2010

Automated Selected of Mixed Integer Program Solver Parameters

Charles Stewart
Virginia Commonwealth University

Automated Selection of Mixed Integer Program Solver Parameter Settings

Charles Robert Stewart

Thesis submitted to the Faculty of the

Virginia Commonwealth University

in partial fulfillment of the requirements for the degree of

Master of Sciences in Mathematical Sciences

Concentration in

Operations Research

J. Paul Brooks, Chair

David J. Edwards, Co-Chair

Jose H.Dula

April 2010

Richmond, Virginia

Keywords: parameter tuning, design of experiments, mixed integer programming, CPLEX

Automated Selection of Mixed Integer Program Solver Parameter Settings

Charles R. Stewart

(ABSTRACT)

This paper presents a method that uses designed experiments and statistical models to extract information about how solver parameter settings perform for classes of mixed integer programs. The use of experimental design facilitates fitting a model that describes the response surface across all combinations of parameter settings, even those not explicitly tested, allowing identification of both desirable and poor settings. Identifying parameter settings that give the best expected performance for a specific class of instances and a specific solver can be used to more efficiently solve a large set of similar instances, or to ensure solvers are being compared at their best.

Dedication

To my grandfathers, even if my undergraduate degree is a ‘trade-school’ degree.

Acknowledgments

I would like to thank my amazingly understanding and supportive wife, without whom I would not have gotten nearly this far. I also would like to thank my advisers, Dr. Brooks and Dr. Edwards, for the tremendous amount of time they spent with me helping to ensure my success.

Contents

List of Figures

List of Tables

Chapter 1

Introduction & Literature Review

Many important optimization problems can be described as a mixed integer program (MIP). MIPs are a set of algebraic expressions with several decision variables, and a linear objective function in terms of the decision variables. The objective function is maximized (or minimized) subject to linear constraints in terms of the decision variables, and some of the decision variables can be constrained to integer (or binary) values. In cases where a user must solve many instances of a certain class of MIPs, identifying the best solver settings to use can result in significant time savings. Previous efforts using CPLEX (Baz et al., 2009), an industry standard solver, have shown 55% reductions in average solve time.

As this work occurs at the intersection of two different disciplines, each with their own terminology, the terms used must be specified.

Within mixed integer programming solvers, there are many options controlling the execution

of the underlying algorithm, called *parameters*. Each parameter is set to a specific *parameter value* each time the software is used. A *setting* is the set of parameter values when the solver is executed. The MIP solver considered in this paper is CPLEX (CPL, 2009), which has a considerable array of parameters that can be passed to the solver, totaling over 50 (see Appendix A for the complete list of parameters considered).

Each unique optimization problem is termed an *instance*; instances that share the same basic structure in constraint and objective functions are grouped into *instance classes*.

The collection of settings tested on a class of instances are termed *designed experiments*, consisting of performing a predetermined set of *runs* (or trials) where the effects of a different combination of *factors* on a response is observed. The response can be considered the 'result' of the experiment; it is the quantity the experimenter is interested in understanding, and is observed rather than directly set or manipulated. The factors are set explicitly by the experimenter, and are the quantities whose effects on the response are being investigated.

For each run, the factors are set by the experimenter to a predetermined quantity called a *level*. The fact that each run is predetermined is what makes an experiment 'designed'.

Consider the example of finding the combination of route to work and type of gas that maximizes fuel efficiency. The fuel consumed is the response, while the routes and fuel types are the factors. If we only have two routes A and B, and two types of gas C and D, then all possible combinations of the factors would result in the tests in Table 1.1. Each individual test would be a run, and the set of runs taken together form a designed experiment.

Test	Route	Gas
1	A	C
2	A	D
3	B	C
4	B	D

Table 1.1: Example of a Designed Experiment

Once the designed experiment has been performed, the data is analyzed by forming a mathematical expression, or *model*, of the effects of the factors (or independent variables) on the response (or dependent variables). A model is a mathematical expression that relates a function of the factors to the response. Since all combinations of each factor were tried in the example above, one model is to report the response for each combination. When its not possible to try all combinations, other modeling techniques such as regression can be applied.

1.1 Mixed Integer Programs (MIP)

A mixed integer programming problem (MIP) is a set of algebraic expressions that describes an optimization problem. It consists of a set of decision variables, an objective function in terms of the decision variables(either minimization or maximization), and a set of linear constraints that the decision variables must satisfy. In addition, in an MIP, decision variables may either be constrained to integer values.

For example,

$$\max z = 3x_1 + 2x_2 \tag{1.1}$$

subject to

$$x_1 + x_2 \leq 6 \tag{1.2}$$

$$x_1, x_2 \geq 0 \tag{1.3}$$

$$x_1 \in \{0, 1, 2, 3, \ldots\} \tag{1.4}$$

is an MIP with two decision variables (x_1 and x_2), and an objective function $f(x_1, x_2) = z$. In this case, Equation 1.1 is the objective function, while the remaining lines are constraints. In particular, Equation 1.4, along with the fact that x_2 has no such constraints, is what characterizes this problem as an MIP. See Winston and Venkataramanan (2003) for further discussion of MIPs.

1.2 Regression Analysis

Regression analysis is a standard statistical method for estimating the effects of factors on a response. Given a vector of n runs, consisting of pairs of factors and responses, where y_i is the response of the ith set of factors. Further, given a set of factors $x_1, x_2, \ldots, x_k$, and a set of k values for each response such that x_{ij} is the value of the ith factor for the jth response, then the model

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + \epsilon_i, \text{ where } i = 1, 2, \ldots, n \tag{1.5}$$

describes a linear relationship between the response and the corresponding factors. Each ϵ_i is an error term, representing the randomness and error in any process, and β_0 is the intercept term. In the notation of Myers (1990), Equation 1.5 can also be written

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{1.6}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{k1} \\ 1 & x_{12} & x_{22} & \cdots & x_{k2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{1n} & x_{2n} & \cdots & x_{kn} \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

The usual method to find estimates, $\hat{\beta}_i$, for each β_i, is ordinary least squares (OLS). OLS has the goal of minimizing the squared distance between the observed y_i and the estimate $\hat{y}_i$ across all n observations, or equivalently minimizing the sum of squared error (SSE), defined as $\text{SSE} = (\mathbf{y} - \hat{\mathbf{y}})^{\mathrm{T}}(\mathbf{y} - \hat{\mathbf{y}})$, where $\hat{\mathbf{y}}$ is a vector of the n estimates. From Equation 1.6, we let $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$, and find that the minimum squared distance between the observations and predictions is found when

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}. \tag{1.7}$$

The form in Equation 1.5 is what is termed a *first-order* model, as the response is related to the factors by a first-order polynomial function. A *second-order* model includes all possible interactions between the factors and a squared term for each factor. For example, in a model

with factors x_1 and x_2, a second-order model would take the form

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{1i} x_{2i} + \beta_4 x_{1i}^2 + \beta_5 x_{2i}^2 + \epsilon_i, \text{ where } i = 1, 2, \ldots, n. \tag{1.8}$$

All the regression models fit in this application are of the usual linear regression form shown in Equations 1.5 or 1.8, where the response has been transformed $y_i' = \ln(y_i)$ in order to stabilize the variance as solution time increases. For an individual instance, the variation in solve times between settings increases as the average solve time of the instance increases, so per Myers (1990) the log transformation is appropriate, as it reduces large responses more than small ones, while maintaining their order. See Myers (1990) for further discussion of regression and OLS.

1.2.1 LARS Model Reduction

Least angle regression (LARS) is an algorithm for identifying which of the coefficient estimates, or β_is, from Equation 1.5 should be set to something other than 0 (i.e., which of the factors should be included in the model). As presented by Efron et al. (2004), all estimates for the variable coefficients required to describe the parameters are initially set to 0. The vector of responses $\mathbf{y}$ is centered by subtracting the mean response $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ from each y_i,, and the factors are centered and scaled such that their squared sum is 1. Thus, for n observations, we have

$$\sum_{i=1}^{n} y_i = 0, \quad \sum_{i=1}^{n} x_{ij} = 0, \quad \sum_{i=1}^{n} x_{ij}^2 = 1, \quad j = 1, 2, \ldots, \tag{1.9}$$

The factor x_1 with the largest correlation to the response $\mathbf{y}$ is identified. The magnitude of x_1's coefficient is increased until another factor x_2 has an equal amount of correlation with the current vector of residuals $\mathbf{y} - \hat{\mathbf{y}}*$, where $\hat{\mathbf{y}}*$ is the current predicted values for y based on the model containing only x_1. The coefficient of x_2 is then increased from 0 until a third factor x_3 is as strongly correlated to the new residual vector as the first two variables. The coefficient of x_3 is then increased until a fourth factor is identified, and so on.

Without a stopping criterion, this method of selection would result in all variable coefficients being added to the model, with the resulting coefficient estimates the same as the OLS method. Thus, a stopping criterion must be specified to identify a model with the 'best' subset of variable coefficients, and only factors included in the model up to that point are retained. One criterion is the Schwarz Bayesian Information Criterion (SBC) (Schwarz, 1978), which minimizes a function of SSE subject to a penalty for too many factors in the model. SBC is formally defined as

$$\text{SBC} = n \ln\left(\frac{\text{SSE}}{n}\right) + m \ln(n), \quad m = \text{ the number of factors in the current model.} \tag{1.10}$$

Note that $m \leq k$, where k is the total number of factors available for consideration, and n is still the number of observations. SSE, or the sum of squared error, is as for OLS. The function is minimized, and the first factor whose addition would increase the SBC terminates the LARS procedure at the previous step. The SSE decreases as more factors enter the model (m increases), decreasing the first term in the right hand side of Equation 1.10. At the same time the second term in the right hand side of Equation 1.10 is increasing, again as m increases.

As the coefficients calculated by LARS are always less than or equal to those calculated using OLS, a hybrid method is suggested by Efron et al. (2004). The coefficients to include in the model are selected using LARS (with SBC as the stopping criterion), and then the coefficients are estimated using OLS.

1.3 Design of Experiments

In any experiment, the environment is manipulated, and the effects of these manipulations are observed. The distinct quantities that can be manipulated are termed factors, and all possible combinations of the manipulations can be considered the design space.

In terms of k factors $D_1, D_2, \ldots, D_k$, each with q_j levels, the entire design space $\mathcal{D}$ is usually formed from the Cartesian product of the set of factors; i.e., $\mathcal{D} = D_1 \times D_2 \times \cdots \times D_k$. If n runs are to be performed for the experiment, then a subset of n points is selected from $\mathcal{D}$ with the goal of maximizing the knowledge obtained from the experiment. This subset $\mathbf{D}$ is known as the design matrix, where the settings for each of the k factors for the ith run is defined by the ith row of $\mathbf{D}$ (i.e. $\mathbf{d_i} = (d_1, d_2, \ldots, d_k)$, where $d_j \in \{1, 2, \ldots, q_j\}$ for $1 \leq j \leq k$).

Per Atkinson et al. (2007), continuous factors are usually considered bounded, and are often transformed into a variable on the $[-1, 1]$ interval to allow direct comparison of the variable coefficient estimates. Specifically, if a factor D has maximum and minimum values $d_{\max}$ and

Z	X_1	X_2
1	0	0
2	1	0
3	0	1

Table 1.2: Example of Encoding a Categorical Variable

$d_{\min}$, then for any given value d, the transformed variable d' is found by

$$d' = \frac{d - d_0}{\Delta}, \tag{1.11}$$

where $d_0 = (d_{\max} - d_{\min})/2$ and $\Delta = d_{\max} - d_0$.

Categorical factors cannot be scaled in the same way as continuous factors, and so are often represented by dummy variables. For example, consider a 3-level categorical factor Z, with levels 1, 2, and 3. This is encoded using binary $(0, 1)$ dummy variables X_1 and X_2, as seen in Table 1.2.

Thus, a categorical variable with q levels will be encoded into $q - 1$ dummy variables. As a result of this encoding, the transformed design matrix for k factors can have more than k columns. Let p be the number of columns of the transformed design matrix $\mathbf{D}$, which identifies the total number of scaled and dummy variables required to encode the corresponding continuous and categorical factors.

Many designed experiments require an assumption about the mathematical relationship, or model, between the response and the considered factors, with designs selected based on how efficiently data resulting from the experiment will allow estimation of a given model

(Montgomery, 2008). Defining y to be the response with range $\mathcal{Y}$, the usual model proposed is of the form

$$y = f(d_1, \ldots, d_k) + \epsilon, \tag{1.12}$$

where the function $f\colon \mathcal{D} \to \mathcal{Y}$ is specified, ϵ is a random error term, and k is again the number of factors. If $f(\cdot)$ is a continuous function, then categorical factors must be transformed as outlined above.

Once a function $f(\cdot)$ has been specified, the model matrix $\mathbf{X}$ that corresponds to the design matrix $\mathbf{D}$ can be defined. $\mathbf{X}$ is the transformation of $\mathbf{D}$ required to evaluate $f(\cdot)$, and is not always distinct from $\mathbf{D}$. For example, if the model proposed is a first-order polynomial with a constant (or intercept) term, then we have

$$\mathbf{X} = \begin{bmatrix} 1 & d_{11} & d_{12} & \ldots & d_{1p} \\ 1 & d_{21} & d_{22} & \ldots & d_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & d_{n1} & d_{n2} & \ldots & d_{np} \end{bmatrix} \tag{1.13}$$

and defining β_i as the coefficient for the ith variable the proposed model is

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon, \tag{1.14}$$

which is the same first-order form as appears in Equations 1.5 and 1.6.

1.3.1 Space-filling Designs

Space-filling designs pick runs from the design space $\mathcal{D}$ with the aim of uniformly covering (or distributing points throughout) $\mathcal{D}$. However, these designs are unsuitable anytime categorical factors are considered, as they require the notion of distance between different levels of a given factor, which are not present for categorical factors (see Montgomery (2008) for further discussion).

1.3.2 Optimal Designs

Optimal designs are selected based on maximizing (or minimizing) properties of transformations of the model matrix $\mathbf{X}$. There are a wide variety of optimal designs, each attempting to maximize (or minimize) a different property of a transformation $\mathbf{X}$ (Atkinson et al., 2007; Montgomery, 2008).

Optimal designs offer several advantages over other types of designs (Atkinson et al., 2007). They allow models to be estimated with fewer experimental runs, can form models with both categorical and continuous factors (after the transformations outlined above), and are appropriate in the case of a constrained design region (i.e., cases when $\mathcal{D}$ is not a Cartesian product of the factors).

The criteria optimal designs minimize (or maximize) can usually be expressed as a function of the moments matrix, $\frac{\mathbf{X}^{\mathrm{T}}\mathbf{X}}{n}$. Defining $M(\mathbf{X}) = \frac{\mathbf{X}^{\mathrm{T}}\mathbf{X}}{n}$, the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ of M($\mathbf{X}$) are inversely proportional to the square of the lengths of the axes of the confidence ellipsoid

of the variable coefficients. That is, the smaller λ_i is, the less confident we will be about the estimate of the linear combination $\mathbf{a}_i^{\mathrm{T}}\hat{\beta}$, where $\mathbf{a}_i$ is the eigenvector corresponding to λ_i and $\hat{\beta}$ is the variable coefficient estimates (see Atkinson et al. (2007) for further discussion).

Using this concept, several different optimal criteria can be defined. Some of the more common are (per Atkinson et al. (2007))

***A*-optimal** Minimize the sum of the variances of the variable coefficient estimates; i.e., $\min \sum_{i=1}^{p} \frac{1}{\lambda_i}$.

***D*-optimal** Minimize the product of the generalized variances of the variable coefficient estimates; i.e., $\min \prod_{i=1}^{p} \frac{1}{\lambda_i}$.

***E*-optimal** Minimize the variance of the least-well estimated linear combination of variable coefficient estimates $\mathbf{a}_i^{\mathrm{T}}\hat{\beta}$ as defined above; i.e., $\min \max_i \frac{1}{\lambda_i}$.

As the optimality criteria are tied to the underlying model, there can be first-order, second-order, and so on optimal designs, corresponding to the models presented in Equations 1.5 and 1.8, respectively. For example, an optimal design that used a D-optimal criterion and the first order model appearing in Equation 1.5 would be termed a first-order D-optimal design.

D-optimal Designs

D-optimality minimizes the generalized variance of the variable coefficient estimates, resulting in the smallest joint confidence region of the variable coefficient estimates. This property makes them well suited for use when the experimenter is attempting to identify which subset of factors are important in estimating the response, and is why they are used here. Formally, *D*-optimality is defined as minimizing $\min \prod_{i=1}^{p} \frac{1}{\lambda_i}$, which is equivalent to minimizing

$$D = |(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}|, \tag{1.15}$$

where $|\cdot|$ denotes the determinant of a matrix.

Per Box (2005), and assuming that the model form is appropriate, the $1-\alpha$ joint confidence region for p estimated parameters is defined as the sum of squares contour bounded by

$$S = \mathrm{SSE}\left[1 + \frac{p}{n-p} F_{\alpha}(p, n-p)\right], \tag{1.16}$$

where $\mathrm{SSE} = (\mathbf{y} - \hat{\mathbf{y}})^{\mathrm{T}}(\mathbf{y} - \hat{\mathbf{y}})$ as above, and $F_{\alpha}(p, n-p)$ is the α percentile of an $F_{n,n-p}$ distribution. The sum of squares contour is all the coefficient estimates $\boldsymbol{\beta}^*$ that satisfy the bound $(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*)^{\mathrm{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*) \leq S$. This contour defines an ellipsoid centered on $\hat{\boldsymbol{\beta}}$ (the OLS estimates).

Bayesian D-Optimal Designs

When the number of runs is less than the number of variable coefficients to be estimated, the *D*-optimal design is no longer applicable as $|(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}| = 0$. However, it can be modified

along Bayesian principles (DuMouchel and Jones, 1994), allowing construction of designs when $n < p$ and it is assumed that only some of the factors have an effect on the response (i.e., only some of the variable coefficients are non-zero). The factors are divided into two groups, primary and potential, with the primary factors required to be estimable based on the design, and the potential factors considered optional.

Per DuMouchel and Jones (1994), the model now becomes

$$y = f_r(x)\theta_r + f_s(x)\theta_s, \tag{1.17}$$

where θ_r represents the vector of r primary factors, and θ_s represents the vector of s potential factors. The functions $f_r(x)$ and $f_s(x)$ are the proposed form of the model, as is the case in Equation 1.12.

The coefficients of the potential factors are assumed to have some distribution about 0, usually a normal $(0, \tau^2\mathrm{I})$ distribution, where τ is some small positive value. τ^2 indicates the amount of prior knowledge about the potential factors such that large values of τ^2 indicate less precise information. Let K be the $(r+s) \times (r+s)$ diagonal matrix where the first r diagonal elements are 0, and the last s diagonal elements are 1, the function of X to be minimized is

$$|(\mathbf{X}^T\mathbf{X} + \mathbf{K}/\tau^2)^{-1}| \tag{1.18}$$

compared to $|(\mathbf{X}^T\mathbf{X})^{-1}|$ in the usual D-optimal design.

Constructing *D*-Optimal Designs

D-optimal designs can be constructed using a search algorithm, as a solution to the relevant minimization cannot be found using the usual derivative methods. The method employed here is the coordinate-exchange algorithm, derived by Meyer and Nachtsheim (1995). It can handle categorical factors, does not require a subset of $\mathcal{D}$ to be specified, and does not have to consider the entire model space at once.

For each iteration, a random starting design matrix $\mathbf{X}$ is selected from $\mathcal{X}$, and $D = |\mathbf{X}^{\mathrm{T}}\mathbf{X}|$ is calculated. The algorithm then identifies the row x_i in $\mathbf{X}$ that contributes the smallest reduction to Equation 1.15 by finding the run x_i with the smallest value of the deletion function

$$d_D(x_i) = v(x_i) = f^{\mathrm{T}}(x_i)(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}f(x_i), \tag{1.19}$$

where $v(x_i)$ is the variance function, $f^{\mathrm{T}}(x_i)$ is the corresponding row of $\mathbf{X}$, and $f(x_i)$ is the transpose of $f^{\mathrm{T}}(x_i)$.

After this row is identified, a replacement for this row is found by selecting the value of x_i that maximizes the *D*-efficiency of the new design. Computational efficiency is ensured by only allowing x_i to change in one variable (or set of variables corresponding to a categorical variable). The algorithm then continues for the user-specified number of iterations, and for the user-specified number of different random starts. The design that yields the highest *D*-efficiency from all random starts and iterations is the design that is finally selected.

1.4 Parameter Tuning and Benchmarking

There has been some effort to find the best parameter settings for a variety of algorithms, especially in the area of machine learning.

Imbault and Lebart (2004) demonstrate the value of parameter tuning when using support vector machines (SVM), specifically two parameters within the kernel function. Fundamentally, SVMs are a set of hyperplanes that classify examples from two or more classes based on their attributes in multi-dimensional space (Cristianini and Shawe-Taylor, 2000), and the kernel function is used to make the algorithm non-linear.

The methods suggested by Imbault and Lebart (2004) to tune these parameters are genetic algorithms and simulated annealing. Both methods require a large number of successive runs to be performed, each dependent on the results of previous runs. Simulated annealing is also restricted to continuous parameters, and neither method constructs a model to estimate the effects of parameters not explicitly tested.

Kohavi and John (1995) consider parameter optimization in order to minimize prediction errors of C4.5 trees, by manipulating the parameters used in their construction, but their method is limited to numeric or binary parameters. Their application uses best-first search, which treats the parameter space as a set of connected nodes, and pursues the unvisited successor node of visited nodes that has the best score.

Best-first search is applied to the parameters used in generating a binary classification tree using the C4.5 algorithm. However, best-first search is an iterative search procedure, and

so requires an unknown number of runs. It also has difficulty as the number of parameters considered increases, as this dramatically increases the number of nodes that need to be considered.

Audet and Orban (2006) give a more general method of searching through the potential parameters using mesh adaptive direct search, iteratively searching over a decreasing area of the parameter space. It relies on a calculation of distance between parameters, which does not apply to categorical factors. The search method is iterative, and requires an unknown number of steps to find a minimum.

The CALIBRA method proposed by Adenzo-Díaz and Laguna (2004) applies a combination of designed experiments and local search to parameter optimization. A two step procedure is used:

1. Perform a 2^k full factorial experiment, with each parameter restricted to two extreme values, using the results to identify the most important parameter.

2. Search the remaining parameters by repeatedly performing Taguchi $L_9(3^4)$ experiments to find optimal values of the parameters.

2^k factorial designs have k parameters, each restricted to two values. The design table is all possible combinations of the parameters, requiring 2^k runs, and so rapidly increases in size as k increases. Taguchi $L_9(3^4)$ experiments are based on orthogonal Latin squares, and is 9 run design for 4 parameters with 3 levels. The use of full factorial and Taguchi $L_9(3^4)$ experiments limits the procedure to 5 parameters, and also excludes categorical parameters.

Hutter et al. (2009) consider several methods to build a Gaussian process model of the effects of each parameter. A Latin hyper-cube (LHD) design for the parameters is generated, and a Gaussian process model is fit to the results. A new set of runs is then selected based on the expected improvement, and the process repeats until a specified number of iterations have completed, or negligible improvement is found in the Gaussian process model. This process is iterative, and requires a large number of runs to make a recommendation. In addition, the models used do not necessarily offer good extrapolation properties, as Gaussian process models make no attempt to model points not tested, and so may or may not estimate performance well at points not explicitly tested. See Sacks et al. (1989) for details on Gaussian process models.

1.4.1 MIP Parameter Tuning

Tuning the various parameters involved in solving MIP problems can lead to significant reductions in solve time, but there has been little work in systematically approaching the problem.

Laundy et al. (2007) show how dramatic the effects of parameter selection can be in the performance of MIP solvers, and give several examples of MIP problems that cannot be feasibly solved with certain parameter settings. Similarly, Atamtürk and Savelsbergh (2005) also give several examples of problems that greatly benefit from solver parameter tuning. Furthermore, Bixby (2002) demonstrates that half of the increased solve speed observed

from 1992 to 2002 was due to improvement in algorithm design; it stands to reason that the algorithm tuning represented by parameters may give different improvements across different instances.

STOP **Method**

The STOP method (Baz et al., 2007, 2009) demonstrates that significant reduction in solution times can be made by applying a combination of heuristic and machine learning techniques. In brief, the STOP method consists of the following steps (Baz et al., 2007), which are discussed in further detail below:

1. Select an initial set of parameter settings to test either randomly, using a greedy heuristic, or pairwise coverage.
2. Capture the results of running the solver at these settings.
3. Use some form of machine learning or modeling to identify other potentially 'good' settings based on the results from steps 1 and 2.
4. Capture the results of running the solver at these additional settings.
5. Output the best observed setting.

Baz et al. (2007) suggest three different methods for step 1: random, greedy heuristic, and pairwise coverage. Steps 3 and 4 are optional, with regression trees and neural networks considered by Baz et al. (2007).

Random selection is exactly as it sounds; settings to test are selected randomly from all possible combinations, and unsurprisingly gives inconsistent performance in finding good settings.

The greedy heuristic proposed for the STOP method iteratively selects settings based on minimizing the number of parameters in common with settings already selected. The first setting is selected randomly, and for each additional setting all possible settings are compared with the settings already selected. The next setting is chosen by minimizing the maximum number of parameter values in common with any individual previously selected setting. Any ties are broken by selecting the setting with the minimum sum of parameter values in common with all previously selecting settings. As the greedy heuristic requires searching the entire space of possible settings, it quickly becomes computationally infeasible as the number of parameters increases.

The pairwise coverage method used by STOP is proposed by Cohen et al. (1997), and selects a test set where all pairs of parameter settings appear at least twice. Given k factors, where the ith factor has q_i different values, the first setting to test is chosen at random. If n tests are to be run, the $n-1$ remaining settings are picked one at a time by generating a user specified M number of test settings, and selecting the one that covers the largest number of new pairs of parameter values. The following greedy algorithm is used to generate the M candidate settings (Cohen et al., 1997) for each of the $n-1$ remaining settings:

1. Randomly choose a parameter x and a value q for x such that x occurs most frequently

among uncovered pairs.

2. Let $x_1 = x$, and randomly choose an order for the remaining k parameters.

3. For each remaining x_i, $1 < i \leq k$, choose from the parameter setting of the q_i parameter settings for x_i that appears in the maximum number of uncovered pairs, given the levels already chosen for the $i - 1$ variables already considered.

The random choice of the first setting, and random ordering of the parameters in generating candidate sets, means that each run of the algorithm results in a different set of points.

Within the machine learning step, Baz et al. (2007) suggest using either neural networks or regression trees. As STOP uses neural networks to predict the response at every single point in the parameter space to identify further points to test, it is unsuitable for consideration here, given the extremely large parameter space.

By contrast, STOP uses regression trees to identify the two settings that had the largest effect on reducing solution time. Regression trees create a binary tree where each split is based on a single factor. Splits are selected with the goal of minimizing the within partition sum-of-squares for each of the resulting partitions. As presented by Breiman et al. (1984), given a set of data t, a split s is found in a single parameter such that partitions t_L and t_H result. The split s that maximizes

$$\Delta R(s,t) = R(t) - R(t_L) - R(t_H), \tag{1.20}$$

where $R(t) = (1/n_t)\sum_{i=1}^{n_t}(y_i - \bar{y}(t))^2$ is the sum-of-squares within a partition t with n_t elements and an average response of $\bar{y}(t) = (1/n_t)\sum_{i=1}^{n_t} y_i$. See Breiman et al. (1984) for further details and discussion of regression trees.

As regression trees are still feasible with the large parameter space considered, they are the method of machine learning that comparisons will be based on.

Chapter 2

Automated Selection of MIP Solver Parameters

2.1 Introduction

Characterizing the best, or even good, parameter settings for a class of integer programming problems has become difficult due to the increasing abundance of parameter choices in industry standard software. A systematic method for finding good parameter settings can provide significant time savings over the default settings when problems from the same class need to be solved repeatedly (Baz et al., 2009; Atamtürk and Savelsbergh, 2005). Areas where such a method is useful include use of MIPs to allocate resources in real-time computing (Gertphol et al., 2002), using Monte Carlo methods to sample uncertain inputs

(Boone et al., 2009), and for optimization-based data mining algorithms where the data changes at each iteration (Brooks and Lee, 2010). Methods for generating good parameter settings can also facilitate the comparison of solvers, as each solver can be tested at their best.

Parameter tuning, the process of finding the best parameter settings, is common in machine learning. Imbault and Lebart (2004) tune the parameters of support vector machines, and Kohavi and John (1995) gives a general method for tuning parameters for a C 4.5 tree.

Hutter et al. (2009) consider several methods using Latin hyper-cube designs to optimize the parameter effects for a gradient-free global optimization algorithm for continuous functions, and for a local-search algorithm for the propositional satisfiability problem. The CALIBRA method proposed by Adenzo-Díaz and Laguna (2004) applies a combination of designed experiments and local search to parameter tuning in metaheuristic methods. Each of these studies is concerned with finding the optimal setting for a small number of continuous parameters, whereas the parameters for integer programming solvers are often categorical. In addition, all are essentially iterative, and require an unknown (at the start of the procedure) number of runs to identify better settings.

Relatively little work has been done for tuning integer programming solver settings. Laundy et al. (2007) show how dramatic the effects of parameter selection can be in the performance of MIP solvers, and give several examples of MIP problems that cannot be feasibly solved with certain parameter settings. Similarly, Atamtürk and Savelsbergh (2005) also give several examples of problems that greatly benefit from solver parameter tuning. Furthermore,

Bixby (2002) demonstrates that half of the increased solve speed observed from 1992 to 2002 was due to improvement in algorithm design; it stands to reason that the algorithm tuning represented by parameters may give different improvements across different instances.

The STOP method (Baz et al., 2007, 2009) applies a combination of heuristic and machine learning techniques to MIP solver parameter optimization. A first set of runs is selected, with the goal of covering as much of the parameter space (i.e., the set of all possible combinations of parameter values) as possible, and then an optional set of follow-up runs is selected in the same manner after fixing the most important parameters. When applied to large numbers of parameters, several options in the STOP cannot be feasibly applied. The only set that will work is selecting points using pairwise selection (Cohen et al., 1997) for the selection of points, and regression trees (Breiman et al., 1984) for any further learning. To date, no rigorous analysis of integer programming parameter settings has been conducted that accounts for the large number of categorical variables, forms a model allowing for extrapolation, and does not require an unknown number of iterations to arrive at a recommendation.

This paper presents a method that uses designed experiments and statistical models to extract information about how parameter settings perform for classes of integer programs. The use of experimental design facilitates fitting a model that describes the effects of the different parameter settings across the entire parameter space, even those not explicitly tested, allowing identification of both desirable and poor settings. The model also provides an assessment of the likely contribution of each parameter to solution time. Applying experimental design to optimization software parameter tuning is not straightforward due to the large number of

parameters, and is further complicated by their categorical nature.

We overcome these obstacles by using computer-generated (*optimal*) designs that provide flexibility in the number of test runs and the number of parameter choices. The model formed allows for extrapolation to points not explicitly tested, and does not require an unknown number of iterations to identify a better setting.

Within this paper, a method for modeling the effects of different parameter settings is presented, which is compared with a modified STOP method. Results of this method applied to two different solvers (CPLEX and GLPK) and two different instance classes follow.

2.2 Method

Once the solver and instance class have been identified, and the set of parameters of interest has been chosen, a two-step process is followed to identify important parameters, and model their effects:

1. *Screening*: Construct a first-order D-optimal design, and use the results to identify parameters with important effects on solution time, using either reduction of a first-order regression model or regression trees.

2. *Optimization*: Construct a second-order D-optimal design to fit a second-order regression model the effects of the important parameters, and identify which non-default settings give the best performance.

The screening design can be either constructed as a D-optimal, or Bayesian D-optimal design, depending on the number of parameters of interest (Atkinson et al., 2007). LARS (Efron et al., 2004) or regression trees (Breiman et al., 1984) are used on the results of the screening design to identify the most important parameters, which are the only parameters considered in the optimization step.

If regression trees are used to reduce the model in the screening step, the parameters in the left-most nodes (that is, the nodes predicting the smallest response) are retained, and each parameter identified is restricted to the values used in the split. If LARS is used to reduce the model formed in either step, then the parameter space is restricted to the parameters selected by LARS. Note that regression trees are only used in the screening step; the model in the optimization step is left unreduced.

If the number of parameters of interest is small enough and a second-order design of all the parameters is practical, the first step can be skipped completely. In this case, a second-order model is fit directly to the results of the experiment.

In both steps, the models fit are linear models regressing the response on the parameters. If the response considered is solution time, the log transform is used, in order to reduce the effects of increased variance in solution time as solution time increases. If more than one example of the instance class is to be used, the designs are run on all instances, and the response used in the models is the log of the average solution time, or average gap, as appropriate, again to reduce the effects of increased variance.

The second-order model used in the optimization step contains all values of the parameters, and all potential interactions. For example, in a model with factors x_1 and x_2, a second-order model would take the form

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{1i} x_{2i} + \epsilon_i, \text{ where } i = 1, 2, \ldots, n. \tag{2.1}$$

If the interaction term β_3 is significant (non-zero), then the effects of one factor depends on the level of the other. For example, if we consider a case in which both x_1 and x_2 are binary $(0, 1)$ variables, then the estimates of y can be seen in Table 2.1. Note that the when $x_1 = x_2 = 1$, the estimate is not just the sum of the respective first order coefficients (β_1 and β_2), but also include the interaction term, β_3.

Value of x_1	Value of x_2	Estimate for y
0	0	β_0
0	1	$\beta_0 + \beta_2$
1	0	$\beta_0 + \beta_1$
1	1	$\beta_0 + \beta_1 + \beta_2 + \beta_3$

Table 2.1: Effects of 2nd Order Iteraction

Gap is often given as a indicator of a solution to an MIP problem, and for minimization problems is calculated as

$$G = \frac{I - L}{I}, \tag{2.2}$$

where G is the current gap, and I is the objective value of the incumbent solution. The incumbent solution is the best-known feasible solution in terms of objective function value.

For minimization, I represents the best-known upper bound on the optimal objective value. For minimization, L represents the best-known lower bound on the optimal objective value. It is the smallest objective value among all active linear programming relaxations. It will always be positive, and the lower the gap, the higher quality the current found solution to the MIP.

2.3 GLPK Results

The above method is applied using GLPK and a set of 6 GLPK parameters used by Baz et al. (2007) (see Appendix B for the list of parameters), using only the optimization stage of the method. All 768 possible combinations of the parameters are run on 20 MIPs from a set of telecommunication network design problems. A second-order D-optimal design with 92 runs is found using the coordinate-exchange method (Meyer and Nachtsheim, 1995), and a second-order regression model is fit to the log of the average solution times across the 20 instances. This model includes all the parameter values for each of the GLPK parameters considered, and all pairs of combinations of these values. It is of the form

$$\ln y = \beta_0 + \beta_1 x_1 + \ldots, \tag{2.3}$$

where y is the average solution time, β_0 is the intercept, and each remaining term is either an indicator variable on the categorical parameter values indicated, or the unscaled value of the categorical parameter indicated, and all pairs of combinations between parameters.

An unreduced second-order model is used to identify the setting that would give the fastest

Parameter	Recommended Value	Meaning
Presolve	GLP_ON	Presolve On
Pricing Strategy	GLP_PT_PSE	Projected Steepest Edge
Cutting Plane Settings	NEW_WITH_CUTS	Generate MIR Cuts
Scaling Process	GLP_SF_EQ	Perform Equilibration Scaling
Variable Selection	GLP_BR_MFV	Most Fractional Variable
Backtracking Method	GLP_BT_BFS	Breadth First Search

Table 2.2: Recommended Settings in GLPK Case

predicted solution time, and gives the recommended settings appearing in Table 2.2.

For the sake of comparison, 1,000 sets of 92 points are selected using STOP's pairwise selection algorithm with no machine learning step (Baz et al., 2007), and the fastest run in each of the sets was identified. The results appear in Table 2.3, and while the model recommends the 19th fastest parameter setting out of the 768 possibilities, STOP selects faster settings nearly 95% of the time. The default settings given an average solution time of 138.9 seconds, in the 28th percentile of all the tested settings, which is beat by both all STOP recommendations, and the model recommendation.

In looking at the full results in Figure 2.1, it is clear that the model does much better than the defaults, and nearly all other possible settings.

STOP also does well, as can be seen in Figure 2.2. Viewing the 1,000 STOP runs as a simulation of STOP's capabilities, STOP will find one of the 19 fastest settings with a

Avg. Solve Time (seconds)	Avg. Solve Time Rank	Predicted Solve Time Rank	Cumulative Probability of Selection by STOP
56.99511	1	39	0.162
57.63932	2	13	0.240
58.24973	3	35	0.331
59.18838	4	14	0.426
59.3633	5	5	0.462
59.61411	6	32	0.551
59.71484	7	37	0.630
59.97544	8	11	0.702
60.03032	9	47	0.748
60.22806	10	22	0.777
61.1366	11	21	0.800
61.32975	12	4	0.819
63.00328	13	6	0.836
63.276	14	27	0.871
63.39602	15	26	0.890
64.1878	16	28	0.912
64.28338	17	33	0.935
64.64919	18	36	0.944
64.67862	19	1	0.952
65.40978	20	12	0.965

Table 2.3: Selected Results of GLPK Case

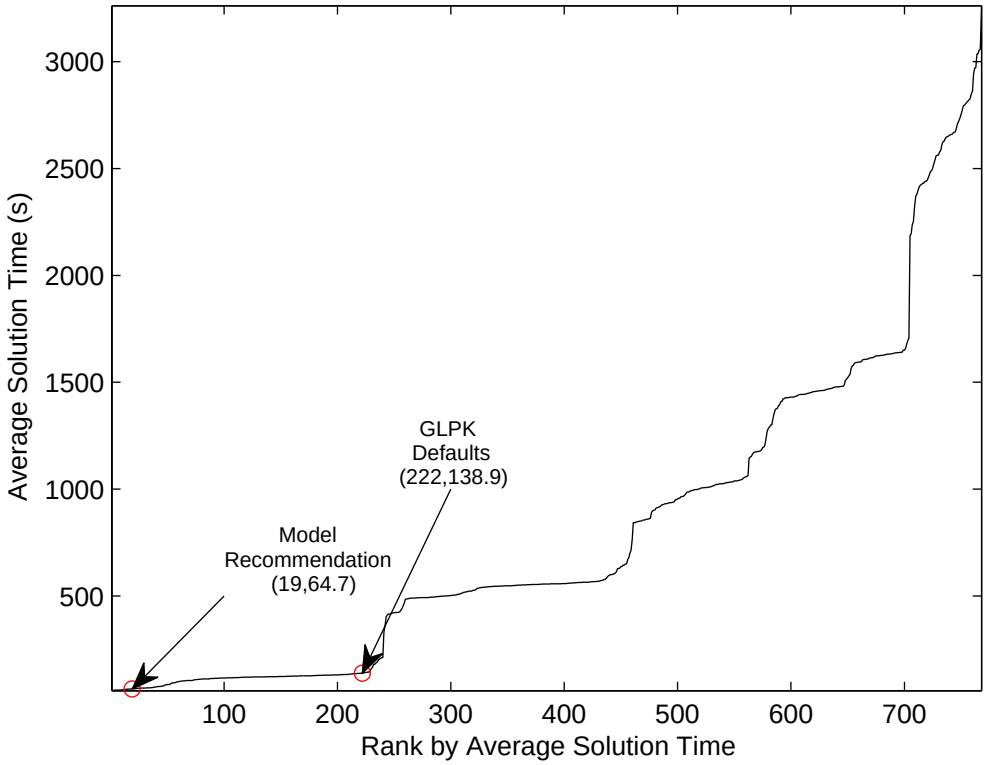

Figure 2.1: Solution Time Rank by Average Solution Time of GLPK Case

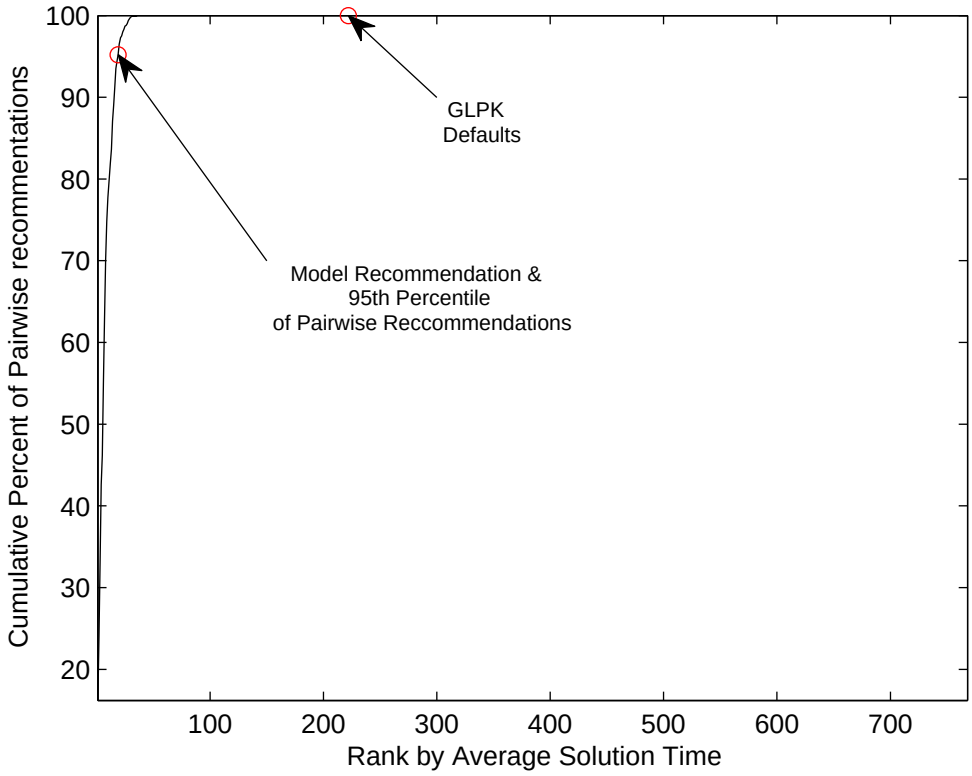

Figure 2.2: Solution Time Rank by Cumulative Percentile of STOP Recommendations of GLPK Case

probability of 0.95, and will almost certainly beat the defaults.

While STOP is highly likely to recommend a setting faster than that found by the model, the model recommendation is only 13.5% slower than the fastest observed setting, and it still much faster than the defaults. Using designed experiments also gives an interpretable model (see Table 2.4 for the factors with the largest coefficients). Cut Method and Variable Selection dominate the large parameters, and are clearly very important in solution time for these instances. In particular, parameter values with large positive estimates increase

solution time, and parameter values with large negative positive estimates reduce it. See Appendix C for the full list of estimates. Since there are significant interaction terms, the effects of the value of each parameter depend on the value of the parameter in the interaction. This can be seen specifically with Cut Method [NEW]; if Variable Selection is set to [GLP_BR_LFV], then the solution time is increased, but if Variable Selection is set to [GLP_BR_FFV], then solution time is decreased.

2.4 CPLEX Results

Using CPLEX as the solver, the above method is applied to two cases; a limited subset of 6 CPLEX parameters used by Baz et al. (2007), and the full set of 54 parameters available to CPLEX when solving an MIP. In the limited case, only the second modeling step is performed, as the number of parameters considered is few enough that a second-order design is practical immediately. In the full case, both the screening and optimization steps are used. Results are compared to the STOP method (Baz et al., 2007, 2009), as it also attempts to find better parameter settings for MIP solvers, and does not require an open-ended number of iterations.

The two instance classes considered are 20 MIPs from a set of telecommunication network design problems, and a class of five instances of an integer program used for building models of cellular metabolism.

Parameter [Value]	Estimate
Cut Method [NEW_WITH_CUTS]*Variable Selection [GLP_BR_LFV]	1.92192
Cut Method [NEW]*Variable Selection [GLP_BR_LFV]	1.801171
Pricing [GLP_PT_PSE]	-1.761248
Cut Method [NEW]*Variable Selection [GLP_BR_FFV]	-1.199737
Cut Method [NEW_WITH_CUTS]*Variable Selection [GLP_BR_FFV]	-1.137504
Variable Selection [GLP_BR_FFV]	1.129192
Variable Selection [GLP_BR_DTH]	0.822817
Cut Method [NEW]*Variable Selection [GLP_BR_DTH]	-0.66454
Pricing [GLP_PT_PSE]*Cut Method [NEW_WITH_CUTS]	0.579577
Pricing [GLP_PT_PSE]*Cut Method [NEW]	0.550387
Cut Method [NEW_WITH_CUTS]*Variable Selection [GLP_BR_DTH]	-0.548912
Cut Method [NEW_WITH_CUTS]	-0.44036
Variable Selection [GLP_BR_LFV]	-0.410779
Cut Method [NEW]	-0.351167

Table 2.4: Selected Sorted 2nd Order Coefficient Estimates in GLPK Case

2.4.1 Limited Case

The method is applied to the following six CPLEX parameters, as selected by Baz et al. (2007). Only the optimization step is applied, as the number of parameters considered is small enough that the screening step can feasibly be skipped.

MIP Emphasis Controls trade-offs between speed, feasibility, optimality, and moving bounds in MIP (4 settings).

Node Selection Sets the rule for selecting the next node to process when backtracking (3 settings).

Branching Selection Sets the rule for selecting the branching variable at the node which has been selected for branching (3 settings).

Dive Type Controls the MIP dive strategy (4 settings).

Fractional Cuts Decides whether or not Gomory fractional cuts should be generated for the problem (3 settings).

MIR Cuts Decides whether or not to generate MIR cuts (mixed integer rounding cuts) for the problem (3 settings).

All of these parameters are categorical, with a total of 1,296 potential combinations of settings. All potential combinations of settings are then run for 20 instances from the telecommunication network design class of MIPs.

Parameter	Recommended Value	Meaning
MIP Emphasis	1	Emphasize feasibility over optimality
Node Selection	1	Best-bound search
Branching Selection	3	Strong branching
Dive Type	1	Traditional dive
Fractional Cuts	-1	Do not generate Gomory fractional cuts
MIR Cuts	-1	Do not generate MIR cuts

Table 2.5: Recommended Settings in Limited CPLEX Case

A second-order D-optimal design with 96 runs is found using the coordinate-exchange method (Meyer and Nachtsheim, 1995), and a second-order regression model is fit to the log of the average solution times across the 20 instances. This model includes all the parameter values for each of the CPLEX parameters considered, and all pairs of combinations of these values. It is of the form

$$\ln y = \beta_0 + \beta_1 x_1 + \ldots, \tag{2.4}$$

where y is the average solution time, β_0 is the intercept, and each remaining term is either an indicator variable on the categorical parameter values indicated, or the unscaled value of the categorical parameter indicated, and all pairs of combinations between parameters.

The unreduced second-order model is used to identify the setting that would give the fasted predicted solution time, and gives the recommended settings appearing in Table 2.5 (meanings from CPL (2009)).

For the sake of comparison, 1,000 sets of 96 points are selected using STOP's pairwise selection algorithm with no machine learning step (Baz et al., 2007), and the fastest run in each of the sets was identified.

The results were encouraging, as can be seen in Table 2.6. The rank of model predictions is the rank of the predicted solution time, and the number of STOP recommendations is the number of STOP instances that recommended that setting as the fastest. The parameter combination identified by the model as the fastest (ranked first among model predictions) is the third fastest setting among all 1,296 possibilities, and is as fast or faster than the fastest setting found by STOP 92% of the time. The default settings give an average solve time of 4.1265 seconds, in the 65th percentile of all the tested settings, which is beat quite handily by every run of STOP, and the settings recommended by the model.

In looking at the full results in Figure 2.3, it is clear that the model does much better than the defaults, and nearly all other possible settings.

STOP also does well, as can be seen in Figure 2.4. Viewing the 1,000 STOP runs as a simulation of STOP's capabilities, STOP will find one of the 43 fastest settings with a probability of 0.95, and will almost certainly beat the defaults.

From these results, both STOP and a designed experiment perform well given a limited subset of solver parameters. Using a designed experiment and model gives better results than most STOP runs, and has the added benefit of providing an interpretable model of the effects each parameter value has on solution time across the instance class (see Table 2.7 for

Avg. Solve Time (seconds)	Avg. Solve Time Rank	Predicted Solve Time Rank	Cumulative Probability of Selection by STOP
0.9205	1	10	0.052
0.9205	2	8	0.076
0.9235	3	1	0.115
0.9235	4	12	0.153
0.9240	5	35	0.221
0.9250	6	34	0.252
0.9270	7	15	0.301
0.9280	8	42	0.372
0.9300	9	51	0.450
0.9390	10	9	0.497

Table 2.6: Selected Results of Limited CPLEX Case

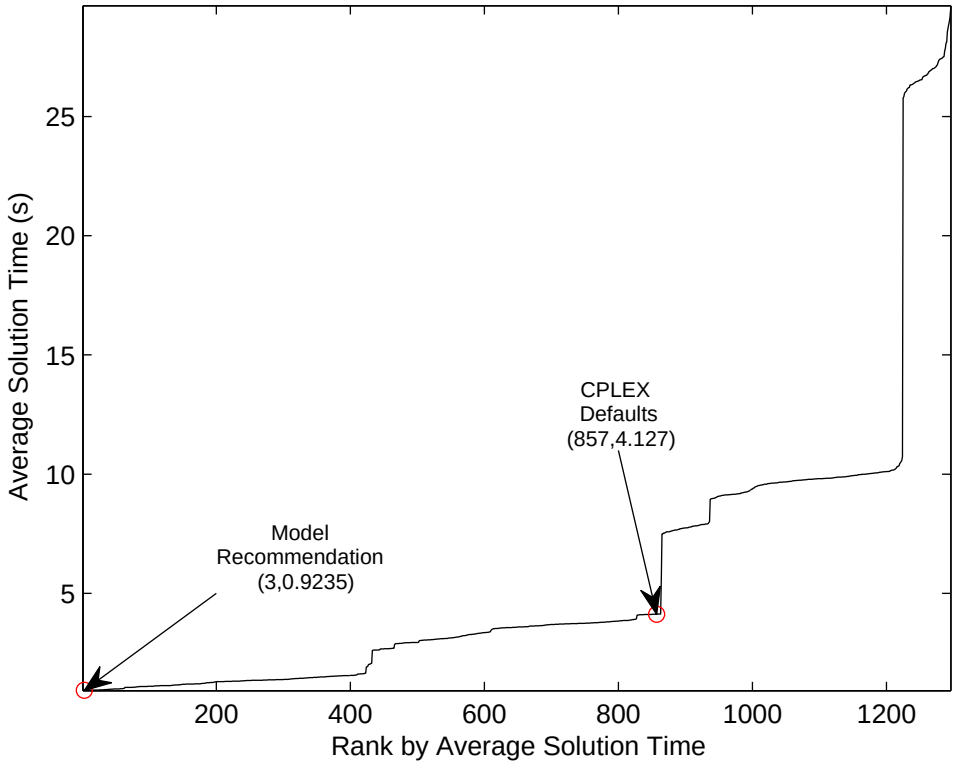

Figure 2.3: Solution Time Rank by Average Solution Time of Limited CPLEX Case

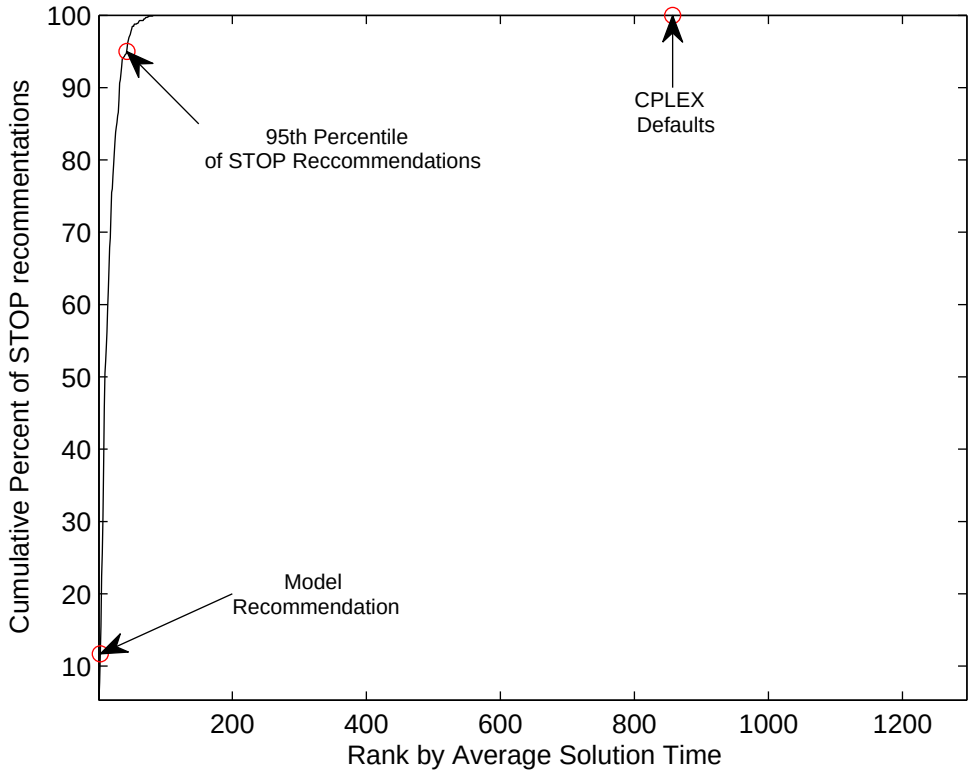

Figure 2.4: Solution Time Rank by Cumulative Percentile of STOP Recommendations of Limited CPLEX Case

selected coefficient estimates). MIP Emphasis and Branching Selection dominate the large effects, appearing in nearly all of the largest 15 coefficients, and are thus very important to solve time for this instance class.

2.4.2 Full Case

In the full case, all 54 CPLEX parameters in Appendix A are considered, and both steps of the method are applied. For the first screening step, a D-optimal design with 136 runs was used, as that is the minimum size required to estimate variable coefficients for all 54 parameters. This method was applied to both the telecommunications instance class, using solution time as the response, and the cellular metabolism instance class, using optimality gap as the response.

Telecommunications Class

A first-order model was fit to the 136 run initial design, and the recommended setting of all 54 parameters was tested. For the sake of comparison, 10 STOP runs were constructed, again using only the pairwise coverage algorithm. The results appear in Table 2.8. The recommendation identified by the model still beats the default average time of 4.1265 seconds, but is slower than the fastest runs found by each of the STOP runs.

To identify a better setting, a regression tree was fit to the results of the initial 136 run design, and the parameter space restricted to the values predicted to give the best solution time.

Parameter [Value]	Estimate
MIP Emphasis [1]	-1.028217
MIP Emphasis [0]	-0.836883
MIP Emphasis [2]	-0.827975
MIP Emphasis [2]*Branching Selection [2]	0.744255
MIP Emphasis [2]*Branching Selection [0]	0.741006
MIP Emphasis [1]*Branching Selection [2]	0.529388
Branching Selection [0]	0.366774
MIP Emphasis [0]*Branching Selection [2]	0.332516
MIP Emphasis [0]*Branching Selection [0]	0.320321
Branching Selection [2]	0.303102
Branching Selection [0]*Fractional Cuts [1]	-0.133259
MIP Emphasis [1]*Dive Type [2]	0.125175
Branching Selection [2]*Fractional Cuts [1]	-0.109013
MIP Emphasis [1]*Branching Selection [0]	-0.104109
Branching Selection [2]*Dive Type [1]	0.101834

Table 2.7: Selected Sorted 2nd Order Coefficient Estimates in Limited CPLEX Case

Recc. By	Avg. Solve Time
Model	2.3365
STOP run 1	0.5355
STOP run 2	1.6735
STOP run 3	1.6275
STOP run 4	1.3955
STOP run 5	1.3825
STOP run 6	0.954
STOP run 7	1.352
STOP run 8	0.865
STOP run 9	1.427
STOP run 10	1.266

Table 2.8: Results of Initial Run in the Full CPLEX Case- Telecommunications Class

Parameter	Subset of Settings
MIP variable selection strategy	0,2,3,4
MIP subproblem algorithm	0,2,4
MIP starting algorithm	0,1,2,4,5
MIP cliques switch	-1,0,3
Dual simplex pricing algorithm	0,1,2,3,4
Symmetry breaking	-1,1,2,3
Primal simplex pricing algorithm	-1,1
Constraint aggregation limit for cut generation	>=5.5

Table 2.9: Parameters Identified by Regression Tree in Full CPLEX- Telecommunications Class

The restricted parameter space appears in Table 2.9, and reduces the number of parameters considered from 54 to 8. All the settings identified are categorical except for the last, which is continuous. A follow-up second-order D-optimal design with 211 runs is created, allowing for estimation of all parameter effect estimates.

The resulting second-order linear regression model model includes all the parameter values for each of the CPLEX parameters considered, and all pairs of combinations of these values. It is of the form

$$\ln y = \beta_0 + \beta_1 x_1 + \ldots, \tag{2.5}$$

where y is the average solution time, β_0 is the intercept, and each remaining term is either

MIP subproblem algorithm	0,1,2,4
MIP branching direction	1
Dual simplex pricing algorithm	0,1,2,3,4
Primal simplex pricing algorithm	-1,0,2
MIP implied bound cuts switch	-1
MIP strategy best bound interval	0

Table 2.10: Parameters Identified by Regression Tree in Full CPLEX- Cellular Metabolism Class

an indicator variable on the categorical parameter values indicated, or the unscaled value of the categorical parameter indicated, and all pairs of combinations between parameters (see Appendix E for the full model). The model is used to identify a setting in these parameters with an average solution time of 1.702 seconds, substantially better than the 4.127 seconds observed at the default settings. STOP still offers better absolute performance, but again provides no model.

Cellular Metabolism Class

The same first-order D-optimal design is applied to the five instances in the cellular metabolism class, using optimality gap as the response. After running the design on all instances, a regression tree is used to identify the parameters and parameter levels that lead to the smallest (that is, best) optimality gap. These parameters appear in Table 2.10.

A 36 run second-order D-optimal design is created in the parameters in Table 2.10. As three of them are constant, only the three with parameter ranges need to be included in the design. The results of this design are used to model the effects of the three parameters on gap, which appears in Appendix F, and a recommendation found. The resulting second-order linear regression model model includes all the parameter values for each of the CPLEX parameters considered, and all pairs of combinations of these values. It is of the form

$$y = \beta_0 + \beta_1 x_1 + \ldots, \tag{2.6}$$

where y is the average gap, β_0 is the intercept, and each remaining term is either an indicator variable on the categorical parameter values indicated, or the unscaled value of the categorical parameter indicated, and all pairs of combinations between parameters The recommendation found by this model resulted in an average gap of 50.4%, while a single 136 run STOP design found a recommendation of 40.8%, better than recommended by the model.

2.5 Conclusions & Further Work

Applying the method presented here to GLPK yields results that are substantially faster than defaults, but unlikely to beat the recommendation found by STOP. In the limited CPLEX case, which considers only six parameters, the method outlined here compares favorably to STOP, and recommends a setting with a 0.3% slower average solve time than the fastest observed, and as fast or faster than 92% of STOP reommendations. Unlike STOP, it also provides an interpretable model of the effects of each parameter values, which may be useful

in understanding how instance classes interact with the various parameters.

The STOP method is adapted to consider a much larger number of parameters in the full case, and the use of pairwise-coverage and regression trees for learning provided results significantly faster than the default settings provided by CPLEX. In the case that no model of parameter effects is desired, these tweaks to the STOP method give a promising way to tune large numbers of solver parameters at once.

Applying designed experiments and models to the full case also yields encouraging results, again finding a setting substantially better than the default, and yielding an explanatory model. However, as the recommendation does not best STOPs results, there are some potential improvements. Different experimental design procedures or different modeling techniques could be employed, as there is still some difficulty in constructing models given the large number of categorical factors. Based on the results here, regression trees in particular may be promising as a method of modeling solution time.

Other performance measures than solution time or optimality gap can also be considered. The quality of a solution at a given cutoff may be more important for a real-time optimization system than the average solution time, and the parameter settings needed for such an objective may be different from those required for fastest average solution time.

Bibliography

(2009), "GLPK (GNU Linear Programming Kit)," http://www.gnu.org/software/glpk/.

(2009), "IBM ILOG CPLEX," http://www-01.ibm.com/software/integration/optimization/cplex

(2010), "Maximal Software," http://www.maximalsoftware.com/.

Adenzo-Díaz, B. and Laguna, M. (2004), "Fine-tuning of Algorithms Using Fractional Experimental Designs and Local Search," .

Atamtürk, A. and Savelsbergh, M. W. P. (2005), "Integer-Programming Software Systems," *Annals of Operations Research*, 140, 67Ű124.

Atkinson, A. C., Donev, A. N., and Tobias, R. D. (2007), *Optimum Experimental Designs, with SAS*, vol. 34 of *Oxford Statistical Science Series*, Oxford University Press.

Audet, C. and Orban, D. (2006), "Finding Optimal Algorithmic Parameters Using Derivative-Free Optimization," *SIAM Journal on Optimization*, 17, 642–664.

Baz, M., Hunsaker, B., Brooks, J. P., and Gosavi, A. (2007), "Automated tuning of op-

timization software parameters," Tech. Rep. 7, University of Pittsburgh Department of Industrial Engineering.

Baz, M., Hunsaker, B., and Prokopyev, O. (2009), "How much do we "pay" for using default parameters?" *Computational Optimization and Applications*.

Bixby, R. E. (2002), "Solving Real-World Linear Programs: A Decade and More of Progress," *Operations Research*, 50, 3–15.

Boone, E., Brooks, J., Bullene, C., Lipchin, C., Sorrell, T., Springer, J., Stewart, C., and Tanvir, Y. (2009), "Network Design for Middle East Water Distribution," .

Box, G. (2005), *Statistics for Experimenters*, New York: John Wiler & Sons, Inc.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. (1984), *Classification and Regression Trees*, Boulder: Westview.

Brooks, J. P. and Lee, E. K. (2010), "Analysis of the consistency of a mixed integer programming-based multi-category constrained discriminant model," *Annals of Operations Research*, 174, 147–168.

Cohen, D. M., Dalal, S. R., Fredman, M. L., and Patton, G. C. (1997), "The AETG System: An Approach to Testing Based on Combinatorial Design," *IEEE Transactions on Software Engineering*, 23, 437–444.

Cristianini, N. and Shawe-Taylor, J. (2000), *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge, UK: Cambridge University Press.

DuMouchel, W. and Jones, B. (1994), "A Simple Bayesian Modification of D-Optimal Designs to Reduce Dependence on an Assumed Model," *Technometrics*, 36, 37–47.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004), "Least Angle Regression," *The Annals of Statistics*, 32, 407–451.

Gertphol, S., Yu, Y., Gundala, S. B., Prasanna, V. K., Ali, S., Kim, J.-K., Maciejewski, A. A., and Siegel, H. J. (2002), "A Metric and Mixed-Integer-Programming-Based Approach for Resource Allocation in Dynamic Real-Time Systems," *Parallel and Distributed Processing Symposium, International*, 1, 0010.

Hutter, F., Hoos, H. H., Leyton-Brown, K., and Murphy, K. P. (2009), "An Experimental Investigation of Model-Based Parameter Optimisation: SPO and Beyond," in *GECCO 09*, Genetic and Evolutionary Computation.

Imbault, F. and Lebart, K. (2004), "A stochastic optimization approach for parameter tuning of Support Vector Machines," in *Proceedings of the 17th International Conference on Pattern Recognition*, International Conference on Pattern Recognition.

Kohavi, R. and John, G. H. (1995), "Automatic Parameter Selection by Minimizing Estimated Error," in *Machine Learning: Proceedings of the Twelfth International Conference*.

Laundy, R., Perregaard, M., Tavares, G., Tipi, H., and Vazacopoulos, A. (2007), "Solving Hard Mixed Integer Programming Problems With Xpress-MP: A MIPLAB 2003 Case Study," Tech. Rep. 2, Rutgers Center for Operations Research.

Meyer, R. K. and Nachtsheim, C. J. (1995), "The Coordinate-Exchange Algorithm for Constructing Exact Optimal Experimental Designs," *Technometrics*, 37, 60–69.

Montgomery, D. C. (2008), *Design and analysis of experiments*, Hoboken, NJ: Wiley.

Myers, R. H. (1990), *Classical and modern regression with applications*, Pacific Grove, CA: Duxbury.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989), "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409–423.

Schwarz, G. (1978), "Estimating the Dimension of a Model," *The Annals of Statistics*, 6, 461–464.

Winston, W. L. and Venkataramanan, M. (2003), *Introduction to Mathematical Programming*, Pacific Grove, CA: Thompson Learning.

Appendix A

List of CPLEX Parameters

The 54 CPLEX parameters selected for consideration in the full case are described here, using information from CPL (2009).

Backtracking tolerance Controls how often backtracking is done during the branching process. (continuous)

Bound strengthening switch Decides whether to apply bound strengthening in mixed integer programs (MIPs). Bound strengthening tightens the bounds on variables, perhaps to the point where the variable can be fixed and thus removed from consideration during branch & cut. (3 level categorical)

Candidate limit for generating Gomory fractional cuts Limits the number of candidate variables for generating Gomory fractional cuts (continuous)

Coefficient reduction setting Decides how coefficient reduction is used. Coefficient reduction improves the objective value of the initial (and subsequent) LP relaxations solved during branch & cut by reducing the number of non-integral vertices. (3 level categorical)

Constraint aggregation limit for cut generation Limits the number of constraints that can be aggregated for generating flow cover and mixed integer rounding (MIR) cuts. (continuous)

Dependency switch Decides whether to activate the dependency checker. If on, the dependency checker searches for dependent rows during preprocessing. If off, dependent rows are not identified. (5 level categorical)

Dual simplex pricing algorithm Decides the type of pricing applied in the dual simplex algorithm. The default pricing (0) usually provides the fastest solution time, but many problems benefit from alternate settings. (6 level categorical)

Limit on the number of presolve passes made Limits the number of presolve passes that CPLEX makes during preprocessing. When this parameter is set to a nonzero value, invokes CPLEX presolve to simplify and reduce problems. (3 level categorical)

Linear reduction switch Decides whether linear or full reductions occur during preprocessing. (2 level categorical)

Local branching heuristic Controls whether CPLEX applies a local branching heuristic to try to improve new incumbents found during a MIP search. (2 level categorical)

MIP branching direction Decides which branch, the up or the down branch, should be taken first at each node. (3 level categorical)

MIP candidate list limit Controls the length of the candidate list when CPLEX uses variable selection as the setting for strong branching. (continuous)

MIP cliques switch Decides whether or not clique cuts should be generated for the problem. Setting the value to 0 (zero), the default, indicates that the attempt to generate cliques should continue only if it seems to be helping. (5 level categorical)

MIP covers switch Decides whether or not cover cuts should be generated for the problem. Setting the value to 0 (zero), the default, indicates that the attempt to generate covers should continue only if it seems to be helping. (5 level categorical)

MIP disjunctive cuts switch Decides whether or not disjunctive cuts should be generated for the problem. Setting the value to 0 (zero), the default, indicates that the attempt to generate disjunctive cuts should continue only if it seems to be helping. (5 level categorical)

MIP dive strategy Controls the MIP dive strategy. The MIP traversal strategy occasionally performs probing dives, where it looks ahead at both children nodes before deciding which node to choose. (4 level categorical)

MIP emphasis switch Controls trade-offs between speed, feasibility, optimality, and moving bounds in MIP. (5 level categorical)

MIP flow cover cuts switch Decides whether or not to generate flow cover cuts for the problem. Setting the value to 0 (zero), the default, indicates that the attempt to generate flow cover cuts should continue only if it seems to be helping. (4 level categorical)

MIP flow path cut switch Decides whether or not flow path cuts should be generated for the problem. (4 level categorical)

MIP Gomory fractional cuts switch Decides whether or not Gomory fractional cuts should be generated for the problem. (4 level categorical)

MIP GUB cuts switch Decides whether or not to generate GUB cuts for the problem. (4 level categorical)

MIP heuristic frequency Decides how often to apply the periodic heuristic. (3 level categorical)

MIP implied bound cuts switch Decides whether or not to generate implied bound cuts for the problem. (4 level categorical)

MIP MIR (mixed integer rounding) cut switch Decides whether or not to generate MIR cuts (mixed integer rounding cuts) for the problem. (4 level categorical)

MIP node selection strategy Used to set the rule for selecting the next node to process when backtracking. The depth-first search strategy chooses the most recently created node. The best-bound strategy chooses the node with the best objective function for the associated LP relaxation. The best-estimate strategy selects the node with the best

estimate of the integer objective value that would be obtained from a node once all integer infeasibilities are removed. An alternative best-estimate search is also available. (4 level categorical)

MIP priority order generation Selects the type of generic priority order to generate when no priority order is present. (4 level categorical)

MIP priority order switch Decides whether to use the priority order, if one exists, for the next mixed integer optimization. (2 level categorical)

MIP probing level Sets the amount of probing on variables to be performed before MIP branching. Higher settings perform more probing. Probing can be powerful but time-consuming at the start. (5 level categorical)

MIP repeat presolve switch Decides whether to re-apply presolve, with or without cuts, to a MIP model after processing at the root is otherwise complete. (5 level categorical)

MIP starting algorithm Sets which continuous optimizer will be used to solve the initial relaxation of a MIP. (7 level categorical)

MIP strategy best bound interval Sets the best bound interval for MIP strategy. (3 level categorical)

MIP subproblem algorithm Decides which continuous optimizer will be used to solve the subproblems in a MIP, after the initial relaxation. (6 level categorical)

MIP variable selection strategy Sets the rule for selecting the branching variable at the node which has been selected for branching. (6 level categorical)

Node presolve switch Decides whether node presolve should be performed at the nodes of a mixed integer programming (MIP) solution. Node presolve can significantly reduce solution time for some models. The default setting is generally effective at deciding whether to apply node presolve, although runtimes can be reduced for some models by the user turning node presolve off. (4 level categorical)

Number of cutting plane passes Sets the upper limit on the number of cutting plane passes CPLEX performs when solving the root node of a MIP model. (3 level categorical)

Pass limit for generating Gomory fractional cuts Limits the number of passes for generating Gomory fractional cuts. (2 level categorical)

Preprocessing aggregator application limit Invokes the aggregator to use substitution where possible to reduce the number of rows and columns before the problem is solved. If set to a positive value, the aggregator is applied the specified number of times or until no more reductions are possible. (continuous)

Preprocessing aggregator fill Limits variable substitutions by the aggregator. If the net result of a single substitution is more nonzeros than this value, the substitution is not made. (continuous)

Presolve dual setting Decides whether CPLEX presolve should pass the primal or dual linear programming problem to the linear programming optimization algorithm. By default, CPLEX chooses automatically. (3 level categorical)

Presolve switch Decides whether CPLEX applies presolve during preprocessing. (2 level categorical)

Primal and dual reduction type Decides whether primal reductions, dual reductions, both, or neither are performed during preprocessing. (4 level categorical)

Primal simplex pricing algorithm Sets the primal simplex pricing algorithm. (6 level categorical)

Relaxed LP presolve switch Decides whether LP presolve is applied to the root relaxation in a mixed integer program (MIP). Sometimes additional reductions can be made beyond any MIP presolve reductions that were already done. By default, CPLEX applies presolve to the initial relaxation in order to hasten time to the initial solution. (3 level categorical)

RINS heuristic frequency Decides how often to apply the relaxation induced neighborhood search (RINS) heuristic. This heuristic attempts to improve upon the best solution found so far. It will not be applied until CPLEX has found at least one incumbent solution. (3 level categorical)

Row multiplier factor for cuts Limits the number of cuts that can be added. The number of rows in the problem with cuts added is limited to CutsFactor times the original

number of rows. If the problem is presolved, the original number of rows is that from the presolved problem. (continuous)

Scale parameter Decides how to scale the problem matrix. (3 level categorical)

Simplex crash ordering Decides how CPLEX orders variables relative to the objective function when selecting an initial basis. (3 level categorical)

Simplex perturbation limit Sets the number of degenerate iterations before perturbation is performed. (2 level categorical)

Simplex perturbation switch Setting this parameter to 1 (one) causes all problems to be automatically perturbed as optimization begins. (2 level categorical)

Simplex pricing candidate list size Sets the maximum number of variables kept in the list of pricing candidates for the simplex algorithms. (3 level categorical)

Simplex refactoring frequency Simplex refactoring frequency (3 level categorical)

Simplex singularity repair limit Restricts the number of times CPLEX attempts to repair the basis when singularities are encountered during the simplex algorithm. When this limit is exceeded, CPLEX replaces the current basis with the best factorable basis that has been found. (continuous)

Symmetry breaking Decides whether symmetry breaking reductions will be automatically executed, during the preprocessing phase, in a MIP model. (5 level categorical)

Time spent probing Limits the amount of time in seconds spent probing. (continuous)

Appendix B

List of GLPK Parameters

The 6 GLPK parameters selected for consideration are described here, using information from GLP (2009) and Max (2010).

Scaling process Specifies which scaling method to employ. (4 level categorical)

Pricing strategy Sets the pricing strategy for both the Primal and Dual Simplex algorithms. (2 level categorical)

Variable Selection The variable selection option is used to set the rule for selecting the branching variable. (4 level categorical)

Backtracking method The node selection option is used to set the rule for selecting the next node to process when backtracking. (4 level categorical)

Presolve Sets whether the built-in LP presolver is used. (2 level categorical)

Cutting plane settings Sets what cutting planes are added to the MIP problem. (3 level categorical)

Appendix C

Complete GLPK Second Order Model

This model is of the form

$$\ln y = \beta_0 + \beta_1 x_1 + \ldots, \tag{C.1}$$

where y is the average solution time, β_0 is the intercept, and each remaining term is either an indicator variable on the categorical parameter values indicated, or the unscaled value of the categorical parameter indicated.

Table C.1: Complete Sorted 2nd Order Coefficient Estimates in GLPK Case

Parameter [Value]	Estimate
Intercept	4.707021
MIP Emphasis [1]	-1.028217
MIP Emphasis [0]	-0.836883
MIP Emphasis [2]	-0.827975
MIP Emphasis [2]*Branching Selection [2]	0.744255
MIP Emphasis [2]*Branching Selection [0]	0.741006
MIP Emphasis [1]*Branching Selection [2]	0.529388
Branching Selection [0]	0.366774
MIP Emphasis [0]*Branching Selection [2]	0.332516
MIP Emphasis [0]*Branching Selection [0]	0.320321
Branching Selection [2]	0.303102
Branching Selection [0]*Fractional Cuts [1]	-0.133259
MIP Emphasis [1]*Dive Type [2]	0.125175
Branching Selection [2]*Fractional Cuts [1]	-0.109013
MIP Emphasis [1]*Branching Selection [0]	-0.104109
Branching Selection [2]*Dive Type [1]	0.101834

Continued on Next Page...

Table C.1 – Continued

Parameter [Value]	Estimate
Branching Selection [2]*Fractional Cuts [0]	-0.096703
Node Selection [1]*Fractional Cuts [1]	0.086816
Branching Selection [0]*Fractional Cuts [0]	-0.085105
Branching Selection [0]*Dive Type [1]	0.084964
Dive Type [0]	-0.083438
Branching Selection [0]*Dive Type [0]	0.082245
MIP Emphasis [0]*Node Selection [1]	-0.081682
Node Selection [1]*Dive Type [2]	-0.07942
Branching Selection [2]*Dive Type [2]	0.075756
MIP Emphasis [0]*Fractional Cuts [0]	0.073567
Dive Type [2]*MIR Cuts [0]	-0.073239
Branching Selection [2]*Dive Type [0]	0.072434
MIR Cuts [0]	0.069754
Node Selection [2]*Dive Type [2]	-0.069003
MIP Emphasis [1]*MIR Cuts [1]	0.061076
Node Selection [2]*Dive Type [1]	-0.061007
Node Selection [1]*Fractional Cuts [0]	0.060524
MIP Emphasis [1]*Dive Type [0]	0.054905

Continued on Next Page...

Table C.1 – Continued

Parameter [Value]	Estimate
MIP Emphasis [1]*Node Selection [1]	-0.053777
MIP Emphasis [0]*Dive Type [1]	-0.052698
MIP Emphasis [2]*Node Selection [1]	-0.052586
Node Selection [1]*Dive Type [1]	-0.049379
MIP Emphasis [0]*MIR Cuts [0]	-0.048823
Node Selection [1]*Branching Selection [0]	-0.048439
Node Selection [2]	0.048264
MIR Cuts [1]	-0.047273
Branching Selection [0]*Dive Type [2]	0.046615
MIP Emphasis [2]*Dive Type [0]	0.045919
MIP Emphasis [0]*Node Selection [2]	-0.045917
MIP Emphasis [0]*Fractional Cuts [1]	0.044416
MIP Emphasis [1]*MIR Cuts [0]	-0.043223
Dive Type [1]*Fractional Cuts [0]	0.041559
MIP Emphasis [2]*Dive Type [1]	-0.040355
MIP Emphasis [2]*Fractional Cuts [0]	0.039816
MIP Emphasis [1]*Dive Type [1]	0.039389
MIP Emphasis [2]*Node Selection [2]	-0.038711

Continued on Next Page...

Table C.1 – Continued

Parameter [Value]	Estimate
Node Selection [2]*MIR Cuts [0]	-0.037064
Node Selection [1]*MIR Cuts [1]	0.036217
Branching Selection [2]*MIR Cuts [1]	0.035069
Node Selection [2]*Branching Selection [0]	-0.03442
Fractional Cuts [0]	0.033986
Fractional Cuts [1]	0.033098
Node Selection [2]*Fractional Cuts [1]	0.032095
MIP Emphasis [2]*Fractional Cuts [1]	0.031673
Fractional Cuts [1]*MIR Cuts [0]	-0.030667
Node Selection [2]*Fractional Cuts [0]	0.029566
MIP Emphasis [1]*Fractional Cuts [0]	0.028589
Node Selection [1]*MIR Cuts [0]	0.028158
Fractional Cuts [0]*MIR Cuts [1]	0.021895
MIP Emphasis [0]*Dive Type [0]	0.021618
Node Selection [2]*MIR Cuts [1]	0.018873
Dive Type [0]*Fractional Cuts [1]	-0.017807
MIP Emphasis [2]*Dive Type [2]	0.017196
MIP Emphasis [1]*Node Selection [2]	-0.016586

Continued on Next Page...

Table C.1 – Continued

Parameter [Value]	Estimate
Branching Selection [2]*MIR Cuts [0]	-0.016052
Node Selection [1]*Dive Type [0]	0.015854
Dive Type [1]	-0.015845
MIP Emphasis [0]*MIR Cuts [1]	-0.013608
Dive Type [2]*MIR Cuts [1]	-0.012595
Node Selection [1]	0.011065
Node Selection [1]*Branching Selection [2]	0.011058
Fractional Cuts [1]*MIR Cuts [1]	-0.010216
Dive Type [0]*Fractional Cuts [0]	-0.008969
MIP Emphasis [2]*MIR Cuts [0]	-0.008417
Dive Type [0]*MIR Cuts [1]	-0.007506
Fractional Cuts [0]*MIR Cuts [0]	-0.006928
MIP Emphasis [1]*Fractional Cuts [1]	-0.006621
Dive Type [1]*Fractional Cuts [1]	0.006254
Dive Type [2]*Fractional Cuts [1]	0.00608
Dive Type [1]*MIR Cuts [0]	-0.005569
Dive Type [1]*MIR Cuts [1]	-0.004434
Dive Type [2]*Fractional Cuts [0]	0.004246

Continued on Next Page...

Table C.1 – Continued

Parameter [Value]	Estimate
MIP Emphasis [2]*MIR Cuts [1]	0.003715
Node Selection [2]*Dive Type [0]	0.002711
Branching Selection [0]*MIR Cuts [1]	0.002623
Node Selection [2]*Branching Selection [2]	-0.001064
Branching Selection [0]*MIR Cuts [0]	-0.000921
Dive Type [0]*MIR Cuts [0]	0.000454
Dive Type [2]	0.000175
MIP Emphasis [0]*Dive Type [2]	0.000152

Appendix D

Complete Limited CPLEX Case

Second Order Model

This model is of the form

$$\ln y = \beta_0 + \beta_1 x_1 + \ldots, \tag{D.1}$$

where y is the average solution time, β_0 is the intercept, and each remaining term is either an indicator variable on the categorical parameter values indicated, or the unscaled value of the categorical parameter indicated.

Table D.1: Complete Sorted 2nd Order Coefficient Estimates in Limited CPLEX Case

Parameter [Value]	Estimate
Intercept	0.60935
MIP Emphasis [1]	-1.028217
MIP Emphasis [0]	-0.836883
MIP Emphasis [2]	-0.827975
MIP Emphasis [2]*Branching Selection [2]	0.744255
MIP Emphasis [2]*Branching Selection [0]	0.741006
MIP Emphasis [1]*Branching Selection [2]	0.529388
Branching Selection [0]	0.366774
MIP Emphasis [0]*Branching Selection [2]	0.332516
MIP Emphasis [0]*Branching Selection [0]	0.320321
Branching Selection [2]	0.303102
Branching Selection [0]*Fractional Cuts [1]	-0.133259
MIP Emphasis [1]*Dive Type [2]	0.125175
Branching Selection [2]*Fractional Cuts [1]	-0.109013
MIP Emphasis [1]*Branching Selection [0]	-0.104109
Branching Selection [2]*Dive Type [1]	0.101834

Continued on Next Page...

Table D.1 – Continued

Parameter [Value]	Estimate
Branching Selection [2]*Fractional Cuts [0]	-0.096703
Node Selection [1]*Fractional Cuts [1]	0.086816
Branching Selection [0]*Fractional Cuts [0]	-0.085105
Branching Selection [0]*Dive Type [1]	0.084964
Dive Type [0]	-0.083438
Branching Selection [0]*Dive Type [0]	0.082245
MIP Emphasis [0]*Node Selection [1]	-0.081682
Node Selection [1]*Dive Type [2]	-0.07942
Branching Selection [2]*Dive Type [2]	0.075756
MIP Emphasis [0]*Fractional Cuts [0]	0.073567
Dive Type [2]*MIR Cuts [0]	-0.073239
Branching Selection [2]*Dive Type [0]	0.072434
MIR Cuts [0]	0.069754
Node Selection [2]*Dive Type [2]	-0.069003
MIP Emphasis [1]*MIR Cuts [1]	0.061076
Node Selection [2]*Dive Type [1]	-0.061007
Node Selection [1]*Fractional Cuts [0]	0.060524
MIP Emphasis [1]*Dive Type [0]	0.054905

Continued on Next Page...

Table D.1 – Continued

Parameter [Value]	Estimate
MIP Emphasis [1]*Node Selection [1]	-0.053777
MIP Emphasis [0]*Dive Type [1]	-0.052698
MIP Emphasis [2]*Node Selection [1]	-0.052586
Node Selection [1]*Dive Type [1]	-0.049379
MIP Emphasis [0]*MIR Cuts [0]	-0.048823
Node Selection [1]*Branching Selection [0]	-0.048439
Node Selection [2]	0.048264
MIR Cuts [1]	-0.047273
Branching Selection [0]*Dive Type [2]	0.046615
MIP Emphasis [2]*Dive Type [0]	0.045919
MIP Emphasis [0]*Node Selection [2]	-0.045917
MIP Emphasis [0]*Fractional Cuts [1]	0.044416
MIP Emphasis [1]*MIR Cuts [0]	-0.043223
Dive Type [1]*Fractional Cuts [0]	0.041559
MIP Emphasis [2]*Dive Type [1]	-0.040355
MIP Emphasis [2]*Fractional Cuts [0]	0.039816
MIP Emphasis [1]*Dive Type [1]	0.039389
MIP Emphasis [2]*Node Selection [2]	-0.038711

Continued on Next Page...

Table D.1 – Continued

Parameter [Value]	Estimate
Node Selection [2]*MIR Cuts [0]	-0.037064
Node Selection [1]*MIR Cuts [1]	0.036217
Branching Selection [2]*MIR Cuts [1]	0.035069
Node Selection [2]*Branching Selection [0]	-0.03442
Fractional Cuts [0]	0.033986
Fractional Cuts [1]	0.033098
Node Selection [2]*Fractional Cuts [1]	0.032095
MIP Emphasis [2]*Fractional Cuts [1]	0.031673
Fractional Cuts [1]*MIR Cuts [0]	-0.030667
Node Selection [2]*Fractional Cuts [0]	0.029566
MIP Emphasis [1]*Fractional Cuts [0]	0.028589
Node Selection [1]*MIR Cuts [0]	0.028158
Fractional Cuts [0]*MIR Cuts [1]	0.021895
MIP Emphasis [0]*Dive Type [0]	0.021618
Node Selection [2]*MIR Cuts [1]	0.018873
Dive Type [0]*Fractional Cuts [1]	-0.017807
MIP Emphasis [2]*Dive Type [2]	0.017196
MIP Emphasis [1]*Node Selection [2]	-0.016586

Continued on Next Page...

Table D.1 – Continued

Parameter [Value]	Estimate
Branching Selection [2]*MIR Cuts [0]	-0.016052
Node Selection [1]*Dive Type [0]	0.015854
Dive Type [1]	-0.015845
MIP Emphasis [0]*MIR Cuts [1]	-0.013608
Dive Type [2]*MIR Cuts [1]	-0.012595
Node Selection [1]	0.011065
Node Selection [1]*Branching Selection [2]	0.011058
Fractional Cuts [1]*MIR Cuts [1]	-0.010216
Dive Type [0]*Fractional Cuts [0]	-0.008969
MIP Emphasis [2]*MIR Cuts [0]	-0.008417
Dive Type [0]*MIR Cuts [1]	-0.007506
Fractional Cuts [0]*MIR Cuts [0]	-0.006928
MIP Emphasis [1]*Fractional Cuts [1]	-0.006621
Dive Type [1]*Fractional Cuts [1]	0.006254
Dive Type [2]*Fractional Cuts [1]	0.00608
Dive Type [1]*MIR Cuts [0]	-0.005569
Dive Type [1]*MIR Cuts [1]	-0.004434
Dive Type [2]*Fractional Cuts [0]	0.004246

Continued on Next Page...

Table D.1 – Continued

Parameter [Value]	Estimate
MIP Emphasis [2]*MIR Cuts [1]	0.003715
Node Selection [2]*Dive Type [0]	0.002711
Branching Selection [0]*MIR Cuts [1]	0.002623
Node Selection [2]*Branching Selection [2]	-0.001064
Branching Selection [0]*MIR Cuts [0]	-0.000921
Dive Type [0]*MIR Cuts [0]	0.000454
Dive Type [2]	0.000175
MIP Emphasis [0]*Dive Type [2]	0.000152

Appendix E

Complete Full CPLEX Case in Telecommunications Instances Second Order Model

This model is of the form

$$\ln y = \beta_0 + \beta_1 x_1 + \ldots, \tag{E.1}$$

where y is the average solution time, β_0 is the intercept, and each remaining term is either an indicator variable on the categorical parameter values indicated, or the unscaled value of the categorical parameter indicated.

Table E.1: Complete Sorted 2nd Order Coefficient Estimates in Full CPLEX Case - Telecommunications Instance

Parameter [Value]	Estimate
Intercept	4.777063
MIP subproblem algorithm [1]	-4.150064
MIP subproblem algorithm [2]	-3.093273
MIP subproblem algorithm [3]*MIP variable selection strategy [0]	2.783764
MIP subproblem algorithm [3]*MIP variable selection strategy [4]	2.544374
MIP subproblem algorithm [3]*MIP variable selection strategy [2]	2.543261
MIP subproblem algorithm [3]*MIP variable selection strategy [3]	2.531492
Simplex refactoring frequency [10]*MIP subproblem algorithm [4]	-2.430574
Dual simplex pricing algorithm [2]*MIP variable selection strategy [2]	2.33667
Primal and dual reduction type [2]*MIP variable selection strategy [1]	-2.285881
MIP subproblem algorithm [2]*MIP variable selection strategy [4]	2.279372
Presolve switch [0]*MIP subproblem algorithm [1]	2.265423
MIP disjunctive cuts switch [1]*MIP subproblem algorithm [4]	-2.249864
MIP subproblem algorithm [1]*MIP variable selection strategy [3]	2.236945
Simplex refactoring frequency [10]*MIP subproblem algorithm [2]	-2.2266

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
MIP subproblem algorithm [1]*MIP variable selection strategy [4]	2.220872
MIP subproblem algorithm [1]*MIP variable selection strategy [2]	2.195133
Dual simplex pricing algorithm [0]	-2.166168
Dual simplex pricing algorithm [2]*MIP subproblem algorithm [0]	-2.161117
MIP disjunctive cuts switch [3]*MIP subproblem algorithm [2]	2.061012
Dual simplex pricing algorithm [2]*MIP subproblem algorithm [1]	-2.011317
MIP disjunctive cuts switch [2]*Simplex refactoring frequency [10]	-2.009259
MIP disjunctive cuts switch [0]*MIP variable selection strategy [3]	-1.949516
MIP disjunctive cuts switch [0]*MIP subproblem algorithm [4]	-1.944943
MIP disjunctive cuts switch [1]*MIP variable selection strategy [4]	-1.931502
MIP disjunctive cuts switch [0]*MIP variable selection strategy [2]	-1.92958
MIP subproblem algorithm [3]*MIP variable selection strategy [1]	1.92768
MIP subproblem algorithm [3]	-1.86544
Dual simplex pricing algorithm [0]*MIP subproblem algorithm [2]	-1.843451
Dual simplex pricing algorithm [2]*MIP variable selection strategy [0]	1.815714
Dual simplex pricing algorithm [4]*Primal and dual reduction type [0]	-1.814076
Simplex refactoring frequency [10]*MIP subproblem algorithm [0]	-1.80147
MIP subproblem algorithm [1]*MIP variable selection strategy [1]	1.795146

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
Dual simplex pricing algorithm [3]*MIP subproblem algorithm [2]	-1.790257
MIP variable selection strategy [0]	-1.779069
Dual simplex pricing algorithm [4]*MIP variable selection strategy [3]	-1.775963
MIP disjunctive cuts switch [1]*Dual simplex pricing algorithm [4]	-1.771442
MIP subproblem algorithm [2]*MIP variable selection strategy [0]	1.732482
Dual simplex pricing algorithm [0]*MIP subproblem algorithm [0]	-1.625144
Primal and dual reduction type [1]	-1.612863
MIP disjunctive cuts switch [0]	1.594382
Dual simplex pricing algorithm [1]	-1.587414
Dual simplex pricing algorithm [2]*MIP variable selection strategy [1]	1.586874
MIP disjunctive cuts switch [0]*MIP variable selection strategy [1]	-1.586181
MIP subproblem algorithm [2]*MIP variable selection strategy [3]	1.56234
Dual simplex pricing algorithm [3]*MIP subproblem algorithm [0]	-1.5241
MIP disjunctive cuts switch [0]*MIP subproblem algorithm [0]	-1.521889
Dual simplex pricing algorithm [2]*MIP variable selection strategy [3]	1.515264
Dual simplex pricing algorithm [1]*MIP variable selection strategy [2]	1.502955
MIP disjunctive cuts switch [2]*MIP subproblem algorithm [4]	-1.491945
Dual simplex pricing algorithm [2]	-1.472351

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
MIP disjunctive cuts switch [3]*MIP variable selection strategy [1]	-1.471977
Dual simplex pricing algorithm [4]*MIP subproblem algorithm [1]	-1.470478
Primal and dual reduction type [2]*MIP variable selection strategy [2]	-1.453753
Dual simplex pricing algorithm [2]*MIP subproblem algorithm [3]	-1.448395
MIP disjunctive cuts switch [2]*MIP subproblem algorithm [0]	-1.427034
Dual simplex pricing algorithm [4]*MIP priority order switch [0]	1.421934
MIP disjunctive cuts switch [2]*MIP subproblem algorithm [3]	-1.4107
MIP disjunctive cuts switch [3]*MIP subproblem algorithm [1]	1.396015
MIP variable selection strategy [3]	-1.395375
MIP subproblem algorithm [0]*MIP variable selection strategy [4]	1.383465
Dual simplex pricing algorithm [3]*MIP subproblem algorithm [3]	-1.37474
Dual simplex pricing algorithm [2]*Presolve switch [0]	1.335194
MIP disjunctive cuts switch [2]*MIP variable selection strategy [1]	-1.334378
Dual simplex pricing algorithm [3]*MIP subproblem algorithm [1]	-1.323895
Presolve switch [0]	-1.316385
MIP subproblem algorithm [1]*MIP variable selection strategy [0]	1.308678
Simplex refactoring frequency [0]*MIP subproblem algorithm [3]	-1.292103
MIP disjunctive cuts switch [3]*Dual simplex pricing algorithm [1]	1.28852

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
Simplex refactoring frequency [0]*MIP subproblem algorithm [4]	-1.285697
MIP disjunctive cuts switch [1]*MIP variable selection strategy [2]	-1.244512
MIP disjunctive cuts switch [2]*Primal and dual reduction type [0]	1.234708
Dual simplex pricing algorithm [1]*Simplex refactoring frequency [10]	1.221589
Dual simplex pricing algorithm [3]*MIP subproblem algorithm [4]	-1.221539
Primal and dual reduction type [2]*MIP variable selection strategy [0]	-1.211632
MIP disjunctive cuts switch [3]*Dual simplex pricing algorithm [0]	1.210807
Dual simplex pricing algorithm [2]*Primal and dual reduction type [0]	-1.209897
MIP disjunctive cuts switch [2]*MIP variable selection strategy [2]	-1.208204
Dual simplex pricing algorithm [0]*Presolve switch [0]	1.181433
MIP disjunctive cuts switch [0]*Dual simplex pricing algorithm [4]	-1.148705
Presolve switch [0]*MIP subproblem algorithm [2]	1.147694
MIP variable selection strategy [1]	1.105296
MIP disjunctive cuts switch [1]*MIP subproblem algorithm [3]	-1.102286
MIP disjunctive cuts switch [0]*Primal and dual reduction type [1]	1.095289
MIP disjunctive cuts switch [3]*Dual simplex pricing algorithm [4]	-1.08507
MIP disjunctive cuts switch [0]*Simplex refactoring frequency [10]	-1.070829
MIP disjunctive cuts switch [2]*Dual simplex pricing algorithm [3]	1.068482

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
Dual simplex pricing algorithm [2]*Simplex refactoring frequency [10]	1.060429
Dual simplex pricing algorithm [0]*MIP variable selection strategy [1]	1.05844
MIP disjunctive cuts switch [1]*Dual simplex pricing algorithm [1]	1.051353
Primal and dual reduction type [1]*MIP variable selection strategy [2]	-1.042395
Simplex refactoring frequency [0]	1.02936
MIP disjunctive cuts switch [0]*MIP subproblem algorithm [3]	-1.022852
MIP disjunctive cuts switch [0]*MIP variable selection strategy [0]	-1.007028
MIP disjunctive cuts switch [2]*Simplex refactoring frequency [0]	-0.992155
Primal and dual reduction type [1]*Simplex refactoring frequency [10]	0.986081
MIP subproblem algorithm [0]*MIP variable selection strategy [1]	-0.97728
Primal and dual reduction type [0]*MIP variable selection strategy [0]	-0.9708
Dual simplex pricing algorithm [0]*Primal and dual reduction type [1]	0.961742
Presolve switch [0]*MIP subproblem algorithm [3]	0.958763
Simplex refactoring frequency [10]*MIP subproblem algorithm [3]	-0.957671
Primal and dual reduction type [0]*MIP variable selection strategy [2]	-0.955916
MIP disjunctive cuts switch [2]*Dual simplex pricing algorithm [2]	0.938676
MIP disjunctive cuts switch [2]*MIP subproblem algorithm [1]	-0.938357
Primal and dual reduction type [2]*Simplex refactoring frequency [10]	0.926141

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
Primal and dual reduction type [0]*MIP variable selection strategy [4]	-0.924248
MIP subproblem algorithm [2]*MIP variable selection strategy [1]	0.895234
MIP disjunctive cuts switch [1]*Dual simplex pricing algorithm [2]	0.884651
MIP disjunctive cuts switch [1]*MIP subproblem algorithm [0]	-0.883456
Dual simplex pricing algorithm [1]*MIP variable selection strategy [0]	0.876526
Dual simplex pricing algorithm [0]*MIP variable selection strategy [0]	0.873237
MIP subproblem algorithm [2]*MIP variable selection strategy [2]	0.849647
Dual simplex pricing algorithm [0]*MIP variable selection strategy [3]	0.84164
Presolve switch [0]*Primal and dual reduction type [1]	0.84081
Simplex refactoring frequency [10]	0.840166
Dual simplex pricing algorithm [0]*Primal and dual reduction type [2]	0.836044
MIP disjunctive cuts switch [1]	0.831131
MIP subproblem algorithm [0]	-0.822737
Dual simplex pricing algorithm [1]*Primal and dual reduction type [1]	0.820065
Dual simplex pricing algorithm [4]*Simplex refactoring frequency [0]	-0.794072
MIP disjunctive cuts switch [3]	0.790673
MIP disjunctive cuts switch [2]*MIP subproblem algorithm [2]	0.788174
MIP subproblem algorithm [4]	0.788084

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
MIP priority order switch [0]*Simplex refactoring frequency [0]	-0.788079
MIP disjunctive cuts switch [1]*Dual simplex pricing algorithm [0]	0.785428
Primal and dual reduction type [2]*MIP subproblem algorithm [2]	0.783891
Dual simplex pricing algorithm [4]	0.783428
MIP disjunctive cuts switch [0]*MIP priority order switch [0]	0.780525
Dual simplex pricing algorithm [1]*MIP priority order switch [0]	0.77887
Dual simplex pricing algorithm [1]*MIP variable selection strategy [1]	0.775365
MIP disjunctive cuts switch [1]*MIP subproblem algorithm [2]	0.76987
MIP disjunctive cuts switch [2]*MIP priority order switch [0]	0.76497
Primal and dual reduction type [0]*MIP subproblem algorithm [2]	0.760073
Dual simplex pricing algorithm [2]*MIP variable selection strategy [4]	0.757846
MIP disjunctive cuts switch [2]*Primal and dual reduction type [1]	0.757697
Primal and dual reduction type [2]	0.756334
Primal and dual reduction type [0]*MIP variable selection strategy [1]	-0.738037
MIP disjunctive cuts switch [2]*Dual simplex pricing algorithm [0]	0.731777
Simplex refactoring frequency [10]*MIP variable selection strategy [1]	-0.729352
Simplex refactoring frequency [0]*MIP subproblem algorithm [2]	-0.728826
MIP disjunctive cuts switch [3]*Primal and dual reduction type [1]	0.728656

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
MIP disjunctive cuts switch [0]*Simplex refactoring frequency [0]	-0.72038
MIP subproblem algorithm [0]*MIP variable selection strategy [0]	0.715735
MIP priority order switch [0]*MIP variable selection strategy [4]	-0.712885
Primal and dual reduction type [1]*MIP variable selection strategy [0]	-0.708819
Primal and dual reduction type [0]*Simplex refactoring frequency [0]	-0.701052
Primal and dual reduction type [1]*Simplex refactoring frequency [0]	0.697824
MIP priority order switch [0]*Primal and dual reduction type [0]	0.68535
MIP priority order switch [0]	-0.678042
MIP disjunctive cuts switch [3]*Dual simplex pricing algorithm [3]	0.669728
Dual simplex pricing algorithm [2]*Simplex refactoring frequency [0]	-0.668246
Primal and dual reduction type [2]*MIP variable selection strategy [4]	-0.666252
MIP disjunctive cuts switch [1]*MIP variable selection strategy [1]	-0.663295
MIP disjunctive cuts switch [1]*Simplex refactoring frequency [0]	-0.660622
Simplex refactoring frequency [10]*MIP variable selection strategy [0]	-0.659792
Presolve switch [0]*MIP subproblem algorithm [0]	0.658047
Dual simplex pricing algorithm [0]*MIP variable selection strategy [2]	0.655541
Presolve switch [0]*MIP variable selection strategy [4]	-0.653374
MIP disjunctive cuts switch [3]*MIP variable selection strategy [2]	-0.650834

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
Primal and dual reduction type [0]*MIP subproblem algorithm [0]	0.650112
MIP disjunctive cuts switch [1]*Primal and dual reduction type [0]	0.642269
Dual simplex pricing algorithm [2]*MIP subproblem algorithm [4]	-0.637629
Simplex refactoring frequency [0]*MIP subproblem algorithm [0]	-0.632647
MIP variable selection strategy [4]	-0.626014
Dual simplex pricing algorithm [1]*MIP subproblem algorithm [3]	0.619366
MIP disjunctive cuts switch [1]*Primal and dual reduction type [2]	0.617794
MIP disjunctive cuts switch [3]*Simplex refactoring frequency [10]	-0.617723
Primal and dual reduction type [0]*MIP subproblem algorithm [3]	0.606476
Presolve switch [0]*MIP variable selection strategy [0]	0.598149
MIP disjunctive cuts switch [1]*MIP variable selection strategy [0]	0.590155
Simplex refactoring frequency [10]*MIP variable selection strategy [2]	-0.576424
MIP disjunctive cuts switch [0]*Primal and dual reduction type [2]	-0.573381
MIP disjunctive cuts switch [0]*MIP variable selection strategy [4]	-0.57084
Dual simplex pricing algorithm [0]*Simplex refactoring frequency [10]	0.568652
Primal and dual reduction type [1]*MIP subproblem algorithm [0]	0.565889
MIP disjunctive cuts switch [3]*MIP priority order switch [0]	0.564969
Dual simplex pricing algorithm [1]*Primal and dual reduction type [2]	0.550937

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
Dual simplex pricing algorithm [1]*Presolve switch [0]	0.549853
MIP disjunctive cuts switch [2]*Primal and dual reduction type [2]	-0.548359
Dual simplex pricing algorithm [1]*MIP subproblem algorithm [0]	-0.546487
Dual simplex pricing algorithm [4]*MIP subproblem algorithm [0]	0.541198
Presolve switch [0]*Simplex refactoring frequency [0]	0.541009
MIP disjunctive cuts switch [3]*MIP subproblem algorithm [0]	0.521742
MIP disjunctive cuts switch [3]*MIP variable selection strategy [4]	0.521458
MIP subproblem algorithm [0]*MIP variable selection strategy [3]	0.510262
Dual simplex pricing algorithm [4]*MIP subproblem algorithm [3]	0.507548
Dual simplex pricing algorithm [4]*Simplex refactoring frequency [10]	0.506659
Dual simplex pricing algorithm [2]*MIP subproblem algorithm [2]	-0.500272
Dual simplex pricing algorithm [4]*Presolve switch [0]	0.498711
MIP subproblem algorithm [4]*MIP variable selection strategy [2]	-0.497825
MIP disjunctive cuts switch [2]	0.490765
MIP disjunctive cuts switch [0]*Primal and dual reduction type [0]	0.490631
Primal and dual reduction type [1]*MIP subproblem algorithm [4]	0.480195
MIP priority order switch [0]*Primal and dual reduction type [2]	0.473161
MIP disjunctive cuts switch [3]*Presolve switch [0]	-0.464772

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
Primal and dual reduction type [1]*MIP subproblem algorithm [1]	-0.460404
Simplex refactoring frequency [10]*MIP variable selection strategy [4]	-0.458014
Dual simplex pricing algorithm [3]*MIP variable selection strategy [0]	0.457767
MIP priority order switch [0]*Presolve switch [0]	-0.457527
MIP disjunctive cuts switch [1]*Dual simplex pricing algorithm [3]	0.45708
Dual simplex pricing algorithm [0]*Simplex refactoring frequency [0]	-0.45435
Primal and dual reduction type [2]*MIP subproblem algorithm [4]	-0.443533
MIP priority order switch [0]*MIP subproblem algorithm [1]	0.440119
Dual simplex pricing algorithm [0]*MIP subproblem algorithm [4]	-0.437537
Primal and dual reduction type [2]*MIP subproblem algorithm [1]	0.436539
Simplex refactoring frequency [0]*MIP variable selection strategy [4]	-0.420217
MIP priority order switch [0]*Simplex refactoring frequency [10]	0.41872
MIP disjunctive cuts switch [0]*Dual simplex pricing algorithm [2]	0.413455
Simplex refactoring frequency [0]*MIP variable selection strategy [3]	-0.41131
MIP disjunctive cuts switch [3]*Primal and dual reduction type [2]	-0.409835
Primal and dual reduction type [0]*MIP variable selection strategy [3]	-0.402312
Dual simplex pricing algorithm [3]*MIP variable selection strategy [2]	0.401961
MIP disjunctive cuts switch [2]*Presolve switch [0]	0.400383

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
Dual simplex pricing algorithm [3]*MIP variable selection strategy [1]	0.400278
MIP disjunctive cuts switch [0]*Dual simplex pricing algorithm [3]	0.395244
Dual simplex pricing algorithm [0]*MIP priority order switch [0]	0.37734
Primal and dual reduction type [2]*MIP variable selection strategy [3]	-0.375854
Dual simplex pricing algorithm [3]*Presolve switch [0]	0.37332
Presolve switch [0]*Primal and dual reduction type [0]	0.372295
Dual simplex pricing algorithm [4]*MIP variable selection strategy [1]	0.369923
Dual simplex pricing algorithm [1]*MIP variable selection strategy [3]	-0.368721
Dual simplex pricing algorithm [4]*Primal and dual reduction type [2]	-0.368689
Dual simplex pricing algorithm [2]*Primal and dual reduction type [2]	-0.367162
MIP disjunctive cuts switch [0]*Dual simplex pricing algorithm [1]	-0.366588
MIP priority order switch [0]*MIP variable selection strategy [2]	-0.365733
MIP subproblem algorithm [4]*MIP variable selection strategy [3]	0.362052
MIP disjunctive cuts switch [3]*Simplex refactoring frequency [0]	0.360261
MIP disjunctive cuts switch [0]*MIP subproblem algorithm [2]	-0.348978
Presolve switch [0]*Simplex refactoring frequency [10]	0.344228
Dual simplex pricing algorithm [1]*MIP subproblem algorithm [4]	0.341501
MIP disjunctive cuts switch [2]*MIP variable selection strategy [3]	0.339859

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
Primal and dual reduction type [2]*Simplex refactoring frequency [0]	-0.338825
MIP disjunctive cuts switch [0]*Presolve switch [0]	0.329149
Primal and dual reduction type [1]*MIP subproblem algorithm [2]	0.327023
MIP disjunctive cuts switch [2]*Dual simplex pricing algorithm [1]	0.32083
Presolve switch [0]*Primal and dual reduction type [2]	0.318855
MIP priority order switch [0]*MIP variable selection strategy [3]	-0.317236
Primal and dual reduction type [0]	0.316662
Primal and dual reduction type [2]*MIP subproblem algorithm [0]	-0.312748
Dual simplex pricing algorithm [2]*MIP priority order switch [0]	0.311065
Dual simplex pricing algorithm [3]*MIP variable selection strategy [4]	-0.308511
Dual simplex pricing algorithm [3]*Simplex refactoring frequency [10]	0.304908
MIP priority order switch [0]*MIP subproblem algorithm [4]	0.299688
Dual simplex pricing algorithm [3]*MIP priority order switch [0]	0.297893
MIP disjunctive cuts switch [3]*MIP variable selection strategy [3]	0.29758
MIP disjunctive cuts switch [1]*Simplex refactoring frequency [10]	0.288008
Primal and dual reduction type [2]*MIP subproblem algorithm [3]	0.286864
Dual simplex pricing algorithm [3]*MIP variable selection strategy [3]	-0.279526
Presolve switch [0]*MIP variable selection strategy [3]	-0.277475

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
Dual simplex pricing algorithm [4]*MIP subproblem algorithm [2]	-0.275574
MIP disjunctive cuts switch [3]*Primal and dual reduction type [0]	-0.267556
MIP disjunctive cuts switch [2]*MIP variable selection strategy [4]	0.253973
Primal and dual reduction type [0]*Simplex refactoring frequency [10]	0.252261
Dual simplex pricing algorithm [0]*MIP variable selection strategy [4]	-0.249984
MIP priority order switch [0]*MIP subproblem algorithm [0]	0.244308
Dual simplex pricing algorithm [3]*Simplex refactoring frequency [0]	-0.244301
Dual simplex pricing algorithm [1]*Primal and dual reduction type [0]	0.241932
MIP disjunctive cuts switch [3]*MIP subproblem algorithm [3]	-0.235185
MIP priority order switch [0]*Primal and dual reduction type [1]	0.234788
Dual simplex pricing algorithm [4]*MIP variable selection strategy [4]	-0.232193
Dual simplex pricing algorithm [0]*Primal and dual reduction type [0]	0.231307
Dual simplex pricing algorithm [1]*MIP subproblem algorithm [2]	-0.223375
Primal and dual reduction type [1]*MIP variable selection strategy [1]	0.216737
MIP priority order switch [0]*MIP subproblem algorithm [2]	-0.215408
MIP disjunctive cuts switch [0]*Dual simplex pricing algorithm [0]	0.214974
Dual simplex pricing algorithm [3]*Primal and dual reduction type [1]	-0.214259
MIP subproblem algorithm [4]*MIP variable selection strategy [1]	-0.213028

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
MIP disjunctive cuts switch [3]*MIP variable selection strategy [0]	-0.202419
Dual simplex pricing algorithm [4]*MIP variable selection strategy [2]	-0.199425
MIP disjunctive cuts switch [2]*MIP variable selection strategy [0]	0.196407
Simplex refactoring frequency [0]*MIP subproblem algorithm [1]	0.195661
Primal and dual reduction type [0]*MIP subproblem algorithm [4]	0.180113
Primal and dual reduction type [1]*MIP subproblem algorithm [3]	-0.178077
Dual simplex pricing algorithm [0]*MIP subproblem algorithm [1]	-0.174618
Simplex refactoring frequency [0]*MIP variable selection strategy [0]	-0.168967
MIP disjunctive cuts switch [1]*MIP variable selection strategy [3]	0.161569
Dual simplex pricing algorithm [0]*MIP subproblem algorithm [3]	-0.157771
MIP disjunctive cuts switch [3]*MIP subproblem algorithm [4]	0.150515
Simplex refactoring frequency [0]*MIP variable selection strategy [1]	0.148478
MIP subproblem algorithm [0]*MIP variable selection strategy [2]	-0.140629
Presolve switch [0]*MIP variable selection strategy [2]	0.131363
Simplex refactoring frequency [10]*MIP variable selection strategy [3]	0.128036
Presolve switch [0]*MIP variable selection strategy [1]	-0.126722
Dual simplex pricing algorithm [4]*MIP variable selection strategy [0]	0.12051
MIP subproblem algorithm [4]*MIP variable selection strategy [0]	0.113814

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
MIP disjunctive cuts switch [1]*MIP subproblem algorithm [1]	-0.109921
Dual simplex pricing algorithm [4]*Primal and dual reduction type [1]	-0.108423
Presolve switch [0]*MIP subproblem algorithm [4]	0.107975
Dual simplex pricing algorithm [1]*Simplex refactoring frequency [0]	0.104131
Primal and dual reduction type [1]*MIP variable selection strategy [4]	-0.089863
MIP disjunctive cuts switch [1]*Presolve switch [0]	-0.086418
Dual simplex pricing algorithm [3]*Primal and dual reduction type [0]	0.086056
MIP priority order switch [0]*MIP subproblem algorithm [3]	0.084703
Dual simplex pricing algorithm [1]*MIP variable selection strategy [4]	0.084152
MIP disjunctive cuts switch [2]*Dual simplex pricing algorithm [4]	0.081639
MIP disjunctive cuts switch [3]*Dual simplex pricing algorithm [2]	-0.076679
Simplex refactoring frequency [0]*MIP variable selection strategy [2]	-0.071916
Dual simplex pricing algorithm [4]*MIP subproblem algorithm [4]	-0.070806
Dual simplex pricing algorithm [3]	0.070739
MIP subproblem algorithm [4]*MIP variable selection strategy [4]	-0.070587
MIP variable selection strategy [2]	-0.068198
MIP disjunctive cuts switch [1]*Primal and dual reduction type [1]	0.065911
Dual simplex pricing algorithm [3]*Primal and dual reduction type [2]	-0.056535

Continued on Next Page...

Table E.1 – Continued

Parameter [Value]	Estimate
Dual simplex pricing algorithm [2]*Primal and dual reduction type [1]	0.04556
Primal and dual reduction type [1]*MIP variable selection strategy [3]	0.043924
MIP priority order switch [0]*MIP variable selection strategy [0]	-0.037842
MIP disjunctive cuts switch [0]*MIP subproblem algorithm [1]	-0.035784
MIP priority order switch [0]*MIP variable selection strategy [1]	-0.030335
Simplex refactoring frequency [10]*MIP subproblem algorithm [1]	0.028402
Dual simplex pricing algorithm [1]*MIP subproblem algorithm [1]	0.014452
MIP disjunctive cuts switch [1]*MIP priority order switch [0]	-0.010424
Primal and dual reduction type [0]*MIP subproblem algorithm [1]	-0.008189

Appendix F

Complete Full CPLEX Case in Cellular Instances Second Order Model

This model is of the form

$$y = \beta_0 + \beta_1 x_1 + \ldots, \tag{F.1}$$

where y is the average gap, β_0 is the intercept, and each remaining term is either an indicator variable on the categorical parameter values indicated, or the unscaled value of the categorical parameter indicated.

Table F.1: Complete Sorted 2nd Order Coefficient Estimates in Full CPLEX Case - Cellular Instance

Parameter [Value]	Estimate
Intercept	9.274482
MIP subproblem algorithm [3]*Primal simplex pricing algorithm [4]	1.854123
MIP subproblem algorithm [2]*Primal simplex pricing algorithm [4]	1.539124
MIP subproblem algorithm [0]*Primal simplex pricing algorithm [4]	1.153275
MIP subproblem algorithm [1]*Primal simplex pricing algorithm [4]	1.055512
Primal simplex pricing algorithm [4]	-1.015214
MIP subproblem algorithm [3]*Primal simplex pricing algorithm [2]	1.008815
MIP subproblem algorithm [4]*Primal simplex pricing algorithm [4]	0.986451
MIP subproblem algorithm [4]*Primal simplex pricing algorithm [2]	0.65368
Dual simplex pricing algorithm [3]*Primal simplex pricing algorithm [3]	0.60813
MIP subproblem algorithm [1]*Primal simplex pricing algorithm [3]	-0.599896
Dual simplex pricing algorithm [1]*Primal simplex pricing algorithm [1]	-0.56211
Dual simplex pricing algorithm [0]*Primal simplex pricing algorithm [4]	-0.548466
MIP subproblem algorithm [2]*Primal simplex pricing algorithm [2]	0.509155
Dual simplex pricing algorithm [2]*Primal simplex pricing algorithm [3]	0.453119
MIP subproblem algorithm [1]*Primal simplex pricing algorithm [2]	0.44011

Continued on Next Page...

Table F.1 – Continued

Parameter [Value]	Estimate
MIP subproblem algorithm [0]*Primal simplex pricing algorithm [2]	0.43963
Primal simplex pricing algorithm [2]	-0.434686
MIP subproblem algorithm [3]	-0.43036
MIP subproblem algorithm [3]*Dual simplex pricing algorithm [1]	-0.390944
MIP subproblem algorithm [2]*Dual simplex pricing algorithm [0]	-0.375904
MIP subproblem algorithm [4]*Primal simplex pricing algorithm [1]	-0.364064
MIP subproblem algorithm [2]	0.355358
MIP subproblem algorithm [0]*Dual simplex pricing algorithm [1]	0.355159
MIP subproblem algorithm [2]*Dual simplex pricing algorithm [4]	-0.345037
MIP subproblem algorithm [0]*Dual simplex pricing algorithm [2]	0.338469
MIP subproblem algorithm [3]*Primal simplex pricing algorithm [1]	0.33669
MIP subproblem algorithm [3]*Primal simplex pricing algorithm [0]	0.328202
MIP subproblem algorithm [3]*Dual simplex pricing algorithm [0]	0.315164
MIP subproblem algorithm [0]*Dual simplex pricing algorithm [3]	0.307739
Dual simplex pricing algorithm [4]*Primal simplex pricing algorithm [3]	0.304757
MIP subproblem algorithm [4]*Dual simplex pricing algorithm [3]	-0.302078
Dual simplex pricing algorithm [4]	0.300141
MIP subproblem algorithm [0]*Primal simplex pricing algorithm [0]	-0.287244

Continued on Next Page...

Table F.1 – Continued

Parameter [Value]	Estimate
Dual simplex pricing algorithm [3]*Primal simplex pricing algorithm [4]	-0.277015
MIP subproblem algorithm [0]*Primal simplex pricing algorithm [1]	-0.267267
MIP subproblem algorithm [3]*Dual simplex pricing algorithm [3]	0.266494
MIP subproblem algorithm [4]	0.265503
Dual simplex pricing algorithm [3]*Primal simplex pricing algorithm [0]	0.264447
MIP subproblem algorithm [0]*Primal simplex pricing algorithm [3]	-0.262732
Dual simplex pricing algorithm [2]*Primal simplex pricing algorithm [1]	0.254671
MIP subproblem algorithm [1]*Dual simplex pricing algorithm [2]	0.247399
MIP subproblem algorithm [1]*Dual simplex pricing algorithm [3]	0.246688
Dual simplex pricing algorithm [3]	-0.224985
Dual simplex pricing algorithm [4]*Primal simplex pricing algorithm [2]	-0.224146
Primal simplex pricing algorithm [1]	0.223104
Primal simplex pricing algorithm [0]	0.219098
Dual simplex pricing algorithm [4]*Primal simplex pricing algorithm [0]	-0.217697
MIP subproblem algorithm [1]	0.216062
Dual simplex pricing algorithm [3]*Primal simplex pricing algorithm [2]	0.211714
Dual simplex pricing algorithm [1]*Primal simplex pricing algorithm [4]	-0.210702
MIP subproblem algorithm [1]*Primal simplex pricing algorithm [0]	-0.20239

Continued on Next Page...

Table F.1 – Continued

Parameter [Value]	Estimate
MIP subproblem algorithm [1]*Dual simplex pricing algorithm [0]	-0.19331
Dual simplex pricing algorithm [4]*Primal simplex pricing algorithm [4]	-0.190089
Dual simplex pricing algorithm [1]*Primal simplex pricing algorithm [3]	-0.177912
MIP subproblem algorithm [3]*Dual simplex pricing algorithm [2]	-0.170107
MIP subproblem algorithm [4]*Dual simplex pricing algorithm [4]	-0.167167
MIP subproblem algorithm [2]*Primal simplex pricing algorithm [0]	-0.156334
MIP subproblem algorithm [4]*Dual simplex pricing algorithm [0]	-0.153415
MIP subproblem algorithm [3]*Primal simplex pricing algorithm [3]	0.150154
MIP subproblem algorithm [2]*Dual simplex pricing algorithm [3]	-0.149421
MIP subproblem algorithm [2]*Dual simplex pricing algorithm [1]	-0.145882
Dual simplex pricing algorithm [0]	0.137592
Dual simplex pricing algorithm [2]	-0.133045
MIP subproblem algorithm [1]*Dual simplex pricing algorithm [1]	0.113608
Dual simplex pricing algorithm [1]*Primal simplex pricing algorithm [2]	0.10671
MIP subproblem algorithm [2]*Primal simplex pricing algorithm [3]	-0.089438
MIP subproblem algorithm [0]*Dual simplex pricing algorithm [0]	0.08889
Primal simplex pricing algorithm [3]	-0.088764
Dual simplex pricing algorithm [1]*Primal simplex pricing algorithm [0]	-0.088301

Continued on Next Page...

Table F.1 – Continued

Parameter [Value]	Estimate
Dual simplex pricing algorithm [2]*Primal simplex pricing algorithm [2]	-0.084226
MIP subproblem algorithm [2]*Primal simplex pricing algorithm [1]	-0.079496
MIP subproblem algorithm [4]*Dual simplex pricing algorithm [2]	-0.078885
MIP subproblem algorithm [4]*Primal simplex pricing algorithm [3]	0.071153
MIP subproblem algorithm [3]*Dual simplex pricing algorithm [4]	0.071042
MIP subproblem algorithm [0]*Dual simplex pricing algorithm [4]	0.069398
Dual simplex pricing algorithm [0]*Primal simplex pricing algorithm [1]	-0.06663
MIP subproblem algorithm [4]*Dual simplex pricing algorithm [1]	0.064805
Dual simplex pricing algorithm [2]*Primal simplex pricing algorithm [4]	0.062498
MIP subproblem algorithm [1]*Primal simplex pricing algorithm [1]	0.05246
MIP subproblem algorithm [2]*Dual simplex pricing algorithm [2]	-0.052295
Dual simplex pricing algorithm [4]*Primal simplex pricing algorithm [1]	-0.046975
Dual simplex pricing algorithm [3]*Primal simplex pricing algorithm [1]	0.045566
Dual simplex pricing algorithm [2]*Primal simplex pricing algorithm [0]	0.045463
MIP subproblem algorithm [4]*Primal simplex pricing algorithm [0]	-0.044633
Dual simplex pricing algorithm [0]*Primal simplex pricing algorithm [0]	0.027179
Dual simplex pricing algorithm [0]*Primal simplex pricing algorithm [3]	-0.0229
MIP subproblem algorithm [1]*Dual simplex pricing algorithm [4]	-0.017294

Continued on Next Page...

Table F.1 – Continued

Parameter [Value]	Estimate
MIP subproblem algorithm [0]	0.014067
Dual simplex pricing algorithm [0]*Primal simplex pricing algorithm [2]	0.005279
Dual simplex pricing algorithm [1]	-0.00237

Vita

Charles Robert Stewart was born October 6, 1983 in Salt Lake City, Utah, and is an American citizen. He graduated from West Springfield High School, Springfield, Virginia in 2001. Mr. Stewart received his Bachelor of Arts in Philosophy and Bachelor of Science in Computer Engineering from Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, Virginia in 2006.