best result.

The third type of automatic algorithm configuration problem takes a model-based optimization approach. Bartz-Beielstein et al., (2005) developed the sequential parameter optimization(SPO) method which uses Latin hypercube sampling to choose design points that are then used to evaluate the performance of the algorithm being tuned. A regression model and a

Gaussian correlation function are created in order to be able to estimate untested parameter configurations. SPO considers only six quantitative factors (parameters) and use single problem instances. Hutter et al., (2011) developed a sequential model-based algorithm configuration (SMAC) method to be able to tune categorical and continuous factors and utilize more than one instance for a training set. Their method is able to handle large numbers of parameters (76 for one of their experiments). The models used are based on random forests to take advantage of the fact that they perform well with categorical variables (Bartz-Beielstein and Markon, 2004; Baz et al., 2007). This is especially useful in the case of CPLEX, as categorical parameters dominate their parameter set. The time limit for each run is capped at five seconds which can be a draw back when hard instances are used because there may not be enough time to garner any useful results.

The fourth approach is to use experimental designs and analysis of variance (ANOVA) to tune parameters. Adenso-Diaz and Laguna, (2006) developed CALIBRA which tunes up to five parameters. CALIBRA uses fractional factorial experimental designs and local search methods to find good parameter values which may or may not be optimal. This method has equal or improved performance of the algorithm that has been tuned. Ridge and Kudenko, (2007) used ANOVA to distinguish important parameters using linear and quadratic models. The design used for the research was a resolution V face-centered composite (FCC) design which is a specific type of central composite designs (CCD) in which a set distance, from the center to axial points, guarantees axial points (green points in Figure 2.1) lie on the faces of the square defined by the embedded factorial design (red points in Figure 2.1). Figure2.1 illustrates a FCC design with only two variables. The experiment conducted needed 1452 runs in order to tune 10 parameters. Coy et al., (2001) use two-level fractional factorial designs involving six factors (parameters)

and then apply linear regression to develop the response surface. Steepest decent (minimizing objective) is applied to the response surface and continue until the best solution is the same after a pre-specified number of steps.

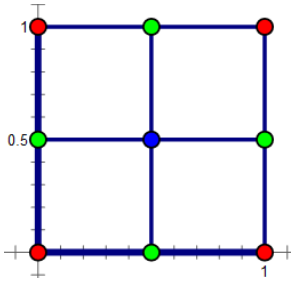


Figure 3-1 Illustrates a FCC design where the red points are the factorial design points, the blue point is the center design point, and the green points are the axial points which lie on the face of the square.

The last approach is sequential statistical testing. Examples of this are F-race and iterated F-race developed by Balaprakash et al., (2007); Birattari et al., (2002); Birattari et al., (2010) These methods evaluate the performance of each candidate setting on a set of instances and then eliminate poor performing settings once enough statistical evidence is collected (Birattari et al., 2010).

After reviewing the literature, this research provides methodology that fills a gap in this field of research by addressing the following:

- Tuning both a large and small number of parameters of optimization software to identify significant parameters that impact the efficiency of the optimization software while improving upon or remaining competitive to default settings.
- Tuning categorical, continuous, and ordinal variables (parameters).
- Work on a portfolio of instances of similar structure, which will be a class of MIPs that contain problems that are hard to solve.

- Report not only on the change in the performance of the optimizer but also incorporate results on the computational time needed to complete the tuning process by using the primal integral performance metric.

2.2 Benchmarking

Another important consideration for the practitioner is to determine which optimization software (MIP solvers) will provide the best performance for the types of problems they solve. Hans Mittelmann provides a website with benchmarking information for optimization software (Mittelmann, 2016). MIPLIB2010, Koch et al., (2011) is a test-bed library consisting of 361 instances of which 62% are classified as easy, 16% as hard, and 22% as not solved Mittelmann, (2016). Mittelmann's benchmarking work involving MIPs offers a comparison of leading commercial and open source optimization software using 24% of the MIPLIB2010 test-bed containing only 'easy' instances (Mittelmann, 2016). Commercial optimization software outperforms the open source in terms of the number of instances solved and the time it takes to find an optimal solution. Mittelmann's results are based on using the optimizers tested at default settings. For the easy problems, the average scaled time ranges between 1 to 7 seconds for the top three optimizers under default settings. This information would not offer the necessary insights needed for users working with more difficult problems.

In order to address the fact that MIPLIB2010's instances might not be as relevant, based on the current need of researchers, because many of the instances solve quickly, the researchers that curated MIPLIB2010 have decided to update the collection in their library. In the fall of 2016, MIPLIB placed a call for submission of relevant, challenging and real-world problems to

offer a modernized test-bed of MIPs in order to address the need for difficult test instances. Some criticism to this collection method are: suggested instances will not be diverse enough in type and level of difficulty, instances will be biased towards performing well with the researcher's developed work, the curators of the repository may not recognize a representative problem of a specific class because of the possibility of limited access to proprietary instances and emerging new problems (Hooker, 1995). Bowly et al., (2017); Hooker, (1995), recommend a more systematic approach of developing a testing pool of instances that will offer researchers a more robust group that provides a way to highlight algorithmic strengths and deficiencies.

Bowly et al., (2017) recent work developed a constructor generation approach to creating instances of LPs and MIPs. Although this work is promising because it tackles the limited diversity provided by simple random generation of instances, it needs strengthening in its ability to produce more difficult instances Bowly et al., (2017). This suggests that this process of collecting instances, does not ensure the diversity of the instances and how well they will perform at providing clarity into algorithmic strengths or weaknesses. This leads to the question as to what one hopes to gain from the information the benchmarking results provide.

2.3 Fundamental Concepts

In this section, relevant background information is provided for concepts used in the subsequent chapters of this dissertation.

2.3.1 Linear Programming

Research into linear programming problems dates back to the 1940's when developing efficient ways of supplying wartime efforts in order to reduce the cost of war for the United States and its allies became crucial. After World War II, industries took advantage of this research and used it to help increase their profitability and improve human life in a multitude of areas such as scheduling employees Hanssmann and Hess, (1960), locating placement of facilities such as factories or warehouses ReVelle and Swain, (1970), telecommunication networks Gendron et al., (1999), portfolio selection Pogue, (1970), and radiation treatments used in curing cancer in humans Sonderman and Abrahamson, (1985).

Linear Programming (LP) problems have a linear objective function maximized or minimized subject to linear constraints. Decision variables represent quantitative and/or logical decisions. The objective function is a mathematical expression written in terms of the decision variables and the coefficients of these variables. The decision variables represent the decisions to be made and can be any real number value as long as they satisfy the linear constraints. The coefficients of the decision variables express a factor of the degree that the decision variable contributes to the objective function. The constraints which are inequalities or equations reveal the amount of resources use by each decision variable while adhering to the resource limit expressed by the right-hand side of the inequality.

There are four types of possible solutions for an LP: no feasible solution, one optimal solution, unbounded and infinite optimal solutions (alternative optima). The optimal solution(s) lies in the feasible region and produces the largest (maximization) or smallest (minimization)

objective function value. When the solution space is unbounded, the objective value may increase (maximization) or decrease (minimization) indefinitely.

To illustrate an LP, consider the following problem:

Example 2.1

Pusateri's Market in Pittsburgh, Pennsylvania does a brisk lunch business. In order to meet the needs of the customers and reduce the line at the deli counter during peak times, the manager decided to offer two types of gourmet prepackaged lunches that change daily. A vegetarian and lean protein choice provides a profit of $8.75 per lunch and $10.50 per lunch respectively. It takes the chef five minutes to prepare every lean protein lunch and three minutes for the vegetarian lunch. It takes the deli clerk who packages all components of the lunch three minutes for every lean protein lunch and five minutes for the vegetarian lunch. The manager has allocated 60 minutes of prep time for the chef and 70 minutes for the deli clerk. The manager wants to know how many of each type of lunch should be made in order to maximize the profit obtained by the market within the time limits given for each employee?

Let x_1 = the number of lean protein lunches prepared.

x_2 = the number of vegetarian lunches prepared.

The objective function expresses the profit earned from each type of lunch and therefore needs to be maximized. The maximize overall profit (z) equals the sum of the profit from both types of lunches.

$\max z = 10.50x_1 + 8.75x_2$

subject to:

$5x_1 + 3x_2 \leq 60$ (chef prep time constraint)

$3x_1 + 5x_2 \leq 70$ (deli clerk time constraint)

$x_1, x_2 \geq 0$ (sign restrictions)

$x_1, x_2 \in \mathbb{R}$

When plotting the constraints and the sign restrictions the feasible region emerges. The feasible region is the set of all points that satisfy the constraints and sign restrictions of the LP. In Figure 2.2 the deli clerks time constraint is the green line, the chef's time constraint is the orange line, the sign restrictions are navy blue, and the feasible region is shaded in turquoise. Given that this problem offers no alternative optima, the vertices of the polygonal feasible region are the possible locations of the optimal solution. Alternative optima occur when there is an infinite number of optimal solution which form a line segment of the polygon.

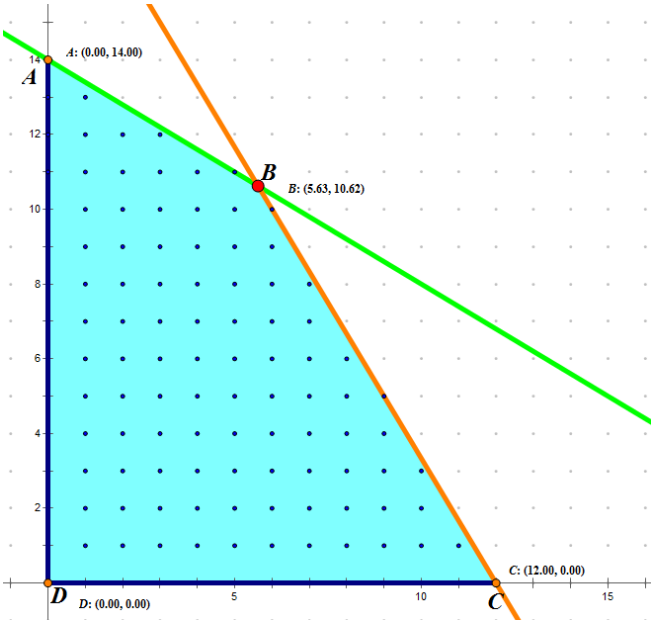


Figure 3-2 shows the graphical representation of the LP. Points A, B, C, and D are the corner points of the feasible region. Point B, in red, shows the location of the optimal solution.

Table 2.1 lists the corner points of the feasible region and uses them to evaluate the objective function. In Table 2.1, point B shows the location of the optimal solution. From the solution, the manager knows that preparing 5.625 lean protein lunches and 10.625 vegetarian lunches will provide the maximum profit of $152.03.

Table 3-1 contains the corner points, evaluation of the objective function and the objective function value at the specific corner point.

Point	$10.50x_1 + 8.75x_2$	Value of Objective Function
(0,0)	10.50(0) + 8.75(0)	$ 0.00
(12,0)	10.50(0) + 8.75(12)	$126.00
(5.625, 10.625)	10.50(5.625) + 8.75(10.625)	$152.03
(0,14)	10.50(0) + 8.75(14)	$122.50

2.3.2 Mixed Integer Programming

Mixed integer programming (MIP) problems have some decision variables that can only take on integer values. Below is an example of a MIP where x_1 and x_2 are the decision variables. In the objective function, $p = f(x_1, x_2)$, p represents maximized profit. Requiring x_2 to be an integer whereas x_1 is greater than or equal to zero, establishes this as a MIP. Consider the following problem as an illustration of a MIP:

Example 2.2 - Mixed Integer Programming

In this example, there will be one change from the previous example 2.1. In this example, the manager tells the chef, that lean protein lunches cannot be broken apart and sold piece by piece. This implies that the decision variable x_1, which is the number of lean protein meals prepared, will only assume integer values. The MIP that includes this change is stated below. The main change is that x_1 is now an element of the integers.

$$\max z = 10.50x_1 + 8.75x_2$$

subject to:

$$5x_1 + 3x_2 \leq 60 \qquad \text{(chef prep time constraint)}$$

$$3x_1 + 5x_2 \leq 70 \qquad \text{(deli clerk time constraint)}$$

$$x_1,\ x_2 \geq 0 \qquad \text{(sign restrictions)}$$

$$x_1 \in \mathbb{Z}$$

$$x_2 \in \mathbb{R}$$

The LP relaxation of a MIP is obtained by omitting all integer or binary constraints on decision variables. In Figure2.3 the feasible region of LP relaxation of the MIP is the polygon ABCD. Notice that by making a cut from point A to point E, we exclude a small portion of the turquoise region and obtain the convex hull of this MIP. The green point (6,10), seen in Figure 2.3, is the location of the optimal solution with an objective value of $150.50. This implies that six lean protein lunches and 10 vegetarian lunches should be produced giving the store a profit of $150.50.

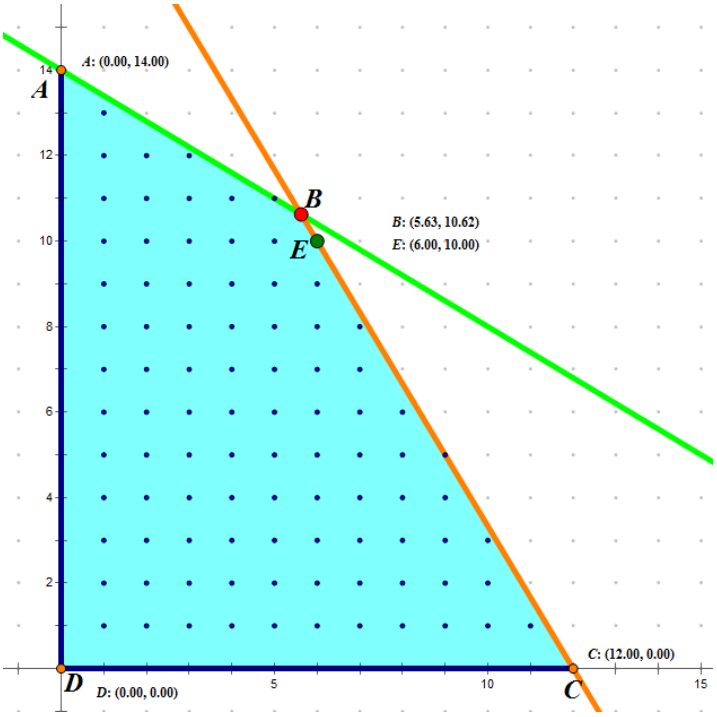


Figure 3-3illustrates the MIP in example two. Notice that the optimal solution must fall a point, shown in navy blue, red or green points. Point E, in green, is the location of the optimal solution.

2.3.3 Design of Experiments

The development of the field of experimental design grew out of the pioneering work of Sir R.S. Fisher in the 1920's (J. F. Box, 1980). The ideas of factorial design and ANOVA are attributed to Fisher (J. F. Box, 1980). Fisher researched the impact of experimentation methodology on data analysis in the field of agricultural sciences. Later in the 1950's, E.P. Box and K.B. Wilson were recognized for their development of response surface methodology (Telford, 2007). Collaborating with Box and many other influential statisticians during the 1950's and 1960's, Genichi Taguchi worked with orthogonal arrays, and developed quality improvement methods that changed the face of industrial production of goods and service in many different industries, but he is most noted for the impact his ideas had on quality control for Toyota Motor

Corporation during the 1970's (Telford, 2007). Designed experiments are widely accepted and utilized by researchers; and in the United States, the National Research Council recommends that all students learn about designing experiments, with a stronger emphasis beginning in middle school (National Research Council, 2012).

The experiments conducted for this research used some of the basic principles of design of experiments (DOE). The design dictates the number of computer runs and the settings to be tested. In addition, the design should aid in the analysis of the response in order to identify effects of the parameters and their interactions. There are many different types of designs available to researchers. Full factorial, fractional factorial, Plackett-Burman, D-optimal, and Bayesian D-optimal designs are discussed here and used in this research.

One type of desirable design would be an orthogonal array (OA). An OA (N, k, s, t) is $N \times k$ in size, where N (number of rows in the array) represents the number computer runs that will be necessary for the experiment and k (the number of columns in the array) represents the number of factors (parameters) being tuned, and where any t-columns projects into an equally replicated full factorial. The t represents the strength of the designs and it identifies the coverage of interactions of factors being tested. For example, if t=2 then all two-factor interactions are covered by the design. Two-factor interactions are the effects, if any, on the response, caused by the interaction of two different parameters. This means that the effect of one of the parameter on the response is different based on the value of the second parameter involved in the interaction. Aliasing is an effect that causes different factors, or interaction of factor to become indistinguishable from another. With an OA design the main effects are completely orthogonal of each other which means there is no correlation between them. In an OA, the number of times

each *t*-tuple occurs in every $N \times t$ sub-array must be equal. An example of an orthogonal array is OA (9, 4, 3, 2):

0 0 0 0
0 1 1 2
0 2 2 1
1 0 1 1
1 1 2 0
1 2 0 2
2 0 2 2
2 1 0 1
2 2 1 0

When a design is an OA then we know that the main effects are orthogonal which can make it easier to identify significant main effects. However large interactions could bias main effect estimates making it still difficult to identify significant main effects. Orthogonal arrays produce a situation ideal for developing statistical models which can be used for prediction. For example, the OA design helps identify important parameters that affect the efficiency of the optimization software. Full and fractional factorial designs, Plackett-Burman are all examples of orthogonal arrays.

An optimal design is one that is "best" with respect to some criterion (Montgomery, 2009). A D-optimal design minimizes the determinant of $(X'X)^{-1}$, which equivalently means to maximize the determinant of the information matrix X'X where X is the model matrix containing a column for each term to be estimated by the statistical model. D-optimal designs minimize the variance of model regression coefficients (Montgomery, 2008). D-optimal designs that are not orthogonal will produce effect estimates that are correlated which may make it more difficult to discern important parameters.

When the budget for experimentation is small it may be advantageous to use a Bayes D-optimal design. Bayes D-optimal designs maximize the determinant of the inverse of the

posterior covariance matrix, $\left|\left(X'X+K/\tau^2\right)^{-1}\right|$ where K is a $(p + q) \times (p + q)$ diagonal matrix whose first p diagonal elements equal zero and last q diagonal elements equal one, p is the number of primary terms, q is the number of potential terms, and τ^2 is the standard deviation of the prior distribution (DuMouchel and Jones, 1994). With the Bayesian approach the designer can specify factors that they believe are active, for example main effects, along with a list of potential effects that may be active, in this case two-factor interactions. Just like D-optimal designs, once a model has been selected, one can choose the number of design points beyond the minimum required by the model. However, Bayesian D-optimal designs offer the advantage of specifying a run size that is less than the total number of primary and potential terms. For example, if the experiment had 10 two-level, 10 three-level, and 5 four-level factors, then the minimum number of design points for a D-optimal would be 1,011 for a first order model with two-factor interactions. For a Bayes D-optimal design that considered all two-factor interactions as potential effects, the minimum number of design points would be 46. However, a good rule of thumb would to create a Bayes D-optimal design with about half the number of runs needed for a model with main effects and two-factor interactions, in order to have available degrees of freedom to estimate the potential two-factor interactions.

Covering arrays (CA) are an alternative to D-optimal designs. Using a CA allows for fewer runs than an optimal design which leads to a shorter tuning process. Software interaction testing uses covering arrays (Dunietz et al., 1997; Hoskins et al., 2004; Orso and Rothermel, 2014). A covering array $CA_\lambda(N; t, k, v)$ is a $N \times k$ array, where N is the number computer runs that will be necessary for the experiment and k is the number of parameter being tuned. The strength of the coverage of interactions is t and the number of different levels is v. The *strength t,* indicates that any t-columns in the array must contain all of the possible combinations of

parameter values for each parameter. The number of times each *t*-tuple occurs in every $N \times t$ sub-array is λ. Minimizing *N* optimizes a covering array for tuning because there would be a smaller budget for computer runs. For a specific number of factors, as the strength of a CA increases so does the number of runs. For example, CA_1 (30, 2, 6, 4) is a covering array with 30 computer runs and six parameters that are being tuned. Each of the six parameters can take on four different values. Since λ=1 and *t*=2, for every pair of columns, all possible combinations of factor levels appear together at least once in the CA. Covering arrays can be used with mixed level categorical and discrete variables.

A full factorial design includes all possible combinations of the parameters and their settings. A common type of factorial design consists of all factors having two levels and this is called a two-level design. The two levels are often referred to as the high and low level. The number of design points, *n*, in the full factorial experiment would be 2^k where *k* is the number of variables in the experiment. Two-level designs have k degrees of freedom for main effects and $n-k-1$ degrees of freedom for two-factor interactions and higher order interactions. An example of a two-level full factorial design is the 2^3. Table 2.2 shows the 2^3 design that has three factors and eight design points. Each factor in the 2^3 design is tested at a high and low level indicated with a plus, or minus respectively. In Figure 2.4, a geometric representation of the same 2^3 design is given.

Table 3-2 contains a full factorial two-level design of three factors with eight design points. A plus indicates to use the high level of the factor and a minus indicates that the factor should be placed at the low level.

Design Point Number	Factor A	Factor B	Factor C
1	+	+	+
2	-	+	+
3	+	-	+
4	+	+	-
5	-	-	+
6	-	+	-
7	+	-	-
8	-	-	-

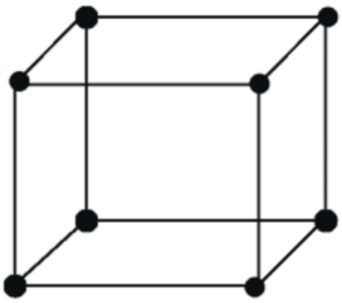


Figure 3-4 shows a geometric representation of a full factorial two-level design with three factors and eight design points.

In many situations, it is not cost effective or impossible due to other limitations, like time or availability of instrumentation etc., to conduct a full factorial experiment. In Table 2.3, the

number of design points needed for experiments with two through 10 factors are listed for a two-level design.

Table 3-3 list the number of variables in an experiment and the corresponding number of design points needed to have a full factorial design.

Number of Variables in the Experiment	Number of Design Points for a Full Factorial Two-Level Design.
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
5	$2^5 = 32$
6	$2^6 = 64$
7	$2^7 = 128$
8	$2^8 = 256$
9	$2^9 = 512$
10	$2^{10} = 1024$

When a full factorial design can't be used, a fractional factorial design may be the best possible design. A fractional factorial design is a design that has a fraction of the full factorial's design points. For example: A one-half fractional factorial design, of the 2^3 full factorial design would be expressed as a 2^{3-1}, and would have four design points, which is half the number of design points in the full factorial. An example of a 2^{3-1} design is in Table 2.4. A geometric representation of the one-half fraction design space is in Figure 2.5, one of the designs includes the four pink points and the other half would be the four green points. Notice neither design selects all four points from the same face because this would produce a situation where one of the factors would not be part of the experiment. Instead two points are chosen in such a way that all levels of each factor are tested two times. In this design, main effects are unbiased if all interactions are negligible. A main effect is the average effect of a factor, across all levels, on the response (Wu

and Hamada, 2011). Thus, if the researcher was only interested in the main effects of the factors and not the effect of two-factor interactions, then the 2^{3-1} design would permit the researcher to conduct half of the number of tests or computer runs. One drawback of fractional factorial designs is that information is lost. In the case of the 2^{3-1} design in Table 2.4, the information about the effects of two-factor interactions is lost because it becomes aliased with main effects. In this case the effects of the interactions of factors, AB, BC, and AC are aliased with main effects C, A, and B respectively.

Table 3-4 illustrates a 2^{3-1} design.

Design Point	Factor A	Factor B	Factor C
1	+	+	+
2	-	-	+
3	-	+	-
4	+	-	-

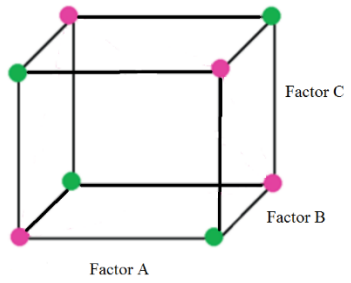


Figure 3-5 illustrates two different one-half fractional factorial design of a 2^3 full factorial design. The pink points represent one of the designs and the green the other design.

The resolution of a design, describes the degree to which the estimated main effects are aliased with estimated two-level or higher order interactions. The higher the resolution the better because this indicates that the main effects are aliased with higher order interactions which

typically are not significant effects. The aliasing properties of resolution III, IV, and V designs described by Mee, (2009), are in Table 2.5. Mee, (2009) suggests that resolution III designs should be avoided, but if used, then conduct follow-up experiments.

Table 3-5 The aliasing properties of Resolution III, IV, and V designs.

Resolution Type	Properties
Resolution III	Main effects may be aliased with two-factor interactions, but not each other.
Resolution IV	Main effects are not aliased with themselves or two-factor interactions. Two-factor interactions may be aliased with each other.
Resolution V	No main effects or two-factor interactions are aliased with each other, but they may be aliased with higher order effects.

Plackett and Burman, (1946) discovered a type of design that was efficient in that it needed fewer designs points to ensure that the main effects were not aliased with each other. The Plackett-Burman designs discovered were two-level, orthogonal, non-regular, resolution III having n-1 factors and n design points, where n is a multiple of four. Plackett-Burman designs are best utilized for screening large numbers of factors when experimentation is expensive. For example, if an experiment had 17 factors and wanted to screen for just main effects the smallest fractional factorial design, 2^{17-12}, would require 32 design points whereas a Plackett-Burman design would only need 20 design points, which is a savings of 12 design points. G. E. Box and Wilson, (1992) changed the utilization of Plackett-Burman designs by using the augmentation of

the resolution III design and a foldover of the resolution III design with n-1 factors, to produce a resolution IV design with n factors. An example of a mirror image foldover of the 2^{5-2} design is in Table 2.6. The design in Table 2.6 is a Plackett-Burman foldover design which offers efficiency in run size along with the ability to estimate main effects and some two-factor interaction making it an attractive choice for a screening design.

Table 3-6 contains a foldover Plackett-Burman design. It contains a mirror image foldover, design points 8 - 16, of a 2^{5-2} design.

Design Point	Factor A	Factor B	Factor C	Factor D	Factor E
1	-1	-1	-1	-1	1
2	-1	-1	1	1	-1
3	-1	1	-1	1	-1
4	-1	1	1	-1	1
5	1	-1	-1	1	1
6	1	-1	1	-1	-1
7	1	1	-1	-1	-1
8	1	1	1	1	1
9	1	1	1	1	-1
10	1	1	-1	-1	1
11	1	-1	1	-1	1
12	1	-1	-1	1	-1
13	-1	1	1	-1	-1
14	-1	1	-1	1	1
15	-1	-1	1	1	1
16	-1	-1	-1	-1	-1

The screening process will help identify the significant factors from a list of potential factors. To help guide one through the screening process there are three fundamental principles to consider: effect hierarchy, effect sparsity, effect heredity. The effect hierarchy principle states that lower order effects have a greater chance of being important than higher order effects; this principle is useful when screening a large number of factors with a low budget for the number of runs (Wu and Hamada, 2011). The effect sparsity principle states, the number of important effects is relatively small(Wu and Hamada, 2011). Although both of these empirical principles

do not always hold true, they were used when designing and conducting this experiment. The last guiding principle, effect heredity, states that an interaction between two or more factors can only be considered significant and added to the model if at least one of its parent's main effects are also in the model (Wu and Hamada, 2011). Effect heredity is used during the model selection process in order to narrow down the number of possible models.

2.3.4 Variable Types

There are three types of variables used in this experiment: quantitative (continuous and discrete), ordinal, and categorical. The variables in the experiment that are discrete are treated as continuous and therefore when the model produced by the experiment recommends a non-integer value for a parameter, that number is rounded and then used. Ordinal variables represent categories that have a logical order. For example, freshman, sophomore, junior, and senior are categories we give to students in high school and college. We could assign one through four to represent freshman through senior respectively, and those number would have meaning as oppose to just arbitrarily assigning a number to each category. Categorical variables represent categories that have no specified order. For example, fruit, vegetable, and protein are three classes of food. Assigning a number to represent the class of food gives us no additional information.

2.3.5 Degrees of Freedom

The number of degrees of freedom (df) available for the purpose of the experiment is important to keep track of when creating a design. For each observation or in this case computer run, we

gain a degree of freedom giving us N degrees of freedom. In regression analysis, there is one df used for the intercept and one df used for each continuous variable where g is the total number of continuous variables. For example, if there are 10 computer runs and two continuous variables that are to be estimated N = 10, g = 2, and 1 df is used for the intercept, then the number of df remaining will be equal to N - (g+1) = 10 - (2+1) = 7df remain. Categorical and ordinal variables use more degrees of freedom because of the way we must code the different levels. For example, if there is a categorical variable where the number of levels n is 5, then the categorical variable will need (n-1) degrees of freedom which in this example is equal to four. This implies that the number of observations needed to fit the specified regression model will increase more rapidly using categorical variables with more than two levels when compared to continuous variables. In regards to designing an experiment for parameter tuning, there is a need to balance the tradeoff between the number of parameters to be tuned and the budget for computer runs in order to maintain the number of degrees of freedom needed to estimate all parameters.

2.3.6 Variable Selection – Generalized Linear Models

Three types of variable selection methods were used in this research, forward selection, the least absolute shrinkage and selection operator (LASSO), and adaptive LASSO. Forward selection is a method used to reduce the number of predictor variables to those that are necessary and account for almost as much variance that was found with all of the predictor variables. For each predictor variable, the F statistics are calculated, and this shows the variables contribution to the model, the variable with the largest F statistic enters the until no remaining variable produces a significant F statistic. Forward selection brings in the regressor that most improves the fit given the term is significant at the level specified. The least absolute shrinkage and

selection operator (LASSO) (Tibshirani, 1996) is a regularization technique for simultaneous estimation and variable selection and is defined as $\hat{\beta} = \arg\min_{\beta} \left\| \mathbf{y} - \sum_{j=1}^{p} \mathbf{x}_j \beta_j \right\|^2 + \lambda \sum_{j=1}^{p} \left| \beta_j \right|$ where λ is a nonnegative regularization parameter, $\mathbf{x}_j$ are the regressors, and β_j are the parameter estimates of the coefficients. $\hat{\beta} = \arg\min_{\beta} \left\| \mathbf{y} - \sum_{j=1}^{p} \mathbf{x}_j \beta_j \right\|^2 + \lambda \sum_{j=1}^{p} w_j \left| \beta_j \right|$ is the adaptive Lasso (Zou, 2006). Notice that the only difference between the two are the weights, w_j, that can be assigned to different coefficients and these weights can be different values.

2.4 Test-bed of MIPs

When looking for a test-bed of instances to use to conduct the experiments, we looked for three classes of MIPs that provided ten to twelve instances of a similar type of problem. IBM ILOG CPLEX is and commercial optimization software capable of solving MIPs. The instances were chosen so that CPLEX could find at least one feasible solution (or optimal) in ten minutes when set to default settings. If default settings could not find at least one feasible solution, then potentially the instance would be too hard using any setting. Depending on the evaluation criterion, it is possible that if no setting found a feasible solution given the specific time limit, then the response for each design point would be equivalent, therefore providing no additional information about the factors that are significant in the solution process.

The test-bed used for all experiments in this paper are from the following three classes of MIP problems:

1. Class M - A formulation of the support vector machine with the ramp loss and L1-norm regularization (Hess and Brooks, 2015)
2. Class E - Survivable fixed telecommunication network design(Orlowski et al., 2010) (The mps files were obtained from (Raack, 2014).)
3. Class H – Coding theory graphs – node packing problems (Slone, 2011)

All of the instances used in the experiments can be expressed as a minimization problem of the form:

$$\tilde{x}_{opt} = \arg\min\left\{c^T x \mid Ax \le b, x_j \in \mathbb{Z} \text{ for all } j \in J\right\} \text{ with } A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n, \text{ and } J \subseteq \{1,...,n\}$$

2.2.1 Support Vector Machines – Class M

Support vector machines (SVM) are useful for classifying data. The goal of using the SVM is to find a hyperplane that will minimize the error in misclassifying data, while also maximizing the distance between the two correctly classified groups of data. The instances used are from Hess and Brooks, (2015), and are SVM with the ramp loss and L1-norm regularization, classified in their paper as (GSVM2-RL). In the SVM formulation s_i is the absolute value of the dual variable α_i. The data points x_i and y_i are the classification labels. The K represents the kernel function, z_i is the indicator variable used in conjunction with M, which represents big M in the formulation.

$$\min_{\alpha,s,b,\xi,\mu} \left\{ \sum_{i=1}^{n} s_i + C \sum_{i=1}^{n} (\xi_i + 2z_i) \right\},$$

subject to:
$$y_i \left(\sum_{j=1}^{j=n} y_j K(x_i, x_j) \alpha_j + b \right) \geq 1 - \xi_i + M z_i, \quad i = 1, 2, \ldots, n,$$
$$s_i \geq \left(\sum_{j=1}^{n} y_j K(x_i, x_j) \alpha_j + b \right) \geq -s_i, \qquad i = 1, 2, \ldots, n,$$
$$\xi_i \geq 0, \qquad i = 1, 2, \ldots, n,$$
$$z_i \in \{0, 1\}, \qquad i = 1, 2, \ldots, n.$$

The specific instance from Hess and Brooks, (2015), is listed in Table 2.7 along with preliminary solution times or gap after 10minutes, the number of rows, columns, non-zeros, continuous variables and integer variables.

Table 3-7 contains the SVM instance information including pereliminary solution time or gap after 10 minutes.

SVM Instance Name	Name of Specific Instance from the Model GSVM2-RL (Hess & Brooks, 2015)	Solution Time or Gap after a 10 min. Time Limit using Default Settings	# Rows	# Columns	# Non-zeros	# Continuous Variables	# Integer Variables
1.lp	n60d2BTj0F4.lp	7.81sec	180	241	4020	181	60(60binary)
2.lp	n60d5BTj0F4.lp	9.58 sec	180	241	4020	181	60(60binary)
3.lp	n60d10BTj0F4.lp	74.02sec	180	241	4020	181	60(60binary)
4.lp	n100d2BTj0F4.lp	12.63%gap	300	401	10700	301	100(100binary)
5.lp	n100d5BTj0F4.lp	48.71%gap	300	401	10700	301	100(100binary)
6.lp	n100d10BTj0F4.lp	15.40%gap	300	401	10700	301	100(100binary)
7.lp	n200d2BTj0F4.lp	43.12% gap	600	801	41400	601	200(200binary)
8.lp	n200d5BTj0F4.lp	77.51%gap	600	801	41400	601	200(200binary)
9.lp	n200d10BTj0F4.lp	81.78%gap	600	801	41400	601	200(200binary)
10.lp	n500d2BTj0F4.lp	95.46%gap	1500	2001	253500	1501	500(500binary)
11.lp	n500d5BTj0F4.lp	91.41 %gap	1500	2001	253500	1501	500(500binary)
12.lp	n500d10BTj0F4.lp	97.07% gap	1500	2001	253500	1501	500(500binary)

2.2.2 Survivable Fixed Telecommunication Network Design – Class E

Orlowski et al., (2007) thoroughly describe the survivable fixed telecommunication network design instances that are contained in the SNDlib. The MIP below is formulated for the telecommunication network instances chosen for the experiments and Table 2.8 defines the variables.

$$\min C + \sum_{e \in E} \left(\kappa_e z_e + \sum_{t \in T_e} k_e^t y_e^t + K_e \; f_e \right),$$

$$\text{Subject to: } \sum_{p \in P_d} x_p = h_d, d \in D$$

$$\mathrm{Y}_e := C_e + \sum_{t \in T_e} c_e^t y_e^t$$

$$f_e := \max \left\{ \sum_{p \in Q_e^+} r_p x_{p,} \sum_{p \in Q_e^-} r_p x_{p,} \right\}$$

$$f_e \le Y_e, e \in E$$

$$Y_e \le M z_e, e \in E$$

$$z_e \in \{0,1\}$$

$$\mathrm{x}_p \in \mathbb{R}_+$$

$$y_e^t \in \mathbb{Z}_+$$

Table 3-8 contains the definitions of all of the variables in the telecommunication network MIP.

C	The sum of all preinstalled cost values.
κ_e	Fixed charge cost. In these instances, $\kappa_e = 0$.
z_e	Indicator variable that indicates if the link is being used, $z_e = 1$ or not, $z_e = 0$.
k_e^t	Link capacity cost that occurs for each module t, on each link e.
y_e^t	The number of modules of type t, installed on link e.
K_e	Routing cost which is incurred for every unit (working or backup) of flow through the link e.
f_e	The maximum used capacity on a link e in any operating state.
x_p	Path flow variable that specifies the number of units of size r_d on the path $p \in P_d$
P_d	The set of all admissible paths with the same end nodes for each demand.
h_d	The demand values. $h_d \in \mathbb{Z}_+$

D	The set of point to point demands.
Y_e	The total capacity on link e.
C_e	The preinstalled capacity for each link e, where $C_e \in \mathbb{Z}_+$.
c_e^t	The capacity of each module t, where $c_e^t \in \mathbb{Z}_+$.
r_p	The routing unit of path p.
Q_e^+	The set of all routing paths traversing link e in a forward direction.
Q_e^-	The set of all routing paths traversing link e in a backward direction.
M	Big M, a sufficiently large enough fixed value.

The instances in Table 2.10 are from the SNDlib (Orlowski et al., 2010). The naming convention of the network models in the SNDlib are based on the options chosen, the first letter of the option chosen is in the name of the instance in the order presented in Table 2.10. These options are listed in Table 2.9.

Table 3-9 contains the attribute type and the options chosen when selecting telecommunication networks instance of a similar type from the SNDlib.

Attribute	Options Chosen for All Instances
Demand model	Directed demands
Link model	Bidirected links
Link capacity model	Modular link capacities
Fixed-charge model	No fixed-charge model
Routing model	Continuous
Admissible path model	All paths
Hop-limit model	No hop-limits
Survivability model	No survivability

Table 3-10 contains the telecommunications network instance information including preliminary solution time or gap after 10minutes.

Telecommunication Network Instance Number	SNDlib Name	Solution Time or Gap after a 10 min. Time Limit using Default Settings	# Rows	# Columns	# Non-zeros	# Continuous Variables	# Integer Variables
1.mps	Atlanta--D-B-M-N-C-A-N-N.mps	2.95 sec.	269	660	1936	616	44
2.mps	france--D-B-M-N-C-A-N-N.mps	1.85% gap	715	2205	6570	2160	45
3.mps	pdh--D-B-M-N-C-A-N-N.mps	126 sec.	178	703	2007	601	102
4.mps	pioro40--D-B-M-N-C-A-N-N.mps	0.47% gap	1738	6942	20648	6764	178
5.mps	polska--D-B-M-N-C-A-N-N.mps	1.04 sec.	168	396	1152	360	36
6.mps	ta1--D-B-M-N-C-A-N-N.mps	91.97 sec.	566	2606	7213	2001	605
7.mps	ta2--D-B-M-N-C-A-N-N.mps	2.18 sec.	2947	10101	29113	8913	1188
8.mps	cost266--D-B-M-N-C-A-N-N.mps	3.51% gap	1483	4275	12654	4104	171
9.mps	dfn-gwin--D-B-M-N-C-A-N-N.mps	83.08 sec.	216	1031	3001	941	90
10.mps	germany50--D-B-M-N-C-A-N-N.mps	5.26% gap	2526	8189	24479	8101	88
11.mps	norway--D-B-M-N-C-A-N-N.mps	28.83% gap	831	2754	8160	2652	102
12.mps	newyork--D-B-M-N-C-A-N-N.mps	22.55% gap	354	1568	4606	1470	98

2.2.3 Coding Theory – Class H

Node packing problems, also referred to as vertex packing problems, are a combinatorial optimization problem in which the objective is to select the maximum number of nodes in a graph such that no two nodes are adjacent. Node packing problems have a variety of applications such as, routing of trains, Zwaneveld et al., (2001), scheduling of machines, sensor coverage, harvesting of trees (Goycoolea et al., 2005; Synder and ReVelle, 1996; Weintraub and Murray, 2006), and coding theory. Tree harvesting is an easy example to understand the node packing problem. In order to limit soil erosion and loss of habitat, it is vital to not harvest a region that is next to another region that has recently been harvested. By finding the node packing of the forest regions, it is possible to spread out the harvesting while preserving the environment.

The set of instances used for this experiment represent binary correcting codes. These codes have many applications but one way many people experience, unbeknownst to them, using these codes is when they are using the internet. Anytime users interface with the internet, a

transmission of data occurs, almost flawlessly. This seemingly flawless transmission is due to the use of error correcting codes. Unseen by users of the internet is the numerous times data fails to reach its destination, called packet loss, and corruption of data, both of which often occur when the network is being heavily used and is experiencing network congestion. Correcting codes are special because they provide a way for receiver of corrupted data to fix the problem and decode the information.

A graph is one way to visualize a small node packing problem. A graph $G = (V, E)$ is a set of vertices V (nodes), and edges E. In the realm of coding theory, the words of the code (codewords) are the vertices of the graph and the edges usually represent the Hamming distance between code words. The Hamming distance $d(x, y)$, between words x and y is found by calculating the number of changes you need to change x into y or vice-versa. For example, the Hamming distance between the two code words 1111, and 1100 is $d(1111,1100) = 2;$ and the Hamming distance between 01110, and 01010 is $d(01110,\ 01010) = 1.$ Given the Hamming distance of a specific code d_{code}, the number of errors that are detectable, e_d (bit errors) is $e_d \leq d_{code} - 1$. For example, if the Hamming distance is three, then it is possible to detect up to two errors. The Hamming distances can also be used to calculate the number of errors that can be corrected, e_c which is $e_c \leq \frac{d_{code} - 1}{2}$. Continuing with the previous example, if the Hamming distance is three, then the code can correct one error.

The naming of an edge of a graph, is done by listing the two vertices at the endpoints of the edge. For example, edge e = {a, b} where a, b $\in V$. Edges may also have weights which designate the cost or benefit of utilizing an edge. A node packing contains a set of vertices $V' \subseteq V$

such that there is no edge $\{a, b\} \in E$ for any $a, b \in V'$. Therefore, the objective of the node packing problem is to find the largest set of vertices in a graph, such that no two vertices are adjacent. Figure 2.6 a-c, illustrate the node packing, a non-node packing, and an node packing that is not at maximum level respectively.

a.) Node Packing

b.) Non-node Packing

c.) Node packing that is not at maximum level

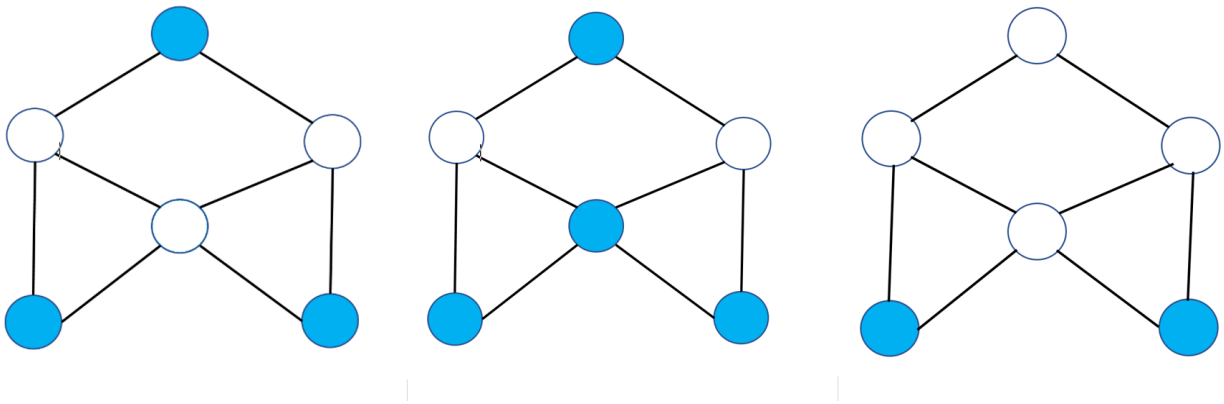


Figure 3-6 a-c contains graphs illustrating examples/non-examples of node packing.

Equivalent to node packing problem is the maximal clique problem on the graphs complement. A clique is a subset of vertices on an undirected graph where every two distinct vertices are adjacent. The complement of a graph G will have the same vertices that are in graph G such that two distinct vertices are adjacent (connected with an edge) if and only if they are not adjacent in graph G. Figure 2.7 a-c illustrates the maximal clique, a non-clique, and a clique that is not maximal respectively. Note that the graphs in Figure 2.7 a-c are the complement of the graphs in Figure 2.6 a-c.

a.) Maximal Clique b.) Not a Clique c.) Clique – Not Maximal

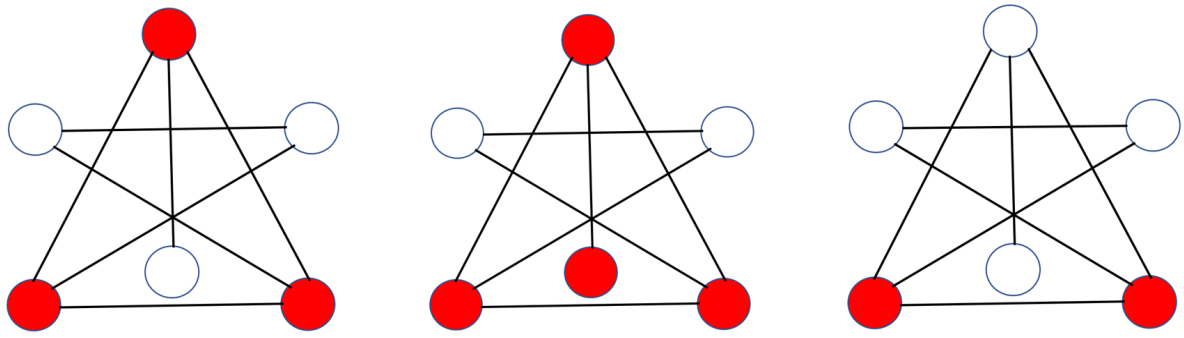


Figure 3-7 a -c illustrates the maximal clique, a non-clique, and a clique that is not maximal respectively.

The basic formulation of a weighted node packing problem is the following:

$$\text{Max } Z = \sum_{i=1}^{n} w_i x_i$$

$$\text{Subject to: } x_i + x_j \le 1 \qquad \forall \{i, j\} \in E$$

$$x_i \in \{0,1\} \qquad \forall\ i \in E$$

Where i is the index of the nodes in V, w_i is the weight associated with node i and the corresponding decision variable x_i. If $x_i = 1$ then vertex i, is selected in the packing.

There are many different types of error correcting codes and this class of instances has five different types and can be identified by the name of the instances first three characters as seen in Table 2.11. If the instance name begins with 1dc, 2dc, 1zc, 1tc, and 1et, then the corresponding type of codes are single deletion correcting, double deletion correcting, single asymmetrical error corrected (also known as a Z channel error), correcting single transposition, and correcting single transposition with end wraparound, respectively.

Although, the coding theory class of instance are maximization problems and are described in this section as such. For interpretation of the performance metric, the primal integral, to be consistent across the three classes of instances, the instances for the coding theory graphs were converted to minimization problems. Table 2.11 contains coding theory instance information, including preliminary solution time or gap after 10 minutes, the number of rows, the number of columns, the number of non-zeros, the number of continuous variables and the number of integer variables. In the coding theory class, all of the integer variables are binary.

Table 3-11 contains the coding theory instance information including preliminary solution time or gap after 10 minutes.

Coding Theory Instance Name	Node Packing Instance Name	Solution Time or Gap after a 10 min. Time Limit using Default Settings	# Rows	# Columns	# Non-zeros	# Continuous Variables	# Integer Variables
1.lp	1dc.1024.txt	26.25% gap	24063	1024	48126	0	1024 (1024binary)
2.lp	1dc.2048.txt	32.26%gap	58367	2048	116734	0	2048 (2048binary)
3.lp	2dc.1024.txt	32.37 % gap	169162	1024	338324	0	1024 (1024binary)
4.lp	2dc.2048.txt	56.82% gap	504451	2048	1008902	0	2048 (2048binary)
5.lp	1zc.512.txt	6.45%gap	13824	512	27648	0	512 (512binary)
6.lp	1zc.1024.txt	12.73%gap	33280	1024	66560	0	1024 (1024binary)
7.lp	1zc.2048.txt	42.44%gap	78848	2048	157696	0	2048 (2048binary)
8.lp	1tc.2048.txt	6.77%gap	18944	2046	37888	0	1504 (1504binary)
9.lp	1et.1024.txt	4.37%gap	9600	1022	19200	0	1022 (1022binary)
10.lp	1et2048.txt	7.45%gap	22528	2046	45056	0	2046 (2046binary)

2.5 Performance Metrics

Careful thought was given to the selection of the performance metric used for the experiments conducted. There are at least four performance metrics that can be used to compare the performance of a solver. First, there is the time needed to find the first feasible solution. This metric is good to use when the practitioner needs a feasible solution in the shortest amount of time and solution quality is not a concern or a low priority. For the experiment conducted with a

limited number of parameters, all instances found a feasible solution in ten minutes or less. However, it is possible that settings would cause the optimization software to have such a poor performance, that no feasible solution would be found within the time limit, and this did happen when experimenting with 59 parameters. If no incumbent was found within the time limit then this metric would only inform the user that the setting did not perform as well as others within the time limit. So not only would you not have solution quality, you would also not know when the first solution is found. Thus, assigning a numeric quantity for the response may cause bias.

The second metric is the time needed to find the optimal solution. This solution can be proven optimal when the gap between the primal objective value and the dual objective function value is zero or when the gap between the upper bound and lower bound of the objective function value is zero. This metric focuses on solution quality but ignores suboptimal solutions which may have been close to optimal, and the suboptimal solutions may be attractive for practical use. Unfortunately, there is the potential that a solver would never find an optimal solution or find one and not be able to prove it is optimal. Proving optimality is time consuming especially when considering the connection between the number of parameter settings and difficulty in proving optimality is not clear.

The third metric often used is the time needed to find a solution within a certain gap to optimality. The optimality gap is percent difference in the incumbent solution's objective value (upper bound for a minimization problem) and the lower bound (best bound) of the objective value function. This metric tries to balance the need for quickly found feasible solution with the solution quality. The optimality gap chosen by the practitioner is an arbitrary value based on the user's experience with the problem. However, the optimality gap chosen may be a random guess when the practitioner has no prior experience with the problem being solved. The drawback of

using this metric for the experiment conducted in this research is that is sets up an experiment where the amount of time needed to conduct the experiment is an unknown quantity. Once again this is problematic due to the large number of parameter settings being tested.

The fourth metric, the primal integral, was proposed in Berthold, (2013); Achterberg et al., (2012) and used in Fischetti and Monaci, (2014b), Fischer and Pfetsch, (2015), and Boland et al., (2016). The different metrics are listed in Table 2.12 which highlights the focus and drawback of the metrics. Listed as the fourth metric in Table 2.12, the primal integral considers both the time to finding a feasible/optimal solution and the solution quality for the entire time that optimization is taking place.

Table 3-12 List four types of metrics considered for the experiment and list what the focus of the metric is and its drawback.

Metric Types	Focus	Drawback
1. Time to first feasible solution	Speed to first feasible solution	Solution quality
2. Time to optimal solution	Solution quality	Ignores suboptimal solutions (attractive in practice)
3. Time to find solution within a certain gap to optimality	Tries to balance between metric 1 and metric 2	May not reach gap within a certain time limit,
4. Primal Integral (Berthold, 2013)	Considers trade-offs between speed of finding a feasible solution and the quality of the solution over the entire optimization or a chosen time limit given by the user.	May not agree with metric 2.

Below the primal integral metric developed by Berthold (2013) is defined.

Let $\overline{z}_{opt}$ denote the optimal objective function value for a given MIP problem and $z(t)$ be the value of the best-known objective function value found at time t. The *primal gap function* p can be computed as:

$$p(t) = \begin{cases} 1 & \text{if no incumbent found until time } t. \\ \gamma(z(t)) & \text{otherwise} \end{cases}$$

where $\gamma(\cdot) \in [0,1]$ is the primal gap, and is defined as follows:

$$\gamma(z) = \begin{cases} 0 & \text{if } |\tilde{z}_{opt}| = |z| = 0, \\ 1 & \text{if } \tilde{z}_{opt} \cdot z < 0, \\ \dfrac{z - \tilde{z}_{opt}}{\max\{|\tilde{z}_{opt}|, |z|\}} & \text{otherwise.} \end{cases}$$

The value of the primal integral of a run until time $t_{\max}$ is defined as $P(t_{\max}) = \int_0^{t_{\max}} p(t)dt$ and measures the quality of the primal heuristics. Primal heuristics are procedures used to find integer feasible solutions early in the search tree during the branch-and-bound algorithm (Bertacco, 2006; Berthold, 2013). Primal heuristics improve the upper bound for a minimization problem whereas cutting planes are used to strengthen the lower bound of an optimal solution (Bertacco, 2006). The sooner an incumbent solution is found, the smaller $P(t_{\max})$. The implication of this, and the reason this metric was used in this paper, is that when comparing runs with different CPLEX parameters settings, this metric favors finding high-quality solutions earlier in the optimization process. The ability to consider both solution quality and the time needed for optimization, or a chosen time limit given by the user, makes this metric ideal for the research conducted.

Figure 2.8 illustrates how parameter settings for CPLEX can affect the progression of three runs. This can easily be seen in the plot of their primal gap functions, p(t). The red line, *primalGap1* shows an example of a run where no incumbent solutions were found in the 600

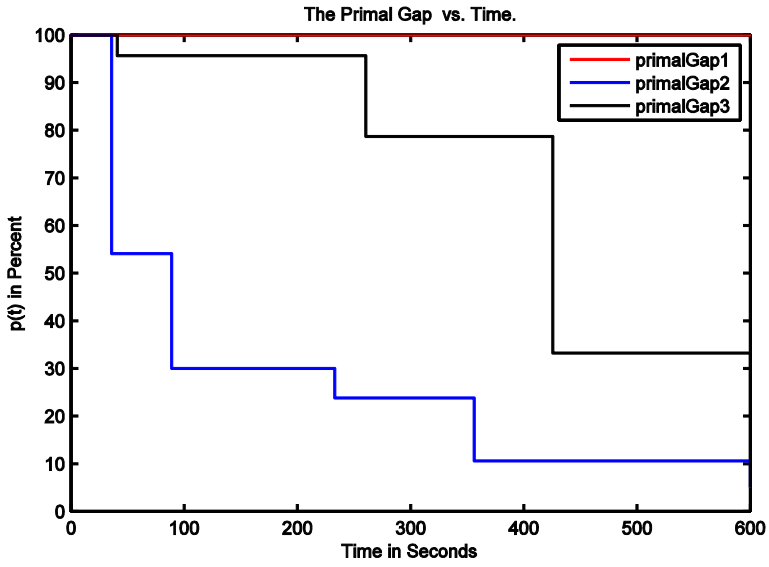


Figure 3-8 Shows the primal gap function of three runs of the same MIP instance with different parameter settings. The blue function would generate the smallest primal integral value indicating that one should choose that setting for the parameters in order to attain the best solution values anytime during the 600 seconds.

seconds given to the solver. The value of the primal integral, $P(t_{\max})$, would be the largest value possible because the area under the red line is the largest. The blue, *primalGap2*, would produce a $P(t_{\max})$ that would be the smallest, because the solver reduced the primal gap by finding better incumbent solutions earlier than *primalGap3*, shown in black.

Chapter 4 Limited Experiment

3.1 Background

Optimization solver's efficiency in solving a multitude of different types of mathematical programming problems are impacted by the solver's parameter settings and the structure of the instance being solved. Although improvements in solving some MIPs have been made, the time it takes to solve an instance of a MIP increases rapidly as the size of the instance grows (as measured by the number of variables and/or constraints), thus making some problems difficult to solve in practice. Tuning the parameters of the solvers offers the practitioners another avenue to pursue that can have a significant impact on the time it takes to obtain a feasible or optimal solution.

In this chapter, a designed experiment approach was used to fit a model that would suggest a setting for six CPLEX parameters, described in Table 3.1, that provided the greatest impact on the performance metric. Determining good parameter settings for a specific solver and class of instances can reduce the computational requirements needed to produce a feasible or optimal solution. In doing so, important parameters and interaction of parameters can be identified, along with parameters that do not have a significant impact on the efficiency of the solver and are therefore unnecessary to tune. In designing a parameter tuning experiment for a portfolio of similar instances we gain the added value of having a recommended setting that

should work well on any future instance with similar structured MIPs, and valuable insights into the structures of the class of instances being solved.

The experiment conducted was one with six CPLEX parameters, used also in Baz et al., (2007), for the first experiment and they are listed below along with the number of different values they can assume.

- Solving approach parameter (5)
- Node selection parameter (4)
- Branching parameter (6)
- Diving parameter (4)
- Generate Gomory cuts parameter (4)
- Generate Mixed Integer Rounding cuts (4)

Each setting will require a computer run, the solving of a MIP instance with a time limit of ten minutes, for each of the thirty-four instances (all classes) of MIPs. In order to identify the best parameter's setting, an exhaustive look at the results from the full factorial of settings was conducted. The number of runs in the full factorial can be calculated by multiplying the number of levels of each parameter together ($4 \cdot 4 \cdot 4 \cdot 4 \cdot 5 \cdot 6$) resulting in 7,680 runs for each instance. This was done to judge the performance of an individual setting, when compared to CPLEX default parameter settings and competing designs. To estimate the amount of time it would take to run this experiment, multiply the number of instances, number of settings, and the time limit together, $(34 \cdot 7680 \cdot 10)$ which is 2,611,200 minutes. With only 1 processor, this would take about 1813.3 days of computing time. The experimental runs took about 23 days running on a Linux Beowulf cluster using 70 of the 500 processors it contains, 2.6 GHz Opteron, 1 TB RAM

(4GB-32GB per node), 2TB direct attached Fiber Storage, and 16.8 TB internal disk storage (73GB per node).

Table 4-1- Listed are the six parameters explored and their settings. CPLEX's default setting is identified in bold. The information in this table comes from the CPLEX parameter manual International Business Machines Corporation, 2009)

Parameter	Settings
MIPEMPHASIS	0 Balance optimality and feasibility; **default** 1 Emphasize feasibility over optimality 2 Emphasize optimality over feasibility 3 Emphasize moving best bound 4 Emphasize finding hidden feasible solutions
NODESEL	Controls the approach CPLEX uses for selecting the next node to process when backtracking 0 Depth-first search 1 Best-bound search; **default** 2 Best-estimate search 3 Alternative best-estimate search
VARSEL	Controls the type of branching used in the branch and bound algorithm. -1 Branch on variable with minimum infeasibility 0 Let CPLEX choose variable to branch on; **default** 1 Branch on variable with maximum infeasibility 2 Branch based on pseudo costs 3 Strong branching 4 Branch based on pseudo reduced costs
DIVETYPE	Controls the MIP traversal strategy for performing probing dives. 0 Let CPLEX choose; **default** 1 Traditional dive 2 Probing dive 3 Guided dive
FRACCUTS	Controls the production of Gomory fractional cuts. -1 Do not generate Gomory fractional cuts 0 Let CPLEX choose; **default** 1 Generate Gomory fractional cuts moderately 2 Generate Gomory fractional cuts aggressively
MIRCUTS	Controls the production of Mixed Integer Rounding cuts. -1 Do not generate MIR cuts 0 Let CPLEX choose; **default** 1 Generate MIR cuts moderately 2 Generate MIR cuts aggressively

Comparisons of results from the recommended setting of the statistical model, created with the response of the computer experiment using a D-optimal design with 134 design points, with the recommended setting of CPLEX's automated tuner, the best run of two designs produced by the Selection Tool for Optimization Parameters (STOP) Baz et al., (2007) and covering arrays produce with JMP statistical software is provided using the geometric mean of the primal integral metric. The $\text{geometric mean} = \left(\prod_{i=1}^{n} r_i \right)^{\frac{1}{n}}$ where r_i represents the response of each instance *i* for a specific setting, and *n* is the number of instances in the class of MIPs. CPLEX's automated tuner can tune individual instances and a portfolio of instances. In this paper, we focus on the latter because our interest is to tune parameter settings for a class of instances. STOP's methods of producing parameter settings are pairwise coverage, greedy heuristic and random design. First, the pairwise coverage method use by STOP and developed by Cohen et al., (1997), produces an array of strength two which means that all pairs of parameter values will appear at least one time in a design (Baz et al., 2007). The pairwise coverage method in STOP also lets the user create a coverage array where all pairs of parameter settings appear twice in the design (Baz et al., 2007). Second, the greedy heuristic is one in which, after the first randomly selected parameter setting is produced, then the next setting is one in which the new setting minimizes the maximum number of parameter values in common with the previous settings (Baz et al., 2007). Third, the random parameter setting values are selected uniformly at random (Baz et al., 2007). The random method can miss parameter interaction, whereas the first two methods are trying to ensure that interactions are included in their runs (Baz et al., 2007). In this paper, the first two of STOP's methods of producing designs was used for comparison.

3.2 Methodology

The full factorial of all parameter settings was initially run because we wanted to ascertain the quality of our results in order to provide a proof of concept for an experiment with a larger number of factors. Also with the full factorial results it is possible to answer questions like the following:

- Was the model created able to recommend the best possible setting and if not, how did it rank when compared to all other settings?
- How much room for improvement above the default setting is there not only by ranking but also in the difference between the responses?

The basic principles of design of experiments (DOE) were used in the development of the designs. Although an orthogonal array would be preferred, a D-optimal design was used for two reasons. First an OA would likely require too many runs and second, there does not exist a table containing all possible OAs for every situation. The D-optimal designs were created using JMP12.0 statistical software. Each design is based on the number of factors, the type of factor (categorical, ordinal, or quantitative-continuous), the number of levels for each factor, and the type of model to be fitted with the results. In this work, a first order model with two-factor interactions is sufficient.

This experiment has 6 categorical factors. However, two of the factors FRACCUTS and MIRCUTS, could be treated as continuous due to their ordinal structure (Agresti, 2010; Rhemtulla et al., 2012). The structure of factor, if we remove the default setting which is let CPLEX choose, is the following: do not generate cuts, moderately generate cuts, and aggressively generate cuts. By changing these two factors to continuous, the number of runs needed for D-optimality is reduced 29% from 204 to 134. The number of runs are based on the

degrees of freedom needed to estimate the 6 main effects, 15 two-factor interactions and error for the model. Table 3.2 shows how many df each main effect and two-factor interaction need in order to be estimated in the statistical model. When interpreting the recommended setting for the two factors now considered continuous, rounding of any decimal recommended value was used to obtain the parameter value entered into CPLEX.

Table 4-2 shows the number of df need to estimate each ME and 2fi when all factors are categorical and in the last column the df are for when x1-x4 are categorical (representing mipemp, nodesel, varsel, and divetype) and x5 and x6 are continuous (representing fraccut and mircuts). In the last column entries that are darkened are where the in the reduction in the number of runs is attributed.

Main Effect (ME) or Two-factor Interaction (2fi)	Factor or 2fi	Degrees of Freedom All Factors Categorical	Degrees of Freedom X5 and X6 are Continuous
ME	x1	4	4
ME	x2	3	3
ME	x3	5	5
ME	x4	3	3
ME	x5	3	1
ME	x6	3	1
2fi	x1x2	12	12
2fi	x1x3	20	20
2fi	x1x4	12	12
2fi	x1x5	12	4
2fi	x1x6	12	4
2fi	x2x3	15	15
2fi	x2x4	9	9
2fi	x2x5	9	3
2fi	x2x6	9	3
2fi	x3x4	15	15
2fi	x3x5	15	5
2fi	x3x6	15	5
2fi	x4x5	9	3
2fi	x4x6	9	3
2fi	x5x6	9	3
error	error	1	1
Total ------>		204	134

With the results of the computer runs of the full factorial combination of all parameter settings, matching any design point with the corresponding response was all that was necessary to gather the data to evaluate. This was done for each instance in every class of MIPs. The data obtained from the optimization process and follow-up calculations included the starting time of the optimization, time when each incumbent (feasible or optimal) solution was found, the value

of the objective function each time an incumbent was found, the bestbound at the end of ten minutes if the optimal solution was not found, and the primal integral value at each time an incumbent was found (including the ending time of ten minutes). The geometric mean of the primal integral value for each parameter setting was also calculated using the instances for each class of MIPs. After we had paired the D-optimal design (created in JMP) with the results, a statistical model was produced so that we could make recommendations for the parameter settings for each class. The design used can be found in appendix A.

Model fitting was performed using JMP 12.0 statistical software and a first order model with two-way interactions was fit. For the experiments, there is the dependent variable *y*, which is the geometric mean of a chosen metric (primal integral) for each design point, and *k* independent variables x_1, x_2, …, x_k, which are the parameters of an optimization solver. The general linear regression model can be written in matrix notation as $y = X\beta + \varepsilon$

where $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{k1} \\ 1 & x_{12} & x_{22} & \cdots & x_{k2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{1n} & x_{2n} & \cdots & x_{kn} \end{bmatrix}, \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}$, and $\varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$. The type of model considered is reflected in the model matrix X which contains the variables for the main effects, two-factor interactions and the indicator variables for categorical variables that have more than two levels, where y_n is the response variable, β_0 is the intercept, β_k is the coefficient of the main effect and the interaction terms, and ε_n is the random error. The model matrix contains main effects and two-factor interactions.

The model for all of the classes was developed using the response from the computer runs that were obtained from a D-optimal design of 134 runs. Forward selection method using AICc, Hurvich andTsai, (1989) criteria for validation and to prevent overfitting, was used in the

variable selection process and to fit a first order model with two-factor interactions. The corrected Akaike's information criterion (AICc), which is the criterion used in the model selection process, is defined as follows:

$AICc = AIC + \frac{2k(k+1)}{n-k-1}$ Hurvich and Tsai, (1989) where $AIC = 2k - 2\ln(L)$ (Akaike, 1973). In these formulae L is the maximum likelihood value for the model, n is the sample size and k is the number of estimated parameters in the model.

The effects test table identifies the significant main effects and two factor interactions. The first column of the effects test table in Figure 3.1 is *Sources*, which list the effects that are in the model. *Nparm* shows the number of parameters associated with an effect. The number of parameters for a continuous variable is one, a categorical variable will be one less than the number of levels, and for crossed effect (an interaction) it is the product of the number of parameters for each individual effect. The column with *DF* contains the degrees of freedom. The next column is the test statistic and in this case, because we have categorical factors, a *Wald Chi Square* statistic is given. The next column is the *p-value* and any value less than .05 will be red to indicate that the effect is significant.

Looking at Figure 3.1 which is the effects test for the telecommunication networks class, we see that both quadratic terms fraccut*fraccut and mircuts*mircuts are removed from the model which indicates that there is no fraccuts and mircuts do not have quadratic effects. We also notice that the main effects for the divetype and fraccut parameters are removed which implies that these parameters alone do not have a significant main effect. Removing the fraccut parameter indicates that producing Gomory cuts, which are relatively easy to generate, does not significantly help the optimizer progress to the optimal solution. However, divetype is part of a

significant interaction with nodesel and with help of the profiler feature in JMP, it can be shown that when both parameters are at their low level, indicating that when selecting the next node to explore, the depth first search combined with letting CPLEX choose the type of dive to take to traverse the nodes, may cause the geometric mean of the primal integral to become large. The model summary in Figure 3.1 shows a good generalized R^2 value of 0.9553. However, we can still see that several more two factor interactions are not significant.

Effect Tests

Source	Nparm	DF	Wald ChiSquare	Prob > ChiSquare	
varsel	5	5	492.86264	<.0001*	
mipemp*nodesel	12	2	34.220471	<.0001*	Levels removed: 10
nodesel*varsel	15	4	16.174359	0.0028*	Levels removed: 11
nodesel*divetype	9	2	6.1093694	0.0471*	Levels removed: 7
mipemp*varsel	20	5	10.940241	0.0526	Levels removed: 15
varsel*mircuts	5	1	1.6572366	0.1980	Levels removed: 4
nodesel	3	2	2.4721254	0.2905	Levels removed: 1
divetype*fraccut	3	1	0.8284109	0.3627	Levels removed: 2
fraccut*mircuts	1	1	0.7110276	0.3991	
varsel*divetype	15	2	1.5744013	0.4551	Levels removed: 13
mircuts	1	1	0.4569706	0.4990	
nodesel*mircuts	3	2	1.3754356	0.5027	Levels removed: 1
mipemp*fraccut	4	3	2.2116995	0.5296	Levels removed: 1
mipemp*divetype	12	4	2.977728	0.5616	Levels removed: 8
mipemp*mircuts	4	1	0.144452	0.7039	Levels removed: 3
varsel*fraccut	5	1	0.0785077	0.7793	Levels removed: 4
divetype*mircuts	3	1	0.0158729	0.8997	Levels removed: 2
mipemp	4	1	2.2925e-5	0.9962	Levels removed: 3
divetype	3	0	0	1.0000	Removed
fraccut	1	0	0	1.0000	Removed
nodesel*fraccut	3	0	0	1.0000	Removed
fraccut*fraccut	1	0	0	1.0000	Removed
mircuts*mircuts	1	0	0	1.0000	Removed

Model Summary

Response	t600
Distribution	Normal
Estimation Method	Lasso
Validation Method	AICc
Mean Model Link	Identity
Scale Model Link	Identity

Measure	Training
Number of rows	134
Sum of Frequencies	134
-LogLikelihood	142.37778
BIC	485.56699
AICc	404.19034
Generalized RSquare	0.9553778

Figure 4-1 shows the Effect Test and the Model Summary after the LASSO method was applied for model.

After removing more insignificant interactions, the resulting model includes the effects listed in the Effects Test section of Figure 3.2. The model's generalized R^2 value is 0.94. Using the model produced, we then predict settings for each parameter so that the response variable, in this case the primal integral value, is minimized. This is done by using the profiler feature in JMP in which a desirability function along with importance weight is set by the user and applied to the

model. Here the desirability function was set to minimize both the geometric mean of the primal integral and the geometric variance. Importance weight were varied from the geometric mean of the primal integral being assigned weights of 1, .9, .8, .75 while the geometric variance was weighted 0, .1, .2, .25 respectively. In Figure 3.3 we see the results of this with the output of the prediction profiler. For an instance from Telecommunication network class the recommended settings are to set mipemphasis=0, nodesel = 1, varsel = 2, divetype = 0, fraccuts = -1 and mircuts = -1. Along the left in red is the estimated value of the geometric mean of the primal integral for the recommended setting.

Effect Tests

Source	Nparm	DF	Wald ChiSquare	Prob > ChiSquare	
varsel	5	3	466.05697	<.0001*	Levels removed: 2
mipemp*nodesel	12	1	43.547265	<.0001*	Levels removed: 11
nodesel	3	1	15.349145	<.0001*	Levels removed: 2
mircuts	1	1	6.3729734	0.0116*	
nodesel*divetype	9	1	4.8652572	0.0274*	Levels removed: 8
mipemp	4	2	5.1700442	0.0754	Levels removed: 2
fraccut*mircuts	1	1	1.9031467	0.1677	
mipemp*fraccut	4	1	1.2658064	0.2606	Levels removed: 3
mipemp*varsel	20	5	5.0641625	0.4081	Levels removed: 15
nodesel*varsel	15	3	1.9643837	0.5798	Levels removed: 12
varsel*fraccut	5	1	0.176045	0.6748	Levels removed: 4
mipemp*divetype	12	2	0.7104142	0.7010	Levels removed: 10
nodesel*mircuts	3	1	0.1465875	0.7018	Levels removed: 2
divetype*fraccut	3	1	0.0538617	0.8165	Levels removed: 2
divetype	3	0	0	1.0000	Removed
fraccut	1	0	0	1.0000	Removed

Model Summary

Response	t600
Distribution	Normal
Estimation Method	Adaptive Lasso
Validation Method	AICc
Mean Model Link	Identity
Scale Model Link	Identity

Measure	Training
Number of rows	134
Sum of Frequencies	134
-LogLikelihood	161.26066
BIC	449.86516
AICc	387.64282
Generalized RSquare	0.9408507

*Figure 4-2 shows the Effect Test and the Model Summary after the quadratic terms, mircuts*mircuts and fraccuts*fraccuts and the two-factor interaction of nodesel*fraccuts were removed from the model.*

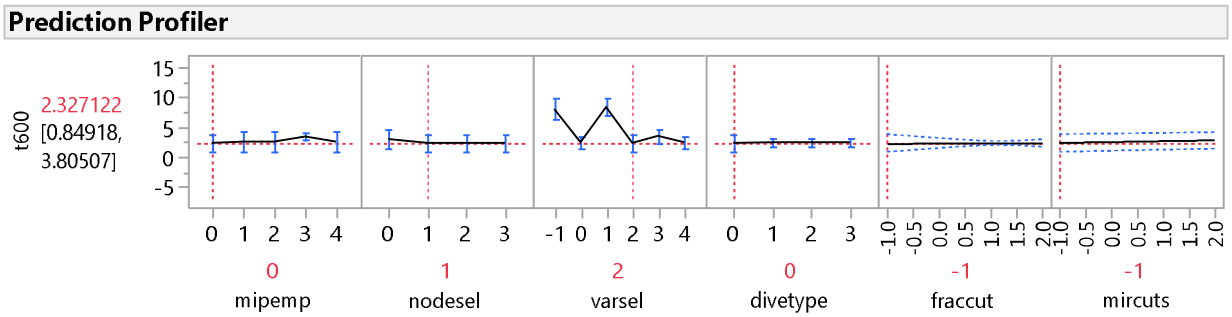


Figure 4-3 shows the prediction profiler. On the left the number in red is the estimated geometric mean of the primal integral at 10 minutes. The red numbers below the graphs are the recommended settings.

A similar process was completed for all classes of problems and the resulting parameter suggestions are found in Table 3.3. For all three classes, a first order model with two-way interactions was fitted. Refer to Table 3.1 for the meaning of the levels in Table 3.3.

Table 4-3 Recommend settings for each class of MIPs from our model.

Class	MIPEMP	NODESEL	VARSEL	DIVETYPE	FRACCUT	MIRCUTS
E	0	1	2	0	-1	-1
M	0	2	2	0	2	2
H	2	2	0	2	-1	0

These settings were used on the test instances from each class of problems and their performance compared to CPLEX's default settings and the other method's best run of their design. They are also compared to the best runs of three covering arrays created in JMP and four D-optimal designs. This can be seen in Tables 3.4 -3.6.

3.3 Results

Before looking at the results of tuning experiment we first gain some insights by looking at the histograms of the response, the geometric means of the primal integral for all of the classes. For

telecommunication network class, in Figure 3.4, the data has a mean of 7.53 and a standard deviation of 3.95.

Figure 4-4– Histogram of the response, the geometric mean of the primal integral, for telecommunication network class instances.

The data for the telecommunication network class appears to be bimodal and skewed somewhat to the right. This may indicate a difference in the instances such as the number of variables and constraints or structural difference caused by the different data used in the development of the instance.

Figure 3.5 contains the histogram of the response, the geometric mean of the primal integral, for the SVM class and the mean is 255.99 with a standard deviation of 163.62. Compared to the mean of the telecommunication network class, the mean for the SVM class is larger indicating that these instances are more difficult to solve on average.

Figure 4-5 shows histogram of the response, the geometric mean of the primal integral, of the SVM class.

Figure 3.6 shows the histogram of the response, the geometric mean of the primal integral, of the coding theory class and has a mean of 123.93 with a standard deviation of 10.61.

Figure 4-6 shows histogram of the response, the geometric mean of the primal integral, of the coding theory class.

Looking at the histograms for coding theory class in Figure 3.6, we see that the response data appears to be close to a normal distribution although skewed.

The results from our tuning experiment are contained in Tables 3.4 - 3.6, telecommunication class (E), SVM class (M), and the coding theory class (H) respectively. These results were produced using our testing instances and the recommended parameter settings from our method and the competing methods. When looking at the methods column of these tables you will see a number to the right of each method. This number indicates the number of runs that were conducted for each method. The pairwise method creates a covering array of

strength 2 and the number of runs is then determined by the number of two-way interactions being covered by the array. To cover all two-way interactions multiple times 60 runs were needed. The greedy method lets the user choose the number of runs. In this case 22 runs were chosen as a minimum to provide a design with a smaller number of runs than could be obtained with the pairwise method. Sixty runs were also chosen for the greedy method to compare to the pairwise method and 204 was chosen to compare to the d-optimal design. (A smaller run size is preferred when comparing two responses of equal value, because they are less time consuming to run.) Multiplying the number of runs by 10 for the ten minutes each run could take, if the solver does not find an optimal solution by the time limit, can give you an estimate of how long each method would take on a single core computer. As an additional comparison, results from the best run of designs from three covering arrays (CA) and four D-optimal designs, are also included. These additional designs were created in JMP. The very last method listed in the tables is CPLEX's automated tuner recommended settings. CPLEX's automated tuner was given a time limit of 600 seconds for each run. The operator can select the number of times the tuning is repeated, but the actual number of things it tries as it is tuning is up to CPLEX. One thing that can be seen in when looking at results from all three classes is that sampling generating a design and picking the best level is not a good strategy because you never know what you are going to get, a good or bad performing setting. The model framework, while not always giving the best setting, offers a more consistent approach to provide a reasonable setting.

Table 4-4 shows the results for telecommunication network class's instances for the limited experiment. The percent change of all designs created by various methods when compared to CPLEX's default setting is located in the second column. A negative percent change indicates a decrease in the geometric mean of the primal integral value and is more desirable than a positive change. The ranks within the test instance show that out of 7680 settings, the particular setting's rank. (Rank 1 is the best.) Our design and modeling results, Class E Rec134, are in bold.

Methods	The Geometric Mean of the Primal Integral at 10min. For the Recommended Setting	Percent Change from Default	Ranking
Class E - Telecommunications Network		**All Instances**	
Cplex default	1.996117	0.00%	157
Class E Rec 134	**1.96626**	**-1.50%**	**141**
pairwise32	2.231403	11.79%	523
pairwise60	2.11914	6.16%	415
greedy22	2.149108	7.66%	363
greedy60	2.313012	15.88%	428
greedy204	1.89476	-5.08%	125
greedy240	1.573784	-21.16%	9
ca_s2_opt30	2.150971	7.76%	518
ca_s3_not_opt173	1.934282	-3.10%	142
ca_s3_opt140	1.98667	-0.47%	218
dopt22	2.056831	3.04%	120
dopt60	1.468441	-26.44%	1
dopt204	1.881836	-5.73%	54
dopt240	1.890125	-5.31%	53
cplex_tune600sec	3.697482	85.23%	1142

For telecommunication network class, the dopt60 performed the best offering a 26.44 % improvement over the default settings. Our method, Class E Rec134, (the notation Rec134 indicates that the setting being used is the recommended setting obtained from the modeling approach using a D-optimal design with 134 runs) is in the top six best settings giving the user a 1.50% improvement over CPLEX's default settings. CPLEX's auto tuning did 85.23% worse than default settings. Figure 3.7 visualizes the amount of room for improvement over the default settings for Class E -Survivable Fixed Telecommunication Network Design.

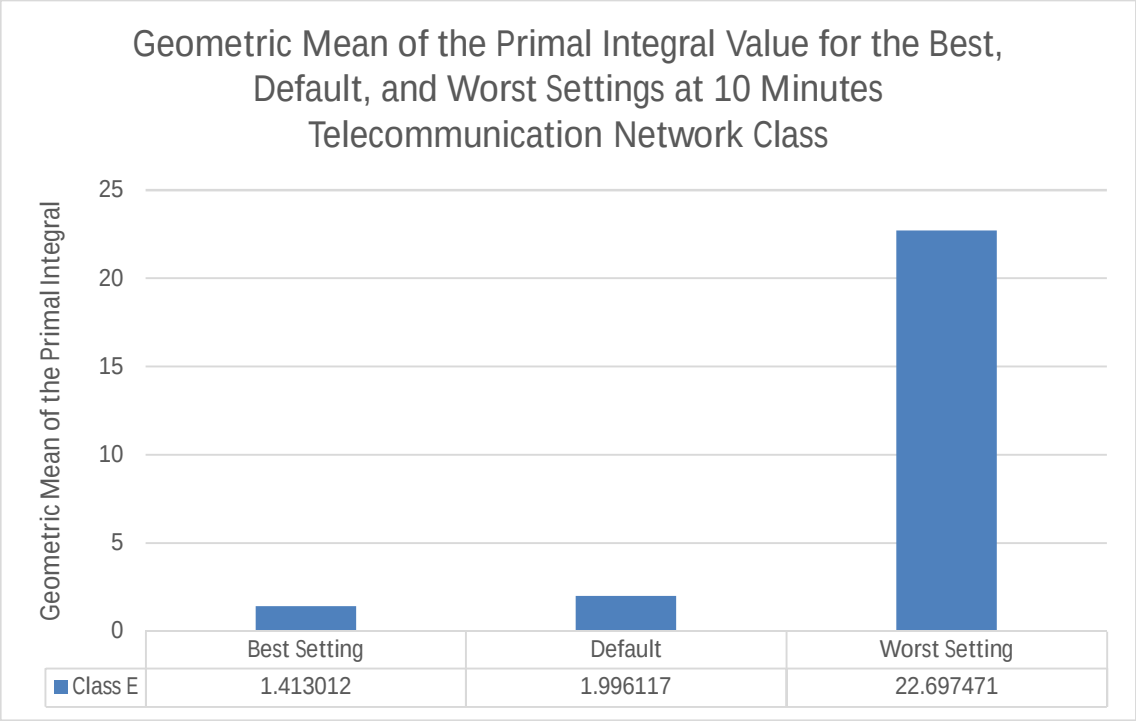


Figure 4-7 indicates how much room for improvement over the default settings for Class E - Survivable Fixed Telecommunication Network Design. The best setting is 29.21% better than the default setting and 93.92% better than the worst setting. Default setting is 91.21% better than the worst setting.

In Table 3.5, which contains the results for the SVM class, the Greedy204 performed the best 29.43% improvement over the default settings. Class M rec134 came in a close second with a

29.41% improvement over CPLEX's default settings. CPLEX's auto tuning did 0.40% worse than default.

Table 4-5 – Shows the results for SVM class's instances. The percent change of all designs created by various methods when compared to CPLEX's default setting is located in the second column. The ranks within the test instance show that out of 7680 settings, the particular setting's rank. (Rank 1 is the best.) Our design and modeling results are in bold.

Methods	The Geometric Mean of the Primal Integral at 10 min. for the Recommended Setting	Percent Change from Default	Ranking
Class M – SVM Instances	**Limited Experiment with All**		
Cplex default	43.327314	0.00%	265
Class M Rec 134	30.586346	-29.41%	10
pairwise32	35.811581	-17.35%	18
pairwise60	42.745704	-1.34%	221
greedy22	47.830702	10.39%	833
greedy60	41.633284	-3.91%	157
greedy204	30.575983	-29.43%	9
greedy240	38.146965	-11.96%	35
ca_s2_opt30	40.457175	-6.62%	101
ca_s3_not_opt173	39.978112	-7.73%	85
ca_s3_opt140	39.669085	-8.44%	76
dopt22	41.209246	-4.89%	134
dopt60	40.690786	-6.09%	107
dopt204	37.919413	-12.48%	31
dopt240	36.834725	-14.98%	23
cplex_tune600sec	43.501843	0.40%	283

Figure 3.8 provides an image to help the reader visualize the amount of room for improvement over the default settings for Class M - A formulation of the support vector machine with the ramp loss and L1-norm regularization.

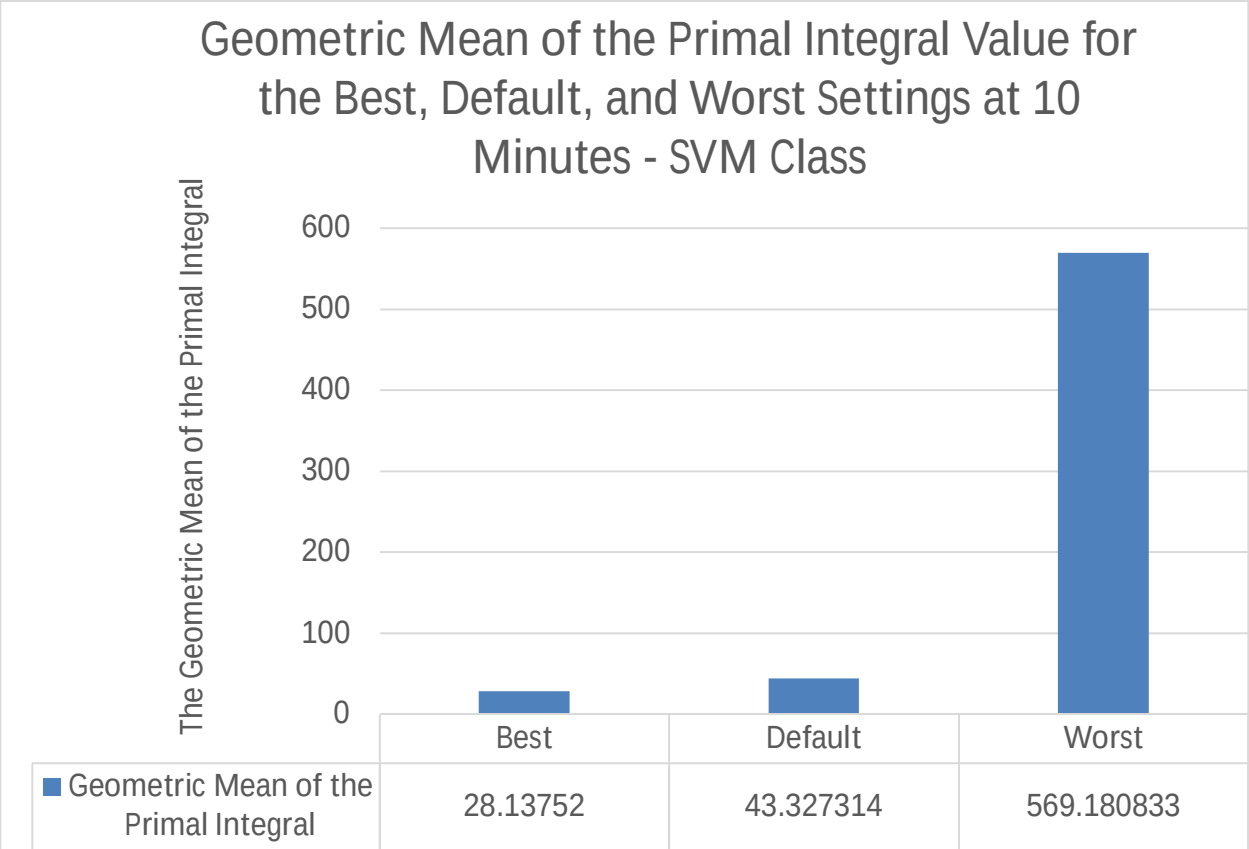


Figure 4-8 indicates how much room for improvement over the default settings for Class M - A formulation of the support vector machine with the ramp loss and L1-norm regularization. The best setting is 35.06% better than the default setting and 95.06% better than the worst setting. The default setting is 92.39% better than the worst setting. This class has the largest amount for improvement above default.

Figure 3.9 illustrates a portion of the interaction profiler for the SVM class. To identify important interactions, look for intersecting colored lines. From Figure 3.9, mircuts*varsel and

mircuts*nodesel are identified as important interactions. To verify that these interactions are significant look at Figure 3.10 for the effect tests results.

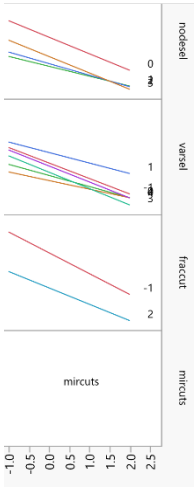


*Figure 4-9 shows a portion of the interaction profiler. To help identify important interactions, look for the colored lines that are intersecting. Here we see that mircuts*varsel and mircuts*nodesel may be significant.*

In Figure 3.10 the effect test table produced by JMP statistical software has five out of the six parameters as significant for the model. Diving strategy is the one parameter that is not significant and removed from the model. Nodesel*mircuts, varsel*mircuts, fraccut*mircut, varsel*fraccut, and nodesel*fraccut are the significant interactions. Three of the five significant interactions involve fraccut which tells us that the setting of Gomory cuts plays an important part of producing an efficient optimization solution process for the SVM class of MIPs.

Figure 4-10 is the effect test produce by JMP statistical software provides an easy way to identify significant main effects and two-way interactions by writing the p-value in red. This table contains the effect test for the SVM class.

Table 3.6 shows the results for coding theory class's instances. The percent change of all designs created by various methods when compared to CPLEX's default setting is located in the second column. The ranks within the test instance show that out of 7680 settings, the particular setting's rank with rank one being the best. For coding theory class, the greedy240 design had a run that performed the best offering a 25.36 % improvement over the default settings. Our method, Class H Rec134 offer 16.15 % improvement over the default settings. CPLEX's auto tuner returned the default setting so there was no improvement after using the tuner.

Table 4-6 has the results for the coding theory class. The modeling approach is in bold. The percent change of all designs created by various methods when compared to CPLEX's default setting is located in the second column. The ranks within the test instance show that out of 7680 settings, the particular setting's rank. (Rank 1 is the best.)

Methods	The Geometric Mean of the Primal Integral at 10 min. For the Recommended Setting	Percent Change from Default	Ranking
Class H- Coding Theory All Instances			
Cplex default	121.719375	0.00%	7259
Class H Rec 134	**102.066409**	**-16.15%**	**527**
pairwise32	101.183565	-16.87%	429
pairwise60	93.866078	-22.88%	22
greedy22	93.497773	-23.19%	13
greedy60	99.345166	-18.38%	170
greedy204	98.706841	-18.91%	68
greedy240	90.85334	-25.36%	8
ca_s2_opt30	97.359713	-20.01%	44
ca_s3_not_opt173	96.300482	-20.88%	52
ca_s3_opt140	95.777497	-21.31%	67
dopt22	99.574478	-18.19%	140
dopt60	98.858427	-18.78%	187
dopt204	93.537509	-23.15%	14
dopt240	92.991104	-23.60%	19
cplex_tune600sec	121.719375	0.00%	7259

Figure 3.11 provides an image to help the reader visualize the amount of room for improvement over the default settings for Class H – coding theory.

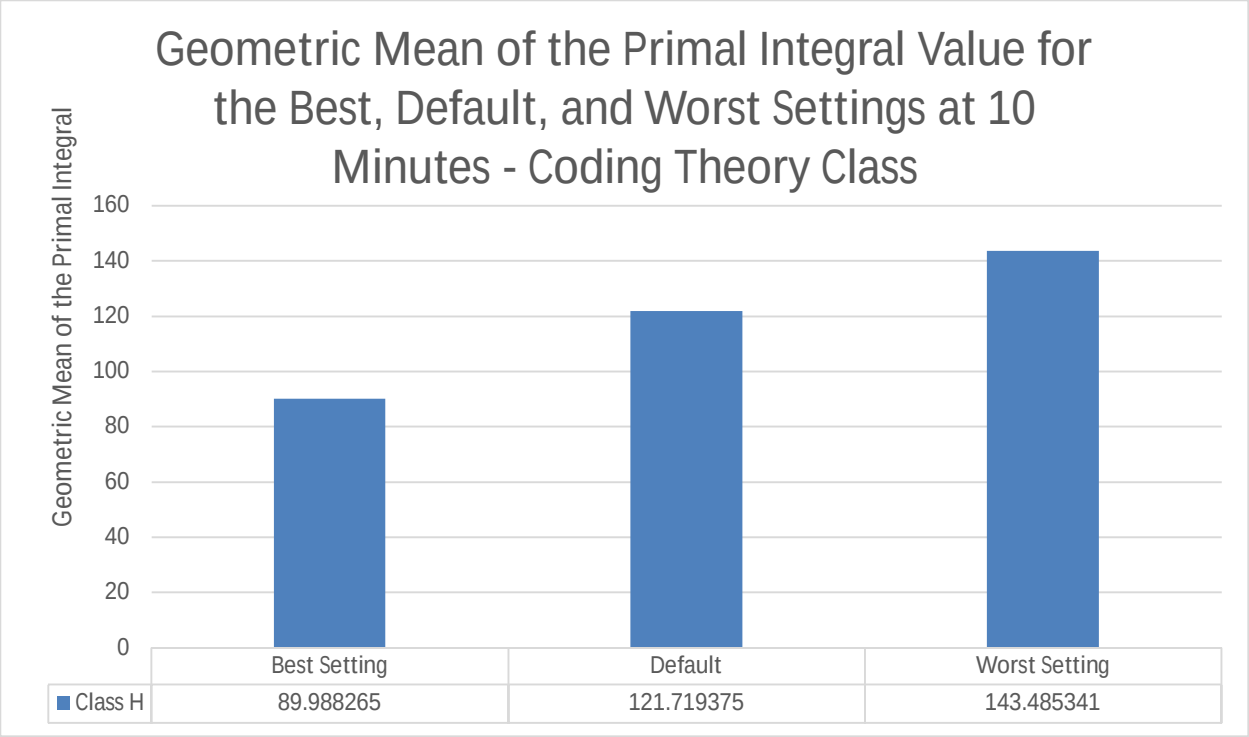
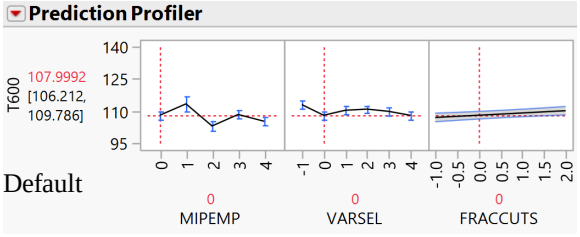


Figure 4-11 indicates how much room for improvement over the default settings for Class H – coding theory. The best setting is 24.72% better than the default setting and 36.55% better than the worst setting. The default setting is 15.71% better than the worst setting. Out of all three classes, the coding theory class has the least amount of room to improve over default settings.

Figure 3.12 depicts the prediction profiler for the coding theory class at default settings and recommended settings. The number on the left-hand side of both is the predicted geometric mean of the primal integral at the respective settings.



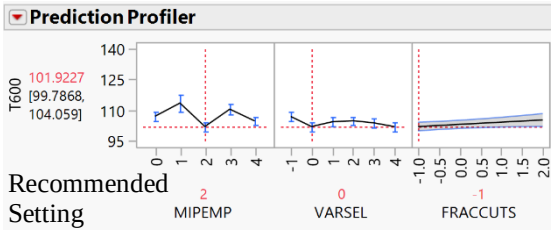


Figure 4-12 shows the prediction profiler for the coding theory class. On the left is the default settings and on the right is the recommended setting. The predicted value of the geometric mean of the primal integral of the recommended setting is 101.9227 which is smaller than the value for the default setting which is 107.9992.

In Figure 3.13 the significant main effects for the coding theory class are the *solving approach* and *branching*. The significant interaction is *solving approach*Gomory Cuts*. Mixed integer rounding cuts, node selection and the diving strategy were not significant and these factors were removed from the model. With this information, we learn that there is not a significant response when choosing a strategy to perform a probing dive or when a rule is chosen for selecting a node when backtracking. Since the variables in a node packing are integer, it is expected that the mixed integer rounding cuts would not be necessary.

Figure 4-13 is the effect test table produce by JMP statistical software for the coding theory class. It is easy to identify significant effects because the p-values are in red in this figure. Note that non-significant effects may still be in the model and this can be seen here with FRACCUTS (Gomory fractional cuts) which has a p-value of .3888.

Figure 3.14 illustrates the interaction profile of *solving approach*Gomory Cuts* on the left and on the right of Figure 3.14 the prediction profilers show the effect of changing *Gomory Cuts(FRACCUTS) parameter value from 2 to 3.* By looking at interaction profile in Figure 3.12 (left) and noticing where the blue and orange lines intersect, tells us that the strategy of generating Gomory fractional cuts aggressively (parameter value set to 2) while placing an emphasis on proving optimality by moving the best bound (parameter value set to 3), increases performance of the optimization software.

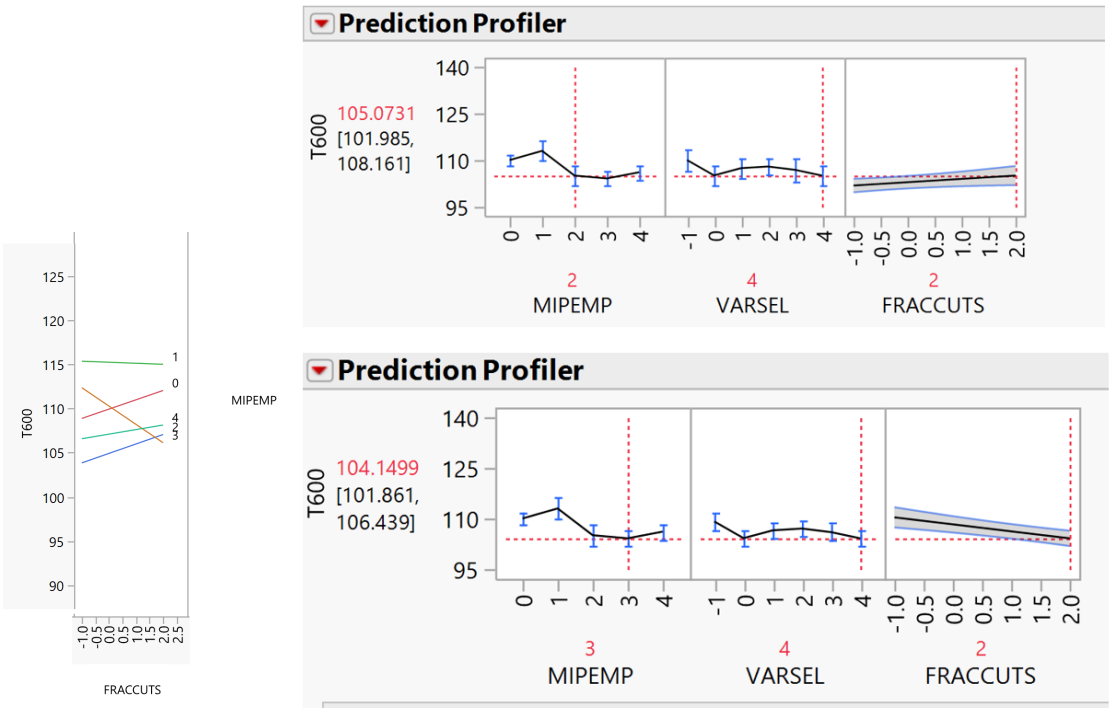


*Figure 4-14 shows the interaction profile (left) between solving approach*Gomory Cuts (MIPEMP*FRACCUTS) for the coding theory class. Potential interactions can be identified by looking for the different colored lines to intersect. On the right, the two prediction profiler images show that changing MIPEMP's parameter value from 2 to 3 reduces the predicted geometric mean of the primal integral from 105.0731 to 104.1499.*

Table 3.7 contains the best, default and worst parameter settings for the response data which was found exhaustively searching through all 7680 parameter settings for each class of MIPs. The column labeled *% Improvement* contains the percent change found when comparing the best setting to the default and worse setting. The data comes from using all of the instances to calculate the geometric mean of the primal integral.

Table 4-7 list best, default, and worst settings of the 6 parameters for all three classes. The performance of the settings in terms of the geometric mean of the primal integral at 10 minutes is given. The ranking of the default setting is given and a comparison of the best setting to the default and worst setting is in the last column. These results were obtained using CPLEX 12.6.1

Type of Setting from an Exhaustive Search	Run	MIPEMPHASIS	NODESEL	VARSEL	DIVETYPE	FRACCUTS	MIRCUTS	Geometric Mean of the Primal Integral at 10 Minutes (all instances)	Rank (out of 7680)	%Improvement of Best/Default or Worst
Class E – Telecommunication Network										
Best	2505	1	2	2	0	1	-1	1.413012	1	
Default	454	0	1	0	0	0	0	1.996117	135	29.21%
Worst	1598	1	0	-1	3	2	0	22.697471	7680	93.77%
Class M – SVM										
Best	452	0	1	0	0	-1	2	28.13752	1	
Default	454	0	1	0	0	0	0	43.327314	265	35.06%
Worst	1713	1	0	1	3	-1	-1	569.180833	7680	95.06%
Class H – Coding Theory										
Best	2795	1	3	0	2	1	1	89.988265	1	
Default	454	0	1	0	0	0	0	121.719375	7381	26.07%
Worst	6438	4	0	3	2	0	0	143.485341	7680	37.28%

In Table 3.8 the results of the limited experiment conducted using CPLEX 12.7.1. In all three classes, the methodology we used outperformed default settings when using the geometric mean of the primal integral metric, as seen in table 3.8 in bold. When using the geometric mean of the solution time as the metric our methodology does better than default for the telecommunication class and then ties with default for the SVM and Coding theory classes. The recommended setting for the telecommunication network, SVM, and coding theory classes were, 7.04%, 10.82%, and 3.56% better than default settings respectively. The best setting for the telecommunication network, SVM, and coding theory classes is 28.99%, 31.07%, and 18.45% better than the respective default settings.

It is interesting to note that the best setting, for all three classes has changed using this new version of CPLEX. The other settings, default, worst and the recommended setting obtained using the modeling framework also differed.

Table 4-8The best, recommended setting from model, default, and worst settings of the six parameters are listed. The geometric mean of the primal integral and solution time is given along with the percent change from default. These results were obtained using CPLEX 12.7.1.

Type of Setting from an Exhaustive Search	Run	MIPEMPHASIS	NODESEL	VARSEL	DIVETYPE	FRACCUTS	MIRCUTS	Geometric Mean of the Primal Integral	%Change from Default Geometric Mean of Primal Integral	Geometric Variance of the Primal Integral	Geometric Mean of Solution Time	%Change from Default GeoMean Solution Time
Class E- Telecommunication Network												
Best	761	0	1	4	3	1	-1	9.888146	-22.47%	499.535799	115.49871	-14.87%
EasyRec134	**2125**	**1**	**1**	**2**	**0**	**2**	**-1**	**11.85714**	**-7.04%**	**536.582147**	**112.370904**	**-17.17%**
Default	454	0	1	0	0	0	0	12.75477	0.00%	369.777225	135.665536	0.00%
Worst	1590	1	0	-1	3	0	0	50.26717	294.10%	295.439955	256.520378	89.08%
Class M - SVM												
Best	590	0	1	2	0	2	0	80.73233	-23.71%	1216.229437	245.602569	-6.95%
MedRec134	**608**	**0**	**1**	**2**	**1**	**2**	**2**	**94.37427**	**-10.82%**	**227.512422**	**263.934903**	**0.00%**
Default	454	0	1	0	0	0	0	105.8191	0.00%	918.877821	263.934903	0.00%
Worst	1617	1	0	0	1	-1	-1	573.8456	442.29%	1.051285	610	131.12%
Class H – Coding Theory												
Best	767	0	1	4	3	2	1	89.58883	-15.58%	6.642505	610	0.00%
HardRec134	**6737**	**4**	**1**	**2**	**1**	**-1**	**-1**	**102.3389**	**-3.56%**	**5.478799**	**610**	**0.00%**
Default	454	0	1	0	0	0	0	106.1179	0.00%	5.457973	610	0.00%
Worst	288	0	0	3	1	2	2	151.7406	42.99%	5.938418	610	0.00%

To give the reader a broader look at CPLEX's automated tuner, the results provided are based on the training data for each class of MIPs in Table 3.8. When reading the table, find the tuning time limit set by the user for each run under the column headed group tuning. This was not the overall time the tuner took to do the tuning. Instead, the tuner chooses the number of runs and how much time to give each run. Often CPLEX's tuner would use 10% of the time limit given for each run, but the auto tuner determined the number of runs. This is why we also give the actual time it took for the tuner to produce its recommendation. The tuning time is reported with three different units for the ease of interpretation by the reader.

Table 4-9 results of CPLEX's automated tuner. The first column is the user chosen tuning time limit in seconds.

Group Tuning	Tuning Time (seconds)	Tuning Time (minutes)	Tuning Time (hours)	Default Settings Chosen	MIPEMPHASIS	NODESEL	VARSEL	DIVETYPE	FRACCUTS	MIRCUTS
Class E										
45	364	6.07	0.10	YES	0	1	0	0	0	0
300	11298	188.30	3.14	YES	0	1	0	0	0	0
600	21085	351.42	5.86	NO	0	1	4	0	0	0
900	29230	487.17	8.12	NO	0	1	0	0	0	1
1200	36829	613.82	10.23	NO	0	1	0	0	0	1
1800	52574	876.23	14.60	YES	0	1	0	0	0	0
2400	67356	1122.60	18.71	YES	0	1	0	0	0	1
4800	130184	2169.73	36.16	NO	0	1	0	0	0	0
Class M										
45	2445	40.75	0.68	NO	0	1	0	0	0	1
300	14461	241.02	4.02	NO	0	1	0	0	0	1
600	27726	462.10	7.70	NO	0	1	0	0	0	1
900	40473	674.55	11.24	NO	0	1	0	0	0	1
1200	53250	887.50	14.79	NO	0	1	0	0	0	1
1800	80212	1336.87	22.28	NO	0	1	0	0	0	1
2400	106293	1771.55	29.53	NO	0	1	0	0	0	1
4800	204992	3416.53	56.94	NO	0	1	0	0	0	1
Class H										
45	1236	20.60	0.34	NO	0	1	4	0	0	0
300	15451	257.52	4.29	NO	0	1	4	0	0	0
600	30841	514.02	8.57	YES	0	1	0	0	0	0
900	47859	797.65	13.29	NO	0	1	0	0	0	1
1200	63790	1063.17	17.72	NO	0	1	0	0	2	0
1800	98883	1648.05	27.47	NO	0	1	0	0	2	2
2400	131820	2197.00	36.62	NO	0	1	0	0	2	2
4800	263577	4392.95	73.22	NO	0	1	0	0	2	2

3.4 Conclusions for the Limited Experiment

Using a DOE approach with a modeling framework by creating D-optimal designs to tune the parameter settings offers an improvement over CPLEX's default and autotuned settings. Although this approach does not always give the best recommended setting it competes well against other design's best run. The one thing we learn with the modeling framework that choosing the best run of a design does not offer is the ability to discern important parameters for a class of MIPs.

In the case of telecommunication network class, we see that the *divetype* and *fraccuts* factors are not significant and not included in the model for telecommunication network class. This can provide added information about our MIP. For example, since the *fraccuts* factor is not

included in the model, we know that the production of Gomory cuts don't play an important part in the overall performance of the optimizer.

In the case of the SVM class, we found that the only main effect that was not significant was the diving strategy. While not only was Gomory cuts a significant parameter but it also played a part in three out of the five significant interactions.

The coding theory class had two significant main effects which are *solving approach* and *branching*. The coding theory class had only one significant interaction which is *solving approach*Gomory Cuts*. It is important to note this method was able to correctly identify that mixed integer rounding cuts were not significant in the node packing problem.

The best setting for the telecommunications network, SVM, and coding theory classes are 29.21%, 35.06% and 26.07% better than default setting respectively. The potential performance improvement for just tuning six parameters is substantial. Exploration with a larger set of parameters could provide further insights into a class of MIPs and the performance of the optimizer.

Chapter 5 Extended Experiment with Screening

4.1 Background

Variable screening is an essential step in selecting important variables that have the most impact on the response. The parameter estimates of the regression model can be positive or negative, but either way, once identified a model can be fit that will enhance the desirable effects and mitigate the undesirable ones. The fundamental principles, effect hierarchy, effect sparsity, and effect heredity help one navigate the screening process. In this section, we consider tuning 59 parameters that are mixed level categorical and ordinal, and continuous. To try and keep the budget for computer runs low the effect hierarchy principle tells the researcher to design experiments that focus on lower order effects because they will have the greatest chance of being important. This leads to consider screening with just the main effects at first, and then spending more of the budget on second screening where a design can focus on the parameters, and their interactions, that seem to be important to the performance of the optimization solver. Effect sparsity principle provides the researcher the knowledge to expect that there is a small number of important effects and therefore the list of parameters to be tuned for a class of instance will be limited.

Often there are numerous models that can be created to model the response of the experiment but using effect heredity in the model selection process will help to limit the number of models to choose from because, if followed, effect heredity ensures that at least one of the main effects of an interaction must be present in the model.

The designed experiment and modeling framework used to tune 59 parameters differs only by the addition of the first step (steps listed below) when compared to the steps used with the limited factor experiment.

- Screen for significant factors in order to reduce the number of factors to be tuned
- Generate a design for the reduced list of parameters
- Run the computer experiment
- Fit a model using response from computer experiment
- Interpretation of the model
- Recommend a setting

In this extended factor experiment, there are 16 discrete/continuous factors and 43 categorical factors. Considering the discrete/continuous factors and all categorical factor's levels, this experiment has 26 two-level, 10 three-level, 16 four-level, 5 five-level, 1 six-level, and 1 seven-level factors. This gives us $2^{26} \cdot 3^{10} \cdot 4^{16} \cdot 5^{5} \cdot 6 \cdot 7 \approx 2.23381657 \times 10^{27}$ possible settings to test. Even setting a time limit of ten minutes for each of the 34 instances to be solved, it is impossible to test the full factorial of settings in a reasonable amount of time because the experiment would take $7.526168539 \times 10^{21}$ years using the same computing power that we used to conduct the experiments in this paper.

When conducting the extended experiment, a change from the methodology used in the limited number of factors experiment was needed. This was necessary because with the limited experiment it was possible to create D-optimal designs for a first-degree linear model with

interactions that only required 134 design points (about 3.5 hours of computer time to conduct the experiment per class of instances); but with the 59 factors being used, a D-optimal design for a first-degree linear model with interactions would minimally require 7888 design points which equates to about 26.6 days for 70 cores to process this work so that the three different classes of MIPs could be tuned (about 9 days of computer time to conduct the experiment per class of instances). An alternative method that takes less time would be beneficial in practice.

4.2 Screening by Grouping

First, consider a screening with grouping suggested in Mee, (2009). Screening with grouping offers a way to consolidate two-level factors into groups in which a similar effect on the response is expected. Once grouped, a screening design like a Plackett-Burman design or a fractional factorial design can be used to screen for significant groups of factors thus reducing the overall number of factors to be considered. The benefit of this method is that it dramatically reduces the number of computer runs needed to tune a set of parameters of the optimization software. Unfortunately, the majority of the factors in the experiment are categorical with more than 2-levels, which implied that less than half of the 59 factors could be used. In order to increase this number, we can consider, categorical variables that are ordinal in nature may be treated as continuous if the order suggests a continuum (Agresti, 2010; Rhemtulla et al., 2012). For example, a parameter that creates cuts may have levels like the following:

1. create no cuts
2. let the optimizer choose how to set cuts (default)
3. create a minimal number of cuts

4. create a moderate number of cuts
5. create a large number of cuts
6. create an aggressively large number of cuts

By removing the default value as a choice, what remains are levels which imply that as the parameter value increases so does the number of cuts being created by the solver. This is the type of situation in which one could consider this factor to be continuous. Interpretation of the recommended setting of variables that are being considered continuous, must be done after rounding any of the recommended values that are not integer due to the fact that all of parameters being used are discrete variables. Therefore, 39 of the 59 factors that were either discrete, continuous, ordinal that could be considered continuous (minus the default value), or 2-level categorical, were used and the remaining 20 factors were discarded. In order to screen the 39 factors via grouping, the following steps were followed:

1. Create 12 groups that contain factors that are compatible. This means that all factors in a group are expected to have like sign effects so that the effects can sum to a number and therefore do not cancel each other. All factors in a group will simultaneously be set to either a high or low value as determined by the design. Table 4.1 contains a list of all of the factors in each of the twelve groups created. Create a two-level screening design, preferably with resolution IV or higher, that was a Plackett-Burman design with 20 design points. A Plackett-Burman design is mostly used when N, the number of runs, is a multiple of 4 but not a power of 2. (A fractional factorial design could also be used, but in this case the Plackett-Burman design had less runs that a corresponding fractional factorial design with the same resolution.) With this design, main effects that are not aliased with each other or any two-factor interaction and this makes it easier to identify

significant main effects. When creating the design, treat each group as if it were only one factor, so in this case where there were 12 groups, then the design had 12 factors.

Table 5-1contains the 12 groups used for the screening by grouping technique.

X1-Cuts1 1. cutsfactor 2. eachcutlimit 3. cutpass 4. prelim 5. reinv 6. singlim 7. strongcandlim 8. strongitlim	**X2 – Cliques** 1. cliques	**X3 – Disjcut** 1. disjcut	**X4-Gomory** 1.fraccut 2.fraccand 3.fracpass	**X5- Covers** 1. cover 2. flowpaths 3. mircuts 4. flowcovers 5. aggcutlim	**X6-Aggregate** 1. aggfill 2. aggind
X7- Bounds 1. bndstrenind	**X8 – Preprocessing** 1. predual 2. prelinear 3. prepass 4. relaxpreind 5. symmetry 6. preind	**X9-Nodes/ branch&bound** 1. bbinterval 2. brdir 3. lbheur 4. mipsearch 5. rinsheur	**X10-Boundcuts** 1.gubcuts 2.implbd 3.zerohalfcuts	**X11-Heuristic** 1. fpheur 2. heurfred	**X12-Probing** 1. Probe 2. Probetime

2. Assign the same high or low value to all parameters in the group. In the case of the low-level value, it was unclear as to whether to set the value to a non-operational level or to a minimum operational level. To determine which would provide the best results two different screenings by grouping were conducted.

   - Grouping 1 -Low level for ordinal was set to minimum operational
   - Grouping 2- Low level for ordinal was set to non-operational

   As seen in Figure 4.1 a, b, c which contains the empirical cumulative distribution function (CDF) for each class, grouping two (blue line) tended to perform better than grouping one because it has a higher probability of attaining a lower geometric mean of the primal integral. Looking at Figure 4.1c, the coding theory class, there are several times when grouping one is slightly better; but overall, setting the low level to non-

operational value, as done in grouping two, has the best chance of a lower primal integral value.

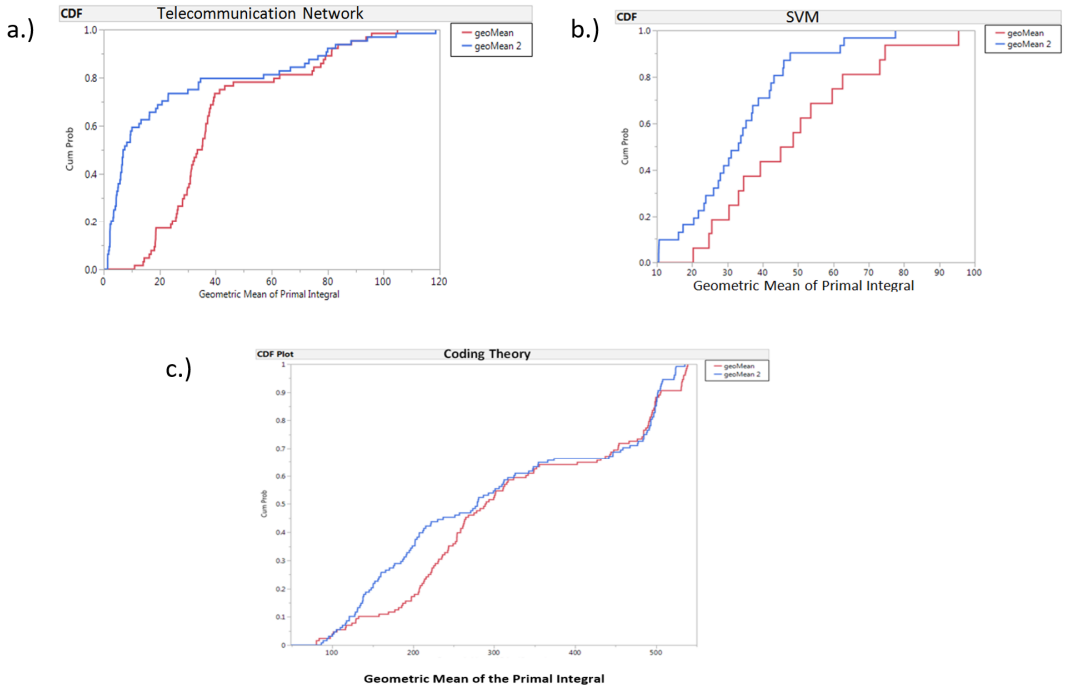
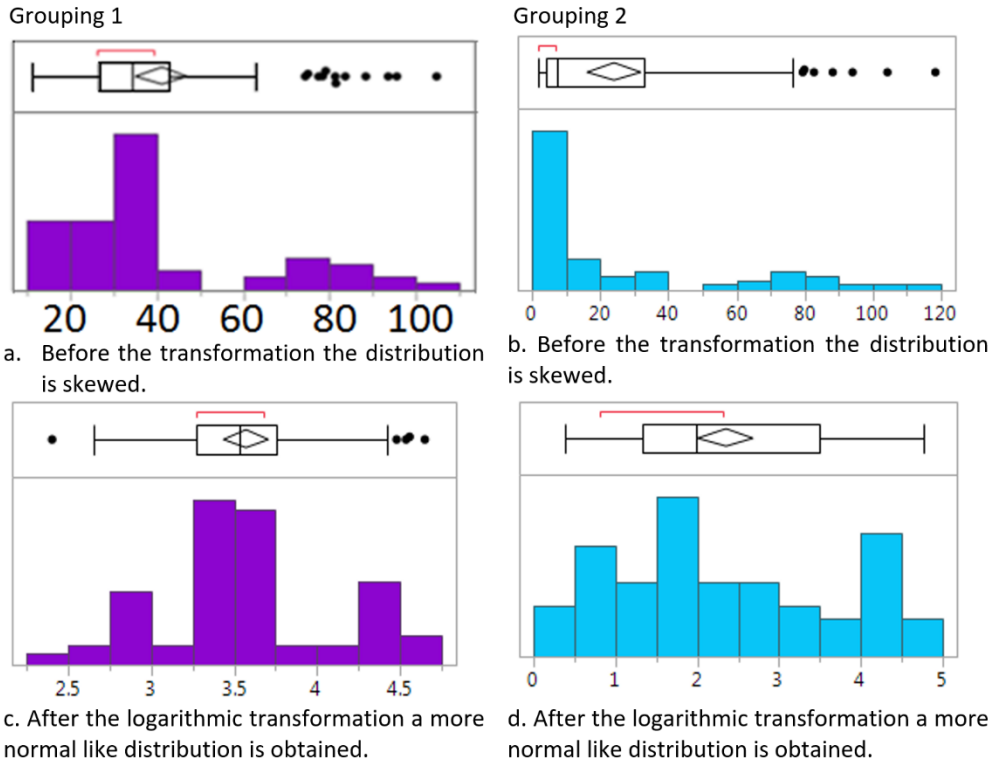


Figure 5-1 a, b, and c contain the empirical CDFs of the response variable which is the geometric mean of the primal integral. Grouping two in blue has more area under its curve, which indicates a higher probability of obtaining a low value of the response variable, than grouping one.

3. Run the screening experiments on the computer for grouping one and two. Then calculate the geometric mean and variance of the response, which is the primal integral value, for each design point across each instance in the class of MIPs.

4. As a precursor to variable selection check to see if the distributions are normal. If not, a transformation may be possible that would help the variable selection process. In the case of the Telecommunication Network class, Figure 4.2a and 4.2b display the skewed distribution of the data for grouping one and grouping two respectively.

5. Figure 4.2c and 4.2d illustrate grouping one and two, respectively, after a logarithmic transformation of the data was performed. Therefore, the response used for the variable selection process was the log(y) where y is the geometric mean of the primal integral.



a. Before the transformation the distribution is skewed.

b. Before the transformation the distribution is skewed.

c. After the logarithmic transformation a more normal like distribution is obtained.

d. After the logarithmic transformation a more normal like distribution is obtained.

Figure5-2 illustrates how the skewed distribution can be transformed into a more normal looking distribution by applying a logarithmic transformation. Grouping one is in purple and grouping two is in light blue.

6. Perform variable selection to screen for significant groups of factors. Generalized linear models with forward selection using the AICc criterion. Table 4.2 identifies the groups that are kept and the total number of the 39 factors that remain after the initial screening.

Table 5-2 illustrates the remaining groups of factors after the first screening takes place.

Class of Instances	Groups Kept After Screening (Grouping 1)	Number of Remaining Factors (Grouping 1)	Groups Kept After Screening (Grouping 2)	Number of Remaining Factors (Grouping 2)
Class E - Telecommunication Network	X8, X11	8	X8, X11	8
Class M – SVM	X4, X3	4	X11, X10, X7	6
Class H – Coding Theory	X1, X8, X4	17	X1, X8, X9, X2	20

7. With the remaining factors a two-level design (full-factorial, fractional factorial, or Plackett-Burman) was created to tune the significant parameters for each class and grouping. All factors that were removed from tuning were set to their default values.
8. To determine suggested settings from the first order linear model with interactions, a desirability function for each response (geometric mean and geometric variance of the primal integral) was created in JMP 12.0 statistical software. In each desirability function the objective was to minimize the response. Also, an importance weight was placed on the geometric mean and the geometric variance. By changing the ratio of the importance weight, it is possible to fine tune the recommended setting for the parameters.
9. If necessary sequential screening may be used if the number of factors is still too large. To make this determination, consider the results of the recommended settings given by the model created. If the results are not better than default settings and the number of remaining factors after screening was greater than ten, conducted a second screening. This was necessary for the Coding Theory class. However, if the number of factors is

sufficiently reduced, you can dispense with the two-level requirement and proceed, as in the limited case, with constructing a design with a variety of categorical factors at multiple levels.

In order to utilize all 59 factors, the 20 categorical factors that were not previously used in the group screening process were used in a second extended experiment. This was done by limiting the number of levels to two. In Table 4.3, the two levels utilized for the categorical variables is listed. In this new experiment CPLEX12.7.1 was used along with the group screening process described previously.

Table 5-3 List the categorical variables and their corresponding values assigned to the two levels. These were used for the extended grouping experiment using all 59 factors. CPLEX 12.7.1 was used.

Nominal Variable	Two Levels Used for Extended Grouping Experiment with 59 Factors
Mipemp	1 feasibility over optimality 2 optimality over feasibility
Nodesel	1 Depth-first search 2 Best-estimate search
Varsel	1 Branch on variable with maximum infeasibility 3 Strong branching
Divetype	1 Traditional dive 2 Probing dive
Craind	LP Primal 0 ignore coefficient during crash 1 Alternate ways of using obj coefficient LP Dual 0 aggressive starting basis 1 default starting basis
dpriind	1 standard dual pricing 5 devex pricing (if many col and few rows devex pricing may not help much)
ppriind	-1 reduced-cost pricing 2 steepest edge
siftalg	1 Primal Simplex 4 barrier
scaind	-1 No scaling 1 more aggressive scaling
Subalg	1 primal 4 barrier
Parallelmode	-1 opportunistic 1 deterministic
Startalg	1 primal 4 Barrier
Coeredind	0 turns off coefficient reduction in preprocessing 3 most aggressive coefficient reduction called Tilting
Depind	0 off, do not use dependency checker in preprocessing 3 turn on at the beginning and the end of preprocessing
Mipcbredlp	0 off 1 on
Preslvnd	1 no node presolve 3 aggressive node probing
Reduce	0 No primal or dual reductions 3 Both primal and dual reductions
Repeatpresolve	1 turn off re-presolve 3 re-presolve with cuts and new root node
Perind(int)	0 off -simplex perturbation switch 1 on
threads	1 one thread 4 four threads

With the additional factors, it became necessary to add additional groups. For the 59 factors we created 20 groups as seen in Table 4.4.

Table 5-4 contains a listing of the 20 groups used for the group screening for the extended experiment with 59 factors.

Twenty Groups Created for Group Screening – Extended Experiment with 59 Factors				
X1-Cuts1 1. cutsfactor 2. eachcutlimit 3. cutpass 4. prelim 5. reinv 6. singlim 7. strongcandlim 8.strongitlim	**X2 – Cliques** 1.cliques	**X3 – Disjcut** 1. disjcut	**X4-Gomory** 1. fraccut 2. fraccand 3. fracpass	**X5- Covers** 1. cover 2. flowpaths 3. mircuts 4. flowcovers 5. aggcutlim
X6-Aggregate 1.aggfill 2.aggind	**X7- Bounds** 1. bndstrenind	**X8 – Preprocessing** 1. predual 2. prelinear 3. prepass 4. relaxpreind 5. symmetry 6. preind	**X9-Nodes/ branch&bound** 1. bbinterval 2. brdir 3. lbheur 4. mipsearch 5. rinsheur	**X10-Boundcuts** 1. gubcuts 2. implbd 3. zerohalfcuts
X11- Heuristic 1. fpheur 2. heurfred	**X12- Probing** 1.probe 2.probetime	**X13- Parallel** 1. parallelmode 2. threads	**X14 - Coefficients/Scale** 1. coeredind 2. depend 3. reduce 4. perind 5. craind 6. scaind	**X15 Simplex** 1. siftalg 2. subalg 3. startalg
X16- Pricing 1.dpriind 2.ppriind	**X17 – Represolve** 1.repeatpresolve	**X18- Mipcbredlp** 1.mipcbredlp	**X19 -Solving Strategy** 1.mipemphasis	**X20 – Node** 1.divetype 2.nodesel 3.varsel 4.preslvnd

After the groups were created, group screening was conducted for all three classes. Sequential group screening was necessary for the telecommunication network class due to the large number of factors remaining after the initial screening. The groups that remain for the extended

experiment with group screening utilizing all 59 factors are in Table 4.5 along with the number of remaining factors.

Table 5-5 illustrates the remaining groups of factors after the group screening takes place. Note that sequential group screening was conducted for the telecommunications class. These results are for the extended experiment using group screening with all 59 factors.

Class of Instances	Groups Kept After Screening	Number of Remaining Factors
Class E - Telecommunication Network	X1, X4, X8, X13, X14, X15, X16, X20 (After initial group screening)	34
	X14, X15, X16 (After second group screening)	11
Class M – SVM	X2, X4, X6, X13, X18	9
Class H – Coding Theory	X1, X5, X10, X18, X19	18

4.3 Results Utilizing the Group Screening

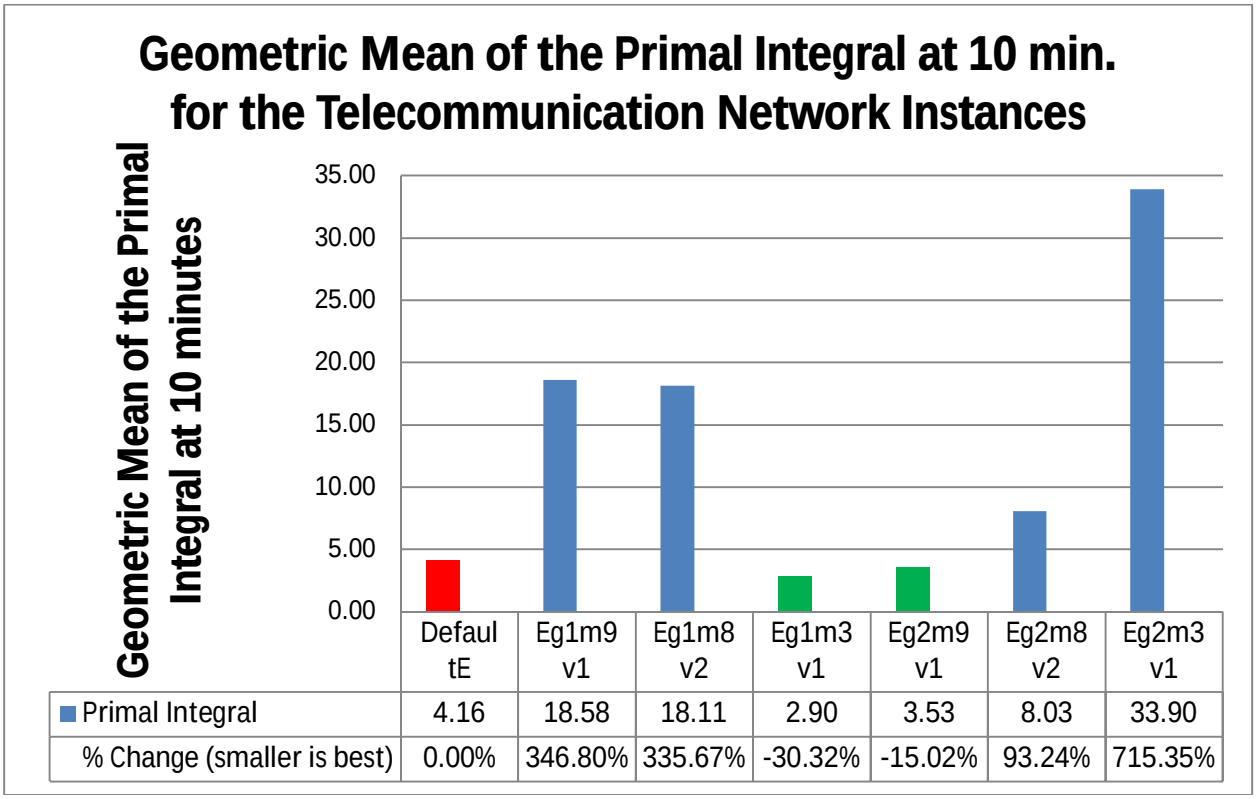
In the Telecommunication Network class, the significant main effects are: Predual, Preind, Prelinear, and Prepass. The significant interactions are: fpheur*Symmetry, Heurfreq*Relaxpreind, Prelinear*Prepass, and Prepass*Relaxpreind. In Figure 4.3 the recommended setting of parameters is given in terms of high (+1) or low (-1) value; along with this the value of the mean and variance can be found in red on the left of the figure. The

desirability number 0.647116 is also found to the bottom left of Figure 4.3, and the closer this number is to one, the better job of optimizing the desirability function. The actual corresponding Cplex value is then used in any computer experimentation. (The coding and recoding of all parameter values was done via a python script.)

Figure 5-3 The performance profiler in JMP statistical software provides a way to view multiple responses when changing the values of the parameters from high (1) to low (-1). The desirability functions are show at the far-right side of the graphs. By changing the parameters value, one can see the predicted effect it will have on the mean and variance which are the response variables.

In Figure 4.4 the geometric mean of the primal integral at 10 minutes for the telecommunication network class is given for the default values and for the recommended settings given with the different groupings and ratios of the importance weights. The name of each recommended setting gives the grouping and ratio. For example, from the name of the recommended setting Eg1m9v1, (E) represents the Telecommunication Network class using grouping 1(g1) and the mean to variance importance ratio is 9: 1 (m9v1). If the name was Mg2m9v2 this would be interpreted the SVM class (M) using grouping 2 (g2) and the mean to variance importance ratio is 9:2 (m9v2). If the first letter of the name was H then this would indicate that the class was

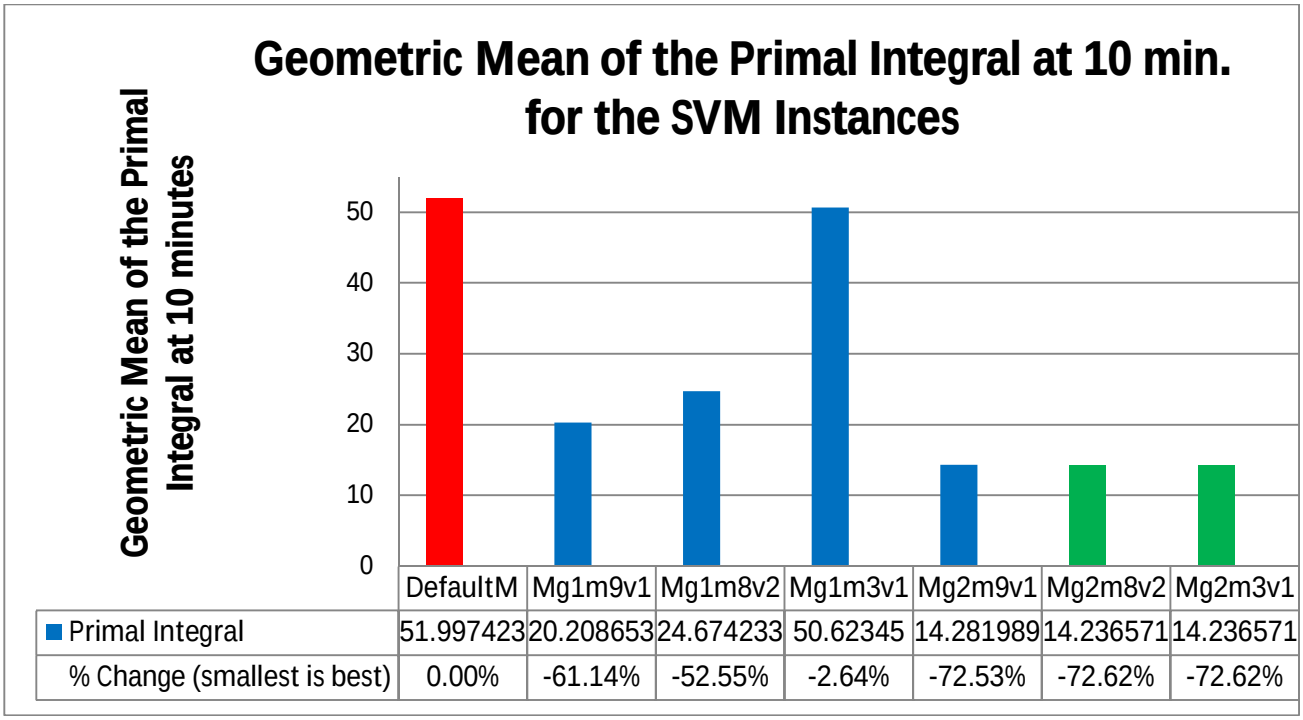
Coding Theory. In Figure 4.4, the green bars indicate that the recommended setting improves upon the default setting which is in red. The blue bars indicate that the performance of the model is poorer when compared to the default setting. Figure 4.4 indicates that with grouping 1 and 3:1 mean to variance importance ratio the recommended setting (Eg1m3v1) produces a 30.32% improvement over the default settings; and with grouping 2 and 9:1 mean to variance importance ratio the recommended setting (Eg2m9v1) produces a 15.02% improvement over the default settings. In this case Eg1m3v1 performs best and is selected as the recommended setting for the telecommunication network class.



	Default	Eg1m9v1	Eg1m8v2	Eg1m3v1	Eg2m9v1	Eg2m8v2	Eg2m3v1
Primal Integral	4.16	18.58	18.11	2.90	3.53	8.03	33.90
% Change (smaller is best)	0.00%	346.80%	335.67%	-30.32%	-15.02%	93.24%	715.35%

Figure 5-4 compares the performance of CPLEX's default settings to the recommended settings when varying grouping and mean to variance importance ratio for the Telecommunication Network class.

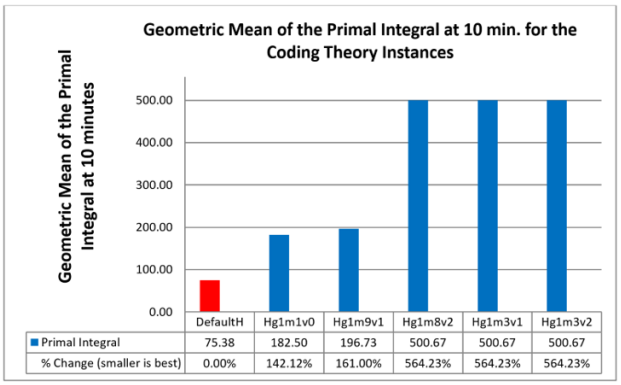
In Figure 4.5 all of the recommended settings outperform the default setting for the SVM class. The green bars indicate the settings (Mg2m8v2 or Mg2m3v1), grouping 2 with either a 8:2 or 3:1 mean to variance importance ratio, that are tied for best with a 72.62% improvement over default.



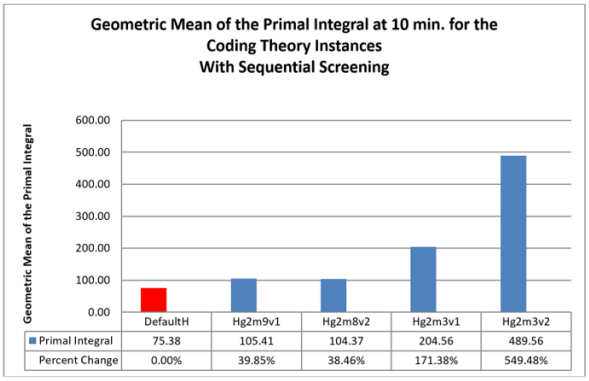
	DefaultM	Mg1m9v1	Mg1m8v2	Mg1m3v1	Mg2m9v1	Mg2m8v2	Mg2m3v1
Primal Integral	51.997423	20.208653	24.674233	50.62345	14.281989	14.236571	14.236571
% Change (smallest is best)	0.00%	-61.14%	-52.55%	-2.64%	-72.53%	-72.62%	-72.62%

Figure 5-5 compares the performance of CPLEX's default settings to the recommended settings when varying grouping and mean to variance importance ratio for the SVM class.

Figure 4.6a shows that the initial screening results are far from default and illustrates that with just a single screening, the recommended parameter settings (blue) perform worse than the default settings(red). Therefore, sequential screening was conducted and the results, in Figure 4.6b, showed that the recommended parameter settings (blue) were closer to outperforming the default setting(red). The recommended settings with an extra sequential screening are better than not having it.



a.



b.

Figure 5-6 a,b – Figure 5.6a illustrates results with just two screenings and Figure 5.6b illustrates that by adding an additional sequential screening the recommended settings are more competitive with the default settings.

In Table 4.5 the best result for each class and grouping are shown. The screening by grouping and then modeling with the forward selection method, improved the geometric mean of the primal integral for two out of the three classes. However, another benefit gained from this experiment is the knowledge of the parameters that play a significant role in the performance of the optimizer on the class of instances being tuned. Listed below in Table 4.6 are the significant main effects and two-factor interactions for the three different classes. For a listing of all of the variables included in the final model and the corresponding parameter estimates, see appendix B.

Table 5-6 summary results when utilizing screening by grouping technique.

Telecommunication Network				
Screening Type	**#Screening Design Points**	**Estimated Tuning Time in Hours per Class**	**Geometric Mean of Primal Integral**	**Percent Change from Default Setting**
Default			4.16	
Grouping 1 – Forward Selection	20/64 total 84	2.4	**2.90**	-30.32
Grouping 2 – Forward Selection	20/64 total 84	2.4	3.53	-15.02
SVM				
Default			52.00	
Grouping 1 – Forward Selection	20/16 total 36	1.03	20.21	-61.14
Grouping 2 – Forward Selection	20/32 total 52	1.49	**14.24**	-72.62
Coding Theory				
Default			75.38	
Grouping 1 – Forward Selection	20/128 total 148	4.23	182.50	142.12
Grouping 2 – Forward Selection	20/128 total 148	4.23	500.67	564.23
Gr2 – Sequential Screening	20/128/24/128 total 300	8.57	104.37	38.46

Table 5-7 List the significant parameters and interactions for the three classes of instances.

Significant Parameter or Interaction of Parameters – Telecommunication Networks Class	
Grouping 1	Grouping 2
Fpheur	Prelinear
Heurfreq	Prepass
Predual	Relaxpreind
Preind	Prelinear*Prepass
Prelinear	Prepass*Relaxpreind
Prepass	
Fpheur*Heurfreq	
Fpheur*Prelinear	
Fpheur*Symmetry	
Heurfreq*Predual	
Predual*Symmetry	
Preind*Prelinear	
Preind*Prepass	
Prelinear*Prepass	
Prelinear*Relaxpreind	
Prepass*Relaxpreind	
Significant Parameter or Interaction of Parameters – Coding Theory Class	
Cutsfactor	Cliques
Prelim	Cutpass
Reinv	Cutsfactor
Strongitlim	Prelim
Cutsfactor*Prelim	Reinv
Cutsfactor*Reinv	Singlim
Cutsfactor*Strongitlim	Rinsheur
Fraccand*Fracpass	Preind
Fraccand*Strongitlim	Prelinear
Reinv*Strongitlim	Symmetry
Eachcutlim*Singlim (v)	Cliques*Cutsfactor
Fraccand*Reinv (v)	Cliques*Preind
Fracpass*Singlim (v)	Cliques*Prelinear
Prelim*Reinv (v)	Cutpass*Cutsfactor
Prelim*Singlim(v)	Cutsfactor*Reinv
	Cutsfactor*Preind
	Prelim*Singlim
	Reinv*Singlim
	Reinv*Preind
	Singlim*Preind
	Preind*Symmetry
	Mipsearch (v)
	Cliques*Predual (v)

	Singlim*Rinsheur (v)
	Singlim*Prelinear (v)
	Mipsearch*Preind (v)
Significant Parameter or Interaction of Parameters – SVM Class	
Fraccand	Zerohalfcuts*Bndstreng (v)
Fracpass	Implbd (v)
Disjcuts*Fraccand	Zerohalfcuts (v)

In Table 4.7 the parameter estimates for the SVM class model obtained using screening with grouping one can be found along with three different settings to illustrate the effect the parameter values have on the predictive model. In this case we are trying to minimize the geometric mean of the primal integral. At first one might think to choose the lowest value of each parameter to obtain the smallest metric value, but as seen with the results of setting one in Table 4.7 this does not necessarily produce the smallest setting because the interaction Disjcuts*Fraccand having a negative coefficient. Setting two and three have smaller predicted geometric mean of the primal integral values.

Table 5-8 contains the parameter estimates for the SVM class model obtained using screening with grouping one.

Term	Estimate	General Setting	Setting One	Setting Two	Setting Three
Intercept	48.13	48.13	48.13	48.13	48.13
Disjcuts	3.56	3.56*DisjcutsParameterValue	-1	1	2
Fraccand	8.79	8.79*FraccandParmeterValue	10	10	10000
Fracpass	9.13	9.13*FracpassParameterValue	0	0	0
Disjcuts*Fraccand	-9.56	-9.56			
Predicted Geometric Mean of the Primal Integral	-------->		228.07	43.99	-103244.8

4.4 Screening Using a Marginal Analysis and General Linear Models

4.4.1 Marginal Analysis Screening

Another way to screen a large number of factors is to conduct a marginal analysis which uses analysis of variance to test factors one at a time as a predictor of the response. The key indicator for the test is the false discovery rate (FDR) p-value. It is may be important to control for FDR when conducting a large number of tests. Benjamini and Hochberg, (1995) define FDR as the expected proportion of errors among the rejected hypothesis. This technique considers not only if a type-1 error occurred, but it also considers the number of errors made. A type-1 error is a false positive. In Figure 4.7 the FDR's P-value, blue dot, and the P-value, red dot, are ranked by significance. The blue line indicates that blue points that fall below that line have corresponding FDR p-values that are significant. The red line indicates that red points that fall below the line have corresponding p-values that are significant. Both p-values increase from left to right, therefore the points on the left of the graph under the blue line indicate factors that have a significant effect on the response while controlling for false discovery.

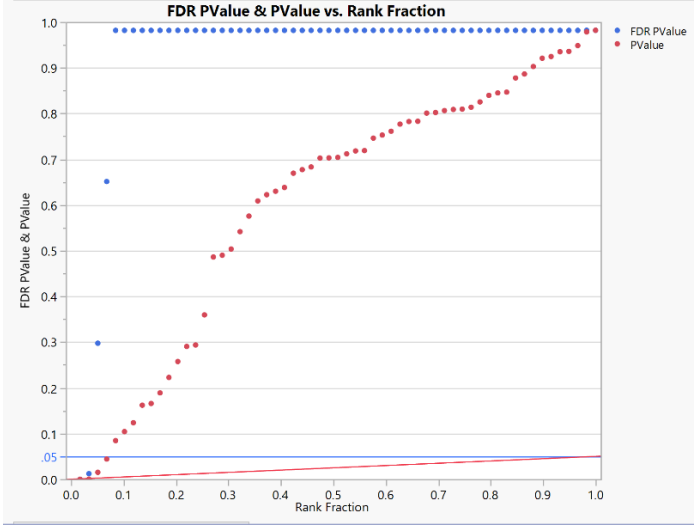


Figure 5-7 Plots the FDR P-value and P value versus the rank function. Blue points that are below the blue line are of more interest because they have a significant FDR P-value. This plot is for the telecommunications class with a Plackett-Burman design with 120 design points.

Creating an optimal design is an optimization problem itself, which becomes more complex with the high dimensionality of this problem. Due to the nature of the 59 variables being discrete, continuous, ordinal, and mixed level categorical, I first considered creating a Bayes D-optimal design. A D-optimal design for these factors would take 4974 designs points and using a rule of thumb that suggest using at least half of the 4974 design points, the Bayes D-optimal design would have 2487 design points. Due to the limitations of my computer Ram (16 GB), I was not able to create that size design with JMP. All categorical factors were reduced to only two-levels so that two-level experiment could be designed. A D-optimal design would minimally require 1771 design points and the Bayes D-optimal design would have 886 design points. (Note that the Bayes D-optimal design took a little under 15 minutes to create now that the dimensionality was reduced.) To use this design for screening all three classes, it would take about 209 days for a single core to complete the screening (about 3 days for 70 cores). The length of time for this screening may not be practical. Instead of using the Bayes D-optimal design, the experiments

conducted focus on smaller designs that can be created now that all factors have just two levels. Table 4.9 contains a list of the CPLEX parameters that were reduced to two-levels for the marginal analysis screening along with the value of the two levels chosen ..

Table 5-9 contains a listing of all of the categorical factors that were changed to two-levels for the marginal analysis screening; it also lists the two levels selected for the design.

Categorical Parameter	Level 1	Level 2
Mipemphasis	0 balance feasibility with optimality (default)	1 Emphasize feasibility
nodesel	0 depth first	1 bestbound search default
Varsel	-1 min infeasible rule	3 strong branching
Nodetype	1traditional dive	2 probing dive
Craind	0	1
Dpriind	1 standard dual pricing	2steepest-edge pricing
Ppriind	-1 reduced cost pricing	1 devex pricing
siftalg	1 primal	4 barrier
Scaind	-1 no scaling (0 equilibrium scaling default)	1 more aggressive scaling
fpheur	-1 turn off heuristic	1 turn on heuristic
Subalg	1 primal simplex	2 dual simplex (this is the method default will choose for MILP)
Startalg	2 dual simplex	6 concurrent primal, dual, barrier
Coeredind	1 reduce to integral coefficients	2 reduce all potential coefficients
depind	0 turn off looking for dependencies	3 look for dependencies at the beginning and the end of pre-solve
Preslvnd	-1 no node pre-solve	1 force pre-solve at nodes
Reduce	2 Only dual reductions during preprocessing	3 Both primal and dual reductions (default
Repeatpresolve	0 Turn it off	2 Repeat pre-solve with cuts

Plackett-Burman and fractional factorial designs can both be used as good screening designs for two-level factors and would take 60 or 64 design points respectively, to screen for just main effects. The advantage the Plackett-Burman design has over the fractional factorial design is that

it has four fewer design points while still being a resolution three design. Two different Plackett-Burman designs were used; a resolution three Plackett-Burman design with 60 runs, and a resolution four folded Plackett-Burman with 120 runs. The marginal analysis for the telecommunications class had one significant FDR p-value for the factor: Rinsheur. Due to the fact that the marginal analysis produced either one or zero significant factors the criteria was relaxed to include factors that have significant p-values makes a list of 4 factors: Rinsheur, Perlim, Scaind and Threads. Table 4.10 shows the results of the initial marginal analysis screening for all three classes of MIPs. For the Plackett-Burman 60res3_2level design, using the FDR p-value is too conservative and thus factors with just a significant p-value were included. The follow up screening of the Plackett-Burman60res3_2level design for all three classes of MIPs are full factorial designs with 16 runs each.

Table 5-10 list the factors obtained by using marginal analysis for the initial screening of all three classes of MIPs.

Marginal Analysis for the First Screening		**Design: Plackett-Burman60res3_2level**
Factors – 59		Metric – Geometric mean of the primal integral
Blue lettering indicates that the factor has a significant FDR p-value.		
Red lettering indicates that the factor has a significant p-value.		
Telecommunication Network -E	Coding Theory - H	SVM -M
Rinsheur	**Reduce**	Nodesel
Prelim	Cuts factor	mircuts
Scaind	eachcutlim	cutpass
Threads	Preind(int)	Cutsfactor

Table 4.11 shows the results of the initial marginal analysis screening for all three classes of MIPs for the Plackett-Burman_folded_120res3_2level design. When creating follow-up designs for the three classes, only the factors that had a significant FDR p-value were used for the telecommunication network and the coding theory class. The SVM class had just one significant factor determined with the FDR p-value, so the screening criteria was relaxed to include factors that also had a significant p-value.

Table 5-11 list the factors obtained by using marginal analysis for the initial screening of all three classes of MIPs

Marginal Analysis for the First Screening **Design: Plackett-Burman_folded_120res3_2level** Factors – 59 Metric – Geometric mean of the primal integral **Blue** lettering indicates that the factor has a significant FDR p-value. Red lettering indicates that the factor has a significant p-value. * Indicates that the factor was used for the next round of screening.		
Telecommunication Network Class - E	Coding Theory - H	SVM -M
Scaind*	Preind(int)*	Nodesel*
Rinsheur*	Reduce*	Cutpass*
Threads*	Cutsfactor	Cutsfactor*
Reinv	Eachcutlim	Rinsheur*
Fpheur		
Cutsfactor		

The follow up designs for the telecommunication network, coding theory and SVM classes are a full factorial with eight runs, a full factorial with four runs, and a full factorial with 16 runs

respectively. Due to the small number of remaining factors after the first screening using a marginal analysis, the second screening and then modeling was completed using general linear models with double adaptive LASSO.

Belloni et al., (2014), developed double LASSO to remove bias that occur in LASSO from underestimating coefficients that are nonzero and omitting covariates. Belloni et al., (2012; Zou, (2006) developed adaptive LASSO to overcome the inconsistencies that sometimes arise when using LASSO, developed by Tibshirani, (1996), as a variable selection procedure. Adaptive LASSO penalizes coefficients in the L_1 penalty using adaptive weights (Zou, 2006). However, the double LASSO used by JMP is not Belloni's method. Instead JMP's double adaptive LASSO feature performs variable selection with an initial adaptive LASSO model and then uses the variables selected in stage one as the input variables for the final adaptive LASSO model.

Two criteria were experimented with, the Corrected Akaike's Information Criterion (AICc) Hurvich andTsai, (1989) Akaike, (1973); and the Extended Regularized Information Criterion (ERIC) (Hui et al., 2015). The AICc tended to help select a model that produced a lower geometric mean of the primal integral, as seen in Table 4.12. For the telecommunications network class and the coding theory class, model *PB120EMA64f6runs6resLaicc_.9min_.1max* and model *PB60HMA4f16runRes6Laicc_min1_max0* respectively, are the best models found using the marginal analysis, yet both do not improve on default settings. However, for the SVM class, model *PB60MMA3f8runs6resLaicc_min.9_max.1* does 11% better than default settings. The coding theory class's recommended settings do not improve upon default. Model *PB60HMA4f16runRes6Laicc_min1_max0* is 3.62% more than default. A key to explain the naming convention used in Table 4.12can be found at the bottom row of Table 4.12.

Table 5-12 Shows results from screening using a marginal analysis for the first screen and double LASSO for the second screening. A negative percent change indicates that the results were better than default settings. To interpret the model name, the key is provided at the bottom of the table.

Model Name	Geometric Mean of Primal Integral	Geometric Variance	Percent Change from Default Setting of the Geometric Mean of the Primal Integral (smaller is best)
Telecommunications Network			
PB60EMA2f4runs6resLaicc_min1_max0	7.87	145.12	126.19%
PB60EMA2f4runs6resLaicc_min.9_min.1	7.80	80.07	124.27%
defaultPB60	3.48	202.01	0.00%
PB120EMA6f64runs6resLaicc_1min_0max	10.11	109.81	201.56%
PB120EMA64f6runs6resLaicc_.9min_.1max	5.90	650.35	76.07%
PB120EMA6f64runs6resLaicc_.8min_.2max	11.52	218.74	243.67%
PB120EMA6f64runs6resLaicc_.75min_.25max	8.92	519.24	165.97%
PB120EMA6f64runs6resLaicc_.9min_.1min	11.57	92.04	244.94%
PB120EMA6f64runs6resLaicc_.8min_.2min	6.08	645.12	81.31%
defaultPB120	3.35	198.17	0.00%
Coding Theory			
PB60HMA4f16run6resLeric_min1_max0	104.35	5.72	5.21%
PB60HMA4f16run6resLeric_min.9_min.1	104.61	5.58	5.48%
PB60HMA4f16run6resLeric_min.8_min.2	114.48	4.46	15.43%
PB60HMA4f16runRes6Laicc_min1_max0	102.76	5.31	3.62%
PB60HMA4f16runRes6Laicc_min.9_min.1	115.61	4.51	16.57%
defaultPB60	99.18	4.80	0.00%
PB120HMA4f16runs6resLaiccmin1_max0	488.04	1.30	395.42%
defaultPB120	98.51	4.87	0.00%
Support Vector Machine			
PB60MMA3f8runs6resLaicc_min1_max0	43.23	712.64	18.82%
PB60MMA3f8runs6resLaicc_min.9_max.1	32.38	854.70	-11.00%
PB60MMA3f8runs6resLaicc_min.8_min.2	41.17	777.29	13.17%
PB60MMA3f8runs6resLeric_min.9_min.1	44.66	620.00	22.76%
PB60MMA3f8runs6resLeric_min.8_min.2	42.42	735.78	16.61%
defaultPB60	36.38	854.36	0.00%
PB120MMA4f16runs6resLaicc_min1_max0	63.60	316.64	0.57%
PB120MMA4f16runs6resLaicc_min.9_min.1	74.24	225.58	17.39%
PB120MMA4f16runs6resLaicc_min.8_max.2	75.14	224.36	18.81%
defaultPB120	63.24	310.84	0.00%

Key to the Naming of the Models										
PB	**120, or 60**	**E, H, M**	**MA**	**#f**	**# runs**	**#res**	**L**	**aicc, or eric**	**_min# or _max#**	**_min# or _max#**
Plackett-Burman design was the initial screening design.	The number of initial design points.	E represents the telecommunication network class, H represents the coding theory class, M represents the SVM clas	The first screening was done with a marginal analysis	This represents the number, #, of factors that remain after the first screening	This represents the number, #, of design points in the second screening.	Represents the resolution of the fractional or full factorial second screening design	Represents that LASSO was used for the second screening.	Represent the criterion use for LASSO	Represents that the desirability function of the geometric **mean** of the primal integral is being minimized or maximized. The number, # represents the importance weight.	Represents that the desirability function of the geometric **variance** of the primal integral is being minimized or maximized. The number, # represents the importance weight.

Appendix C contains the parameter estimates for the models created with the marginal analysis.

4.4.2 Screening with General Linear Models

After the initial screening of the 59 two-level factors from the Plackett-Burman resolution III design with 60 runs was completed using adaptive double LASSO with an alpha =.05, the factors remaining are listed in Table 4.9. After the second screening, the factors listed in red in Table 4.13 are removed.

Table 5-13 Contains the list of factors that remain after the first screening of the Plackett-Burman resolution III design with 60 runs using adaptive double LASSO for the three classes. The telecommunication network, coding theory, and SVM classes have 13, 9, and 6 factors remaining respectively. The factors that are written in red were later removed after the second screening.

Telecommunication Network	Coding Theory	SVM
Cliques	Flowcovers	Cutpass
Cutsfactor	Cutsfactor	Cutsfactor
Eachcutlim	Eachcutlim	Nodesel
Flowpaths	Strongcandlim	Rinsheur
Gubcovers	Cutpass	Prelinear
Dpriind	Preind(int)	Preind(int)
Perlim	Reduce	
Reinv	Reinv	
Scaind	Mipsearch	
Bbinterval		
Subalg		
Rinsheur		
Predual		

After the initial screening of the 59 two-level factors from the folded Plackett-Burman resolution IV design with 120 runs was completed using adaptive double LASSO with an alpha =.05, the factors remaining are listed in Table 4.13. After the second screening, the factors listed in red in Table 4.14 are removed.

Table 5-14 Contains the list of factors that remain after the first screening of the Plackett-Burman resolution IV design with 120 runs using adaptive double LASSO for the three classes. The telecommunication network, coding theory, and SVM classes have 8, 6, and 10 factors remaining respectively. The factors that are written in red were later removed after the second screening.

Telecommunications Network	Coding Theory	SVM
Varsel	Cliques	Nodesel
Cutsfactor	Cutsfactor	Cutsfactor
Ppriind	Eachcutlim	Rinsheur
Scaind	Reinv	Scaind
Fpheur	Reduce	Cutpass
Subalg	Preind(int)	Fpheur
Rinsheur		Eachcutlim
Nodesel		Preind(int)
		Heurfreq
		Mircuts

Utilizing the factors in Table 4.14 for the second screening, full factorial and fractional factorial designs were created. For the telecommunications network, and coding theory classes, fractional factorial designs of the type 2^{13-6} resolution IV, and 2^{9-2} resolution V were created respectively. For the SVM class, a full factorial 2^6 design was created. After running the computer experiment, the second screening was completed using the adaptive double LASSO. A first order model with interactions was created. At this stage, two different desirability functions were applied with different importance weighting in order to obtain recommended settings for a class of instances. These recommended settings were then tried and the results are provided in Table 4.15. For the telecommunications network class and the coding theory class the best models are

PB120ELaicc8f128runs5res_1min_0max and *PB120HLaicc6f64runs6_min.8_max.2* respectively. Both aforementioned models are not better than default settings. However, notice that for both classes, their best performing models have the lowest number of factors, use AICc for the criterion in the double LASSO process and have a resolution of five or higher. The model that does the best for the SVM class is *PB120MLaicc10f128runs5res_min_1max_0* which is 29.33% better than default settings. In the case of the SVM class of instances, five other models also do better than default settings. In all cases estimation of the time it takes for tuning is calculated based on the number of design points, 120 + 64 =184, and the time limit, 10 minutes, set for the solving of an individual instance. For example, multiply the number of design points by the time limit and for model *PB120HLaicc6f64runs6_min.8_max.2* it takes at most 1840 minutes with one core per instance, or about 27minutes for 70 cores per instance. That is about four hours and 22 minutes for the coding theory class of instances.

Table 5-15. Shows results from screening using regression analysis with general linear models. Both the first and second screening used double LASSO feature in JMP statistical software. A negative percent change indicates that the results were better than default settings. To interpret the model name, the key is provided at the bottom of the table.

Model Name	Geometric Mean of Primal Integral	Geometric Variance	Percent Change from Default Setting of the Geometric Mean of the Primal Integral (Smallest is best)
Telecommunications Network			
PB60ELaicc13f128runsRes4_min1_max0	15.03	73.08	332.22%
PB60ELaicc13f128runsRes4_min.9_max.1	26.26	346.56	655.23%
PB60ELaicc13f128runsRes4_min.8_max.2	18.27	134.27	425.30%
PB60ELaicc13f128runsRes4_min.9_min.1	14.95	75.82	329.85%
defaultPB60	3.48	202.01	0.00%
PB120ELeric6f64runs6res_1min_0max	7.23	144.56	115.59%

PB120ELeric6f64runs6res_.9min_.1max	7.61	162.87	126.85%
PB120ELeric6f64runs6res_.8min_.2max	13.78	249.30	311.05%
PB120ELeric6f64runs6res_.9min_.1min	9.17	85.94	173.41%
PB120ELaicc16f128runs4res_1min_0max	13.55	511.38	304.26%
PB120ELaicc16f128runs4res_.9min_.1max	33.54	454.93	900.20%
PB120ELaicc16f128runs4res_.9min_.1min	24.35	279.55	626.24%
PB120ELaicc16f128runs4res_.6min_.4max	42.74	210.76	1174.82%
PB120ELaicc8f128runs5res_1min_0max	7.08	850.49	111.03%
PB120ELaicc8f128runs5res_.9min_.1max	7.75	948.57	131.03%
PB120ELaicc8f128runs5res_.9min_.1min	7.25	126.19	116.10%
PB120ELaicc8f128runs5res_.1min_.9min	13.92	205.73	315.29%
defaultPB120	3.35	198.17	0.00%
Coding Theory			
PB60HLaicc9f128runs5res_min1_max0	111.83	6.47	12.75%
PB60HLaicc9f128runs5res_min.8_min.2	116.14	6.64	17.10%
defaultPB60	99.18	4.80	0.00%
PB120HLeric4f16runs6resmin1_max0	102.75	5.63	4.31%
PB120HLeric4f16runs6resmin.9_max.1	101.61	5.22	3.15%
PB120HLeric4f16runs6resmin.9_min.1	101.43	4.72	2.97%
PB120HLeric4f16runs6resmin.8_min.2	101.04	4.72	2.56%
PB120HLaicc6f64runs6res_min1_max0	110.42	6.62	12.09%
PB120HLaicc6f64runs6res_min.9_min.1	114.75	6.80	16.49%
PB120HLaicc6f64runs6res_min.8_max.2	100.20	5.18	1.71%
PB120HLaicc9f128runs5res_min1_max0	108.48	6.75	10.12%
PB120HLaicc9f128runs5res_min.9_max.1	107.67	6.52	9.30%
PB120HLaicc9f128runs5res_min.9_min.1	118.93	6.56	20.73%
PB120HLaicc9f128runs5res_min.8_min.2	114.27	6.69	16.00%
defaultPB120	98.51	4.87	0.00%
Support Vector Machine			
PB60MLaicc6f64runs6res_min1_max0	88.57	213.07	143.47%
PB60MLaicc6f64runs6res_min.9_max.1	573.21	1.04	1475.62%
PB60MLaicc6f64runs6res_min.9_min.1	64.61	222.95	77.59%
PB60MLaicc6f64runs6res_min.8_min.2	60.48	281.07	66.26%
defaultPB60	36.38	854.36	0.00%
PB120MLaicc10f128runs5res_min_1max_0	44.69	382.03	-29.33%
PB120MLaicc10f128runs5res_min_.9min.1	70.03	239.50	10.72%
PB120MLaicc4f16runs6res_min_1max0	58.65	304.63	-7.26%
PB120MLaicc4f16runs6res_min_.9min_.1	58.78	309.62	-7.06%
PB120MLaicc4f16runs6res_min_.8min.2	60.48	289.28	-4.37%
PB120MLeric3f8runs6res_min1_max0	63.04	315.37	-0.33%
PB120MLeric3f8runs6res_min.9_min.1	62.96	319.83	-0.45%
PB120MLeric3f8runs6res_min.8_min.2	96.67	201.25	52.86%
defaultPB120	63.24	310.84	0.00%

Key to the Naming of the Models									
PB	**120, or 60**	**E, H, M**	**L**	**aicc, or eric**	**#f**	**# runs**	**#res**	**_min# or _max#**	**_min# or _max#**
Plackett-Burman design was the initial screening design.	The number of initial design points.	E represents the telecommunication network class, H represents the coding theory class, M represents the SVM class	The first screening was done with general linear model utilizing adaptive double LASSO.	Represent the criterion used for LASSO	This represents the number, #, of factors that remain after the first screening	This represents the number, #, of design points in the second screening.	Represents the resolution of the fractional or full factorial second screening design	Represents that the desirability function of the geometric **mean** of the primal integral is being minimized or maximized. The number, # represents the importance weight.	Represents that the desirability function of the geometric **variance** of the primal integral is being minimized or maximized. The number, # represents the importance weight.

Parameter estimates for the Plackett-Burman resolution III design with 60 runs models and the folded Plackett-Burman resolution IV designs with 120 runs (that performed the best) can be found in appendix D.

4.5 Results for the Extended Experiments

Table 4.16 contains a summary of the results of the extended experiment; listing the best performing models for each class and each screening technique. For the telecommunication network and SVM classes, it is clear that the method of using group screening performs best. The drawback for this technique is that it is only valid for a two-level experiment. Therefore, categorical variables that had more than two levels were not used and only 39 parameters were considered for tuning. The default setting outperforms all models created with the three screening techniques for the coding theory class. However, one can learn more from our models, even the models that did not outperform the default settings. One thing learned from the process

of creating and using these models is the important parameters that should be considered when tuning new problems from one of the classes.

From Tables 4.10, 4.11, 4.13, and 4.14 we note that all classes tend to have *cutsfactor* as an important factor this is because it limits the number of cuts that can be added. This is often useful when a lot of cuts are being made and little progress is being made in shrinking the solution space. Also, *fpheur* and *rinsheur*, two of the parameters that act as the on off switch for their respective heuristics are found in both the telecommunication network and SVM classes. The coding theory class has *preind (int)* as an important factor and this tells us that pre-solve plays an important role in the solution process of this class. It also has *eachcutlim* as important, and this parameter limits all of the cuts made to a specific number specified by the user. Controlling the number of cuts made in this class must be important to the performance of the solver for the coding theory class. Initially the coding theory class had the *cliques* parameter, in Table 4.14, but later it was screened out. This may seem counter intuitive because clique cuts are useful in node packing problems. However, note that the default settings for all of the cut parameters is to let CPLEX choose how many cuts to make and when to stop making those cuts and also *eachcutlimit* would also apply to *cliques* so it may be that it is not necessary because *eachcutlimit* is limiting how many cliques cuts that can be made. The fact that this was screened out may just indicate that the default setting is doing a fairly good job with deciding how many cuts to make.

Table 5-16 Summary results from the extended experiment with 59 parameters. Cplex optimization solver was used along with three different screening strategies.

Summary Results from the Extended Experiment- Tuning 59 Parameters Using CPLEX					Best from Each Type of Screening		
Class of Instances	**Screening Type and Design**	**Design Name**	**Number of Screening Design Points**	**Estimated Tuning Time in Hours per Class with 70 Cores**	**Geometric Mean of Primal Integral**	**Default**	**Percent Change from Default**
Telecommunication Network	Grouping with Forward Selection (39 factors)	Eg1m3v1	20/64 total 84	2.4	2.9	4.16	-30.32
	Grouping with Forward Selection (59 factors)	Eff11f128_min1_min0	24/128/128	8	14.11	21.95	-35.71
	Marginal Analysis with Adaptive Double LASSO	PB120EMA6f64runs6resLaicc_.9min_.1max	120/64 total 184	5.26	5.9	3.35	76.07
	Adaptive Double LASSO used Sequentially	PB120ELaicc8f128runs5res_1min_0max	120/128 total 248	7.09	7.08	3.35	111.03
SVM	Grouping with Forward Selection (39 factors)	Mg2m8v2 or Mg2m3v1 (tied)	20/32 total 52	1.49	14.24	52.00	-72.62
	Grouping with Forward Selection (59 factors)	Mff18f128r_min.8_min.2	24/128	3.66	95.42	201.90	-52.74
	Marginal Analysis with Adaptive Double LASSO	PB60MMA3f8runs6resLaicc_.9min_.1max	60/8 total 68	1.94	32.38	36.38	-11.00
	Adaptive Double LASSO used Sequentially	PB120MLaicc10f128runs5res_1min_0max	120/128 total 248	7.09	44.69	63.24	-29.33
Coding Theory	Grouping with Forward Selection (39 factors)	Hg2n9v1	20/128/24/128 total 300	8.57	104.37	75.38	38.46

	Grouping with Forward Selection (59 factors)	Hff9f128r_min.9_min.1	24/128	3.05	81.15	78.59	3.25
	Marginal Analysis with Adaptive Double LASSO	PB60HMA4f16runs6resLaicc_1min_0max	60/16 total 76	1.81	102.76	99.18	3.62
	Adaptive Double LASSO used Sequentially	PB120HLaicc6f64runs6res_.8min_.2max	120/64 total 184	4.38	100.2	98.51	1.71

In Tables 4.17 and 4.18, parameter estimates for the geometric mean and variance models for the telecommunication network class. These models, both classified as Eg1m3v1, obtained the best results in making a recommendation for a setting that would do better than the default setting for this class. These models contain parameters and their interactions that are significant in the tuning process of the telecommunication network class.

Table 5-17 Parameter estimates for the geometric mean model along with the standard error, Wald ChiSquare statistic, and the p-value. This model along with its corresponding geometric variance model gave the best recommendation for the parameter settings for the extended experiment's telecommunications network class.

Telecommunications Network - Mean Group1 Screening PB20 & Fractional Factorial 2^{8-3}				
Term	Parameter Estimate Geometric Mean Model	Std Error	Wald ChiSquare	Prob > ChiSquare
Fpheur	3.9072455	0.499658831	61.14969069	<.0001
Heurfreq	3.681366344	0.499658831	54.28388732	<.0001
Predual	5.090103281	0.499658831	103.778181	<.0001
Preind	2.710042531	0.499658831	29.41745371	<.0001
Prelinear	11.812015	0.499658831	558.8571927	<.0001
Prepass	11.15726928	0.499658831	498.6188527	<.0001
Fpheur*Heurfreq	1.544345125	0.499658831	9.553039825	0.002
Fpheur*Prelinear	1.360224656	0.499658831	7.41095457	0.0065
Fpheur*Symmetry	3.216588813	0.499658831	41.44231042	<.0001
Heurfreq*Predual	-3.856416281	0.499658831	59.56905091	<.0001
Predual*Symmetry	1.038604219	0.499658831	4.320689226	0.0377
Preind*Prelinear	2.4764305	0.499658831	24.56434304	<.0001
Preind*Prepass	2.950566281	0.499658831	34.87093685	<.0001
Prelinear*Prepass	12.09217913	0.499658831	585.6821782	<.0001
Prelinear*Relaxpreind	1.375280906	0.499658831	7.575925448	0.0059
Prepass*Relaxpreind	1.656139688	0.499658831	10.98618213	0.0009

Table 5-18 Parameter estimates for the geometric variance model along with the standard error, Wald ChiSquare statistic, and the p-value. This model along with its corresponding geometric mean model in Table 7.19. l gave the best recommendation for the parameter settings for the extended experiment's telecommunications network class.

Telecommunications Network - Variance Group1 Screening - Forward Selection PB20 & Fractional Factorial 2^{8-3}				
Term	Parameter Estimate Geometric Variance Model	Std Error	Wald ChiSquare	Prob > ChiSquare
Intercept	37.01746913	2.317558614	255.1243975	<.0001
Fpheur	-12.72442463	2.317558614	30.14496982	<.0001
Heurfreq	-18.55911144	2.317558614	64.1287693	<.0001
Predual	-23.10056584	2.317558614	99.35363988	<.0001
Preind	-5.56169975	2.317558614	5.75908682	0.0164
Prepass	-4.915411625	2.317558614	4.498402698	0.0339
Fpheur*Heurfreq	6.08528125	2.317558614	6.894453662	0.0086
Fpheur*Predual	11.11017603	2.317558614	22.98160874	<.0001
Fpheur*Symmetry	-4.974289438	2.317558614	4.606813704	0.0318
Heurfreq*Predual	18.85409641	2.317558614	66.18354007	<.0001
Heurfreq*Preind	6.0104825	2.317558614	6.726005536	0.0095
Predual*Preind	4.683973031	2.317558614	4.084767276	0.0433
Preind*Prepass	-5.588291563	2.317558614	5.814289609	0.0159
Relaxpreind*Symmetry	-6.854877031	2.317558614	8.748586362	0.0031

Table 4.19 contains the parameter estimates for the geometric variance model for the SVM class. This model, Mg2M8V2, obtained the best results in making a recommendation for a setting that would do better than the default setting for this class. This model contains parameters and their interactions that are significant in the tuning process of the SVM class. There is no geometric mean model for this design due all factors being removed during the second screening using forward selection.

Table 5-19 Parameter estimates for the geometric variance model along with the standard error, Wald ChiSquare statistic, and the p-value. This model gave the best recommendation for the parameter settings for the extended experiment's SVM class.

SVM - Variance			PB20 & Fractional Factorial 2^{6-1}	
Term	Parameter Estimate Geometric Variance Model	Std Error	Wald ChiSquare	Prob > ChiSquare
Intercept	460.8488231	40.56379868	129.0742851	<.0001
Implbd	80.92702944	40.56379868	3.980246427	0.046
Zerohalfcuts	-79.87085056	40.56379868	3.877031964	0.049
Implbd*Bndstreng	75.73284538	40.56379868	3.485710471	0.0619
Zerohalfcuts*Bndstreng	114.9061139	40.56379868	8.024334734	0.0046

In Tables 4.20 and 4.21 parameter estimates for the geometric mean and variance models for the coding theory class. Default settings outperformed the recommended setting from these models, both classified as *PB120HLaicc6f64runs6res_.8min_.2max*. These models contain parameters and their interactions that are significant in the tuning process of the telecommunication network class.

Table 5-20The parameter estimates for the geometric mean of the primal integral model for the coding theory class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table. The original screening file was a folded Plackett-Burman design with 120 runs and the follow-up design was a fractional factorial 2^6.

Coding Theory - Mean Factorial 2^6				PB120 & Full
Term	Parameter Estimate Geometric Mean Model	Std Error	Wald ChiSquare	Prob > ChiSquare
Intercept	155.3585191	18.336465	269.9722565	<.0001
Cliques	-3.266308234	18.336465	0.177925865	0.6732
Cutsfactor	-13.37826089	18.336465	2.984860748	0.084
Eachcutlim	-12.98937892	18.336465	2.813853776	0.0935
Reinv	-32.8287088	18.336465	17.9734876	<.0001
Preind	300.463174	18.336465	653.6302046	<.0001
Reduce	309.5079306	18.336465	676.1394682	<.0001
Cliques*Cutsfactor	-11.11477342	18.336465	8.095295086	0.0044
Cliques*Eachcutlim	-10.83075002	18.336465	7.686852016	0.0056
Cliques*Reinv	-3.939580891	18.336465	1.0170247	0.3132
Cliques*Preind(int)	-7.981126656	18.336465	1.043517822	0.307
Cliques*Reduce	-7.363186719	18.336465	0.8881843	0.346
Cutsfactor*Eachcutlim	-30.54417836	18.336465	61.13477954	<.0001
Cutsfactor*Reinv	-10.78300886	18.336465	7.619235204	0.0058
Cutsfactor*Preind(int)	-18.00996297	18.336465	5.313700895	0.0212
Cutsfactor*Reduce	-19.09404453	18.336465	5.972653596	0.0145
Eachcutlim*Reinv	-10.15644945	18.336465	6.759511001	0.0093
Eachcutlim*Preind(int)	-17.47447241	18.336465	5.002413751	0.0253
Eachcutlim*Reduce	-18.40210872	18.336465	5.547619286	0.0185
Reinv*Preind(int)	-0.541357406	18.336465	0.004801088	0.9448
Reinv*Reduce	-1.430377344	18.336465	0.033517584	0.8547
Preind(int)*Reduce	-298.6789524	18.336465	365.3597593	<.0001

Table 5-21 The parameter estimates for the geometric variance of the primal integral model for the coding theory class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table. The original screening file was a folded Plackett-Burman design with 120 runs and the follow-up design was a fractional factorial 2^6

Coding Theory - Variance 2^6				PB120 & Full Factorial
Term	Parameter Estimate Geometric Variance Model	Std Error	Wald ChiSquare	Prob > ChiSquare
Intercept	5.813794688	0.1914643875	918.7057015	<.0001
Cliques	0.333345594	0.1914643875	12.22099172	0.0005
Cutsfactor	0.55448775	0.1914643875	33.8143407	<.0001
Eachcutlim	0.571412719	0.1914643875	35.91011682	<.0001
Preind(int)	-3.768400313	0.1914643875	189.0544517	<.0001
Reduce	-3.835066625	0.1914643875	194.4570607	<.0001
Cliques*Cutsfactor	0.316382031	0.1914643875	11.00881649	0.0009
Cliques*Eachcutlim	0.341904813	0.1914643875	12.85663871	0.0003
Cutsfactor*Eachcutlim	0.586830156	0.1914643875	37.87405967	<.0001
Preind(int)*Reduce	3.744758875	0.1914643875	96.39283357	<.0001

Chapter 5 Benchmarking

5.1 Benchmarking

Another important consideration for the practitioner is to determine which optimization software (MIP solvers) will provide the best performance for the types of problems they solve. Hans Mittelmann provides a website with benchmarking information for optimization software (Mittelmann, 2016). MIPLIB2010 Koch et al., (2011) is a test-bed library consisting of 361 instances of which 62% are classified as easy, 16% as hard, and 22% as not solved (Mittelmann, 2016). Mittelmann's benchmarking work involving MIPs offers a comparison of leading commercial and open source optimization software using 24% of the MIPLIB2010 test-bed containing only 'easy' instances (Mittelmann, 2016). Commercial optimization software outperforms the open source in terms of the number of instances solved and the time it takes to find an optimal solution. Mittelmann's results are based on using the optimizers tested at default settings. For the easy problems, the average scaled time ranges between 1 to 7 seconds for the top three optimizers under default settings. This information would not offer the necessary insights needed for users working with more difficult problems. Also, when comparing solvers with instances that solve in such a small amount of time, the differences between solvers may be attributed to variance that occurs in the operating system of the computer. Therefore, it is important that instances overall solution time is large enough that the small differences between

the solvers, do not dominate the measure and thus move the bias away from the operating system.

In order to address the fact that MIPLIB2010's instances might not be as relevant, based on the current need of researchers, because many of the instances solve quickly, the researchers that curated MIPLIB2010 have decided to update the collection in their library. In the fall of 2016, MIPLIB placed a call for submission of relevant, challenging and real-world problems to offer a modernized test-bed of MIPs in order to address the need for difficult test instances. Some criticism to this collection method are: suggested instances will not be diverse enough in type and level of difficulty, instances will be biased towards performing well with the researcher's developed work, the curators of the repository may not recognize a representative problem of a specific class because of the possibility of limited access to proprietary instances and emerging new problems (Hooker, 1995). Bowly et al., (2017); Hooker, (1995), recommend a more systematic approach of developing a testing pool of instances that will offer researchers a more robust group that provides a way to highlight algorithmic strengths and deficiencies.

Bowly et al., (2017) recent work developed a constructor generation approach to creating instances of LPs and MIPs. Although this work is promising because it tackles the limited diversity provided by simple random generation of instances, it needs strengthening in its ability to produce more difficult instances (Bowly et al., 2017). This suggests that collecting instances that are randomly generated, does not ensure the diversity of the instances and how well they will perform at providing clarity into algorithmic strengths or weaknesses. This leads to the question: What do we hope to gain from the information the benchmarking results provide?

The answer to that question varies because of how people use the information. Regular practitioners in business may use the information to decide which product (solver) will best meet their needs, and commercial solver businesses may use the benchmark results to help sell their product. Realistically the benchmark is not the only thing being considered by the practitioners. For example, many companies have chosen to use XPRESS which has been ranked third in commercial software benchmark results Mittelmann, (2017) for May and June 2017 results with MIPs. Companies use benchmarking as hype to help sell their product, but they are also offering other services that influence the consumer. Recent benchmark documents produced by the two leading commercial solvers have two distinct approaches to reporting benchmarking information. CPLEX compare to previous versions of their own product, and used benchmark results completed in house (IBM, 2016). Gurobi also compared their previous version with their current version but then also compares the top three commercial solvers using Mittelmann's benchmark results that were current at the time (Gurobi, 2016a; Mittelmann, 2017). However, researchers may have different interests in the information provided by benchmarking data. Yes, researchers want to know performance times for both parallelized and non-parallelized computers but researchers also want to compare algorithms or demonstrate the viability of new ideas and they need to provide conditions that highlight the strengths and weakness of their work. Optimization software is not only a tool used to solve problems, it is often the experimental environment in which we conduct the tests, along with the computers operating system. In this regard, it is always best when trying to show that an algorithm is better than the current leader that the environment we are testing in highlights the current leader's strengths in order to have a meaningful result from the comparison test. This implies that optimization software should be tuned in order to create that "best" test environment. More consideration of benchmarking of

portfolio of instances of the same class of problems could help researchers with their work, more information should be gained from the benchmark results.

The emphasis of this paper is the benchmarking process and therefore it was not attempted to show the results of many solvers, just to be illustrative. The benchmarking was conducted on Bach at Virginia Commonwealth University, which is a Linux Beowulf cluster with 500 processors it contains, 2.6 GHz Opteron, 1 TB RAM (4GB-32GB per node), 2TB direct attached Fiber Storage, and 16.8 TB internal disk storage (73GB per node). For the time period of this experiment six nodes with 24 cores were set completely apart from the rest of the cluster. Each instance tested was limited to one core unless benchmarking the parallel processing of the commercial software. The experiment compares the performance of CPLEX 12.7.1 (IBM, 2017) and Gurobi 7.01(Gurobi, 2016b) utilizing three classes of MIPs. The test-bed used for all experiments in this paper are from the following three classes of MIP problems:

1. Class M - A formulation of the support vector machine with the ramp loss and L1-norm regularization (Hess and Brooks, 2015)
2. Class E - Survivable fixed telecommunication network design(Orlowski et al., 2010) (The mps files were obtained from (Raack, 2014).)
3. Class H – Coding theory graphs – node packing problems (Slone, 2011)

All of the instances used in the experiments can be expressed as a minimization problem of the form:

$$\tilde{x}_{opt} = \arg\min\left\{c^T x \mid Ax \le b, x_j \in \mathbb{Z} \text{ for all } j \in J\right\} \text{ with } A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n, \text{ and } J \subseteq \{1,...,n\}$$

Each class of instances was run separately from the other classes. The conditions of the experiments were the following:

Conditions - 1

Both CPLEX and Gurobi were set to the researcher's defined default settings. The researcher's default settings changes some of the parameter values to help with numerical stability, limiting the use of parallel processing, and for the tracking of time and these can be found in Table 5.1.

Table 0-1 contains the settings for condition 1 which are the researcher's defined default settings.

Researcher Defined Default Settings for CPLEX	Researcher Defined Default Settings for Gurobi
CPX_PARAM_THREADS = 1	Threads =1
CPX_PARAM_TILIM = 600	TimeLimit = 600
CPX_PARAM_EPRHS = 1e-9	FeasibilityTol = 1e-9
CPX_PARAM_EPOPT = 1e-9	OptimalityTol =1e-9
CPX_PARAM_EPMRK = 0.99999	MarkowitzTol = 0.999
CPX_PARAM_EPINT = 0.0	IntFeasTol = 1e-9
CPX_PARAM_EPGAP = 0.0	MIPGap = 0.0
CPX_PARAM_EPAGAP = 0.0	MIPGapAbs = 0.0
CPX_PARAM_NUMERICALEMPHASIS = 1	NumericFocus = 3
CPX_PARAM_SCRIND = 0	LogToConsole = 0
CPX_PARAM_CLOCKTYPE = 2	

Conditions - 2

Both CPLEX and Gurobi were set to the respective software's default parameter values while still limiting the use of parallel processing limiting each instance to one thread, and the setting of the clock and time limit parameters. Table 5.2 contains the parameters and their settings.

Table 0-2 contains condition – 2 settings.

Researcher Defined Default Settings for CPLEX	Researcher Defined Default Settings for Gurobi
CPX_PARAM_THREADS = 1	**Threads =1**
CPX_PARAM_TILIM = 600	TimeLimit = 600

CPX_PARAM_CLOCKTYPE = 2	LogToConsole = 0
CPX_PARAM_SCRIND = 0	

Condition - 3

Both CPLEX and Gurobi were set to manufacture default settings except for the clock type and the time limit. The settings for condition – 3 are in table 5.3.

Table 0-3 contains the parameter settings for condition -3.

Researcher Defined Default Settings for CPLEX	Researcher Defined Default Settings for Gurobi
CPX_PARAM_TILIM = 600	TimeLimit = 600
CPX_PARAM_CLOCKTYPE = 2	LogToConsole = 0
CPX_PARAM_SCRIND = 0	

Condition - 4

For the benchmarking of the parallel processing capabilities of the solver, each solver was operated under Condition 1 with one difference. The number of threads was varied with in this set {1, 2, 4, 6, 8} For these experiments the number of cores used was also varied to correspond to the number of threads. For example, if the solver was using 6 threads, then 6 cores were set aside for its use.

Table 5.4 contains the results from the test run for conditions one through three and Table 5.5 contains the results from condition four.

Table 0-4 contains benchmarking information on the three different types of default defined in conditions 1-3. All three classes of instances were used to compare CPLEX and Gurobi. The bolded number identifies the solver that performed the best in that category.

Telecommunication Network						
Benchmark	Geometric Mean of the Primal Integral at 10 min		Geometric Variance of the Primal Integral at 10 min		Geometric Mean of the Solution Time	
	CPLEX	Gurobi	CPLEX	Gurobi	CPLEX	Gurobi
Condition1	**3.256878**	5.179621	201.188678	387.27349	**78.933865**	145.500703
Condition 2	12.323629	**3.648498**	37.013264	513.762196	307.476247	**130.015308**
Condition 3	7.212864	**2.029239**	79.808445	718.103239	131.106519	**70.838853**
SVM						
Condition1	37.410698	**32.916371**	794.198416	1606.17559	**215.290689**	330.665014
Condition 2	49.501945	**46.707048**	985.853657	1396.955964	390.124608	**377.965808**
Condition 3	106.571019	**30.550724**	225.633616	1588.991071	619.035495	**318.328135**
Coding Theory						
Condition1	100.02537	**54.008972**	5.072543	25.932053	600	**569.260218**
Condition 2	82.806567	**58.262951**	4.683364	35.919267	600	**550.931211**
Condition 3	126.683949	**45.311106**	4.210015	37.078859	600	**424.250121**

For the telecommunication network class and for both the geometric mean of the primal integral and solution time, Gurobi performs the best under condition 3 which is manufacture defaults with the geometric mean of the primal integral value of 2.029239 and the geometric mean of the solution time value of 70.838853 seconds. The SVM class had mixed results. Gurobi outperforms CPLEX under condition 3 with the best geometric mean of the primal integral value of 30.550724. However, CPLEX outperforms Gurobi with the best geometric mean of solution time value of 215.290689 seconds which is slightly over 103 seconds better that Gurobi's best time under condition 3 for the SVM class. With the

coding theory class, Gurobi out performs CPLEX under all three conditions, but performs best under condition 3.

Table 0-5 The results for the benchmarking for the use of parallelization. All three classes were tested.

Telecommunication Network						
Parallel Benchmark	Geometric Mean of the Primal Integral at 10 min		Geometric Variance of the Primal Integral at 10 min		Geometric Mean of the Solution Time	
	CPLEX	Gurobi	CPLEX	Gurobi	CPLEX	Gurobi
1 Thread	**3.259629**	4.779528	198.873633	436.358538	**79.342959**	140.718352
2 Thread	**2.394308**	2.972577	193.600244	483.756492	**58.620492**	104.790482
4 Thread	2.894798	**2.326789**	199.55357	723.41388	**67.341782**	81.696582
6 Thread	2.720452	**2.144849**	186.231989	716.305554	**61.295791**	74.408832
8 Thread	2.151491	**1.21874**	144.38364	459.253177	**42.770398**	57.372854
SVM						
1 Thread	36.641699	**23.28195**	815.628465	1375.269706	**208.137722**	229.193022
2 Thread	26.865271	**25.134694**	1104.196158	1478.991152	**168.399512**	206.462972
4 Thread	23.736463	**13.305731**	1223.382442	2715.820067	157.369099	**151.163926**
6 Thread	26.488685	**11.266299**	1137.351765	3499.938783	**155.26339**	166.290602
8 Thread	25.12425	**13.262276**	1052.941348	2380.660002	160.175299	**146.100172**
Coding Theory						
1 Thread	100.786986	**52.181655**	5.22363	27.110677	600	**537.170824**
2 Thread	95.499157	**48.532234**	5.195841	33.860779	600	**512.409697**
4 Thread	81.694179	**43.687127**	4.539458	36.078019	600	**464.930006**
6 Thread	73.570349	**51.386296**	6.128761	35.138686	600	**433.661071**
8 Thread	73.455649	**50.919968**	4.736249	36.963767	600	**428.18372**

Condition 4 is about how well CPLEX and Gurobi deal with utilizing a parallel environment. Here we varied the number of threads being used and consider two metrics, the geometric mean of the primal integral and solution time. For the telecommunication network class and the geometric mean of the primal integral metric, CPLEX performs better when one or two threads are being used, but when 4, 6, or 8 threads are being used, Gurobi performs better than CPLEX.

However, when looking at the geometric mean of the solution time for the telecommunication network class, CPLEX performs best with all of the number of threads. For the SVM class and the geometric mean of the primal integral metric, Gurobi performs best, under condition four, with all of the different number of threads tested. But when considering the geometric mean of the solution time CPLEX performs best with 1, 2 and 6 threads while Gurobi performs best with 4 and 8 threads. The coding theory class is a clean sweep for Gurobi under condition four. It performs better than CPLEX using both metric and with all of the different number of threads tested.

Table 5.6 contains the results obtained from benchmarking solvers that have been tuned for a specific class of MIPs. For a list of the parameters that have been tuned by CPLEX 12.71 and Gurobi 7.02 refer to appendix E and appendix F respectively. Table 5.6 shows us that CPLEX performs best for the three classes in terms of the geometric mean of the primal integral metric and it also does better than Gurobi when considering the geometric mean of the solution time metric for the telecommunication network and SVM classes. Gurobi's default setting does best for the coding theory class when utilizing the geometric mean of the solution time metric.

Table 0-6 has the performance of both CPLEX and Gurobi after being tuned. Tuning for each class and solver was under five hours.

Chapter 6 Conclusions, Limitations and Future Research

6.1 Conclusions

Using a DOE approach with a modeling framework by creating D-optimal designs to tune the six parameter settings of the limited experiment offers an improvement over CPLEX's default and autotuned settings. Although this approach does not always give the best recommended setting it competes well against other design's best run. No screening is necessary to fit a model when working with the limited case that had only six factors. The one thing we learn with the modeling framework that choosing the best run of a design does not offer is the ability to discern important parameters for a class of MIPs. From the model found for each class of MIPs we are able to discern important parameters and two-way interactions. It is also possible to identify parameters that have little to no effect on the response and can be removed from the model which makes it possible to also remove those parameters from the list of parameters that should be tuned for a class of MIPs. The models created help to recommend parameter settings, for a class of problems, that provided the greatest impact on a performance metric. For example, in the limited case the best setting for the telecommunications network, SVM, and coding theory classes are 29.21%, 35.06% and 26.07% better than default setting respectively.

Screening by grouping and then modeling out performed default settings in two out of three classes and shows promise with the third in the extended experiment that had 39 factors.

(We still gain information about the parameters.) In general, we were able to keep the time it takes to tune each class of MIPs to less than five hours for the extended experiment. Another contributing factor to the performance of this method is that we found treating categorical variables that have an ordinal quality as continuous reduces the number of computer runs, which reduces the overall time taken for tuning, and may give better tuning results. The screening by grouping and then modeling outperformed default settings on the newer version of CPLEX 12.71and Gurobi 7.021 as seen in the benchmarking work.

Benchmarking portfolio of instances and tuned instances can give more information that helps to identify solvers that will work best with a particular class and the important parameters for the class and solver combination. By performing benchmarking with classes of instances we not only increase insights about a class of MIPs, but there is no loss of information about the individual instances.

6.2Limiting Factors

Due to the nature of the design and modeling framework, it was necessary to use a Beowulf cluster for experimentation due to the large number of computer runs necessary for the experiments. Every effort was made to provide identical environments for each test such as, keeping a separated queue of nodes set aside for these experiments, assigning a specific number of cores per number of threads and in the case of the benchmarking experiment separating a portion of the cluster completely so that no other user was utilizing any of the resources of the "new" cluster. However, although the cluster does load balancing, it is not possible to perfectly balance the load on the nodes being used. Also, the computer operating system has variability that cannot be controlled. However, with cloud computing becoming more utilized, the results

here show that it is possible to obtain good results and information utilizing large computer arrays.

When performing benchmarking using tuned instances and equal tuning is not possible due to the individualistic nature of each solver's code. Tuning does not guarantee that the best setting is found, although that is possible but not easily identified when working with a large number of factors. But what we do gain is an idea of how long both solvers will take to tune and given that time, what type of performance improvement is obtained. We also gain obtain information about important parameters and interactions, so that future tuning of a new instance of the same class can concentrate on these parameters.

6.3 Future Work

In this research we dealt with discrete, continuous, and categorical variables at the same time. However, to provide a tuning experience in a minimal amount of time, it was necessary to limit the designs used to two-levels for the extended experiment. One way to partially dispense with the two-level requirement for follow-up screenings would be to proceed, as in the limited case, with constructing a design with a variety of categorical factors at multiple levels if the number of factors has been sufficiently reduced. Performance of the optimization system with this change in the follow-up could improve. Future work in the development of methods that deal with mixed-level designs for a large number of factors would be beneficial especially for an optimization solver that has a large number of categorical variables such as CPLEX. In choosing the groups for the group screening, documentation about parameters, provided by IBM ILOG CPLEX was used to assist in the assignment of parameters to groups. Further research into ensuring that members of the group are having similar effects on the response would improve

this screening method. However, if the number of factors is sufficiently reduced, you can dispense with the two-level requirement and proceed, as in the limited case, with constructing a design with a variety of categorical factors at multiple levels.

Bibliography

References

Achterberg, T., Berthold, T., & Hendel, G. (2012). Rounding and propagation heuristics for mixed integer programming. *Operations research proceedings 2011* (pp. 71-76) Springer.

Achterberg, T., & Wunderling, R. (2013). Mixed integer programming: Analyzing 12 years of progress. *Facets of combinatorial optimization* (pp. 449-481) Springer.

Adenso-Diaz, B., & Laguna, M. (2006). Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research, 54*(1), 99-114.

Agresti, A. (2010). *Analysis of ordinal categorical data* John Wiley & Sons.

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle,[w:] Proceedings of the 2nd international symposium on information, BN petrow, F. *Czaki, Akademiai Kiado, Budapest,*

Akaike, H. (2011). Akaike's information criterion. *International encyclopedia of statistical science* (pp. 25-25) Springer.

Ansótegui, C., Sellmann, M., & Tierney, K. (2009). A gender-based genetic algorithm for the automatic configuration of algorithms. Paper presented at the *International Conference on Principles and Practice of Constraint Programming*, 142-157.

Audet, C., & Orban, D. (2006). Finding optimal algorithmic parameters using derivative-free optimization. *SIAM Journal on Optimization, 17*(3), 642-664.

Balaprakash, P., Birattari, M., & Stützle, T. (2007). Improvement strategies for the F-race algorithm: Sampling design and iterative refinement. Paper presented at the *International Workshop on Hybrid Metaheuristics,* 108-122.

Bartz-Beielstein, T., Lasarczyk, C. W., & Preuß, M. (2005). Sequential parameter optimization. Paper presented at the *2005 IEEE Congress on Evolutionary Computation, , 1* 773-780.

Bartz-Beielstein, T., & Markon, S. (2004). Tuning search algorithms for real-world applications: A regression tree based approach. Paper presented at the *Congress on Evolutionary Computation CEC2004. , 1* 1111-1118.

Baz, M., Hunsaker, B., Brooks, P., & Gosavi, A. (2007). Automated tuning of optimization software parameters. *University of Pittsburgh Department of Industrial Engineering Technical Report 2007-7,*

Baz, M., Hunsaker, B., & Prokopyev, O. (2011). How much do we “pay” for using default parameters? *Computational Optimization and Applications, 48*(1), 91-108.

Belloni, A., Chen, D., Chernozhukov, V., & Hansen, C. (2012). Sparse models and methods for optimal instruments with an application to eminent domain. *Econometrica, 80*(6), 2369-2429.

Belloni, A., Chernozhukov, V., & Hansen, C. (2014). Inference on treatment effects after selection among high-dimensional controls. *The Review of Economic Studies, 81*(2), 608-650.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society.Series B (Methodological),* , 289-300.

Bertacco, L. (2006). *Exact and heuristic methods for mixed integer linear programs*

Berthold, T. (2013). Measuring the impact of primal heuristics. *Operations Research Letters, 41*(6), 611-614.

Birattari, M., Stützle, T., Paquete, L., & Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. Paper presented at the *Gecco, , 2* 11-18.

Birattari, M., Yuan, Z., Balaprakash, P., & Stützle, T. (2010). F-race and iterated F-race: An overview. *Experimental methods for the analysis of optimization algorithms* (pp. 311-336) Springer.

Bixby, R., & Rothberg, E. (2007). Progress in computational mixed integer programming—a look back from the other side of the tipping point. *Annals of Operations Research, 149*(1), 37-41.

Boland, N., Fischetti, M., Monaci, M., & Savelsbergh, M. (2016). Proximity benders: A decomposition heuristic for stochastic programs. *Journal of Heuristics, 22*(2), 181-198.

Borghetti, A. (2013). Using mixed integer programming for the volt/var optimization in distribution feeders. *Electric Power Systems Research, 98*, 39-50.

Bowly, S., Smith-Miles, K., Baatar, D. & Mittelmann, H. (2017). Optimization online - generation techniques for linear and integer programming instances with controllable properties. Retrieved from http://www.optimization-online.org/DB_HTML/2017/04/5976.html

Box, G. E., & Wilson, K. B. (1992). On the experimental attainment of optimum conditions. *Breakthroughs in statistics* (pp. 270-310) Springer.

Box, J. F. (1980). R. A. fisher and the design of experiments, 1922-1926. *The American Statistician, 34*(1), 1-7. doi:10.2307/2682986

Brain, Z. E., & Addicoat, M. A. (2010). Using meta-genetic algorithms to tune parameters of genetic algorithms to find lowest energy molecular conformers. Paper presented at the *Alife*, 378-385.

Cohen, D. M., Dalal, S. R., Fredman, M. L., & Patton, G. C. (1997). The AETG system: An approach to testing based on combinatorial design. *IEEE Transactions on Software Engineering, 23*(7), 437-444.

Coy, S. P., Golden, B. L., Runger, G. C., & Wasil, E. A. (2001). Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics, 7*(1), 77-97.

Danna, E. (2008). Performance variability in mixed integer programming. *Presentation Slides from MIP Workshop*, New York, NY.

DuMouchel, W., & Jones, B. (1994). A simple bayesian modification of D-optimal designs to reduce dependence on an assumed model. *Technometrics, 36*(1), 37-47.

Dunietz, I. S., Ehrlich, W. K., Szablak, B., Mallows, C. L., & Iannino, A. (1997). Applying design of experiments to software testing: Experience report. Paper presented at the *Proceedings of the 19th International Conference on Software Engineering*, 205-215.

Fischer, T., & Pfetsch, M. E. (2015). Branch-and-cut for linear programs with overlapping SOS1 constraints. *Mathematical Programming Computation*, , 1-36.

Fischetti, M., & Monaci, M. (2014a). Exploiting erraticism in search. *Operations Research, 62*(1), 114-122.

Fischetti, M., & Monaci, M. (2014b). Exploiting erraticism in search. *Operations Research, 62*(1), 114-122.

Gendron, B., Crainic, T. G., & Frangioni, A. (1999). Multicommodity capacitated network design. *Telecommunications network planning* (pp. 1-19) Springer.

Goycoolea, M., Murray, A. T., Barahona, F., Epstein, R., & Weintraub, A. (2005). Harvest scheduling subject to maximum area restrictions: Exploring exact approaches. *Operations Research, 53*(3), 490-500.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics, 16*(1), 122-128.

Gurobi Optimization. (2015). Gurobi 6.5 performance benchmarks. Retrieved from http://www.gurobi.com/pdfs/benchmarks.pdf

Gurobi, O. I. (2016a). Gurobi 7.0 performance benchmarks. Retrieved from http://www.gurobi.com/pdfs/benchmarks.pdf

Gurobi, O. I. (2016b). *Gurobi optimizer 7.1*. Houston, TX:

Hanssmann, F., & Hess, S. W. (1960). A linear programming approach to production and employment scheduling. *Management Science,* (1), 46-51.

Hess, E. J., & Brooks, J. P. (2015). The support vector machine and mixed integer linear programming: Ramp loss SVM with L1-norm regularization.

Hooker, J. N. (1995). Testing heuristics: We have it all wrong. *Journal of Heuristics, 1*(1), 33-42.

Hoskins, D., Turban, R. C., & Colbourn, C. J. (2004). Experimental designs in software engineering: D-optimal designs and covering arrays. Paper presented at the *Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research,* 55-66.

Hui, F. K., Warton, D. I., & Foster, S. D. (2015). Tuning parameter selection for the adaptive lasso using ERIC. *Journal of the American Statistical Association, 110*(509), 262-269.

Hurvich, C. M., & Tsai, C. (1989). Regression and time series model selection in small samples. *Biometrika, 76*(2), 297-307.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. Paper presented at the *International Conference on Learning and Intelligent Optimization*, 507-523.

Hutter, F., Hoos, H. H., Leyton-Brown, K., & Stützle, T. (2009). ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research, 36*(1), 267-306.

Hutter, F., Hoos, H. H., & Stützle, T. (2007). Automatic algorithm configuration based on local search. Paper presented at the *Aaai, , 7* 1152-1157.

IBM, C. (2017a). IBM knowledgecenter - list of CPLEX parameters 12.7.1. Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.7.0/ilog.odms.cplex.help/CPLEX/Parameters/topics/introListAlpha.html

IBM, C. (2016). **IBM ILOG CPLEX optimizer performance benchmarks**. Retrieved from https://www-01.ibm.com/software/commerce/optimization/cplex-performance/

IBM, C. (2017b). *IBM ILOG CPLEX optimization studion V12.7.1* (12.7.1 ed.) IBM Corporation.

Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R. E., . . . Heinz, S. (2011). Miplib 2010. *Mathematical Programming Computation, 3*(2), 103-163.

Koch, T., & Hendel, G. (2014). *Empirical analysis of solving phases in mixed integer programming.* (Thesis). Berlin, Germany:

Lodi, A., & Tramontani, A. (2013). Performance variability in mixed-integer programming. *TutORials in Operations Research: Theory Driven by Influential Applications,* , 1-12.

Mee, R. (2009). *A comprehensive guide to factorial two-level experimentation* Springer Science & Business Media.

Mittelmann, H. (2016). Decision tree for optimization software. Retrieved from http://plato.asu.edu/bench.html

Mittelmann, H. (2017). Decison tree for optimization software. Retrieved from http://plato.asu.edu/guide.html

Montgomery, D. C. (2009). *Design and analysis of experiments* (7th ed.). Hoboken, NJ: Wiley.

Montgomery, D. C. (2008). *Design and analysis of experiments* John Wiley & Sons.

Nannen, V., & Eiben, A. E. (2006). A method for parameter calibration and relevance estimation in evolutionary algorithms. Paper presented at the *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation,* 183-190.

National Research Council. (2012). *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas* National Academies Press.

Orlowski, S., Pióro, M., Tomaszewski, A., & Wessaly, R. (2007). Survivable network design library. Paper presented at the *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium SNDlib,*

Orlowski, S., Wessäly, R., Pióro, M., & Tomaszewski, A. (2010). SNDlib 1.0—survivable network design library. *Networks, 55*(3), 276-286.

Orso, A., & Rothermel, G. (2014). Software testing: A research travelogue (2000–2014). Paper presented at the *Proceedings of the on Future of Software Engineering,* 117-132.

Plackett, R. L., & Burman, J. P. (1946). The design of optimum multifactorial experiments. *Biometrika, 33*(4), 305-325.

Pochet, Y., & Wolsey, L. A. (2006). *Production planning by mixed integer programming* Springer Science & Business Media.

Pogue, G. A. (1970). An extension of the markowitz portfolio selection model to include variable transactions'costs, short sales, leverage policies and taxes. *The Journal of Finance, 25*(5), 1005-1027.

Raack, C. (2014). Index of /raack/downloads/INOC2007_instances. Retrieved from http://www.zib.de/raack/downloads/INOC2007_instances/

Raack, C., Koster, A. M., Orlowski, S., & Wessäly, R. (2011). On cut-based inequalities for capacitated network design polyhedra. *Networks, 57*(2), 141-156.

ReVelle, C. S., & Swain, R. W. (1970). Central facilities location. *Geographical Analysis, 2*(1), 30-42.

Rhemtulla, M., Brosseau-Liard, P. E., & Savalei, V. (2012). When can categorical variables be treated as continuous? A comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. *Psychological Methods, 17*(3), 354.

Ridge, E., & Kudenko, D. (2007). Tuning the performance of the MMAS heuristic. *Engineering stochastic local search algorithms. designing, implementing and analyzing effective heuristics* (pp. 46-60) Springer.

Smit, S. K., & Eiben, A. E. (2009). Comparing parameter tuning methods for evolutionary algorithms. Paper presented at the *2009 IEEE Congress on Evolutionary Computation*, 399-406.

Smit, S. K., & Eiben, A. E. (2010). Beating the 'world champion'Evolutionary algorithm via REVAC tuning. Paper presented at the *IEEE Congress on Evolutionary Computation*, 1-8.

Sonderman, D., & Abrahamson, P. G. (1985). Radiotherapy treatment design using mathematical programming models. *Operations Research, 33*(4), 705-725.

Stützle, T., & López-Ibáñez, M. (2013). Automatic (offline) configuration of algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference 2013 (GECCO '13). Companion Material Proceedings*, New York, NY,. 893-918.

Synder, S., & ReVelle, C. (1996). The grid packing problem: Selecting a harvesting pattern in an area with forbidden regions. *Forest Science, 42*(1), 27-34.

Telford, J. K. (2007). A brief introduction to design of experiments. *Johns Hopkins APL Technical Digest, 27*(3), 224-232.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society.Series B (Methodological),* , 267-288.

Weintraub, A., & Murray, A. T. (2006). Review of combinatorial problems induced by spatial forest harvesting planning. *Discrete Applied Mathematics, 154*(5), 867-879.

Wu, C. J., & Hamada, M. S. (2011). *Experiments: Planning, analysis, and optimization* John Wiley & Sons.

Yuan, Z., De Oca, Marco A Montes, Birattari, M., & Stützle, T. (2012). Continuous optimization algorithms for tuning real and integer parameters of swarm intelligence algorithms. *Swarm Intelligence, 6*(1), 49-75.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association, 101*(476), 1418-1429.

Zwaneveld, P. J., Kroon, L. G., & Van Hoesel, S. P. (2001). Routing trains through a railway station based on a node packing model. *European Journal of Operational Research, 128*(1), 14-33.

Chapter 7 Appendices

Appendix A - Design for Limited Experiment

Here is the D-optimal design used for the limited 6-factor experiment complete on CPLEX. There are 134 design points.

Table 7-1 contains the D-optimal design with 134 design points. It was used for the limited 6-factor experiment on CPLEX.

MIPEMPHASIS	NODESEL	VARSEL	DIVETYPE	FRACCUTS	MIRCUTS
0	0	-1	0	2	-1
0	0	-1	1	-1	1
0	0	0	2	2	1
0	0	0	3	-1	-1
0	0	1	3	-1	2
0	0	2	2	-1	2
0	0	2	3	2	-1
0	0	3	0	-1	-1
0	0	4	1	2	2
0	1	-1	2	2	2
0	1	-1	3	-1	-1
0	1	0	3	2	2
0	1	1	0	2	2
0	1	2	1	-1	2
0	1	3	1	2	-1
0	1	3	2	-1	2
0	1	4	2	1	-1
0	2	-1	1	1	2
0	2	0	0	-1	-1
0	2	1	2	1	-1
0	2	2	0	2	2
0	2	3	3	2	1

MIPEMPHASIS	NODESEL	VARSEL	DIVETYPE	FRACCUTS	MIRCUTS
0	2	4	1	-1	-1
0	3	-1	3	2	2
0	3	0	1	2	2
0	3	2	1	-1	-1
0	3	2	2	2	-1
0	3	3	2	-1	-1
0	3	4	0	-1	2
0	3	4	3	2	-1
1	0	-1	1	2	-1
1	0	0	0	2	-1
1	0	1	2	-1	-1
1	0	1	2	2	2
1	0	2	1	-1	2
1	0	4	3	2	1
1	1	-1	3	-1	2
1	1	0	1	-1	-1
1	1	1	3	-1	-1
1	1	2	0	-1	-1
1	1	2	2	2	1
1	1	3	3	2	-1
1	1	4	0	2	2
1	2	-1	0	1	-1
1	2	0	2	-1	2
1	2	1	0	-1	2
1	2	2	3	-1	-1
1	2	3	1	2	2
1	2	4	2	-1	2
1	2	4	2	2	-1
1	3	-1	2	-1	2
1	3	0	3	2	-1
1	3	1	1	2	1
1	3	2	0	2	2
1	3	3	0	2	-1
1	3	3	3	-1	2
1	3	4	1	1	-1
2	0	-1	0	2	2
2	0	0	1	1	-1
2	0	1	3	2	-1
2	0	2	3	-1	2
2	0	3	1	2	-1
2	0	4	0	2	-1
2	0	4	2	-1	2
2	1	-1	0	-1	-1

MIPEMPHASIS	NODESEL	VARSEL	DIVETYPE	FRACCUTS	MIRCUTS
2	1	0	0	1	2
2	1	0	2	-1	-1
2	1	1	1	-1	2
2	1	2	2	2	2
2	1	4	3	-1	-1
2	2	-1	1	-1	-1
2	2	0	3	2	2
2	2	1	2	2	2
2	2	2	1	2	-1
2	2	2	2	-1	-1
2	2	3	2	-1	2
2	3	-1	2	2	-1
2	3	0	3	-1	1
2	3	1	0	-1	-1
2	3	3	0	-1	2
2	3	4	1	2	2
3	0	-1	2	1	1
3	0	0	3	2	2
3	0	1	0	-1	1
3	0	1	1	2	-1
3	0	2	0	2	-1
3	0	3	2	-1	2
3	0	4	1	-1	-1
3	1	-1	3	2	-1
3	1	0	2	1	2
3	1	1	2	2	-1
3	1	2	1	-1	-1
3	1	3	0	2	2
3	1	4	0	-1	2
3	2	-1	0	-1	2
3	2	0	1	-1	2
3	2	0	3	2	-1
3	2	1	0	2	-1
3	2	1	3	-1	2
3	2	2	2	2	2
3	2	3	1	-1	-1
3	2	4	3	2	2
3	3	-1	1	2	2
3	3	0	0	-1	-1
3	3	1	0	2	2
3	3	2	3	-1	2
3	3	3	3	2	-1
3	3	4	2	-1	-1

MIPEMPHASIS	NODESEL	VARSEL	DIVETYPE	FRACCUTS	MIRCUTS
4	0	-1	3	-1	-1
4	0	0	0	-1	2
4	0	1	0	2	-1
4	0	2	1	2	2
4	0	3	3	1	2
4	0	4	2	2	2
4	1	-1	1	-1	-1
4	1	0	3	-1	-1
4	1	1	3	2	2
4	1	2	0	2	-1
4	1	3	2	-1	-1
4	1	4	1	2	1
4	2	-1	2	-1	-1
4	2	-1	3	2	2
4	2	0	2	2	2
4	2	1	1	-1	-1
4	2	2	0	-1	1
4	2	3	0	2	-1
4	2	4	0	-1	-1
4	3	-1	0	2	1
4	3	0	1	2	-1
4	3	1	2	-1	2
4	3	2	3	2	-1
4	3	3	1	-1	2
4	3	3	2	2	2
4	3	4	3	-1	2

Appendix B – Parameter Estimates for the Models Utilizing the Screening by Grouping Technique

Group screening parameter estimates, summary of fit, and analysis of variance for each model is listed here. Forward selection was used.

Table 7-2 contains the parameter estimates for the telecommunication network class using grouping 1. This is the geometric mean model.

Parameter Estimates - Telecommunication Networks - Group1 - Geometric Mean of Primal Integral					
	Term	Estimate	Std Error	t Ratio	Prob>|t|
1	Intercept	3.573059977	0.018727662	190.79	<.0001
2	Fpheur	0.112260952	0.018727662	5.99	<.0001
3	Heurfreq	0.125713509	0.018727662	6.71	<.0001
4	Predual	0.16248954	0.018727662	8.68	<.0001
5	Preind	0.040317708	0.018727662	2.15	0.0363
6	Prelinear	0.242137513	0.018727662	12.93	<.0001
7	Prepass	0.224299258	0.018727662	11.98	<.0001
8	Symmetry	-0.00921796	0.018727662	-0.49	0.6248
9	Fpheur*Predual	-0.032856466	0.018727662	-1.75	0.0856
10	Fpheur*Symmetry	0.053314068	0.018727662	2.85	0.0064
11	Heurfreq*Predual	-0.12964871	0.018727662	-6.92	<.0001
12	Preind*Prepass	0.046739634	0.018727662	2.5	0.016
13	Preind*Symmetry	-0.036033196	0.018727662	-1.92	0.0602
14	Prelinear*Prepass	0.258395906	0.018727662	13.8	<.0001
15	Prelinear*Symmetry	-0.032851931	0.018727662	-1.75	0.0856

Table 7-3 is the analysis of variance for the geometric mean model in Table 7.2.

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Ratio	Prob > F
Model	14	16.4814	1.17724	52.4468	<.0001
Error	49	1.099875	0.02245		
C. Total	63	17.581275			

Table 7-4 is the summary of fit for the geometric mean model in table 7.2.

Summary of Fit	
RSquare	0.937440567
RSquare Adj	0.919566444
Root Mean Square Error	0.149821295
Mean of Response	3.573059977
Observations (or Sum Wgts)	64

Table 7-5 contains the parameter estimates for the telecommunication network class using grouping 1. This is the variance model.

Parameter Estimates - Telecommunication Networks - Group1 - Geometric Variance				
Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	3.079949872	0.045873719	67.14	<.0001
Fpheur	-0.249518331	0.045873719	-5.44	<.0001
Heurfreq	-0.342530862	0.045873719	-7.47	<.0001
Predual	-0.5131969	0.045873719	-11.19	<.0001
Preind	-0.145278927	0.045873719	-3.17	0.0027
Prelinear	-0.165872322	0.045873719	-3.62	0.0007
Prepass	-0.15025546	0.045873719	-3.28	0.002
Symmetry	0.050000355	0.045873719	1.09	0.2812
Fpheur*Predual	0.141848463	0.045873719	3.09	0.0033
Fpheur*Prepass	0.114511701	0.045873719	2.5	0.016
Fpheur*Symmetry	-0.177918435	0.045873719	-3.88	0.0003
Heurfreq*Predual	0.362789191	0.045873719	7.91	<.0001

Predual*Symmetry	-0.078829432	0.045873719	-1.72	0.0922
Preind*Prelinear	-0.117485923	0.045873719	-2.56	0.0136
Preind*Prepass	-0.187327919	0.045873719	-4.08	0.0002
Prelinear*Prepass	-0.171915815	0.045873719	-3.75	0.0005

Table 7-6 The analysis of variance for the geometric variance model in table 0-5.

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Ratio	Prob > F
Model	15	51.060664	3.40404	25.2748	<.0001
Error	48	6.464711	0.13468		
C. Total	63	57.525375			

Table 7-7 This is the summary of fit for the geometric variance model in table 0-5.

Summary of Fit	
RSquare	0.887619838
RSquare Adj	0.852501038
Root Mean Square Error	0.366989753
Mean of Response	3.079949872
Observations (or Sum Wgts)	64

Table 7-8 Coding Theory - Screening Group 1- then Sequential Screening- Geometric Mean of Primal Integral

Summary of Fit

RSquare	0.907978
RSquare Adj	0.889748
Root Mean Square Error	46.20597
Mean of Response	324.8872
Observations (or Sum Wgts)	128

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	21	2232997.7	106333	49.8050
Error	106	226309.1	2135	Prob > F
C. Total	127	2459306.9		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	324.88723	4.08407	79.55	<.0001*
Cutsfactor	-45.20216	4.08407	-11.07	<.0001*
Eachcutlim	-7.78411	4.08407	-1.91	0.0594
Fraccand	3.7783715	4.08407	0.93	0.3570
Fracpass	-4.518807	4.08407	-1.11	0.2710
Prelim	-14.32087	4.08407	-3.51	0.0007*
Reinv	-30.82625	4.08407	-7.55	<.0001*
Strongitlim	-109.5657	4.08407	-26.83	<.0001*
Cutsfactor*Eachcutlim	4.7641544	4.08407	1.17	0.2460
Cutsfactor*Fracpass	-7.161743	4.08407	-1.75	0.0824
Cutsfactor*Prelim	-9.711022	4.08407	-2.38	0.0192*
Cutsfactor*Reinv	-16.60382	4.08407	-4.07	<.0001*
Cutsfactor*Strongitlim	29.636819	4.08407	7.26	<.0001*
Eachcutlim*Fracpass	-4.56209	4.08407	-1.12	0.2665
Eachcutlim*Prelim	-7.902405	4.08407	-1.93	0.0557
Eachcutlim*Strongitlim	-7.676175	4.08407	-1.88	0.0629
Fraccand*Fracpass	-17.2661	4.08407	-4.23	<.0001*
Fraccand*Strongitlim	9.1094005	4.08407	2.23	0.0278*
Fracpass*Prelim	-4.781881	4.08407	-1.17	0.2443
Fracpass*Reinv	4.7251709	4.08407	1.16	0.2499
Prelim*Reinv	5.4377729	4.08407	1.33	0.1859
Reinv*Strongitlim	15.047018	4.08407	3.68	0.0004*

Table 7-9 Coding Theory - Screening Group 1- then Sequential Screening- Geometric Variance Model

Summary of Fit

RSquare	0.742805
RSquare Adj	0.676596
Root Mean Square Error	1.099708
Mean of Response	3.311892
Observations (or Sum Wgts)	128

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	26	352.76743	13.5680	11.2192
Error	101	122.14514	1.2094	Prob > F
C. Total	127	474.91257		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	3.311892	0.097201	34.07	<.0001*
Cutsfactor	0.9792819	0.097201	10.07	<.0001*
Eachcutlim	0.1646549	0.097201	1.69	0.0934
Fraccand	-0.13003	0.097201	-1.34	0.1840
Fracpass	0.1700916	0.097201	1.75	0.0832
Prelim	0.2601775	0.097201	2.68	0.0087*
Reinv	0.1801158	0.097201	1.85	0.0668
Singlim	-0.182476	0.097201	-1.88	0.0634
Strongitlim	0.798719	0.097201	8.22	<.0001*
Cutsfactor*Fracpass	0.189533	0.097201	1.95	0.0540
Cutsfactor*Prelim	0.2552828	0.097201	2.63	0.0100*
Cutsfactor*Reinv	0.1734113	0.097201	1.78	0.0774
Cutsfactor*Singlim	-0.172577	0.097201	-1.78	0.0788
Cutsfactor*Strongitlim	-0.260952	0.097201	-2.68	0.0085*
Eachcutlim*Fracpass	0.189357	0.097201	1.95	0.0542
Eachcutlim*Prelim	0.167845	0.097201	1.73	0.0873
Eachcutlim*Singlim	-0.266346	0.097201	-2.74	0.0073*
Fraccand*Prelim	-0.183995	0.097201	-1.89	0.0612
Fraccand*Reinv	0.2363853	0.097201	2.43	0.0168*
Fraccand*Singlim	0.16602	0.097201	1.71	0.0907
Fraccand*Strongitlim	-0.29115	0.097201	-3.00	0.0034*
Fracpass*Prelim	0.1814631	0.097201	1.87	0.0648
Fracpass*Singlim	-0.268591	0.097201	-2.76	0.0068*
Fracpass*Strongitlim	0.1484612	0.097201	1.53	0.1298
Prelim*Reinv	-0.224962	0.097201	-2.31	0.0227*
Prelim*Singlim	-0.241011	0.097201	-2.48	0.0148*
Reinv*Strongitlim	-0.388316	0.097201	-3.99	0.0001*

Table 7-10 Coding Theory - Screening Group 2- then Sequential Screening – Geometric Mean of Primal Integral

Summary of Fit

RSquare	0.96365
RSquare Adj	0.955611
Root Mean Square Error	31.98077
Mean of Response	303.3487
Observations (or Sum Wgts)	128

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	23	2819858.8	122603	119.8731
Error	104	106368.1	1023	Prob > F
C. Total	127	2926226.9		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	303.34868	2.826728	107.31	<.0001*
Cliques	-19.12167	2.826728	-6.76	<.0001*
Cutpass	-8.604633	2.826728	-3.04	0.0030*
Cutsfactor	-43.9497	2.826728	-15.55	<.0001*
Prelim	-6.646347	2.826728	-2.35	0.0206*
Reinv	11.657679	2.826728	4.12	<.0001*
Singlim	-20.59018	2.826728	-7.28	<.0001*
Rinsheur	-5.932409	2.826728	-2.10	0.0383*
Preind	-122.8072	2.826728	-43.45	<.0001*
Prelinear	-7.156553	2.826728	-2.53	0.0128*
Prepass	-4.998441	2.826728	-1.77	0.0799
Symmetry	-7.612993	2.826728	-2.69	0.0083*
Cliques*Cutpass	5.2086704	2.826728	1.84	0.0682
Cliques*Cutsfactor	-15.75049	2.826728	-5.57	<.0001*
Cliques*Preind	22.784593	2.826728	8.06	<.0001*
Cliques*Prelinear	-6.541277	2.826728	-2.31	0.0226*
Cutpass*Cutsfactor	-13.37039	2.826728	-4.73	<.0001*
Cutsfactor*Reinv	-6.763956	2.826728	-2.39	0.0185*
Cutsfactor*Preind	33.251487	2.826728	11.76	<.0001*
Prelim*Singlim	27.666323	2.826728	9.79	<.0001*
Reinv*Singlim	-12.86996	2.826728	-4.55	<.0001*
Reinv*Preind	21.227256	2.826728	7.51	<.0001*
Singlim*Preind	-10.28571	2.826728	-3.64	0.0004*
Preind*Symmetry	-12.21212	2.826728	-4.32	<.0001*

Table 7-11 Coding Theory - Screening Group 2- then Sequential Screening – Geometric Variance Model

Summary of Fit

RSquare	0.637267
RSquare Adj	0.577367
Root Mean Square Error	1.239232
Mean of Response	3.164018
Observations (or Sum Wgts)	128

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	18	294.08097	16.3378	10.6387
Error	109	167.39084	1.5357	Prob > F
C. Total	127	461.47181		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	3.1640179	0.109534	28.89	<.0001*
Cliques	0.4458194	0.109534	4.07	<.0001*
Cutsfactor	0.8078345	0.109534	7.38	<.0001*
Singlim	0.1282061	0.109534	1.17	0.2444
Strongitlim	0.1487084	0.109534	1.36	0.1774
Mipsearch	0.2500826	0.109534	2.28	0.0244*
Rinsheur	0.1501973	0.109534	1.37	0.1731
Predual	0.1348126	0.109534	1.23	0.2211
Preind	0.6130773	0.109534	5.60	<.0001*
Prelinear	-0.186905	0.109534	-1.71	0.0908
Cliques*Cutsfactor	0.4161762	0.109534	3.80	0.0002*
Cliques*Predual	0.2232088	0.109534	2.04	0.0440*
Cliques*Preind	-0.498613	0.109534	-4.55	<.0001*
Cutsfactor*Preind	-0.43698	0.109534	-3.99	0.0001*
Cutsfactor*Prelinear	-0.160247	0.109534	-1.46	0.1463
Singlim*Rinsheur	0.2533924	0.109534	2.31	0.0226*
Singlim*Prelinear	-0.256351	0.109534	-2.34	0.0211*
Mipsearch*Rinsheur	0.164467	0.109534	1.50	0.1361
Mipsearch*Preind	0.2197481	0.109534	2.01	0.0473*

Table 7-12 SVM - Screening Group 1– Geometric Mean of Primal Integral

Summary of Fit

RSquare	0.643213
RSquare Adj	0.513472
Root Mean Square Error	14.61333
Mean of Response	48.12592
Observations (or Sum Wgts)	16

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	4	4234.8368	1058.71	4.9577
Error	11	2349.0431	213.55	Prob > F
C. Total	15	6583.8798		0.0157*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	48.12592	3.653332	13.17	<.0001*
Disjcuts	3.5627622	3.653332	0.98	0.3504
Fraccand	8.7941391	3.653332	2.41	0.0348*
Fracpass	9.1255929	3.653332	2.50	0.0296*
Disjcuts*Fraccand	-9.558802	3.653332	-2.62	0.0240*

Table 7-13 SVM - Screening Group 1– Geometric Variance Model

Summary of Fit

RSquare	0.812974
RSquare Adj	0.744965
Root Mean Square Error	137.6991
Mean of Response	409.0734
Observations (or Sum Wgts)	16

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	4	906630.7	226658	11.9539
Error	11	208571.6	18961	Prob > F
C. Total	15	1115202.3		0.0005*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	409.07339	34.42479	11.88	<.0001*
Disjcuts	-40.3249	34.42479	-1.17	0.2662
Fraccand	-146.8083	34.42479	-4.26	0.0013*
Fracpass	-81.54168	34.42479	-2.37	0.0372*
Disjcuts*Fraccand	163.81882	34.42479	4.76	0.0006*

Table 7-14 SVM - Screening Group 2– Geometric Mean of Primal Integral

Summary of Fit

RSquare	0.003935
RSquare Adj	-0.10279
Root Mean Square Error	16.40097
Mean of Response	33.14623
Observations (or Sum Wgts)	32

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	3	29.7549	9.918	0.0369
Error	28	7531.7719	268.992	Prob > F
C. Total	31	7561.5268		0.9903

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	33.146233	2.899309	11.43	<.0001*
Fpheur	-0.866172	2.899309	-0.30	0.7673
Heurfreq	-0.423147	2.899309	-0.15	0.8850
Fpheur*Heurfreq	-0.023093	2.899309	-0.01	0.9937

Table 7-15 SVM - Screening Group 2– Geometric Variance Model

Summary of Fit

RSquare	0.378036
RSquare Adj	0.258428
Root Mean Square Error	254.3624
Mean of Response	460.8488
Observations (or Sum Wgts)	32

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	5	1022462.6	204493	3.1606
Error	26	1682206.5	64700	Prob > F
C. Total	31	2704669.2		0.0232*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	460.84882	44.96535	10.25	<.0001*
Implbd	80.927029	44.96535	1.80	0.0835
Zerohalfcuts	-79.87085	44.96535	-1.78	0.0874
Bndstreng	-9.194676	44.96535	-0.20	0.8396
Implbd*Bndstreng	75.732845	44.96535	1.68	0.1041
Zerohalfcuts*Bndstreng	114.90611	44.96535	2.56	0.0168*

Appendix C Screening – Marginal Analysis Parameter Estimates

This appendix contains parameter estimates, and model summary output from JMP, telecommunication network class. The design used was *PB60EMEMA2f4runsres6* using double adaptive LASSO with the criterion ERIC (Using AICc for the criterion caused all variables to be removed from the model.)

Table 7-16 Marginal Analysis Screening– Geometric Mean Model

Model Summary

Response	geoMean
Distribution	Normal
Estimation Method	Adaptive Double Lasso
Validation Method	ERIC
Mean Model Link	Identity
Scale Model Link	Identity

Measure	
Number of rows	5
Sum of Frequencies	4
-LogLikelihood	1.7248443
Number of Parameters	4
BIC	8.994866
AICc	.
ERIC	4.2341842
RSquare	0.9905975

Parameter Estimates for Original Predictors

Term	Estimate	Std Error	Wald ChiSquare	Prob > ChiSquare	Lower 95%	Upper 95%
Intercept	10.843162	0.186211	3390.7911	<.0001*	10.478195	11.208129
Scaind	0.6144191	0.2028609	9.1734537	0.0025*	0.2168191	1.0120191
Rinsheur(1,100)	-3.723838	0.187097	396.13932	<.0001*	-4.090541	-3.357134
Scaind*Rinsheur	0	0	0	1.0000	0	0
Scale	0.3724221	0.0445414	69.910753	<.0001*	0.2851226	0.4597215

Table 7-17 Telecommunications Network - Marginal Analysis Screening– Geometric Variance Model

Model Summary

Response	geoVar
Distribution	Normal
Estimation Method	Adaptive Double Lasso
Validation Method	ERIC
Mean Model Link	Identity
Scale Model Link	Identity

Measure	
Number of rows	5
Sum of Frequencies	4
-LogLikelihood	6.3204278
Number of Parameters	5
BIC	19.572327
AICc	.
ERIC	5.4178031
RSquare	0.9995564

Parameter Estimates for Original Predictors

Term	Estimate	Std Error	Wald ChiSquare	Prob > ChiSquare	Lower 95%	Upper 95%
Intercept	91.567752	0.5874414	24297.186	<.0001*	90.416388	92.719116
Scaind	20.676131	0.5874414	1238.8235	<.0001*	19.524767	21.827495
Rinsheur(1,100)	17.940767	0.5874414	932.72348	<.0001*	16.789403	19.092131
Scaind*Rinsheur	-47.63733	0.5874414	6576.0584	<.0001*	-48.78869	-46.48596
Scale	1.1748828	0	.	.	1.1748828	1.1748828

Appendix D- Parameter Estimates for the Plackett-Burman designs with Sequential Screening Using Adaptive Double LASSO

This appendix contains the parameter estimates, standard error, Wald ChiSquare statistic and the p-values for the models obtained using Plackett-Burman resolution III design with 60 runs, and a folded Plackett-Burman resolution IV design with 120 runs.

In the tables 9.1-9.6 the parameter estimates for the mean and variance first ordered models with interactions is given for all three classes of instances in which the first screening design was the Plackett-Burman resolution III design with 60 runs. All of the model estimates in tables 9.1 – 9.6 are the result of sequentially screening using adaptive double LASSO.

Table 7-18 contains the parameter estimates for the geometric mean of the primal integral model for the telecommunication network class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table.

Telecommunications Network – Mean Factorial 2^{13-6}		Designs: PB60 & Fractional	
Term	Parameter Estimate Geometric Mean Model	Wald ChiSquare	Prob > ChiSquare
Intercept	44.53749296	1179.101865	<.0001
Cutsfactor	-8.06672218	176.223916	<.0001
Eachcutlim	-5.420567664	79.57198425	<.0001
dpriind	18.16057169	82.38558474	<.0001
Perlim	2.927594867	23.21093322	<.0001
Subalg	-4.005810969	7.856807185	0.0051
Rinsheur	-20.7463779	407.5649896	<.0001
Predual	-3.824793359	9.904364388	0.0016
Cliques*Cutsfactor	1.744457258	8.24121632	0.0041
Cliques*Perlim	5.345085898	77.3713235	<.0001
Cutsfactor*Eachcutlim	-6.602229711	118.0461722	<.0001

Cutsfactor*Rinsheur	5.96932168	96.49851147	<.0001
Eachcutlim*Rinsheur	5.197929102	73.16971291	<.0001
dpriind*Subalg	-12.70367597	27.31552052	<.0001
dpriind*Rinsheur	-4.242652453	12.18668458	0.0005
Subalg*Rinsheur	5.561444953	20.94044674	<.0001

Table 7-19 contains the parameter estimates for the geometric variance of the primal integral model for the telecommunication network class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table.

Telecommunications Network - Variance Fractional Factorial 2^{13-6}			PB60 &
Term	Parameter Estimate Geometric Variance Model	Wald ChiSquare	Prob > ChiSquare
Intercept	150.4979667	1430.074869	<.0001
Cliques	-10.00347624	18.42355337	<.0001
Cutsfactor	11.97856795	11.39501263	0.0007
Eachcutlim	-11.21987279	23.17647444	<.0001
dpriind	41.04689116	33.45072671	<.0001
Perlim	9.194889711	5.952842835	0.0147
Reinv	7.858573914	4.348293521	0.037
Subalg	-32.04061216	31.87972878	<.0001
Rinsheur	46.30636449	160.6123813	<.0001
Cliques*Perlim	-21.92135171	88.47206555	<.0001
Cliques*Reinv	-6.384210914	7.503881366	0.0062
Cliques*Rinsheur	-4.140635367	3.156497731	0.0756
Cutsfactor*Eachcutlim	-10.21048646	19.19395067	<.0001
Cutsfactor*Reinv	5.983068102	6.590515703	0.0103
Cutsfactor*Subalg	-18.3152873	15.43970458	<.0001
Cutsfactor*Rinsheur	-14.23460173	37.30457939	<.0001
Eachcutlim*Rinsheur	-20.72837716	79.10466785	<.0001
Flowpaths*dpriind	8.85327875	5.731306937	0.0167
dpriind*Perlim	7.197934734	2.384664708	0.1225
dpriind*Reinv	18.25591683	15.33976874	<.0001
dpriind*Subalg	-31.86600972	11.68441282	0.0006
dpriind*Rinsheur	20.27142867	18.91386264	<.0001
Perlim*Rinsheur	5.903097227	6.415512853	0.0113
Perlim*Predual	-13.01316414	7.794303829	0.0052
Reinv*Rinsheur	7.832119242	11.29353985	0.0008

Reinv*Predual	6.931968641	2.21169224	0.137
Subalg*Rinsheur	-22.44276608	23.18271419	<.0001

Table 7-20 contains the parameter estimates for the geometric mean of the primal integral model for the coding theory class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table.

Coding Theory Model - Mean Designs: PB60 & Fractional Factorial 2^{9-2}			
Term	Parameter Estimate Geometric Mean Model	Wald ChiSquare	Prob > ChiSquare
Intercept	159.952355	381.1559705	<.0001
Cutsfactor	-14.9081441	5.145718025	0.0233
Eachcutlim	-15.87393655	5.83402251	0.0157
Reinv	-35.15842644	51.67741078	<.0001
Preind(int)	286.3231226	873.0036932	<.0001
Reduce	291.6722729	889.1441751	<.0001
Cutsfactor*Eachcutlim	-40.98990307	174.4775619	<.0001
Cutsfactor*Reinv	-14.51993168	21.89348747	<.0001
Cutsfactor*Preind(int)	-24.51599433	15.60360092	<.0001
Cutsfactor*Reduce	-28.31690711	20.81696644	<.0001
Eachcutlim*Reinv	-14.69373073	22.42074064	<.0001
Eachcutlim*Preind(int)	-22.7669323	13.45658447	0.0002
Eachcutlim*Reduce	-27.04154345	18.98404534	<.0001
Reinv*Preind(int)	-6.190562516	0.994915134	0.3185
Preind(int)*Reduce	-285.1859518	527.8640376	<.0001

Table 7-21 contains the parameter estimates for the geometric variance of the primal integral model for the coding theory class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table.

Coding Theory Model - Variance Designs: PB60 & Fractional Factorial 2^{9-2}			
Term	Parameter Estimate Geometric Variance Model	Wald ChiSquare	Prob > ChiSquare
Intercept	5.505516906	1300.406679	<.0001
Cutsfactor	0.45198993	13.79339533	0.0002
Eachcutlim	0.393926602	10.47717709	0.0012
Reinv	0.030403609	0.109584512	0.7406
Preind(int)	-3.161517281	293.0332101	<.0001
Reduce	-3.228406281	308.8981202	<.0001
Cutsfactor*Eachcutlim	0.761460711	183.6773314	<.0001
Cutsfactor*Reinv	0.210932773	14.09446797	0.0002
Cutsfactor*Preind(int)	0.346708109	9.519816325	0.002
Cutsfactor*Reduce	0.322641734	8.244069736	0.0041
Eachcutlim*Reinv	0.175484008	9.755188148	0.0018
Eachcutlim*Preind(int)	0.377623453	11.29323819	0.0008
Eachcutlim*Reduce	0.356954266	10.09080287	0.0015
Reinv*Preind(int)	0.262520766	5.457929153	0.0195
Preind(int)*Reduce	3.089420781	188.9708016	<.0001

Table 7-22 contains the parameter estimates for the geometric mean of the primal integral model for the SVM class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table.

SVM - Mean PB60 & Full Factorial 2^6			
Term	Parameter Estimate Geometric Mean Model	Wald ChiSquare	Prob > ChiSquare
Intercept	390.8391285	8385.31477	<.0001
Nodesel	92.133658	121.7895711	<.0001
Cutpass	-66.24511075	251.8495184	<.0001
Cutsfactor	-85.65146834	421.020351	<.0001
Rinsheur	-70.78205897	275.0234214	<.0001
Nodesel*Rinsheur	32.53316038	15.18539106	<.0001
Cutpass*Cutsfactor	-66.26513856	252.0018242	<.0001
Cutpass*Rinsheur	-26.35195644	39.85283229	<.0001
Cutsfactor*Rinsheur	-30.00598834	51.67129383	<.0001

Table 7-23 contains the parameter estimates for the geometric variance of the primal integral model for the SVM class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table.

SVM - Variance			PB60 & Full Factorial 2^6
Term	Parameter Estimate Geometric Variance Model	Wald ChiSquare	Prob > ChiSquare
Intercept	33.02791281	49.36511463	<.0001
Cutpass	31.59218103	45.16656986	<.0001
Cutsfactor	31.73316125	45.57058125	<.0001
Rinsheur	31.42756575	44.69710326	<.0001
Cutpass*Cutsfactor	31.59246659	45.16738638	<.0001
Cutpass*Rinsheur	31.18689428	44.01514603	<.0001
Cutsfactor*Rinsheur	31.26319319	44.23077607	<.0001

Tables 24 -contain the parameter estimates of the best geometric mean and variance models with folded Plackett-Burman resolution IV designs having 120 runs for each of the three classes of instances.

Table 7-24 contains the parameter estimates for the geometric mean of the primal integral model for the telecommunications network class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table. The original screening file was a folded Plackett-Burman design with 120 runs and the follow-up design was a fractional factorial 2^{8-1}.

Telecommunications Network - Mean			PB120 & Fractional Factorial 2^{8-1}
Term	Parameter Estimate Geometric Mean Model	Wald ChiSquare	Prob > ChiSquare
Intercept	21.12626009	432.4245998	<.0001
Varsel	15.36184359	132.9747319	<.0001
Cutsfactor	-11.08427083	170.6534721	<.0001
Scaind	-7.255326094	73.11639074	<.0001
Fpheur	4.175786781	9.159451343	0.0025
Rinsheur	-6.735393797	63.01252123	<.0001
Varsel*Cutsfactor	-10.78245925	142.5616956	<.0001

Varsel*Scaind	-3.357226969	13.82065441	0.0002
Varsel*Fpheur	7.129225188	15.58087615	<.0001
Varsel*Rinsheur	-6.942560375	59.10259253	<.0001
Cutsfactor*Scaind	1.896261344	17.63695375	<.0001
Cutsfactor*Fpheur	-1.662429	3.38885833	0.0656
Cutsfactor*Rinsheur	7.558616641	280.228196	<.0001
Scaind*Fpheur	8.251581719	83.49137457	<.0001
Scaind*Rinsheur	1.869787125	17.14792291	<.0001
Fpheur*Rinsheur	-3.817109938	17.86637881	<.0001

Table 7-25 The parameter estimates for the geometric variance of the primal integral model for the telecommunications network class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table. The original screening file was a folded Plackett-Burman design with 120 runs and the follow-up design was a fractional factorial 2^{8-1}.

Telecommunications Network - Variance		PB120 & Fractional Factorial 2^{8-1}	
Term	Parameter Estimate Geometric Variance Model	Wald ChiSquare	Prob > ChiSquare
Intercept	386.1984096	655.0793436	<.0001
Varse	-84.12923978	26.00337152	<.0001
Cutsfactor	88.80866442	44.00765223	<.0001
Scaind	206.8747544	238.7990053	<.0001
Fpheur	-202.3862658	150.4865919	<.0001
Cutsfactor*Scaind	35.3352757	18.3490123	<.0001
Cutsfactor*Fpheur	-99.61627425	36.45829219	<.0001
Cutsfactor*Rinsheur	-48.80389203	35.00294927	<.0001
Scaind*Fpheur	-213.6677579	167.7311473	<.0001
Scaind*Rinsheur	-29.88510491	13.12518074	0.0003
Fpheur*Rinsheur	55.03730261	32.58268509	<.0001

Table 7-26 contains the parameter estimates for the geometric mean of the primal integral model for the coding theory class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table. The original screening file was a folded Plackett-Burman design with 120 runs and the follow-up design was a fractional factorial 2^6.

Coding Theory - Mean Factorial 2^6			PB120 & Full
Term	Parameter Estimate Geometric Mean Model	Wald ChiSquare	Prob > ChiSquare
Intercept	155.3585191	269.9722565	<.0001
Cliques	-3.266308234	0.177925865	0.6732
Cutsfactor	-13.37826089	2.984860748	0.084
Eachcutlim	-12.98937892	2.813853776	0.0935
Reinv	-32.8287088	17.9734876	<.0001
Preind	300.463174	653.6302046	<.0001
Reduce	309.5079306	676.1394682	<.0001
Cliques*Cutsfactor	-11.11477342	8.095295086	0.0044
Cliques*Eachcutlim	-10.83075002	7.686852016	0.0056
Cliques*Reinv	-3.939580891	1.0170247	0.3132
Cliques*Preind(int)	-7.981126656	1.043517822	0.307
Cliques*Reduce	-7.363186719	0.8881843	0.346
Cutsfactor*Eachcutlim	-30.54417836	61.13477954	<.0001
Cutsfactor*Reinv	-10.78300886	7.619235204	0.0058
Cutsfactor*Preind(int)	-18.00996297	5.313700895	0.0212
Cutsfactor*Reduce	-19.09404453	5.972653596	0.0145
Eachcutlim*Reinv	-10.15644945	6.759511001	0.0093
Eachcutlim*Preind(int)	-17.47447241	5.002413751	0.0253
Eachcutlim*Reduce	-18.40210872	5.547619286	0.0185
Reinv*Preind(int)	-0.541357406	0.004801088	0.9448
Reinv*Reduce	-1.430377344	0.033517584	0.8547
Preind(int)*Reduce	-298.6789524	365.3597593	<.0001

Table 7-27 contains the parameter estimates for the geometric variance of the primal integral model for the coding theory class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table. The original screening file was a folded Plackett-Burman design with 120 runs and the follow-up design was a fractional factorial 2^6

Coding Theory - Variance Factorial 2^6			PB120 & Full
Term	Parameter Estimate Geometric Variance Model	Wald ChiSquare	Prob > ChiSquare
Intercept	5.813794688	918.7057015	<.0001
Cliques	0.333345594	12.22099172	0.0005
Cutsfactor	0.55448775	33.8143407	<.0001
Eachcutlim	0.571412719	35.91011682	<.0001
Preind(int)	-3.768400313	189.0544517	<.0001
Reduce	-3.835066625	194.4570607	<.0001
Cliques*Cutsfactor	0.316382031	11.00881649	0.0009
Cliques*Eachcutlim	0.341904813	12.85663871	0.0003
Cutsfactor*Eachcutlim	0.586830156	37.87405967	<.0001
Preind(int)*Reduce	3.744758875	96.39283357	<.0001

Table 7-28 The parameter estimates for the geometric mean of the primal integral model for the SVM class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table. The original screening file was a folded Plackett-Burman design with 120 runs and the follow-up design was a fractional factorial 2^{10-3}.

SVM - Mean Factorial 2^{10-3}			PB120 & Full
Term	Parameter Estimate Geometric Mean Model	Wald ChiSquare	Prob > ChiSquare
Intercept	465.6419108	2049.548942	<.0001
Nodesel	65.75219328	26.81716683	<.0001
Mircuts	1.822672016	0.082426991	0.774
Cutsfactor	-6.488419391	1.044552382	0.3068
Eachcutlim	-7.195199516	1.284511743	0.2571
Fpheur	-25.42107216	3.245606633	0.0716
Heurfreq	2.550881727	0.085620577	0.7698
Rinsheur	-70.46759477	65.33958932	<.0001
Nodesel*Fpheur	37.42695122	3.789502766	0.0516
Nodesel*Heurfreq	-3.871198922	0.162167676	0.6872

Nodesel*Rinsheur	39.67375327	17.03255867	<.0001
Mircuts*Cutsfactor	-29.43921035	37.51333041	<.0001
Mircuts*Eachcutlim	-14.59728524	9.223112327	0.0024
Mircuts*Fpheur	-23.38891523	5.919615209	0.015
Mircuts*Rinsheur	3.068302211	0.407501841	0.5232
Cutsfactor*Eachcutlim	-17.10426609	12.663163	0.0004
Cutsfactor*Fpheur	-26.77679892	7.758729878	0.0053
Cutsfactor*Rinsheur	-1.928809164	0.161031868	0.6882
Eachcutlim*Fpheur	-16.85006727	3.072391786	0.0796
Eachcutlim*Heurfreq	2.332191164	0.23542989	0.6275
Eachcutlim*Rinsheur	-2.95725968	0.378540397	0.5384
Fpheur*Heurfreq	-4.626408109	0.231612076	0.6303
Fpheur*Rinsheur	15.3024052	2.533918913	0.1114
Heurfreq*Rinsheur	-10.78399307	5.033760042	0.0249

Table 7-29 The parameter estimates for the geometric variance of the primal integral model for the SVM class along with the standard error, Wald ChiSquare statistic and the p-values are also in the table. The original screening file was a folded Plackett-Burman design with 120 runs and the follow-up design was a fractional factorial 2^{10-3}.

SVM - Variance 2^{10-3}		PB120 & Fractional Factorial	
Term	Parameter Estimate Geometric Variance Model	Wald ChiSquare	Prob > ChiSquare
Intercept	8.080060484	6.449881252	0.0111
Rinsheur	3.048367212	6.1092322	0.0134

Appendix E CPLEX Parameters

This appendix contains a table that list all of the parameters tuned for the experiments. This information was compiled from the CPLEX parameter guide (IBM, 2017). The levels provide the possible setting for most of the parameters, however with the discrete and continuous parameters, some values were user selected. However, for the most up to date information, always refer to IBM ILOG CPLEX's website which can be found in the references of this paper.

Table 7-30 CPLEX parameter names, levels, default, identifier, and type.

#	Parameter	Levels	Default	Identifier	Type
1	*CPX_PARAM_MIPEMPHASIS*	[0,1,2,3,4]	0	2058	int
2	*CPX_PARAM_NODESEL*	[0,1,2,3]	1	2018	int
3	*CPX_PARAM_VARSEL*	[-1,0,1,2,3,4]	0	2028	int
4	*CPX_PARAM_DIVETYPE*	[0,1,2,3]	0	2060	int

#	Parameter	Levels	Default	Identifier	Type
5	*CPX_PARAM_FRACCUTS*	[-1,0,1,2]	0	2049	int
6	*CPX_PARAM_MIRCUTS*	[-1,0,1,2]	0	2052	int
7	*CPX_PARAM_AGGCUTLIM*	[3,10,100,1000, 10000]	3	2054	int
8	*CPX_PARAM_CLIQUES*	[-1,0,1,2]	0	2003	int
9	*CPX_PARAM_COVERS*	[-1,0,1,2]	0	2005	int
10	*CPX_PARAM_CUTPASS*	[-1, 0, 10, 100, 1000, 10000]	0	2056	int long
11	*CPX_PARAM_CUTSFACTOR*	[4,10]	4	2033	int

#	Parameter	Levels	Default	Identifier	Type
12	*CPX_PARAM_DISJCUTS*	[-1,0,1,2]	0	2053	int
13	*CPX_PARAM_EACHCUTLIM*	[0, 10, 100, 1000, 10000, 2100000000]	2100000000	2012	int
14	*CPX_PARAM_FLOWCOVERS*	[-1,0,1,2]	0	2040	int
15	*CPX_PARAM_FLOWPATHS*	[-1,0,1,2]	0	2051	int
16	*CPX_PARAM_FRACCAND*	[10, 100, 200, 1000, 10000]	200	2048	int
17	*CPX_PARAM_FRACPASS*	[0,10,100]	0	2050	int
18	*CPX_PARAM_GUBCOVERS*	[-1,0,1,2]	0	2044	int
19	*CPX_PARAM_IMPLBD*	[-1,0,1,2]	0	2041	int
20	*CPX_PARAM_ZEROHALFCUTS*	[-1,0,1,2]	0	2111	int

#	Parameter	Levels	Default	Identifier	Type
21	*CPX_PARAM_CRAIND*	[0,1]	1	1007	int
22	*CPX_PARAM_DPRIIND*	[0,1,2]	0	1009	int
23	*CPX_PARAM_PERIND* (int)	[0,1]	0	1027	int
24	*CPX_PARAM_PERLIM*	[0, 10, 100, 1000, 10000]	0	1028	int
25	*CPX_PARAM_PPRIIND*	[0,1,2]	0	1029	int
26	*CPX_PARAM_REINV*	[0, 10, 100, 1000]	0	1031	int
27	*CPX_PARAM_SIFTALG*	[0,1,2]	0	1077	int
28	*CPX_PARAM_SINGLIM*	[10,100,1000,10000]	10	1037	int
29	*CPX_PARAM_STRONGCANDLIM*	[10,100,1000,10000]	10	2045	int

#	Parameter	Levels	Default	Identifier	Type
30	*CPX_PARAM_STRONGITLIM*	[0,10,100,1000, 10000]	0	2046	int long
31	*CPX_PARAM_SCAIND*	[-1,0,1]	0	1034	int
32	*CPX_PARAM_BBINTERVAL*	[0, 1, 7, 10, 100, 1000, 10000]	7	2039	int long
33	*CPX_PARAM_BRDIR*	[-1,0,1]	0	2001	int
34	*CPX_PARAM_FPHEUR*	[-1,0,1,2]	0	2098	int
35	*CPX_PARAM_HEURFREQ*	[-1, 0, 10, 100, 1000, 10000]	0	2031	int
36	*CPX_PARAM_LBHEUR*(*int*)	[0,1]	0	2063	int
37	*CPX_PARAM_MIPSEARCH*	[0,1,2]	0	2109	int

#	Parameter	Levels	Default	Identifier	Type
38	*CPX_PARAM_SUBALG*	[0,1,2,5]	0	2026	int
39	*CPX_PARAM_PARALLELMODE*	[-1,0,1]	0	1109	int
40	*CPX_PARAM_PROBE*	[-1,0,1,2,3]	0	2042	int
41	*CPX_PARAM_RINSHEUR*	[-1, 0, 10, 100, 1000, 10000]	0	2061	int long
42	*CPX_PARAM_STARTALG*	[0,1,2,3,4,5,6]	0	2025	int
43	*CPX_PARAM_AGGFILL*	[10,100,1000,10000]	10	1002	int

#	Parameter	Levels	Default	Identifier	Type
44	*CPX_PARAM_AGGIND*	[-1, 0, 10 ,100, 1000, 10000]	-1	1003	int
45	*CPX_PARAM_BNDSTRENIND*	[-1,0,1]	-1	2029	int
46	*CPX_PARAM_COEREDIND*	[0,1,2]	2	2004	int
47	*CPX_PARAM_DEPIND*	[-1,0,1,2,3]	-1	1008	int
48	*CPX_PARAM_MIPCBREDLP*	[0,1]	1	2055	int
49	*CPX_PARAM_PROBETIME*	[0, 10 ,100, 1000 ,10000, 1E+75]	1.00E+75	2065	d

#	Parameter	Levels	Default	Identifier	Type
50	*CPX_PARAM_PREDUAL*	[-1,0,1]	0	1044	int
51	*CPX_PARAM_PREIND*(*int*)	[0,1]	1	1030	int
52	*CPX_PARAM_PRELINEAR*	[0,1]	1	1058	int
53	*CPX_PARAM_PREPASS*	[-1, 0, 10, 100, 1000, 10000]	-1	1052	int
54	*CPX_PARAM_PRESLVND*	[-1,0,1,2]	0	2037	int
55	*CPX_PARAM_REDUCE*	[0,1,2,3]	3	1057	int
56	*CPX_PARAM_RELAXPREIND*	[0,1]	-1	2034	int

#	Parameter	Levels	Default	Identifier	Type
57	*CPX_PARAM_REPEATPRESOLVE*	[-1,0,1,2,3]	-1	2064	int
58	*CPX_PARAM_SYMMETRY*	[-1,0,1,2,3,4,5]	-1	2059	int
59	*CPX_PARAM_THREADS*	[0,1,4]	0	1067	int
	Static Parameters				
	CPX_PARAM_TILIM	600		1039	double
	CPX_PARAM_EPRHS	0.000000001		1016	double
	CPX_PARAM_EPOPT	0.000000001		1014	double
	CPX_PARAM_EPMRK	0.99999		1013	double
	CPX_PARAM_EPINT	0		2010	double
	CPX_PARAM_EPGAP	0		2009	double
	CPX_PARAM_EPAGAP	0		2008	double
	CPX_PARAM_NUMERICALEMPHASIS	1		1083	int
	CPX_PARAM_SCRIND	0		1035	int
	CPX_PARAM_CLOCKTYPE	2		1006	int

Appendix F Gurobi Parameters

Appendix F contains table 7.31 that list all of the parameters tuned for the experiments. This information was compiled from the Gurobi website for parameter documentation.

Table 7-31 list Gurobi's parameters that are tuned for the experiment.

Gurobi Parameter Name	Parameter Values ***default value is in bold**					
BarIterLimit	0	**1000**	2.00E+09			
IterationLimit	0	1.00E+09	**2.00E+09**			
NodeLimit	0	1.00E+09	**2.00E+09**			
SolutionLimit	1	1.00E+09	**2.00E+09**			
NormAdjust	**-1**	0	1	2	3	
ObjScale	-1	**0**	2.00E+09			
PerturbValue	0	**0.0002**	0.01			

Quad	**-1**	1				
ScaleFlag	0	**1**	2			
Sifting	**-1**	0	1	2		
SiftMethod	**-1**	0	1	2		
SimplexPricing	**-1**	0	1	2	3	
BarCorrectors	**-1**	2000000000				
BarHomogeneous	**-1**	0	1			
BarOrder	**-1**	0	1			
Crossover	**-1**	0	1	2	3	4
CrossoverBasis	**0**	1				
BranchDir	**-1**	0	1			
Heuristics	0	**0.05**	1			
ImproveStartGap	**0.0**	2.00E+09				
ImproveStartNodes	0.0	**2.00E+09**				
ImproveStartTime	0.0	**2.00E+09**				
MinRelNodes	**-1**	2.00E+09				
MIPFocus	**0**	1	2	3		
NodeMethod	0	**1**	2			
PumpPasses	**-1**	2.00E+09				
RINS	**-1**	2.00E+09				

SubMIPNodes	0	**500**	2.00E+09			
Symmetry	**-1**	0	1	2		
VarBranch	**-1**	0	1	2	3	
ZeroObjNodes	**-1**	2.00E+09				
Cuts	**-1**	0	1	2		
CliqueCuts	**-1**	0	1	2		
CoverCuts	**-1**	0	1	2		
FlowCoverCuts	**-1**	0	1	2		
FlowPathCuts	**-1**	0	1	2		
GUBCoverCuts	**-1**	0	1	2		
ImpliedCuts	**-1**	0	1	2		
MIPSepCuts	**-1**	0	1	2		
MIRCuts	**-1**	0	1	2		
ModKCuts	**-1**	0	1	2		
NetworkCuts	**-1**	0	1	2		
SubMIPCuts	**-1**	0	1	2		
ZeroHalfCuts	**-1**	0	1	2		
CutAggPasses	**-1**	0	1	2		
CutPasses	**-1**	0	1	2		
GomoryPasses	**-1**	0	1	2		
AggFill	**-1**	2.00E+09				
Aggregate	0	**1**				
DualReductions	0	**1**				

FeasRelaxBigM	0	**1.00E+06**	2.00E+09			
IISMethod	**-1**	0	1	2	3	
Method	**-1**	0	1	2	3	4
PrePasses	**-1**	2.00E+09				
Presolve	**-1**	0	1	2		
Threads	**0**	1	4			
Parameters below were kept static.						
TimeLimit	600					
BarConvTol	1.00E-08					
FeasibilityTol	1.00E-09					
IntFeasTol	1.00E-05					
MarkowitzTol	0.999					
MIPGap	0.0					
MIPGapAbs	0.0					
OptimalityTol	1.00E-09					

Vita

Toni Pusateri Sorrell was born on July 14, 1967, in Pittsburgh, Pennsylvania, and is a United States of America citizen. She graduated from North Allegheny Senior High School, Wexford, Pennsylvania in 1985. She received her Bachelor of Science in Secondary Education with a Math Option from Pennsylvania State University, State College, Pennsylvania in 1989 and subsequently taught in private and public schools in the Richmond, Virginia area for 12 years. During that time, she was awarded a GTE G.I.F.T grant to develop and teach an integrated Algebra II and Biology curriculum to encourage students to pursue further education in the area of Science, Mathematics and Technology. She received a Master of Interdisciplinary Studies from Virginia Commonwealth University, Richmond, Virginia, in 2002. She continued teaching in Hanover County Public Schools and during this time earned National Board Certification in Adolescent and Young Adult Mathematics, and the Virginia Council of Teachers of Mathematics' William C. Lowery Award as Mathematics Educator of the Year Award. In 2006, she became a mathematics instructor at Virginia Commonwealth University and worked as such until she pursued her doctorate degree full-time in 2014.