University of Tennessee, Knoxville

TRACE: Tennessee Research and Creative Exchange

Doctoral Dissertations

Graduate School

12-2004

Com- putational Subset Model Selection Algorithms and Applications

Xinli Bao
University of Tennessee, Knoxville

Follow this and additional works at: https://trace.tennessee.edu/utk_graddiss

 Part of the Theory and Algorithms Commons

Recommended Citation

Bao, Xinli, "Com- putational Subset Model Selection Algorithms and Applications. " PhD diss., University of Tennessee, 2004.
https://trace.tennessee.edu/utk_graddiss/1887

This Dissertation is brought to you for free and open access by the Graduate School at TRACE: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of TRACE: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

To the Graduate Council:

I am submitting herewith a dissertation written by Xinli Bao entitled "Com- putational Subset Model Selection Algorithms and Applications." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Management Science.

Hamparsum Bozdogan, Major Professor

We have read this dissertation and recommend its acceptance:

Kenneth Gilbert, Hamila Bensmail, Chanaka Edirisinghe

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:
I am submitting herewith a dissertation written by Xinli Bao entitled "Computational Subset Model Selection Algorithms and Applications." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Management Science.

Hamparsum Bozdogan
Major Professor

We have read this dissertation
and recommend its acceptance:

Kenneth Gilbert

Hamila Bensmail

Chanaka Edirisinghe

Accepted for the Council:

Anne Mayhew
Vice Chancellor and Dean of
Graduate Studies

(Original Signatures are on file with official student records.)

Computational Subset Model Selection Algorithms and Applications

A Dissertation

Presented for the

Doctor of Philosophy Degree

The University of Tennessee, Knoxville

Xinli Bao

December 2004

Copyright © 2004 by Xinli Bao

All rights reserved.

To My Mom and Dad

Thank you for your love and support.

Acknowledgments

I am indebted to many individuals who have directly and indirectly contributed to and influenced the development of this dissertation. Many thanks to my co-advisors Dr. Hamparsum Bozdogan and Dr. Kenneth Gilbert. This dissertation would not have been materialized without their guidance, support and attention to details. They have helped to provide considerable amount of growth for future research work and in my professional career. In addition, I would like to thank the other committee members, Dr. Chanaka Edirisinghe and Dr. Hamila Bensmail for their valuable comments that significantly improved this dissertation.

The Bonham dissertation award provided by the College of Business Administration at the University of Tennessee, Knoxville is also greatly acknowledged and much appreciated.

I am also grateful to the other members of the faculty and staff of the department of Statistics, Operations and Management Science (SOMS). They have helped to make my experience in UT a fruitful, pleasant and wonderful one.

I owe a great debt of gratitude to my mother, father and sister for their love and support as well as their great sacrifice during the past five years. They are always there for me, which helped me to take on this challenging journey of pursuing a Ph.D. degree and succeed.

Abstract

This dissertation develops new computationally efficient algorithms for identifying the subset of variables that minimizes any desired information criteria in model selection.

In recent years, the statistical literature has placed more and more emphasis on information theoretic model selection criteria. A model selection criterion chooses model that "closely" approximates the true underlying model. Recent years have also seen many exciting developments in the model selection techniques. As demand increases for data mining of massive data sets with many variables, the demand for model selection techniques are becoming much stronger and needed. To this end, we introduce a new Implicit Enumeration (IE) algorithm and a hybridized IE with the Genetic Algorithm (GA) in this dissertation.

The proposed Implicit Enumeration algorithm is the first algorithm that explicitly uses an information criterion as the objective function. The algorithm works with a variety of information criteria including some for which the existing branch and bound algorithms developed by Furnival and Wilson (1974) and Gatu and Kontoghiorghies (2003) are not applicable. It also finds the "best" subset model directly without the need of finding the "best" subset of each size as the branch and bound techniques do.

The proposed methods are demonstrated in multiple, multivariate, logistic regression and discriminant analysis problems. The implicit enumeration algorithm converged to the optimal solution on real and simulated data sets

with up to 80 predictors, thus having

$$2^{80} = 1,208,925,819,614,630,000,000,000$$

possible subset models in the model portfolio. To our knowledge, none of the existing exact algorithms have the capability of optimally solving such problems of this size.

Keywords and Phrases: *Implicit Enumeration; Branch and Bound; Variable Selection; Subset Selection; Model Selection; Integer Programming; Genetic Algorithm; Optimization; and Information Criteria.*

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A critical issue in data analysis is to select a good approximating model. Over the past three decades, a number of criteria that are important for model selection have come into existence since Akaike's initial introduction of the final prediction error criterion (Akaike 1969). In this chapter, we will examine the importance of model selection in statistical modeling and the role of information theoretic model selection criteria in this endeavor. We then discuss the motivation behind this work in section 1.3. The organization of this dissertation appears in section 1.4. Finally, section 1.5 describes the contributions of this dissertation to the existing knowledge.

1.1 On the Importance of Subset Model Selection

Subset model selection arises when one wants to model the relationship between a variable of interest and a subset of potential explanatory variables or predictors, but there is uncertainty about which subset to use. It is a central problem in science as stated in Myung, et. al. 2003: "The question of how one should choose among competing explanations (models) of observed data is at the core of science."

A model is an abstract mechanism that one can imagine as having generated the data. It is the nearest representation of the observed phenomenon or the "true situation". In trying to understand the effect of one variable on another, it is often desirable to include many variables, which are either known or believed to have an effect. Given such a set of data, statisticians may need to select some appropriate model from a set of competing models. Subset model selection is desired in several situations. In some cases, there may be substantial extra cost to collect the additional data and measure the variables. In other cases, inclusion of an extra variable into the model may have a negative impact on the model complexity and the prediction accuracy. Decisions based on highly inaccurate prediction and inclusion of irrelevant variables can be detrimental. Thus, a good parsimonious model is often desirable. Based on the model, interpretations, predictions and decisions can be made. A good model should give adequate prediction accuracy, easy to

interpret and cost reasonably for data measurement.

1.2 Information Theoretic Model Selection and Information Criteria

The classical model selection method is based on hypothesis testing. Models are stated as statistical hypotheses. A significance value α is used as a rule for the acceptance or rejection of the null hypothesis. For example, the hypothesis testing approach will test the null hypothesis that a certain set of variables' coefficients equals 0, i.e. they are not included in the model, at an arbitrarily specified level of significance.

Many statistical researchers, or other scientists have long been aware that the so-called significance levels in hypothesis testing are totally without foundation (Linhart and Zucchini 1986). In recent years, an alternative approach, the information theoretic approach has been well accepted by both the researchers and practitioners. Burnham and Anderson (1998) wrote the following in reference to the criterion based model selection: "Inference from multiple models, or the selection of a single 'best' model, by methods based on the Kullback-Leibler distance are almost certainly better than other methods commonly in use now (hypothesis testing of various sorts or merely the use of just one available model)."

The informational approach selects models based on some information criterion. A model selection criterion is an objective measure of model per-

formance and chooses the model that "closest" approximates the true model.

Information criteria seek a compromise between an adequate goodness of fit and a small number of parameters by adding a penalty term for over-parameterization (complexity) to the lack of fit measure. As more parameters are needed to be estimated, the lack of fit term decreases, but causes the model to become more complex and may result in over fitting the data. Information-based criterion provides a trade-off between the ability of the model to explain the data and the model complexity in order to adhere to the principle of parsimony when it comes to statistical modeling.

Information criteria can be written in the general form as:

$$z(w) = f(w) + g(w), \tag{1.1}$$

where w is a binary vector that indicates which variable is included in the subset model. $w_j = 1$ if variable j is included in the model. $w_j = 0$ if variable j is not included in the model. $z(w)$ represents the information criterion given a sub-model. The first term $f(w)$ is a measure of lack of fit (the maximized log likelihood or the error sum of squares) between the model and the data. While the second term $g(w)$ is a "penalty function" associated with the number of parameters that must be estimated. In general, if we add another variable to the set defined by a given w, the result would be a decrease in $f(w)$ and an increase in $g(w)$. That is, in general, both $f(w)$ and

$g(w)$ satisfy the monotonic condition:

$$f(w_1) \leq f(w_2) \; and \; g(w_1) \geq g(w_2), if \; w_1 \geq w_2$$

Among all information criteria, Bozdogan's new information theoretic measure of complexity criterion termed as ICOMP (Bozdogan, 1987, 1988, 1990, 1994, 1996, 2000) has been used extensively in linear and nonlinear model selection and evaluation. ICOMP is based on entropic or maximal information theoretic measure of complexity. It has a strong theoretical foundation and it has been shown to be a powerful tool that is particularly suited in many application areas including regression analysis.

We introduce several popular information criteria as follows:

1. Akaike's (1973) information criterion (AIC) is defined as:

$$AIC = -2 \sum_{i=1}^{n} logf(x_i|\hat{\theta}) + 2k = -2logL(\hat{\theta}) + 2k, \qquad (1.2)$$

   where $L(\hat{\theta})$ is the maximized likelihood function, and k is the number of free parameters in the model. The model with minimum AIC value is chosen as the best model to fit the data. In AIC, the compromise takes place between the maximized log likelihood , i.e., $-2logL(\hat{\theta})$ (the lack of fit component) and k, the number of free parameters estimated within the model (the penalty term). See Bozdogan (1987).

2. Schwarz's (1978) Bayesian information criterion (SBC), also called Bay-

esian information criterion (BIC) is defined as:

$$SBC = -2\sum_{i=1}^{n} logf(x_i|\hat{\theta}) + klog(n) = -2logL(\hat{\theta}) + klog(n), \quad (1.3)$$

where n is the number of the observations.

3. Generalized Akaike's information criterion (GAIC)(Shibata 1989, Bozdogan 2000):

$$GAIC = -2\sum_{i=1}^{n} logf(x_i|\hat{\theta}) + 2tr(\hat{\mathcal{F}}^{-1}\hat{\mathcal{R}}) = -2logL(\hat{\theta}) + 2tr(\hat{\mathcal{F}}^{-1}\hat{\mathcal{R}}), \quad (1.4)$$

where $\hat{\mathcal{F}}$ is the estimated Fisher information in inner product or Hessian form, and $\hat{\mathcal{R}}$ is the outer product form of the Fisher information matrix both with dimensions $(k \times k)$. $tr(\hat{\mathcal{F}}^{-1}\hat{\mathcal{R}})$ is known as the Lagrange Multiplier Test (LMT) statistic.

4. Bozdogan's (1987) consistent Akaike's information criterion (CAIC) is:

$$CAIC(k) = -2logL(\hat{\theta}_k) + k(logn + 1), \quad (1.5)$$

5. After adding more penalty terms, we get Consistent AIC with Fisher information (CAICF) (Bozdogan 1987):

$$CAICF(k) = -2logL(\hat{\theta}_k) + k(logn + 2) + log|\hat{\mathcal{F}}|, \quad (1.6)$$

6. Bozdogan's (1988, 1994, 1998, 2000) new information measure of complexity (ICOMP) for model selection is defined as:

$$ICOMP(IFIM) = -2logL(\hat{\theta}) + 2C_1(\hat{\mathcal{F}}^{-1}(\hat{\theta})), \tag{1.7}$$

where C_1 denotes the maximal information complexity of $\hat{\mathcal{F}}^{-1}$, and C_1 is defined as

$$C_1(\hat{\mathcal{F}}^{-1}) = \frac{s}{2}log[\frac{tr(\hat{\mathcal{F}}^{-1})}{p}] - \frac{1}{2}log|\hat{\mathcal{F}}^{-1}|,$$

where s is the dimension or rank of $\hat{\mathcal{F}}^{-1}$. ICOMP chooses simpler models that provides more accurate estimates over more complex over-specified models.

7. Bozdogan's (2004b) misspecification resistent criterion is defined as:

$$ICOMP_{Misspec} = -2logL(\hat{\theta}) + 2C_1(\hat{Cov}^{-1}(\hat{\theta})_{Misspec}), \tag{1.8}$$

where $\hat{Cov}^{-1}(\hat{\theta})_{Misspec} = \hat{\mathcal{F}}^{-1}\hat{R}\hat{\mathcal{F}}^{-1}$ is the estimate of the covariance matrix of $\hat{\theta}$ when a model is misspecified. This criterion guards whether the probability model is misspecified or not as we fit models to the data.

Bozdogan and Haughton (1998) showed that ICOMP class criteria tend to agree with decisions based on minimizing the Kullback-Leibler distance between the true model and each estimated model more often than AIC

and BIC through simulations. Bozdogan (2000) pointed out that "the difference between ICOMP class criteria and AIC, SBC/MDL, CAIC is that with ICOMP we have the advantage of working with both biased as well as unbiased estimates of the parameters and measure of the complexities of their covariances to study the robustness properties of different methods of parameter estimates." AIC and AIC type criteria are based on maximum likelihood estimator's, which are often biased and do not fully take into account the concept of parameter redundancy, accuracy and the parameter interdependencies in model fitting and selection process. In a large scale Monte Carlo misspecification environment, when the true model is not in the model set, experiments under different configurations with varying sample sizes and the error variances are demonstrated in Bozdogan (2000) and (2004b). The numerical results clearly show the excellent performance of the ICOMP class criteria as compared to AIC and SBC/MDL.

In our computational tests, AIC, ICOMP(IFIM) and $ICOMP_{Misspec}$ are used to demonstrate the performance of the newly proposed algorithms in this dissertation.

1.3 Motivation

The existing leaps and bounds procedure implemented in several statistical software packages is proposed and developed by Furnival and Wilson (1974). This algorithm finds the smallest sum of squared error (SSE) (also called the

residual sum of squares or error sum of squares) for subset of each size. It is generalized by Narendra and Fukunaga (1977) to the minimization of a general quadratic form. This algorithm can also be applied with criteria of goodness-of-fit such as the minimum sum of absolute deviations or maximum likelihood. As far as information criteria are concerned, the algorithm can be applied with information criteria, whose second term $g(w)$ depends only on the number of variables, i.e. $g(w)$ is the same for subsets of the same size. Some examples of such criteria are Cp (Mallows 1973), AIC (Akaike 1973), SBC (Schwarz 1978) and HQ (Hannan and Quinn 1979). For criteria whose second term $g(w)$ does not only depend on the subset size, the branch and bound algorithm may converge to suboptimal subset model. Such criteria include GAIC, CAICF, ICOMP, ICOMP(IFIM), $ICOMP_{Misspec}$ etc.

Branch and bound algorithms find the best subset of each size. In most situations, we do not know what size the best model is. Thus, in order to find the best subset according to a certain criterion, which is commonly an objective of the research, the best of each subset size has to be computed. As pointed out by Hamamoto et. al. (1990), "a disadvantage of the branch and bound algorithm is that it is computationally expensive when the number of the features selected is small."

The drawbacks of the existing branch and bound algorithms call for new, more effective and efficient procedures that can be applied to a wider range of information criteria.

1.4 Organization of the Dissertation

This dissertation consists of seven chapters. The layout of the dissertation is as follows: chapter 1 gives an introduction to the topic, motivation and the expected contribution to the literature. Chapter 2 is an overview of computational techniques in model selection. It surveys the existing computational techniques in model selection. Chapter 3 introduces the implicit enumeration algorithm. Chapter 4 gives the computational results of the implicit enumeration algorithm and compares it with the existing branch and bound algorithm and the stepwise method. Chapter 5 explores the genetic algorithm and presents certain hybridization of the methods. Chapter 6 gives examples of several application areas of the algorithms, tested and compared the performance of the algorithms and the information criteria. Chapter 7 summarizes the major findings, gives future directions of research and some concluding remarks.

1.5 Contributions to the Literature

This dissertation makes several key contributions to the subset model selection field.

First, it proposes a new algorithm — the implicit enumeration algorithm that optimally solves the model selection problem. This algorithm overcomes the limitations of all the existing exact algorithms and is the first exact algorithm that explores the mathematical structure of the information criteria.

Thus, it works with a wide variety of information criteria including some for which the existing exact algorithms are not applicable.

Second, the performance of the implicit enumeration algorithm is demonstrated through computational tests with some real and simulated data sets. This new algorithm has shown to optimally solve problems of up to 80 variables in size. To our knowledge, none of the existing exact algorithms have solved problems of this size.

Third, this dissertation introduces a heuristic version of the implicit enumeration algorithm that can be used to solve even larger problems. It also develops a combined methodology of heuristic search and exact algorithm in model selection. This methodology improves the performance of pure exact and pure heuristic approaches.

Fourth, this research demonstrates the applications of these new algorithms in several different areas, namely, multiple and multivariate regression, logistic regression and discriminant analysis.

Finally, this dissertation has presented a unified view of the different computational techniques in subset model selection. The techniques have in many cases come to their existence independently in different fields. Their relatedness and the relative pros and cons are not easy to see. Thus, the aim of this dissertation is to present the latest developments, highlight their practical values, provide an understanding of the methods and compare different methods.

Chapter 2

Overview of Subset Model Selection Techniques

The most straightforward method in subset model selection is to generate all possible models to find the best one. However, the model portfolio soon becomes tremendously large and computationally impossible as the number of models increases exponentially with the number of variables. Thus, various optimization techniques have been developed in model selection. Recent years have seen many exciting developments in the model selection techniques. In this chapter, we survey the existing computational techniques in model selection.

We classify the algorithms into exact and heuristic algorithms. Section 2.1 discusses the exact algorithms. Exact algorithms guarantee finding the optimal subset model. Section 2.2 describes the heuristic algorithms. Heuris-

tic algorithms do not guarantee finding the optimal subset model.

2.1 Exact Algorithms

Branch and Bound (BB) is a global exhaustive search method. It has been widely used in the integer programming problems in operations research. This technique has proven to be reasonably efficient on practical problems. One cannot talk about a branch-and-bound algorithm that can solve all discrete and combinatorial optimization problems. A specific algorithm has to be designed for a specific class of problems.

A specific branch and bound procedure was developed and applied in model selection for the first time by Furnival and Wilson (1974). The objective is to find the minimum residual sum of squares (RSS) for a model of a given size. Their algorithm constructs two trees: One regression tree and one bound tree. The bound tree contains all regressions excluding the last variable. The regression tree contains all the regressions that includes the last variable. The bound tree was traversed in a lexicographical order. The regression tree was traversed in a modified familial order. At each step, two models are generated simultaneously. The algorithm can be adapted to find the best subset of each size for some type of information criterion. The first term of information criteria can be some other goodness-of-fit measure such as the maximum likelihood. The second term of the information criteria has to depend only on the subset size, meaning the second term has to be the

same for subsets of the same size.

Gatu and Kontoghiorghes (2003) presented a branch and bound algorithm that is designed specifically to find subsets with minimum error sum of squares for subset of each size. Their tree was built by dropping variables so that each node contains several evaluations of subsets of different sizes. QR decomposition and Givens rotation are used to compute the RSS efficiently. Their algorithm can be adapted to use information criteria that are functions of error sum of squares. Similar to Furnival and Wilson's algorithm, their algorithm only finds the minimum of the first term of the information criteria, $f(w)$, thus requiring the second term, $g(w)$, to be the same for subset of the same size in order to be able to locate the optimal model.

In the pattern recognition field, there is a similar problem called feature selection. Variables are called features in pattern recognition. Many authors have described branch and bound procedures. All these branch and bound algorithms consider the problem of selecting the best subset of m features out of n original features that maximizes some criterion function J. The function J satisfies the monotonic property, that is, $J(w_2) \geq J(w_1)$, if $w_2 \geq w_1$.

Narendra and Fukunaga (1977) was the first to propose a branch and bound algorithm for feature subset selection in the pattern recognition field. Since then many researchers have developed different variations of the branch and bound algorithm. Yu and Yuan (1997) proposed a BAB+ algorithm and made the first substantial improvement to the original branch and bound

algorithm proposed by Narendra and Fukunaga (1977). Their modification recognizes that in selecting a certain number of features, any intermediate nodes in a single path from the root to the terminal node need not be evaluated and thus, the procedure immediately skips to the terminal node, thereby saving intermediate evaluations.

Somol et al. (2000) introduced a fast branch and bound (FBB) procedure. It maintains a prediction mechanism for individual feature and expects that its contribution to the criterion value do not change much in relation to different subsets. Thus, nodes in the solution tree that are between the root and the leaf can be predicted.

Chen (2003) made further improvements to the BAB+ algorithm by utilizing the monotonic conditions of the criterion function. By monoticity, a subset of the feature set that has previously been evaluated need not be evaluated. He showed that the improved branch and bound algorithm he proposed outperformed the FBB and BAB+ in terms of number of computations and computational time.

The previously mentioned feature selection algorithms are based on the depth first search of the tree. Siedlecki and Sklansky (1988) is the first one to discuss a best first search strategy. They developed a "beam search" algorithm. However, the beam search algorithm is suboptimal, and it does not guarantee to find the optimal subset.

A common characteristic of all proposed branch and bound algorithms is that they are all based on selecting the best subset that has a specific number

of variables. They aim to find the minimum of $f(w)$ and can only be adapted to information criteria where the second term $g(w)$ is the same for subsets of the same size.

To our knowledge after reviewing the literature, no algorithm has been given for selection of the optimal model without finding the best of each size and that also works with all types of information criteria.

In the next section, we give a specific example to illustrate when the branch and bound algorithm fails to find an optimal model.

2.1.1 Limitations of the Existing Branch and Bound Algorithms

The existing branch and bound algorithms aim to find the best subset of each size according to a measure of lack of fit, i.e. $f(w)$. Therefore, it only works with information criterion where the second term $g(w)$ depends solely on the subset size, such as AIC, Cp and SBC. For information criteria where the second term is not a function of the number of variables in the model, the existing branch and bound algorithm does not guarantee to find the optimal subset.

Table 2.1 gives an example that the existing branch and bound algorithms fail to find the optimal subset model. It displays part of the complete enumeration of the four variable subsets of the steam data set with 9 variables. A complete enumeration shows that the best four-variable subset and the

Table 2.1: ICOMP(IFIM) Values for the Steam Data Set

Subset	ICOMP(IFIM)	Lackoffit (f(w))	C1ifim (g(w))
....			
6 7 8 9	19.7925	-13.2636	16.5281
5 7 8 9	2.5093	-27.0734	14.7913
4 7 8 9	29.4648	-16.0471	22.756
1 6 7 9	1.1626	-26.6908	13.9267
4 5 7 9	10.0893	-30.6967	20.393
3 5 7 9	-0.483	-26.4595	12.9883
2 5 7 9	-2.1188	-27.3872	12.6342
1 5 7 9	**-10.09**	**-32.0642**	**10.9871**
3 4 7 9	24.4803	-16.133	20.3066
....			
1 2 6 7	5.9566	-25.8418	15.8992
3 4 5 7	12.4014	-31.0731	21.7373
2 4 5 7	6.4702	-31.2534	18.8618
1 4 5 7	**9.0488**	**-32.5412**	**20.795**
2 3 5 7	0.2665	-27.1041	13.6853
....			

overall "best" subset contains variables {1,5,7,9} according to the information criterion ICOMP(IFIM). However, branch and bound algorithm finds the subset {1,4,5,7} as the best subset for subset of size four because it has the smallest lack of fit. The assumption that the penalty term depends only on the size of the subset does not hold true for the ICOMP(IFIM) criterion. Thus, Branch and Bound algorithms fail to find the "best" subset {1,5,7,9}.

2.2 Heuristic Algorithms

When a problem size exceeds a certain limit, for example, when the number of variables is larger than 30, one might begin to think about using a heuristic method instead of the exact methods. The heuristic algorithms do not guarantee finding an optimal subset that minimizes a given information criterion. Stepwise method is perhaps the first of the kind in model selection and it is sill popular among practitioners. The "stepwise regression" can be dated back to the algorithm proposed by Efroymson (1960). Miller (2002 p. 49) has a detailed description of this algorithm. An and Gu (1989) developed a fast stepwise procedure for the selection of variables by using AIC and BIC (Akaike 1978) criteria, which is similar to the standard stepwise regression procedure in computing steps, but has a faster rate and can be used for a large number of candidate variables. With stepwise methods, only a small number of subsets need to be evaluated. However, the significant levels to enter and stay for candidate variables have to be specified in advance to use the stepwise methods. Stepwise methods do not guard presence of multicollinearity and they have other drawbacks which are documented well in the literature.

Other popular heuristics include genetic algorithm (GA), simulated annealing (SA), tabu search (TS) and threshold accepting (TA).

A genetic algorithm (GA) has been widely used in problems where large numbers of solutions exist. It is an intelligent random search algorithm based

on evolutionary ideas of natural selection and genetics. It follows the principles first laid down by Charles Darwin of survival of the fittest. The algorithm searches within a pre-defined search space to solve a problem. It has outstanding performance in finding the optimal solution for problems in many different fields. Siedlecki and Sklansky (1989) first presented the GA in feature selection. Their experiments indicate that "GA is a powerful tool for feature selection when the dimensionality of the initial feature set is large". Recently, Bearse and Bozdogan (1998) introduced a Genetic Algorithm (GA) with the Information Complexity (ICOMP) criterion as the fitness function to select an optimal subset Vector Autoregressive (VAR) model. Choosing an optimal subset VAR model has been a vexing problem because of its large search space. Genetic Algorithm provides a computationally feasible multi-criteria optimization tool that can be exploited to avoid the "curse of dimensionality" inherent in VAR modeling. Bozdogan and Bearse (2003) applied the algorithm in detecting the influential observations in VAR models. Bozdogan (2004a) introduced the genetic algorithm in intelligent data mining using information complexity in subset selection in linear regression models. Bao and Bozodgan (2004) used the genetic algorithm in model selection in kernel regression models. Chatpattananan and Bozdogan (2004) introduced the genetic algorithm in subset selection of regularized radial basis function (RBF) regression trees and neural networks.

Most recently, Brooks et. al. (2003) presented a transdimensional simulated annealing algorithm that can be used to locate models and parameters

that minimize an information criterion. Their algorithm is an extension to the traditional simulated annealing algorithm that allows for moves that not only change parameter values but also move between competing models. They demonstrated the algorithms through applications to autoregressive time series, logistics regression and recapture-recovery models.

Winker (2001) introduced a heuristic optimization technique threshold accepting and applied it in vector autoregressive order selection. Threshold accepting resembles the simulated annealing. It can also be considered as a refinement to the standard local search procedure. The standard search procedure always rejects the new solution if the objective becomes worse. The threshold acceptance may accept the "worse" solution if the objective worsens less than a previously set threshold.

Chapter 3

The Implicit Enumeration Algorithm

This chapter formally introduces the implicit enumeration algorithm for subset selection. The proposed implicit enumeration algorithm differs and outperforms the existing branch and bound algorithms for subset selection in a number of ways. It is the first algorithm that explicitly uses the information criterion as the objective function. The algorithm works with some information criteria for which the existing branch and bound algorithms are not applicable. It also finds the "best" subset model directly without the need for finding the "best" subset of each size as the existing branch and bound algorithms do.

This chapter shows how the mathematical structure of the information criteria can be exploited to develop an algorithm that is applicable to a wide

variety of subset selection problems. Section 3.1 describes the implicit enumeration algorithm and gives a simple illustrative example. Section 3.2 outlines the steps of the algorithm. Section 3.3 shows ways to compute bounds so that branches that do not contain optimal solutions can be pruned. Different branching strategies are presented in section 3.4. A numerical example is given to illustrate the computational steps of the algorithm in section 3.5. Section 3.6 presents a heuristic version of the algorithm. This heuristic version does not guarantee the finding of the best subset, but computational results show that by choosing a good parameter, the algorithm finds the best subset with a smaller number of model evaluations than the non-heuristic version. We demonstrate the algorithm through computational results on some real data sets and compare it with the existing algorithms in chapter 4.

3.1 Implicit Enumeration

Implicit enumeration algorithms have been used to solve linear integer programming problems for many years, Trotter and Shetty (1974)'s algorithm was the first of such algorithms. However, none of these existing algorithms are applicable to the nonlinear subset selection problem. This section explains how the mathematical structure of the subset selection problem can be exploited to develop an algorithm that is applicable to a wide variety of model selection problems using information criteria.

3.1.1 Notation and Terminology

Implicit Enumeration (IE) algorithm is a binary branch and bound algorithm. Variables are coded as either 0 or 1. A variable is coded as 1 if it is in the model and 0 if out of the model. Let $\{x_1, x_2, ..., x_J\}$ denote the set of candidate variables. A selection of variables is given by $w = \{w_1, w_2, ..., w_J\}$, where $w_j = 0$ if x_j is not included in the model and $w_j = 1$ if x_j is included in the model.

Implicit enumeration initiates the construction of an enumeration tree of all possible solutions, but avoids complete enumeration by identifying and eliminating branches of the tree that do not contain the optimal solution. For example, consider a variable selection problem with three candidate independent variables. Figure 3.1 shows one possible tree that enumerates all eight possible solutions by first branching on w_2, then on w_1 and finally on w_3 . Nodes 0-6 of the enumeration tree represent partial solutions. A partial solution has some of the fixed, either at 0 or 1, while some of the variables are free. A free variable's value is undetermined. Nodes 7 through 14 represent the 8 possible complete solutions. For illustration purposes, we assume values of $f(w)$, $g(w)$ and $z(w)$ for each complete solution and show these values in table 3.1.

Let w^* denote the optimal solution, i.e., w^* minimizes $z(w)$ and let $z^* = z(w^*)$. In the example $w^* = (1, 1, 0)$ with $f(w^*) = 2$, $g(w^*) = 6$ and $z^* = z(w^*) = 8$. This solution is shown at node 13.

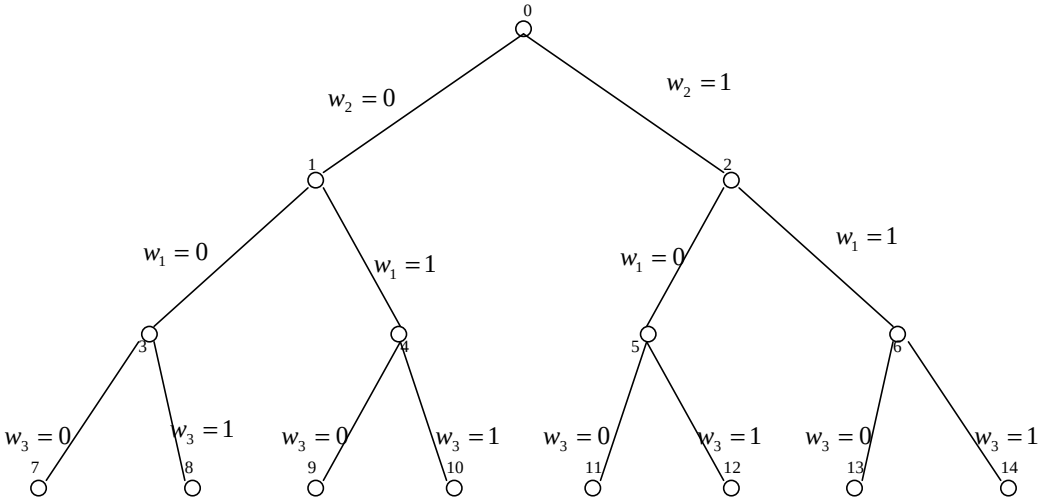

Figure 3.1: An Enumeration Tree

Table 3.1: Values of All the Complete Solutions

Node	7	8	9	10	11	12	13	14
w	(0,0,0)	(0,0,1)	(1,0,0)	(1,0,1)	(0,1,0)	(0,1,1)	(1,1,0)	(1,1,1)
$f(w)$	10	9	7	6	5	4	2	1
$g(w)$	2	4	4	6	4	6	6	8
$z(w)$	12	13	11	12	9	10	8	9

3.1.2 Evaluations of Partial Solutions

Implicit enumeration works with partial solutions. It attempts to discard as many solutions as possible that cannot lead to the optimal solution. To represent a partial solution, let $S = (j_1, j_2, ...)$ contain the indices of those variables that are currently fixed in the subset in the order in which they become fixed. And let $w^S = (w_{j_1}, w_{j_2}, ...)$ be the partial solution vector that contains the values assumed by the fixed variables.

A partial solution having i fixed variables represents one of the nodes in the ith level of an enumeration tree. In figure 3.1 , $S = \{2, 1\}$ with $w^S = \{1, 0\}$ represents node 5 at which $w_2 = 1$ and $w_1 = 0$ have been fixed in that order, and w_3 is free.

A completion of a partial solution is a solution w formed by the assignment of values to its free variables. A partial solution will have 2^q possible completions, where q is the number of free variables. The completions of $S = \{(2, 1\}$ with $w^S = \{1, 0\}$ in the example are $S = \{2, 1, 3\}$ with $w^S = (1, 0, 1)$ and $S = \{2, 1, 3\}$ with $w^S = \{1, 0, 0\}$. In figure 3.1 these two completions are represented at node 11, $w = \{1, 0, 0\}$ and node 12, $w = \{1, 0, 1\}$.

3.1.3 Bounding and Fathoming Partial Solutions

For a given partial solution, let $\bar{z}(S, w^S)$ represent an upper bound on $z(w)$ for the best completion of (S, w^S), i.e., there exists at least one completion

w of (S, w^S) such that $z(w) \leq \bar{z}(S, w^S)$. Note that evaluating $z(w)$ for any completion of (S, w^S) provides such a bound.

In the example, an upper bound on the partial solution $S = \{2\}$, $w^S = \{0\}$ (node 1) can be obtained by computing the objective function of any completion obtained by assigning values to all the free variables. For example, if all free variables are assigned a value of 0, the resulting completion is $w = \{0, 0, 0\}$ with $z(w) = 12$. Thus $\bar{z}(\{2\}, \{0\}) = 12$ is an upper bound on the z, meaning that at least one completion of $S = \{2\}$, $w = \{1\}$ has a solution less than or equal to 12.

For a given partial solution let $\underline{z}(S, w^S)$ represent a lower bound on $z(w)$ for the best completion of (S, w^S), i.e., there exists no w which is a completion of (S, w^S) for which $z(w) \leq \underline{z}(S, w^S)$.

For example, consider the partial solution $S = \{2\}$, $w^S = \{0\}$ (node 1). A crude lower bound can be obtained by combining the smallest possible values of $f(w)$ and $g(w)$. The smallest value of $f(w)$ for any completion of a partial solution is obtained by setting all free variables to 1. Hence the smallest possible value for $f(w)$ for any completion of (S, w^S) is $f(0, 0, 1) = 9$. The smallest possible value for $g(w)$ for any completion of a partial solution is obtained by setting all free variables to 0. Hence the smallest value of $g(y)$ for any completion of (S, w^S) is $g(0, 0, 0) = 2$. Combining these two best cases tells us that no completion of (S, w^S) will have $z(w) \leq 6 + 2 = 8$.

A partial solution is fathomed when either (1) it is shown that no completion can be optimal or (2) it is shown that the best feasible completion

has been found. In particular, if the lower bound on a particular partial solution is greater than or equal to the upper bound on another partial solution already examined, then that partial solution cannot have a completion that is optimal. Therefore further examination of that partial solution is unnecessary.

3.1.4 A Simple Example

Table 3.2 shows the iterations for the application to the example in figure 3.1. The order in which the variables are selected for branching (in step 6 of the algorithm) was artitrarily chosen to be first w_2, then w_1 and then w_3. Also in step 6 the variable selected for branching was set to 1. The upper bound in each iteration (step 2) was computed by arbitrarily setting the free variable to 0. The lower bound (step 4) was computed by adding the value $f(w)$ obtained setting all free variables to 1 to the value of $g(w)$ obtained by setting all free variables to 0.

After evaluating 3 of the 14 nodes in the complete enumeration tree, the algorithm finds the optimal solution (1,1,0). The algorithm evaluates 7 of the nodes before it determines that this solution is optimal. In other words, the algorithm proves the optimality of the solution (1,1,0) after evaluating 7 nodes.

Obviously, for any given problem, the speed of convergence of the algorithm will vary depending on (1) the branching strategy (the order in which variables are added to the enumeration and whether they are first fixed at

Table 3.2: Implicit Enumeration Solution to the Example in Figure 3.1

S	w^S	Node	Upper Bound	Lower Bound	Incumbent	$\bar{z}^*$	Next Step
{}	{}	0	z(0,0,0)=12	f(1,1,1)+g(0,0,0)=3	(0,0,0)	12	Branch
{2}	{1}	2	z(1,0,0)=9	f(1,1,1)+g(1,0,0)=5	(1,0,0)	9	Branch
{2,1}	{1,1}	6	z(1,1,0)=8	f(1,1,1)+g(1,1,0)=7	(1,1,0)	8	Branch
{2,1,3}	{1,1,1}	14	z(1,1,1)=9	f(1,1,1)+g(1,1,1)=9	(1,1,0)	8	Fathomed, Backtrack
{2,1,$\underline{3}$}	{1,1,0}	13	z(1,1,0)=8	f(1,1,0)+g(1,1,0)=8	(1,1,0)	8	Fathomed, Backtrack
{2,$\underline{1}$}	{1,0}	5	z(1,0,0)=9	f(1,0,1)+g(1,0,0)=8	(1,1,0)	8	Fathomed, Backtrack
{$\underline{2}$}	{0}	1	z(0,0,0)=12	f(0,1,1)+g(0,0,0)=8	(1,1,0)	8	Fathomed, Backtrack

0 or 1) and (2) the "tightness" of the upper and lower bounds computed. Bounding procedures and branching strategies are discussed in sections 3.3 and 3.4.

3.2 Algorithm Description

In this section, we formally introduce the implicit enumeration algorithm for subset selection. At any point in an enumeration, let $\bar{z}^*$ denote the best solution found thus far.

We also adopt an underlining notation. Underlining any elements of S denotes that both values (0 or 1) of the underlined elements have been enumerated, i.e., a partial solution identical to S but having the opposite values of w^S for the underlined elements has previously been examined. For example: $S = \{2, \underline{3}\}$ and $w^S = \{1, 0\}$ indicates that $w^S = \{1, 1\}$ has already been enumerated.

The steps of the implicit enumeration algorithm are as given below:

1. *Initialization:* Set $S = \emptyset$, $w^S = \emptyset$ and $\bar{z}^* = \infty$. Let the vector $w = \{0, 0, ..., 0\}$ represent the incumbent solution.

2. *Computing an upper bound:* Compute the criterion value for selected completions (described in section 3.3.1) of (S, w^S). Set $\bar{z}(S, w^S)$ equal to the smallest criterion value of these completions. Go to step 3.

3. *Updating:* If $\bar{z}(S, w^S) < \bar{z}^*$, a solution better than incumbent has been found. Replace the incumbent solution with the best feasible completion of (S, w^S), and set $\bar{z}^* = \bar{z}(S, w^S)$.

4. *Computing a lower bound:* Compute $\underline{z}(S, w^S)$. The method for computing the lower bound is described in section 3.3.2.

5. *Fathoming:* (S, w^S) is fathomed if

   (a) $\bar{z}(S, w^S) > \bar{z}^*$, indicating that no completion of (S, w^S) can be optimal.

   (b) $\bar{z}(S, w^S) = \bar{z}^*$, indicating that the best feasible completion of (S, w^S) has been found.

6. *Branching:* Add another variable to S and add the corresponding component (0 or 1) to w^S. The method used to select a variable and to choose the value of the new component of w^S are described in section 3.4.

7. *Backtracking:* If all of the elements of S are underlined, the incumbent represents the optimal solution. Otherwise, choose the right most non-underlined element of S, underline it and change the corresponding element of w^S (from 0 to 1 or from 1 to 0). Then drop all elements of S (and the corresponding component of w^S) to the right of the newly underlined element. Go to step 2.

In practice, the success of the algorithm depends on getting good upper and lower bounds, and on a good branching strategy, i.e., determining the variable to be added to S in step 6 and determining whether to fix the corresponding component of w^S at 0 or 1. The following sections describe the bounding and branching methods.

3.3 Bounding Procedures

The efficiency of an implicit enumeration algorithm depends on tight upper and lower bounds and a good branching strategy. These are the essential elements that determine the rate of convergence to the optimal solution. In this section, we will describe the bounding procedures.

3.3.1 The Methods for Computing the Upper Bound

Consider any partial solution (S, w^S) and let p denote the number of variables fixed in the solution (p=Σw) and let q denote the number of free variables.

Define the completion w_c obtained by setting all free variables to zero as the **core solution**.

Define the completion w_r obtained by setting all of the free variables to 1 as the **replete solution**. This solution has $p + q$ variables.

Both $z(w_c)$ and $z(w_r)$ provide upper bounds. However, we can get much tighter upper bounds by finding two additional completions and choosing the smallest of the bounds obtained.

One of these two additional completions is the **augmented core solution** w_{ac}, which is defined as the best completion that can be obtained by adding exactly one free variable to the core solution. This subset is obtained by evaluating the q different solutions of $p + 1$ variables obtained by adding each of the free variable to and choosing the solution that minimizes $z(w)$.

The other completion examined is the **diminished replete solution**, which is the best solution that can be obtained by deleting exactly one free variable from the replete subset. This variable is obtained by evaluating all of the q solutions of $p + q - 1$ variables obtained by deleting one free variable from the replete solution and choosing the solution that minimizes $z(w)$.

The upper bound is then computed as:

$$\bar{z}(S, w^S) = min(z(w_c), z(w_{ac}), z(w_r), z(w_{dr})).$$

3.3.2 The Methods for Computing the Lower Bound

The discussion of this section exploits the facts that $f(w)$ and $g(w)$ are monotonic functions. We write $f(w) = l(s)$ and $g(w) = h(s)$, where $s =$ the number of variables in the solution defined by w, i.e. Σw.

Two cases are considered to compute a lower bound. The first case is that when $g(w) = h(s)$ satisfies the condition that: $h(s_1) \geq h(s_2)$ if $s_1 \geq s_2$. That is, when the second term of the criterion function only depends on the number of variables in the subset, we have the following theorem:

Theorem 1 *A lower bound on any completion of* (S, w^S) *can be computed as:*

$$\underline{z}(S, w^S) = min(z(w_c), z(w_{ac}), z(w_r), z(w_{dr}), \tilde{z}),$$

where $\tilde{z} = f(w_{dr}) + h(p + 2)$.

Proof

First observe that the core solution represents the only (and therefore the best) completion of (S, w^S) having p variables. The augmented core solution represents the best completion having $p + 1$ variables. The replete solution represents the only, and therefore the best completion having $p + q$ variables. The diminished replete solution represents the best completion having $p + q - 1$ variables.

Hence, if we can also obtain a lower bound on those completions having s variables, where $p + 2 \leq s \leq p + q - 2$, we can compute a lower bound for all completions.

Any completion w having s variables, where $p+2 \leq s \leq p+q-2$, will have $f(w) \geq f(w_{dr})$ and will have $g(w) \geq h(p+2)$. Therefore, $f(w_{dr}) + h(p+2)$ is a lower bound on any completion having s variables, $p+2 \leq s \leq p+q-2$.

A lower bound on any completion of (S, w^S) can be computed as:

$$\underline{z}(S, w^S) = min(z(w_c), z(w_{ac}), z(w_r), z(w_{dr}), \tilde{z}) = min(\bar{z}(S, w^S), \tilde{z}).$$

The second case is that when the condition $h(s_1) \geq h(s_2)$ if $s_1 \geq s_2$ is not satisfied. However, the monotonic condition of both $l(s)$ and $h(s)$ still holds. That is, $l(s_2) \geq min(l(s_1))$ and $h(s_1) \geq min(h(s_2))$ if $s_1 \geq s_2$. In other words, $l(s)$ of subset of size s_2 is greater than the minimum of that of all subsets having s_1 variables. $h(s)$ of subset of size s_1 is greater than the minimum of that of all subsets having s_2 variables if $s_1 \geq s_2$. We then have:

Theorem 2 *A lower bound on any completion of* (S, w^S) *can be computed as:*

$$\underline{z}(S, w^S) = min(z(w_c), z(w_{ac}), z(w_r), z(w_{dr}), \tilde{z}),$$

where $\tilde{z} = min(l(p+q-1)) + min(h(p+1))$.

Proof.

From the proof of theorem 1, we only need to obtain a lower bound on those completions having s variables, where $p+2 \leq s \leq p+q-2$.

From the monotonic condition of $l(s)$, $l(s) \geq min(l(p+q-1))$. Similarly, we have $h(s) \geq min(h(p+1))$. It follows that $\tilde{z} = min(l(p+q-1)) +$

$min(h(p+1)) \leq l(s) + h(s)$.

Thus, $\tilde{z} = min(l(p+q-1)) + min(h(p+1))$ is a lower bound for any subsets having s variables.

A lower bound on any completion of (S, w^S) can be computed as:

$$\underline{z}(S, w^S) = min(z(w_c), z(w_{ac}), z(w_r), z(w_{dr}), \tilde{z}).$$

3.4 Branching strategies

In branching it must be decided which free variable to add to S and the value (0 or 1) to add to the corresponding element of w^S. There are two obvious choices:

1. Choose the free variable that was added to the core solution to form the augmented core solution. Fix the value of this variable to 1.

2. Choose the free variable that was deleted from the replete solution to form the diminished replete solution. Fix the value of this variable at 0.

In our computational test with regression problems, we tested three branching strategies, namely, the greedy, variable augmentation and variable deletion strategy. A recursive procedure is used for all the branching strategies.

3.4.1 The Greedy Branching Strategy

In the greedy branching strategy, the variable chosen for branching is determined by comparing the augmented core and diminished replete solutions.

If $z(w_{ac}) \leq z(w_{dr})$, the free variable that was added to the core solution to form the augmented core solution will be added to S. The corresponding element of w^S will be set to 1. If $z(w_{ac}) > z(w_{dr})$, the free variable that was deleted from the core regression to form the diminished core solution will be added to S. The corresponding element of w^S will be set to 0.

3.4.2 The Variable Augmentation Branching Strategy

In the variable augmentation branching strategy, the free variable that was added to the core solution to form the augmented core solution is added to S. The corresponding element of w^S is set to 1. Bounds in variable augmentation branching strategy could be computed as shown in section 4.2. We term it Aug1. We can also eliminate the computation of the diminished replete solutions. That is, the upper bound can be computed as $\bar{z}(S, w^S) = min(z(w_c), z(w_{ac}), z(w_r))$. and the lower bound can be computed as $\underline{z}(S, w^S) = min(z(w_c), z(w_{ac}), z(w_r), \tilde{z})$. where $\tilde{z} = f(w_r) + h(p+1)$ or $min(l(p+q)) + min(h(p+1))$. We term this strategy Aug2. Aug1 provides tighter bounds than Aug2. However, Aug1 requires more model evaluations than Aug 2 in order to obtain tighter bounds. Thus, Aug1 strategy may require more number of model evaluations if there are many competitive

models to the best model than Aug2.

3.4.3 The Variable Deletion Branching Strategy

In the variable deletion strategy, the free variable that was deleted from the replete solution to form the diminished core solution is added to S. The corresponding value of w^S will be set to 0. Similar to the variable augmentation branching strategy, bounds can be computed following section 4.2. We term this strategy Del1. We can also eliminate the computation of the augmented core solutions. That is, the upper bound can be computed as $\bar{z}(S, w^S) = min(z(w_c), z(w_r), z(w_{dr}))$. and the lower bound can be computed as $\underline{z}(S, w^S) = min(z(w_c), z(w_r), z(w_{dr}), \tilde{z})$. where $\tilde{z} = f(w_{dr}) + h(p)$ or $min(l(p+q-1)) + min(h(p))$. We term this strategy Del2.

3.5 A Numerical Example

The implicit enumeration (IE) algorithm is best illustrated by using an example. We use linear multiple regression as an example. In linear multiple regression, a response variable Y is assumed to be linearly related to the J predictor variables, X_1, X_2, ..., X_J thus

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_J X_J + \epsilon$$

where the residuals ϵ are normally and independently distributed with mean 0 and variance σ^2. The coefficients β_0, β_1, ..., β_J are estimated using the least squares (LS) method. The LS estimates of the regression coefficients, to be denoted by b's, are given in matrix notation by

$$b = (X'X)^{-1}X'Y$$

where $b' = (\beta_0, \beta_1, ..., \beta_J)$, X is an $n \times (J+1)$ matrix in which row i consists of a 1 followed by the values of variables X_1, X_2,...,X_J for the ith observation, and Y is a vector of length n containing the observed values of the variable to be predicted. We can predict the response for a given vector $x' = (1, x_1, .., x_J)$ of the predictor variables, using

$$\hat{Y} = X'b = \beta_0 + \beta_1 X_1 + ... + \beta_J X_J.$$

Subset selection in linear multiple regression is to select w variables from J variables in order to have good prediction accuracy.

The data set used is a sample data set in the statistical software JMP—fitness data set. This data set is the result of an aerobic fitness study and has 7 predictor variables. They are sex (X_1), age (X_2), weight (X_3), run-time (X_4), run pulse (X_5), maximum pulse (X_6) and rest pulse (X_7). The response variable is the oxygen uptake of a person running on a treadmill for a prescribed distance. In order to find a good oxygen uptake prediction equation, we try to find the best regression model by using the AIC as the

model selection function. AIC for linear multiple regression can be computed as:

$$AIC = n \log\left(\frac{SSE_w}{n}\right) + 2k_w$$

where n is the number of observations, k_w is the number of parameters in the model with variables w and SSE_w is the sum of squared errors.

Writing it in the standard format:

$$AIC = f(w) + g(w)$$

where $f(w) = n \log\left(\frac{SSE_w}{n}\right)$ and $g(w) = 2k_w$. We assume that the model always has an intercept term in it.

3.5.1 Computational Steps of The Variable Augmentation Branching Strategy

The computing steps are depicted in an enumeration tree in figure 3.2. The numbers listed next to the Iteration numbers are $\bar{z}(S, w^S)$ and $\underline{z}(S, w^S)$. If the $\bar{z}^* < \underline{z}(S, w^S)$, the branch is fathomed.

It took 9 iterations for the variable augmentation branching strategy to converge to the optimal solution. Iteration 1 and 2 are given in details in tables 3.3 and 3.4. IE starts with a null model as the core solution. To obtain the augmented core model, we compute models by adding variables to the core one by one. The model that contains variable 4 has the smallest

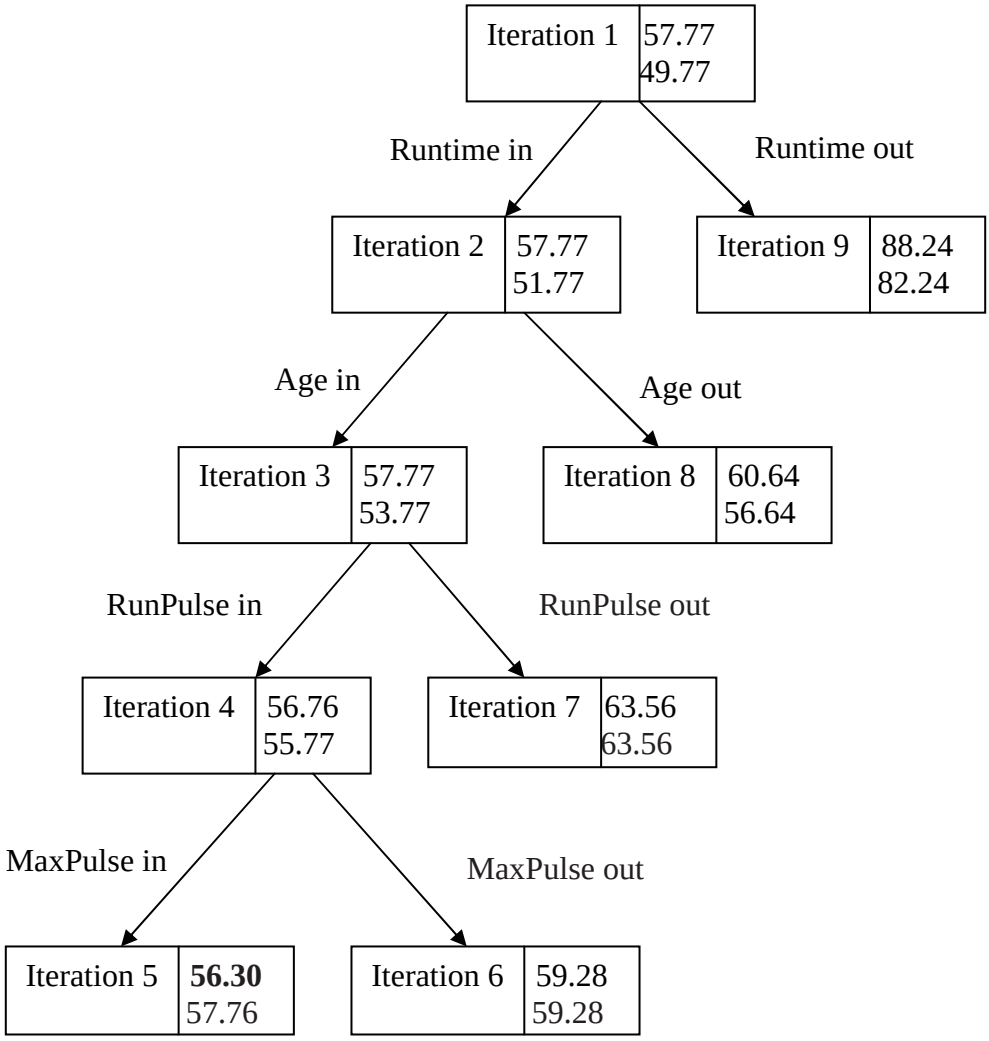

Figure 3.2: Enumeration Tree by Variable Augmentation Branching Strategy.

Table 3.3: Computational Steps of the Implicit Enumeration with Augmentation Strategy: Iteration 1

Iteration 1: S={}, Ws={}

	Model	f(M)	g(M)	z(M)	Upper Bound	Lower Bound
Core	{}			104.7		
Augmented	{1}	94.4126	4	98.4126		
Core	{2}	99.5387	4	103.5387		
	{3}	101.8717	4	105.8717	=43.7645+2*3	
	{4}	60.5424	4	64.5424		
	{5}	97.3569	4	101.3569		
	{6}	97.3197	4	101.3197	57.7645	49.7645
	{7}	100.9169	4	104.9169		
Replete	{1,2,3,4,5,6,7}	43.7398	16	59.7398		
Diminished	{2,3,4,5,6,7}	44.411	14	58.411		
Replete	{1,3,4,5,6,7}	48.6153	14	62.6153		
	{1,2,4,5,6,7}	44.2971	14	58.2971		
	{1,2,3,5,6,7}	75.8764	14	89.8764		
	{1,2,3,4,6,7}	53.6189	14	67.6189		
	{1,2,3,4,5,7}	**43.7645**	14	**57.7645**		
	{1,2,3,4,5,6}	49.4353	14	63.4353		

Table 3.4: Computational Steps of the Implicit Enumeration with Augmentation Strategy: Iteration 2

Iteration 2: S={4}, Ws={1}

	Model	f(M)	g(M)	z(M)	Upper Bound	Lower Bound
Core	{4}	60.5424	4	64.5424		
Augmented	{1,4}	58.599	6	64.599	=43.7645+2*4	
Core	{2,4}	57.8503	6	63.8503		
	{3,4}	60.3527	6	66.3527		
	{4,5}	58.2785	6	64.2785	57.7645	51.7645
	{4,6}	60.524	6	66.524		
	{4,7}	60.3189	6	66.3189		
Replete	{1,2,3,4,5,6,7}	43.7398	16	59.7398		
Diminished	same as iteration 1					
Replete						

criterion value. Thus it is the augmented core solution and the next iteration will have the model containing only variable 4 as the core solution. Similarly, adding variable 2 to the core model with variable 4 in the second iteration gives the smallest criterion value. Thus, model containing variable 2 and 4 will be the core for iteration 3. The rest of the iterations follow the same procedure.

3.5.2 Computational Steps of The Greedy and The Variable Deletion Branching Strategies

It took 15 iterations for the variable deletion and greedy branching strategy to converge to the optimal solution. The iterations of the two strategies are identical in this case. The details of the computation of the first two steps of the greedy strategies are the same and they are shown in table 3.5 and 3.6. Computational steps for all iterations are shown in figure 3.3.

Although the computations of the iterations are the same for both strategies, the notions behind are not the same. For the greedy strategy, we compare the augmented core solution and the diminished replete solution. If the criterion value of the augmented core solution is smaller than that of the diminished replete solution, we add the variable that was added to the core to form the augmented core solution. Otherwise, we add the variable that was deleted from the replete solution to form the diminished replete solution and fix it out of the model. In this case, the criterion value of the

Table 3.5: Computational Steps of the Implicit Enumeration with Greedy Branching Strategy: Iteration 1

Iteration 1: S={}, Ws={}

	Model	f(M)	g(M)	z(M)	Upper Bound	Lower Bound
Core	{}			104.7		
Augmented	{1}	94.4126	4	98.4126		
Core	{2}	99.5387	4	103.5387		
	{3}	101.8717	4	105.8717	=43.7645+2*3	
	{4}	60.5424	4	64.5424		
	{5}	97.3569	4	101.3569		
	{6}	97.3197	4	101.3197	57.7645	49.7645
	{7}	100.9169	4	104.9169		
Replete	{1,2,3,4,5,6,7}	43.7398	16	59.7398		
Diminished	{2,3,4,5,6,7}	44.411	14	58.411		
Replete	{1,3,4,5,6,7}	48.6153	14	62.6153		
	{1,2,4,5,6,7}	44.2971	14	58.2971		
	{1,2,3,5,6,7}	75.8764	14	89.8764		
	{1,2,3,4,6,7}	53.6189	14	67.6189		
	{1,2,3,4,5,7}	**43.7645**	14	**57.7645**		
	{1,2,3,4,5,6}	49.4353	14	63.4353		

Table 3.6: Computational Steps of the Implicit Enumeration with Greedy Branching Strategy: Iteration 2

Iteration 2: S={6}, Ws={0}

	Model	f(M)	g(M)	z(M)	Upper Bound	Lower Bound
Core						
Augmented	same as iteration 1					
Core					=44.2971+2*3	
Replete	{1,2,3,4,5,7}	43.7645	14	57.7645		
Diminished	{2,3,4,5,7}	44.5231	12	56.5231		
Replete	{1,3,4,5,7}	48.782	12	60.782	56.2971	50.2971
	{1,2,4,5,7}	44.2971	12	**56.2971**		
	{1,2,3,5,7}	79.0858	12	91.0858		
	{1,2,3,4,7}	53.7468	12	65.7468		
	{1,2,3,4,5}	49.4741	12	61.4741		

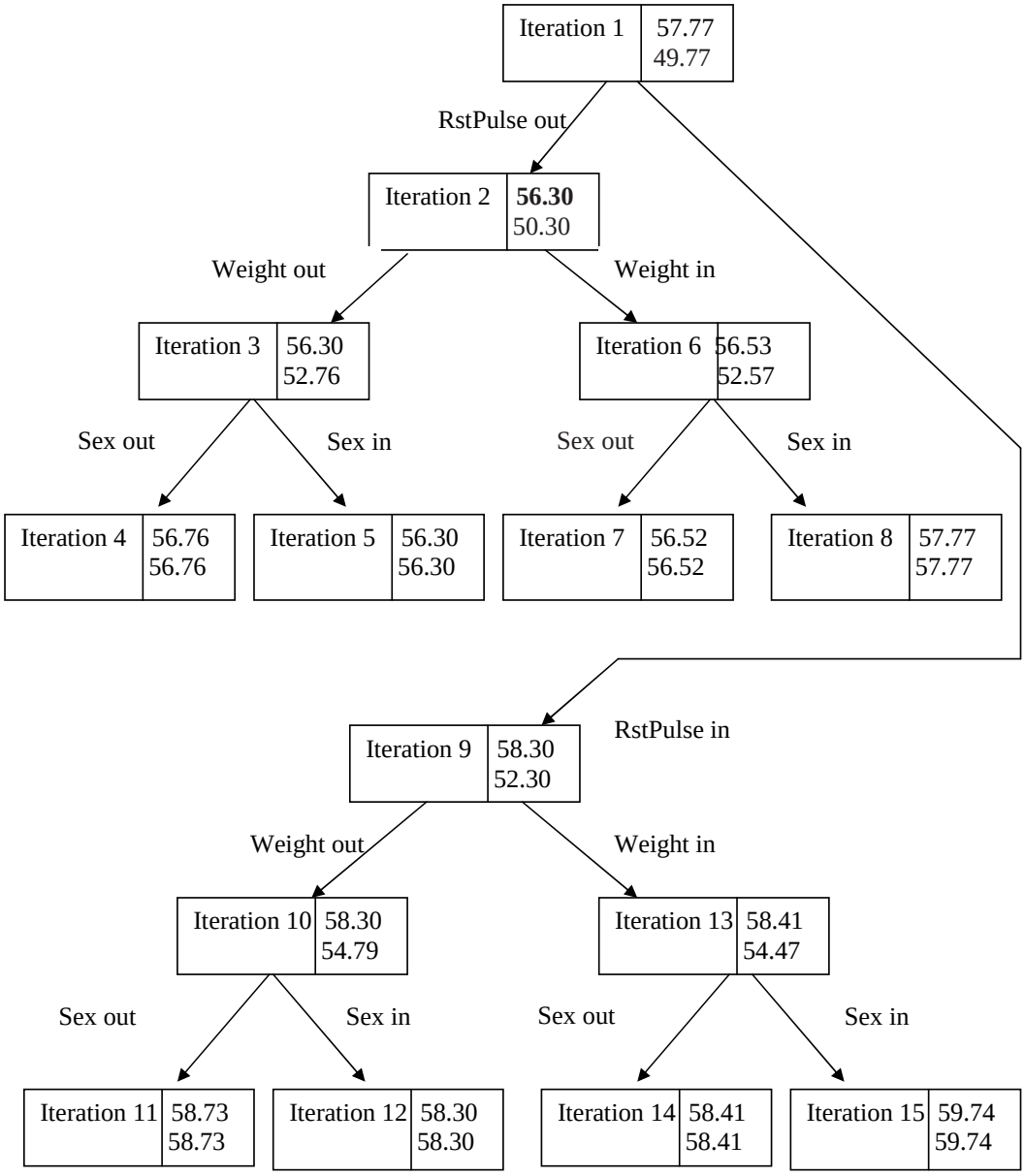

Figure 3.3: Enumeration Tree by Greedy Branching Strategy.

diminished replete solution is 57.7645 and is less than the augmented core solution 64.5424. Thus, we add variable 6 to S and fix it out of the model by adding a 0 to the w^S. Similarly, we observe that in iteration 2 the diminished replete solution has a criterion value less than the augmented core solution. Therefore, in iteration 3 the S set will contain variable 3 and 6 and they will be fixed out of the model. For the deletion strategy, we do not do any comparisons between the augmented core and diminished replete solution. We always add the variable that was deleted from the replete solution to form the diminished replete solution and fix it out of the model. In this example, the diminished replete models have criterion values less than the augmented core models, thus the deletion and greedy strategy turn out to be the same for the first 9 iterations.

3.6 Heuristic Implicit Enumeration

The computational efficiency of the implicit enumeration improves when more branches are cut. The upper bound we computed is exact. However, the lower bound we obtained is an estimated bound. If we can increase the lower bound, fewer model evaluations might be needed to find the optimal solution. In this section, we introduce a method to increase the lower bound to cut more branches. This method does not guarantee finding the optimal solution, thus termed heuristic implicit enumeration (HIE).

We introduce a multiplier $(1 + \tau)$ to the lower bound we obtained in

section 4.2.2. That is, the lower bound is computed as:

$$\underline{z}(S, w^S) = min(z(w_c), z(w_{ac}), z(w_r), z(w_{dr}), (1+\tau)\tilde{z}) = min(\bar{z}(S, w^S), (1+\tau)\tilde{z})$$

where

$$\tau = \begin{cases} 0 \leq \tau \leq 1 & \tilde{z} \geq 0 \\ -1 \leq \tau \leq 0 & \tilde{z} < 0 \end{cases}$$

When $\tau = 0$, $\underline{z}(S, w^S)$ is equal to the lower bound in section 4.2.2 and the solution is optimal. When $\tau > 0$, $\underline{z}(S, w^S)$ is a bigger value than the original lower bound value. Thus, this lower bound is able to cut more branches, that is, to fathom more solutions in the early stages of the execution of the algorithm. Some branches that were cut may contain the optimal solution. Therefore, there is no guarantee that the optimal solution can be found.

The HIE algorithm does not guarantee to find the best subset. However, the model it finds, if it is not the best, is very close to the best. Thus, HIE can provide some good alternative models for practitioners besides the best model.

Chapter 4

Computational Experience with the Implicit Enumeration Algorithm

This chapter reports the computational results of a series of experiments conducted on some real data sets with the implicit enumeration algorithm. The objectives of this chapter have two fold: First, real world data sets and AIC and ICOMP criteria are used to demonstrate the performance of the exact and heuristic implicit enumeration algorithms and compare different branching strategies. Second, the algorithm is compared with the existing algorithms.

Section 4.1 gives numerical examples on real data sets. Section 4.2 demonstrates the performance of the heuristic version of the implicit enumeration

algorithm. In section 4.3, some numerical examples on information criteria that the branch and bound algorithms fail to find the optimal model are provided. Finally, comparisons with the existing branch and bound, stepwise and all subset methods are made in section 4.4.

4.1 Computational Experience

In this section, we report on a set of examples that we have solved to demonstrate the algorithm and compare the three branching strategies. The implicit enumeration algorithm was implemented in Matlab. The three different branching strategies developed in section 3.4 are tested and compared on seven real data sets.

4.1.1 Experience on Real Data Sets

The data sets studied are chosen from the literature and are summarized in table 4.1.

We tried both Aug1 and Aug2 strategies, Del1 and Del2 strategies. Aug2 and Del2 have fewer total number of model evaluations on our test problems. Thus, the results reported here are from Aug2 and Del2 strategies.

We compared the branching strategies on two criteria: (1) the number of iterations required to find the optimal solution (but not necessarily proving optimality) and (2) the number of iterations required to find the optimal solution and prove its optimality. The variable deletion branching outper-

Table 4.1: Details of the Studied Data Sets

Data Set	# of Var.	# Obs	Description	Source
			Belle Ayr Liquefaction	Montgomery&Peck(1992)
Belle	7	27	Runs	p.491
			Aerobic	
Fitness	7	31	Fitness	JMP sample dataset
			Chemical	
Chemical	7	59	Degrassing	Wetherill (1986) p.250
			Monthly Use	
Steam	9	25	of Steam	Draper&Smith (1998)
			Gasoline	
			Mileage	Montgomery&Peck(1992)
Gas	11	30	Performance	p.489
			Body Fat	http://lib.stat.cmu.edu/datasets/
Bodyfat	13	252	Percentage	bodyfat
Crime	15	47	Crime Rate	Raftery (1995)

forms the other branching strategies on criteria (1) on all but one of the data sets. The variable augmentation branching strategy outperforms the other branching strategies with respect to the criteria (2) on all data sets.

Based on our limited number of experiments, the variable augmentation strategy would be considered the best choice, since it consistently converged to a proven optimal solution in fewer iterations. An exception would be for very large problems in which the user plans to terminate the iterations before optimality has been demonstrated. In this case the variable deletion strategy would be a better choice. The variable deletion strategy has a higher probability of terminating with the optimal solution as the incumbent, when enumeration is stopped before proof of optimality has been achieved.

Table 4.2 gives a comparison of the results of the three branching strate-

Table 4.2: Comparison of The Three Branching Strategies

	# of	Total	Total_Iter			Opt_Iter			Eval		
Data Set	**Var.**	**Models**	**Aug**	**Del**	**Gdy**	**Aug**	**Del**	**Gdy**	**Aug**	**Del**	**Gdy**
Belle	7	128	5	11	5	2	5	2	21	36	38
Fitness	7	128	9	15	15	5	2	2	30	47	80
Chemical	7	128	9	19	19	5	10	10	33	55	92
Steam	9	512	25	47	47	13	3	3	103	151	256
Gas	11	2048	57	89	57	53	8	53	229	311	378
Bodyfat	13	8192	43	423	423	8	5	5	199	1510	2598
Crime	15	32768	45	1057	1057	16	7	7	253	4156	7256

gies. The first column is the name of the data set. The second column is the number of candidate independent variables. The third column gives the total number of all possible subsets. Total_Iter represents the total number of iterations taken to determine the optimal solution (and prove its optimality). Opt_Iter represents the number of iterations taken to first reach the optimal solution (but not necessarily prove its optimality). Eval gives the number of models evaluated to find the optimal solution and verify its optimality. The last column gives the total number of possible models.

The variable augmentation strategy dominates the other two. The number of models evaluated to find and verify the optimal solution is smallest for the augmentation for each of the test problems. Figure 4.1 shows the evaluations required for each of the branching strategies.

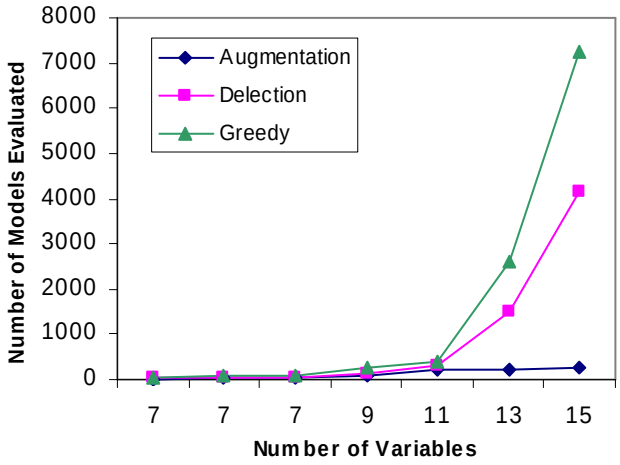

Figure 4.1: Comparison of Augmentation, Deletion and Greedy Branching Strategies: Number of models evaluated to find and verify optimal solution

4.2 Performance of the Heuristic Implicit Enumeration

The performance of the HIE is tested on the seven data sets from the previous section and the results are shown in table 4.3.

In order to test the effects of a fairly large range of τ values, we choose τ to be 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1 and 0.2. Tables 4.3 shows that when τ is greater than or equal to 0.1, the HIE algorithm was not able to find the optimal solution for any of the test data sets. The algorithm finds the optimal solution with fewer model evaluations for four out of the seven data sets. The best case happens to the gas data set, it is able to find the optimal solution with fewer than half of the model evaluations of the exact version. For practitioners, we suggest to use a τ value between 0-0.1.

Table 4.3: Heuristic Implicit Enumeration Performance

Data Set	τ	0	0.001	0.002	0.005
Belle	**Subset**	{6,7}	{6,7}		
	AIC	126.7495	126.7495		
	#Eval	**21**	21		
Fitness	**Subset**	{1,2,4,5,7}	{1,2,4,5,7}		
	AIC	56.2971	56.2971		
	#Eval	**30**	30		
Chemical	**Subset**	{1,3,4,5,7}	{1,3,4,5,7}	{1,3,4,5,7}	{1,3,4,7}
	AIC	408.6022	408.6022	408.6022	408.8136
	#Eval	**33**	33	**30**	26
Steam	**Subset**	{1,3,5,7,8,9}	{1,3,5,7,8,9}		
	AIC	-24.6577	-24.6577		
	#Eval	**103**	103		
Gas	**Subset**	{5,8,10}	{5,8,10}		{5,8,10}
	AIC	68.2899	68.2899		68.2899
	#Eval	**229**	217		206
Bodyfat	**Subset**	{1,2,4,6,7,8,12,13}	{1,2,4,6,7,8,12,13}	{1,2,4,6,7,8,12,13}	{1,2,4,6,8,12,13}
	AIC	741.8514	741.8514	741.8514	741.9088
	#Eval	**199**	179	**154**	107
Crime	**Subset**	{1,3,4,7,10,11,13,14}	{1,3,4,7,10,11,13,14}	{1,3,4,7,10,11,13,14}	{1,3,4,7,10,11,13,14}
	AIC	503.9349	503.9349	503.9349	503.9349
	#Eval	**253**	236	194	**179**

Table 4.3: Continued Heuristic Implicit Enumeration Performance

Data Set		τ = 0.01	τ = 0.02	τ = 0.05	τ = 0.1	τ = 0.2
Belle	**Subset**	{6,7}		{6}		
	AIC	126.7495		129.9051		
	#Eval	**15**		8		
Fitness	**Subset**	{1,2,4,5,7}	{2,4,5,7}			{1,2,3,4,5,6,7}
	AIC	56.2971	56.7603			59.7398
	#Eval	30	26			15
Chemical	**Subset**	{1,3,4,7}	{1,2,3,4,5,6,7}	{1,2,3,4,5,6,7}		
	AIC	408.8136	411.9596	411.9596		
	#Eval	26	15	8		
Steam	**Subset**	{1,3,5,7,8,9}	{1,3,5,7,8,9}	{1,3,5,7,8,9}	{1,3,5,6,7,8,9}	{4,5,7}
	AIC	-24.6577	-24.6577	-24.6577	-23.6086	-22.6252
	#Eval	103	100	**82**	72	58
Gas	**Subset**	{5,8,10}		{5,8,10}	{1,4}	{1}
	AIC	68.2899		68.2899	70.1968	70.2349
	#Eval	187		**100**	50	12
Bodyfat	**Subset**	{1,2,4,6,12,13}	{2,6,12,13}	{1,2,3,4,5,6,7,8,9,10,11,12,13}		
	AIC	743.6612	745.0747	749.3574		
	#Eval	99	50	14		
Crime	**Subset**	{1,3,4,11,13,14}	{1,3,4,11,13,14}	{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}		
	AIC	504.7859	504.7859	514.6488		
	#Eval	141	95	31		

4.3 Information Complexity ICOMP Criterion

In this section, we will demonstrate the performance of the implicit enumeration algorithm with the information criteria where the second term $g(w)$ is not the same for subsets of the same size. IE only requires that $g(w_2) \geq g(w_1)$ if $w_2 \geq w_1$.

Bozdogan has developed a family of criterion that follows the property mentioned above in a series of papers. The criteria includes CAICF, ICOMP-(IFIM), $ICOMP_{Misspec}$ etc. The ICOMP criterion family resembles AIC, but incorporates a measure of the complexity of the model into the penalty term. ICOMP penalizes the covariance complexity of the model instead of penalizing the number of free parameters directly.

For general univariate and multivariate models, the ICOMP criterion is defined by

$$ICOMP = -2\log(L(\hat{\theta})) + 2C_1(\hat{\mathcal{F}}^{-1})$$

where $L(\hat{\theta})$ is the likelihood function and $C_1(\hat{\mathcal{F}}^{-1})$ is the maximal informational complexity of the estimated IFIM of the model. In particular,

$$C_1(\hat{\mathcal{F}}^{-1}) = \frac{s}{2}\log[\frac{trace(\hat{\mathcal{F}}^{-1})}{s}] - \frac{1}{2}\log|\hat{\mathcal{F}}^{-1}|$$

where $s = dim(\hat{\mathcal{F}}^{-1}) = rank(\hat{\mathcal{F}}^{-1})$.

We write ICOMP(IFIM) criterion shown in section 1.2 in a general format

of the information criterion:

$$ICOMP(IFIM) = f(w) + g(w)$$

where

$$f(w) = -2\log(L(\hat{\theta}))$$

and

$$g(w) = C_1(\hat{\mathcal{F}}^{-1}) = \frac{s}{2}\log[\frac{trace(\hat{\mathcal{F}}^{-1})}{s}] - \frac{1}{2}\log|\hat{\mathcal{F}}^{-1}|$$

$f(w)$ and $g(w)$ follows the monotonic condition. Bozdogan (1990) has the proof that $2C_1(\hat{\mathcal{F}}^{-1})$ is a monotonic function. Interested readers are referred to that paper for details of the proof.

We first look at a small example. The data set is the hald data set (Hald 1952). It has four variables. All possible subsets are computed, ICOMP(IFIM) criterion value and the value of $f(w)$ and $g(w)$ is listed in table 4.4.

The results from all possible subsets show that ICOMP(IFIM) achieves its minimum at subset {1,2} with a value of 32.7304. The C1ifim ($C_1(\hat{\mathcal{F}}^{-1})$) column clearly indicates that the second term of this criterion varies for subset of the same size differs. In fact, the second term $g(w)$ does not necessarily decrease as the number of variables decreases. The $C_1(\hat{\mathcal{F}}^{-1})$ of the three-variable subset {1,3,4} is 10.6824 and is smaller than the two-variable subset of {2,4}, which has a value of 11.6356. In this case, the branch and bound

Table 4.4: ICOMP(IFIM) Scores for All Subsets of Hald Data Set

k	Subset	ICOMP(IFIM)	Lackoffit	C1ifim
1	1 2 3 4	71.2321	17.1146	27.0587
2	2 3 4 0	56.1833	22.5898	16.7968
3	1 3 4 0	39.2156	17.8508	**10.6824**
4	1 2 4 0	49.5387	17.136	16.2013
5	1 2 3 0	37.464	17.1916	10.1362
6	3 4 0 0	48.3779	33.8162	7.2809
7	2 4 0 0	77.8703	54.5991	**11.6356**
8	1 4 0 0	35.7037	22.8636	6.42
9	2 3 0 0	61.8813	44.9898	8.4457
10	1 3 0 0	75.8235	59.084	8.3698
11	1 2 0 0	**32.7303**	19.5204	6.6049
12	4 0 0 0	67.001	54.8233	6.0888
13	3 0 0 0	76.7967	65.0218	5.8874
14	2 0 0 0	67.3635	55.1439	6.1098
15	1 0 0 0	70.2307	59.4826	5.374

algorithm is not able to find the optimal subset. The branch and bound algorithm only finds the smallest $f(w)$ of each size. A smallest value of $f(w)$ does not often mean a smallest ICOMP(IFIM). For example, subset {1,2,4} has the smallest $f(w)$ for three variable subset, but its ICOMP(IFIM) value 49.5387 is by no means the smallest among subsets of size 3. The implicit enumeration is successful in finding the "best" subset {1,2}.

We use the fitness data set as another example. Part of the result of all possible subsets is shown in table 4.5. By looking at the "C1ifim" column, we can see that the C1ifim of the six-variable Subset {1,3,4,5,6,7} is 22.6263, while the C1ifim of the five-variable subset {1,2,3,6,7} is 22.9913. Another example is that the two-variable subset {1,4} has a C1ifim value of 4.7297, while the one-variable subset {7} has a C1ifim value of 7.6528. However, in

Table 4.5: ICOMP(IFIM) Scores for All Subsets of Fitness Data Set

k	Subset	ICOMP(IFIM)	C1Cov	AIC
1	1 2 3 4 5 6 7	99.7123	27.9863	59.7398
2	2 3 4 5 6 7 0	96.3492	25.9691	58.411
3	1 3 4 5 6 7 0	92.6707	22.0277	62.6153
4	1 2 4 5 6 7 0	90.7012	23.202	58.2971
5	1 2 3 5 6 7 0	127.68	25.9018	89.8764
6	1 2 3 4 6 7 0	102.9584	24.6697	67.6189
7	1 2 3 4 5 7 0	91.5961	23.9158	57.7645
8	1 2 3 4 5 6 0	97.9333	24.249	63.4353
9	3 4 5 6 7 0 0	90.9931	20.4007	62.1916
10	2 4 5 6 7 0 0	90.2686	21.7678	58.7329
11	1 4 5 6 7 0 0	84.3228	17.8503	60.6222
12	2 3 5 6 7 0 0	125.748	23.8697	90.0086
13	1 3 5 6 7 0 0	122.856	19.9174	95.0212
14	1 2 5 6 7 0 0	118.6078	21.1834	88.2409
15	2 3 4 6 7 0 0	99.8695	22.6592	66.5511
16	1 3 4 6 7 0 0	96.2755	19.0262	70.2231
17	1 2 4 6 7 0 0	93.2655	19.8121	65.6414
18	1 2 3 6 7 0 0	127.9625	22.5038	94.9549
19	2 3 4 5 7 0 0	88.7032	22.0901	56.5231
20	1 3 4 5 7 0 0	85.598	18.408	60.782
21	1 2 4 5 7 0 0	82.9186	19.3107	**56.2971**
22	1 2 3 5 7 0 0	122.4956	21.7049	91.0858
....				
114	1 5 0 0 0 0 0	108.0043	8.5746	96.8551
115	3 4 0 0 0 0 0	76.5297	8.0885	66.3527
116	2 4 0 0 0 0 0	72.8705	7.5101	63.8503
117	1 4 0 0 0 0 0	**67.8408**	4.6209	64.599
118	2 3 0 0 0 0 0	117.8641	10.2292	103.4056
119	1 3 0 0 0 0 0	107.932	7.1566	99.6189
120	1 2 0 0 0 0 0	103.1685	6.0901	96.9882
121	7 0 0 0 0 0 0	115.7693	7.4262	104.9169
122	6 0 0 0 0 0 0	107.8595	5.2699	101.3197
123	5 0 0 0 0 0 0	111.8904	7.2668	101.3569
124	4 0 0 0 0 0 0	68.0227	3.7401	64.5424
125	3 0 0 0 0 0 0	113.6997	5.914	105.8717
126	2 0 0 0 0 0 0	110.3462	5.4037	103.5387
127	1 0 0 0 0 0 0	95.246	0.4167	98.4126

general, C1ifim does decrease as the number of variables decreases.

The implicit enumeration algorithm is run and after 15 iterations and evaluation of 78 models out of 127 total possible models, it finds the optimal subset of {1,4} with ICOMP(IFIM) score of 68.0584.

4.4 Comparison with the Existing Methods

In this section, we first compare the implicit enumeration algorithm with the branch and bound algorithm proposed by Gatu and Kontoghiorghes (2003). In their paper, they demonstrate that their branch and bound algorithm is faster than the leaps and bound algorithm by Furnival and Wilson (1974). Then we will compare IE with the all possible subset method and the stepwise method.

We note that branch and bound algorithms are not applicable to information criteria where the second term is a monotone function, but is not the same for subsets of the same size. The branch and bound algorithms need to find the best subset of each size in order to obtain the "best" subset. The implicit enumeration algorithm, on the other hand, finds the best subset directly. It is worth noting that although the implicit enumeration algorithm does not find the best subset of each size, the core subsets in some cases are the best of that size. If the core subset does not have the best criterion value of that size, it gives quite competitive subset.

4.4.1 Comparison with the Branch and Bound Algorithms

Gatu and Kontoghiorghes (2003) has proposed a branch and bound algorithm (BBA) that only builds one tree and they showed that the BBA algorithm has outperformed the Furnival and Wilson's leaps and bounds algorithm. They also designed two heuristic versions of the BBA termed HBBA. When $\tau = 0$, HBBA is equivalent to the exact BBA and HIE is equivalent to the exact IE.

We use the same experimental data sets as those used in Gatu et al. (2003). The number of model evaluations and computational time for both data sets are shown in table 4.6. The ozone data set has 8 variables and 330 observations. The pollute data set has 15 variables and 60 observations.

Table 4.6 gives the comparison on number of node evaluations for HBBA and model evaluations for HIE. GK's tree was built so that each node contains evaluations of subsets of different sizes. The actual number of subset

Table 4.6: Comparison with GKBB

τ	Ozone		Pollute	
	HBBA Node Eval	**HIE Subset Eval**	**HBBA Node Eval**	**HIE Subset Eval**
0	*143	*51	*710	*799
0.01	*130	25	*608	*277
0.025	105	10	*523	95
0.05	90	10	438	59
0.1	60	10	335	17
0.25	21	10	134	17

* optimal solution was found.

evaluations can be a lot more than the node evaluations. The results are reproduced from Gatu et al. (2003).

For the pollute data set, HIE with τ value of 0.01 was able to evaluate 277 subsets to find the optimum. The smallest number HBBA needs is 523 node evaluations to find the optimum. HIE seems to be more sensitive to the τ value in that the number of evaluations decreases more dramatically than the HBBA does. By looking at both the exact and heuristic version results, we observe that HIE has the smaller number of subset model evaluations and much faster convergence overall.

4.4.2 Comparison with All Possible Subsets Method

Table 4.7 showed the time comparison of all possible subsets method and the implicit enumeration on data sets with 16-30 variables. The all subsets method is terminated prematurely for data sets with 21-30 variables because of the substantial computation time required.

Table 4.8 reports the computational time comparison of all possible subsets method and IE algorithm on data sets with various sample sizes. We simulated seven data sets with 16 variables and with 1000, 5000, 10000, 20000, 50000, 100000 and 500000 observations respectively. The predictor variables are generated from uniform distributions. The response variable is generated by using some linear combinations of the predictors.

Both algorithms are implemented and tested on a PC with 3.0 GHz Pentium 4 processor and 1 Gb RAM. The computation time to find the best

Table 4.7: Comparison with All Possible Subsets

	All Subsets		IE	
Variables	**Time**	**#Eval**	**Time**	**#Eval**
16	63.10	65,536	0.38	460
17	105.23	131,072	0.34	376
18	274.14	262,144	0.59	688
19	796.19	524,288	3.72	3,918
20	2513.20	1,048,576	1.98	2,040
21	49913*	2,097,152	0.27	230
22	49913*	4,194,304	3.89	3,608
23	49913*	8,388,608	26.02	20,986
24	49913*	16,777,216	1.69	1,278
25	49913*	33,554,432	11.17	8,684
26	49913*	67,108,864	15.92	11,484
27	49913*	134,217,728	5.95	3,896
28	49913*	268,435,456	39.61	24,972
29	49913*	536,870,912	15.70	8,682
30	49913*	1,073,741,824	40.80	23,636

* Computation time of 21-variable data set up to subset size 12;
All other data sets are terminated before completion

Table 4.8: Time Comparison of All Possible Subsets and IE on Simulated Data Sets

Observations	**All Subsets**	**IE**
1000	49.3	0.3
5000	279.4	3.8
10000	655.8	3.6
20000	1413.0	3.8
50000	3224.1	3.6
100000	5520.8	9.7
500000	28595.0	90.4

subsets are shown in the table.

The implicit enumeration algorithm clearly outperforms the all possible subsets method on all simulated data sets.

4.4.3 Comparison with Stepwise Methods

We also ran our test problems using the stepwise regression procedure in SAS. The results of the stepwise methods depend on the entering and stopping rules. Different rules often lead to different subsets for the same problem. We have chosen the entering and staying rules according to SAS default. That is, the significance level for a variable to enter and stay in the model is 0.15. A variable will not enter the model if the p-value is greater than 0.15.

The stepwise procedure found the optimal solution in one of the seven test data sets and failed to find the optimal subset in six of the data sets.

These results are shown in table 4.9.

Table 4.9: Results Comparison with Stepwise Procedure in SAS

Data Set	Solution		AIC	
	IE	Stepwise	IE	Stepwise
Belle	6,7	6,7	127	127
Fitness	1,2,4,5,7	2,4,5,7	**56**	**57**
Chemical	1,3,4,5,7	1,3,4,7	**409**	**409**
Steam	1,3,5,7,8,9	1,4,5,7	**-25**	**-23**
Gas	5,8,10	1	**68**	**70**
Bodyfat	1,2,4,6,7,8,12,13	1,2,4,6,12,13	**742**	**744**
Crime	1,3,4,7,10,11,13,14	1,3,4,11,13,14	**504**	**505**

Chapter 5

Genetic Algorithm and the Hybrid Approach

Genetic Algorithms (GA) were formally introduced in the 1970s by John Holland. Goldberg (1989) and many others have popularized the GAs in the 1980s. Since then, GA has been widely used in many different areas to solve various kinds of problems such as production scheduling problems, traveling sales man's problem and circuit layout.

The GA presented in this chapter is based on the work of Luh, Minesky and Bozdogan (1997), which in turn basically follows Goldberg (1989). A brief introduction to the genetic algorithm in model selection is given in section 5.1. Section 5.2 discusses the advantages and disadvantages of the GAs. GA is a stochastic search algorithm, which can be applied to huge model selection portfolios. GA does not often guarantee finding the optimal solution,

but they give near optimal solutions for complex problems. Thus, a hybridization of the genetic algorithm and the implicit enumeration algorithm is proposed and developed in section 5.3. The hybrid method can be more efficient in finding the "best" model than the pure exact algorithm, which greatly improves the performance of the pure exact and heuristic algorithms. We demonstrate this combinatorial methodology with some numerical examples in section 5.4.

5.1 Algorithmic Description

Genetic Algorithms were developed to mimic some of the processes observed in natural evolution. To use a genetic algorithm, the solution of the problem must be represented as a genome (or chromosome). The genetic algorithm then creates a population of solutions and applies genetic operators such as mutation and crossover to evolve the solutions in order to find the best one. There are three most important aspects of using genetic algorithms: (1) the definition of the objective function, (2) the definition and implementation of the genetic coding scheme, and (3) the definition and implementation of the genetic operators. Once these three have been defined, the generic genetic algorithm should work fairly well. The algorithm is implemented using the following steps in model selection:

1. *Implementing a genetic coding scheme:* The first step of the GA is to represent each subset model as a binary string. A binary code of 1

indicating presence and a 0 indicating absence. Every string is of the same length, but contain different combinations of predictor variables, for example, a binary string of 10010 represents a model including variable x_1 and x_4, but excluding x_2, x_3 and x_5.

2. *Generating an initial population of the models:* The initial population consists of randomly selected models from all possible models. We have to choose an initial population of size p. Our algorithm allows one to choose any population size. The best population size to choose depends on many different factors and requires further investigation.

3. *Using a fitness function to evaluate the performance of the models in the population:* A fitness function provides a way of evaluating the performance of the models. Information criteria are used as the fitness functions. In general, the analyst has the freedom of using any appropriate model selection criterion as the fitness functions. The comparison of different model selection criteria's performances has been reported in many other literatures, for example, McQuarrie and Tsai (1998).

4. *Selecting the parents models from the current population:* This step is to choose models to be used in the next step to generate new population. The selection of parents' models is based on the natural selection. That is, the model with better fitness value has greater chance to be selected as parents. Let z_i denote the information criterion value for the ith model in the population. Let z_{MAX} denote the maximum criterion

value of that generation. We calculate the differences:

$$\triangle z_{(i)} = z_{MAX} - z_i$$

for $i = 1, ..., p$, where p is the population size. We then average these differences. The ratio of each model's difference value to the mean difference value is then calculated. The chance of a model being used as parent is proportional to this ratio.

5. *Produce offspring models by crossover and mutation process:* The selected parents are then used to generate off-springs by performing crossover and/or mutation processes on them. Both the crossover and mutation probabilities are determined by the analyst. A higher crossover probability will introduce more new models into the population in each generation. However, it will remove more of the good models from the previous generation. A mutation probability is a random search operator. It helps to jump to another search area within the solutions' scope. Lin and Lee (1996) states that mutation should be used sparingly because the algorithm will become little more than a random search with a high mutation probability.

There are different ways of performing the crossover, for example, uniform crossover, single point crossover and two-point crossover. Interested readers are referred to Goldberg (1989) for more details. The method of crossover is

also determined by the analyst.

5.2 Advantages and Disadvantages of Using Genetic Algorithm

Similar to the problems of applications of genetic algorithms in many different fields, the algorithm in the model selection framework does have some advantages and disadvantages. They are summarized as follows:

Advantages:

1. It can handle data sets of virtually any size.

2. There are no specific requirements on the information criterion function. The function does not need to be monotone, continuous or differentiable.

3. It can return several good models, instead of a single best model.

Disadvantages:

1. One of the more challenging aspects of using genetic algorithms is to choose the configuration parameter settings. Discussion of GA theory provides little guidance for proper selection of the settings. The parameters include the number of generations, population size, crossover and mutation probabilities. There are also various ways of performing the crossover operation and selecting the mating parents. Common

crossover operations are the uniform, single point, two-point and shuffle crossover. Common parents selection methods are natural and random selection. The parameter and operations choices can greatly influence the performance of the algorithm. Although many researchers have attempted to use experimental design to assist GA parameters configuration, choosing the optimal combinations of them still remains a question to be solved.

2. The genetic algorithm does not guaranteed to find the optimal solution. It only finds a "good" solution. The solution can be optimal sometimes depending on the various choices of parameters and operations by the analyst.

5.3 Hybrid of GA and IE

GAs are very effective in high-dimensional search spaces. However, the major drawback of GA is that there are several parameters need to be set before the runs. No real guidance to these parameter choices exists and bad parameter choices can lead to suboptimal solutions. GA does not guarantee to find the optimal solution because it is a stochastic search algorithm. Different runs of GA can produce very different solutions.

Due to the drawbacks of the GA, we propose a hybrid of GA and the implicit enumeration algorithm to find the optimal solution. This hybrid methodology can be less computationally expensive than using the pure im-

plicit enumeration algorithm.

The implicit enumeration algorithm is ideal for the hybrid approach. It can use any solution as a starting point to find the optimal solution. A good starting point can greatly reduce the computations.

It is worth noting that given a good solution by GA, the existing branch and bound algorithm can only find the best solution of the size given by the solution of GA or the solution with sizes greater than that given by the GA. The size of the solution has to be specified by the researcher.

We propose three ways to implement this:

First, the solution from the GA runs provides a starting point for the implicit enumeration algorithm. This initial solution provided by the GA is fixed in the final solution. This approach does not guarantee the finding of the optimal solution because the initial solution provided by GA may contain variables that are not present in the optimal solution. The final solution is optimal when the initial solution provided by GA has less than or has exactly the same variables as those in the optimal solution. Thus, a criterion that tends to under-fit a model is used for the initial GA runs. Some examples of such criteria are CAIC and MDL as suggested by Bozdogan (1994) "CAIC and MDL penalizes models too stringently, to the point that it under fits the true model". We call this approach GAIE-1. Variable augmentation branching strategy is used with GAIE-1.

Second, the solution from the GA runs is used as a starting point just as in the first approach. However, the variables that are not in the solution

are fixed out of the final solution. Variable deletion branching strategy is the most suitable to be used with this approach. We call this approach GAIE-2. Like GAIE-1, GAIE-2 does not guarantee the finding of the optimal solution because the initial solution provided by the GA may exclude some variables that are in the optimal solution.

GAIE-1 and GAIE-2 could be used parallel to each other. The solution provided by GA serves as a starting solution for both the variable augmentation and variable deletion strategies. The combined usage of the two approaches greatly increased the chance of finding the optimal solution. We will demonstrate this in the next section.

Third, The initial solution provided by the GA is used as an upper bound in the implicit enumeration instead of being the starting point of the algorithm. The computational savings of this hybrid methodology comes from fewer model evaluations in the implicit enumeration by being able to cut branches faster due to tighter bounds. The worst situation is that the solution provided by GA is no better than the upper bound obtained in the first iteration from the implicit enumeration. There are no extra fathoming of the solutions. We call this approach GAIE-3.

Another possible approach is that the solution provided by the GA is used as a starting point for the implicit enumeration algorithm. The implicit enumeration algorithm is then used to find the best solution with the given set of variables. The solution from the implicit enumeration can be feed back into the GA as a parent again. The solution from GA can then serve as a

starting point of the implicit enumeration algorithm. The loop could go on and on until a satisfactory solution is found or until there is no improvement on the current solution.

In the next section, we use the small real world data from chapter 4 to demonstrate the hybrid methodology and show its advantages over the pure GA and pure IE approach. Results of the applications on larger data sets will be given in the next chapter.

5.4 Computational Examples and Comparisons

In this section, we demonstrate the performance of the hybrid approaches GAIE-1, GAIE-2 and GAIE-3. Their performances will also be demonstrated in the next chapter with large data sets.

5.4.1 GAIE-1

The seven real world data sets from chapter 4 is used as benchmark data sets. GA parameters are chosen as shown in table 5.1. The population size and number of generations of the GA are chosen arbitrarily. The probability of crossover is 0.5 and the probability of mutation is 0.01 for all runs. CAIC is used as the fitness function for the GA runs.

From table 5.2, we see that for the fitness, gas and bodyfat data sets, GA & IE hybrid approach needs fewer model evaluations to find the optimal solution. It works very well with the gas and bodyfat data sets. For the

Table 5.1: GA Parameter Choices

Data Set	# of Var.	Population Size	No. of Generation	GA Eval
Belle	7	5	5	25
Fitness	7	5	5	25
Chemical	7	5	5	25
Steam	9	10	6	60
Gas	11	10	8	80
Bodyfat	13	15	10	150
Crime	15	15	10	150

Table 5.2: GAIE1 results

	Solution			Model Evals	
Data Set	**Optimum**	**Pure GA**	**GAIE-1**	**Pure IE**	**GAIE-1**
Belle	6,7	4,5,6,7	4,5,6,7	21	29
Fitness	1,2,4,5,7	1,2,4,5	1,2,4,5,7	**30**	**29**
Chemical	1,3,4,5,7	1,2,3,4,5,7	1,2,3,4,5,7	33	26
Steam	1,3,5,7,8,9	1,2,3,5,7,8,9	1,2,3,4,5,7,8,9	103	63
Gas	5,8,10	8,10	5,8,10	**229**	**107**
Bodyfat	1,2,4,6,7,8,12,13	2,4,6,7,8,12,13	1,2,4,6,7,8,12,13	**199**	**163**
Crime	1,3,4,7,10,11,13,14	1,3,4,7,9,10,11,13,14	1,3,4,7,9,10,11,13,14	253	157

gas data set, only 107 models are evaluated before it obtains the optimal solution compared with the evaluation of 229 models by the pure implicit enumeration approach. For the bodyfat data set, only 163 models needed to be evaluated compared with the 199 model evaluations by the pure IE approach. Although the hybrid approach did not find the optimal solution for the belle, chemical, steam and crime data sets, the solutions it found are quite competitive with the optimal solution. Except for the Belle data set, GAIE-1 greatly improved the initial solution provided by GA.

5.4.2 Combined Usage of GAIE-1 and GAIE-2

In addition to the GAIE-1 approach, GAIE-2 are run parallel with GAIE-1. Table 5.3 gives the results of the combined usage of GAIE-1 and GAIE-2. Total represents the total number of model evaluations by the GA and IE. GAIE-1 obtains the optimal solutions for the fitness, gas and bodyfat data set. GAIE-2 obtains the optimal solutions for the belle, chemical, steam and crime data Set. The combined methodology finds the optimal solution for all tested data sets. In particular, it finds the optimal solution for the gas and crime data sets with fewer number of model evaluations than the pure implicit enumeration algorithm.

Table 5.3: GAIE1 and GAIE2 Combined Usage Results

	Solution		Model Evals			
Data Set	**Optimum**	**Pure GA**	**GAIE-1**	**GAIE-2**	**Total**	**Pure IE**
Belle	6,7	4,5,6,7	4	**9**	38	21
Fitness	1,2,4,5,7	1,2,4,5	**4**	9	38	30
Chemical	1,3,4,5,7	1,2,3,4,5,7	1	**27**	53	33
Steam	1,3,5,7,8,9	1,2,3,5,7,8,9	3	**47**	110	103
Gas	5,8,10	8,10	**27**	4	**111**	**229**
Bodyfat	1,2,4,6,7,8,12,13	2,4,6,7,8,12,13	**13**	47	210	199
Crime	1,3,4,7,10,11,13,14	1,3,4,7,9,10,11,13,14	7	**93**	**250**	**253**

Table 5.4: GAIE3 Results

	Solution		Model Evals	
Data Set	**Optimum**	**Pure GA**	**Pure IE**	**GAIE-3**
Belle	6,7	4,5,6,7	21	46
Fitness	1,2,4,5,7	1,2,4,5	30	55
Chemical	1,3,4,5,7	1,2,3,4,5,7	33	58
Steam	1,3,5,7,8,9	1,2,3,5,7,8,9	103	163
Gas	5,8,10	8,10	229	309
Bodyfat	1,2,4,6,7,8,12,13	2,4,6,7,8,12,13	199	349
Crime	1,3,4,7,10,11,13,14	1,3,4,7,9,10,11,13,14	253	403

5.4.3 GAIE-3

We again tested the GAIE-3 on the same data sets. AIC is used as the fitness function for the GA runs. Table 5.4 shows the results. The GAIE-3 does not perform better than the pure IE approach for all data sets. This is due to the fact that our upper bounds obtained in IE in early stages are very good bounds. The solution provided by GA does not improve this upper bound much. Thus, not many extra solutions can be fathomed.

Chapter 6

Applications

In this chapter, we demonstrate the applications of the implicit enumeration algorithm using some real and simulated large data sets. The performance of the implicit enumeration algorithm is compared with that of the genetic algorithm and the hybrid approach. Computational time and number of model evaluations are used as criteria to compare the performance. All the test runs were done on a PC with a Pentium IV 3.0 GHz processor and 1 Gb of RAM.

The algorithm can be applied to a wide variety of subset model selection problems. We demonstrate its applications in logistic regression in section 6.1, multivariate regression in section 6.2 and discriminant analysis in section 6.3.

6.1 Logistic Regression

Logistic regression is a technique for analyzing problems that have one or more independent variables that determine a dichotomous outcome. This situation poses problems for the assumptions of the ordinary least squares regression that the error variances are normally distributed. Thus, rather than choosing parameters that minimize the sum of squared errors, estimation in logistic regression chooses parameters that maximize the likelihood of observing the sample values.

From a practical standpoint, logistic regression and least squares regression are almost identical. Both methods produce prediction equations. In both cases the regression coefficients measure the predictive capability of the independent variables.

The logistic regression model can be written as follows:

$$\pi' = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_J X_J + \epsilon, \tag{6.1}$$

where π' is logit transformation of the probability of a outcome of an event π. This logit transformation is defined as

$$\pi' = log_e \left(\frac{\pi}{1 - \pi} \right). \tag{6.2}$$

The logit mean has a range from $-\infty$ to ∞ as X ranges within -∞ to ∞.

Since each Y_i observation is a Bernoulli random variable, where:

$$\begin{aligned} P(Y_i = 1) &= \pi_i \\ P(Y_i = 0) &= 1 - \pi_i \end{aligned} \tag{6.3}$$

We represent the probability distribution of each observation Y_i as follows:

$$f_i(Y_i) = \pi_i^{Y_i}(1 - \pi_i)^{1-Y_i} \tag{6.4}$$

where $Y_i = 0$ or 1 and i=1, ...,n.

Since Y_i observations are independent, their joint probability function is:

$$\begin{aligned} g(Y_1, ..., Y_n) &= \prod_{i=1}^{n} f_i(Y_i) = \prod_{i=1}^{n} \pi_i^{Y_i}(1 - \pi_i)^{1-Y_i} \\ &= \sum_{i=1}^{n} \left[Y_i log_e \left(\frac{\pi_i}{1 - \pi_i} \right) \right] + \sum_{i=1}^{n} log_e(1 - \pi_i) \end{aligned} \tag{6.5}$$

The logarithm of the joint probability function is:

$$log_e[g(Y_1, ..., Y_n)] = log_e \prod_{i=1}^{n} \pi_i^{Y_i}(1 - \pi_i)^{1-Y_i} \tag{6.6}$$

Since $log_e \left(\frac{\pi_i}{1-\pi_i} \right) = \beta_0 + \beta_1 x_1 + ... + \beta_J x_J$, Hence, equation (6.4) can be

expressed as follows:

$$log_e(L(\hat{\beta})) = \sum_{i=1}^{n} Y_i(\beta_0+\beta_1 x_1+...+\beta_J x_J) - \sum_{i=1}^{n} log_e[1+exp(\beta_0+\beta_1 x_1+...+\beta_J x_J)] \tag{6.7}$$

where $L(\hat{\beta})$ is the likelihood function.

Thus, AIC for logistic regression can be computed as:

$$AIC = -2log_e(L(\hat{\beta})) + 2k, \tag{6.8}$$

where k is the number of parameters in the model.

ICOMP for logistic regression can be computed as:

$$ICOMP(IFIM) = -2log_e(L(\hat{\beta})) + 2C_1(\hat{\mathcal{F}}^{-1}(\hat{\theta})), \tag{6.9}$$

where C_1 denotes the maximal information complexity of $\hat{\mathcal{F}}^{-1}$, and C_1 is defined as

$$C_1(\hat{\mathcal{F}}^{-1}) = \frac{s}{2} log[\frac{tr(\hat{\mathcal{F}}^{-1})}{p}] - \frac{1}{2} log|\hat{\mathcal{F}}^{-1}|,$$

where s is the dimension or rank of $\hat{\mathcal{F}}^{-1}$.

In this section, we will use four large examples to demonstrate the performance of the implicit enumeration algorithm with logistic regression.

The first example is a data set of n=200 subjects who were part of a much larger study on survival of patients following admission to an adult intensive care unit (ICU). The data set has 19 explanatory variables. The detailed

Table 6.1: ICU Data Description

Vars	Description
$X1$	Age (years)
$X2$	Sex: 0,Male; 1,Female
$X3$	Race: 1,While;2,Black;3,Other
$X4$	Service at ICU admission: 0,Medical;1,Surgical
$X5$	Cancer part of present problem: 0,No;1,Yes
$X6$	History of chronic renal failure: 0,No;1,Yes
$X7$	Infection probability at ICU admission: 0,No;1,Yes
$X8$	CPR prior to ICU admission: 0,No;1,Yes
$X9$	Systolic blood pressure at ICU admission (mmHg)
$X10$	Heart rate (Beats/min)
$X11$	Previous admission within 6 months: 0,No;1,Yes
$X12$	Type of admission: 0,Elective;1,Emergency
$X13$	Long bone, or hip fracture: 0,No;1,Yes
$X14$	PO2 from initial blood bases: 0,>=60;1,=60
$X15$	PH from initial blood gases: 0,=7.25;1<7.25
$X16$	PCO2 from initial blood gases: 0,=45;1,>45
$X17$	Bicarbonate from initial blood gases: 0,=18;1,<18
$X18$	Creatinine from initial blood gases: 0,=2;1,>2
$X19$	Level of consciousness: 0,No Coma or Stupor;1,Deep Stupor;2,Coma

description of the data set is in table 6.1. More details can be found in the book by Hosmer & Lemeshow (2000). The major goal of this study was to develop a logistic regression model to predict the probability of survival to hospital discharge of these patients. So the response or the dependent variable is vital status (0, lived; 1, died) of the patient.

An exhaustive search will need to explore a model space comprising

$$2^{19} = 524,288$$

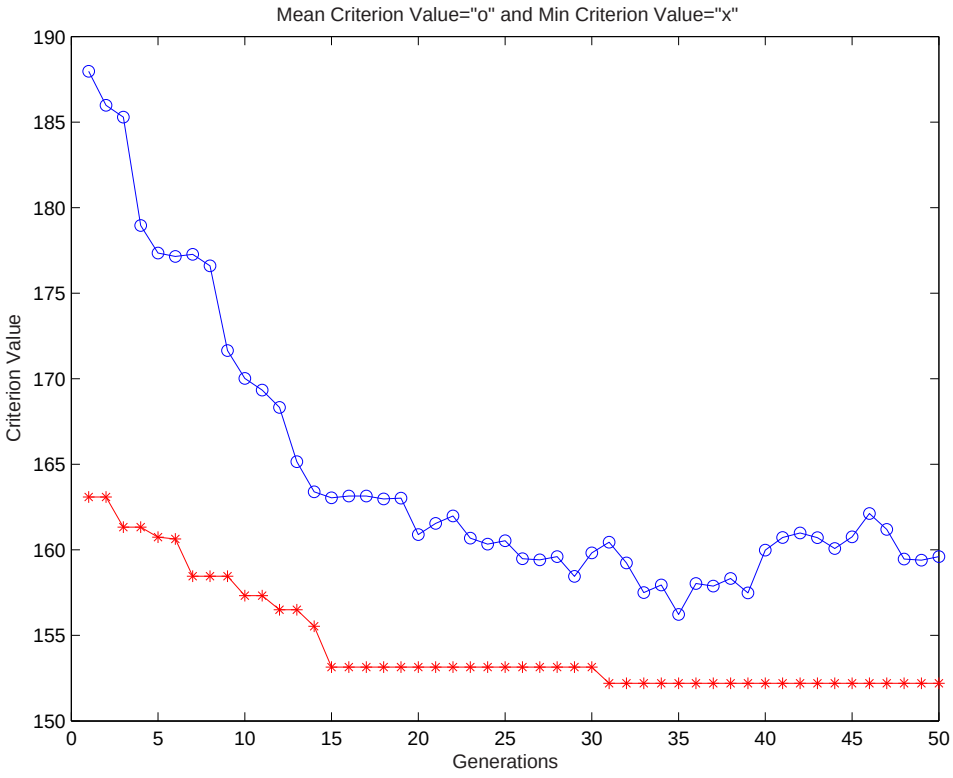

Figure 6.1: One Run of GA for the ICU Data Set

models. The implicit enumeration algorithm with the variable augmentation branching strategy, however, only needs to evaluate 688 models to determine the best model. The best model consists of variables 1, 5, 9, 12, 15, 16, 19 and has an associated AIC value of 152.2019. It took 4.782 seconds to find the optimal model. In comparison, one run of GA took 12.7030 seconds to complete and found the optimal model. The GA parameters are chosen as shown in table 6.2 and the GA run is shown in figure 6.1.

The second example is a credit scoring data set with 19 variables and 1000

Table 6.2: GA parameters

Generation	50
Population Size	76
Crossover Rate	0.5
Mutation Rate	0.01

observations. The aim of the study is to model or predict the probability that a consumer fails to pay back a loan based on certain characteristics of the individual. Details of this data set can be found in Brooks, Friel and King (2003). The data set is obtained courtesy of Dr. Friel. Brooks et. al. (2003) applied the transdimensional simulated annealing (TDSA) algorithm to this data set to find the best model defined by the information criterion AIC. "The TDSA algorithm was run 20 times from various initial model configurations and each took approximately 6 hours to complete." The algorithm chose a model with variables 1, 2, 3, 5, 6, 7, 8, 9, 13, 18 and 19. The associated AIC value is 1017.828. The complete enumeration of all possible models took approximately 250 hours and it confirms the chosen model is the optimal model according to AIC.

We applied the implicit enumeration algorithm with Aug1 strategy to this data set. It took only 29.9850 seconds and evaluation of 1,038 models out of 524,288 all possible models to find the exact same optimal model. Aug2 strategy took 13.828 seconds and evaluation of 837 models to find the optimal model. GA was also run on this data set. The parameters chosen are shown in table 6.2. One run of GA did not find the optimal model. It

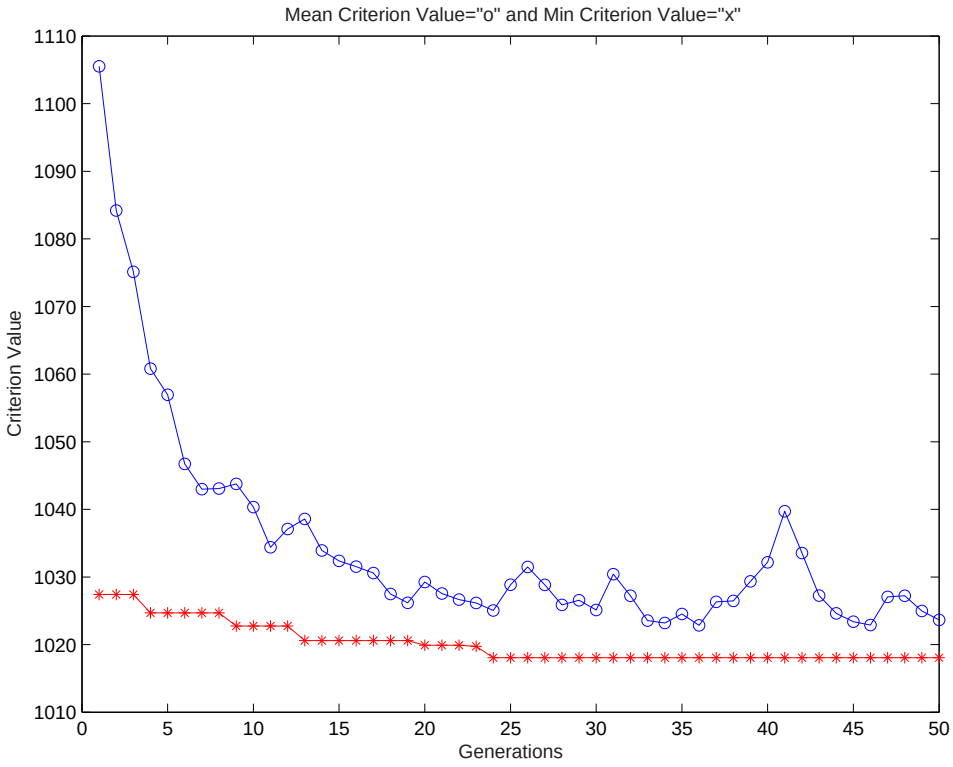

Figure 6.2: One Run of GA for the Credit Data Set

chose the model with variables 1, 2, 3, 5, 6, 7, 8, 11, 13, 18 and 19. The model has an AIC score of 1018.0663. It took GA 37.453 seconds to find this model. This GA run is shown in figure 6.2.

The third example is a cardiac data set. The data set has 27 variables and 558 observations. It can be found on the web site http://www.stat.ucla.edu/data/. The objective of this data set has two folds: one is to see if the stress echocardiography test was still effective in predicting cardiac events when the stress on the heart was produced by a medicine called dobutamine; the other is to

"pinpoint which measurements taken during the stress echocardiography test were most helpful in predicting whether or not a patient suffered a cardiac event over the next year". Details of this data set can be found on the aforementioned web site. The outcome is whether a patient's heart goes wrong. A "1" indicates that the patient did not suffer from any cardiac event and a "0" means that he/she did.

A logistic regression is run to select a subset of variables that are most helpful in predicting any cardiac event. AIC is used as the selection criterion. The 27 variables together with an intercept term implying a model portfolio of

$$2^{28} = 268,435,456$$

possible subset models. The implicit enumeration with Aug1 strategy evaluated 169,605 models, which is 0.0263% of its model space, to find the best model and verify its optimality. The best model consists of variables 4, 8, 9, 10, 11, 13, 16, 18, 20, 21, 22, 23 and 25. The AIC score is 416.6008. The time it took to find the optimal model is 3024 seconds. HIE with a $\tau = 0.005$ was run. HIE took 1662 seconds and evaluation of 92,737 models to find the optimal model.

We also applied the GA to choose a subset model for this data set. The parameters are set as shown in table 6.2. Fifty generations with a population size of 76 implying that the GA evaluated

$$50 \times 76 = 3,800$$

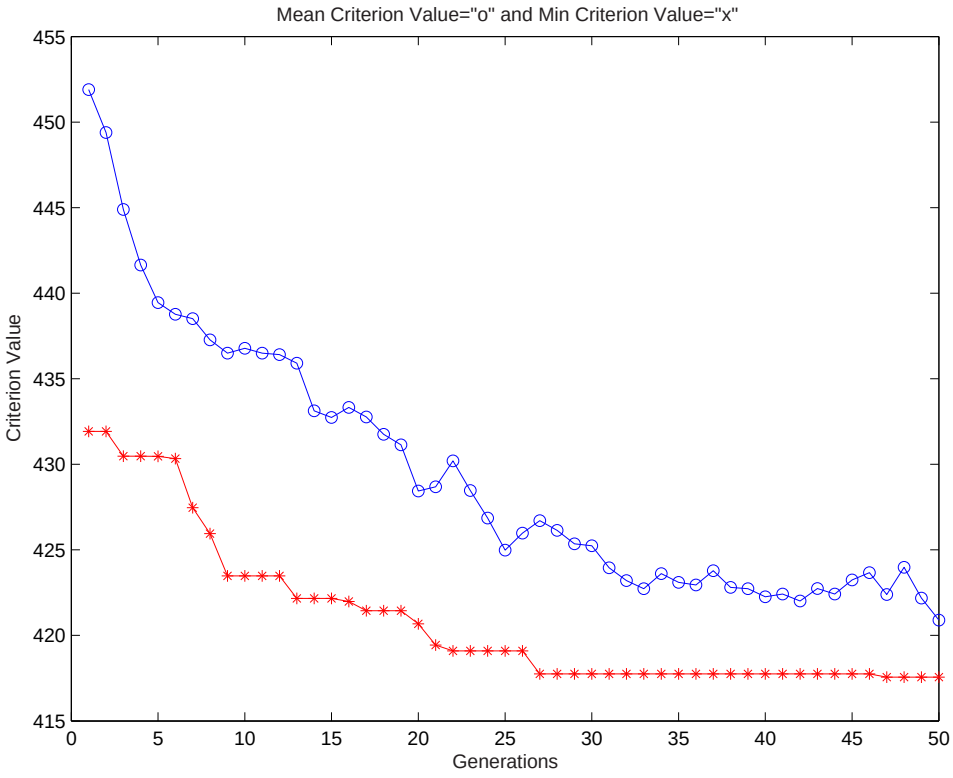

Figure 6.3: One Run of GA for the Cardiac Data Set

models. One run of GA found a model with an AIC score of 417.5572. This model contains variables 4, 8, 9, 13, 16, 18, 20, 21, 22, 23 and 25. The time it took is 29.062 seconds. Figure 6.3 shows this GA run.

We consider using the GAIE-1 and GAIE-2 hybrid methodology using this GA solution as a starting point. GAIE-1 took 56.641 seconds and 1684 model evaluations to find the optimal subset model with an AIC score of 416.6008 by adding variables 10 and 11. GAIE-2 took 3.141 seconds and 263 model evaluations and was not able to improve the solution any more. The

total time it took to run GA, GAIE-1 and GAIE-2 is 87.672 seconds. In this particular run, GAIE was able to find the optimal model with significant less time than the pure IE and HIE. GAIE also improved the performance of the pure GA solution. One hundred runs of the GA and GAIE showed that 26% of the time GA found the optimal subset model, while 54% of the time GAIE found the optimal subset model.

The fourth example is a general demographics portion of the 10th Graphical Visualization and Usability (GVU) center WWW User Survey. The web site is http://www.cc.gatech.edu/gvu/. The original data set consists of 38 predictors and 5022 observations. After deleting the records that contain missing values, the complete data set we used contain 2384 observations. There are two response variables. The first response variable is whether or not a respondent made a purchase of $>$ \$100 and the second is whether or not a respondent ever changed cookie preferences. Detailed description of this data set is listed in table 6.3.

We pick the best logistic regression model for the second response variable. The intercept is treated as another variable. Therefore, the model portfolio contains

$$2^{39} = 549,755,813,888$$

all possible models. The pure implicit enumeration algorithm with Aug1 strategy took 3070.7 seconds and evaluation of 27,939 models to find the optimal model. The optimal model has an AIC score of 2429.879 and it

Table 6.3: GVU WWW User Survey Data Description

Vars	**Description**
$X1$	Years on internet: <0.5, 0.5-1, 1-3, 4-6, ≥ 7
$X2$	Language: 2,Chinese;3,Japanese;4,Russian;5,English;6,French; 7,German;8,Spanish;9,Danish;10,Dutch;11,Italian;12,Greek; 13,Portuguese;14,Hebrew;15,Norwegian;16,Swedish;17,Korean; 18,Other
$X3$	Vision Disability: 0,No;1,Yes
$X4$	Hearing Disability: 0,No;1,Yes
$X5$	Motor Disability: 0,No;1,Yes
$X6$	Cognitive Disability: 0,No;1,Yes
$X7$	Access www from home: Daily, Weekly, Monthly, <One/Month, Never
$X8$	Access www from work: Daily, Weekly, Monthly, <One/Month, Never
$X9$	Access www from school: Daily, Weekly, Monthly, <One/Month, Never
$X10$	Access www from public terminal: Daily, Weekly, Monthly, <One/Month, Never
$X11$	Self pays for access: 0,No;1,Yes
$X12$	Parents pays for access: 0,No;1,Yes

Table 6.3: Continued GVU WWW User Survey Data Description

Vars	Description
$X13$	Work pays for access: 0,No;1,Yes
$X14$	School pays for access: 0,No;1,Yes
$X15$	Registered to vote: 2,No;3,Yes;4,NA
$X16$	Type of industry
$X17$	Type of occupation
$X18$	Sector: 1,Public;2,Private;3,Notforprofit;5,other
$X19$	Profession correspondence with: 1,Public;2,Private;3,Notforprofit;5,other
$X20$	Organization's total budget in $M: <1,1-10,10-100,100-500,500-1000
$X21$	Revenues are from sales to private: 0,No;1,Yes
$X22$	Revenues are from sales to government: 0,No;1,Yes
$X23$	Revenues are from contracts with private: 0,No;1,Yes
$X24$	Revenues are from contracts with government: 0,No;1,Yes
$X25$	Revenues are from contracts from other: 0,No;1,Yes
$X26$	Revenues are from government appropriations: 0,No;1,Yes
$X27$	Revenues are from user fees: 0,No;1,Yes
$X28$	Revenues are from donations: 0,No;1,Yes
$X29$	Educational attainment
$X30$	Gender: 0,Female;1,Male
$X31$	Race: 2,White;3,African American;4,Indigenous;5,Asian; 6,Hispanic; 7,Latino;8,Multiracial;9,Other
$X32$	Marital status: 2,divorced;3,other;4,married;5,separated; 6,single;7,windowed
$X33$	Location: Africa, Antarctica, Asia, Oceania, Europe, USA, Canada, Mexico, Central America, South America, Middle East, West Indies
$X34$	Area: Urban, Suburban, Rural
$X35$	Number of children in household: 0-1, 1-2, 2-3, 3-4, ≥ 4
$X36$	Household income
$X37$	Age category in 5 year increments: <5, 5-10, ..., 81-85, >85
$X38$	Primary computing platform: 1,DOS;2,Mac/Sys;3,Mac;4,OS2; 5,Unix;6,PC Unix;7,Windows;8,NT;9,Win95;10,Win98; 11,VT100;12,WebTV;14,Other

contains variables 1, 2, 7, 8, 9, 10, 14, 21, 23, 26, 30, 34, 37, 38 without the intercept term. We tried the HIE with a $\tau = 0.002$. It took HIE 726.454 seconds and evaluation of 6747 models to find the exact same optimal model.

The pure genetic algorithm was run on this data set. The parameters are chosen as shown in table 6.2. One run of GA took 147.563 seconds and found a model with an AIC score of 2434.0218. This model contains variables 1, 2, 6, 7, 8, 9, 12, 13, 14, 18, 21, 23, 26, 30, 34, 37, 38 without the intercept term. The GA run is shown in figure 6.4. The pure GA does not perform very well. Thus, we tried the hybrid approach GAIE-1 and GAIE-2. This GA solution is used as a starting point for the implicit enumeration algorithm. GAIE-1 improves the solution by adding variable 10 after 356 additional model evaluations. This model has an AIC score of 2433.872. GAIE-2, however, finds a better model according to the information criterion by eliminating variable 27 after evaluations of 629 models. Model found by GAIE-2 has an AIC score of 2430.1198. The total computational time of the GA, and the two hybrid approach is 220.1560 seconds. Although the best model was not found, the hybrid approach does improve the performance of the pure GA and provides a very competitive alternative model. It is worth noting that the performance of the GAIE approach does depend on the GA solution that feeds into the IE algorithm.

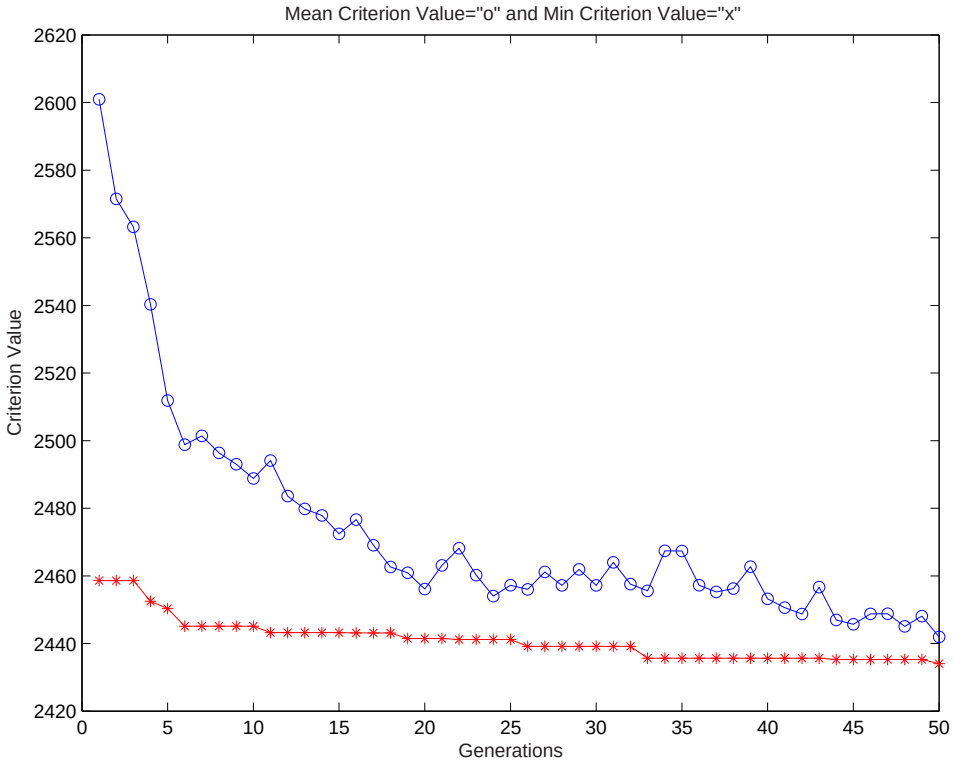

Figure 6.4: One Run of GA for the General Data Set

6.2 Multivariate Regression

In this section, we will demonstrate the performance of the implicit enumeration algorithm with multivariate regression and compare it with that of the GA approach using some large data sets.

6.2.1 Vectorized Multivariate Regression

Bearse and Bozdogan (2000) developed the model selection procedures based on the information-theoretic measure of complexity (ICOMP) with the genetic algorithm. In multivariate regression setting with p response variables and q predictor variables, the exhaustive approach will require evaluation of 2^{pq} models.

Following the notations in their paper, we write the multivariate regression model using the vectorized notation

$$vec(Y) = vec(XB) + vec(E) = (I_p \otimes X)vec(B) + vec(E)$$

where $Y_{(n\times p)} = (y_1, y_2, ..., y_n)'$ is a $n \times p$ dimensional response matrix. $X_{(n\times q)} = (x_1, x_2, ..., x_n)'$ is a corresponding $n \times q$dimensional matrix of predictors and $E_{(n\times p)} = (\epsilon_1, \epsilon_2, ..., \epsilon_n)'$ is $n \times p$ dimensional error matrix. Thus, $vec(Y)$ is $np \times 1$. $I_p \otimes X$, which can be written as X_{sup} is $np \times pq$. $vec(B)$ is $np \times 1$ and $vec(E)$ is $np \times 1$. $vec(E)$ has normal distribution with mean 0 and variance Ω).

For subset MVR models, without loss of generality, we define

$$vec(B) = R\gamma$$

where m is equal to the number of unrestricted elements in $vecB$. Thus, we can write $X^*_{sup} = X_{sup}R$.

The feasible generalized least squares (FGLS) estimator can be then written as

$$\tilde{\gamma} = (X^{*'}_{sup}\hat{\Omega^{-1}}X^*_{sup})X^{*'}_{sup}\hat{\Omega^{-1}}y$$

The FGLS residuals are given by

$$\tilde{e} = y - X^*_{sup}\tilde{\gamma}$$

and a consistent estimator of the covariance matrix of $\tilde{e}$ is given by

$$\tilde{\Sigma} = \frac{1}{n}\tilde{E}'\tilde{E}$$

A consistent estimator of the covariance matrix of $\tilde{\gamma}$ is given by

$$\hat{Cov}(\tilde{\gamma}) = (X^{*'}_{sup}\tilde{\Omega^{-1}}X^*_{sup})^{-1}$$

where

$$\tilde{\Omega} = \tilde{\Sigma} \otimes I_n.$$

The ICOMP criterion is then given by

$$ICOMP = np[1 + log(2\pi)] + nlog|\tilde{\Sigma}| + slog(\frac{tr(\hat{Cov}) + \frac{1}{2}G}{s})$$

$$-log|\hat{Cov}| - plog(2) + \frac{p(p+1)}{2}log(n) - (p+1)log|\tilde{\Sigma}|$$

where

$$G = tr(\tilde{\Sigma}^2) + (tr\tilde{\Sigma})^2 + 2\sum_{j=1}^{p}(\tilde{\sigma}_j^2)^2$$

and $\tilde{\sigma}_j^2$ is the jth diagonal element of $\tilde{\Sigma}$. The detailed derivation can be found in Bearse and Bozdogan (2000).

Here we use the same data set as is used in Bearse and Bozdogan (2000). The data set is on Japanese wine with 30 observations on two response variables ($p = 2$) and nine predictor variables ($q = 9$). So there are

$$2^{18} = 262,144$$

subset MR models. They chose the GA parameters as shown in table 6.2.

One run of GA found a subset model with a ICOMP(IFIM) score of 52.1637. An exhaustive search of the model space shows that the chosen model by GA is indeed the one that minimizes ICOMP.

One hundred runs of GA on the rice data set show that GA chooses the optimal solution 53% of the time. Each run of GA requires evaluations of $50 \times 76 = 3,800$ models and an average of 6 seconds per run.

The implicit enumeration algorithm was run on the same data set. It took 202.25 seconds and evaluated 87,363 models to find the best model by the Aug1 branching strategy. Aug2 branching strategy took 156.67 seconds and evaluation of 54,290 models to find the best model.

6.2.2 Multivariate Regression under Misspecification

In reality, wrong model could often fit to the observed data. There are several situations that misspecification of models can occur as stated in Bozdogan and Magnus (2004b):

1. The functional form of the model is not correctly specified.

2. There are "near-linear" dependencies among the predictor variables which is known as the problem of multicollinearity in regression.

3. There are high skewness and kurtosis in the variables which causes nonnormality of random error or disturbances.

4. There is autocorrelation and heteroscedasticity.

In Bozdogan and Magnus (2004b), a new statistical methodology called misspecification resistant model selection was developed. They showed the performance of the newly developed criterion $ICOMP_{Misspec}$, also termed ICOMPcov is superior to other information criteria such as AIC through some computational experimentations with real and simulated data sets.

In this section, we first give an example to show the superior performance of ICOMPcov information criterion. Then we demonstrate the performance of implicit enumeration with ICOMPcov and compare it with that of the GA.

We use similar Monte Carlo simulation protocol for subset selection of variables as in Bozdogan and Magnus (2004b).

Let

$$x_1 = 10 + \epsilon_1,$$

$$x_2 = 10 + 0.3\epsilon_1 + \alpha\epsilon_2, where \alpha = \sqrt{1 - 0.3^2} = 0.9539,$$

$$x_3 = 10 + 0.3\epsilon_1 + 0.5604\alpha\epsilon_2 + 0.8282\alpha\epsilon_3,$$

where ϵ_2 is independent and identically distributed according to N(0,1).

The response variables are generated from:

$$Y_{(n\times 2)} = [1X_{or}]_{(n\times 4)} B_{(4\times 2)} + E_{(n\times 2)}$$

with $X_{or} = [x_1, x_2, x_3]$ and

$$B = \begin{bmatrix} 8 & -5 \\ 1 & 0.5 \\ 0.5 & 0 \\ 0.3 & 0.3 \end{bmatrix}$$

To make the simulation difficult, we specifically consider the random error matrix E to be a multivariate power exponential distribution which is a generalization of multivariate Gaussian.

We simulated a data set with 1000 observations and 10 variables, in which x_4 through x_{10} are random uniform variables. The true model contains the intercept and variables 1, 2 and 3. It is difficult for any criteria to pick up the "right" number of variables in the multivariate regression model. However, the ICOMP misspecification criterion picked the true model 99% of the time. AIC chose the correct model 32% of the time. The results are shown in table 6.4. One hundred replications of simulations took all subset methods 1375.1 seconds, while took the implicit enumeration algorithm 174.8 seconds, which is only 13% of the time all subset methods took.

Using the same simulation protocol, we added some more uniformly distributed noise variables and simulated a data set with 50 variables and 1000 observations. It took the implicit enumeration algorithm about 246 seconds and evaluation of 12,275 models to find the true model with a ICOMP misspecification criterion score of 4251.2728.

GA was also used on the same data set. The number of generations is 100 and population size is 80. The probability of mutation is chosen to be 0.01 and probability of crossover is 0.7. It took GA 273.609 seconds and evaluation of $100 \times 80 = 8000$ models to find the model containing the intercept and variables 1, 2, 3, 35, 39, 40, 46 and 48 with a ICOMP misspecification score of 4326.7. The GA run is shown in figure 6.5.

Table 6.4: Frequency of Choosing the Best Subset MVR Model in 100 Replications of the Monte Carlo Simulation

Model											IcompCov	AIC
0	1	2	3	4	5	6	7	8	9	10	0	0
1	2	3	4	5	6	7	8	9	10	0	0	0
.....												
0	1	2	3	5	9	10	0	0	0	0	0	1
0	1	2	3	4	7	10	0	0	0	0	0	1
0	1	2	3	5	6	10	0	0	0	0	0	1
.....												
0	1	2	3	6	7	9	0	0	0	0	0	1
0	1	2	3	4	7	9	0	0	0	0	0	1
0	1	2	3	4	5	9	0	0	0	0	0	1
0	1	2	3	6	7	8	0	0	0	0	0	2
0	1	2	3	4	5	6	0	0	0	0	0	1
.....												
0	1	2	3	7	10	0	0	0	0	0	0	1
0	1	2	3	6	10	0	0	0	0	0	0	2
0	1	2	3	5	10	0	0	0	0	0	0	1
0	1	2	3	4	10	0	0	0	0	0	0	1
.....												
0	1	2	3	8	9	0	0	0	0	0	0	3
0	1	2	3	6	9	0	0	0	0	0	0	1
0	1	2	3	5	9	0	0	0	0	0	0	1
.....												
0	1	2	3	7	8	0	0	0	0	0	0	2
0	1	2	3	6	8	0	0	0	0	0	0	1
0	1	2	3	5	8	0	0	0	0	0	0	1
0	1	2	3	4	8	0	0	0	0	0	0	1
.....												
0	1	2	3	5	7	0	0	0	0	0	0	1
.....												
0	1	2	3	5	6	0	0	0	0	0	0	1
0	1	2	3	4	6	0	0	0	0	0	0	1
0	1	2	3	4	5	0	0	0	0	0	0	2
.....												
0	1	2	3	10	0	0	0	0	0	0	0	5
.....												
0	1	2	3	9	0	0	0	0	0	0	0	4
.....												
0	1	2	3	8	0	0	0	0	0	0	1	6
.....												
0	1	2	3	7	0	0	0	0	0	0	0	7
.....												
0	1	2	3	6	0	0	0	0	0	0	0	7
.....												
0	1	2	3	5	0	0	0	0	0	0	0	7
0	1	2	3	4	0	0	0	0	0	0	0	3
.....												
0	1	2	3	0	0	0	0	0	0	0	**99**	**32**
.....												
0	0	0	0	0	0	0	0	0	0	0	0	0

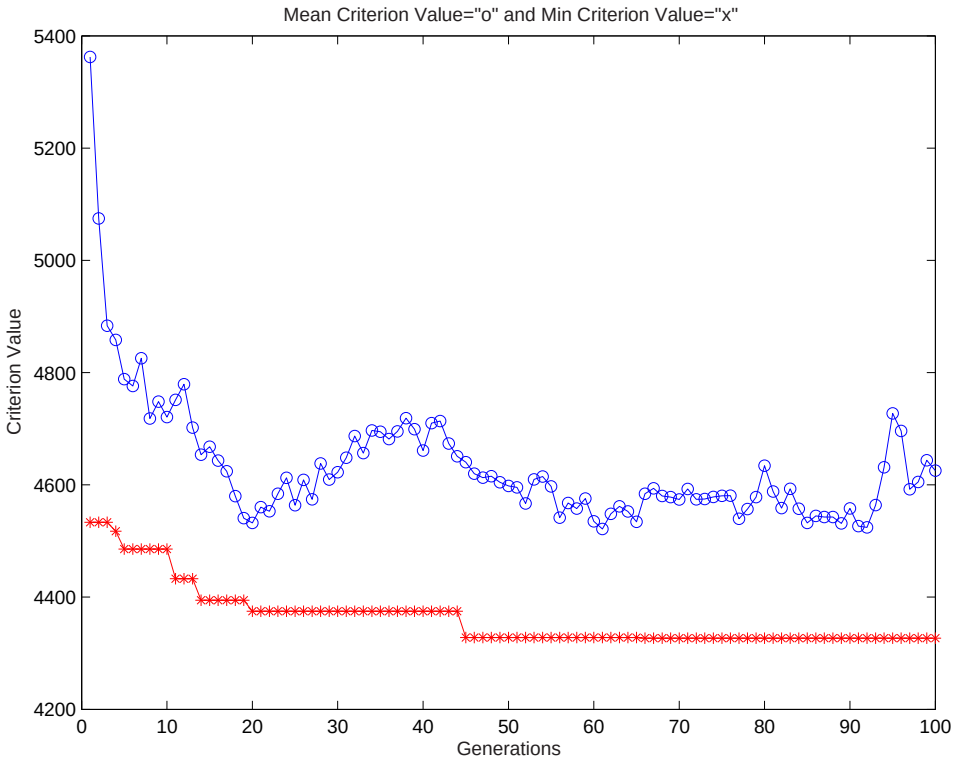

Figure 6.5: One run of GA for the Simulated Data Set with 50 Variables

By using the same simulation procedures, we simulated data sets with 30-80 variables and 1000 observations. Using implicit enumeration, the true models were picked for all simulation runs. The results are displayed in table 6.5.

Table 6.5: Results of the Implicit Enumeration on Simulated Data Sets

# of Var.	Total_Models	Eval	Time
30	1,073,741,824	554	8
40	1,099,511,627,776	4,597	86
50	1,125,899,906,842,620	12,275	246
60	1,152,921,504,606,850,000	142,386	3,766
70	1,180,591,620,717,410,000,000	376,952	11,787
80	1,208,925,819,614,630,000,000,000	803,666	26,648

6.3 Discriminant Analysis

In many applications of discriminant analysis, a large number of variables are available. For example, Mucciardi and Gose (1971) studied a rather extreme case of discriminant analysis based on 157 variables. There are various reasons for not using the entire set of variables. A very important reason is that the misclassification rate may very well increase as the number of variables increases. Huberty (1994, p.117-118,130) has given a good discussion of the reasons for selecting a suitable subset of variables in discriminant analysis, rather than using all of the variables.

In this example, we apply information criteria to define the "best" subset of variables in two-group and multiple discriminant analysis. Particularly, Fujikoshi's (1985a, b) selection method based on Akaike's Information Criterion (AIC) is used. Thus the variable selection problem can be defined as that of finding the subset of variables that minimizes the criterion proposed by Fujikoshi.

In discriminant analysis, we assume that there are two populations $\pi 1$ and $\pi 2$ with J variables. As before let $x = x_1, x_2, ..., x_J$ denote the entire set of variables measured. Let w define any subset of x and let k denote the number of variables in the subset defined by w. Also let $\dot{w} = (1, 1, ..., 1)$ define the solution consisting of all of the variables in x.

Let D denote the Mahalanobis distance between the two samples taken from $\pi 1$ and $\pi 2$ based on w. and let $D(w)$ denote the Mahalanobis distance

based on a subset of variables defined by w. Let N_1 and N_2 denote the number of observations in the samples and let $N = N_1 + N_2$ and $n = N - 2$.

Fujikoshi (1985a, b) has proposed that the optimal subset is the one minimizing the following criterion:

$$A(w) = AIC(w) - AIC(\dot{w})$$

$$= Nlog(1 + \frac{D^2 - D(w)^2}{n(N_1^{-1} + N_2^{-1}) + D(w)^2}) - 2(J - k)$$

For multiple discriminant analysis, suppose that there are populations. Let W denote the sample within groups matrix and let B be the sample between groups matrix. $W(w)$ and $B(w)$ represent the respective sample within groups matrix and sample between groups matrix for subset defined by w. Let I be an identity matrix.

Assume that Σ is unknown. Fujikoshi (1985b) proved that

$$A(w) = Nlog|I + W^{-1}B| - log|I + W(w)^{-1}B(w)| - 2l(J - k)$$

A subset w should be selected if it minimizes $A(w)$. We call this criterion the Fujikoshi criterion.

6.3.1 Mathematical Statement of the Bounding Procedures

The computations of the upper bounds are as follows:

Let $D(w_c)$ denote the mahalanobis distance for the core solution and $M(w_c)$ denote the product $W^{-1}B$ for the core subset w_c.

Let

$$z(w_c) = A(w_c) = Nlog1 + \frac{D^2 - D(w_c)^2}{n(N_1^{-1} + N_2^{-1}) + D(w_c)^2} - 2(J - p)$$

For multiple discriminant analysis:

$$z(w_c) = Nlog\frac{|I + W^{-1}B|}{|I + M(w_c)|} - 2l(J - p)$$

For the two-group case, the augmented core subset, denoted by w_{ac} is the subset obtained by adding to the core subset the free variable that maximizes the mahalanobis distance of the resulting subset. For the multiple group case, the free variable that maximizes the product $W^{-1}B$ is added to the core subset to form the w_{ac}.

Let

$$z(w_{ac}) = A(w_{ac}) = Nlog1 + \frac{D^2 - D(w_{ac})^2}{n(N_1^{-1} + N_2^{-1}) + D(w_{ac})^2} - 2(J - p - 1)$$

For multiple discriminant analysis:

$$z(w_{ac}) = Nlog\frac{|I + W^{-1}B|}{|I + M(w_{ac})|} - 2l(J - p - 1)$$

For the replete solution we have:

$$z(w_r) = A(w_r) = Nlog1 + \frac{D^2 - D(w_r)^2}{n(N_1^{-1} + N_2^{-1}) + D(w_r)^2} - 2(J - p - q)$$

For multiple discriminant analysis:

$$z(w_r) = Nlog\frac{|I + W^{-1}B|}{|I + M(w_r)|} - 2l(J - p - q)$$

The diminished replete subset, is the subset obtained by removing from the replete subset the variable that maximizes the mahalanobis distance of the resulting subset. For the multiple group case, delete the variable that maximizes the product $W^{-1}B$ of the resulting subset.

let

$$z(w_{dr}) = A(w_{dr}) = Nlog1 + \frac{D^2 - D(w_{dr})^2}{n(N_1^{-1} + N_2^{-1}) + D(w_{dr})^2} - 2(J - p - q + 1)$$

For the multiple discriminant analysis:

$$z(w_{dr}) = Nlog\frac{|I + W^{-1}B|}{|I + M(w_{dr})|} - 2l(J - p - q)$$

To obtain a lower bound on those completions having r variables, where $p + 2 \leq r \leq p + q - 2$, we first observe that any such completion will have the quantity $\frac{D^2 - D(w)^2}{n(N_1^{-1} + N_2^{-1}) + D(w)^2}$ at least as large as $\frac{D^2 - D(w_{dr})^2}{n(N_1^{-1} + N_2^{-1}) + D(w_{dr})^2}$ and have at least $p + 2$ variables. Therefore we compute a lower bound for those

completions having r variables, $p+2 \leq r \leq p+q-2$ as:

$$\tilde{z} = A(w_{dr}) = Nlog1 + \frac{D^2 - D(w_{dr})^2}{n(N_1^{-1} + N_2^{-1}) + D(w_{dr})^2} - 2(J-p-2)$$

Similarly, for multiple discriminant analysis:

$$\tilde{z} = Nlog\frac{|I + W^{-1}B|}{|I + M(w_{dr})|} - 2l(J-p-2)$$

Then $\underline{z}(S, w^S) = \bar{z} = min(z(w_c), z(w_{ac}), z(w_r), z(w_{dr}))$ is an upper bound on any completion of (S, w^S) and $\underline{z}(S, w^S) = min(\tilde{z}, \bar{z})$ is a lower bound on any completion of (S, w^S).

6.3.2 Numerical Examples

In this section, we provide two numerical examples for variable selections in two-group and multiple discriminant analysis respectively.

6.3.3 Two-group Discriminant Analysis

Three test problems were run to evaluate the algorithm. They have 6, 13 and 30 variables respectively. For example 1, the data is from Lubischew (1962) and it is analyzed by McKay (1977) and McLachlan (1980) to find which subsets of the 6 variables provide adequate discrimination between the two species Ch. concinna and Ch. heikertingeri. The sample sizes are $N_1 = 21$

and $N_2 = 31$.

The implicit enumeration algorithm picked variables 1, 2, 4 and 6 after evaluating 37 out of 64 possible subsets. Although we are not concerned with a F-test procedure at some specified significance level α, it is worth noting that McLachlan (1980) also suggests that variables 3 and 5 can be eliminated with fairly high degree of confidence and that it will not cause any increase in the overall conditional error rate. One other subset deletion from the original set is questionable. Subset 3, 5, 6 would be retained at $\alpha = 0.1$ and deleted at $\alpha = 0.05$.

The second example studied is taken from Huberty (1994, p. 277). The data set consists of J=13 variables and are used to characterize differences between the two groups of students enrolled in the beginning and intermediate level of college French course. The sample sizes for the two groups are $N_1 = 35$ and $N_2 = 81$.

The implicit enumeration algorithm evaluated 297 of the 8191 possible number of subsets ($2^{13} - 1 = 8191$) and picked 9 out of 13 variables. They are variables 2, 3, 4, 5, 6, 9, 11, 12 and 13. The Fujikoshi criterion is -5.9954. This result differs from the "best" subset of size 9 picked by the three hit rate procedures listed in Huberty (1994, p.119, 121, and 123). The results from the three hit rate procedures also differ from each other. The Fujikoshi criteria for the three results are -2.4810, 1.4720 and -2.1484 respectively.

The third example data set used is a mammogram data set from the Wisconsin Diagnostic Breast Center (Wolbert et al. 1994). This data set

consists of 569 samples with 357 benign and 212 malignant samples. The benign samples are denoted as 0 and malignant samples are denoted as 1. Thirty variables were extracted from a digitalized image of a breast lump. All features have been standardized. Implicit Enumeration algorithm chose variables 1, 6, 7, 8, 11, 15, 17, 18, 21, 22, 24, 27, 29, 30. The Fujikoshi criterion value for the chosen model is -27.1738. It took the algorithm 600.8130 seconds and evaluations of 203,149 models out of the possible 1,073,741,823 models. The heuristic version of the IE is also applied to the data set. The value of τ is chosen to be 0.1. The HIE also found the optimal model after 336.156 seconds and evaluation of 110,909 models, which is about half of the time and evaluations of the non-heuristic version.

6.3.4 Multiple Discriminant Analysis

We again use the data from Lubischew (1962) to demonstrate the procedure. Six characteristics about the three species, Ch. concinna, Ch. heikertingeri and Ch. heptapotamica are investigated. The sample sizes are $N_1 = 21$, $N_2 = 31$ and $N_3 = 22$.

The implicit enumeration algorithm evaluated 31 out of 63 possible subsets and decided that all 6 variables are needed to discriminate between the three species. McKay (1977) noted that "at simultaneous level 0.235, the six characteristics discriminate significantly among all three species and between each pair of species".

For the second example, we use the data from Huberty (1994, p. 277).

The two groups in section 18.1 and the third group of students, $N_3 = 37$, enrolled in the advanced level of French course are studied. Only variable 12 is picked to discriminate between the three groups of students after evaluating 27 out of 8191 subsets. Variable 12 is ETS Grammar French test score. The result is the same as the best subset of size 1 picked by BMDP 7M (stepwise discriminant analysis) and the Smith Forward Selection Strategy in Huberty (1994, P. 121, Table VIII.2).

Chapter 7

Summary and Conclusion

7.1 Summary

The main goal of this dissertation is to present a new algorithm that optimally solves the subset model selection problem based on any information criterion. We first examine the current approaches and point out their limitations. Then, we introduced the implicit enumeration algorithm that overcomes these limitations. The implicit enumeration algorithm explores the structure of the information criteria so that it is applicable to a wide variety of information criteria. A heuristic version is introduced so that fewer model evaluations might be needed to find the optimal model. The heuristic version does not guarantee to find the best model for a given information criterion. However, it does provide some very competitive alternatives.

The new algorithm can be combined with the heuristic algorithm to make

it more flexible and capable to deal with some real large problems.

The proposed methodology was applied to real and simulated data sets. Our computational experiences show that the implicit enumeration algorithm outperforms the popular branch and bound techniques both in terms of the number of models evaluations and the computational time.

From our computational tests, we suggest that when the number of variables are greater than 80. We should consider using the heuristic version of the implicit enumeration algorithm. The absolute value of the multiplier in the HIE increases as the number of variables increases. A good range of value is between -0.1 and 0.1.

Finally, we demonstrate the wide application of the new algorithm by applying these approaches to three different areas: the logistic regression, multivariate regression and discriminant analysis. The performance of the implicit enumeration algorithm on some large data sets is shown through the applications.

7.2 Conclusion and Future Studies

In conclusion, we proposed a novel algorithm that is very efficient and flexible. It optimally solves the model selection problem based on any information criterion for the first time.

This algorithm has many potential applications. Besides finding the "best" model in multiple, multivariate, logistic regression and discriminant

analysis problems. It can also be used in other important modeling applications. An example is the Autoregressive Integrated Moving Average (ARIMA) order selection in time series analysis. Gaeton (2000) applied the genetic algorithm to ARIMA model identification. It will be interesting to see the performance of the implicit enumeration algorithm as compared to the genetic algorithm in ARIMA subset model selection.

The implicit enumeration algorithm can generate all of the models having information criteria values close to the minimum in multi-model inference applications and compare different information criteria in Monte Carlo studies.

In multi-model inference, the goal is to use all of the models that have information criterion values close to the minimum in order to estimate the uncertainty in model selection (Hoeting 1996, Brieman 2001, Burnham and Anderson 2002). The algorithm presented in this paper provides a means to identify all subset models that are within a specified range of the model having the minimum information criterion value. To generate these models the implicit enumeration algorithm can be rerun while fathoming only those solutions for which $\underline{z}(S, w^S) > z^* + c$. All completions evaluated having $z(w) \leq z^* + c$ are retained on a list of incumbent solutions.

Comparative studies of information criteria have used Monte Carlo generated data. Different information criteria have been compared based on their ability to choose the known true model that was used to generate the data. Until now these studies were based on very simple generating models, since it was necessary to generate all possible solutions to identify the one that a

given information criterion would select. With our method, larger and more realistic generating models can be used in future comparisons.

The implicit enumeration algorithm developed in this dissertation has been applied with information criteria having the two part structure in subset selection. Another direction of future research is that the algorithm can be adapted to some other unconstrained linear or nonlinear programming problems with the objective function having the two part structure.

Bibliography

Bibliography

[1] Akaike, H. (1969). *Fitting Autoregressive Models for Prediction.* Ann. Inst. Stat. Mat., 21, 243-247.

[2] Akaike, H. (1973). *Information Theory and an Extension of the Maximum Likelihood Principle.* in: B.N. Petrov and F.Csaki, eds., 2nd International Symposium on Information Theory. Akademia Kiado, Budapest, 267-281.

[3] Akaike, H. (1978). *A Bayesian Analysis of the Minimum AIC Procedure.* Annals of the Institute of Mathematical Statistics A(30), 9-14.

[4] An, H. and L. Gu. (1989). *Fast Stepwise Procedures of Selection of Variables by Using AIC and BIC Criteria.* Acta Math. Appl. Sinica (English Ser.), 5, 60-67.

[5] Baker, B and M. Ayechew. (2003). *A Genetic Algorithm for the Vehicle Routing Problem.* Computers and Operations Research, Volume 30, Issue 5, 787-800.

[6] Bao, X. and H. Bozdogan. (2004). *Subsetting Kernel Regression Models Using Genetic Algorithm and the Information Measure of Complexity.* Classification, Clustering and Data Mining Applications. Proceedings of the 9th International Federation of Classification Society Conference, 185-196.

[7] Bauer, R. J. (1994). *Genetic Algorithms and Investment Stratiges*. New York: John Wiley.

[8] Bearse, P. and H. Bozdogan. (1998). *Subset selection in vector autoregressive models using the genetic algorithm with informational complexity as the fitness function*. Systems Analysis Modelling Simulation (SAMS), 31, 61-91.

[9] Bozdogan, H. (1987a). *Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions*. Psychometrika, 52, 345-370.

[10] Bozdogan, H. (1987b). *ICOMP: A new model selection criterion*. Preprint paper in Hans H. Bock Ed., Classification and related methods of data analysis. Amsterdam: Elsevier Science(North-Holland).

[11] Bozdogan, H. (1988a). *ICOMP: A New Model-Selection Criterion*. Classification and Related Methods of Data Analysis, Hans H. Bock (ed.), Amsterdam, Elsevier Science Publishers B. V. (North_Holland), 599-608.

[12] Bozdogan, H. (1988b). *The theory and applications of information-theoretic measure of complexity (ICOMP) as a new model selection criterion*. Unpublished rearch report, the Institute of Statistical Mathematics, tokyo, Japan, and the Department of Mathematics, university of Virgina, Charlottesville, VA, March 1988.

[13] Bozdogan, H. (1990). *On the Information-Based Measure of Covariance Complexity and Its Application to the Evaluation of Multivariate Linear Models*. Communications in Statistics Theory and Methods, 19(1), 221-278.

[14] Bozdogan, H. (1994). *Mixture Model Cluster Analysis Using Model Selection Criteria and A New Informational Measure of Complexity*. Proceedings of the US/Japan Conference on the Frontiers of Statistical Modeling: An Informational Approach, 69-113.

[15] Bozdogan, H. and D. Haughton,(1998). *Informational Complexity Criteria for Regression Models*. Computational Statistics & Data Analysis, 28, 51-76.

[16] Bozdogan, H. (2000). *Akaike's Information Criterion and Recent Developments in Informational Complexity*. Journal of Mathematical Psychology, 44, 62-91.

[17] Bozdogan, H. and P. M. Bearse. (2003). *Information Complexity Criteria for Detecting Influential Observations in Dynamic Multivariate Linear Models Using the Genetic Algorithm*. Journal of Statistical Planning and Inference, 114, 31-44.

[18] Bozdogan, H. (2004a). *Statistical Modeling and Model Evaluation: A New Informational Approach*. Chapman and Hall/CRC.

[19] Bozdogan, H. and Jan Magnus (2004b). *Missecification Resistent Model Selection Using Information Complexity*. Submitted for publication.

[20] Brieman, L. (2001). *Statistical Modeling: the Two Cultures (with discussion)*. Statistical Science, 16, 199-231.

[21] Brooks, S. P., N. Friel and R. King. (2003). *Classical Model Selection via Simulated Annealing*. Journal of Royal Statistical Society, Series B, 65, 503-520.

[22] Burnham, K. P. and D. R. Anderson, (2002). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer-Verlag, New York.

[23] Chatpattananan, V. and H. Bozdogan. (2004). *Regularized Regression Trees and RBF Networks, Subset Selection of Best Predictors with Kernel Regression Models Using Genetic Algorithm and the Information Measure of Complexity*. Working Paper.

[24] Chatterjee, S. and A.S. Hadi. (1988). *Sensitivity Analysis in Linear Regression*. John Wiley & Sons: New York.

[25] Chatterjee, S and B. Price. (1991). *Regression Analysis by Example*. New York: John Wiley & Sons, Inc.

[26] Chen, X. (2003). *An Improved Branch and Bound Algorithm for Feature Selection*. Pattern Recognition Letters, 24, 1925-1933

[27] Draper, N. R. and H. Smith. (1998). *Applied Regression Analysis*. New York: Wiley.

[28] Dorsey, R. E. and W. J. Mayer. (1995). *Genetic Algorithms for Estimation Problems with Multiple Optima, Nondifferentiability, and Other Irregular Features*. Journal of Business and Economic Statistics, 13(1), 53-66.

[29] Efroymson, M. A. (1960). *Multiple Regression Analysis, in Mathematical Methods for Digital Computers* (eds A. Ralston and H.S. Wilf). New York: Wiley.

[30] Fujikoshi, Y. (1985a). *Selection of Variables in Two-group Discriminant Analysis by Error Rate and Akaike's Information Criteria*. Journal of Multivariate Analysis, 17, 27-37.

[31] Fujikoshi, Y. (1985b). *Selection of Variables in Discriminant Analysis and Canonical Correlation Analysis*. Multivariate Analysis -VI, ed, P.R. Krishnaiah, 219-236.

[32] Furnival, G. M. and R. W. Wilson. (1974). *Regression by Leaps and Bounds*. Technometrics, 16, 499-511.

[33] Gaetan, C. (2000). *Subset ARMA Model Identification Using Genetic Algorithms*. Journal of Time Series Analysis, Vol. 21. No. 5, 559-570.

[34] Gatu, C and E. Kontoghiorghies. (2003). *Branch-and-Bound Algorithms*

for Computing the Best Subset Regression Models. Submitted for publication.

[35] Goldberg, D. E. (1989). *Genetic Algorithm in Search, Optimization, and Machine Learning*. New York: Addison-Wesley.

[36] Hald, A. (1952). *Statistical Theory with Engineering Applications*. New York: Wiley.

[37] Hamamoto, Y. Uchimura, S. Matsuura, Y., Kanaoka, T. and Tomita, S. (1990). *Evaluation of the Branch and Bound Algorithm for Feature Selection*. Pattern Recognition Letters, 11, 453-456.

[38] Hannan, E.J. and B. G. Quinn. (1979). *The Determination of the Order of an Autoregress*. Journal of the Royal Statistical Society, B 41, 190-195.

[39] Hocking, R. R. (1976). *The Analysis and Selection of Variables in Linear Regression*. Biometrics, 32, 1-49.

[40] Hoeting, J. A. and J. G. Ibrahim. (1996). *Bayesian Predictive Simultaneous Variable and Transformation Selection in the Linear Model*. Department of Statistics, Colorado State University, Fort Collins. Technical Report No. 96/39.

[41] Hosmer, D.W. and S. Lemeshow. (2000). *Applied Logistic Regression*. New York: Wiley.

[42] Huberty, C. J. (1994). *Applied Discriminant Analysis*. New York: Wiley.

[43] Kullback, S. And R. Leibler (1951). *On information and Sufficiency.* Ann. Math. Statist., 22, 79-86.

[44] Lin, C-T. and Lee, C.S.G. (1996). *Neural Fuzzy Systems.* Upper Saddle River: Prentice Hall.

[45] Linhart, H. and W. Zucchini. (1986). *Model Selection.* John Wiley & Sons, New York.

[46] Luh, H. K., J. J. Minesky and H. Bozdogan. (1997). *Choosing the Best Predictors in Regression Analysis Via the Genetic Algorithm with Informational Complexity as the Fitness Function.* unpublished paper.

[47] Lubischew, A. A. (1962). *On the Use of Discriminant Functions in Taxonomy.* Biometrics, 18, 455-477.

[48] Mahfoud, S. (1994). *Population Sizeing for Sharing Methods.* Illinois Genetic Algorithms Laboratory Report No. 94005.

[49] Mallows, C. L. (1973). *Some Comments on Cp.* Technometrics, 15(4), 661-675.

[50] Mantel, N. (1970). *Why Stepdown Procedures in Variable Selection.* Technometrics, 12, 591-612.

[51] McClave, J. (1975). *Subset Autoregression.* Technometrics, 17, 213-220.

[52] McKay, R. J. (1977). *Simultaneous Procedures for Variable Selection in Multiple Discriminant Analysis.* Biometrika, 64, 283-290.

[53] McLachlan G. J. (1980). *On the Relationship between the F Test and the Overall Error Rate for Variable Selection in Two-Group Discriminant Analysis*. Biometrics, 36, 501-510.

[54] McQuarrie, A. and C. L. Tsai (1998). *Regression and Time Series Model Selection*. World Scientific.

[55] Miller, A. J. (2002). *Subset Selection in Regression*. Chapman and Hall.

[56] Montgomery, D. C. and E. A. Peck. (1992). *Introduction to Linear Regression Analysis*. New York: Wiley.

[57] Mucciardi, A. N. and E. E. Gose. (1971). *A Comparison of Seven Techniques for Choosing Subsets of Pattern Recognition Properties*. IEEE Transactions on Computers, C-20, 1023-1031.

[58] Myung, I. J. (2000). *The Importance of Complexity in Model Selection*. Journal of Mathematical Psychology, 44(1), 190-204.

[59] Myung, I. J., M. A. Pitt, and W. Kim. (in press). *Model evaluation, testing and selection*. In K. Lambert and R. Goldstone (eds.) The Handbook of Cognition. Sage Publication.

[60] Narendra P. M. and K. Fukunaga (1977). *A Branch and Bound Algorithm for Feature Subset Selection*. IEEE Transactions on Computers, 26, 917-922.

[61] Navarro, D. J. and I. J. Myung. (in press). *Model evaluation and selection.* In B, Everitt and D. Howel (eds.), Encyclopedia of Behavioral Statistics, New York: Wiley.

[62] Potscher, B. M. (1991). *Effects of model selection on inference.* Econometric Theory, 7, 163-185.

[63] Raftery, A. E. (1995). *Bayesian model selection in social research (with Discussion).* In Sociological Methodology 1995, (Peter V. Marsden, ed.), Cambridge, Mass.: Blackwells, 111-196.

[64] Schwarz, G. (1978). *Estimating the Dimension of a Model.* Annals of Statistics, 6, 461-464.

[65] Shibata, R. (1989). *Statistical Aspects of Model Selection.* From Data to Model, ed. J.C. Williems, New York: Springer-Verlag, 215-240.

[66] Siedlecki, W. and J. Sklansky. (1988). *On Automatic Feature Selection.* International Journal of Pattern Recognition and Artificial Intelligence, Vol. 2, No. 2. 197-320.

[67] Somol, P., P. Pudil, F. Ferri and J. Kittler. (2000). *Fast Branch and Bound Algorithm for Feature Selection.* Invited paper for the 4th World Multiconference on Systemics, Cybernetics and Informatics, Proceedings, Orlando, Florida, 646-651.

[68] Trotter JR. L. E. and E. M. Shetty. (1974). *An Algorithm for the*

Bounded Variable Integer Programming Problem. Journal of Association for Computing Machinery, 21, 505-513.

[69] Wetherill, G. Barrie (1986). *Regression Analysis with Applications*. New York: Chapman and Hall.

[70] Winker, Peter. (2001). *Optimization Heuristics in Econometrics-Applications of Threshold Accepting*. John Wiley & Sons: Chichester.

[71] Wolberg, W., W. Street and O. Mangasarian. (1994). *Machine learning techniques to diagnose breast cancer from fine needle aspirates*. Cancer Letter, 77, 163-171.

[72] Yu, B. and B. Yuan. (1997). *A More Efficient Branch and Algorithm for Feature Selection*. Pattern Recognition, Vol. 36, No. 6, 883-889.

Appendix

Appendix A: Matlab Code for Implicit Enumeration

Note: All codes are programmed by Xinli Bao and Vuttichai Chatpattananan.

1. Main Function

```
function main
% Model selection using Implicit Enumeration Algorithm
% Required functions: FindBranch.m, FindBound.m, FindSSE.m

% Input:
%   1) Predictor variables (X in data matrix) file name
%   2) Response variable (Y in data vector) file name
%   3) User-keyed in output file name
% Output:
%   1) On screen output
%   2) the output file named by the user


% Input Data -- File must be in the same dir of main.m
xfilename = input('Enter predictor vars file name (X): ','s');

x = load(xfilename);
```

```
yfilename = input('Enter response var file name (Y): ','s');

y = load(yfilename);

outfname = input('Enter the output file name (Y): ','s');
[row,col]=size(x);
nobs = length(y);

x = [ones(nobs,1) x];

% Declare Output Variables

output_node = struct('zbndnode',{},'Snode',{},'Wsnode',{});

minAIC = struct('node_opt',{},'zopt',{},'Sopt',{});

Iteration = [];

% Initialization
Zupper = inf;
Zlower = -inf;
zopt = inf;
```

```
S = [1]; % Intercept
Ws= [1]; % Add Intercept
[zvector,SAdd,S_min,Model] = FindBound(S,Ws,x,y);

zupper = zvector(6);
zlower = zvector(7);
node = 1;

iteration = [node];

if zupper < zopt
    zopt = zupper;
    minAIC(1).node_opt = node;
    minAIC(1).zopt = zopt;
    minAIC(1).Sopt = S_min;
end
output_node(node).Snode = S;
output_node(node).Wsnode = Ws;
output_node(node).zbndnode = [zupper zlower];

if zopt > zlower
    [output_node,node,zopt,SAdd_left,SAdd_right,Ws_left,...
    Ws_right, iteration,minAIC] ...
```

```
        = FindBranch(node,S,Ws,x,y,SAdd,...
        zopt,iteration,output_node,minAIC);
end

% Output
disp(' ');
disp('Output');
ModelNo=size(Model,1);

for i = 1:node
   disp(['Node     ' num2str(iteration(i))]);
   disp(['S        ' num2str(output_node(i).Snode)]);
   disp(['Ws       ' num2str(output_node(i).Wsnode)]);
   disp(['Bounds   ' num2str(output_node(i).zbndnode)]);
   disp(' ');
end

disp(['Possible Number of Models:  ' num2str(2^col)]);
disp(['Number of Models evaluated: ' num2str(ModelNo)]);

disp('The Best Subset ');

disp(['Found at Iteration:' num2str(minAIC(1).node_opt)]);
```

```
disp(['The minimum AIC:'num2str(minAIC(1).zopt)]);

disp(['Variables: ' num2str(minAIC(1).Sopt)]);

disp(['Total Iterations:'num2str(node)]);



for i = 1:node
    z_mat(i,:) = output_node(i).zbndnode;
end
plot([1:node],z_mat(:,1),'b-o',[1:node],z_mat(:,2),'r-*')
xlabel('Iteration');
ylabel('Bounds');

title('Augment Algorithm:Upper Bound="o" and Lower Bound="x"');
```

2. Augmentation Branching Function

```
 function
[output_node,node,zopt,SAdd_left,SAdd_right,Ws_left,
   Ws_right,iteration,minAIC]...
   =FindBranch(node,S,Ws,x,y,SAdd,zopt,iteration,...
```

```
    output_node,minAIC)

% Variable Added
node = node+1;
iteration = [iteration node];
S_left = [S SAdd];
output_node(iteration(node)).Snode = S_left;
Ws_left = [Ws 1];
output_node(iteration(node)).Wsnode = Ws_left;

[zvector, SAdd_left,S_min] = FindBound(S_left,Ws_left,x,y);

zupper = zvector(6);

zlower = zvector(7);

if zupper < zopt
    zopt = zupper;
    minAIC(1).node_opt = node;
    minAIC(1).zopt = zopt;
    minAIC(1).Sopt = S_min;
end
output_node(iteration(node)).zbndnode = [zupper zlower];
```

```
if zopt > zlower
   [output_node,node,zopt,SAdd_left,SAdd_right,Ws_left,
   Ws_right,iteration,minAIC] ...
       = FindBranch(node,S_left,Ws_left,x,y,SAdd_left,...
       zopt,iteration,output_node,minAIC);
end

% Variable Not Added
node = node+1;
iteration = [iteration node];
S_right = [S SAdd];

output_node(iteration(node)).Snode = S_right;

Ws_right = [Ws 0];

output_node(iteration(node)).Wsnode = Ws_right;

[zvector, SAdd_right,S_min] = FindBound(S_right,Ws_right,x,y);

zupper = zvector(6);
zlower = zvector(7);
```

```
if zupper < zopt
    zopt = zupper;
    minAIC(1).node_opt = node;
    minAIC(1).zopt = zopt;
    minAIC(1).Sopt = S_min;
end

output_node(iteration(node)).zbndnode = [zupper zlower];

if zopt > zlower
  [output_node,node,zopt,SAdd_left,SAdd_right,Ws_left,...
  Ws_right,iteration,minAIC] ...
      = FindBranch(node,S_right,Ws_right,x,y,SAdd_right,...
      zopt,iteration,output_node,minAIC);
end
```

3. Bound function

```
function [zvector,SAdd,S_min,Model] = FindBound(S,Ws,x,y);

% Find five criterion values with Implicit Enumeration
% Required functions: FindSSE.m
```

```
[nobs, npars] = size(x);
fullmodel = [1:npars];
Model='1';

% Find z1
core = S(find(Ws~=0));
SSEz1 = FindSSE(x(:,core),y);

z1 =nobs*log(SSEz1/nobs)+2*length(core);
zvector(1) = z1;

% Find z2
core_plus_one = fullmodel;
core_plus_one(S) = [];

for i = 1:length(core_plus_one)
   Model=strvcat(Model,num2str([core core_plus_one(i)]));
   SSEcore(i) = FindSSE(x(:,[core core_plus_one(i)]),y);
end
SSEz2 = min(SSEcore);

z2 = nobs*log(SSEz2/nobs)+2*(length(core)+1);
```

```
zvector(2) = z2;

SAdd = core_plus_one(find(SSEcore==SSEz2));

% Find z3
replete = fullmodel;
Model=strvcat(Model,num2str(replete));
replete(S(find(Ws==0))) = [];
SSEz3 = FindSSE(x(:,replete),y);

z3 = nobs*log(SSEz3/nobs)+2*length(replete);
zvector(3) = z3;

% Find z4
for i = 1:length(core_plus_one)
   replete_minus_one = replete;
   replete_minus_one(find(replete_minus_one==core_plus_one(i)))=[];
   Model=strvcat(Model,num2str(replete_minus_one));
   SSEreplete(i) = FindSSE(x(:,replete_minus_one),y);
end
SSEz4 = min(SSEreplete);

z4 = nobs*log(SSEz4/nobs)+2*(length(replete)-1);
```

```
zvector(4) = z4;
SDelete = core_plus_one(find(SSEreplete==SSEz4));

% Find z5
z5 = z4-2*(length(replete)-1)+2*(length(core)+2);
zvector(5) = z5;

% Find Upperbound
zub = min(zvector(1:4));
zvector(6) = zub;

% Find Minimum Upperbound from z1 to z4
zub_id = find(zub==zvector(1:4));
```

4. Function to calculate SSE

```
function SSE = FindSSE(x,y)
% Calculate SSE

b = inv(x'*x)*(x'*y);
yhat = x*b;
res = y-yhat;
SSE = res'*res;
```

5. Variable Deletion Branching Strategy

```
function
[output_node,node,zopt,SDelete_left,SDelete_right,Ws_left,...
     Ws_right,iteration,minAIC,binary_model_eval] ...
   = FindBranchbw(node,S,Ws,x,y,SDelete,zopt,iteration,...
   output_node,minAIC,binary_model_eval)

% Variable Deleted
node = node+1;
iteration = [iteration node];
S_left = [S SDelete];
output_node(iteration(node)).Snode = S_left;
Ws_left = [Ws 0];

output_node(iteration(node)).Wsnode = Ws_left;
[zvector,SAdd,SDelete_left,S_min,binary_model_eval] = ...
   FindBound(S_left,Ws_left,x,y,binary_model_eval);
zupper = zvector(6);
zlower = zvector(7);

if zupper < zopt
    zopt = zupper;
    minAIC(1).node_opt = node;
```

```
    minAIC(1).zopt = zopt;
    minAIC(1).Sopt = S_min;
end
output_node(iteration(node)).zbndnode = [zupper zlower];
output_node(iteration(node)).num_model_eval =
length(binary_model_eval(:,1))-1;
if zopt > zlower
   [output_node,node,zopt,SDelete_left,SDelete_right,Ws_left,...
   Ws_right,iteration,minAIC,binary_model_eval] ...
   = FindBranchbw(node,S_left,Ws_left,x,y,SDelete_left,
   zopt,iteration,output_node,minAIC,binary_model_eval);
end

% Variable Not Deleted
node = node+1;
iteration = [iteration node];

S_right = [S SDelete];

output_node(iteration(node)).Snode = S_right;

Ws_right = [Ws 1];
output_node(iteration(node)).Wsnode = Ws_right;
```

```
[zvector,SAdd,SDelete_right,S_min,binary_model_eval] = ...
   FindBound(S_right,Ws_right,x,y,binary_model_eval);
zupper = zvector(6);
zlower = zvector(7);

if zupper < zopt
    zopt = zupper;
    minAIC(1).node_opt = node;
    minAIC(1).zopt = zopt;
    minAIC(1).Sopt = S_min;
end
output_node(iteration(node)).zbndnode = [zupper zlower];
output_node(iteration(node)).num_model_eval =
length(binary_model_eval(:,1))-1;

if zopt > zlower
   [output_node,node,zopt,SDelete_left,SDelete_right,Ws_left,
   Ws_right,iteration,minAIC,binary_model_eval] ...
   = FindBranchbw(node,S_right,Ws_right,x,y,SDelete_right,...
   zopt,iteration,output_node,minAIC,binary_model_eval);
end
```

6. Greedy Branching Strategy

```
function
[output_node,node,zopt,SAdd_left,SAdd_right,Ws_left,Ws_right,...
      iteration,minAIC,binary_model_eval] ...
   = FindBranchgd(node,S,Ws,x,y,SAdd,SDelete,zopt,iteration,
   zvector,output_node,minAIC,binary_model_eval)

% Variable Added
node = node+1;
iteration = [iteration node];

if zvector(2) <=zvector(4)
    S_left = [S SAdd];
    Ws_left = [Ws 1];
else
    S_left = [S SDelete];
    Ws_left = [Ws 0];
end

output_node(iteration(node)).Snode = S_left;
output_node(iteration(node)).Wsnode = Ws_left;

[zvector,SAdd_left,SDelete_left,S_min,binary_model_eval] = ...
```

```
   FindBound(S_left,Ws_left,x,y,binary_model_eval);

zupper = zvector(6);
zlower = zvector(7);

if zupper < zopt
    zopt = zupper;
    minAIC(1).node_opt = node;
    minAIC(1).zopt = zopt;
    minAIC(1).Sopt = S_min;
end
output_node(iteration(node)).zbndnode = [zupper zlower];
output_node(iteration(node)).num_model_eval =
length(binary_model_eval(:,1))-1;
if zopt > zlower
   [output_node,node,zopt,SAdd_left,SAdd_right,Ws_left,
   Ws_right,iteration,minAIC,binary_model_eval] ...
      =FindBranchgd(node,S_left,Ws_left,x,y,SAdd_left,
      SDelete_left,zopt,iteration,zvector,output_node,...
      minAIC,binary_model_eval);
end

% Variable Not Added
```

```
node = node+1;
iteration = [iteration node];

if zvector(2) <= zvector(4)
    S_right = [S SAdd];
    Ws_right = [Ws 0];
else
    S_right = [S SDelete];
    Ws_right = [Ws 1];
end

output_node(iteration(node)).Snode = S_right;
output_node(iteration(node)).Wsnode = Ws_right;
[zvector,SAdd_right,SDelete_right,S_min,binary_model_eval] = ...
   FindBound(S_right,Ws_right,x,y,binary_model_eval);
zupper = zvector(6);
zlower = zvector(7);

if zupper < zopt
    zopt = zupper;
    minAIC(1).node_opt = node;
    minAIC(1).zopt = zopt;
```

```
    minAIC(1).Sopt = S_min;
end
output_node(iteration(node)).zbndnode = [zupper zlower];
output_node(iteration(node)).num_model_eval =
length(binary_model_eval(:,1))-1;

if zopt > zlower
   [output_node,node,zopt,SAdd_left,SAdd_right,Ws_left,
         Ws_right,iteration,minAIC,binary_model_eval]...
   =FindBranchgd(node,S_right,Ws_right,x,y,SAdd_right,...
    SDelete_right,zopt,iteration,zvector,output_node,...
    minAIC,binary_model_eval);
end
```

7. Simulation Function

```
function sim
% Function to generate the simulated data set with user defined
% number of predictor variables, observations and the number of
% predictors that form the response

maxN=200; %Maximum number of generate the prime number
randp = primes(maxN);
```

```
nvars=30;  %Number of predictor variables
nobs=1000; %Number of observations
nvars_in=8; % Number of variables to generate the response

% Generate independent variables matrix
for i = 1:nobs
    for j = 1:nvars
        x(i,j) = log(randn(1)*randp(j));
    end
end

% Generate response variable: y = x1+x2+...+x(nvars_in)
for i = 1:nobs
   y(i) = sum(x(i,1:nvars_in));
end

% Output files

% Output predictor variables file (X)
fid = fopen('simx30.m','w');
for i = 1:nobs
   for j = 1:nvars
      fprintf(fid,'%f\t',x(i,j));
```

```
   end
   fprintf(fid,'\n');
end
fclose(fid);

% Output response variable file (Y)
fid = fopen('simy30.m','w');
for i = 1:nobs
   fprintf(fid,'%f\n',y(i));
end
fclose(fid);
```

Vita

Xinli Bao is a graduate teaching assistant for the College of Business Administration at the University of Tennessee, Knoxville. She completed her undergraduate studies in International Business Management from the Harbin Engineering University in Harbin, China. After one year of graduate study in Industrial Economics in Harbin, she joined the Management Science program in the University of Tennessee, Knoxville. She completed her Master's degree in Management Science in May 2001, a second Master's degree in statistics in August 2004, and a Ph.D. degree in Management Science in December 2004.

Xinli is a member of the Beta Gamma Sigma International Honorary Society, American Statistical Association, Institute of Operations Research and Management Science (INFORMS) and Decision Science Institute. She was the speaker at the 2003 INFORMS annual conference.

Xinli loves math since she was in middle school. She has won many mathematics and physics competition awards. She enjoyed the time spent in UT and greatly appreciate the faculty and staff who have helped her to become a mature researcher and shape her professional career.