HEC Montréal
Affilié à l'Université de Montréal

Data mining for commerce problems: global optimization of clusterwise regression and neural networks applied to electronic negotiations

par

Réal A. Carbonneau

École des Hautes Études Commerciales

Thèse présentée à la Faculté des études supérieures

en vue de l'obtention du grade de

Philosophiae Doctor (Ph.D.)

en administration, spécialisation : Méthodes Quantitatives

Décembre 2011

Cette thèse intitulée :

Data mining for commerce problems: global optimization of clusterwise regression and neural networks applied to electronic negotiations

présentée par

Réal A. Carbonneau

a été évaluée par un jury composé des personnes suivantes:

_____

À déterminer
(Président-rapporteur)

_____

Gilles Caporossi
(Co-directeur de recherche)

_____

Pierre Hansen
(Co-directeur de recherche)

_____

À déterminer
(Membre du jury)

_____

À déterminer
(Examinateur externe)

_____

À déterminer
(Représentant du doyen de la FES)

RÉSUMÉ

Ce travail explore deux sujets d'exploitation de données reliés au commerce. Le premier est l'optimisation du problème de régression par classe et le deuxième est la modélisation prédictive de négociations par réseaux de neurones.

L'optimisation globale exacte du problème de régression par classe (Charles 1977; Diday 1979; Späth 1979) est difficile. Avant de commencer la recherche courante, aucune méthode pour le résoudre de façon exacte n'avait été publiée durant plus de trente ans depuis que le problème a été proposé. Cette première partie du travail explore l'optimisation exacte de la régression par classe, basée sur trois méthodes d'optimisation; des modèles de programmation mathématique, l'optimisation par séparation et évaluation progressive, et la génération de colonnes.

La première méthode d'optimisation appliquée au problème de régression par classe est par un modèle de programmation mathématique. Une formulation par programmation logique-quadratique mixte avec implication de contrainte est présentée et comparée avec une formulation quadratique traditionnelle avec grand-M, cette dernière ne pouvant garantir l'optimalité car les coefficients, et donc les erreurs, peuvent être arbitrairement grands. Les temps d'optimisation ainsi que l'optimalité des solutions pour deux classes ont étés vérifiés empiriquement sur vingt ensembles de données réels et sur trois séries de données synthétiques qui varient de vingt à cent observations et de deux à dix variables indépendantes. Quelques ensembles de données ont aussi étés partitionnés en trois classes.

Deuxièmement, une méthode d'optimisation par séparation et évaluation progressive est proposée pour résoudre le problème de régression par classe. Celle-ci est une extension de l'optimisation par séparation et évaluation progressive répétitive de Brusco (*repetitive branch and bound algorithm*, RBBA) (Brusco & Stahl 2005; Brusco 2006). La nouvelle stratégie est composée d'optimisation heuristique itérative, de nouvelles méthodes d'ordonnancement des observations et de l'optimisation d'un nombre restreint de sous-ensembles de fin. Ces trois caractéristiques importantes offrent de grandes améliorations dans la rapidité de l'optimisation du problème. Cette même stratégie s'applique à une

gamme d'applications qui s'étend au-delà de la seule régression par classe. De plus, une implémentation efficace de calculs incrémentaux dans l'algorithme de la recherche dans l'arbre d'énumération permet d'éliminer la majorité des calculs redondants. Des expériences sur des données réelles et simulées permettent de comparer les diverses caractéristiques de l'algorithme proposé avec l'optimisation du même problème par CPLEX basée sur une formulation par programmation logique-quadratique. Les résultats indiquent que toutes les composantes de l'algorithme proposé offrent des améliorations appréciables en termes de rapidité d'optimisation, et que la meilleure performance correspond généralement à leur utilisation combinée (Carbonneau, Caporossi & Hansen 2011).

La troisième stratégie pour optimiser le problème de régression par classe est basée sur la génération de colonnes (Dantzig & Wolfe 1960; Barnhart, Johnson, Nemhauser, Savelsbergh & Vance 1998). Une partie de la stratégie proposée utilise une heuristique sur le problème original pour insérer dans le problème maître restreint les meilleures colonnes connues et toutes variations obtenues à partir des meilleures colonnes par l'ajout ou la suppression d'une observation. La partie suivante de la stratégie est destinée pour le sous-problème et utilise une heuristique gourmande basée sur les valeurs duales, une autre basée sur la meilleure solution connue pour le problème original, et une optimisation heuristique répétée pour essayer d'insérer une ou plusieurs colonnes rapidement et pour terminer le sous-problème rapidement. Si ces heuristiques échouent à identifier une colonne qui améliore la solution (coût réduit négatif), alors une recherche exhaustive est débutée commençant par des sous-ensembles de fin qui sont progressivement plus grands tout en exécutant itérativement l'optimisation heuristique pour assurer un équilibre entre la recherche exacte et heuristique. Pour l'optimisation exhaustive et heuristique du sous-problème, les observations sont ordonnées par leurs inclusions dans les règles de paires jointes et puis par leurs valeurs duales. Une série d'expériences empiriques démontre que la méthode développée de génération de colonnes surpasse la meilleure alternative connue, celle d'optimisation par séparation et évaluation progressive, lorsque le nombre de classes est supérieur à trois. Ces expériences examinent aussi l'importance de chaque composante de la méthode proposée et démontrent que chacune d'entres elles amène des gains de performance à des coûts minimes. De plus, ce travail de recherche démontre et développe

davantage l'utilisation réussie du nouveau paradigme d'optimisation de sous-ensembles de fin progressivement plus grands tel qu'innové par l'optimisation par séparation et évaluation progressive répétitive de Brusco (*repetitive branch and bound algorithm*, RBBA).

L'application de l'exploitation de données aux négociations électroniques est pertinente en ce monde interconnecté. Les systèmes de négociations électroniques peuvent incorporer des modèles informatisés et des algorithmes pour aider les négociateurs à atteindre leurs objectifs. Il existe une opportunité pour développer une composante qui permet d'estimer et d'analyser une réaction probable d'une contrepartie à une offre tentative avant qu'elle ne soit soumise. Ce travail propose une approche de modélisation par comparaison de paires qui offre la possibilité de développer des modèles flexibles et génériques pour la prédiction des contre-offres quand les cas de négociations sont similaires. La clé pour atteindre ce but est que chaque question de négociation est prédite en paire avec chacune des autres questions de négociation et les permutations de toutes les paires de questions de négociations sont confondues. Cette fusion de données permet l'extraction de modèles communs à toutes les questions de négociation, la résultante étant une fusion de modèles. Des expériences avec des négociations électroniques démontrent que la performance prédictive du modèle proposé est équivalente à celle des modèles qui sont spécifiques à un cas de négociation tout en offrant un haut degré de flexibilité et généralité même pour la prédiction sur des nouvelles questions de négociations (Carbonneau, Kersten & Vahidov 2011).

SUMMARY

This work explores two commerce related data mining topics. The first is the exact optimization of the clusterwise regression problem and the second predictive negotiation modeling using neural networks.

Exact global optimization of the clusterwise regression problem (Charles 1977; Diday 1979; Späth 1979) is a challenging data mining technique. Before starting the current research work, there were no published feasible methods for performing this clustering optimally, even though it has been over thirty years since its original proposal. This first part of this work explores global optimization of the clusterwise regression problem using three optimization methods; mathematical programming, branch and bound optimization and column generation.

The first optimization approach applied to the clusterwise regression problem is by mathematical programming. A mixed logical-quadratic programming formulation with implication of constraints is presented and contrasted against a quadratic formulation based on the traditional big-M, which cannot guarantee optimality because the regression line coefficients, and thus errors, may be arbitrarily large. Clusterwise regression optimization times and solution optimality for two clusters were empirically tested on twenty real datasets and three series of synthetic datasets ranging from twenty to one hundred observations and from two to ten independent variables. Additionally, a few small real datasets were clustered into three lines (Carbonneau, Caporossi & Hansen 2011).

Secondly, a branch and bound strategy is proposed for solving the clusterwise regression problem, extending Brusco's repetitive branch and bound algorithm (RBBA) (Brusco & Stahl 2005; Brusco 2006). The resulting strategy relies on iterative heuristic optimization, new ways of observation sequencing, and branch and bound optimization of a limited number of ending subsets (BBHSE). These three key features lead to significantly faster optimization of the complete set and the strategy has more general applications than only for clusterwise regression. Additionally, an efficient implementation of incremental calculations within the branch and bound search algorithm eliminates most of the redundant

ones. Experiments using both real and synthetic data compared the various features of the proposed optimization algorithm and contrasted them against a benchmark mixed logical-quadratic programming formulation optimized by CPLEX. The results indicate that all components of the proposed algorithm provide significant improvements in processing times, and, when combined, generally provide the best performance, significantly outperforming CPLEX (Carbonneau, Caporossi & Hansen 2011).

The third strategy is a column generation (Dantzig & Wolfe 1960; Barnhart, Johnson, Nemhauser, Savelsbergh & Vance 1998) based approach for solving the clusterwise regression problem. The proposed strategy firstly relies on heuristic optimization of the original problem for insertion of the best known columns and all possible one-observation-perturbations into the restricted master problem. Secondly, for the subproblem, a greedy heuristic based on the dual variables, another based on the best known solution to the original problem, and multistart heuristic optimization are all employed to attempt to insert one or more columns early and stop the subproblem search. If these heuristics fail to identify an improving column, an exhaustive search is performed starting with incrementally larger ending subsets, all the while iteratively performing heuristic optimization to ensure a proper balance of exact and heuristic optimization. For the subproblem heuristics and branch and bound search, observations are sequenced by their inclusion in joint pair rules and by their duals. A series of experiments demonstrated that the extended column generation approach outperforms the best known alternative (BBHSE) when the number of clusters is greater than three. The experiments also explore the importance of each proposed component and demonstrate that all of them provide performance increases with minimal costs. Additionally, the current work further demonstrates and expands the successful use of the new paradigm of using incrementally larger ending subsets to strengthen the lower bounds of a branch and bound search as pioneered by Brusco's Repetitive Branch and Bound Algorithm (RBBA).

The application of data mining techniques to electronic negotiations is relevant to today’s interconnected world. Electronic negotiation systems can incorporate computational models and algorithms in order to help negotiators achieve their objectives. An important opportunity in this respect is the development of a component which can

assess an expected reaction by a counterpart to a given trial offer before it is submitted. This work proposes a pairwise modeling approach that provides the possibility of developing flexible and generic models for counteroffer prediction when the negotiation cases are similar. The key feature is that each negotiated issue is predicted while paired with each of the other issues and the permutations of issue pairs across all negotiation offers are confounded together. This data fusion permits extractions of common relationships across all issues, resulting in a type of pattern fusion. Experiments with electronic negotiation data demonstrated that the model's predictive performance is not compromised as compared to case-specific models while offering a high degree of flexibility and generality even when predicting for a new, previously unseen issue (Carbonneau, Kersten & Vahidov 2011).

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES

ANN – Artificial Neural Networks
BARON – Branch and Reduce Optimization Navigator
BB – Branch and Bound
BBHSE – Branch and Bound with Heuristics, Sequencing and Ending subset optimization
CE – Clustering Error
CEO – Chief Executive Officer
CG – Column Generation
CPU – Central Processing Unit
EM – Expectation Maximization
FPC – Fixed Point Clusters
GA – Genetic Algorithm
GS – Genetic Search
MAE – Mean Absolute Error
MIQP – Mixed Integer Quadratic Programming
MLLP – Mixed Logical-Linear Programming
MLQP – Mixed Logical-Quadratic Programming
MLR – Multiple Linear Regression
MLRP – Multiple Linear Regression Pairwise
NNF – Neural Network Full
NNP – Neural Network Pairwise
NSA – Negotiation Software Agent
OAA – One Against All
OAO – One Against One
OCMLR – Optimal Clusterwise Multiple Linear Regression
OPL – Optimization Programming Language
PAQ – P Against Q
PWM – Pair-Wise Model
RBBA – Repetitive Branch and Bound Algorithm
RF – Random Forests
RSSE – Reduced Sum of Squared Errors
SA – Simulated Annealing
SD – Standard Deviation
SSE – Sum of Squared Errors
SVM – Support Vector Machines
VNS – Variable Neighborhood Search
WCSS – Within-Cluster Sums of Squares

LISTE DES ABRÉVIATIONS

avg. – average
cumul. – cumulative
distr. – distribution
max. – maximum
min. – minimum
obs. – observations
optim. – optimization
proc. – processing
st. dev. – standard deviations

INTRODUCTION GÉNÉRALE

L'exploitation des données est un sujet qui suscite un intérêt croissant autant de la part des chercheurs que des praticiens, une croissance corrélée aux avances exponentielles des matériels informatiques. L'explosion du volume de données collectionnées et entreposées sous format électronique accentue inévitablement les besoins de traitements qui sont plus sophistiqués que de simples rapports. Les techniques d'exploitation de données sont diverses, mais elles reposent souvent sur l'optimisation d'un modèle pour le rendre conforme à un ensemble de données. Pour de petits ensembles de données, des techniques d'optimisation exacte sont possibles, mais quand les ensembles de données sont de tailles importantes, seules des heuristiques peuvent optimiser les modèles dans des temps raisonnables.

Parmi les diverses techniques, celles de classification automatique (non-supervisées) et celles d'approximation de fonction (supervisées) sont relativement courantes. La régression par classe (Charles 1977; Diday 1979; Späth 1979) est une technique de classification automatique basée sur la régression linéaire. La régression linéaire modélise des relations linéaires entre des variables dépendantes et une variable indépendante en minimisant la somme des carrées des erreurs des observations relative à la ligne ou à l'hyperplan. La régression par classe modélise K classes d'observations de telle sorte que la somme des carrés des erreurs est minimisée pour l'ensemble des classes. Avant de commencer la présente recherche, aucune méthode pour résoudre la régression par classe de façon exacte n'avait été publiée durant plus de trente ans depuis que le problème a été proposé. Ce modèle semble être une cible intéressante pour la recherche en optimisation exacte de modèles d'exploitations de données, et sera le sujet des trois premiers chapitres de cette thèse. Spécifiquement, trois grandes approches d'optimisation sont utilisées pour le modèle de régression par classe; des modèles en programmation mathématique, l'optimisation par séparation et évaluation progressive, et la génération de colonnes. Cette première partie sur

la régression par classe explore des questions relatives à l'amélioration des temps d'optimisation exacte du modèle d'exploitation de données.

D'un autre côté, lors de l'application de modèles d'exploitations de données, l'emphase est souvent sur la qualité des modèles. Par exemple, le problème d'assister ou d'automatiser les négociations électroniques est une tâche relativement complexe en informatique car même si les données sont en format électronique, les contreparties sont des humains et l'environnement peut changer. De tels problèmes apportent plusieurs considérations de modélisation qui dépassent le simple choix du modèle d'apprentissage et son optimisation.

En négociation électronique, avoir une approximation de la réaction espérée de la contrepartie sur une offre planifiée est utile. On peut faire appel à des modèles d'apprentissage de fonction, tel que les réseaux de neurones qui sont bien étudiés et utilisés en recherche et en pratique. Par contre, l'application des réseaux de neurones sur un problème difficile, tel que pour la prédiction des contre-offres en négociations électroniques, nécessite le développement de stratégies qui assurent une flexibilité et une robustesse tout en assurant des résultats de qualité. Considérant ces opportunités et défis, le quatrième chapitre traite de l'application des réseaux de neurones à la prédiction des contre-offres en négociation électronique.

Cette thèse aborde quatre des éléments importants de l'exploitation de données; l'optimisation des modèles, la classification automatique (apprentissage non-supervisé), l'apprentissage supervisé et l'application en milieux réalistes et dynamiques.

CHAPTER I:

Globally optimal clusterwise regression by mixed logical-quadratic programming

Réal A. Carbonneau

Gilles Caporossi

Pierre Hansen

Department of Management Sciences

GERAD and HEC Montréal

3000, chemin de la Côte-Sainte-Catherine

Montréal, Québec H3T 2A7, Canada

This paper was published in the European Journal of Operational Research (2011)

Abstract

Exact global optimization of the clusterwise regression problem is challenging and there are currently no published feasible methods for performing this clustering optimally, even though it has been over thirty years since its original proposal. This work explores global optimization of the clusterwise regression problem using mathematical programming and related issues. A mixed logical-quadratic programming formulation with implication of constraints is presented and contrasted against a quadratic formulation based on the traditional big-M, which cannot guarantee optimality because the regression line coefficients, and thus errors, may be arbitrarily large. Clusterwise regression optimization times and solution optimality for two clusters are empirically tested on twenty real datasets and three series of synthetic datasets ranging from twenty to one hundred observations and from two to ten independent variables. Additionally, a few small real datasets are clustered into three lines.

1.1 Introduction

Least squares multiple linear regression is a common statistical tool for modeling linear relationships between one or more independent variables and a dependent variable. This is performed by fitting a line (or hyperplane when there are multiple independent variables) to the data such that it minimizes the sum of squared errors (SSE). The term line will be used for both line and hyperplane in this paper.

This concept has been extended to fitting multiple lines to mutually exclusive subsets of observations of a dataset (Charles 1977; Diday 1979; Späth 1979). A dataset may have been collected from a process that is actually a set of different linear phenomena which would be best explained by a set of linear functions. This is a clustering problem with the objective of finding a predefined number of lines that best fit the data in the least squares sense, thus called clusterwise regression in English and “regression typologique” in the original French work. Although the problem has been known for over thirty years, there has not yet been any published feasible way to solve it optimally. Previous work has mentioned

that “clusterwise regression is a tough combinatorial optimization problem” (Lau, Leung & Tse 1999), possibly referring to its similarity to the set covering problem (Karp 1972), however, there does not seem to be a proof yet that it is NP-complete. There has been an in depth examination of the identifiability of clusterwise regression models which was described as being difficult and partially identifiable (Hennig 2000). Clusterwise regression has been discussed as being important to such areas as spline estimation, utility function clustering and response based segmentations of customers, markets, regions, subjects, strategies or investors (Charles 1977; Diday 1979; Späth 1979; Lau, Leung & Tse 1999; Hennig 2000). Global optimization of clusterwise regression problems themselves may be of theoretical and practical interest in specific situations, and in addition, the failure of the traditional big-M method to guarantee global optimality and the success of mixed-logical quadratic programming shown in this work may provide insight towards exact optimization of other problems.

The clusterwise least squares multiple linear regression problem can be formalized with mathematical programming (MP) notation, similar to the cubic formulation previously proposed (Lau, Leung & Tse 1999). The problem is defined by; the number of clusters (K), the number of independent dimensions (D), and the number of observations (O). The iterators are for; a cluster ($k \in \{1,...,K\}$), an independent dimension ($d \in \{1,...,D\}$), and an observation ($o \in \{1,...,O\}$). The model parameters are; the independent variable for an observation and dimension (x_{od}) and the dependent variable for an observation (y_o). The model variables are; the cluster assignment of an observation to a cluster (Z_{ok}), the regression coefficient (aka β) for a dimension of a cluster (b_{ok}) and the error for an observation of a cluster (e_{ok}).

$$\text{SSE} = \min \sum_{k=1}^{K} \sum_{o=1}^{O} \left(Z_{ok} e_{ok}^2 \right) \tag{1.1}$$

$$\text{s.t.} \quad \sum_{d=1}^{D} \left(b_{dk} x_{od} \right) + e_{ok} = y_o \quad \forall o, \forall k \tag{1.2}$$

$$\sum_{k=1}^{K} \left(Z_{ok} \right) = 1 \quad \forall o \tag{1.3}$$

$$Z_{ok} \in \{0,1\} \quad \forall o, \forall k \tag{1.4}$$

The objective (1.1) is the minimization of the sum over all clusters of the sum of squared errors (SSE) for their observations relative to their regression line. The constraint (1.2) fits the regression lines to the data by adjusting the coefficient and error terms. An observation can only be assigned to one cluster at a time (1.3) and the cluster assignment is binary (1.4). Compared to the previously proposed formulation (Lau, Leung & Tse 1999), the above formulation is simplified, not limited to two clusters and can model an intercept of zero or an intercept that best fits the data. It was also proposed, with proof, that the binary constraint on Z_{ok} (1.4) can be replaced by a non-negative constraint (Lau, Leung & Tse 1999). However, in the exceptional situation where an observation is exactly on the intersection of two or more lines, the assignment could be fractional. Consequently, the binary constraint is kept in the current cubic formulation.

Throughout this paper, the regression models always include an intercept, which is achieved by adding to each dataset a bias variable fixed at a constant of one, and thus the constant is included in the number of independent dimensions *D*. Regression lines with an intercept of zero can be fit to the data by simply not including this constant. A regression line must have *D* observations to be determined and more than *D* observations to be overdetermined. This assumes full rank of the independent variables (excluding the bias variable if specified), or else rank (plus 1 for the bias, if specified) applies instead of D. If all lines are determined or underdetermined, a large number of different cluster configuration all result in a globally optimal solution of zero error. Since *K* lines are to be fit to the data, there must be at least *K-1* lines which are overdetermined to have the potential for identifiable parameters for every regression line. Thus, the minimum number

of observations required to have the potential of an identifiable clusterwise regression solution is *(D)K+(K-1)*.

This paper is structured as follows; section 1.2 provides an overview of some previous heuristic approaches, section 1.3 presents the two quadratic formulations, section 1.4 introduces the datasets, section 1.5 overviews the experimental protocol, section 1.6 presents the results and related discussion and the conclusions are presented in section 1.7.

1.2 Heuristics

There exists a variety of work on solving the clusterwise regression problem, yet all of them are heuristics with no guaranteed bound on the error. This is probably because there are K^O possible clustering configurations and since there is a minimum of D^2O regression computations (Gentleman 1973) to perform per clustering configuration, enumeration of the complete problem search space requires at least K^OD^2O operations. Consequently, complete enumeration is not possible for anything but the most trivial datasets. The first solutions were achieved using the exchange method which is stepwise optimal but not globally optimal. This method consists of an initial random cluster assignment and then tentative assignment of each observation to each cluster until a complete pass over the observations does not result in any improvement (Charles 1977; Diday 1979; Späth 1979; Späth 1981; Späth 1982). The simulated annealing (SA) metaheuristic, which is the analog of heating and cooling a material to reach a more stable (lower energy) state, is used for minimizing the clusterwise regression problem (DeSarbo, Oliver & Rangaswamy 1989). The variable neighborhood search (VNS) metaheuristic, where local minimization is performed using the exchange algorithm and the neighborhood is systematically changed to escape from local optima, is employed for solving clusterwise regression problems (Caporossi & Hansen 2005). A genetic algorithm (GA), which combines both random changes and crossovers of the better solutions in a way similar to the mechanisms of genetics, has also been applied to the clusterwise regression problem (Aurifeille 2000; Aurifeille & Medlin 2001; Aurifeille & Quester 2003). Others present optimization by

Bayesian tree growing (Ciampi, Rich, Dyachenko, Villalobos, Murie & Nadon 2007), and by approximation (Mirkin 2005) which was termed regression-wise clustering.

Expectation maximization (EM) (DeSarbo 1988; Wedel 1990; Wedel & DeSarbo 1995; Wedel 1998) has been used to achieve mixture model clusterwise regression solutions where the clusters overlap by permitting fractional assignments. However, the minimized function differs from the Späth formulation and is therefore not directly comparable. The expectation maximization has been used to provide non-overlapping clusterwise regression solutions by "hardening" the fractional cluster assignments (Aurifeille 2000). A mathematical programming approach has also been proposed, however, since the formulation was cubic, the actual optimization of the formulation is only a heuristic and it "may generate local optimum" (Lau, Leung & Tse 1999).

1.3 Mathematical programming

Mathematical programming formulations for optimization problems are relevant over the long term because they precisely define the problem in a way that can be interpreted by optimizers and because these optimizers are continuously being improved. Even if it may be possible to develop a specialized algorithm which performs better than the best current optimizers for a specific problem, such an algorithm can be included in the optimizers in the future. Thus a mathematical programming formulation remains relevant. Depending on the formulation of the problem, mathematical programming can provide a generic method for identifying globally optimal solutions to certain types of problems.

Although identifying the globally optimal solution to a clusterwise regression problem by no means guarantees identifying the true model, on average, these solutions will lead to better models than random local optima identified by heuristics. In addition, global optimization is an important tool for further research. It is useful as a benchmark because a heuristic should be faster than an algorithm that identifies the globally optimal solution. The optimal solution is also useful as a reference to know how far a heuristic's solution is from the global optimum and if it can identify it. For example, it permits empirical

validation of procedures theoretically shown to “be able to discover all substantial FPCs [(Fixed Point Clusters)] with high probability” (Hennig 2002). A global optimization algorithm is also a useful building block for developing other heuristics by finding optimal solutions for subsets, which can subsequently be used to analyze the complete search space and the structure of the data. Global solutions may also be useful for further research on quality measures of clusterwise regression models.

As stressed by Brusco et al., clusterwise regression makes no effort to distinguish between error explained by clustering and error explained by regression (Brusco, Cradit, Steinley & Fox 2008). Since clusterwise regression fits multiple lines to the data, the overfitting potential is much greater than that of a single regression line and consequently an evaluation procedure has been proposed (Brusco, Cradit, Steinley & Fox 2008). However, evaluating and addressing this overfitting problem is not in the scope of the current research and neither is the statistical validity of identified optimal clusterwise regression models. This research considers only the feasibility and processing time for finding the optimal solution to a clusterwise multiple linear regression problem. It may be called optimal clusterwise multiple linear regression (OCMLR).

Regression alone is a quadratic mathematical programming problem, therefore clusterwise regression becomes a cubic mixed integer mathematical programming problem because of the additional multiplication for the cluster assignment indicator (1.1). Unfortunately, this formulation of the clusterwise regression problem is extremely difficult to solve exactly with proof of optimality. There exist only a few programs, such as BARON (Sahinidis 1996; Tawarmalani & Sahinidis 2002; Tawarmalani & Sahinidis 2004), which may be able to solve this cubic formulation exactly, however, it would only be possible for the most trivial datasets. Yet, even this is not certain since such techniques require bounded variables, and as will be shown in the next section, the bounds for both the coefficients and the error cannot be known in advance because a slope (coefficient) could be of any steepness.

The cubic clusterwise regression mathematical programming formulation can be re-formulated into a mixed integer quadratic problem or a mixed logical-quadratic problem. Quadratic programming formulations take advantage of more efficient optimization algorithms and programs, but most importantly, these will provide a guaranteed globally optimal solution for a given problem as defined by the formulation.

1.3.1 Big-M mixed integer quadratic programming re-formulation

The big-M formulation is commonly used to linearize the product of indicator variables and continuous variables (Kojima, Mizuno & Yoshise 1989; Megiddo 1989; Monteiro & Adler 1989; Spedicato 1994), such as the cluster assignment variable Z_{ok}. The big-M formulation consists of adding a large constant to the regression constraint (1.2) which can be turned on and off by the cluster assignment variable, thus activating the constraint (1.6) which forces the error variable to be used when this error is positive, thus increasing the value of the objective function (1.5).

$$\text{SSE} = \min \sum_{k=1}^{K} \sum_{o=1}^{O} \left(e_{ok}^2 \right) \tag{1.5}$$

$$\text{s.t.} \quad -M\left(1-Z_{ok}\right) \le \sum_{d=1}^{D} \left(b_{dk} x_{od} \right) - y_o + e_{ok} \le M\left(1-Z_{ok}\right) \quad \forall o, \forall k \tag{1.6}$$

$$\sum_{k=1:k\le o}^{K} \left(Z_{ok} \right) = 1 \quad \forall o \tag{1.7}$$

$$Z_{ok} \in \{0,1\} \quad \forall o, \forall k \tag{1.8}$$

When the observation is placed in a cluster using the binary cluster assignment variable Z_{ok}, the terms with the large constant M disappear, thus the constraint is forced between zero and zero, consequently forcing the error to have a non-zero value if the observation does not fit the line perfectly. When the observation is not assigned to the current cluster using the binary cluster variable Z_{ok}, the large constant M is turned on and the constraint is free between $-M$ and M. Regardless of how far (up to the large M) the observation is from the line, neither the error, nor the values of any other variables, nor the value of the

objective function will be affected. This linearization of the Z_{ok} product removes it from the cubic objective (1.1) which thus becomes quadratic (1.5).

To help reduce the problem size, it is simple to implement some partial symmetry breaking by using the minimum of K and o as the maximum number of clusters permitted for the constraint that requires the observation to be assigned to only one cluster (1.7). This means that observation one will always be forced to cluster one and observation two will be forced to either cluster one or two and so on. This will provide perfect symmetry breaking for a two cluster problem, but will provide incrementally less symmetry breaking as the problem has more clusters. For example, if the second observation belongs to the first cluster and the third observation belongs to another cluster, then the third observation should be forced to be in the second cluster, but the one cluster constraint (1.7) will permit it to be in both the second and third cluster. The last constraint of the model is binary cluster assignment (1.8) as before (1.4).

For use in many modern mathematical programming environments, the big-M constraint (1.6) must be split into two inequalities with the constants on the right side. The lower bound for the regression constraint (1.9) will force the equation to be greater than or equal to zero if the observation is in the class and the upper bound for the regression constraint (1.10) will force the equation to be less than or equal to zero thus bringing the two constraints into effect.

$$-MZ_{ok} + \sum_{d=1}^{D}\left(b_{dk}x_{od}\right) + e_{ok} \geq y_o - M \qquad \forall o, \forall k \tag{1.9}$$

$$MZ_{ok} + \sum_{d=1}^{D}\left(b_{dk}x_{od}\right) + e_{ok} \leq y_o + M \qquad \forall o, \forall k \tag{1.10}$$

Although the big-M formulation has some advantages because it is quadratic, it is well known that the big-M formulation can lead to numerical stability problems when the M is large (Lustig 1990). Additionally, with the big-M formulation for clusterwise regression,

there is no guarantee that the constant *M* is large enough to include the optimal solution. The *M* constant must be larger than the maximum distance of all observations to all clusterwise regression lines (intra-cluster and inter-cluster error), and this required value is not known in advance. For example, if one of the lines has a steep slope, an observation on another line can easily deviate enough to cause a large error relative to this line (Figure 1). In addition, the big-M must be large enough for every intermediary solution in the branch and bound or cutting procedures as to avoid potentially cutting off the global optimum.

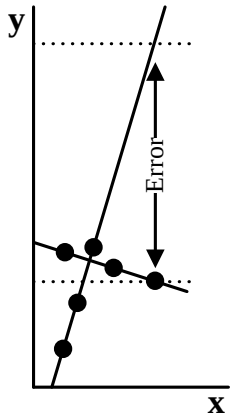

Figure 1 Large error when the slope is steep

1.3.2 Mixed logical-quadratic programming re-formulation

The big-M formulation approach conceives the clusterwise regression problem as continuous with some variables restricted to integers, thus resulting in the numerical problems described previously. However, the natural formulation of the problem is that the Z_{ok} cluster assignment variable is linked to the relevant constraint by logical implication. In an approach called mixed logical-linear programming (MLLP), or in this case, mixed logical-quadratic programming (MLQP), the logical propositions remain in their natural formulation while at the same time taking advantage of the strength of both logic processing and linear or quadratic programming (Hooker & Osorio 1999; Hooker, Ottosson, Thorsteinsson & Kim 2000; Hooker 2002; Hooker 2007).

In contrast to a continuous formulation which is solved to be integer by cuts and branching when required, the mixed logical approach manages the logical constraints and branches on the discrete variables while solving continuous relaxation problems wherever it may help the search. Consequently, the discrete variables are not directly part of the continuous problem and thus all related numerical problems associated with the big-M formulation are avoided. The mixed logical approach also takes advantage of logical processing and constraint propagation to reduce the search space as the search progresses, in particular by proving infeasibility of portions of the search space. Continuous relaxations of the problem are solved to identify feasible solutions early in the search and to strengthen the search bounds throughout the search. These continuous problems are specific to the current nodes and may include a variety of cuts and inequalities to help further strengthen the bounds or to demonstrate infeasibility of parts of the search space. An overview of such a generic MLLP branching algorithm is provided in Figure 9 on page 425 in the original article (Hooker & Osorio 1999). By such a mixed logical-quadratic formulation and optimization algorithm, the clusterwise regression problem can be solved exactly to its global optimum in contrast to the common big-M formulation. The mixed logical-quadratic programming clusterwise regression formulation is achieved by replacing the big-M constraint (1.6) with the logical implication ($\Rightarrow$) of a constraint, where the cluster assignment variable Z_{ok} implies (activates) the regression error constraint (1.11).

$$(Z_{ok} = 1) \Rightarrow \sum_{d=1}^{D} (b_{dk} x_{od}) + e_{ok} = y_o \qquad \forall o, \forall k \tag{1.11}$$

1.3.3 Implementation

In this paper, both the implication of constraints and the big-M formulations of the quadratic programming clusterwise regression model are optimized using the IBM ILOG OPL-CPLEX environment. Thus the models are implemented using the OPL programming language (Van Hentenryck, Lustig, Michel & Puget 1999; IBM 2009; IBM 2009). The OPL quadratic model based on the implication of constraints and the adjustments required for implementing the big-M formulation are presented below. In this modeling language,

the *Z* cluster assignment variables are defined as boolean, which implies the binary constraint. The declarations required for these OPL models are presented in Appendix 1.A.

Quadratic programming clusterwise regression OPL model with implication of constraints

```
1 minimize sum(o in O) sum(k in K) e[o,k]^2;
2 subject to {
3  forall(o in O){
4   forall(k in K){
5    (Z[o,k]==1) => sum(d in D) x[o,d]*b[d,k] + e[o,k] == y[o]; }
6   sum(k in K: k <= o) Z[o,k] == 1; } }
```

Adjustments for big-M formulation (replaces 5)

```
5a    -M * Z[o,k] + sum(d in D) x[o,d]*C[d,k] + e[o,k] >= y[o] - M;
5b     M * Z[o,k] + sum(d in D) x[o,d]*C[d,k] + e[o,k] <= y[o] + M; }
```

1.4 Datasets

Diverse datasets (Table I) are clustered to demonstrate the feasibility, processing times and optimality of the solutions for both quadratic formulations. There are 20 real datasets chosen because they are publicly available, thus permitting others to reproduce and test the results of the globally optimal solutions. These results may also be useful for subsequent research on heuristics and quality measures. No verification has been done to determine the statistical validity of the clusterwise regression models nor is there any claim that clusterwise regression is the appropriate model for these datasets. These real datasets range from 29 to 82 observations, two to six independent variables and cover a variety of subjects from health to food, finance, industry and politics. In addition, three synthetic series of datasets varying from 20 to 100 observations, two to ten independent variables and various levels of perturbation permit further analysis of the clustering optimization performance.

Two datasets are of particular interest since they were optimized in a previous work on clusterwise regression (Lau, Leung & Tse 1999) and the data is publicly available. The electricity consumption dataset (McCormick 1993; Lau, Leung & Tse 1999) presents per capita consumption for 50 US states and the mortality dataset (Hardle & Stoker 1989; Lau, Leung & Tse 1999) presents results from 58 simulated side impact car collision. Both datasets were previously clustered using a cubic programming formulation, however, the algorithm used to optimize the problem was a heuristic, thus the results may not be exact

global minima. The SSE for the mortality dataset was zero and thus must be a global minimum, however, as will be shown below, the presented SSE of 310.907 (Lau, Leung & Tse 1999) for the electricity dataset is only a local minimum.

In some of the real datasets, not all independent variables in the original set are included to keep the resolution times reasonable. A bias variable fixed at a constant of one is added to each dataset to permit estimation of the intercept, thus *D* in Table XVII is always one more than the number of listed independent variables. The remaining model descriptions are presented in Appendix 1.B.

Table I Dataset overview

Dataset	***O***	***D***	**Independent Variables**	**Dependent**
Acorn (Aizen & Patterson 1990)	39	4	Tree Height, Acorn Size, Atlantic Region	Range
Brink's (Finkelstein & Levin 2001)	47	5	Months, City Worker Collect., Brink, Collection Days	Contractor Collection
Car Fuel Economy (Heavenrich, Murrell & Hellman 1991; Velleman 2010)	82	2	Weight	MPG
CEO Salaries (Velleman 2010)	59	2	Age	Salary
Check Off (Velleman 2010)	56	5	Size, Value, Live, Sales	Yes
Cheese Taste (Moore & McCabe 1998)	30	4	Acetic, H2S, Lactic	Taste
Crime Rates (Vandaele 1978; Hand 1994)	47	6	Age, Educ., 1960 Expenditure, 35-39 Unemployed, Income	Crime Rate
Diabetes (Sochett, Daneman, Clarson & Ehrlich 1987; Hastie & Tibshirani 1990)	43	3	Age, Deficit	C-peptide
Electricity (McCormick 1993; Lau, Leung & Tse 1999)	50	4	Elect. Price, Inc., Gas Price	Electricity
Enrollment (Velleman 2010)	29	5	Year, Unempl., HSGrad, Inc.	Enrollment
Extroversion Car (Miles & Shevlin 2001)	40	4	Sex, Age, Extroversion	Car
House Prices (Miles & Shevlin 2001)	40	2	Size	Price
Mercury Bass (Lange, Royals & Connor 1993)	53	5	Alkal., pH, Calc., Chlorophyll.	Avg Mercury
Mortality (Hardle & Stoker 1989; Lau, Leung & Tse 1999)	58	4	Age, Velocity, Acceleration	Mortality
Nuclear Plants (Cox & Snell 1981)	32	4	Date, Time 2, North East	Cost
Polishing Times (Velleman 2010)	59	3	Diameter, Time	Price
Public Expenditure (Velleman 2010)	48	5	ECAB, Metro., Growth, West	Expenditure
Smoking Cancer (Fraumeni 1968)	44	2	Cigarettes	Lung
Temperatures (Peixoto 1990; Hand 1994)	56	3	Lat, Long	Jan. Temp
Votes (Moore & McCabe 1998)	50	2	1980 Dem	1984 Dem
Random (uniform distribution), varying *O*	20-100	4	Random	Random
Random (uniform distribution), varying *D*	50	2-10	Random	Random
2 Lines, varying normal distribution perturbation	100	2	Random (uniform distribution)	y + Perturb.

Since the real datasets contain variables for a range of small and large values, some variables need to be scaled down, such that the maxima do not go much beyond 10, to ensure numerical stability of the optimization. This is a simple procedure and the solution coefficients can be multiplied by their respective variable's divisor and the solution SSE can multiplied by the square of the dependent variable divisor to compute the values in the original scale.

There are three series of synthetic datasets that will be clustered (Appendix 1.C), the first two are simply random (uniform distribution) variables that range from 0 to 10, rounded to two decimals. The first series has four independent variables and varies from 20 to 100 observations, the second series has 50 observations and varies from two to ten independent variables.

The third series of datasets is generated from two lines and thus attempts to reproduce a dataset for which the clusterwise regression technique is truly appropriate. By having only two independent variables, the x variable and the bias constant for the intercept, the data can be represented in the plane (Figure 8). The first line has a slope of -0.356 and an intercept of 7.583 and the second line has a slope of 0.952 and an intercept of 0.909 (line 1: $y = -0.356x + 7.583$, line 2: $y = 0.952x + 0.909$). The independent variables (x) are randomly drawn from a uniform distribution and rounded to two decimal places. The dependent variables (y) are perturbed by values from a normal distribution by an increasing amount (Appendix 1.C). With this third series of datasets, the effects of increasing perturbations can be examined and it is expected that the more perturbation, the more difficult it will be to estimate good solutions and cut the search space since the problem has less structure. As with many other clustering algorithms, if the perturbations are too large relative to the distance between clusters, some observations will be incorrectly assigned to another cluster than their own.

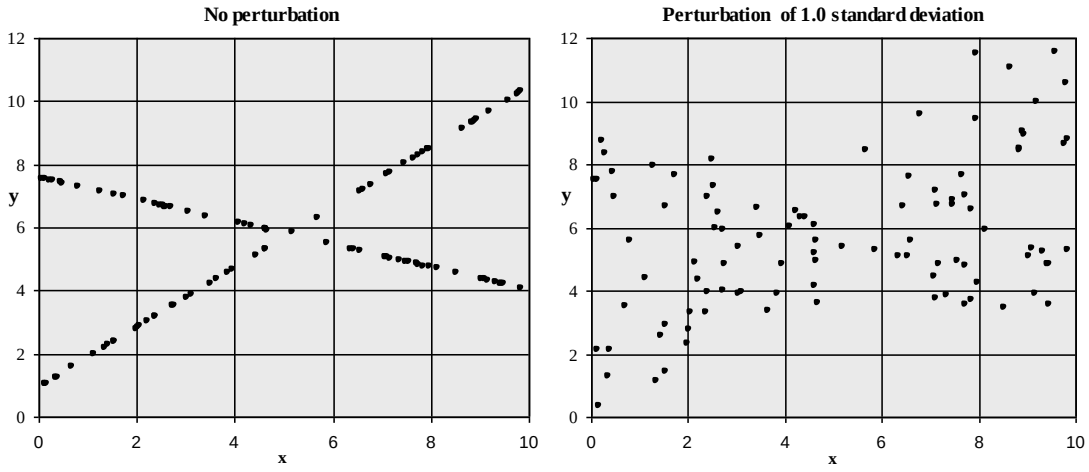

Figure 2 Plots of the two lines dataset with perturbations of zero and one standard deviations

1.5 Experimental protocol

Many algorithms, including those used by CPLEX will explore the search space with varying efficiency depending on the sequence of the data in the dataset. Because of this, it is important to run multiple tests with randomization of the sequence of the data so that an average can be determined. One sequence of data may be advantageous to one algorithm and disadvantageous to another, thus testing only on one sequence of the data does not provide a robust representation of an algorithm's performance both for completion time and optimality. For example, the optimization times obtained by the constraint implication formulation of the Car Fuel Economy dataset, ranged from a minimum of 27 seconds to 125,393 seconds and thus reporting only one value has limited usefulness.

Consequently, for all of the two cluster optimization experiments, the same problem is optimized 100 times while randomizing the sequence of the observations each time. Specifically for the clustering of the synthetic datasets, once the total processing time has passed beyond 100 hours, it is aborted and considered timed out with an average of over one hour. Since only the observation order is randomized and none of the actual observation values are changed, any global optimization algorithms will achieve the same global minimum. Processing time may be different since the starting point and path through

the search space varies. Since there is no guarantee that the big-M formulation will achieve the globally optimal solution, different randomization sequences can lead to different paths through the search space and the big-M constant may not be large enough for intermediate solution along some paths, thus cutting off the global optimal solution. Numerical problems with the big-M formulation may also lead to not finding the optimal solution (larger error) or finding infeasible solutions beyond the constraints (sometimes with a smaller error) and in either case, these non-optimal or infeasible solutions may or may not result in globally optimal clustering. Therefore, both the minimum and maximum solution values are presented for the big-M formulation, as well as the number of times that the actual clustering is not optimal (CE = Clustering Error).

Optimization experiments with three clusters are also performed, however, because of the extensive processing times, it is not feasible to execute them 100 times each. Consequently, optimization is performed only for some datasets and only once and presented as such without the additional statistics.

These experiments examine the effects of the number of observations, independent variables, perturbations and clusters as well as the formulation on the processing time and global optimality of the solutions. As expected from the minimum initial problem space size of $K^O D^2 O$, the number of clusters should have the largest effect, followed by the number of observations and lastly the number of independent variables. However, each of these have a different effect on the actual difficulty of the search space, for example, an increase in the number of observations may weaken the bounds relatively less than an increase in the number of dimensions.

All datasets and models are solved using IBM ILOG OPL 6.3 with CPLEX 12.1.0 (IBM 2009; IBM 2009) on an Intel Core 2 Quad Q8200 CPUs at 2.33GHz with 4 MB of shared L2 cache and 4 GB of RAM running Linux. Each optimization is limited to one CPU core, thus there is no parallel processing. The relevant tolerances parameters are reduced as much as possible in an attempt to ensure optimal and high precision solutions, especially in respects to the big-M formulation, where the constant M was set to 10,000. The node file

flag is set to store nodes on disk (with compression) when the in-memory set is larger than 128 MB.

1.6 Results and discussions

1.6.1 Real datasets

The results of clustering the 20 real datasets are presented in Table II. The electricity dataset (McCormick 1993) is solved by both the implication of constraints and big-M formulation to the global minimum of 284.528 SSE (Table II), thus the previously identified solution of 310.907 SSE (Lau, Leung & Tse 1999) was 9.3% larger than the optimal solution

The big-M formulation is on average much faster than the constraint implication formulation, with the exception of the House Prices and Mortality datasets, which are both relatively small. In some of the more difficult datasets, such as the Car Fuel Economy, Check Off, Electricity, Polishing Times, Public Expenditure and Temperatures datasets, the implication of constraints formulation is from 10 to more than 600 times slower than the big-M formulation. Optimization of the Car Fuel Economy dataset by the implication of constraints formulation took on average more than three hours because of its relatively larger number of observations, however, the big-M formulation averaged at only 18.5 seconds.

Contrary to what was expected, the processing times are usually firstly affected by the number of independent variables and then by the number of observations. This is probably because the increase in the number of dimensions not only increases the size of the search space, but also significantly increases its complexity. Moreover, CPLEX may be far from using the minimum number of computations D^2O for solving the regression part of the problem. There is also a correlation between the search space size and the extent of non-optimal and infeasible results identified by the big-M formulation. Larger problems such as the Check Off, Electricity, Mercury Bass and Public Expenditure datasets result in big-M SSE solutions which are not optimal or infeasible. Notable exceptions are the Cheese Taste

dataset, which is relatively small and has at least one infeasible big-M SSEs and the Temperature dataset which is relatively large and does not have any non-optimal nor infeasible big-M SSEs. Even though the big-M formulation sometimes provides non-optimal and infeasible SSEs, in none of the solutions did this actually affect the optimal clustering of the observations.

The precision of the SSEs reported in Table II vary between datasets because of the prior scaling of the dependent variables. SSEs are reported with a precision of 9 decimal places for the scaled datasets, these are then unscaled back to their original range for reporting in this paper.

Table II Statistics for the real datasets, two clusters, 100 randomized sequences

	Constraint implication			Big-M				
Dataset	**Avg s**	**SD s**	**SSE**	**Avg s**	**SD s**	**Min SSE**	**Max SSE**	**CE**
Acorn	6.0	2.9	223817911.1	3.8	1.2	223817911.1	22381791.11	0
Brinks	1719.2	3290.0	382570037000	63.1	12.5	382570037000	382570037000	0
Car Fuel Economy	11942.8	29334.5	335.9505087	18.5	5.7	335.9505087	335.9505087	0
CEO Salaries	27.0	19.8	768960.336	12.7	2.4	768960.336	768960.336	0
Check Off	1605.1	1700.7	1169.2983490	149.9	33.4	1169.2983490	*1169.2984173	0
Cheese Taste	3.8	1.1	503.7991483	2.5	0.8	*503.7991294	503.7991483	0
Crime Rates	1549.5	1604.1	3329.26314	166.3	59.7	*3329.26214	*3329.26321	0
Diabetes	15.4	11.0	3.640144568	6.9	1.7	3.640144568	3.640144568	0
Electricity	715.3	449.8	284.5280943	76.0	14.1	284.5280943	*284.5281516	0
Enrollment	18.3	5.4	969887.3	7.6	1.2	969887.3	969887.3	0
Extroversion Car	56.5	26.6	1478.422187	14.7	2.3	1478.422187	1478.422187	0
House Prices	45.6	21.1	51655.04437	10.2	1.8	51655.04437	51655.04437	0
Mercury Bass	246.6	156.0	0.550461693	50.1	10.1	*0.550461665	*0.550461718	0
Mortality	0.7	0.9	0.000000000	2.1	2.2	0.000000000	0.000000000	0
Nuclear Plants	24.0	11.9	112810.03462	9.2	1.6	112810.03462	112810.03462	0
Polishing Times	1696.1	4003.5	5315.41062	29.1	5.3	5315.41062	5315.41062	0
Public Expenditure	562.0	357.0	13660.93458	63.3	16.3	13660.93458	*13660.93551	0
Smoking Cancer	12.8	10.8	140.3610520	9.4	2.1	140.3610520	140.3610520	0
Temperatures	750.4	1931.8	428.3865333	8.7	2.0	428.3865333	428.3865333	0
Votes	34.3	15.7	189.3029607	8.7	2.1	189.3029607	189.3029607	0

* Non-optimal or infeasible SSE

1.6.2 Random datasets

The results for clustering the random dataset with four independent variables and an increasing number of observations into two clusters are presented in Table III, where the times increase considerably as the number of observations increase. The constraint

implication formulation can solve the random datasets up to 65 observations in an average time of under one hour and the big-M formulation can solve up to 85 observations. The big-M formulation is many times faster and with less variation (standard deviation) than the constraint implication formulation, presumably because the big-M formulation cuts off portions of the search space that may still contain the global optimum. More importantly, as the number of observations increases, the big-M formulation not only results in more non-optimal and infeasible SSE solutions, but also results in an increasing number of clustering errors. From 55 observations to 85 observations, the average clustering solution error rate (non-optimal) is about 3%. The globally optimal solutions for 90, 95, and 100 observations are presented for reference without statistics.

Table III Statistics for random dataset, 2 clusters, varying observations, 100 randomized sequences

	Constraint implication			Big-M				
O	**Avg s**	**SD s**	**SSE**	**Avg s**	**SD s**	**Min SSE**	**Max SSE**	**CE**
20	1.0	0.4	9.133899158	0.7	0.2	9.133899158	9.133899158	0
25	3.2	1.4	25.136309076	2.1	0.7	*25.136307478	*27.416840770	1
30	8.7	2.8	42.863272347	4.5	1.0	*42.863253721	*43.491956772	1
35	16.9	6.6	52.190342919	7.7	1.6	52.190342919	52.190342919	0
40	46.0	17.7	62.109924028	14.8	2.7	62.109924028	62.109924028	0
45	63.5	26.2	65.746462574	19.4	3.9	65.746462574	65.746462574	0
50	140.0	52.4	71.382739273	35.0	5.9	71.382739273	71.382739273	0
55	253.6	108.4	75.266231828	54.0	10.7	75.266231828	*75.509800476	2
60	640.2	372.1	82.781290364	94.6	16.8	*82.781289335	*83.542342276	1
65	1410.5	1266.9	87.002224378	139.1	24.0	*87.002223693	*89.553171300	2
70	>1h	n/a	94.344584270	314.2	84.8	94.344584270	*99.435352307	5
75	>1h	n/a	118.905538520	1037.6	134.9	*118.905502419	*121.767989821	3
80	>1h	n/a	125.339789460	2404.3	518.0	*125.339765429	*128.866801642	4
85	>1h	n/a	130.038014730	2970.9	561.0	*130.038000024	*131.916263097	3
90	>1h	n/a	140.296203140	>1h	n/a	n/a	n/a	n/a
95	>1h	n/a	147.110688390	>1h	n/a	n/a	n/a	n/a
100	>1h	n/a	163.515942080	>1h	n/a	n/a	n/a	n/a

* Non-optimal or infeasible SSE

Moving from the analysis of optimizing a series of random datasets while varying the number of observations, the effect of the number of independent variables on the optimization is considered. This is explored using a series of random datasets with 50 observations and varying the number of independent variables from two to ten, the results

of which are presented in Table IV. As the number of independent variables is increased from two to ten, the processing times also increase considerably, however, again contrary to what was expected, the number of independent dimensions has a larger impact on processing times and solution optimality than the number of observations. As noted in the previous experiments, the big-M formulation is many times faster and with less variation (SD) than the constraint implication formulation. The constraint implication formulation can solve the random datasets with 50 observations up to six independent variables in an average time of under one hour and the big-M formulation can solve instances with up to eight independent variables. In addition, the number of non-optimal and infeasible solutions as well as clustering errors provided by the big-M formulation increases with the number of independent variables. Globally optimal SSEs for nine and ten independent variables are provided for reference without statistics.

Table IV Statistics for random dataset, varying dimensions, 2 clusters, 50 observations , 100 randomized sequences

	Constraint implication			Big-M				
D	**Avg s**	**SD s**	**SSE**	**Avg**	**SD s**	**Min SSE**	**Max SSE**	**CE**
2	10.2	3.5	74.469007052	9.9	2.1	74.469007052	74.469007052	0
3	36.0	22.4	72.904755763	18.7	3.3	72.904755763	72.904755763	0
4	129.6	49.1	71.382739273	36.6	7.2	71.382739273	*71.792682045	1
5	766.2	588.7	68.988991169	120.9	31.6	*68.988956628	68.988991169	0
6	589.5	398.7	49.127153081	144.5	38.9	*49.127141667	*51.648153240	5
7	>1h	n/a	44.649130777	496.8	120.8	*44.649126310	*45.713554429	4
8	>1h	n/a	39.334926594	2029.0	554.9	*39.334925843	*40.565322861	4
9	>1h	n/a	36.518619272	>1h	n/a	n/a	n/a	n/a
10	>1h	n/a	31.480233113	>1h	n/a	n/a	n/a	n/a

* Non-optimal or infeasible SSE

1.6.3 Two lines with perturbation

The final series of datasets are the synthetic observations generated from two lines with increasing perturbations from the normal distribution. As expected, the processing times increase with the level of perturbation. In only one optimization did the big-M formulation provide a non-optimal SSE, however, this solution also provided non-optimal clustering. This lower error rate may be a result of a more structured search space because of the two distinct lines or it may be because there is only one independent variable that actually

varies. The constraint implication formulation can cluster the 100 observations with a perturbation of up to 0.9 standard deviations into two lines at an average time of under one hour and the big-M formulation can cluster up to 1.0 standard deviation at an average time of under one hour. Globally optimal SSEs for standard deviations of 1.0, 1.1 and 1.2 are provided for reference without statistics.

Table V Statistics for 2 lines dataset, 100 observations, 2 clusters, varying normal distribution perturbation, 100 randomized sequences

	Constraint implication			Big-M				
Pert.	**Avg s**	**SD s**	**SSE**	**Avg**	**SD s**	**Min SSE**	**Max SSE**	**CE**
0.0	9.3	8.1	0.000787706	5.3	1.9	0.000787706	0.000787706	0
0.1	8.4	1.6	1.072328889	6.1	1.9	1.072328889	1.072328889	0
0.2	9.3	2.4	3.992202482	7.9	3.3	3.992202482	3.992202482	0
0.3	11.3	4.1	8.870313459	8.6	3.5	8.870313459	8.870313459	0
0.4	13.0	7.9	15.875624009	11.0	5.3	15.875624009	15.875624009	0
0.5	21.5	12.7	24.918109886	15.9	6.0	24.918109886	*29.031062318	1
0.6	37.7	23.1	36.084968447	27.3	11.7	36.084968447	36.084968447	0
0.7	49.5	30.6	48.580311516	57.7	35.3	48.580311516	48.580311516	0
0.8	143.3	113.7	62.697872680	117.1	50.3	62.697872680	62.697872680	0
0.9	2855.2	9319.7	78.196825542	289.3	136.9	78.196825542	78.196825542	0
1.0	>1h	n/a	94.640006039	1245.3	570.7	94.640006039	94.640006039	0
1.1	>1h	n/a	111.541142790	>1h	n/a	n/a	n/a	n/a
1.2	>1h	n/a	129.232063610	>1h	n/a	n/a	n/a	n/a

* Non-optimal or infeasible SSE

1.6.4 Three clusters

Performing clusterwise regression for three lines is significantly more difficult because, as indicated in the minimum search space formula of $K^O D^2 O$, the base of the exponent of observations increases from two to three. Thus, not all datasets could be solved for three clusters (Table VI), and only one iteration is performed per dataset. The difference in optimization times between the implication of constraints and big-M formulation is significantly larger for three clusters, however, in the few problems that can be compared, the big-M formulations did identify the globally optimal solution.

Table VI Results of clustering the real datasets into three clusters

	Constraint implication		Big-M	
Dataset	**Hours**	**SSE**	**Hours**	**SSE**
Acorn	406.7	70493712.1	54.3	70493712.1
Cheese Taste	>25 days	n/a	3.2	95.0925567
Diabetes	214.1	1.332098236	1.6	1.332098236
Enrollment	>25 days	n/a	34.0	97879.0953
Extroversion Car	>25 days	n/a	41.8	415.7273293
House Prices	267.3	19540.35008	1.0	19540.35008
Nuclear Plants	122.9	10714.82356	0.2	10714.82356
Polishing Times	>25 days	n/a	269.0	1900.03938
Smoking Cancer	95.4	57.7651194	0.1	57.7651194
Temperatures	>25 days	n/a	196.8	145.6685991
Votes	589.8	91.1856555	18.6	91.1856555

1.7 Conclusions

This work explores global optimization of the clusterwise regression problem using mathematical programming and related issues. A mixed logical-quadratic programming (MLQP) formulation based on the logical implication of constraints is proposed for global optimization of the clusterwise regression problem and it is contrasted with a mixed integer quadratic formulation based on the traditional big-M formulation, which cannot guarantee optimality. Both quadratic formulations are evaluated empirically for optimization times and solution optimality, where the big-M formulation is shown to sometimes provide non-optimal and infeasible SSE solutions and non-optimal clustering. The mixed logical-quadratic programming (MLQP) formulation with implication of constraints formulation is observed to be numerically stable and guaranteed to provide exact global optimal solution solutions for the clusterwise regression problem.

Appendix 1.A – OPL model declarations

```
float M = ...;  // Large number for big-M constant
int  nO = ...;  // Number of observations
int  nD = ...;  // Number of dimensions
int  nK = ...;  // Number of clusters

range O = 1..nO;  // Range for observations
range D = 1..nD;  // Range for independent dimensions
range K = 1..nK;  // Range for clusters

float Y[O]   = ...;  // Dependent variables
float X[O,D] = ...;  // Independent variables

dvar float   C[D,K];  // Coefficients
dvar float   E[O,K];  // Errors
dvar boolean Z[O,K];  // Cluster assignments
```

Appendix 1.B – Real dataset descriptions

In the Acorn dataset, the size of the area (range) in which an oak specie is found is modeled based on the acorn size, the height of the tree and an indicator for the Atlantic region (Aizen & Patterson 1990). The next dataset considers only the weight of a car to predict car fuel economy in miles per gallon (Heavenrich, Murrell & Hellman 1991; Velleman 2010). The amount collected from New York City parking meters by contractors is modeled based on the number of months starting with May 1997, the amount collected by city worker from parking meters around city hall, the number of collection days in the month for the city worker and an indicator for Brink's as a contractor. This data was used to determine the amount of theft from parking meter collections by Brink's employees (Finkelstein & Levin 2001). Following this, small company CEO salaries in 1993 are modeled based on the age of the CEO (Velleman 2010). Subsequently, the percentages of yes votes by producers by county for the "checkoff" program that collects one dollar per cattle sold for funding programs to increase demand for the beef industry is modeled. The independent variables are; the average size of the farms in acres, the average value of products sold in thousands of dollars, the average percentage of products sold from livestock and the percent of farms with sales of 100,000$ or more (Velleman 2010). The next dataset considers the relationship between the subjective taste score for a cheddar cheese and the following predictor chemicals including relevant transformations; the natural log of the concentration of acetic acid, the natural log of the concentration of hydrogen sulfide and the concentration of lactic acid (Moore & McCabe 1998). Following this, the crime rates in 1960 in 47 US states is modeled based on the number of males per 1000 between 12 and 24 year old, the 1960 per capita expenditure for police forces, the unemployment rate of urban males per 1000 of 35 to 39 years old and the number of families per 1000 under the median income (Vandaele 1978; Hand 1994).

The next dataset examines the relationship between the independent variables age and base deficit with the C-peptide concentration (Sochett, Daneman, Clarson & Ehrlich 1987; Hastie & Tibshirani 1990). Fall undergraduate enrollment is modeled based on the year, the

unemployment percentage rate, the number of high school graduates and the per capita income (Velleman 2010). Following this, the time spent looking after one's car is predicted by the sex, age and extroversion score of the person (Miles & Shevlin 2001). The subsequent dataset examines the relationship between the size of a house as a predictor of its price (Miles & Shevlin 2001). The mercury concentrations in largemouth bass in 53 lakes is modeled based on the concentration of; alkalinity, pH, calcium, chlorophyll in the lake water (Lange, Royals & Connor 1993). After that, the cost in millions of dollars of a nuclear power plant is modeled based on the date that the permit was issued, the time between receiving the operating license and the construction permit and if the plant was constructed in the north-east region of the USA (Cox & Snell 1981). The next dataset is from Nambe Mills in New Mexico and it considers the relationship between polishing times and the diameter of the item in inches as predictors of its retail price (Velleman 2010). Subsequently, the public expenditures per capita per state is modeled based on the economic ability index, the percentage of the population living in metropolitan areas, the percentage growth of the population between from 1950 to 1960 and if it is a western state (Velleman 2010). Lung cancer deaths per 100,000 per state in 1960 is modeled based on the number of hundreds of cigarettes sold (Fraumeni 1968). Following this, the average minimum January temperature for 56 cities is predicted based on the city's latitude and longitude. Lastly, the percentage of votes for democrats in 1984 for 50 states is modeled based on their percentage of votes in 1980.

Appendix 1.C – Synthetic datasets

The first section contains the 50 observations for the random dataset series where the number of independent variables is varied from one to ten. The first three independent variables of the first section are also the first 50 observations for the series where the number of observations is varied. The second section contains the next 50 observations (51-100) for the series where number of observations is varied. The third section contains the x values for Line 1 and the 1.0 standard deviation perturbation applied to the y value and then rounded to two decimals, the fourth section contains the same details for Line 2. The 1.0 standard deviation perturbation is the basis for all other perturbation levels. A bias variable with a constant of one must be added to every dataset as x1.

Table VII Synthetic datasets

					Random varying dimensions and varying observations (first 50)							Random varying observations cont.					Line 1			Line 2	
o	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**	**x10**	**y**	***o***	**x2**	**x3**	**x4**	**y**	***o***	**x2**	**1 sd**	***o***	**x2**	**1 sd**
1	1.70	0.40	9.84	3.04	7.20	7.56	2.47	9.10	4.83	8.81	**51**	4.36	9.52	3.92	2.64	1	7.17	-0.21	51	1.36	-1.04
2	4.37	9.26	5.47	5.73	4.37	8.73	8.92	4.04	3.20	5.55	**52**	3.75	0.83	4.91	3.89	2	7.98	-0.50	52	7.74	-1.29
3	1.39	9.98	0.51	4.38	2.43	1.06	4.93	4.47	8.08	5.09	**53**	6.50	4.71	8.90	8.41	3	7.74	-1.29	53	2.06	0.42
4	6.12	1.09	6.80	0.84	0.65	0.35	7.47	0.77	6.40	7.24	**54**	1.04	2.06	2.56	4.38	4	9.41	0.60	54	1.53	0.56
5	9.46	9.36	0.40	8.21	0.84	7.84	7.72	9.11	4.36	2.09	**55**	3.82	1.86	0.97	6.78	5	7.86	-1.08	55	7.94	3.02
6	1.86	1.34	8.55	3.12	6.43	2.27	3.75	1.08	1.80	3.83	**56**	2.54	9.36	6.90	5.64	6	5.87	-0.23	56	0.36	0.04
7	9.34	3.83	1.13	0.33	8.46	4.36	4.13	1.55	9.80	9.33	**57**	3.03	9.98	0.43	4.08	7	9.18	-0.43	57	1.43	0.30
8	2.59	5.24	6.84	1.52	2.38	6.33	8.65	8.63	5.69	6.00	**58**	5.56	9.47	4.09	7.38	8	2.58	-0.70	58	7.12	-0.55
9	8.45	1.47	7.98	8.75	4.05	6.85	0.58	4.48	7.40	9.07	**59**	1.72	8.12	2.57	1.96	9	4.64	-1.01	59	0.16	-0.72
10	0.14	4.72	8.80	7.65	6.74	4.36	9.24	6.11	1.24	9.58	**60**	8.41	1.58	9.12	2.18	10	2.49	1.44	60	8.84	-0.86
11	2.55	7.00	6.75	5.87	5.61	5.45	2.19	8.57	1.50	7.79	**61**	9.93	7.48	2.25	2.37	11	8.14	1.22	61	9.85	-1.50
12	3.40	3.49	0.40	8.44	5.15	4.37	1.06	0.36	4.87	7.94	**62**	9.82	6.87	5.95	5.33	12	2.54	0.61	62	7.16	-1.00
13	4.95	6.01	7.58	8.20	5.56	4.79	2.02	5.12	3.16	5.12	**63**	9.90	7.85	9.61	7.00	13	4.67	-2.31	63	3.85	-0.65
14	8.78	5.55	6.08	5.52	3.92	8.66	2.94	5.66	7.27	8.12	**64**	2.19	7.00	3.94	6.36	14	3.06	-1.10	64	4.64	0.23
15	8.78	1.59	2.23	3.23	7.67	8.60	8.48	8.94	9.11	5.95	**65**	3.94	9.87	1.42	6.61	15	1.73	0.70	65	0.13	1.08
16	9.35	9.28	0.83	2.88	9.93	3.29	0.75	4.47	7.55	0.81	**66**	3.16	4.79	9.41	8.64	16	7.56	0.07	66	2.22	1.34
17	9.35	1.78	2.52	4.17	0.80	5.16	1.62	3.06	3.07	5.31	**67**	6.00	7.16	6.34	3.67	17	2.71	-0.68	67	9.77	-1.55
18	4.52	1.59	8.77	0.34	4.23	6.87	3.98	1.79	7.35	2.41	**68**	2.21	9.83	9.98	3.14	18	3.42	0.27	68	9.58	1.53
19	6.75	0.90	8.27	5.61	4.57	4.80	6.08	6.99	3.73	6.10	**69**	5.72	8.04	0.48	5.67	19	7.11	-1.31	69	1.11	2.42
20	4.32	9.77	1.62	8.90	1.13	2.12	2.58	1.73	9.74	9.66	**70**	6.86	5.16	9.08	4.13	20	4.22	0.44	70	2.74	0.46
21	5.55	9.23	7.52	2.42	2.61	4.19	3.19	4.53	7.32	6.32	**71**	0.70	0.29	4.58	6.71	21	9.85	1.21	71	4.43	1.17
22	9.63	4.84	8.69	7.34	6.65	2.04	0.31	2.34	6.03	0.37	**72**	8.87	6.82	7.27	5.07	22	6.36	-0.21	72	3.03	0.09
23	8.70	4.55	7.24	6.36	0.29	8.49	8.46	2.45	6.62	2.46	**73**	5.64	5.30	2.96	4.10	23	7.71	-0.05	73	0.37	0.88
24	4.67	7.33	8.75	5.32	6.76	0.90	7.47	0.61	8.42	2.11	**74**	0.99	1.58	2.26	0.12	24	6.55	-0.15	74	2.02	-0.06
25	0.12	7.35	2.40	9.67	8.73	8.87	1.61	3.10	4.88	9.22	**75**	3.84	1.19	1.75	9.25	25	0.07	-0.05	75	3.49	1.51
26	7.98	6.87	5.62	4.59	7.75	6.24	9.54	4.44	5.28	0.48	**76**	9.67	7.62	0.01	3.66	26	4.63	-1.80	76	1.53	-0.96
27	2.17	6.82	0.46	7.17	0.87	1.35	3.52	2.91	8.61	1.82	**77**	9.24	6.70	6.61	7.98	27	0.30	0.89	77	1.99	-0.46
28	7.63	4.69	3.69	7.64	1.04	2.35	7.70	6.45	7.78	4.54	**78**	5.00	8.49	5.38	8.21	28	4.62	0.11	78	6.57	0.43
29	3.41	3.22	7.19	0.23	8.67	9.38	1.95	4.79	5.00	3.95	**79**	9.78	7.54	4.25	5.44	29	9.34	0.97	79	8.95	-0.48
30	2.54	0.69	1.47	4.18	5.51	3.04	1.33	6.95	4.54	5.20	**80**	8.86	4.63	4.30	7.25	30	1.27	0.84	80	9.20	0.31
31	3.91	4.30	5.00	1.97	6.66	6.32	2.61	7.87	6.87	8.08	**81**	4.00	6.83	5.17	6.16	31	9.09	0.96	81	9.80	0.34
32	5.52	4.38	2.29	6.02	7.46	6.12	9.65	4.21	9.91	2.39	**82**	6.94	6.33	4.66	1.72	32	6.44	1.38	82	3.94	0.19
33	0.16	5.93	2.61	6.87	8.33	9.01	5.15	0.17	5.94	7.18	**83**	1.25	8.08	0.56	3.38	33	1.55	-0.38	83	6.78	2.24
34	5.50	2.48	6.80	1.53	9.15	7.86	0.56	7.01	4.66	2.09	**84**	1.27	9.09	8.12	6.74	34	2.14	-1.93	84	8.91	-0.34
35	6.61	3.38	6.05	5.67	9.59	0.62	9.75	7.00	4.55	8.83	**85**	9.58	1.04	7.16	0.62	35	7.07	-0.63	85	3.12	0.08
36	3.58	2.73	4.36	6.82	4.47	4.91	4.50	7.18	2.45	3.95	**86**	3.71	8.42	1.31	8.66	36	8.53	-1.10	86	8.86	-0.87
37	2.91	0.37	2.24	0.73	0.96	9.87	3.49	9.54	0.36	2.89	**87**	4.76	8.57	9.06	5.43	37	7.48	1.78	87	2.76	1.32
38	3.65	8.93	3.67	7.11	7.93	3.09	0.35	6.99	8.51	5.99	**88**	4.53	8.5	7.59	2.64	38	9.03	0.71	88	5.18	-0.46
39	6.59	7.07	4.44	0.74	4.67	0.52	1.27	5.58	5.08	3.47	**89**	2.42	4.38	1.21	5.91	39	2.39	0.23	89	3.64	-1.02
40	8.74	8.74	7.32	2.18	2.67	7.09	8.91	2.17	5.37	7.52	**90**	5.28	5.36	7.35	1.14	40	0.12	-0.03	90	7.85	-1.79
41	8.55	1.77	8.25	3.77	4.72	8.23	0.32	3.54	8.99	8.28	**91**	2.38	3.41	3.5	8.82	41	9.45	0.61	91	7.65	-0.55
42	8.74	4.07	2.49	7.79	6.56	1.63	0.29	10.00	1.92	1.28	**92**	0.42	6.45	0.6	7.01	42	7.35	-1.13	92	2.39	0.79
43	2.26	3.77	1.11	9.18	6.20	8.59	8.57	9.21	2.50	7.73	**93**	7.14	2.77	9.36	1.97	43	2.62	-0.16	93	7.96	0.92
44	1.89	8.22	9.05	5.61	3.65	8.47	0.72	8.31	1.26	8.99	**94**	1.74	4.05	4.13	3.5	44	0.46	0.32	94	7.46	-1.17
45	7.51	1.68	5.32	8.90	8.10	8.33	2.06	9.90	4.59	5.57	**95**	6.4	1.17	0.7	0.38	45	0.80	-1.71	95	5.68	2.13
46	3.85	9.64	9.89	3.20	7.95	0.95	9.29	3.74	5.30	6.31	**96**	1.94	1.81	7.72	7.04	46	0.49	-0.43	96	4.63	-0.14
47	7.85	4.32	6.15	6.34	9.01	7.91	1.55	3.31	8.59	7.22	**97**	7.99	3.35	4.98	3.57	47	4.33	0.27	97	8.65	1.94
48	5.15	6.30	4.25	1.42	8.85	4.99	2.48	2.18	1.63	7.33	**98**	7.21	5.88	8.47	3.29	48	0.23	1.22	98	6.61	-1.64
49	6.72	1.78	3.46	9.21	7.49	6.16	5.82	6.58	8.48	6.90	**99**	4.36	6.93	5.41	2.11	49	4.09	-0.11	99	0.69	1.94
50	7.61	0.97	2.31	9.69	5.40	5.89	9.31	0.10	6.89	2.92	**100**	3.49	1.77	1.93	8.07	50	9.47	-0.63	100	2.37	0.16

CHAPTER II:

Extensions to the repetitive branch and bound algorithm for globally optimal clusterwise regression

Réal A. Carbonneau

Gilles Caporossi

Pierre Hansen

Department of Management Sciences

GERAD and HEC Montréal

3000, chemin de la Côte-Sainte-Catherine

Montréal, Québec H3T 2A7, Canada

Abstract

A branch and bound strategy is proposed for solving the clusterwise regression problem, extending Brusco's repetitive branch and bound algorithm (RBBA). The resulting strategy relies upon iterative heuristic optimization, new ways of observation sequencing, and branch and bound optimization of a limited number of ending subsets. These three key features lead to significantly faster optimization of the complete set and the strategy has more general applications than only for clusterwise regression. Additionally, an efficient implementation of incremental calculations within the branch and bound search algorithm eliminates most of the redundant ones. Experiments using both real and synthetic data compare the various features of the proposed optimization algorithm and contrasts them against a benchmark mixed logical-quadratic programming formulation optimized by CPLEX. The results indicate that all components of the proposed algorithm provide significant improvements in processing times, and, when combined, generally provide the best performance, significantly outperforming CPLEX.

2.1 Introduction

Clusterwise regression is a clustering technique where multiple lines or hyperplanes are fit to mutually exclusive subsets of a dataset such that the sum of squared errors (SSE) from each observation to its cluster's line is minimized (Charles 1977; Diday 1979; Späth 1979). The term line will be used in this paper for both line and hyperplane. Clusterwise regression has relevance to such areas as spline estimation, utility function clustering and response based segmentations of customers, markets, regions, subjects, strategies or investors (Charles 1977; Diday 1979; Späth 1979; Lau, Leung & Tse 1999; Hennig 2000). Optimization for clusterwise regression is considered "a tough combinatorial optimization problem" (Lau, Leung & Tse 1999), and it appears that the only currently known feasible method for global optimization is

by mixed logical-quadratic programming (MLQP) (Carbonneau, Caporossi & Hansen 2011).

A new paradigm, the repetitive branch and bound algorithm (RBBA) has recently been proposed by Brusco and Stahl for clustering, seriation and variable selection (Brusco & Stahl 2005; Brusco 2006). It works by sequencing the data and optimizing by branch and bound (BB) a series of problems corresponding to ending subsets with one more observation at a time (Brusco 2006). An ending subset is a sequential subset of observations whose last observation is also the last observation of the complete set. Values of the solved problems are used to strengthen the lower bounds of the search at the current iteration. Building on this previous work, the present paper proposes an extended branch and bound strategy which combines iterative heuristic optimization, new ways of sequencing the observations, and branch and bound optimization of a limited number of ending subsets. These three key features lead to significantly faster optimization of the complete set and the strategy has more general applications than only for clusterwise regression.

Clusterwise regression is a cubic optimization problem defined by: the number of clusters (K), the number of independent dimensions (D), and the number of observations (O). The iterators are for: a cluster ($k \in \{1,...,K\}$), an independent dimension ($d \in \{1,...,D\}$), and an observation ($o \in \{1,...,O\}$). The model parameters are: the independent variable for an observation and dimension (x_{od}) and the dependent variable for an observation (y_o). The model variables are: the cluster assignment of an observation to a cluster (z_{ok}), the regression coefficient (aka β) for a dimension of a cluster (b_{dk}) and the error for an observation of a cluster (e_{ok}). The cubic model is as follows:

$$\text{SSE} = \min \sum_{k=1}^{K} \sum_{o=1}^{O} \left(z_{ok} e_{ok}^2 \right) \tag{2.1}$$

$$\text{s.t.} \quad \sum_{d=1}^{D} \left(b_{dk} x_{od} \right) + e_{ok} = y_o \quad \forall o, \forall k \tag{2.2}$$

$$\sum_{k=1}^{K} z_{ok} = 1 \quad \forall o \tag{2.3}$$

$$z_{ok} \in \{0,1\} \quad \forall o, \forall k \tag{2.4}$$

The objective (2.1) is the minimization of the sum over all clusters of the sum of squared errors (SSE) for their observations relative to their regression line. The constraint (2.2) fits the regression lines to the data by adjusting the coefficient and error terms. An observation can only be assigned to one cluster at a time (2.3) and the cluster assignment is binary (2.4). This formulation does not explicitly require an intercept but it can be included in the model simply by adding a variable to the data with a constant of one. All models in this paper include an intercept, thus D is always one more than the number of independent variables in the original dataset. Since there are K^O possible clustering configurations and since there is a minimum of D^2O regression computations (Gentleman 1973) to perform per clustering configuration, the enumeration of the complete problem search space requires at least $K^O D^2 O$ operations.

Although identifying the globally optimal solution to a clusterwise regression problem by no means guarantees identifying the true model, on average, these globally optimal solutions will lead to better models than random local optima identified by heuristics. However, as stressed by Brusco et al., clusterwise regression makes no effort to distinguish between error explained by clustering and error explained by regression (Brusco, Cradit, Steinley & Fox 2008). Also, since clusterwise regression fits multiple lines to the data, the overfitting potential is much greater than that of a single regression line. Consequently an evaluation procedure has been proposed to test if there is overfitting or not (Brusco, Cradit, Steinley & Fox 2008). Nevertheless, evaluating and addressing this overfitting problem is not in the scope of the current research and neither is the statistical validity of identified optimal clusterwise

regression models. This research considers only the feasibility and processing time for finding the optimal solution to a clusterwise multiple linear regression problem (OCMLR).

This paper is structured as follows: Section 2.2 provides an overview of some previous heuristic and exact optimization approaches, Section 2.3 details the proposed exact global optimization strategy, Section 2.4 overviews the experimental protocol and datasets, Section 2.5 presents the results and related discussion. Conclusions are presented in Section 3.6.

2.2 Previous optimization approaches

2.2.1 Heuristics

Various heuristics have been applied to solving the clusterwise regression problem. The exchange method, which is stepwise optimal but not globally optimal, consists in tentatively moving each observation from its cluster to each other cluster, keeping only the reassignments that reduce the error. This is repeated until a complete pass over the observations does not result in any improvement (Charles 1977; Diday 1979; Späth 1979; Späth 1981; Späth 1982). The simulated annealing (SA) (DeSarbo, Oliver & Rangaswamy 1989), variable neighborhood search (VNS) (Caporossi & Hansen 2005), and genetic search (GS) (Aurifeille 2000; Aurifeille & Medlin 2001; Aurifeille & Quester 2003) metaheuristics have provided frameworks to build heuristics for the clusterwise regression problem. Bayesian tree growing (Ciampi, Rich, Dyachenko, Villalobos, Murie & Nadon 2007), approximation (Mirkin 2005) and “hardening” the fractional cluster assignments (Aurifeille 2000) from expectation maximization (EM) (DeSarbo 1988; Wedel 1990; Wedel & DeSarbo 1995; Wedel 1998) have also been applied to the problem. A cubic mathematical programming approach has also been proposed, however the actual optimization is only a heuristic and it “may generate local optimum” (Lau, Leung & Tse 1999).

2.2.2 Mixed logical-quadratic programming

Mathematical programming optimization by state of the art solvers such as CPLEX is used as a benchmark against which to compare new optimization algorithms. The cubic clusterwise regression mathematical programming formulation can be re-formulated into a quadratic problem. However the big-M mixed integer quadratic programming (MIQP) formulation cannot guarantee globally optimal results (Carbonneau, Caporossi & Hansen 2011). This occurs because the slope of a line can be arbitrarily steep, and thus the error, can be arbitrarily large. Consequently, the big-M constant cannot be guaranteed to be large enough for any possible optimal solution nor for any possible intermediate solution of the MIQP branch and bound procedure. In an approach called mixed logical-linear programming (MLLP), or in this case, mixed logical-quadratic programming (MLQP), the logical propositions remain in their natural formulation while at the same time taking advantage of the strength of both logic processing and linear or quadratic programming (Hooker & Osorio 1999; Hooker, Ottosson, Thorsteinsson & Kim 2000; Hooker 2002; Hooker 2007). This leads to the following logical-quadratic reformulation of the cubic model:

$$\text{SSE} = \min \sum_{k=1}^{K} \sum_{o=1}^{O} \left(e_{ok}^2 \right) \tag{2.5}$$

$$\text{s.t.} \quad \left(z_{ok} = 1 \right) \Rightarrow \sum_{d=1}^{D} \left(b_{dk} x_{od} \right) + e_{ok} = y_o \quad \forall o, \forall k \tag{2.6}$$

$$\sum_{k=1}^{\min\{o,K\}} z_{ok} = 1 \quad \forall o \tag{2.7}$$

$$z_{ok} \in \{0,1\} \quad \forall o, \forall k \tag{2.8}$$

In the above MLQP formulation (Carbonneau, Caporossi & Hansen 2011), the regression constraint (2.2) is replaced by the logical implication ($\Rightarrow$) of a constraint, where the cluster assignment variable z_{ok} (constraint indicator) implies (activates) the regression error constraint (2.6). This removes the z_{ok} product from the cubic objective (2.1) and thus the objective is now simply the minimization of the sum of squared error (SSE) which is quadratic (2.5). Some symmetry breaking is provided by limiting some

cluster assignments (2.7) and the last constraint of the model is binary cluster assignment (2.8) as before (2.4).

2.3 Proposed exact global optimization strategy

A branch and bound algorithm can also be used for solving the clusterwise regression problem optimally. Although this is a difficult task, symmetry breaking, identifying stronger bounds and controlling the path through the search space can reduce the actual size of the search and incremental regression calculations will reduce the number of operations for each evaluation. The upper bound can be strengthened by heuristic optimization. The lower bound can be strengthened by exact global optimization of ending subsets as in RBBA (Brusco 2006). An appropriate observation sequence can strengthen the bounds and enhance fathoming. To ensure a precise description of the developed algorithm, simplified pseudocode will be presented which contains the most important parts and specific lines of pseudocode will be referenced with a letter followed by numbers such as (x.9.9).

2.3.1 Symmetry Breaking

When there are multiple clusters, a set of observations assigned to cluster 1 would still be the same set with the same error if those observations, and them only, were assigned to another cluster. Consequently, there are complete branches of the search tree which result in an equivalent clustering solution. Processing these branches more than once is a waste of computation time. To avoid these branches, as the search goes deeper in the tree, the number of permitted clusters is always only one more than the maximum number of clusters in the previous node (i.4.3). In other words, when a new cluster is added, it is important that all current clusters are used. For example, observation 1 is always assigned to cluster 1, then for observation 2 the algorithm will branch to assign it to clusters 1 and 2 only. However, for the branch where the observation 2 is assigned to cluster 1, observation 3 will only be tried in clusters 1 and 2, and not 3 as opposed to

the branch where observation 2 is assigned to cluster 2. Thus, this is a computation which has a linear complexity in respect to the tree nodes processed.

2.3.2 Identifying stronger bounds

During the exploration of the search tree, when the current node's error surpasses the best known complete solution error (an upper bound on the optimal solution value), the branch is pruned due the fact that adding an observation will never decrease the error. However, there are three additional improvements related to the bounds that can reduce the computational effort and processing time. i) A heuristic can be used to find good solutions and thus to obtain a stronger upper bounds (2.3.2.1). ii) Incremental ending subsets can be solved optimally to identify stronger lower bounds (2.3.2.2). iii) Sequencing the observations affects the search path and can enhance fathoming (2.3.2.3).

2.3.2.1 Heuristic Optimization

The estimation of stronger upper bounds can be performed with a variety of heuristics, such as those mentioned in section 3.2.1. However, because a good heuristic often finds an optimal or near-optimal solution in a small fraction of the time it takes to prove a globally optimal solution, there is no significant advantage in using a more complex heuristic. Consequently, the multistart exchange heuristic is employed in this research (Späth 1979). The heuristic starts by assigning every observation to a random cluster and computes the regression lines. Then each observation is reassigned to the cluster where it fits best, if there is an improvement. At the end of a pass over all observations, the regression lines are recalculated based on the new assignments. Once a pass has been executed without any observation reassignment, a local minimum has been achieved and the exchange method iteration terminates.

a. Function Heuristic()

Comment: Any good heuristic. Multistart exchange algorithm is a simple one for clustering

a.1. Set *TrialUpperBound*.*Clustering*[1…*O*] = Random(1,*K*)

a.2. Set *TrialUpperBound* = LocalOptimization(*TrialUpperBound*, 1)

b. Function LocalOptimization(*TrialUpperBound*, *StartingObservation*)

Comment: Exchange local optimization algorithm, returns an Upper Bound

b.1. Exchange Algorithm Loop: Repeat While *LocalImprovement*

b.1.1. Set *LocalImprovement* = False

b.1.2. Initialize *CalculationCache*

b.1.3. Observation Loop: For *o* = *StartingObservation* to *O* Step 1

b.1.3.1. Set *k* = *TrialUpperBound*.*Clustering*[*o*]

b.1.3.2. Set *CalculationCache*[*k*] = IncrementalCalculations(*Observation*[*o*], *CalculationCache*[*k*])

b.1.4. Observation Loop: For *o* = *StartingObservation* to *O* Step 1

b.1.4.1. Set *BestTrialError* = ∞

b.1.4.2. Set *BestTrialK* = 0

b.1.4.3. Cluster Loop: For *k* = 1 to *K* Step 1

b.1.4.3.1. Set *TrialError* = ErrorCalculations(*CalculationCache*[*k*])

b.1.4.3.1.1. If *TrialError* < *BestTrialError*

b.1.4.3.1.1.1. Set *BestTrialError* = *TrialError*

b.1.4.3.1.1.2. Set *BestTrialK* = *k*

b.1.4.4. If *BestTrialK* ≠ *k*

b.1.4.4.1. Set *TrialUpperBound*.*Clustering*[*o*] = *BestTrialK*

b.1.4.4.2. Set *LocalImprovement* = True

b.2. Set *TrialUpperBound*.*Error* = Sum(*CalculationCache*[1…*K*].*Error*)

b.3. Return *TrialUpperBound*

The main challenge is identifying in advance how much time should be spent on the heuristic, since the difficulty of the search and the difference in value between the current and optimal solution are not known in advance. For very structured problems, one iteration of the heuristic will identify the optimal solution. For other complex problems, it may take days or may never happen. At this point there is no automatic method to determine in advance how much time should be spent on the heuristic. Since the cost of identifying good solution by a heuristic is usually small and the payoff may

be large, it is important to perform many heuristic iterations to be safe. The proposed algorithm iterates between the exhaustive branch and bound search and the heuristic search, allocating a predefined ratio of time to each (f.3, f.4, g.3, g.4, i.4.1). The settings used in the optimizations performed for the current research specify that the heuristic is performed for approximately 10% of the processing time in slices of one second. This ratio of 10% is chosen because it provides a good compromise between time spent on the heuristic and the potential performance gains provided and insurance against mediocre local solutions which can adversely affect performance. Even if the optimal solutions can be found very rapidly with a heuristic before the start of the exhaustive search, no more than 10% of processing time is wasted on the heuristic, and it is worth this cost to have an insurance against not finding a good solution. The longer the global optimization takes because of poor heuristic solutions, the longer the heuristic has to improve the solution.

2.3.2.2 Ending Subsets

As detailed in the repetitive branch and bound algorithm (RBBA) (Brusco 2006), stronger lower bounds can be calculated using the optimal solutions of ending subsets. Additionally, when using incrementally larger ending subsets and performing the branch and bound search in the forward direction, the search can benefit recursively from the lower bounds. Note that in work predating RBBA on minimum sum of squares clustering (Brusco & Stahl 2005; Brusco 2006), ending subsets have also been used for computing lower bounds, but in a non-optimal and much less efficient way (Koontz, Narendra & Fukunaga 1975; Hand 1981; Hand 1981). Computations can be reduced while still strengthening the lower bounds by optimizing only a limited number of ending subsets and subset optimization can be assisted by estimating good subset solution from the best known complete set solution and vice-versa.

Both the sequence and direction of optimizing incrementally larger ending subsets are important to achieve the most benefits. The branch and bound algorithm for an individual ending subset must process the observations in the forward direction, which

permits the use of the lower bounds identified by previous smaller ending subset searches. However, the incrementally larger subsets always end at the last observation, thus the subsets are increased in size in the reverse direction, meaning the starting observation backs up, but the ending observation is always the last of the complete set.

Unlike suggested by the RBBA framework (Brusco 2006), we found that for clusterwise regression it is wasteful to process every subset because the more precise intermediate lower bounds are not worth the additional computational cost, especially when the subset size is very close to the complete set size. Thus, we propose a subset step size which leads to the optimization of only a small fraction of the ending subsets. The steps are identified from the first observation because it is important to have the number of observation between the largest ending subset and the complete set to be of the defined step size (f.2, g.1, g.2, h.1.2). For example, if the step size is 10 and there are 51 observations, the last subset should include observation 11 to 51 and not 2 to 51. The reason is that optimizing a subset for a lower bound so close to the start of the complete set is probably useless and is almost as time consuming as optimizing the complete set. Processing only a limited number of ending subsets is a strategy that may provide computational savings for WCSS (within-cluster sums of squares error) clustering and possibly other clustering problems.

Identifying an appropriate step size of the ending subsets is currently not a resolved issue. A step size of one (Brusco 2006) seems to be too small and causes the optimization of too many subsets for the small gain in more precise lower bounds. Once a lower bound is identified for an ending subset, it is also the lower bound for larger ending subsets, thus it is not necessary to optimize all subsets to have good bounds. However, since the problem search space is exponential with respect to the number of observations, it is beneficial to optimize many ending subsets. Preliminary tests indicated that a step size of 10 provided good results for a wide range of datasets, thus this setting was used for all experiments presented in this research. Future work may find benefits in estimating the structuredness of the problem based on good

heuristic solutions and setting the step size accordingly. For example, it seems that if the problem has a high level of structure, the step size should be larger.

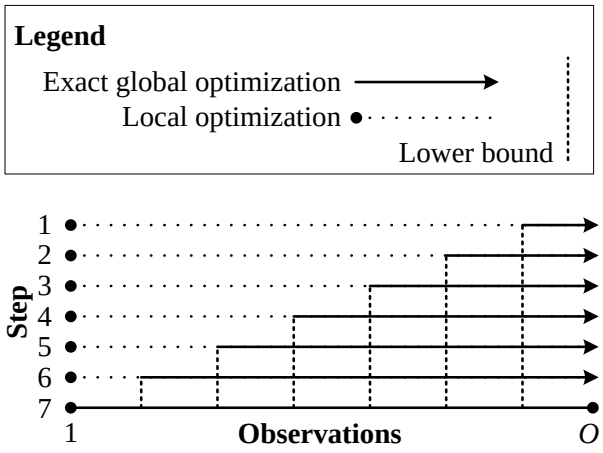

Figure 3 Overview of incremental ending subset optimization

Each time an ending subset optimization is performed, a good solution (upper bound) for this subset can be derived from the best known solution for the complete set, by using it as a reference and performing a local optimization (c). It may also be advantageous to perform repeated heuristic optimizations for the subset to identify a stronger upper bound for the subset itself. However, this has not been tested in the current research since it should usually be possible to estimate a good upper bound for the subset from the best known solution from the complete set. In addition, a good solution for the complete set can be derived from the optimal solution of each ending subset (d). Thus if the optimal solution is difficult to identify using a heuristic, it is possible that the globally optimal solution of a subset provides a better estimate of the global solution than that identified by the previous heuristic optimizations. It is interesting to note that when there is much structure in the dataset, the heuristic optimization is not even required since it only takes one local optimization from the first ending subset to identify the optimal solution.

c. Function HeuristicCStoSS(*o*):

Comment: Using the best known solution for the complete set, create a solution for the subset and try to improve it with a local optimization, returns an Upper Bound

c.1. Set *CurrentUpperBound* = LocalOptimization(*UpperBound*[*O*], *o*)

c.2. If *CurrentUpperBound.Error* < *UpperBound*[*o*].*Error*

c.2.1. Return *CurrentUpperBound*

c.3. Else

c.3.1. Return *UpperBound*(*o*)

d. Function HeuristicSStoCS(*o*):

Comment: Using the optimal solution for the subset, create a solution for complete set and refine it with a local optimization, returns an Upper Bound

d.1. Set *CurrentUpperBound.Clustering*[1…(*o*-1)] = Random(*1*,*K*)

d.2. Set *CurrentUpperBound.Clustering*[*o*…*O*] = *UpperBound*[*o*].*Clustering*[*o*…*O*]

d.3. Set *CurrentUpperBound* = LocalOptimization(*CurrentUpperBound*, 1)

d.4. If *CurrentUpperBound.Error* < *UpperBound*[1].*Error*

d.4.1. Return *CurrentUpperBound*

d.5. Else

d.5.1. Return *UpperBound*[1]

2.3.2.3 Observation Sequencing

Observation sequencing has been previously shown to improve processing times for various clustering problems (Brusco & Stahl 2005) including the clusterwise means / WCSS (minimum within-cluster sums of squares error based partitioning) problem (Brusco 2006). However, these approaches to sequencing are not general and cannot be applied to clusterwise regression in any way since the error or distance of an observation from its own cluster and other clusters cannot be estimated without first knowing the solution (the set of lines). Thus, we propose a more general strategy of using a heuristic to identify a good solution and then sequencing the observations based on metrics with respect to this solution.

Although an algorithm may be deterministic and even exact, thus always finding the global optimum, the path taken through the search space may vary greatly simply as a result of the sequence in which the data points are processed. Extreme variations in

processing times can occur in branch and bound algorithms. When randomizing the sequence of observations for the current clusterwise regression problem, computing times of the simple branch and bound algorithm for 65 random observations and 4 independent variables ranged between 10 seconds and 4 hours with an average of 12 minutes across 100 trials (Figure 5). A similar distribution in processing times occurs (0.7 seconds to 9 hours) with the branch and bound algorithm when the lower bounds are made stronger by incremental ending subset optimization (Figure 5). From these and other results, it seems that the processing time follows a log-normal distribution. The distribution of processing times indicates that there are even better and worse processing times that could be identified at the extremes of the distribution. In fact, the number of possible observation sequences is all permutations, thus the factorial of the number of observations. Even CPLEX experiences the same extreme range of processing times because the algorithms for optimizing mixed logical-quadratic problems also rely on a branch and bound search. For example, CPLEX optimized the synthetic dataset of 300 observations generated from 2 lines and a 0.1 standard deviation perturbation, with processing times ranging from 68 seconds to 15 hours.

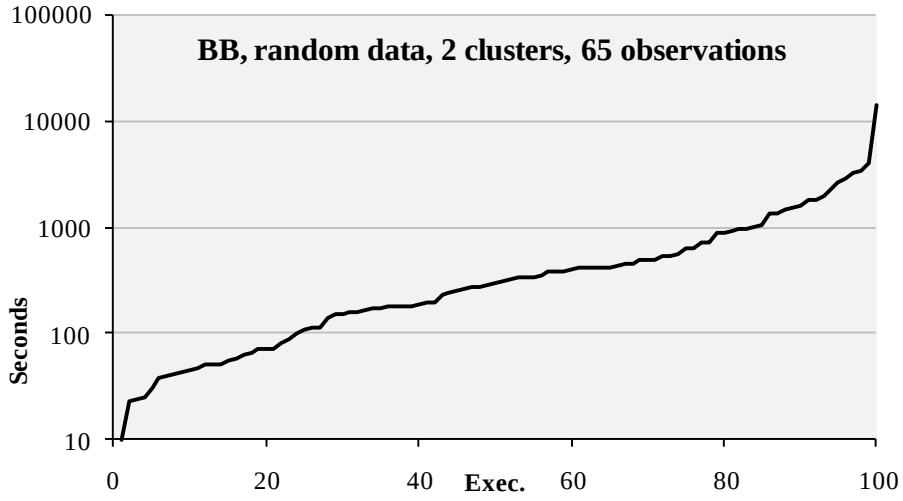

Figure 4 Cumul. distr. of proc. times for BB optim., random data, 2 clusters, 65 obs.

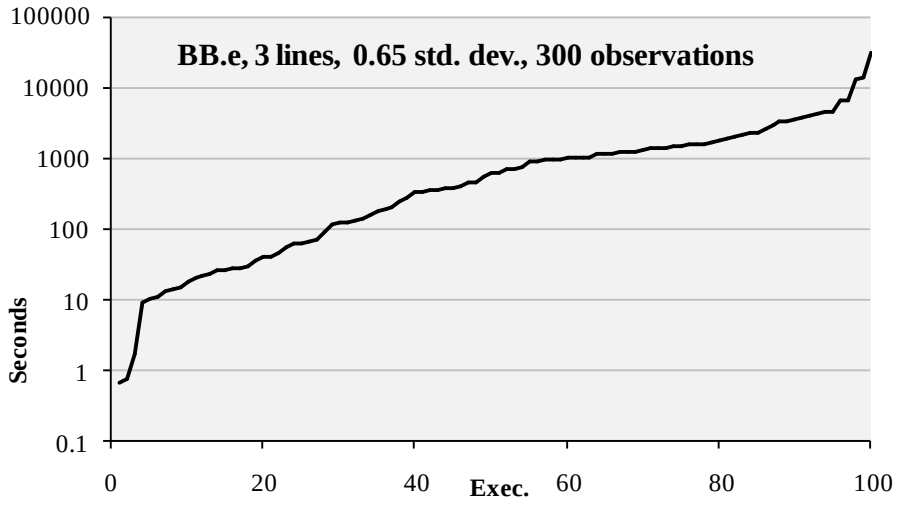

Figure 5 Cumul. distr. of proc. times for BB optim. with ending subsets, 3 lines, 0.65 SD, 300 obs.

These observations indicate that there is a wide range of beneficial and detrimental sequences of data points for the branch and bound algorithm. If a good sequencing strategy can be achieved, it should be possible to at least avoid the extremely poor sequences at the end of the distribution. Ideally, a sequencing strategy could be identified which will provide processing times at the low end of the distribution. Considering a dataset which has a high level of structure, a detrimental sequence of observations is one where the observations are in sequence of their optimal cluster assignments. While branching on the observations at the beginning of the sequence, since they all fit into one cluster with a low amount of error they will also fit into *K* clusters with an even lower error. Thus, once the sequence of observations in the first cluster has been branched on (into *K* clusters), there is already a large number of terminal nodes which will lead to further branching. This should be avoided by having a path that quickly prunes off branches of the search tree where observations in the same cluster are not together. A simple approach is to sequence the observations by

alternating the clusters from which they are drawn based on a known good solution. For this and the following sequencing strategies, such a good solution is required and thus the iterative heuristic optimization presented above is necessary. Ideally the unproven global optimal solution is identified before sequencing.

Combining the sequencing with the ending subset searches involves a balance between a sequence that results in fast optimizing of the complete set and one which is not disruptive to the ending subset searches while at the same time providing strong lower bounds. It may seem that sequencing the observations in descending order of the error against all other clusters would quickly increase the error as the search depth increases, thus enabling a rapid pruning of the search tree. There are two problems with this. The first is that it is detrimental to the incremental ending subset optimization because all of the observations at the end of the sequence have the lowest errors from all clusters, thus they are the most difficult to separate. The second is that if there is structure in the data, the observations with the largest error may all come from one cluster or may also be all relatively clustered together in the observation sequence, thus the path through the search space will result in much larger processing times.

A better strategy is to sequence the observations in descending order of error from their cluster identified by heuristic optimization, while forcing alternating clusters (e). Thus, the first observation in the sequence will be the one with the largest error of those assigned to cluster 1, the next from cluster 2, etc, up to cluster *K*. Subsequently, the observations with the next largest error will follow. When the branch and bound algorithm processes this sequence of observations, there will rapidly be a representative sample in each cluster and these observations will increase the error faster than an average random sequence. In addition, incremental ending subset optimization will also have a representative sample of observations, with lower error, thus their processing times will be a fraction of that of the complete set.

e. Function SequenceObservations(*UpperBound*)

Comment: Sequence observations by descending error within cluster with forced alternating cluster

e.1. Observation Loop: For o = 1 to O Step 1

e.1.1. Set k = *UpperBound.Clustering*[o]

e.1.2. *CalculationCache*[k] = IncrementalCalculations(*Observation*[o], *CalculationCache*[k])

e.2. Observation Loop: For o = 1 to O Step 1

e.2.1. Set k = *UpperBound.Clustering*[o]

e.2.2. *Observation*[o].*Cluster* = k

e.2.3. *Observation*[o].*Error* = ErrorCalculations(*CalculationCache*[k])

e.3. Sort *Observation* by Ascending *Cluster* then Descending *Error*

e.4. Set *PreviousK* = 0

e.5. Observation Loop: For o = 1 to O Step 1

e.5.1. Set k = *Observation*[o].*Cluster*

e.5.2. If $k \neq$ *PreviousK*

e.5.2.1. Set *Sequence* = 1

e.5.3. *Observation*[o].*Sequence* = *Sequence*

e.5.4. Set *Sequence* = *Sequence* + 1

e.5.5. Set *PreviousK* = k

e.6. Sort *Observation* by Ascending *Sequence* then Ascending *Cluster*

However, a problem with regression is that the line can change significantly when adding observations. While exploring the search tree, all combinations of observations are tried within the current bounds, which can cause large fluctuations in the regression line. This is even more pronounced when the data is random and thus has very little structure. The regression line is most sensitive to observations at its extremes, thus, by adding extreme observations from each dimension, the regression line can be stabilized faster. Consequently, another strategy is to add extreme observations from alternating dimensions, with forcing alternating dimensions and clusters. Since this can lead to an observation being eligible to be included at multiple places in the sequence, as soon as it is eligible, it is sequenced and not considered again. In terms of error, compared to the previous sequencing by descending known error, this is a sequencing strategy by descending potential error of paths in the search space.

This sequencing causes the incremental ending subsets optimizations to be more difficult to solve since ending observations provide less stability to the regression lines because all the observations are closer to the center. Thus the processing times for the complete set are usually much faster than the slowest subset and a fraction of the total processing time. In addition, this sequencing strategy is very specific to clusterwise regression and may not be applicable to other types of branch and bound problems as opposed to the sequencing based on descending error with forced alternating clusters.

As a last point, when considering the wide range of processing times for the branch and bound algorithms, it is clear that comparisons of processing time must only be made in the statistical sense, with average over multiple randomized sequences of observations. Considering the distribution with extreme values at either tail, a larger number of trials is important since these extreme values have a significant impact on the averages. Without averages from an adequate number of trials, comparisons of the processing time of such algorithms have limited relevance. Considering that the research process involves artificial selection, analysis without appropriate averages will favor a research path towards algorithms with the largest variations.

The reduction of the computational complexity offered by pruning the search tree can only be determined empirically since it will depend on the structure of the problem created by the data. A clusterwise regression dataset that is completely random will present much bounding and fathoming opportunities compared to one with more structure.

2.3.3 Incremental Calculations

The complete clusterwise regression search problem has a K^O clustering configuration. For each of these search tree nodes, there are K regression equations to evaluate and as the number of observations increases, the regression equations itself becomes more complicated because there are more observations in each regression equation to process. The computations for each regression equation is related to the dimensions and clusters, such that on average it is $D^2 \cdot O/K$ per regression because the observations

are spread across the clusters, accordingly, the total computational complexity is $K^O \cdot D^2 \cdot O$. However, the regression can be computed incrementally every time an observation is added to a cluster so that the same calculation isn't performed again deeper in the search tree. Gentleman (1973; 1974) presents the currently best known algorithm for incrementally calculating the linear least squares (algorithm AS 75) in Algol. Additional algorithms (algorithm AS 274) were also developed for calculating the correlation, covariance, subset coefficients, among others (Miller 1992; Miller 1994) in Fortran 77. The IncrementalCalculations (e.1.2) and the ErrorCalculations (e.2.3) functions are not detailed in this paper since they derive directly from above mentioned previous work on regression.

The incremental regression calculations for any subset can be efficiently performed directly in the tree search structure by keeping an incremental cache (a storage of all the required data to resume the calculations at an intermediary step) of the regression calculations for each cluster at each observation level of the search tree and always copying it forward as the algorithm goes deeper in the search tree (i.4.4.1). Processing only the incremental adjustment to the regression computation cache for the target cluster, the computational complexity is reduced by O/K for each cluster's regression, thus reducing the total computational complexity to $K^O \cdot D^2$.

2.3.4 Combining the components of the algorithm

Having detailed the individual components of the strategy, it is now presented as an integrated algorithm. The major components are: heuristic optimization (2.3.2.1), observation sequencing (2.3.2.2), incremental ending subset optimization (2.3.2.3) and final global optimization. Thus, it can be called a branch and bound algorithm which exploits heuristic, sequencing and ending subsets (BBHSE). The algorithm parameters are set (f), where two of them help avoid processing subsets uselessly (f.1, f.2) and the other two manage the time slices (f.3, f.4). Subsequently, the various initializations are performed (g) and the main loop (h.1) controls observation sequencing and restarting the algorithm when an improvement is identified. The branch and bound loop

implements the exhaustive search while respecting the lower and upper bound and giving processing time slices to the heuristic optimization (i). It also efficiently manages the incremental calculation cache to avoid most redundant calculations (i.4.4.1).

f. Parameters:

f.1. Set *MinimumNumberObservations* = (*D*)*K*+(*K*-1)

f.2. Set *StepSize* = 10

f.3. Set *TimeSlice* = 1

f.4. Set *HeuristicTimeRatio* = 0.1

g. Initialization:

g.1. Set Global *StartSteps* = Max(Round(*O* / *StepSize*) * *StepSize*, *MinimumNumberObservations*)

g.2. Set Global *EndSteps* = Max(*StepSize*, *MinimumNumberObservations*)

g.3. Set Global *HeuristicTimeSlice* = *TimeSlice* * *HeuristicTimeRatio*

g.4. Set Global *BBTimeSlice* = *TimeSlice* * (1 - *HeuristicTimeRatio*)

g.5. Set Global *Improvement* = True

g.6. Set Global *LowerBound*[1…*O*] = 0

g.7. Set Global *UpperBound*[1…*O*].*Error* = ∞

h. Main

h.1. Main Loop: Repeat While *Improvement*

h.1.1. Set *Improvement* = False

h.1.2. Ending Sub Set Loop: For *o* = *StartSteps* to *EndSteps* Step *StepSize*

h.1.2.1. Set *UpperBound*[*o*] = HeuristicCStoSS(*o*)

h.1.2.2. If *UpperBound*[*o*].*Error* < *LowerBound*[*o*] and not *Improvement*

h.1.2.2.1. BBSearchAndHeuristic(*o*)

h.1.2.2.2. If not *Improvement*

h.1.2.2.2.1. Set *LowerBound*[1…*o*] = *UpperBound*[*o*].*Error*

h.1.2.2.2.2. Set *UpperBound*[1] = HeuristicSStoCS(*o*)

h.1.3. If not *Improvement* and *LowerBound*[1] < *UpperBound*[1]

h.1.3.1. BBSearchAndHeuristic(1)

h.1.4. If *Improvement*

h.1.4.1. SequenceObservations(*UpperBound*[1])

h.1.4.2. Set *LowerBound*[2…*O*] = 0

h.1.4.3. Set *UpperBound*[2…*O*].Error = ∞

h.2. Comment: Exact global optimization complete (*Improvement* = False)

i. Function BBSearchAndHeuristic(*StartingObservation*)

Comment: Iterate between exhaustive branch and bound search and heuristic search

i.1. Set *o* = *StartingObservation*

i.2. Set *Clustering*[0…*O*] = 0

i.3. Initialize *CalculationCache*

i.4. Branch and Bound Loop: Repeat Until *Improvement*

i.4.1. Every *BBTimeSlice* seconds for *HeuristicTimeSlice* seconds or Until *Improvement*

i.4.1.1. Set *TrialUpperBound* = Heuristic()

i.4.1.2. If *TrialUpperBound.Error* < *UpperBound*[1].*Error*

i.4.1.2.1. Set *UpperBound*[1] = *TrialUpperBound*

i.4.1.2.2. Set *Improvement* = True

i.4.2. Set k, *Clustering*[*o*] = *Clustering*[*o*] + 1

i.4.3. If *k* > *K or k* > *Clustering*[*o-1*] + 1 or *o* > *O*

i.4.3.1. Set *o* = *o - 1*

i.4.4. Else

i.4.4.1. Set *CalculationCache*[*o*,*k*] = *CalculationCache*[*o-1*,*k*]

i.4.4.2. Set *CalculationCache*[*o*,*k*] = IncrementalCalculations(*Observation*[o], *CalculationCache*[*o*,*k*])

i.4.4.3. If *o* = *O* and *CalculationCache*[*o*,*k*].*Error* < *UpperBound*[*o*].*Error*

i.4.4.3.1. Set *UpperBound*[*o*].*Error* = Sum(*CalculationCache*[*o*,1…*k*].*Error*)

i.4.4.3.2. Set *UpperBound*[*o*].*Clustering* = *Clustering*

i.4.4.3.3. If *StartingObservation* = 1

i.4.4.3.3.1. Set *Improvement* = True

i.4.4.4. If *CalculationCache*[*o*,*k*].*Error* + *LowerBound*[*o*].*Error* < *UpperBound*[*o*].*Error*

i.4.4.4.1. Set *o* = *o* + 1

This algorithm controls what seem to be the three most important aspects of a branch and bound algorithm in a way that does not require settings based on *a priori* information about the problem. The heuristic strengthens the upper bound and provides a good solution for sequencing metrics. The sequencing attempts to control the path the search takes through the solution space. Optimization of the incrementally larger ending subsets strengthens the lower bounds available for pruning the search and also provides additional starting points for local optimization by the heuristic.

2.3.5 Generalization

The main parts of the presented algorithm (c, d, e, g, h, i) are general to any clustering problem for which an incremental calculation can be defined. Although there may exist specialized efficient heuristics for specific problems, there also exists heuristics, such as tabu search (TS), genetic search (GS), variable neighborhood search (VNS), simulated annealing (SA), that can minimize any function, thus is it assumed that the heuristic parts (a, b) of the presented algorithm can also be generalized to most clustering problems. On the other hand, the incremental and error calculations are specific to each clustering problem. These may also require problem specific optimizations since these calculations are the innermost loop of the complete algorithm, and thus can have a large impact on the algorithm's performance. For applications to other branch and bound problems, the change to the objective function would be considered instead of the problem specific term error used in this article.

2.4 Experimental Protocol

As detailed in the section on observation sequencing, because of large variations in processing times, it is important to always use appropriate statistics when comparing the processing times of various algorithms. Consequently, for all of the optimization experiments, the same problem was executed 100 times and the sequence of observations was randomized each time. Once the total processing time for a specific problem and algorithm had passed beyond 100 hours, it was aborted and considered timed out with an average of over one hour for solving the problem. Since only the observation order is randomized and no other data is changed, all global optimization algorithms will achieve the same global minimum. To supplement the averages presented, the standard deviation of the processing times for the last problem in a series dataset is presented at the end of a cluster section of the tables. The impact of the following aspects of the optimization on the processing time will be examined: the optimization algorithms and their relevant components, the number of clusters, the

number of observations, the number of independent dimensions, and the perturbation of the data points around a line.

From the total search space dimension of the order $K^O \cdot D^2 \cdot O$, it could be expected that the computation times increases first with the number of clusters, then with the number of observations and lastly with the number of dimensions. It is also expected that the computation times will increase with the amount of perturbation since there will be less structure from which to quickly reduce the search space. In addition to comparing the presented algorithms, all relevant component of the proposed algorithm were tested and compared to better understand what exactly is providing performance gains and how these components interact.

2.4.1 Datasets

Twelve real datasets and four series of synthetic datasets are used as a basis for the performance analysis of the clusterwise regression algorithms (Table VIII). These sets include data from parking meter collections from a Brink's court case, car fuel economy, CEO salaries, crime rates, electricity consumption, mercury in largemouth bass, iris flowers, polishing times, and temperatures. The series of synthetic datasets include random data with an increasing number of observations, an increasing number of dimensions and two lines and three lines with increasing perturbations.

Table VIII Dataset overview

Dataset	*O*	*D*	Independent Variables	Dependent
Brink's (Finkelstein & Levin 2001)	47	5	Months, City Worker Collect., Brink, Collection Days	Contractor Collection
Car Fuel Economy (Heavenrich, Murrell & Hellman 1991; Velleman 2010)	82	6	Volume, Horsepower, Top Speed, Weight	MPG
CEO Salaries (Velleman 2010)	59	2	Age	Salary
Check Off (Velleman 2010)	56	7	Big Farm, Principal Income, Size, Value, Live, Sales	Yes
Crime Rates (Vandaele 1978; Hand 1994)	47	14	Age, South, Educ., 1959 Expend., 1960 Expend., Labor Force Part., Male Ratio, Population, Non-white Ratio, 14-24 Unempl., 35-39 Unempl., Income	Crime Rate
Electricity (McCormick 1993; Lau, Leung & Tse 1999)	50	4	Elect. Price, Inc., Gas Price	Electricity
Mercury Bass (Lange, Royals & Connor 1993)	53	5	Alkal., pH, Calc., Chlorophyll.	Avg Mercury
Iris 2 clusters (Fisher 1936)	150	2	Sepal Length	Petal Length
Iris 3 clusters (Fisher 1936)	150	2	Sepal Width	Petal Width
Polishing Times (Velleman 2010)	59	7	Bowl, Casserole, Dish, Tray, Diameter, Polishing Time	Price
Temperatures (Peixoto 1990; Hand 1994)	56	3	Lat, Long	Jan. Temp
Random (uniform distribution), varying *O*	20-340	4	Random	Random
Random (uniform distribution), varying *D*	100	2-10	Random	Random
2 Lines, varying normal distribution perturbation	300	2	Random (uniform distribution)	y + Perturb.
3 Lines, varying normal distribution perturbation	300	2	Random (uniform distribution)	y + Perturb.

From Fisher's iris (Fisher 1936) datasets, two regression datasets which can be plotted in two dimensions are created (Figure 6). The first set was created from the sepal length predicting the petal length for the purpose of clustering into two lines, the second set was created from the sepal width predicting the petal width for the purpose of clustering into three lines.

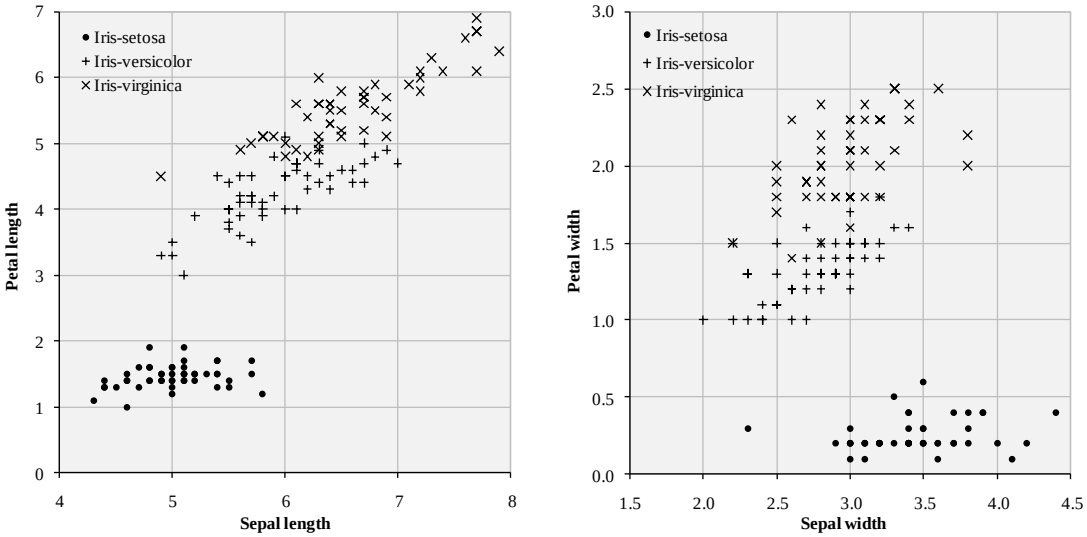

Figure 6 Iris data, 2 cluster petal length based on sepal length and 3 cluster petal width based on sepal width

The series of random datasets come from a uniform distribution. There are four independent dimensions and one dependent variable, which are all randomly generated up to the required number of observations and dimensions and are rounded to two decimal places. The dataset is predefined so that all algorithms processing a specified number of observations are solved exactly the same problem, although with different randomized sequences. The completely random sets is important because it is a difficult type of dataset for clusterwise regression since it has less structure and thus provides less potential for reducing the search space. These sets permit the exploration of the effect of an increasing number of observations on the optimization times of the various algorithms. Similarly, a series with a varying number of independent dimensions and 100 observations is generated to explore the effects on the optimization times of the various algorithms.

The last two datasets are composed of 300 observations synthetically created from 2 and 3 lines and from those lines, secondary datasets are generated with increasing amounts of perturbation from the normal distribution (Figure 7). The first line has a slope of -0.356 and an intercept of 7.583, the second line has a slope of 0.952 and an intercept of 0.909 and the third line has a slope of 0.255 and an intercept of -1.132 (line 1: $y = -0.356x + 7.583$, line 2: $y = 0.952x + 0.909$, line 3: $y = 0.255x – 1.132$). The independent variables (x) are random (uniform distribution) and rounded to two decimal places and the dependent variables (y) are perturbed (normal distribution) by an increasing amount. These sets test the algorithms for optimization time trends which may more closely represents the performance that would be expected for a dataset which results from two or three separate linear phenomena with some normally distributed error. Using these datasets, the effects of increasing perturbations can be examined and it is expected that the more perturbation, the more difficult it is to estimate good solutions and the lower bounds will be weaker, thus the search will be less bounded since the problem has less structure.

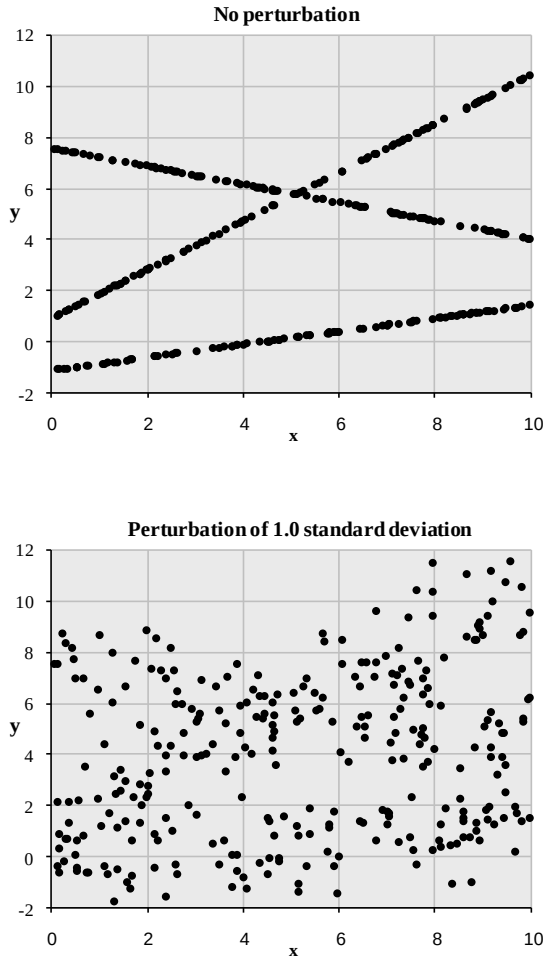

Figure 7 Plots of the three lines dataset without perturbations and of one standard deviation

2.4.2 Implementation

In this paper, the mixed logical-quadratic programming model is optimized using the IBM ILOG OPL-CPLEX (OPL 6.3 and CPLEX 12.1.0) environment, thus the model is implemented using the OPL programming language (Van Hentenryck, Lustig, Michel & Puget 1999; IBM 2009; IBM 2009). The CPLEX precision parameters are set to the maximum available of 9 significant digits. The relevant tolerances parameters are reduced as much as possible in an attempt to ensure optimal and high precision solutions. The node file flag is set to store nodes on disk (with compression) when the in-memory set is larger than 128 MB.

The branch and bound algorithms developed for this research used symmetry breaking, iterative heuristic optimization, observation sequencing, ending subset optimization (BBHSE) and incremental regression calculations. When the ending subset component was employed, the step size was always set to 10 for the current set of experiments since it is a good balance between the additional time required for optimizing subsets and the benefit of stronger lower bounds. The CPLEX and BB algorithms are identified in the results tables and the branch and bound algorithm and its options are identified as BB with periods separating the options (Table IX).

Table IX Algorithm codes and descriptions

Code	Description
CPLEX	Mixed logical-quadratic formulation optimized by CPLEX
BB…	Branch and bound optimization
…h…	Iterative heuristic optimization
…s1…	Sequencing by descending error in cluster with forced alternating of clusters
…s2…	Sequencing by extreme observations from alternating dimensions with forced alternating of clusters
…e	Incremental ending subsets optimization with a step size of 10

The branch and bound algorithm and all related components and computations are implemented using the C programming language with a precision of over 9 significant digits for comparisons (Knuth 1997) and of over 15 significant digits for calculations. The program is compiled using gcc version 4.1.2 with the code optimization option "-O3". All optimizations are performed on an Intel Core 2 Quad Q8200 CPUs at 2.33GHz with 4 MB of shared L2 cache and 4 GB of RAM running Linux. Each optimization is limited to one CPU core, thus there is no parallel processing.

2.5 Results and discussions

The results of optimizing the real datasets into two and three clusters are presented in Table XVII, which indicate that the BBHSE algorithm provides a significant

performance advantage over CPLEX and simpler branch and bound algorithms (BB and BB.h). However, the results also indicate that the heuristic and sequencing alone (BB.h.s1, BB.h.s2) often do as well or even slightly better than adding the ending subset searches (BB.h.s1.e, BB.h.s2.e). In addition, the more general sequencing strategy (s1) provides better overall performance because it is more robust in avoiding the relatively poor performances of the regression specific sequencing (s2) in some cases, such as the Polishing Times dataset. The ending subset alone (BB.e) does not perform as well as those with heuristic and sequencing, presumably because of a low amount of structure in these real datasets.

Table X Statistics for the real datasets, two clusters, 100 randomized sequences

Dataset	*K*	BB.h.s1.e Avg.	BB.h.s2.e Avg.	BB.e Avg.	BB.h.s1 Avg.	BB.h.s2 Avg.	BB.h Avg.	BB Avg.	CPLEX Avg.	SSE
Brinks	**2**	0.7	10.5	1063.2	0.7	9.2	2864.4	>1h	1719.2	382570036510
Car Fuel Economy	**2**	5.9	203.0	>1h	10.3	176.7	>1h	>1h	>1h	171.63417904
CEO Salaries	**2**	0.1	0.1	0.0	0.1	0.2	43.8	1228.3	27.0	768960.33590
Check Off	**2**	172.9	64.5	>1h	185.0	55.8	>1h	>1h	>1h	1045.5425945
Crime Rates	**2**	846.1	933.6	1391.8	658.0	743.9	832.7	1203.2	>1h	226.50481753
Electricity	**2**	1.4	4.4	9.2	1.8	5.8	31.0	39.0	715.3	284.52809432
Iris 2 clusters	**2**	33.2	1847.5	>1h	27.3	>1h	>1h	>1h	>1h	22.534468566
Mercury Bass	**2**	0.2	0.1	0.5	0.2	0.1	6.1	55.1	246.6	0.55046169311
Polishing Times	**2**	19.9	>1h	>1h	15.8	>1h	>1h	>1h	>1h	2678.6813064
Temperatures	**2**	0.1	0.7	50.8	0.1	0.2	1127.4	2224.2	750.4	428.38653329
Brinks	**3**	628.1	248.1	>1h	600.3	202.6	>1h	>1h	614.3	54683555662
CEO Salaries	**3**	0.6	27.0	>1h	0.9	79.2	>1h	>1h	>1h	415596.94438
Electricity	**3**	1216.5	>1h	>1h	1188.9	>1h	>1h	>1h	>1h	93.006906325
Iris 3 clusters	**3**	47.6	8.4	>1h	1734.6	3311.5	>1h	>1h	>1h	3.1389468806
Temperatures	**3**	1.5	5.9	>1h	1.4	1.7	>1h	>1h	>1h	145.66859914

The results for clustering a few of the real datasets into three clusters are presented in Table XVII. The simpler branch and bound algorithms (BB and BB.h) and the ending subset searches alone (BB.e) cannot identify the optimal solution for three clusters for any of the dataset under an average of one hour. Of all the experiments, clustering the Brink's dataset into three clusters is the only one where CPLEX

outperformed the BBHSE algorithm (BB.h.s1.e) and was close to the heuristic and sequencing only version (BB.h.s1).

2.5.1 Random Dataset

The results for clustering the random dataset series into two and three clusters while varying the number of observations are presented in Table XI. As with the real datasets, the BBHSE algorithm provided significant performance advantages over CPLEX and the simpler branch and bound algorithms (BB and BB.h). However, in contrast to the results from the real datasets, the ending subsets and combining heuristics, sequencing and ending subsets enabled the optimization of significantly larger datasets than heuristic and sequencing only. Surprisingly, for the algorithms using the ending subset searches alone (BB.h.e) and combined with sequencing based on extreme observations (BB.h.s2.e), the processing time did not increase continuously as observations were added, it actually fluctuated up and down. This is partially a result of the fact that new observations can significantly alter the optimal solution when the data is random. Such significant changes in the solution also result in significant changes in the sequences of observations and thus the path of the optimization through the search space and subsequently the performance. However, for the ending subset optimization only version which is non-sequencing (BB.e), it is less clear why the processing times did simply continue increasing when additional random observations were added, especially considering that the presented times are average over 100 randomizations of the observation sequences.

Table XI Statistics for random dataset, varying observations, average seconds for 100 randomized sequences

Obs.	*K*	BB.h.s1.e Avg.	BB.h.s2.e Avg.	BB.e Avg.	BB.h.s1 Avg.	BB.h.s2 Avg.	BB.h Avg.	BB Avg.	CPLEX Avg.	Optimal SSE
20	**2**	0.1	0.1	0.0	0.1	0.1	0.1	0.0	1.0	9.1338991580
40	**2**	0.2	0.1	0.2	0.2	0.1	0.3	0.6	46.0	62.109924028
60	**2**	0.8	0.2	0.8	1.4	0.9	28.3	249.1	640.2	82.781290364
80	**2**	1.3	0.6	50.5	12.5	1013.7	> 1h	> 1h	> 1h	125.33978946
100	**2**	1.4	0.5	24.3	113.7	> 1h				151.56607965
160	**2**	1657.1	9.9	3669.3	> 1h					272.59430605
220	**2**	> 1h	21.3	505.1						387.12297509
280	**2**		103.9	1008.7						497.06733881
340	**2**		2301.4	> 1h						613.08514437
StDv	n/a	2.6%	8.1%	371.1%	9.0%	1.8%	452.3%	302.2%	58.1%	n/a
20	**3**	0.1	0.1	0.0	0.1	0.1	0.1	0.0	1072.0	0.15554606658
30	**3**	0.9	0.3	1.7	0.9	0.3	1.3	1.9	>1h	6.9104936100
40	**3**	9.3	4.0	52.5	9.7	3.9	38.0	108.8		16.680686349
50	**3**	916.0	19.7	1308.0	1134.8	12.8	1508.7	2531.5		23.666949463
60	**3**	1345.0	50.8	>1h	2509.9	96.3	>1h	>1h		27.553166232
70	**3**	>1h	417.9		>1h	1426.1				36.854434281
StDv	n/a	10.2%	21.4%	124.9%	10.4%	18.1%	275.8%	174.4%	40.8%	n/a

Interestingly, when comparing the two sequencing strategies without ending subset searches (BB.h.s1, BB.h.s2) the one by descending cluster error outperformed sequencing by extreme observations. However, when combined with ending subset searches, it is the sequencing by extreme observations (BB.h.s2.e) which provided the fastest performance. Indeed, when combined with the sequencing by descending cluster error (BB.h.s1.e), the performance was actually worse than the ending subset searches alone (BB.e). It appears that this sequencing disrupts the ending subset searches (BB.h.s1.e), whereas the ending subset searches algorithm (BB.e) provided an improvement over the sequencing alone (BB.h.s1). It is likely that this is a property of such random datasets for the clusterwise regression problem, since if it was only an artifact, it would not have been repeated as consistently while many random observations were added.

When clustering the random dataset series into 3 clusters, both algorithms based on the regression specific sequencing (…s2…) outperformed all other algorithms and

CPLEX was by far the slowest. Surprisingly, the ending subset (BB.e) strategy fell to about the same performance as the simpler branch and bound with heuristic (BB.h) algorithm.The results on the random dataset indicate that if for some reason optimal clusterwise regression solutions are required for random datasets, it would seem that the BBHSE algorithm with sequencing by extreme observations would provide the best performance.

The results for varying the number of dimensions for 100 observations are presented in Table XII. CPLEX and simpler branch and bound algorithms (BB and BB.h) could not optimize the datasets, and the heuristic and sequencing strategies (BB.h.s1 and BB.h.s2) did not perform as well as those with stronger lower bounds from ending subset optimization (…e). As with the other random dataset series, the combined algorithm with the regression specific sequencing (BB.h.s2.e) outperformed all others.

Table XII Statistics for random dataset, 100 observations, 2 clusters, varying dimensions, average seconds for 100 randomized sequences

Ind. Var.	BB.h.s1.e Avg.	BB.h.s2.e Avg.	BB.e Avg.	BB.h.s1 Avg.	BB.h.s2 Avg.	BB.h Avg.	BB Avg.	CPLEX Avg.	Optimal SSE
2	0.2	0.1	0.0	8.0	175.0	> 1h	> 1h	> 1h	155.71455483
3	0.2	0.1	1.2	15.6	738.4				151.75723049
4	1.4	0.5	26.4	114.1	> 1h				151.56607965
5	43.6	20.8	1396.9	920.9					150.76228181
6	309.0	71.5	3504.8	> 1h					136.81182038
7	> 1h	1522.6	> 1h	> 1h					128.93763613
StDv	7.4%2	8.5%	259.5%	8.7%	5.9%	n/a	n/a	n/a	n/a

2.5.2 Two and Three Lines with Perturbation

The next experiments examined the behavior of the various algorithms on the series of datasets from two and three lines with perturbations. The processing time statistics for clusterwise regression on the synthetic three line dataset are presented in Table XIII for clustering the first two lines into two clusters and the three lines into three clusters. These results indicate that both CPLEX and the simpler branch and bound algorithms (BB and BB.h) were relatively inefficient at solving the problem, even though

sometimes the optimization was quite fast depending on the random sequence of the observations. For all of the algorithms which included either the heuristic optimization or ending subset optimization (and thus local optimization from optimal solutions of subsets), the unproven global optimum was identified very rapidly when the perturbations amount is low because of the high level of structure.

Combining the starting heuristic and observation sequencing by descending within cluster error (BB.h.s1) provided better and more stable processing times which permitted optimizing up to a perturbation level of 0.55 standard deviations. Combining the heuristic with the sequencing based on extreme observations (BB.h.s2) resulted in optimizing up to a perturbation level of 0.60 standard deviations under an average of one hour. Both of these improved performances were a result of the strong upper bound and a good and stable path through the search space.

Table XIII Statistics for 2 lines dataset, 300 observations, varying normal distribution perturbation, average seconds for 100 randomized sequences

Std. Dev.	*K*	BB.h.s1.e Avg.	BB.h.s2.e Avg.	BB.e Avg.	BB.h.s1 Avg.	BB.h.s2 Avg.	BB.h Avg.	BB Avg.	CPLEX Avg.	Optimal SSE
0.00	**2**	0.1	0.1	0.0	0.1	0.1	0.1	> 1h	146.8	0.0025009204738
0.10	**2**	0.1	0.1	0.0	0.1	0.1	2.7		2147.6	3.2665102368
0.20	**2**	0.1	0.1	0.0	0.1	0.1	> 1h		> 1h	12.335947506
0.30	**2**	0.1	0.1	0.3	0.2	0.1				27.455510479
0.40	**2**	0.1	0.2	1.2	4.8	0.2				48.926938261
0.50	**2**	0.2	1.2	115.7	155.9	5.3				76.470119913
0.60	**2**	0.6	15.4	> 1h	> 1h	369.3				109.22501160
0.70	**2**	2.2	1140.0			> 1h				147.70999194
0.80	**2**	4.7	> 1h							190.08004822
0.90	**2**	20.4								237.10335939
1.00	**2**	1056.3								288.47061601
StDv	n/a	2.4%	20.6%	735.1%	2.8%	7.9%	109.2%	n/a	392.1%	n/a
0.00	**3**	0.1	0.1	0.0	0.1	0.1	0.1	> 1h	> 1h	0.0026426623973
0.10	**3**	0.1	0.1	0.0	0.1	0.1	0.2			3.0860096710
0.20	**3**	0.1	0.1	0.0	0.1	0.2	39.6			11.549167409
0.30	**3**	0.1	0.1	0.1	0.2	27.5	> 1h			25.829580541
0.40	**3**	0.1	0.4	0.4	2.7	> 1h				46.237962733
0.50	**3**	0.2	4.0	18.9	181.0					72.557256342
0.60	**3**	1.3	124.7	1088.8	> 1h					103.91878126
0.70	**3**	7.2	1607.3	> 1h						140.01306220
0.80	**3**	237.1	> 1h							179.53596560
StDv	n/a	7.0%	41.2%	529.8%	1.9%	1.9%	116.7%	n/a	n/a	n/a

The ending subset search only (BB.e) optimized up to 0.55 standard deviations of perturbations in part because the local optimization from the optimal solution of the first ending subsets rapidly identified the optimal solution for the complete set and thus the strongest upper bound. In addition, the ending subset searches also identified lower bounds for incrementally larger subsets, thus providing additional strong bounds to the search space. However, the paths it took through the search space depends heavily on the sequence of the observations, thus the processing times for this algorithm (BB.e) vary significantly.

Interestingly enough, the combination of the components that performed the best so far (BB.e and BB.h.s2) provide improved performance, but did not provide the best overall performance. The heuristic with sequencing based on extreme observations and ending subset searches (BB.h.s2.e) optimized up to a perturbation level of 0.70

standard deviations, which is better than all of the previously mentioned algorithms. However, it is significantly outperformed by the heuristic with sequencing by descending within cluster error and ending subset searches (BB.h.s1.e) which optimized up to a perturbation level of 1.00 standard deviations. In addition, the processing times for this algorithm were relatively stable. Even at high levels of perturbation, the sequencing based on extreme observations with ending subsets (BB.h.s2.e) still did not perform as well as on the completely random dataset, even though there is only one independent variable in this dataset as compare to three in the random dataset. Thus, there must be something particular about the interaction between a completely random dataset, its total lack of structure and the sequencing based on extreme observations with ending subsets (BB.h.s2.e).

The performance times for the three line problem were similar to those of the above two line problem (Table XIII). The CPLEX and simple branch and bound (BB) algorithm could not optimize the problem within an average of one hour, even when there is no perturbations. The branch and bound with starting heuristic (BB.h) optimized up to a perturbation of 0.25 standard deviations which, surprisingly, was actually more than with the previous two cluster problem. The sequencing based on extreme observations (BB.h.s2) had significantly higher processing times with three clusters as compared to two clusters and performed much worse than the sequencing by descending within cluster error (BB.h.s1) which had similar processing times to the two cluster problem. The ending subset searches (BB.e) surprisingly performed significantly better on the three cluster problem than the two cluster problem and better than the previously mentioned algorithms. As with the two cluster problem, the complete combination of heuristic, sequencing and ending subset optimization (BB.h.s1.e and BB.h.s2.e) performed better than all other algorithms. The complete BBHSE with sequencing by descending within cluster error (BB.h.s1.e) performed the best overall, optimizing up to a perturbation of 0.80 standard deviations under an average of one hour with stable processing times. The BBHSE with sequencing based

on extreme observations (BB.h.s2.e) did not perform as good, optimizing up to a perturbation of 0.75 standard deviations.

As seen with the other three cluster optimization experiments, the processing times increased significantly when the number of clusters was increased, however, the effect seemed to be a smaller when the problem had more structure. In both cases, this increase in complexity as a result of the increase in the number of clusters may be addresses by a column generation formulation.

These last two series of synthetic datasets should represent the type of problems that clusterwise regression may be applied to. Thus, based on the above results and those of the real datasets, the combination of a heuristic, sequencing of observations by descending within cluster error and ending subset optimizations (BB.h.s1.e) provides significant performance gains over all other tested algorithms for datasets that are expected to have some structure.

An overview of the ranking of the various algorithms by dataset are presented in Table XIV where it is clear that the proposed combined strategy outperforms all others and that when the data is random, extremities based sequencing (BB.h.s2.e) should be used and in all other case, sequencing by cluster error (BB.h.s1.e) should be employed

Table XIV Summary ranking of algorithms by dataset category

Dataset category	***K***	**BB.h.s1.e**	**BB.h.s2.e**	**BB.e**	**BB.h.s1**	**BB.h.s2**	**BB.h**	**BB**	**CPLEX**
Real datasets	2	1	3	5	2	4	6	7	8
Real datasets	3	1	3	>1h	2	4	>1h	>1h	5
Random datasets, varying obs.	2	3	1	2	4	5	6	7	8
Random datasets, varying obs.	3	3	1	6	4	2	5	7	8
Random datasets, varying dim.	2	2	1	3	4	5	>1h	>1h	>1h
2 Lines datasets	2	1	2	4	5	3	6	>1h	7
3 Lines datasets	3	1	2	3	4	5	6	>1h	>1h

In general, the combination of heuristic, sequencing and ending subset searches resulted in relatively stable processing times as can be seen by the standard deviations of the processing times in the previous tables. The first reason is that for most of the presented problems, the heuristic rapidly identifies the optimal solution and

subsequently the observations are sequenced based on this optimal solution. Thus, the actual search sequence is relatively similar for multiple randomized initial sequences. Sometimes, if either the optimal solution is not identified or if there are identical sequencing metrics, the sequence of observations may be different for different initial observation sequences, thus resulting in different processing times. Even though the effect of not achieving the globally optimal solution by the iterative heuristic optimization may result in higher processing times, it does occur that the suboptimal solution results in a better sequence of observations and that the optimization is actually faster. This is because although the proposed sequencing strategies do perform much better on average than a random sequence, they are not the optimal sequence in terms of processing time, there are better sequences, however, the strategy to achieve them is currently unknown.

2.6 Conclusion

The results indicate that the proposed combined heuristic optimization, observation sequencing and global optimization of ending subsets (BBHSE) strategy provides significant performance advantages over all currently available alternatives. The choice of the observation sequencing has major impact on performance and two algorithms are proposed. The first and more general rule is to sequence the observations by descending error in the cluster with forced alternating of clusters. The second rule, more specific to clusterwise regression, is to sequence the extreme observations of each dimension with forced alternating of clusters.

Considering the log-normal distribution of processing times, it is obvious that there exists better observation sequences, however, the challenge is to identify a strategy to provide this sequence in advance. Although the sequencing by extreme observations (BB.h.s2.e) provides exceptional performance for random datasets, sequencing by error in the cluster (BB.h.s1.e) provides better overall performance for datasets which have some structure, which should be the most common use. Although optimizing a random dataset may have limited usefulness, the fact that such exceptional performance was

achieved by a specific sequencing strategy is important since it identifies that there exists sequencing strategies which can reliably increase optimization performance for specific types of problems and datasets. However, considering both its conceptual generality and empirical performance, sequencing by descending error in cluster with forced alternating of clusters (...s1...) is the recommended sequencing algorithm *apriori* for clusterwise regression and possibly other clustering problems.

The results also indicate that overall, the combination of all three components provide the best performance, thus all three are required. Combining the sequencing with the ending subset searches involves a balance between a fast sequence for optimizing the complete set and one which is not disruptive to the ending subset searches while at the same time providing strong lower bounds. The BBHSE algorithm may also be useful for optimizing other problems which can be formulated in terms of a branch and bound search since each component can be defined in general terms of such a search. Based on the current research, the recommended global optimization strategy for clusterwise regression and possibly other clustering problems is the BB.h.s1.e algorithm.

Lastly, the results of this research also indicate the importance of comparing only averages of multiple branch and bound optimizations with randomized sequencing since the processing times can vary significantly with the sequences in which the tree is searched. Comparing only a single result for a particular sequence in which the tree is searched is questionable as the processing times follow a log-normal distribution.

CHAPTER III:

Extensions to column generation for globally optimal clusterwise regression

Réal A. Carbonneau

Gilles Caporossi

Pierre Hansen

Department of Management Sciences

GERAD and HEC Montréal

3000, chemin de la Côte-Sainte-Catherine

Montréal, Québec H3T 2A7, Canada

Abstract

A column generation based approach is proposed for solving the clusterwise regression problem. The proposed strategy relies firstly on several efficient heuristic strategies to insert columns into the restricted master problem. If these heuristics fail to identify an improving column, an exhaustive search is performed starting with incrementally larger ending subsets, all the while iteratively performing heuristic optimization to ensure a proper balance of exact and heuristic optimization. Additionally, observations are sequenced by their dual variables and by their inclusion in joint pair branching rules. The proposed strategy is shown to outperform the best known alternative (BBHSE) when the number of clusters is greater than three. Additionally, the current work further demonstrates and expands the successful use of the new paradigm of using incrementally larger ending subsets to strengthen the lower bounds of a branch and bound search as pioneered by Brusco's Repetitive Branch and Bound Algorithm (RBBA).

3.1 Introduction

Clusterwise regression is a clustering technique where multiple lines or hyperplanes are fit to mutually exclusive subsets of a dataset such that the sum of squared errors (SSE) from each observation to its cluster's line is minimized (Charles 1977; Diday 1979; Späth 1979). The term line will be used in this paper for both line and hyperplane. Clusterwise regression has relevance to such areas as spline estimation, utility function clustering and response based segmentations of customers, markets, regions, subjects, strategies or investors (Charles 1977; Diday 1979; Späth 1979; Lau, Leung & Tse 1999; Hennig 2000). Optimization for clusterwise regression is considered "a tough combinatorial optimization problem" (Lau, Leung & Tse 1999).

Previous work on the clusterwise regression problem has defined a mixed logical-quadratic formulation (Carbonneau, Caporossi & Hansen 2011) that permits its global optimization. Subsequently, a much more efficient method based on branch and bound optimization with iterative heuristics, observation sequencing and ending subset

optimization (BBHSE) has been developed (Carbonneau, Caporossi & Hansen 2011). This branch and bound optimization is an extension of a new paradigm proposed by Brusco, called the Repetitive Branch and Bound algorithm (RBBA) (Brusco & Stahl 2005; Brusco 2006). Since the branch and bound approach works well for a small number of clusters but slows down considerably as the number of clusters increases, this current work attempts to develop an efficient column generation approach which relies heavily on heuristics, observation sequencing and ending subset optimization. Thus, part of the previous work on BBHSE global optimization for the clusterwise regression problem will be transposed into the subproblem of the column generation approach.

Clusterwise regression is a cubic optimization problem defined by: the number of clusters (K), the number of independent dimensions (D), and the number of observations (O). The iterators are for: a cluster ($k \in \{1,...,K\}$), an independent dimension ($d \in \{1,...,D\}$), and an observation ($o \in \{1,...,O\}$). The model parameters are: the independent variable for an observation and dimension (x_{od}) and the dependent variable for an observation (y_o). The model variables are: the assignment of an observation to a cluster (z_{ok}), the regression coefficient (aka β) for a dimension of a cluster (b_{ok}) and the error for an observation of a cluster (e_{ok}). The cubic model is as follows:

$$\text{SSE} = \min \sum_{k=1}^{K} \sum_{o=1}^{O} \left(z_{ok} e_{ok}^2 \right) \tag{3.1}$$

$$\text{s.t.} \quad \sum_{d=1}^{D} \left(b_{dk} x_{od} \right) + e_{ok} = y_o \quad \forall o, \forall k \tag{3.2}$$

$$\sum_{k=1}^{K} \left(z_{ok} \right) = 1 \quad \forall o \tag{3.3}$$

$$z_{ok} \in \{0,1\} \quad \forall o, \forall k \tag{3.4}$$

The objective (3.1) is the minimization of the sum over all clusters of the sum of squared errors (SSE) for their observations relative to their regression line. The constraint (3.2) fits the regression lines to the data by adjusting the coefficient and error terms. An observation can only be assigned to one cluster at a time (3.3) and the cluster assignment is binary (3.4).

This formulation does not explicitly require an intercept but it can be included in the model simply by adding a variable to the data with a constant of one. All models in this paper include an intercept, thus D is always one more than the number of independent variables in the original dataset. Since there are K^O possible clustering configurations and since there is a minimum of D^2O regression computations (Gentleman 1973) to perform per clustering configuration, the enumeration of the complete problem search space requires at least K^OD^2O operations.

Although identifying the globally optimal solution to a clusterwise regression problem by no means guarantees identifying the true model, on average, these solutions will lead to better models than random local optima identified by heuristics. However, as stressed by Brusco et al., clusterwise regression makes no effort to distinguish between error explained by clustering and error explained by regression (Brusco, Cradit, Steinley & Fox 2008). Also, since clusterwise regression fits multiple lines to the data, the overfitting potential is much greater than that of a single regression line. Consequently an evaluation procedure has been proposed to test if there is overfitting or not (Brusco, Cradit, Steinley & Fox 2008). Nevertheless, evaluating and addressing this overfitting problem is not in the scope of the current research and neither is the statistical validity of identified optimal clusterwise regression models. This research considers only the feasibility and processing time for finding the optimal solution to a clusterwise multiple linear regression problem (OCMLR).

This paper is structured as follows: Section 3.2 provides an overview of some previous heuristic and exact optimization approaches, Section 3.3 details the proposed exact global optimization strategy, Section 3.4 overviews the experimental protocol and datasets, Section 3.5 presents the results and related discussion and the conclusions are presented in Section 3.6.

3.2 Previous optimization approaches

3.2.1 Heuristics

Various heuristics have been applied to solving the clusterwise regression problem. The exchange method, which is stepwise optimal but not globally optimal, consists in tentatively moving each observation from its cluster to each other cluster, keeping only the reassignments that reduce the error. This is repeated until a complete pass over the observations does not result in any improvement (Charles 1977; Diday 1979; Späth 1979; Späth 1981; Späth 1982). The simulated annealing (SA) (DeSarbo, Oliver & Rangaswamy 1989), variable neighborhood search (VNS) (Caporossi & Hansen 2005), and genetic search (GS) (Aurifeille 2000; Aurifeille & Medlin 2001; Aurifeille & Quester 2003) metaheuristics have provided frameworks to build heuristics for the clusterwise regression problem. Bayesian tree growing (Ciampi, Rich, Dyachenko, Villalobos, Murie & Nadon 2007), approximation (Mirkin 2005) and "hardening" the fractional cluster assignments (Aurifeille 2000) from expectation maximization (EM) (DeSarbo 1988; Wedel 1990; Wedel & DeSarbo 1995; Wedel 1998) have also been applied to the problem. A cubic mathematical programming approach has also been proposed, however the actual optimization is only a heuristic and it "may generate local optimum" (Lau, Leung & Tse 1999).

3.2.2 Mixed logical-quadratic programming

Mathematical programming optimization is used as a benchmark against which to compare new optimization algorithms. The cubic clusterwise regression mathematical programming formulation can be re-formulated into a quadratic problem, however the big-M mixed integer quadratic programming (MIQP) formulation cannot guarantee globally optimal results (Carbonneau, Caporossi & Hansen 2011). In an approach called mixed logical-linear programming (MLLP), or in this case, mixed logical-quadratic programming (MLQP), the logical propositions remain in their natural formulation while at the same time taking advantage of the strength of both logic processing and linear or quadratic programming

(Hooker & Osorio 1999; Hooker, Ottosson, Thorsteinsson & Kim 2000; Hooker 2002; Hooker 2007). This leads to the following logical-quadratic reformulation of the cubic model (Carbonneau, Caporossi & Hansen 2011):

$$\text{SSE} = \min \sum_{k=1}^{K} \sum_{o=1}^{O} \left(e_{ok}^2 \right) \tag{3.5}$$

$$\text{s.t.} \quad \left(z_{ok} = 1 \right) \Rightarrow \sum_{d=1}^{D} \left(b_{dk} x_{od} \right) + e_{ok} = y_o \quad \forall o, \forall k \tag{3.6}$$

$$\sum_{k=1}^{\min\{o,K\}} z_{ok} = 1 \quad \forall o \tag{3.7}$$

$$z_{ok} \in \{0,1\} \quad \forall o, \forall k \tag{3.8}$$

In the above MLQP formulation, the regression constraint (3.2) is replaced by the logical implication ($\Rightarrow$) of a constraint, where the cluster assignment variable z_{ok} implies (activates) the regression error constraint (3.6). This removes the z_{ok} product from the cubic objective (3.1) and thus the objective is now simply the minimization of the sum of squared error (SSE) which is quadratic (3.5). Some symmetry breaking is provided by limiting some cluster assignments (3.7) and the last constraint of the model is binary cluster assignment (3.8) as before (3.4).

3.2.3 Branch and bound with heuristics, sequencing and ending subset optimization

At present, the fastest known way to find the globally optimal solution for clusterwise regression problems is by branch and bound optimization with iterative heuristics, observation sequencing and optimization of incrementally larger ending subsets (BBHSE) (Carbonneau, Caporossi & Hansen 2011). The iterative heuristic optimization rapidly finds a very good and often globally optimal upper bound, the observation sequencing permits enhanced fathoming. Optimization of incrementally larger ending subsets strengthens the lower bounds of the search. This is a new paradigm in optimization proposed by Brusco with his Repetitive Branch and Bound Algorithm (RBBA) (Brusco 2006). This approach works well for problems with a few clusters but quickly becomes intractable as the number

of clusters increases. The BBHSE algorithm for clusterwise regression will also be used as a benchmark in this paper.

3.3 Column generation

Column generation (Dantzig & Wolfe 1960; Barnhart, Johnson, Nemhauser, Savelsbergh & Vance 1998; Desaulniers, Desrosiers & Solomon 2005; Lübbecke & Desrosiers 2005) is a technique of linear programming designed to solve problems with an exponential number of columns, which cannot all be enumerated in advance. It proceeds by alternating between a restricted master problem which contains only a few columns, and a subproblem which consists in finding an additional improving column. The simplex algorithm is subsequently applied to the restricted master problem and the procedure iterated until it is shown that the master problem is contradictory or no more improving columns can be found by the subproblem. Column generation can be combined with linear and non-linear integer programming as well as with non-linear programming (Aloise, Cafieri, Caporossi, Hansen, Perron & Liberti 2010; Aloise, Hansen & Liberti 2010; Hansen & Meyer 2011).

The original compact clusterwise regression problem can be formulated as an extended column generation problem. The restricted master problem is a binary set partitioning problem and the subproblem attempts to identify columns that have negative reduced cost. If the globally optimal solution to the subproblem is not negative, the master problem has been proven globally optimal.

3.3.1 Master problem

The restricted master problem is a linear program which cannot be considered as a mixed integer program since the dual variables (duals) are required to generate new columns or to prove optimality of its solution. Thus special procedures are required to ensure integer solutions. To describe the restricted master problem, the number of columns (C), the column iterator ($c \in \{1,...,C\}$), the column's sum of squared errors (SSE) (a_c), the selection of a column for the set partitioning (s_c) and the assignment of an observation to a column (z_{oc}) are now introduced. A column corresponds to a cluster and its respective line. The

objective is the minimization of the sum of the errors of each selected column (3.9). The column selection variable s_c is bounded between 0 and 1 inclusively (3.12). In addition, the column selection variable s_c must be continuous because the duals of each row is required. Consequently, fractional column selection is possible thus requiring a procedure to force the solution to an integer one. The Ryan and Foster branch and bound pairing search (Ryan & Foster 1981) is employed to eliminate fractional solution in a way that permits identification of the globally optimal integer solution, thus this column generation approach is also called Branch and Price (Barnhart, Johnson, Nemhauser, Savelsbergh & Vance 1998). When under a Ryan and Foster pairing search branch, every column in the master problem must respect the current pairing search rules. The selection of columns is limited to the specified number of clusters (3.10) and an observation can only be assigned to one column (3.11).

$$\text{SSE} = \min \sum_{c=1}^{C} (s_c a_c) \tag{3.9}$$

$$\text{s.t.} \quad \sum_{c=1}^{C} (s_c) = K \tag{3.10}$$

$$\sum_{c=1}^{C} (s_c z_{oc}) = 1 \quad \forall o \tag{3.11}$$

$$0 \le s_c \le 1 \quad \forall c \tag{3.12}$$

Each time the master linear problem is solved, a dual for each row is available for solving the subproblem. A dual indicates the improvement to the objective given a one unit change to the corresponding constraint. In the current problem, the dual of the column selection variable is usually a penalty because each column usually has a cost associated with it. The duals of the constraints that limit each observation to one class may be a reward or a penalty depending on the current state of the master problem.

3.3.2 Subproblem

The subproblem is a single regression problem that identifies a set of observations that minimizes the reduced cost. More specifically, in the current clusterwise regression

problem, the subproblem minimizes the Reduced Sum of Squared Errors (RSSE). If the globally optimal solution to this subproblem is negative, the master problem is not globally optimal yet and the column should be added to the restricted master problem. The dual (λ_r) of a row in the restricted master problem and the assignment of an observation to the current columns (z_o) are now presented to formulate the subproblem. The objective (3.13) is to minimize the Sum of Squared Errors (SSE) minus the dual of the cluster count constraint (3.10). The regression constraint (3.14) is activated (implied) when the observation is selected for the regression, it includes the subtraction of the dual of the one cluster assignment constraint and forces a change in observation's error variable when the observation does not fit the line perfectly. As shown in previous work, a big-M Mixed Integer Quadratic Programming (MIQP) formulation cannot guarantee globally optimal solutions to the clusterwise regression problem, thus a Mixed-Logical Quadratic Programming (MLQP) formulation is required (Carbonneau, Caporossi & Hansen 2011). The final constraint of the subproblem forces the column assignment variable to be binary (3.15).

$$\text{RSSE} = \min \sum_{o=1}^{O} \left(e_o^2\right) - \lambda_1 - \sum_{o=1}^{O} \left(z_o \lambda_{o+1}\right) \tag{3.13}$$

$$\text{s.t.} \quad \left(z_o = 1\right) \Rightarrow \sum_{d=1}^{D} \left(b_d x_{od}\right) + e_o = y_o \quad \forall o \tag{3.14}$$

$$z_o \in \{0,1\} \quad \forall o \tag{3.15}$$

The subproblem permits identification of columns that should be inserted into the restricted master problem, and when the subproblem's globally optimal solution is not negative, it proves that the master problem has been solved to its global optimum. It will usually be much more efficient to solve the subproblem using heuristics to identify good columns to insert into the restricted master problem and only solve the subproblem to its global optimum when the heuristics can no longer identify good solutions.

3.3.3 Ryan and Foster pairing search

When a non-integer solution is identified as the globally optimal solution to the master problem, a branch and bound procedure takes place. It is usually done following the Ryan and Foster rule (Ryan & Foster 1981). Branching is performed on a pair of fractional variables (column selections S_k , S_j) closest to 0.5 (the furthest from being integer) which have a pair of observations (O_j , O_k) that are joint (1,1) in one column and disjoint (1,0 or 0,1) in the other. In the joint pair branch, the observation are forced to be equal ($O_j = O_k$: 1,1 or 0,0) and in the disjoint pair branch they are forced to be complementary or both excluded from the column ($O_j = O_k$ or $O_j = O_k = 0$: 1,0 or 0,1 or 0,0). Consequently, the selected pair of columns cannot satisfy both the joint and disjoint pairing rules, thus forcing the solution towards an integer one. Additionally, selecting the columns which are the furthest from being integer (closest to 0.5) should be more efficient in bringing the solution closer to being integer than two random columns.

The branch and bound for the pairing search is performed using a binary heap, thus ensuring the least number of solutions are optimized since only the problems associated with the minimum nodes are optimized and branched on. The additional overhead of a binary heap is insignificant at this level of the algorithm. At each pairing search node, the complete column generation procedure is restarted since the heuristics quickly find good columns more efficiently than keeping previous solutions and removing or penalizing the columns that violate the pairing search rules.

3.3.4 Heuristics for the original problem

Heuristics on the original problem are usually fast and can rapidly provide a good set of columns for the restricted master problem. Even though there can be a large integer gap for many of the clusterwise regression problems, these heuristics for inserting starting columns into the restricted master problem provide significant performance gains.

3.3.4.1 Multistart exchange heuristic

The multistart exchange heuristic (Späth 1979) is employed to identify good solutions to the original problem and columns are extracted and inserted into the restricted master problem. The heuristic starts by assigning every observation to a random cluster and computes the regression lines. Then each observation is reassigned to the cluster where it fits best, if there is an improvement. In contrast to previous implementations (Carbonneau, Caporossi & Hansen 2011), this version incrementally adjusts the regression line after each reassignment. At the end of a pass over all observations, the regression lines are recalculated based on the new assignments to avoid the accumulation of small errors because of the repeated addition and removal of observations to and from the regression lines. Once a pass has been executed without any observation reassignment, a local minimum has been achieved and the exchange method iteration terminates. When pairing search rules are imposed, they are applied during the heuristic optimizations initial random assignment and at each exchange.

This heuristic is executed O times at the initialization of the restricted master problem and then once per column generated. The number of columns required in the master set partitioning problem is correlated with the complexity of the search space, thus it provides a good reference point to do a reasonable number of heuristic iterations. Since the exchange heuristic finds very good solutions quite quickly compared to the processing time required for all other steps of the exact global optimization, a more sophisticated heuristic is not required. Even for the more difficult problems, the original problem heuristic processing times including the perturbations described next are around 0.1% of the total optimization times.

3.3.4.2 Generating perturbed columns for the original problem

After each set of heuristic iterations, perturbed columns are generated from the best known solution of the original problem if there has been an improvement. A series of new columns are inserted into the restricted master problem by generating all possible one-observation-variations of each column. For each column of the best known solution, each

observation is individually and independently switched in or out of the column to generate a new column. After each switch, the original column is used as starting point, thus each new generated column has a Hamming distance of one from its original column. This is very fast and efficient since it quickly creates a large set of very good solutions. Each of the K columns has O observations and therefore this strategy creates $K \cdot O$ new columns for each improvement identified to the original problem.

3.3.5 Branch and bound, heuristics, sequencing and ending subsets (BBHSE) for the subproblem

The subproblem can be solved by heuristics first and then by branch and bound optimization assisted by an extended Repetitive Branch and Bound Algorithm RBBA called BBHSE. The first step is to generate columns by using heuristics and then switch to an exhaustive branch and bound search with sequencing of the observations and ending subset optimization while simultaneously continuing the heuristic search for improving columns. There are three heuristics for the subproblem. The first two heuristics exploit what is currently known about the problem, from both the current duals and from best known solution to the original solution. The third heuristic is the full exchange heuristic optimization. As with all parts of the column generation, when the current restricted master problem is being processed under a pairing search node, the subproblem must also be solved all the while respecting the current pairing search rules.

3.3.5.1 Sequencing and a greedy heuristic based on duals

This heuristic incrementally tries every observation in the sequence of their corresponding "one cluster constraint" (11) duals, from most rewarding to most penalizing. In addition, if the current restricted master problem is under a pairing search node, the first sort rule puts observations involved in a joint pair first and the second sort rule is on the duals. It is important to keep observations involved in joint pairs at the beginning of the sequence because once one observation is included in a branch, its joint pair observation must also be included. This forced inclusion prevents bounding of large parts of the search tree,

especially when the second observation of the pair has a very penalizing dual. Sequencing the observations by their reduced error and averaging across joint pairs when applicable was also explored, but the results were poor, both with and without ending subset optimization for stronger lower bounds. This is because any sequencing other than by the duals may place highly rewarding duals deep in the sequence and thus severely limits fathoming of the tree.

Starting at the first observation in the new sequence every observation is tried sequentially in the column. If including the observation improves the RSSE, the assignment is kept. At every observation step, if the RSSE is negative, the column is inserted into the master problem. If there is no possibility for identifying an improving column, the search is stopped, much like pruning a branch of the search tree. This procedure can generate a maximum of O columns and requires a maximum of O steps for each restricted master problem iteration since there is no branching. This heuristic will very quickly find many good columns to insert into the master problem.

3.3.5.2 Greedy heuristic based on the best known solution to the original problem

If the first heuristic described above does not identify an improving column, the next heuristic is employed. Since previous heuristics on the original problem have identified a good (probably even globally optimal) solution to the original problem (3.3.4.1), it can use this as a very good starting point for solutions to the subproblem. Each cluster of the best known solution to the original problem is a starting point for a single local exchange algorithm optimization on the subproblem. The local optimum solution for each starting cluster is inserted into the restricted master problem if it is negative, thus this procedure inserts a maximum of K columns for each restricted master problem iteration. This heuristic will also very quickly find good columns to insert into the master problem.

3.3.5.3 Initial multistart exchange heuristic

If the first two heuristics have not identified a column with a negative RSSE, a multistart exchange heuristic is performed on the subproblem. The exchange algorithm for the

subproblem is very similar to that of the original problem except that observations are simply included/excluded to/from the column and the SSE is adjusted by the relevant duals. The depth of the branch and bound search for the subproblem is the depth of the rewarding duals and the joint pairs. Since depth is related to the difficulty of solving the subproblem, the number of multistart exchange heuristic iterations is set to the depth of the search tree. The purpose of this third heuristic attempt is to perform one last try to quickly identify negative RSSE columns and to find a good upper bound before starting on the exhaustive branch and bound search. The exchange heuristic incrementally updates the regression line at each exchange and considers the duals in the calculations as specified in the subproblem's definition. As with all other optimizations, pairing search rules are respected when generating heuristic solutions.

3.3.5.4 Branch and bound exhaustive search and iterative exchange heuristic

If none of the previous heuristics have identified a negative solution to the subproblem, there is a good chance that the subproblem's globally optimal solution has been identified. Thus, an exhaustive branch and bound search is performed on the subproblem, using the currently best known solution as an upper bound. The observations are already sequenced by being in a joint pair rule and then by their duals from most rewarding to most penalizing (3.3.5.1). Optimizing incrementally larger ending subsets as defined by the Repetitive Branch and Bound Algorithm (RBBA) is also performed to strengthen the lower bounds. The current implementation of the optimization of the subproblem by branch and bound, sequencing and ending subset search optimization is similar to that of previous research in a branch and bound only solution to the clusterwise regression problem (Carbonneau, Caporossi & Hansen 2011). This also includes performing only incremental regression calculations (Carbonneau, Caporossi & Hansen 2011). As shown in previous research, an extension to the RBBA is to only optimize a limited set of ending subsets. In this case the step size is set to 10 as a good balance between strengthening the lower bounds and the added computational cost of these additional calculations (Carbonneau, Caporossi & Hansen 2011). The branch and bound search will be bounded when there is no possibility

of identifying an improvement to the currently known best solution. This is strengthened by a good best known solution (upper bound) and by stronger lower bounds calculated from ending subset optimizations. Additionally, joint pairs pairing search rules can force the search to go deep into the tree, but this is avoided by placing them at the beginning of the search. Disjoint pairs do not cause a problem since the second member of the disjoint pair can always be out of the column, thus it does not have to be explicitly evaluated, it can be deeper without affecting the fathoming nor identification of good solution.

Additionally, 10% of the processing time (in slices of 1 second) is spent on the exchange heuristic to try to identify negative RSSE columns or stronger upper bound. At any time that the heuristic identifies a negative RSSE solution, it is inserted into the restricted master problem, the exhaustive search is aborted and the restricted master problem is solved again to calculate new duals. However, if the subproblem's exhaustive branch and bound search identifies a negative solution, it is not aborted. The reason is that since the heuristics have not yet found an improving column, it is a relatively difficult problem to solve and the branch and bound should continue. Additionally, all negative columns identified by the exhaustive branch and bound search are inserted into the restricted master problem for the same reason. The strategy of inserting all negative columns identified by the exhaustive branch and bound search is beneficial because by that time, the heuristics have had plenty of time to find these columns. Since the exhaustive search usually takes much longer than the heuristics, the investment is used to keep all of these hard to identify columns.

3.3.6 Overview of the complete optimization strategy

Combining all of the above described components results in a complete optimization strategy that can be applied to the set partitioning problem in general. In this case it is specifically used to solve the clusterwise regression problem. The first and highest level loop is for the Ryan and Foster pairing search (j.2) which uses a binary heap and wll continue adding pairing rules until the master problem solves to a globally optimal solution which is integer. For each pairing search iterations (j.2), a new restricted master problem is started with heuristics adding columns at the initialization (j.2.1, j.2.2) then at each iteration

(j.2.3.1, j.2.3.2). The subproblem tries to generate negative columns first by heuristics (j.2.3.4.2, j.2.3.4.3, j.2.3.4.4) and then by and exhaustive branch and bound search (j.2.3.4.5) which uses ending subsets to strengthen the lower bounds (j.2.3.4.5.2) and will simultaneously continue searching using heuristics (j.2.3.4.5.1). As soon as negative columns are identified by the subproblem heuristics, they are inserted and the Generate Column (j.2.3) loop is restarted, the exception is when an improving column is identified by the exhaustive branch and bound search.

j Algorithm overview
- j.1 Add an empty pairing search node to the binary heap
- j.2 For each minimum node in the pairing search binary heap until an integer solution is chosen
  - j.2.1 Heuristic on original problem
  - j.2.2 Generate perturbed columns from best known solution on the original problem
  - j.2.3 Generate Column Loop while (and as soon as) columns can be added to the master problem
    - j.2.3.1 Heuristic on original problem
    - j.2.3.2 Generate perturbed columns from best known solution on the original problem
    - j.2.3.3 Solve the restricted master problem to calculate duals
    - j.2.3.4 Subproblem optimization
      - j.2.3.4.1 Sequencing based on duals
      - j.2.3.4.2 Greedy heuristic based on duals
      - j.2.3.4.3 Greedy heuristic based on the best known solution to the original problem
      - j.2.3.4.4 Multistart exchange heuristic
      - j.2.3.4.5 Exhaustive branch and bound search
        - j.2.3.4.5.1 Spend 10% of processing time on the subproblem multistart exchange heuristic
        - j.2.3.4.5.2 Incrementally larger ending subsets optimized with a step size of 10 observations
  - j.2.4 Choose a pair of observations to branch on and insert a joint and disjoint node to the binary heap
- j.3 The globally optimal integer solution has been identified

3.4 Experimental Protocol

As shown in previous work (Carbonneau, Caporossi & Hansen 2011), because of large variations in processing times, it is important to always use appropriate statistics when comparing the processing times of various clusterwise regression optimization algorithms. Consequently, for all of the optimization experiments, the same problem was executed 100

times and the sequence of the observations was randomized each time. Once the total processing time for a specific problem and algorithm had passed beyond 100 hours, it was aborted and considered timed out with an average of over one hour for solving the problem. Since only the observation order is randomized and no other data is changed, all global optimization algorithms achieve the same global minimum. The impact of the number of lines and of the amount of the level of perturbations on the processing time is examined.

From the total search space dimension of the order $K^O \cdot D^2 \cdot O$, it could be expected that the computation times increases first with the number of clusters, then with the number of observations and lastly with the number of dimensions. However in contrast to branch and bound type algorithms, column generation often becomes more efficient as the number of clusters increases (Aloise, Cafieri, Caporossi, Hansen, Perron & Liberti 2010; Aloise, Hansen & Liberti 2010). It is also expected that the computation times will increase with the amount of perturbation since there will be less structure from which to quickly reduce the search space. In addition to comparing the presented algorithms and relevant benchmarks, all relevant component of the proposed algorithm were tested and compared to better understand what exactly is providing performance gains and how these components interact. This was performed by running experiments with specific components deactivated to evaluate their importance and contribution. Since most of the proposed components of the algorithm have very low expected computational costs, but potential benefits, it is expected that combining all of them will often improve optimization times with minimal negative impact when these algorithms are not beneficial.

3.4.1 Datasets

The datasets used in the following experiments are composed of 100 observations synthetically created from 2 to 6 lines (Table XV). Based on these lines, datasets are generated with increasing amounts of perturbation from the normal distribution (Figure 8 and Figure 9).

Table XV Line definitions

Line Number	Formula
1	y = -0.356x + 7.583
2	y = 0.952x + 0.909
3	y = 0.255x – 1.132
4	y = -0.197x + 10.576
5	y = 0.479x + 6.749
6	y = -0.749x + 5.615

The independent variables (x) are random (uniform distribution) and rounded to two decimal places and the dependent variables (y) are perturbed (normal distribution) by an increasing amount. Using these datasets, the effects of increasing the number of clusters and the amount of perturbations was examined.

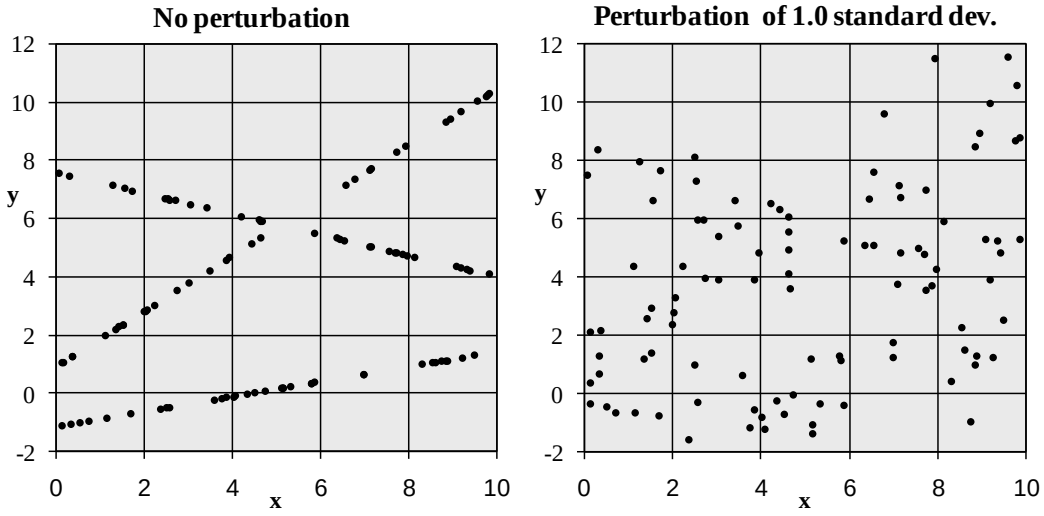

Figure 8 Plots of the three lines dataset without perturbations and of one standard deviation

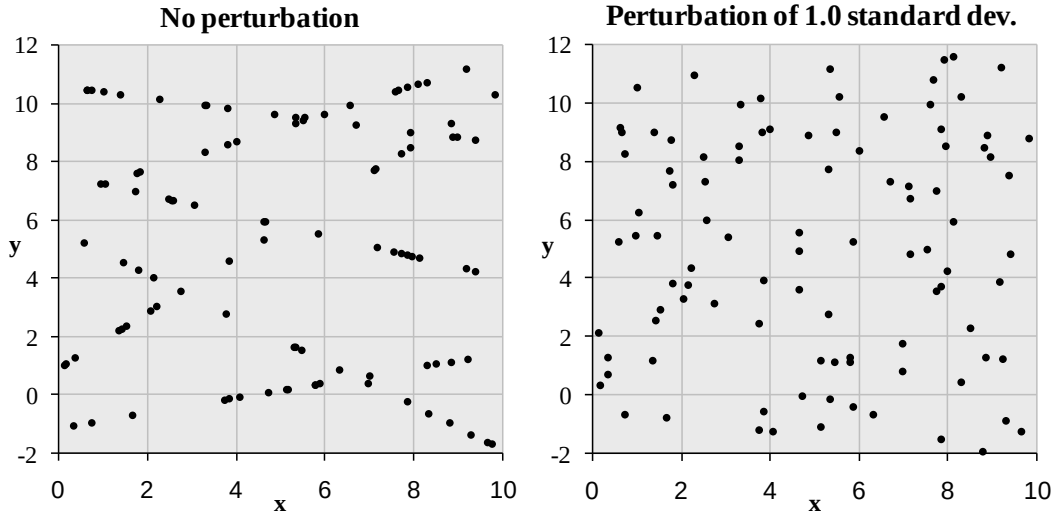

Figure 9 Plots of the six lines dataset without perturbations and of one standard deviation

3.4.2 Implementation

To evaluate the proposed algorithms and perform the required experiments, the column generation and all of its related component extensions were programmed using the C programming language. The actual partitioning restricted master problem is a linear program which is optimized by the free and open source LP_SOLVE 5.5.2.0 Mixed Integer Linear Programming (MILP) solver (Berkelaar, Eikland & Notebaert 2010). Other commercial solvers such as CPLEX would most probably be more efficient, however, only a small percentage of the optimization time is actually spent on the master problem, thus the potential gains would be very small. Using a free and open source solver permits other researchers to reproduce the results even if they do not have access to commercial software.

The benchmark branch and bound algorithm employed in this research exploits symmetry breaking, iterative heuristic optimization, observation sequencing, ending subset optimization (BBHSE) and incremental regression calculations (Carbonneau, Caporossi & Hansen 2011). A step size of 10 was used for ending subset optimization, and 0.1 second is

spend iteratively on heuristic optimization and then 0.9 seconds on exhaustive searching (Carbonneau, Caporossi & Hansen 2011).

The benchmark branch and bound algorithm, column generation algorithm and all related components and computations were implemented using the C programming language with a precision of over 9 significant digits for comparisons (Knuth 1997) and of over 15 significant digits for calculations. The program was compiled using gcc version 4.1.2 with the code optimization option "-O3". All optimizations were performed on an Intel Core 2 Quad Q8200 CPUs at 2.33GHz with 4 MB of shared L2 cache and 4 GB of RAM running Linux. Each optimization was limited to one CPU core, thus there is no parallel processing.

The benchmark mixed logical-quadratic programming model was solved using the IBM ILOG OPL-CPLEX (OPL 6.3 and CPLEX 12.1.0) environment since it is currently the only optimizer that can solve this type of problem. Thus the model was implemented using the OPL programming language (Van Hentenryck, Lustig, Michel & Puget 1999; IBM 2009; IBM 2009) as detailed below. In this modeling language, the *z* cluster assignment variables are defined as boolean.

Quadratic programming clusterwise regression OPL model with implication of constraints:

```
1 minimize sum(o in O) sum(k in K) e[o,k]^2;
2 subject to {
3 forall(o in O){
4  forall(k in K){
5    (z[o,k]==1) => sum(d in D) x[o,d]*b[d,k] + e[o,k] == y[o]; }
6  sum(k in K: k <= o) z[o,k] == 1; } }
```

The CPLEX precision parameters were set to the maximum available of 9 significant digits. The relevant tolerances parameters were reduced as much as possible in an attempt to ensure optimal and high precision solutions. The node file flag was set to store nodes on disk (with compression) when the in-memory set is larger than 128 MB.

The CPLEX and Branch and Bound (BB.h.s.e) algorithm are identified in the results tables and the Column Generation (CG...) algorithm and its options are identified as CG with periods separating the options (Table XVI).

Table XVI Algorithm codes and descriptions

Code	Description
CPLEX	Mixed logical-quadratic formulation optimized by CPLEX
BB.h.s.e	Branch and Bound with heuristic, sequencing and ending subsets
CG…	Column Generation with subproblem solved by branch and bound and sequenced by joint pairs and duals
…m…	Heuristic on original problem for the restricted master problem (O times to start then once per column generated)
…p…	Perturbed good columns from best known solution for the original problem for the restricted master problem
…s…	Greedy Heuristics on subproblem, early termination on improving column after complete sections
…S…	Full Heuristic on subproblem, early termination on first improving column
…a…	Insert all improving BB solutions and no early termination, otherwise no early termination of BB and only insert incremental best improving columns
…e…	Incremental ending subsets optimization with a step size of 10

3.5 Results and discussions

The results of the optimization times and the globally optimal SSE for 2 to 6 lines and for perturbations ranging from 0 to 1.2 standard deviations are presented in Table XVII. For two lines (clusters), both the BBHSE and the CPLEX methods significantly outperformed all tested column generation methods. Additionally, the BBHSE method significantly outperformed all tested column generations methods for three lines. However, the column generation strategy significantly outperformed the BBHSE method for the tested scenarios with more than three lines.

Table XVII Statistics for the synthetic lines datasets, 100 randomized sequences

K	SD	CG.s.S.a.e Avg.	CG.m.s.S.a.e Avg.	CG.m.p.a.e Avg.	CG.m.p.S.a.e Avg.	CG.m.p.s.S.a Avg.	CG.m.p.s.S.e Avg.	CG.m.p.s.S.a.e Avg.	BB.h.s.e Avg.	CPLEX Avg.	SSE
2	0.0	>1h	>1h	>1h	1021.4	>1h	717.9	476.6	0.1	5.7	0.00078770571912
	0.1				129.4		92.5	85.1	0.1	5.7	1.0723288888
	0.2				153.7		114.1	100.6	0.1	10.9	3.9922024816
	0.3				290.5		178.6	161.3	0.1	11.7	8.8703134594
	0.4				1612.0		397.5	1179.5	0.1	11.4	15.8756240090
	0.5				>1h		475.5	1995.2	0.1	18.3	24.9181098860
	0.6						1206.4	2284.4	0.1	28.0	36.0849684470
	0.7						3745.8	2777.9	0.1	22.5	48.5803115160
	0.8						>1h	>1h	0.1	78.6	62.6978726800
	0.9								0.1	508.8	78.1968255420
	1.0								0.1	3870.5	94.6400060390
	1.1								0.1	>1h	111.5411427900
	1.2								0.2		129.2320636100
3	0.0	>1h	61.1	62.2	9.5	676.4	14.2	9.7	0.1	>1h	0.00085391256203
	0.1		16.9	21.0	4.6	118.5	4.6	4.2	0.1		0.97813102750
	0.2		24.4	35.1	7.9	215.0	9.4	7.7	0.1		3.5849567133
	0.3		28.3	39.2	14.0	316.0	13.9	13.0	0.1		7.9377102554
	0.4		38.1	38.4	21.8	465.1	17.4	18.6	0.1		14.235471527
	0.5		180.9	42.9	27.8	901.4	22.2	23.5	0.1		22.338570677
	0.6		124.7	89.3	45.1	2469.1	30.7	31.0	0.1		32.372036752
	0.7		470.5	249.4	79.1	>1h	51.0	59.2	0.1		44.116397725
	0.8		818.9	992.0	140.8		104.8	105.6	0.1		56.603362531
	0.9		>1h	>1h	259.6		205.5	203.4	0.2		70.801852018
	1.0				903.1		741.6	779.8	0.6		86.966652321
	1.1				>1h		3573.7	2038.1	3.4		104.88503660
	1.2						>1h	>1h	69.8		122.26008859
4	0.0	>1h	6.3	4.7	3.4	27.3	3.5	2.8	0.1	>1h	0.000835049001670
	0.1		6.4	2.7	3.3	14.9	3.3	2.9	0.1		0.96421757841
	0.2		8.3	2.7	4.9	20.5	5.2	4.2	0.1		3.4534257881
	0.3		12.2	5.8	6.9	36.0	6.5	5.8	0.1		7.8919280226
	0.4		65.2	11.4	11.0	96.0	10.0	11.0	0.1		14.180944323
	0.5		>1h	>1h	1491.3	>1h	1355.6	1366.6	0.3		22.045618031
	0.6		48.3	19.5	28.7	560.3	27.9	30.1	2.4		29.277665076
	0.7		1714.9	34.1	41.7	2119.6	39.3	40.9	18.4		37.985634502
	0.8		181.1	96.1	97.2	>1h	73.0	100.2	168.3		47.639222868
	0.9		298.0	338.2	208.8		165.9	168.5	596.3		58.073585651
	1.0		391.5	1176.0	375.5		443.8	386.3	2014.0		68.961604153
	1.1		829.5	>1h	757.5		866.8	646.1	2091.6		78.465822977
	1.2		1140.8		1249.8		1370.7	1087.1	>1h		85.654928625
5	0.0	>1h	3.0	1.1	2.1	4.9	2.3	1.9	0.2	>1h	0.00078565415910
	0.1	322.0	3.9	1.1	3.4	5.4	1.9	1.9	0.3		0.90862006887
	0.2	>1h	6.9	1.9	6.2	13.6	3.8	3.4	2.2		3.2009281353
	0.3		13.4	5.7	341.3	107.3	7.2	6.5	8.1		7.2271515309
	0.4		>1h	>1h	>1h	>1h	323.7	305.4	44.5		12.246409278
	0.5						2468.8	3394.4	1176.2		17.276462584
	0.6						>1h	1189.6	2449.7		22.535309036
	0.7							1430.8	>1h		28.264921223
	0.8							3337.0			34.785664602
	0.9							>1h			41.793766152
6	0.0	>1h	2.3	0.7	2.4	2.4	2.0	1.7	2.0	>1h	0.00084401418219
	0.1		3.0	0.7	1.9	2.6	1.9	1.9	6.4		0.79257000333
	0.2		4.6	1.0	2.6	4.5	2.9	2.7	35.3		2.7428020980
	0.3		894.2	>1h	57.6	273.1	76.1	78.2	331.5		6.2404160986
	0.4		>1h		468.4	>1h	496.6	431.7	>1h		10.643516929
	0.5				257.1		296.8	257.3			14.546443517
	0.6				2048.7		2851.9	3256.9			19.509333359
	0.7				>1h		>1h	>1h			23.554518048

As expected, the combination of all of the proposed components often provided significant improvements in optimization times and had minimal negative impact when they did not provide any benefit. The results provide much information on the importance

of each component since they demonstrate average optimization times when each component was deactivated. In all cases, the heuristic on the original problem for inserting columns (...m...) into the restricted master problem clearly provided the most benefit since there was only one problem set which column generation could optimize in an average time of under one hour when this component was deactivated.

The next most important component was the optimization of incrementally larger ending subsets (...e...) which is fundamentally based on the RBBA algorithm (Brusco 2006). This is an important result that further demonstrates the flexibility, efficiency and effectiveness of this new optimization paradigm for strengthening the lower bounds of a branch and bound search.

Following this and related to the heuristic on the original problem for inserting columns in the restricted master problem were two more heuristics. The first was the full exchange heuristic on the subproblem (...S...) with early termination on the first improving column and the second was the generation of perturbed columns (...p...) based on the best known solution to the original problem. The next component in term of importance was the greedy subproblem heuristics based on the duals and the best known original problem solution (...s...), these provided benefits limited to only the scenarios with 2, 3 and 5 lines. Lastly, avoiding early termination of the exhaustive subproblem branch and bound search and inserting all improving columns was only beneficial for the problem set of 5 lines.

It is also evident from the results observed during this research work, although not formally tested in this article, that sequencing observations involved in joint pairing search rules is very important and they must be put first as to avoid forcing the subproblem branch and bound search to go deep just to respect a joint pair rule. This becomes relevant only when there is an integer gap in the master problem, thus requiring the Ryan and Foster pairing search.

Although column generation will often become more efficient as the number of clusters increases, the clusterwise regression problem presents some particular problems because a very large integer gap may occur. The integer gap appears to increase with the number of

dimensions and with the level of perturbation. This is evidenced in the results for optimizing 4 lines with a perturbation level of 0.5 standard deviation which had exceptionally high processing times compared to very similar scenarios such as 0.4 and 0.6 standard deviation. This particular case happens to have a large integer gap, whereas the other similar scenarios do not. This integer gap problem also shows up with 5 lines, where solving the perturbation level of 0.3 standard deviation does not have an integer gap, therefore does not have pairing search iterations. However, a perturbation level of 0.4 resulted in 6 to 52 pairing search iterations. For 6 lines, a perturbation level of 0.2 does not have an integer gap, but 0.3 had 4 to 14 pairing search iterations and 0.4 had 14 to 42. Thus, in all of these cases where the integer gap started to become significant, processing times increased much faster and negated the usual benefit of increased number of clusters for column generation. However, this seems to become a problem when the clusters are no longer clearly defined. Thus, since clusterwise regression should usually be used on datasets that have many linear structures, this may not actually be a problem in practice.

3.6 Conclusion

The observed benefits of each tested column generation component are summarized in (Table XVIII), most importantly, all of the components provide benefits with minimal or no negative side effects for the currently tested range of datasets. These results show that judicious use of heuristics can provide significant performance gains with relatively minimal additional computational cost. Additionally, the current work further demonstrates and expands the successful use of the new paradigm of using incrementally larger ending subsets to strengthen the lower bounds of a branch and bound search as pioneered by Brusco's RBBA (Brusco 2006). At the same time, this research demonstrates that the column generation method only becomes competitive as the number of lines (clusters) increases beyond three clusters for the above tested datasets. It also shows that the integer gap becomes a problem as the number of clusters, or perturbations increases, which is generally also an indication that there are no real linear structures in the data.

Table XVIII Summary of the benefits provided by each component

Code	Benefit
…m…	Very High
…e…	High
…S…	High
…p…	High
…s…	Medium
…a…	Low

CHAPTER IV:

Pairwise issue modeling for negotiation counteroffer prediction using neural networks

Réal A. Carbonneau[1]

Gregory E. Kersten[2]

Rustam M. Vahidov[2]

[1]Department of Management Sciences

GERAD and HEC Montréal

3000, chemin de la Côte-Sainte-Catherine

Montréal, Québec H3T 2A7, Canada

[2]Department of Decision Sciences & MIS

John Molson School of Business

Concordia University

1455 de Maisonneuve Blvd W

Montréal, Québec H3G 1M8, Canada

This paper was published in the Decision Support Systems (2011)

Abstract

Electronic negotiation systems can incorporate computational models and algorithms in order to help negotiators achieve their objectives. An important opportunity in this respect is the development of a component, which can assess an expected reaction by a counterpart to a given trial offer before it is submitted. This work proposes a pairwise modeling approach that provides the possibility of developing flexible and generic models for counteroffer prediction when the negotiation cases are similar. The key feature is that each negotiated issue is predicted while paired with each of the other issues and the permutations of issue pairs across all negotiation offers are confounded together. This data fusion permits extractions of common relationships across all issues, resulting in a type of pattern fusion. Experiments with electronic negotiation data demonstrated that the model's predictive performance is equivalent to case-specific models while offering a high degree of flexibility and generality even when predicting to a new issue.

4.1 Introduction

Business negotiations are an important type of exchange mechanism. The competency in conducting negotiations critically affects long-term business relationships, profitability, and reputations of businesses. Due to the rise of e-business, electronic negotiations have gained heightened importance lately (Kersten & Lai 2007; Kilgour & Eden 2010). Electronic negotiations systems, which are web-based successors of negotiation support systems (Jelassi & Foroughi 1989), allow parties located in various parts of the world to seek mutually acceptable agreements by exchanging offers over the networks in a structured or unstructured fashion. The organic involvement of the digital medium in these exchanges provides new opportunities for employing support and automation tools, such as preference modeling and software agents, for promoting effective decision-making.

The purpose of this paper is to investigate the feasibility of developing a generalized approach for empirically modeling an opponent's future offers. Consequently, the main contribution of this work is a pairwise modeling approach that is flexible with respects to

the set of issues in a negotiation case and even to new unseen issues. In others words, the model has inputs and outputs that are independent of the particular issues of a specific negotiation case.

The approach is tested using data obtained from electronic negotiation experiments, which provide a rich source of information about the relationships between negotiators, their individual actions, and the negotiation dynamics. Advanced negotiation support tools equipped with adaptive capabilities to learn from past negotiations and assist in selecting appropriate negotiation tactics, can effectively utilize this information to help human negotiators in composing offers (Bartolini, Priest & Jennings 2005). This work focuses entirely on the development and testing of the flexible predictive model, rather than its implementation as part of some electronic negotiation system. The model could be potentially incorporated into the analytical toolbox of any given electronic negotiation system, or it could be used by software agents for negotiation automation or for providing guidance to a human user.

As the development of the internet allowed for the natural involvement of computational models and methods in the electronic negotiation process, the question of adequately splitting of human vs. computer tasks had emerged. In addition to providing passive analytical facilities, computational models have been proposed for active negotiation support, as well as negotiation automation. The use of intelligent support tools for facilitating effective and efficient negotiations have been widely investigated in the past. Automated negotiations performed by software agents have been considered by many (Beam & Segev 1997; Chavez, Dreilinger, Guttman & Maes 1997; Maes, Guttman & Moukas 1999; Jennings, Faratin, Lomuscio, Parsons, Wooldridge & Sierra 2001). Early work in this respect applied genetic algorithms to generate rules which relate the negotiators' current offers with the likely subsequent offers (Matwin, Szapiro & Haigh 1991). The algorithm evolves multiple classifiers by assigning a higher fitness to rules that more frequently contribute towards "compromise trajectories". Kasbah is agent-based marketplace in which various agents created by the users engage in bilateral negotiations on behalf of their principals (buyers and sellers) (Chavez & Maes 1996). These agents, follow

one of the three negotiation strategies defined by a price-concession curve over time (Chavez, Dreilinger, Guttman & Maes 1997). More recently, a semantic web-based agent community was used to introduce the concept of pervasive negotiation support (Kwon, Shin & Kim 2006). The system, called "SmartGuide", included user, supplier, and negotiation agents to provide a flexible environment for context-aware automated negotiations.

The above overview of intelligent techniques used in negotiations is not comprehensive, but it provides insights into research on negotiation automation. In most business negotiation contexts however, humans need to be involved in the process with the intelligent software possibly playing a supporting role. Research in this area focuses on solutions for assisting human negotiators. An overview of electronic negotiations, negotiation support systems and negotiation software agents (NSA) includes discussion of the Aspire system, in which an agent uses an inference engine to provide recommendations based on inputs and previously encoded rules (Kersten & Lo 2003). Another application of an agent assistant in commerce negotiations has been implemented in the eAgora marketplace (Chen, Kersten & Vahidov 2004; Chen, Vahidov & Kersten 2005), where an agent watches over the shoulder of the negotiator and critiques trial offers. The agent also advises an action upon receiving the counterpart's offer and generates a package of adequate candidate offers for the consideration by the user. Regardless of the extent of computer involvement (automation or active/passive support) in composing offers, it would be useful to be able to predict potential reaction by a counterpart to a given trial offer. In a "toolbox" mode, a human user could perform "what-if" analysis before committing to an offer. If software agents are involved, the predictive model could help in the search for the most promising offer.

The rest of the paper is organized as follows. First, an overview of negotiation modeling and the proposed pairwise approach is presented (4.2), followed by the hypotheses (4.3). Subsequently, the negotiation case, which is the source of data for testing the hypotheses, is described. Using this negotiation case as the example, the pairwise modeling approach is

explored in detail (4.4). Next, the experimental setup (4.5) is presented followed by the results and relevant discussions (4.6) and finishing with the conclusions (4.7).

4.2 Background

This paper builds upon the past work that proposed a model developed with neural networks for predicting opponents' counteroffers in the context of electronic negotiations (Carbonneau 2007; Carbonneau, Kersten & Vahidov 2008). Although the results were encouraging, a major disadvantage was that a separate neural network had to be trained for each specific negotiation case. This was necessary because the model included inputs and outputs for all negotiation issues of the case. The pairwise modeling approach proposed in this paper represents a radical improvement over the past work, as it allows using the same model for families of cases. The approach is flexible with respects to the set of issues because it considers tradeoffs between pairs of issues rather than individual issues separately. What issues are included in a particular tradeoff is not relevant as long as the tradeoff is made, thus making it possible to learn common counteroffer prediction patterns across different negotiation scenarios.

The preferences of each negotiator with regards to the negotiated issues are typically private. However, these preferences are partially exposed by the concessions made in a sequence of offers. More specifically, the concessions made by a negotiator on one issue compared to those made on another provide information about the relative importance of the issues. Consequently, one would expect that there may be general concession patterns across all pairs of issues for a given class of negotiation cases. These negotiation classes may have common or different sets of issues, and their number may also differ. Their similarity is in the issue importance to the negotiators, and in the trade-off and logrolling coefficients. The proposed approach is based on the expectation that negotiations may be classified on the basis of these characteristics irrespectively of the actual subject of the negotiation.

Thus, to develop a more flexible and general predictive negotiation counteroffer modeling approach, a pairwise modeling of the negotiation issues is proposed, whereby the issue set in an offer is broken down into pairs. The main advantage of this approach is in its flexibility for modeling a varying set of negotiation issues. This is achieved by abstracting specific issues to pairs of issues and thus providing the potential to infer patterns that are common across pairs of issues. Another advantage is that splitting offers into issue-pairs results in more observations than in the original set, which generally tends to reduce "overfitting" of the model and improve its generalization capacity. The issue input dimensions are reduced to those of the pair of issues and the observations are increased to the permutations of pairs of issues. On the other hand, breaking the negotiation model down to a set of issue pairs restricts the range of models that can be learnt from the data, since the issues are now only matched in a pairwise fashion. Thus, any relationship that requires taking into account more than two issues simultaneously cannot be learnt. For clarification, in this paper, any reference to the terms pairwise or pairs or other similar terms is always related to the pairwise analysis of the set of negotiation issues, and not to any other type of pair such as a offer-counteroffer pair.

4.2.1 Pairwise analysis

Breaking down a problem into pairs of components, which are considered individually and then recombined is an intuitive idea studied extensively in different domains. In psychology and decision analysis, pairwise comparison of decision attribute values has been used to identify preferences while reducing the cognitive effort required by decision makers (Thurstone 1994). In statistics and data mining, pairwise coupling has been applied to classification problems, where a classifier for each pair is developed and the results are recombined to provide multi-class classification (Bradley & Milton 1952; Hastie & Tibshirani 1998). When a direct multi-class model is difficult to build, individual one-against-one (OAO), one-against-all (OAA) or p-against-q (PAQ) classifiers can be built and then recombined into a multi-class classifier (Ou & Murphey 2007).

A major difference between much of the previous work on pairwise classification and the current pairwise modeling is that the counteroffer prediction is not modeled as a classification problem. Thus, the concept of modeling classes in a binary or pairwise fashion either as OAO, OAA or PAQ does not apply. Even in a more general sense, the currently proposed application is actually the opposite of previous work, in the sense that the pairs of issues are combined into a single dataset and modeled together, instead of modeling each pair combination or permutation separately. Other than being flexible, this approach may have benefits for predicting opponents' behavior in negotiations for two reasons. The primary one is that each predicted issue should have relatively stronger relationships to past information about the same issue as compared to information about another issue. The second is that across all issues there may be common patterns related to concessions in time between a pair of issues that are relevant to predicting the next counteroffer. Thus, it may be possible to construct a general model that predicts expected counteroffers based on the patterns found in abstracted pairs of issues.

4.2.2 Modeling approach

An important aspect of negotiations is being able to learn the opponent's preferences and anticipate his/her future offers. Some of the studies mentioned earlier take into account the opponent's moves in the process of offer generation. For example, past concessions made by the counterpart have been used to construct the model of the counterpart (Lee 2004). If, on the average, they exceeded a pre-defined threshold level, the opponent was modeled as having a "positive" attitude. Some other past works for profiling an opponent included: game-theoretic approach with Bayesian belief revision for modeling a negotiation counterpart (Zeng & Sycara 1998); probabilistic influence diagrams representing the counterpart's decision-making (Mudgal & Vassileva 2000); and opponent preference modeling using non-linear regression analysis (Hou 2004; Brzostowski & Kowalczyk 2006), Chebyshev's polynomials (Saha, Biswas & Sen 2005) and genetic algorithms (Choi, Liu & Chan 2001). An alternative way of adapting to opponent's moves is to analyze past offers without modeling opponent's preferences directly. For example, in one work, the

method generated offers that are similar to the latest offers by an opponent based on fuzzy similarity measure (Faratin, Sierra & Jennings 2002). Our approach is similar in the way that it does not attempt to profile an opponent explicitly.

The current work builds upon the idea of predicting an opponent's next offer based on inferred patterns from a large set of past negotiations and the history of a given on-going negotiation. Previous work proposed a negotiation counteroffer prediction model based on a neural network that included simultaneous inputs for all negotiation issues (Carbonneau 2007; Carbonneau, Kersten & Vahidov 2008). Theoretically, a neural network could learn any function that described the output (or dependent) variables as a result of the input (independent) variables (de Figueiredo 1980; Hornik 1991). By moving towards a pairwise model in this work, the range of functions that can be learnt is being restricted as compared to a model with all issues as simultaneous inputs.

The proposed approach is based on predicting a value for a given issue while pairing this issue with all the other ones. Subsequently, the average of the pairwise predictions for a target issue is taken as the final predicted issue value. Because all pairs are grouped and modeled together and information identifying the individual issues themselves is removed, the pairwise model is only able to learn patterns that are common to all issue pairs, and not those specific to an issue. Thus, at the expense of more complex issue-specific models, the pairwise model provides flexibility and it should generalize well across all issues, including predicting new issues not involved in training the network.

Combining multiple datasets to achieve a common set of patterns is similar to the idea of data fusion, where data from multiple different sources is combined to increase prediction accuracy (Waltz & Llinas 1990; Abidi & Gonzalez 1992; Hall & Llinas 1997; Torra 2003). Similar to inferring a consensus from multiple sensors observing the same process all the while ignoring patterns that are unique to a specific sensor, the pairwise model infers patterns that are general across all pairs and ignores those which are unique to a specific issue or issue-pair. Combining many similar datasets into one model has also been applied with success to forecasting the demand of multiple products in a supply chain when groups

of products have similar patterns over time. For example, if one of the historical demand patterns of multiple modeled products is seasonal, then some of the future patterns may also have a similar seasonal pattern. When a newly introduced product, which was not considered in the model, has similar seasonal pattern, it can be identified and its future sales predicted with the same general forecasting model (Carbonneau, Vahidov & Laframboise 2008).

It is well known that "ensemble" predictions can improve modeling accuracy (Hansen & Salamon 1990; Haykin 1998), and because the counteroffer prediction for an issue is an average of multiple pairwise predictions, the prediction is actually the result of an "ensemble" of predictions, even though it does not come from an "ensemble" of neural networks. Thus, the proposed pairwise approach is expected to benefit from improved modeling accuracy.

4.2.3 Pairing of negotiation issues

This work proposes that concession patterns through time may be similar between different pairs of issues even though the issues may be completely different. It may appear counter-intuitive that the issues themselves are "anonymized" in the model. Nonetheless, what matters in offer analysis and generation is the expected utility of the subsequent counteroffer from the opponent. The expected utility is calculated by passing the expected counteroffer from a prediction model through the sender's utility function. In other words, the question is: what offer should be sent to the counterparty that will result in receiving back the best possible counteroffer (or acceptance) based on the sender's utility function? This is similar to asking what concessions should be given in the current offer that would result in favorable concessions on the opponent's part. To do this, the expected counteroffer for a given offer must be predicted, and we expect that this can be achieved by analyzing pairs of issues using a common model. The general patterns inferred by the learning algorithm from these combined pairwise observations could be considered a type of pattern fusion across all issue pairs.

The process of construction of pairs of issues is straightforward. For a set of n issues, every issue appears in n-1 pairs as the primary issue of the pair and in n-1 pairs as the secondary issue of the pair. Thus, in total there are $n(n\text{-}1)$ pairs, which are the permutations of all issues $P(n,2) = (n!/(n\text{-}2)!) = n(n\text{-}1)$. To differentiate each issue of a pair of issues, we use the terms primary issue and secondary issue. The primary issue is the same issue that will be modeled as a dependent variable and the secondary issue permits learning of inter-issue patterns. In this paper, the terms primary and secondary are only used to refer to the issues in a pair and have no significance pertaining to the importance of the issues in the negotiations.

It is assumed that in negotiations, the strongest relationships are between the issue value in a counteroffer and the information related to that same issue in the offer, for which the counteroffer was made. Based on this assumption, the primary independent issue in each pair is matched with the same issue in the counteroffer. For example, when predicting the next counteroffer for the price issue, the primary issue will always be the price issue. Because price appears in *n-1* pairs as a primary independent issue, each of these pairs will have one of the other issues present in a secondary role.

For example, a negotiation that has four issues (price, delivery, payment, returns) will have 12 permutations of pairs of issues. As an example, one of the 12 pairs would have price (primary) and delivery (secondary) as independent issues predicting the expected counteroffer for the price as the dependent variable. The independent issue variables are the set of variables for each issue, such as the issue value in the current offer and past offers, and the average, minimum, maximum, and standard deviation of the issue values. Any information that is available at the time of making the current offer and which is abstracted from a specific issue can be included in the independent variables of the model.

It is assumed that the negotiation issues are either continuous or have discrete or ordinal levels which can be approximated as continuous variables. It is expected that modeling a binary (Yes/No) type issue will be very different from modeling a variable that is continuous, since a binary variable will provide very little information in terms of patterns

through time in contrast to a continuous variable. One reason is that the buyer and seller will probably each start with opposite issue levels and there is no middle meeting point, once one party makes a concession on the issue, both parties will probably be on common ground on that issue. Thus, the patterns may be very different from a continuous variable. Similar problems are present with categorical issues as well, since they are usually expanded to a set of binary variables.

4.2.4 Predictive model

There are many different methods for building predictive models from a dataset. The most common linear method is multiple linear regression (MLR). Some advanced non-linear models include artificial neural networks (ANN), support vector machines (SVM) (Vapnik 2000) and random forests (RF) (Breiman 2001). In this paper, the linear benchmark is multiple linear regression and a neural network is employed for building a non-linear predictive model. The pairwise predictive model is extracted as the function that relates the primary issue inputs, the secondary issue inputs and the general inputs with the counteroffer on the primary issue.

As with the neural network model from previous research which considers all negotiation issues simultaneously (Carbonneau 2007; Carbonneau, Kersten & Vahidov 2008), the pairwise model is trained using the Levenberg-Marquardt algorithm (Hagan & Menhaj 1994; Hagan, Demuth & Beale 1996) with Bayesian Regularization (MacKay 1992; Foresee & Hagan 1997). The pairwise neural network (Figure 10) has nP primary issue inputs, nS secondary issue inputs, nG general inputs, nH hidden layer neurons and one output. In the hidden layer neurons, the weighted inputs are summed and transformed with the tan-sigmoid function, which performs non-linear scaling to values between -1 and 1. In the output layer, the results are summed and transformed by the linear transfer function and the output is the expected counteroffer amount for the primary issue of the pair of issues.

Thus, there are $nP+nS+nG$ inputs and $(nP+nS+nG)\times nH$ weights between the inputs and the first layer neurons, with an additional nH weights for the hidden layer neuron biases. Following this, there are nH weights between the hidden layer neurons and the output

neuron with an additional weight for the output neuron bias. This represents a total of $(nP+nS+nG)\times nH+nH\times 2+1$ weights (connections) and thus the power or complexity of the network.

As a rule of thumb, it is advised to keep the ratio of observations to weights above 10. In the case of the pairwise model, meeting this requirement does not pose a problem because the issue input dimensions are reduced to those of a pair of issues and the observations are increased to the permutations of pairs of issues. For example, the final neural network in this study has 23 inputs and 5 hidden layer neurons resulting in 126 weights and the present case has 12 pairwise observations per offer observation. Subsequently, only 100 negotiation offer observations generate 1,200 pairwise observations, which represents an observations-to-weights ratio of almost 10 (1200/126=9.52). Expanding the offer observations to the permutations of pairs will cause a large increase in the number of pairwise observations, and although this is beneficial to the quality of the inferred model, it may require longer learning time. In extreme cases, a random sample of the pairwise observations can be used as long as the observations-to-weights ratio is kept high enough relative to the expected pattern complexity.

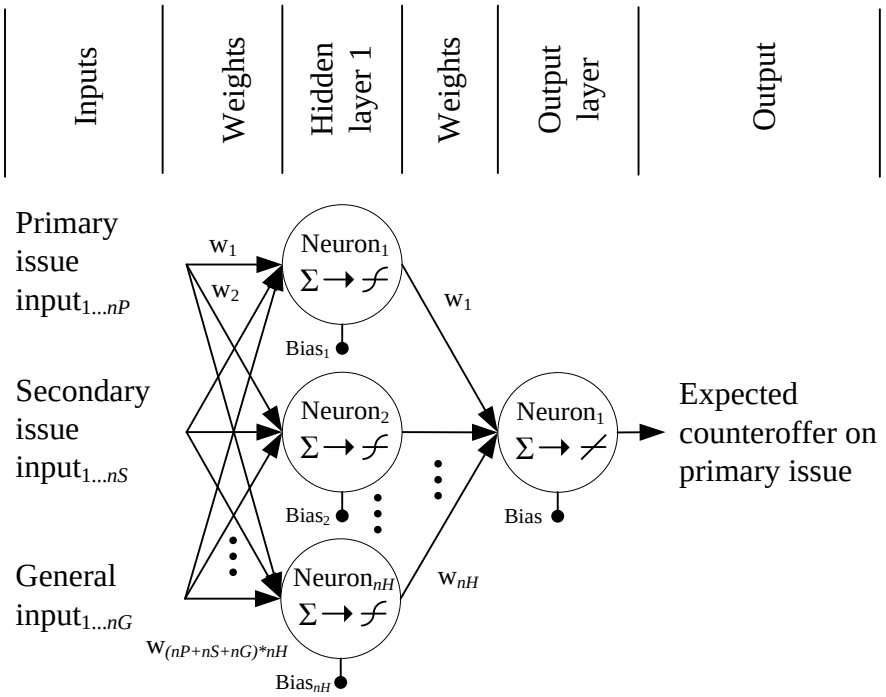

Figure 10 Pairwise issue negotiation modeling neural network design

The details of the modeling approach and an example based on the negotiation case used to test the hypotheses are presented in Section 4.4.

4.3 Hypotheses

Evaluating the performance of the proposed model requires reference benchmark models. The first benchmark proposed is the naïve model, where the next counteroffer for an issue is predicted as being the same as the most recent offer from the opponent on the specified issue. Although very simple, the naïve model is also flexible with respect to any set of negotiation issues, thus as a minimum requirement any proposed predictive model must outperform the naïve model. Therefore, the first hypothesis is:

H1. A non-linear pairwise model will have a lower counteroffer prediction error than the naïve model.

In addition, because the pairwise model uses a neural network (a non-linear universal approximator), it is important to have a linear benchmark to make sure that the non-linearity is providing a significant benefit. Consequently, the second hypothesis is:

H2. A non-linear pairwise model will have a lower counteroffer prediction error than a linear pairwise model.

Previous work has proposed a more complex counteroffer prediction model that was custom-built for a specific negotiation case. The model considered all negotiation issues simultaneously (Carbonneau 2007; Carbonneau, Kersten & Vahidov 2008). The network topology consisted of 39 inputs, 10 hidden layer neurons and 4 outputs (one for each issue). Training of the neural network was performed using the Levenberg-Marquardt algorithm (Hagan & Menhaj 1994; Hagan, Demuth & Beale 1996) with Bayesian Regularization (MacKay 1992; Foresee & Hagan 1997). The dataset used for training was the same Cypress/Itex negotiation case as in the current research, as was the 80% training and 20% testing split of the data. This prediction model is also used as a reference model, and since it considers all issues simultaneously, it should provide as good or better predictions than the one proposed in this paper , i.e. its error is expected to be similar or lower than the more constrained general pairwise model. However, since the pairwise model is compared against this incumbent model, the test will verify that it does not perform significantly worse than the full model, thus the third hypothesis is:

H3. A non-linear pairwise model will not a have higher counteroffer prediction error than a full simultaneous issue non-linear model.

We further expect that the pairwise model can learn general counteroffer patterns from sets of issues and use these patterns to predict counteroffers for new unseen negotiation issues. Testing of this claim can be performed by developing a pairwise model for a subset of issues from a negotiation case and predicting counteroffers for issues from the same case, which were initially excluded during the model development. Thus, the model would be predicting counteroffers for issues that were not available during inference of the model. The non-linear pairwise model prediction error on unseen issues must be significantly

lower than that of the naïve model since the naïve model is also general. However, it should not be significantly worse than that of a pairwise model trained on data including the particular issue. Accordingly, the fourth and fifth hypotheses are as follows:

H4. A non-linear pairwise model will have a lower counteroffer prediction error on new unseen issues than the naïve model.

H5. A non-linear pairwise model will not have a higher counteroffer prediction error on new unseen issues than a non-linear pairwise model on previously modeled issues.

As for the negotiation issues, the type of data can have an impact on counteroffer prediction performance. Continuous or near-continuous variables, such as price or time, permit negotiation participants to make offers from a large or theoretically infinite range of values. Although some issues such as price or time may be in reality continuous, the offers are often subject to reasonable minimum step size (e.g., 0.01$, 1 hour). Continuous or near-continuous negotiation issues can capture richer patterns through time and permit more precise counteroffer predictions as opposed to very few pre-defined discretized continuous or ordinal levels. For example, if the same price issue is fixed to three discrete levels, the information captured for each offer is less precise, as it permits capturing only a very limited number of patterns though time and limits the counteroffer prediction potential. Three issue levels result in only two possible concessions from either negotiation participant on that issue, thus severely limiting the number of patterns which could be learned. In addition, any counteroffer prediction must fit into one of the three fixed levels since the counteroffer can only be one of these. Consequently, even if an expected counteroffer prediction could be more precise on average (because it is a conditional mean), it must be rounded or matched to one of the three discrete levels, thus limiting the prediction power. This leads to the sixth hypothesis:

H6. A higher number of ordinal or discretized continuous levels for a negotiation issue will result in a lower counteroffer prediction error.

These hypotheses are tested with negotiation offers obtained from a large series of experiments in which a manufacturing negotiation case was used. The case and the pairwise treatment of issues are discussed in the following section.

4.4 Development of the predictive model

The feasibility of the pairwise modeling approach is investigated using past data collected by the Inspire system (Kersten & Noronha 1999). The Inspire negotiation system is a web-based system, which permits two parties located anywhere in the world to negotiate on a chosen case. Negotiation issues and issue options are specified in advance for a specific case and each negotiator specifies a rating for each issue, and additionally indicates a rating for each option of an issue. All issue ratings as well as all option ratings per issue should add up to 100. Each selected option rating is multiplied by the issue rating which permits the calculation of the total utility of a package. Package utility ratings are presented to the user for further adjustments since the user may feel that certain package utility ratings do not correctly reflect his or her preferences. Issue, option and package ratings are specific and confidential to each user, allowing private evaluation of all submitted or received offers. A negotiator has no information about the preference structure of the counterpart. As an additional information source, users may examine a graph of the utility of the history of offers and counteroffers constructed in their individual utility space.

The negotiation case selected is a simulated scenario where a seller and a buyer want to enter into a business relationship. Specifically, Itex manufacturing is entering the negotiations as a producer of bicycle gears, wishing to sell to Cypress Cycles; a bicycle producer. The case defines the market as competitive, meaning that either party may terminate the negotiations if they do not find the negotiations promising, since they can find other business partners. The issues include price (3.47$, 3.71$, 3.98$, 4.12$, 4.37$), delivery (20 days, 30 days, 45 days, 60 days), payment (60 days after delivery, 30 days after delivery, upon delivery) and returns (full price, 75% refund with 5% spoilage, 75% refund with 10% spoilage). Users may send offers and messages to their counterparts through the Inspire system as they work towards a compromise that is acceptable for both

parties. At any point, users can unilaterally terminate the negotiations. The Inspire's dataset provides rich information on the negotiations that took place using the system. The actual dataset used for the experiments contained a total of 6,310 offers placed by negotiators from over 100 different countries and the number of offers in each negotiation session varied with the median being 7 and the average 6.7 offers.

4.4.1 Pairwise observations

To describe the pairwise model, the negotiation information is divided into four high level categories. These categories represent the sets of variables included in the general pairwise counteroffer prediction model. The first two categories are the independent issue-specific information known before the counteroffer is received, and because the analysis is pairwise, it is composed of the primary issue and the secondary issue. The third is the independent general information about the negotiation session, which is known before the counteroffer is received. The fourth is the dependent issue information, which represents the counteroffer issue information to be modeled or predicted, and in the simplest case, this will be the value of the counteroffer for the primary issue of the pair. Using only the three general high-level categories that vary for each pairwise observation, Table XIX provides an overview of the sets of pairwise observations generated from one counteroffer for the Cypress/Itex negotiation case. Note that the independent general information category is not included in this table since it is always the same across all pairwise observation for one offer observation.

Table XIX. Overview of a set of pairwise observations for a counteroffer

Independent issue		Dependent issue	Prediction
Primary	**Secondary**		
Price	Delivery	Price	Price
	Returns	Price	
	Payment	Price	
Delivery	Price	Delivery	Delivery
	Returns	Delivery	
	Payment	Delivery	
Returns	Price	Returns	Returns
	Delivery	Returns	
	Payment	Returns	
Payment	Price	Payment	Payment
	Delivery	Payment	
	Returns	Payment	

Both the primary and the secondary independent issue information are composed of the same common abstracted variables but for two different issues. For example, we do not refer to the specific delivery issue level variable anymore since doing so would require that the model has a specific input variable for that issue, and thus it would not be flexible. The specific issues are abstracted from the variables of the pairwise model and subsequently only considered as the primary and the secondary issue level variables as per the pairwise permutation of the set of issues. Each of these variables is used to describe numerically the characteristics of the specified negotiation issue for a given session. These variables can be divided into two types of information: the negotiation session summary statistics and the timeseries data. There are five negotiation session summary statistics, including the first offer value for the issue; the minimum and maximum values achieved during the current negotiation session; the average, referring to a possible common ground; and the standard deviation describing how much variation there has been in the issue value. As for the timeseries data, the last buy and sell amounts provide the most recent historical information on the issue value, with this timeseries data potentially expandable to larger time windows. Consequently, with a time window that includes the current trial offer and the past buy and sell offer, there are eight independent issue variables for both the primary issue and the

secondary one for a total of 16 independent issue specific variables. Additional independent issue variables could be added to the model as could a larger time window for past issue values, without affecting the flexibility of the pairwise model.

The independent general information for the pairwise model attempts to capture some general characteristics of the current negotiation session. This includes the times of the; current trial offer, first offer, last buy offer, last sell offer and the average and standard deviation of the timestamps. These variables provide information on the negotiation offer timing. For example, if one party responds very quickly in comparison with the other, this party may be more anxious to reach an agreement, and thus may value faster resolution time at the expense of issue concessions. In addition, there is an indicator for differentiating the buyer from the seller since they are negotiating in opposite directions on the issues. This results in additional seven independent variables for a total of 23 independent variables.

For each pairwise permutation, the dependent variable is the next counteroffer value for the primary negotiation issue of the pair. All the above presented variables define the complete set of inputs (independent variables) and the output (dependent variable) for the pairwise model, an instance of which is referred to as a pairwise observation. Accordingly, the 23 inputs of the model are summarized in Table XX.

Table XX. Overview of inputs (independent variables)

Primary issue	Secondary issue	General
1. First amount	9. First amount	17. Current timestamp
2. Minimum amount	10. Minimum amount	18. First timestamp
3. Maximum amount	11. Maximum amount	19. Last buy timestamp
4. Average amount	12. Average amount	20. Last sell timestamp
5. St. dev. of amount	13. St. dev. of amount	21. Average timestamp
6. Trial amount	14. Trial amount	22. St. dev. of timestamps
7. Last buy amount	15. Last buy amount	23. Buyer/seller indicator
8. Last sell amount	16. Last sell amount	

4.4.2 Pairwise modeling approach

The input variables and the output variable for each pair permutation are assembled into one common dataset as presented in Figure 11 for the Cypress/Itex case where four sections are identified on the left side.

Section 1 contains the inputs (independent variables), which are combined in a pairwise fashion by negotiation issues, and for which the general information is always present. Information on each pair of issues is composed of both statistical and timeseries data. All inputs arrive only from information available at or before the offer is made, as indicated by "$t_{\leq 0}$". Section 2 represents the pairwise observations sets and since the Cypress/Itex case has four issues, the pair permutations result in 12 pairwise issue sets. The bold lines going into each pairwise set represent the primary independent issue and counteroffer (dependent variable) for this primary issue. Section 3 represents the counteroffer (dependent variable) for this primary issue and is included in each pairwise observation when training the model. Section 4 represents the assembly of all the pairwise observation sets into a combined dataset, in a way similar to data fusion. This combined dataset can be used for pairwise modeling of general counteroffer patterns.

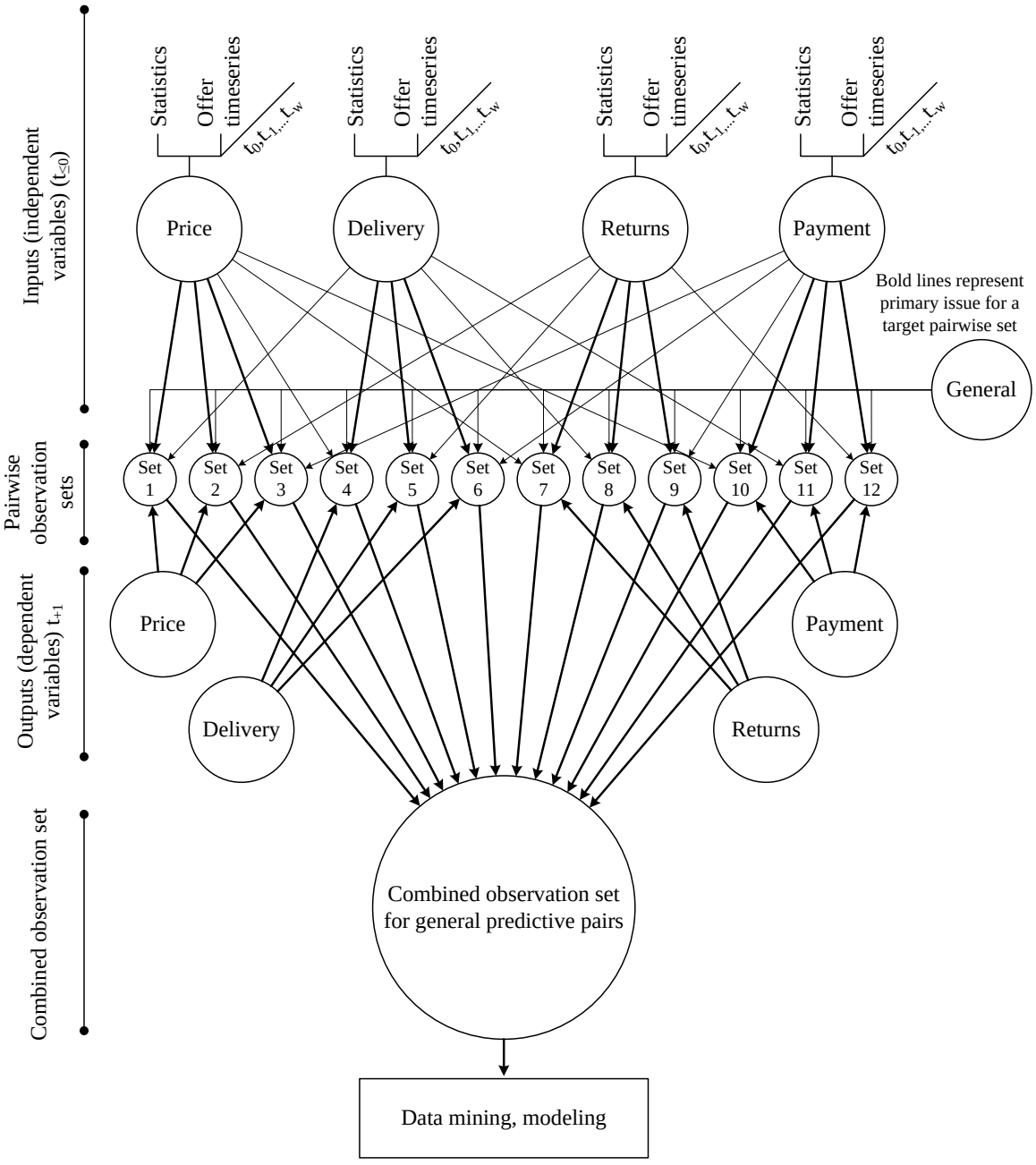

Figure 11 Pairwise model observation assembly for the Itex/Cypress case

Using this trained pairwise predictive model, a counteroffer can be estimated for a given trial offer. To perform this counteroffer prediction, the trial offer is broken down into its pairwise permutations. Each pairwise observation is predicted individually and the results

for an issue are averaged to provide the prediction for each issue of the given trial offer. This process, as applied to the Itex/Cypress case, is presented in Figure 12, where the issue and general inputs (independent variables) are processed through the pairwise modeling approach to provide a prediction for each issue. The pairwise permutations are expanded and each observation is processed through the common general pairwise predictive model. Although the pairwise model (PWM) appears 12 times in Figure 12, it is the *same* common general pairwise model that is used to predict the counteroffer for each pair. There is only one common general pairwise model representing the inferred patterns across all pairwise observations. An expected counteroffer for the primary issue of each observation is predicted for all permutations and these predictions are averaged for each issue, thus providing the overall prediction of the model.

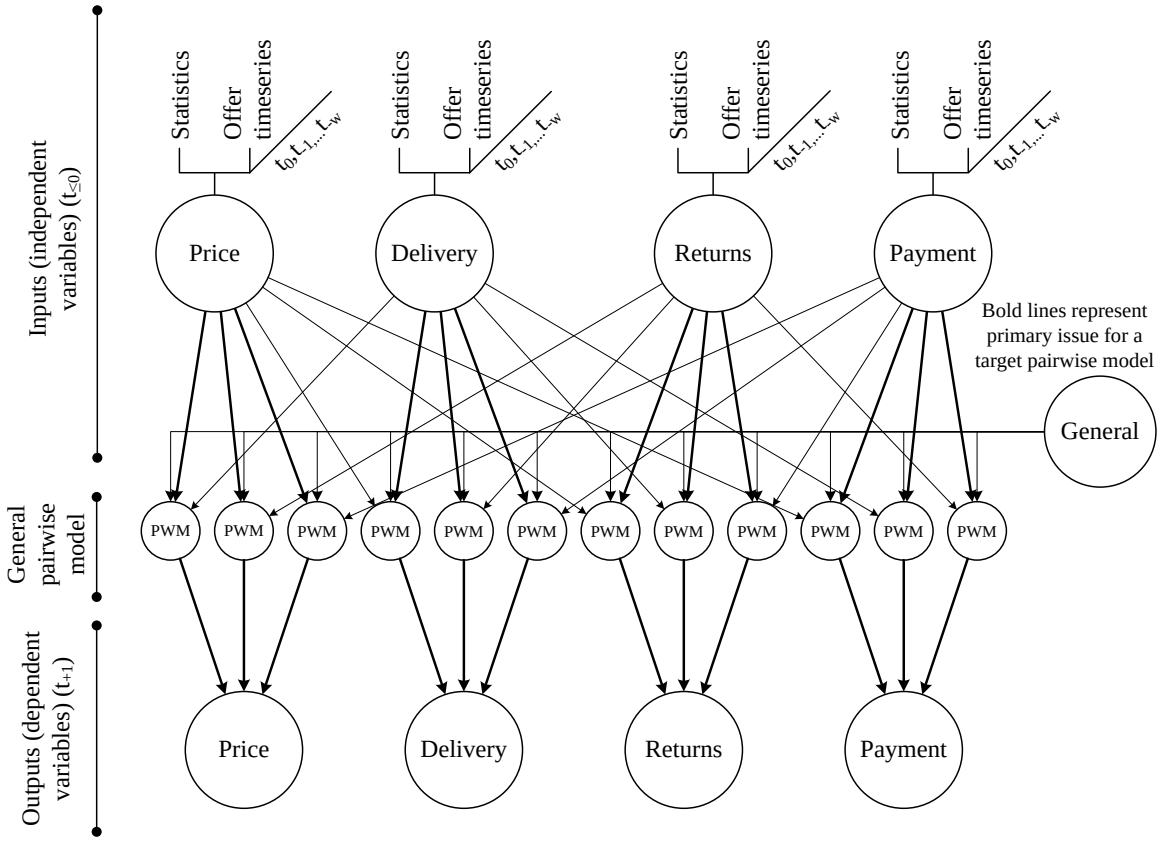

Figure 12 Processing counteroffer predictions with the pairwise model for the Itex/Cypress case

To achieve generality, the data from each issue must be standardized from its own range to a common one before being combined with the other issues, even if the model inference technique subsequently performs normalization across all variables. This will assist the inference of common patterns and will avoid inference of issues specific patterns that could be related to the range of the values.

4.5 Experiments

4.5.1 Prediction process

The starting point for the prediction process is a negotiation session that has multiple offers exchanged between a buyer and a seller, where each offer is for a set of negotiation issues. Either an offer will result in a counteroffer or the negotiation will end with an agreement or no agreement. The offer values for each issue are scaled to match a common distribution for all issues. An offer is then expanded into an observation for each pairwise permutation, which includes the relevant issue statistics, timeseries and general information. When the dataset is prepared for training the predictive model, the relevant counteroffer for the primary issue is added to each pairwise observation, whereas for prediction, the dependent variable is unknown and the expected value is predicted.

When the data is to be modeled using neural networks, each variable is scaled between 0 and 1 to be in the range of the transfer function and for the numerical stability of the computations. This is different from the issue scaling because there are some variables, such as timestamps, which are on completely different scales from the issues and must be within similar scales for many modeling algorithms. The scaled data is then used to train the neural network, and the results are then un-scaled back from the 0-1 range. Before both training and predicting, the data is scaled, however, during the first training, the scaling offset and multiplier are determined, and during predictions, the previously determined scaling parameters are re-used.

For prediction purposes, the issue counteroffer predictions are un-scaled back to their original range and averaged per issue. If the issue values are discretized continuous,

ordinal, categorical or binary, they are rounded to the appropriate level. The final counteroffer predictions are then processed based on the user's utility function to provide a counteroffer utility prediction for "what-if" analysis. This analysis could be repeated multiple times for optimizing the expected counteroffer based on the user's utility

4.5.2 Experimental setup

As defined previously in the hypothesis section, three reference models were included in the experiments as benchmarks for the pairwise counteroffer predictions. The first was the naïve model, which simply predicts the next counteroffer to be the same as the previous offer from the negotiation counterparty. This is the minimum benchmark reference and any proposed predictive model must outperform the naïve model. The second reference model was the full NN model (NNF) which considers all negotiation issues simultaneously and it was trained on the exact same Cypress/Itex case data (Carbonneau 2007; Carbonneau, Kersten & Vahidov 2008) as the pairwise model. The NN pairwise (NNP) predictive model would ideally perform as well as the NNF model, however, any prediction significantly better than the Naïve model would be acceptable, since the pairwise model can generalize to new issues the same way in which the Naïve model can, while the full model lacks such flexibility. The third benchmark was the linear version of the pairwise model, which was used to assess whether there is any advantage to the non-linear NN modeling. The linear version of the pairwise model is a multiple linear regression (MLR) model using the same pairwise independent and dependent variables (MLRP).

Finally, an experiment was performed which compared the ability of the pairwise model to predict new counteroffer issues. In this experiment, four different pairwise models were trained for the four subsets of three issues and used to predict all issues including the excluded issue. Thus, a pairwise model was trained for price, delivery and payment issues only, and was subsequently used to predict counteroffers for these issues as well as the excluded returns issues. The same was repeated for the three remaining subsets of three issues, each time excluding payment, delivery and price respectively and subsequently predicting for the excluded issue.

For the Cypress/Itex case, each issue was scaled from its own range to 0 to 1 and the standard deviation for each issue was also separately scaled from its range to 0 to 1. In the negotiation dataset, there were 6,310 offer observations that have a minimum of one previous buy and sell offer for the timeseries window. These 6,310 observations were then separated into training and testing sets. The training set contained 80% (5,048 observations) and the testing set contained the remaining 20% of the dataset (1,262 observations). Both the training and testing sets were expanded into the pairwise observation sets by the permutations of the issue pairs, consequently the resulting sets had 12 times more observations when including all four issues. The pairwise training set had 60,576 pairwise observations and the testing set had 15,144 pairwise observations, therefore there was a high ratio of observations to weights (60576/126=480.76) which probably helped reducing overfitting. Because the subset pairwise models were trained on only three issues instead of four, the offer observations expanded to only six pairwise issue permutations per offer, thus the pairwise training and testing sets had half the observations and the observation to weight ratio was also halved. All experiments and related modeling were performed in MATLAB 7.7 (The MathWorks Inc. 2008) and the neural networks were implemented using MATLAB's Neural Network Toolbox 6 (Demuth, Beale & Hagan 2008).

4.6 Results and discussion

Although the Levenberg-Marquardt with Bayesian Regularization algorithm for training neural networks can reduce overfitting, it is still useful to try different levels of network sizes. If the neural network is too small, it will not be able to learn all the patterns. On the other hand, if the neural network is too large, it will be slow to train and will tend to overfit data as compared to a smaller network. Within the range of 1 to 30 neurons (Figure 13), five neurons provided the lowest counteroffer prediction error on the test set, before rounding to ordinal issue levels. However, any number of neurons between three and ten resulted in similar performance. This optimization was performed on the pairwise model that included all four issues, and the selected network size of five neurons was used for the four issue pairwise model and three issue subset pairwise models.

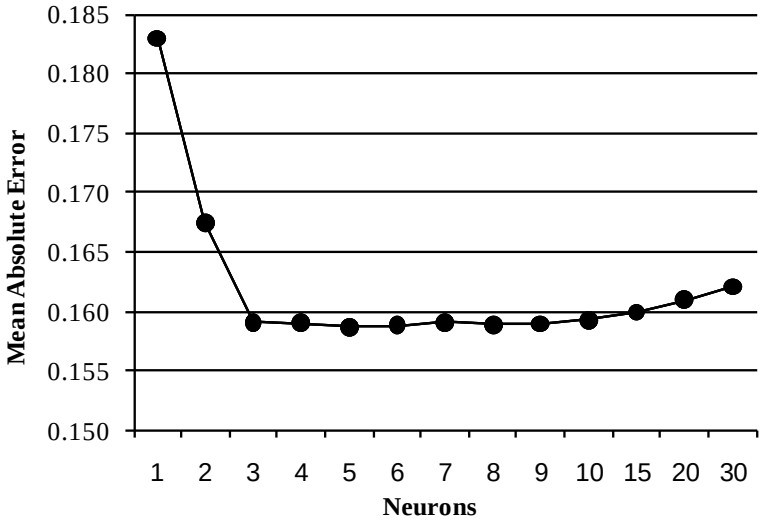

Figure 13 Average of 10 test set mean absolute error (MAE) by size of the neural network

Subsequently, a pairwise neural network model with five hidden layer neurons was trained and its performance was compared to the naïve model, the full NN model and the linear pairwise model. Additionally, the subset pairwise models were trained and subsequently tested on unseen data and compared to the naïve prediction and the pairwise model trained on all the issues. Lastly, the relationship between the number of issue levels and prediction performance was examined. The results are summarized in Table XXI where the percentage error is the mean absolute error (MAE) for an issue, divided by the number of levels for the issue, and then multiplied by 100. This scaling is necessary to make the measures comparable and to have an overall error that is not biased towards issues with more ordinal levels.

Table XXI. Counteroffer prediction error comparison

	Price	Delivery	Payment	Returns	Overall
Ordinal Levels (H6)	5	4	3	3	
Reference Models					
Naïve (H1, H4)	10.78%	11.89%	12.18%	13.47%	12.08%
Full ANN (NNF, H3)	8.29%	9.27%	9.54%	10.38%	9.37%
Pairwise Model – Trained on All Issues					
Pairwise Linear Model (MLRP, H2)	10.29%	10.24%	10.62%	11.54%	10.67%
Pairwise ANN (NNP, H1, H2, H3, H5, H6)	7.59%	9.11%	9.43%	10.88%	9.25%
Pairwise Models – Trained on Issue Subsets (* are for H4, H5)					
Pairwise ANN Excluding Price (NNP)	* 7.73%	9.17%	9.80%	10.99%	9.42%
Pairwise ANN Excluding Delivery (NNP)	7.69%	* 9.03%	9.64%	11.12%	9.37%
Pairwise ANN Excluding Payment (NNP)	7.64%	9.07%	* 9.93%	10.78%	9.35%
Pairwise ANN Excluding Returns (NNP)	7.59%	9.23%	9.72%	* 11.15%	9.42%

* Represents the results for the unseen issue.

The six proposed hypotheses were tested using t-tests based on the scaled prediction errors of each model. The tests were performed with the conservative assumption that the variance of each sample was different. Each test was performed on the predictions for the 1,262 test set offer observations and since there were four issues predicted, the total number of observations for each sample of the t-tests was 5,048.

H1. A non-linear pairwise model will have a lower counteroffer prediction error than the naïve model. (Supported)

The results indicate that the non-linear pairwise (NNP) model has a significantly lower counteroffer prediction error (9.25%, $p = 0.00$) than the naïve model (12.08%). This represents a 23.43% reduction in the error, or, inversely, the naïve model has an error that is 30.59% higher than the non-linear pairwise (NNP) model.

H2. A non-linear pairwise model will have a lower counteroffer prediction error than a linear pairwise model. (Supported)

The results indicate that the non-linear pairwise (NNP) model has a significantly lower counteroffer prediction error (9.25%, $p = 0.00$) than the linear pairwise (MLRP) model (10.67%). This represents a 13.31% reduction in the error, or inversely, the linear pairwise

(MLRP) model has an error that is 15.35% higher than the non-linear pairwise (NNP) model.

H3. A non-linear pairwise model will not have a higher counteroffer prediction error than a full simultaneous issue non-linear model. (Supported)

The results indicate that the non-linear pairwise (NNP) model does not have a higher counteroffer prediction error (9.25%, $p = 0.00$) than the full simultaneous issue non-linear (NNF) model (9.37%). In other words, the less powerful but more flexible pairwise model does not come at the expense of prediction performance. This may represent a lack of patterns that are specific to an issue or a lack of patterns involving interactions between more than two issues. On the other hand, it may be that some modeling accuracy of more complex or issue specific patterns are lost, while at the same time offset by a better generalization because of an increased number of observations and/or because the data is confounded across issues. Either way, this seems to be a very promising result, as it suggests the possibility of promoting generality without significant loss of accuracy. It may even be possible to identify a general descriptive model of the patterns irrespective of the negotiation issues.

H4. A non-linear pairwise model will have a lower counteroffer prediction error on new unseen issues than the naïve model. (Supported)

To test hypotheses 4 and 5, the prediction errors for the four issues when they are excluded from the training of their respective model (indicated with an asterisk in the Table XXI) are compared with the prediction errors of the reference model. For example, the error of predicting a price counteroffer with the model trained without the price issue is compared with the error of predicting a price counteroffer with a reference model. The average prediction error of each issue when excluded from training the model (7.73%, 9.03%, 9.93% and 11.15%) is 9.46%. The results indicate that the non-linear pairwise (NNP) model predicted counteroffers for new issues with a significantly lower error (9.46%, $p = 0.00$) than the naïve prediction (12.08%). This represents a 21.69% decrease in the error, or inversely, the naïve model has an error which is 27.70% higher. It may also be

interesting to estimate the prediction accuracy increase attributed specifically to general patterns that are more complicated than the naïve model (12.08%) up to and including pairwise patterns (9.25%), representing a 2.83% (12.08% - 9.25%) decrease in the prediction error. Defining a perfect model as one with no error (0%), it can be quantified that 23.43% (2.83%/12.08%) of the change in a counteroffer can be effectively predicted by the pairwise model using only patterns common to all issues.

H5. A non-linear pairwise model will not have a higher counteroffer prediction error on new unseen issues than a non-linear pairwise model on previously modeled issues. (Supported)

The results indicate that the non-linear pairwise (NNP) model with issues omitted during training does not have a significantly higher counteroffer prediction error (9.46%, $p = 0.00$) than the pairwise (NNP) model trained on all the issues (9.25%).

H6. A higher number of ordinal or discretized continuous levels for a negotiation issue will result in a lower counteroffer prediction error. (Supported)

The regression line for scaled errors (non-linear pairwise NNP) based on the number of discrete levels has an intercept of 0.15247 and a slope of -0.01646 ($p = 0.00$). Accordingly, for every increase in the number of ordinal levels, there is a decrease of 1.6% in the counteroffer prediction error for the observed range of 3, 4 and 5 discrete levels. This trend probably levels off as the number of discrete or fixed continuous levels becomes high.

4.7 Conclusions

The results of this research indicate that formulating a flexible pairwise counteroffer prediction model is technically feasible. The pairwise counteroffer model prediction error is shown not to be significantly higher than that of the equivalent model which considers all issues simultaneously. Furthermore, the error does not worsen for the prediction of new issues, which were excluded during the model training. It has also been demonstrated that the pairwise neural network based model outperforms the reference naïve and linear pairwise models. These results suggest that such non-linear pairwise counteroffer

prediction models may provide a flexible way to predict counteroffers in real life situations to permit trial offer what-if analysis and trial offer optimization.

The particular importance of the findings is in the possibility of building relatively context-independent models for counteroffer prediction. First, in a given negotiation case, although some of the issues may be added or dropped during the course of negotiations, the model would still be able to make predictions. Second, the "unbinding" of the specific issues from the model inputs promote the applicability of the model to other similar negotiation cases. In this regard, these other cases should be similar in certain respects to the one on which the model had been trained. In particular, we anticipate that if some other case has similar relationships between the pairs of issues (in terms of preferences and trade-offs) then the model should also perform well for this new case. Thus, the model would be applicable to a family of cases, and its performance would degrade gracefully as the characteristics of the case deviate significantly from those of the case utilized for model training.

Beyond proposing an approach for building predictive models, these results (especially hypothesis 3 and 5) might inform future descriptive models because they demonstrate that a portion of the relationships which describe counteroffer patterns are general across different issues. For the current dataset under study, at least 23.43% of the change in a counteroffer can be effectively predicted by relationships common to all issues.

Additionally, this research identifies that negotiation issues with a higher number of ordinal levels or discretized continuous levels provide additional information and larger range of predictions, which in turn permit better counteroffer modeling, at least by certain prediction models. Although humans may be more limited by bounded rationality (Simon 1982) than software systems are by computation limitations, it is possible that higher numbers of issue levels also provide humans with a richer negotiation environment allowing for better human based counteroffer modeling.

The results could allow equipping e-negotiation system users with the powerful tools to perform "what-if" analysis before submitting an offer in order to predict the possible

counteroffer. Moreover, an optimization technique could be used in conjunction with such a counteroffer prediction model to suggest the offer that maximizes the expected utility of the subsequent counteroffer. The model can also be incorporated in software agents to automate negotiations for one of the parties involved.

While the pairwise counteroffer modeling demonstrated potential for predicting counteroffers for various different issues and new unseen issues, there are some limitations. Firstly, the modeling of binary and categorical issues has not been explored in this paper. Additional work is also needed to examine if ordinal and especially binary and categorical negotiation issues should be processed in a different way or in separate predictive models. Working with only the changes or relative changes of issue levels may provide a more common range and distribution. More diverse sets of issues might also require research into improved methods for scaling the issues to a similar distribution. Secondly, as with any data mining or statistical model, the data should come from a representative sample of negotiation cases in terms of number of issues, the issues themselves, the style of negotiation case and the population of negotiators.

Nevertheless, the evidence we obtained from the present work is encouraging as it demonstrates the feasibility of predicting issue values without binding those issues to the particular inputs of the model, and making successful predictions for a new issue, albeit within the context of the same case. Future work should be directed towards using data collected on other negotiation cases for training the pairwise network, in order to examine the generality of the model across different cases and categories of cases.

CONCLUSION GÉNÉRALE

Considérant l'importance accrue de l'exploitation de données, les résultats de recherche présentés dans cette thèse apportent des avancements pour l'optimisation exacte du modèle de régression par classe et pour l'application de réseaux de neurones à la prédiction des contre-offres en négociation.

Pour le modèle de régression par classe, la présente recherche développe une formulation par programmation logique-quadratique mixte avec implication de contrainte ainsi qu'une formulation quadratique traditionnelle avec grand-M. Il est démontré que la formulation par grand-M ne peut garantir l'optimalité car les coefficients, et donc les erreurs, peuvent êtres arbitrairement grands. Ensuite, une méthode d'optimisation par séparation et évaluation progressive est développée. Cette dernière offre des gains de performances considérables et permet par conséquent l'optimisation exacte de plus grands ensembles de données. La méthode développée est une extension de l'optimisation par séparation et évaluation progressive répétitive de Brusco (*repetitive branch and bound algorithm*, RBBA) (Brusco & Stahl 2005; Brusco 2006).

Finalement, pour des problèmes où le nombre de classes est élevé, une méthode par génération de colonne est développée. Elle contient plusieurs extensions, principalement des heuristiques et des techniques d'optimisation par séparation et évaluation progressive qui ont étés développées dans le deuxième article. Cette recherche démontre la faisabilité de l'optimisation exacte pour la régression par classe, ainsi que des améliorations considérables des temps d'optimisation. Elle offre aussi plus d'information sur le comportement des diverses composantes d'optimisation. De plus, ce travail de recherche démontre et développe davantage l'utilisation réussie du nouveau paradigme d'optimisation de sous-ensembles de fin progressivement plus grands, tel qu'innové par l'optimisation par séparation et évaluation progressive répétitive de Brusco (*repetitive branch and bound algorithm*, RBBA).

Pour l'application des techniques d'exploitation de données, la présente recherche développe une stratégie de modélisation des contre-offres en négociation considérablement plus flexible que le modèle antérieur. Cette nouvelle stratégie repose sur la prédiction de chaque question de négociations en paires avec chacune des autres questions de négociation. Ceci ouvre la possibilité de développer des modèles flexibles et génériques pour la prédiction des contre-offres quand les cas de négociations sont similaires. Cet avancement en termes de flexibilité est fait sans réduire la qualité des prédictions, et permet donc l'utilisation de plus grands ensembles de données et son application dans des environnements de négociations plus dynamiques.

En conclusion, cette thèse présente des avancements dans le domaine de l'optimisation et de l'application de modèles d'exploitation de données.

BIBLIOGRAPHIE

Abidi, M. A. and R. C. Gonzalez (1992). Data fusion in robotics and machine intelligence. San Diego, CA, USA, Academic Press Professional.

Aizen, M. A. and W. A. Patterson, III (1990). "Acorn size and geographical range in the North American oaks (Quercus L.)." Journal of Biogeography **17**(3): 327-332.

Aloise, D., S. Cafieri, et al. (2010). "Column generation algorithms for exact modularity maximization in networks." Physical Review E **82**(4): 046112.

Aloise, D., P. Hansen, et al. (2010). "An improved column generation algorithm for minimum sum-of-squares clustering." Mathematical Programming: 1-26.

Aurifeille, J. M. (2000). "A bio-mimetic approach to marketing segmentation: Principles and comparative analysis." European Journal of Economic and Social Systems **14**(1): 93-108.

Aurifeille, J. M. and C. Medlin (2001). "A dyadic segmentation approach to business partnerships." European Journal of Economic and Social Systems **15**(2): 3-16.

Aurifeille, J. M. and P. G. Quester (2003). "Predicting business ethical tolerance in international markets: a concomitant clusterwise regression analysis." International Business Review **12**(2): 253-272.

Barnhart, C., E. L. Johnson, et al. (1998). "Branch-and-price: Column generation for solving huge integer programs." Operations Research: 316-329.

Bartolini, C., C. Priest, et al. (2005). A Software Framework for Automated Negotiation. Software Engineering for Multi-Agent Systems III: Research Issues and Practical Applications. R. Choren, A. Garcia, C. Lucena and A. Ramonovsky. New York, Springer Verlag**:** 213–235.

Beam, C. and A. Segev (1997). "Automated Negotiations: A Survey of the State of the Art." Wirtschaftsinformatik **39**(3): 263-268.

Berkelaar, M., K. Eikland, et al. (2010). lp_solve: Open source (Mixed-Integer) Linear Programming system.

Bradley, R. A. and E. T. Milton (1952). "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons." Biometrika **39**(3/4): 324-345.

Breiman, L. (2001). "Random Forests." Machine Learning **45**(1): 5-32.

Brusco, M. J. (2006). "A repetitive branch-and-bound procedure for minimum within-cluster sums of squares partitioning." Psychometrika **71**(2): 347-363.

Brusco, M. J., J. D. Cradit, et al. (2008). "Cautionary Remarks on the Use of Clusterwise Regression." Multivariate Behavioral Research **43**(1): 29-49.

Brusco, M. J. and S. Stahl (2005). Branch-and-bound applications in combinatorial data analysis, Springer Verlag.

Brzostowski, J. and R. Kowalczyk (2006). Adaptive Negotiation with On-Line Prediction of Opponent Behaviour in Agent-Based Negotiations. International Conference on Intelligent Agent Technology, Hong Kong, China.

Caporossi, G. and P. Hansen (2005). "Variable Neighborhood Search for Least Squares Clusterwise Regression." Les Cahiers du GERAD **G-2005**(61): 18.

Carbonneau, R. (2007). Machine Learning Algorithms for Negotiation Counter-Offer Modeling. Proceedings of the Group Decision and Negotiation (GDN) 2007, Mt. Tremblant, Quebec, Canada.

Carbonneau, R., G. E. Kersten, et al. (2008). "Predicting opponent's moves in electronic negotiations using neural networks." Expert Systems with Applications **34**(2): 1266-1273.

Carbonneau, R., R. Vahidov, et al. (2008). Forecasting Supply Chain Demand Using Machine Learning Algorithms. Distributed Artificial Intelligence, Agent Technology, and Collaborative Applications. V. Sugumaran. Hershy, PA, Information Science Reference**:** 328-365.

Carbonneau, R. A., G. Caporossi, et al. (2011). "Extensions to the Repetitive Branch and Bound Algorithm for Globally Optimal Clusterwise Regression." GERAD **G-2011-10**: 27.

Carbonneau, R. A., G. Caporossi, et al. (2011). "Globally Optimal Clusterwise Regression by Mixed Logical-Quadratic Programming." European Journal of Operational Research **212**(1): 213-222.

Carbonneau, R. A., G. E. Kersten, et al. (2011). "Pairwise issue modeling for negotiation counteroffer prediction using neural networks." Decision Support Systems **50**(2): 449-459.

Charles, C. (1977). Régression typologique et reconnaissance des formes. Paris Université de Paris IX. **Thèse de doctorat 3ième cycle**.

Chavez, A., D. Dreilinger, et al. (1997). A Real-life Experiment in Creating an Agent Marketplace. Software Agents and Soft Computing. H. S. Nwana and N. Azarmi. Berlin, Springer**:** 160-179.

Chavez, A. and P. Maes (1996). Kasbah: An Agent Marketplace for Buying and Selling Goods. First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, Practical Application Company.

Chen, E., G. E. Kersten, et al. (2004). An E-marketplace for Agent-supported Commerce Negotiations. 25th McMaster World Congress Management of Electronic Business, Hamilton, Ontario, Canada.

Chen, E., R. Vahidov, et al. (2005). "Agent-supported negotiations in the e-marketplace." International Journal of Electronic Business **3**(1): 28-49.

Choi, S. P. M., J. Liu, et al. (2001). "A genetic agent-based negotiation system." Computer Networks **37**(2): 195-204.

Ciampi, A., B. Rich, et al. (2007). Locally Linear Regression and the Calibration Problem for Micro-Array Analysis. Selected Contributions in Data Analysis and

Classification. P. Brito, G. Cucumel, P. Bertrand and F. de Carvalho. Berlin, Springer**:** 549-555.

Cox, D. R. and E. J. Snell (1981). Applied statistics : principles and examples. London, Chapman and Hall.

Dantzig, G. B. and P. Wolfe (1960). "Decomposition principle for linear programs." Operations Research **8**(1): 101-111.

de Figueiredo, R. J. P. (1980). "Implications and applications of Kolmogorov's superposition theorem." IEEE Transactions on Automatic Control **25**(6): 1227–1231.

Demuth, H., M. Beale, et al. (2008). Neural Network Toolbox 6 User's Guide. Natick, MA, The MathWorks Inc.

DeSarbo, W. (1988). "A maximum likelihood methodology for clusterwise linear regression." Journal of Classification **5**(2): 249-282.

DeSarbo, W., R. Oliver, et al. (1989). "A simulated annealing methodology for clusterwise linear regression." Psychometrika **54**(4): 707-736.

Desaulniers, G., J. Desrosiers, et al. (2005). Column generation, Springer Verlag.

Diday, E. (1979). Optimization en Classification Automatique. Le Chesnay, INRIA.

Faratin, P., C. Sierra, et al. (2002). "Using similarity criteria to make issue trade-offs in automated negotiations." Artificial Intelligence **142**(2): 205-237.

Finkelstein, M. O. and B. A. Levin (2001). Statistics for lawyers. New York, Springer.

Fisher, R. (1936). "The use of multiple measurements in taxonomic problems." Annals Eugen. **7**: 179-188.

Foresee, F. D. and M. T. Hagan (1997). "Gauss-Newton approximation to Bayesian regularization." Proceedings of the 1997 International Joint Conference on Neural Networks **3**: 1930-1935.

Fraumeni, J. F., Jr. (1968). "Cigarette smoking and cancers of the urinary tract: geographic variation in the United States." Journal of the National Cancer Institute **41**(5): 1205-1211.

Gentleman, W. M. (1973). "Least Squares Computations by Givens Transformations Without Square Roots." IMA Journal of Applied Mathematics **12**(3): 329-336.

Gentleman, W. M. (1974). "Algorithm AS 75: Basic Procedures for Large, Sparse or Weighted Linear Least Problems." Applied Statistics **23**(3): 448-454.

Hagan, M. T., H. B. Demuth, et al. (1996). Neural Network Design. Boston, MA, PWS Publishing.

Hagan, M. T. and M. B. Menhaj (1994). "Training feedforward networks with the Marquardt algorithm." IEEE Transactions on Neural Networks **5**(6): 989–993.

Hall, D. and J. Llinas (1997). "An introduction to multisensor data fusion." Proceedings of the IEEE **85**(1): 6-23.

Hand, D. J. (1981). "Branch and bound in statistical data analysis." The Statistician: 1-13.

Hand, D. J. (1981). "Discrimination and classification." Wiley Series in Probability and Mathematical Statistics, Chichester: Wiley, 1981.

Hand, D. J. (1994). A handbook of small data sets. London, Chapman & Hall.

Hansen, L. and P. Salamon (1990). "Neural network ensembles." IEEE Transactions on Pattern Analysis and Machine Intelligence **12**(10): 993-1001.

Hansen, P. and C. Meyer (2011). "A new column generation algorithm for Logical Analysis of Data." Annals of Operations Research **188**(1): 215-249.

Hardle, W. and T. M. Stoker (1989). "Investigating Smooth Multiple Regression by the Method of Average Derivatives." Journal of the American Statistical Association **84**(408): 986-995.

Hastie, T. and R. Tibshirani (1990). Generalized additive models. London, Chapman and Hall.

Hastie, T. and R. Tibshirani (1998). "Classification by Pairwise Coupling." The Annals of Statistics **26**(2): 451-471.

Haykin, S. (1998). Neural networks : a comprehensive foundation. Upper Saddle River, N.J. , Prentice Hall.

Heavenrich, R. M., J. D. Murrell, et al. (1991). Light-duty automotive technology and fuel-economy trends through 1991. Ann Arbor, U.S. Environmental Protection Agency.

Hennig, C. (2000). "Identifiablity of Models for Clusterwise Linear Regression." Journal of Classification **17**(2): 273-296.

Hennig, C. (2002). "Fixed Point Clusters for Linear Regression: Computation and Comparison." Journal of Classification **19**(2): 249-276.

Hooker, J. N. (2002). "Logic, optimization, and constraint programming." INFORMS Journal on Computing **14**(4): 295-321.

Hooker, J. N. (2007). Integrated methods for optimization. New York, Springer.

Hooker, J. N. and M. A. Osorio (1999). "Mixed logical-linear programming." Discrete Applied Mathematics **96**: 395-442.

Hooker, J. N., G. Ottosson, et al. (2000). "A scheme for unifying optimization and constraint satisfaction methods." The Knowledge Engineering Review **15**(1): 11-30.

Hornik, K. (1991). "Approximation capabilities of multilayer feedforward networks." Neural Networks **4**(2): 251-257.

Hou, C. (2004). Predicting agents tactics in automated negotiation. International Conference on Intelligent Agent Technology, Beijing, China.

IBM (2009). IBM ILOG CPLEX 12.1.0. Armonk, NY, IBM.

IBM (2009). IBM ILOG OPL 6.3. Armonk, NY, IBM.

Jelassi, M. T. and A. Foroughi (1989). "Negotiation support systems: an overview of design issues and existing software." Decision Support Systems **5**(2): 167-181.

Jennings, N. R., P. Faratin, et al. (2001). "Automated Negotiation: Prospects, Methods and Challenges." Group Decision and Negotiation **10**(2): 199-215.

Karp, R. M. (1972). Reducibility among combinatorial problems. Complexity of computer computations. R. E. Miller and J. W. Thatcher. New York, Plenum Press**:** 85–103.

Kersten, G. E. and H. Lai (2007). "Negotiation Support and E-negotiation Systems: An Overview " Group Decision and Negotiation **16**(6): 553-586.

Kersten, G. E. and G. Lo (2003). "Aspire: an integrated negotiation support system and software agents for e-business negotiation." International Journal of Internet and Enterprise Management **1**(3): 293-315.

Kersten, G. E. and S. J. Noronha (1999). "WWW-based negotiation support: design, implementation, and use " Decision Support Systems **25**(2): 135-154.

Kilgour, D. M. and C. Eden (2010). Handbook on Negotiations. New York, Springer.

Knuth, D. E. (1997). Art of Computer Programming, Volume 2: Seminumerical Algorithms. Art of Computer Programming, Addison-Wesley Professional**:** 784.

Kojima, M., S. Mizuno, et al. (1989). "A polynomial-time algorithm for a class of linear complementarity problems." Mathematical Programming **44**(1): 1-26.

Koontz, W. L. G., P. M. Narendra, et al. (1975). "A branch and bound clustering algorithm." IEEE Transactions on Computers **100**(24): 908-915.

Kwon, O., J. M. Shin, et al. (2006). "Context-aware multi-agent approach to pervasive negotiation support systems." Expert Systems with Applications **31**(2): 275-285.

Lange, T. R., H. E. Royals, et al. (1993). "Influence of Water Chemistry on Mercury Concentration in Largemouth Bass from Florida Lakes." Transactions of the American Fisheries Society **122**(1): 74-84.

Lau, K. N., P. L. Leung, et al. (1999). "A mathematical programming approach to clusterwise regression model and its extensions." European Journal of Operational Research **116**(3): 640-652.

Lee, W.-P. (2004). "Towards agent-based decision making in the electronic marketplace: interactive recommendation and automated negotiation." Expert Systems with Applications **27**(4): 665-679.

Lübbecke, M. E. and J. Desrosiers (2005). "Selected Topics in Column Generation." Operations Research **53**(6): 1007-1023.

Lustig, I. J. (1990). "Feasibility issues in a primal-dual interior-point method for linear programming." Mathematical Programming **49**(1): 145-162.

MacKay, D. J. C. (1992). "Bayesian Interpolation." Neural Computation **4**(3): 415-447.

Maes, P., R. Guttman, et al. (1999). "Agents That Buy and Sell." Communications of the ACM **42**(3): 81-91.

Matwin, S., T. Szapiro, et al. (1991). "Genetic algorithms approach to a negotiation support system." IEEE Transactions on Systems, Man and Cybernetics **21**(1): 102-114.

McCormick, R. E. (1993). Managerial economics. Englewood Cliffs, Prentice Hall.

Megiddo, N. (1989). Progress in mathematical programming : interior-point and related methods. New York, Springer.

Miles, J. and M. Shevlin (2001). Applying regression & correlation : a guide for students and researchers. London, Sage Publications.

Miller, A. J. (1992). "Algorithm AS 274: Least Squares Routines to Supplement Those of Gentleman." Applied Statistics **41**(2): 458-478.

Miller, A. J. (1994). "Correction to Algorithm AS 274: Least Squares Routines to Supplement those of Gentleman." Applied Statistics **43**(4): 678.

Mirkin, B. (2005). Clustering for Data Mining. London, Chapman & Hall/CRC.

Monteiro, R. D. C. and I. Adler (1989). "Interior path following primal-dual algorithms. Part I: Linear programming." Mathematical Programming **44**(1): 27-41.

Moore, D. S. and G. P. McCabe (1998). Introduction to the Practice of Statistics. New York, W.H. Freeman.

Mudgal, C. and J. Vassileva (2000). Bilateral Negotiation with Incomplete and Uncertain Information: A Decision-Theoretic Approach Using a Model of the Opponent. Proceedings of the 4th International Workshop on Cooperative Information Agents IV, The Future of Information Agents in Cyberspace M. Klusch and L. Kerschberg. London, UK, Springer-Verlag**:** 107-118.

Ou, G. and Y. L. Murphey (2007). "Multi-class pattern classification using neural networks." Pattern Recognition **40**(1): 4-18.

Peixoto, J. L. (1990). "A Property of Well-Formulated Polynomial Regression Models." The American Statistician **44**(1): 26-30.

Ryan, D. M. and B. A. Foster (1981). "An integer programming approach to scheduling." Computer scheduling of public transport urban passenger vehicle and crew scheduling: 269–280.

Ryan, D. M. and B. A. Foster (1981). An integer programming approach to scheduling. Computer scheduling of public transport urban passenger vehicle and crew scheduling, North-Holland**:** 269–280.

Saha, S., A. Biswas, et al. (2005). Modeling opponent decision in repeated one-shot negotiations. Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, Utrecht, Netherlands.

Sahinidis, N. V. (1996). "BARON: A general purpose global optimization software package." Journal of Global Optimization **8**(2): 201-205.

Simon, H. A. (1982). Models of bounded rationality. Cambridge, Mass., MIT Press.

Sochett, E. B., D. Daneman, et al. (1987). "Factors affecting and patterns of residual insulin secretion during the first year of type 1 (insulin-dependent) diabetes mellitus in children." Diabetologia **30**(7): 453-459.

Späth, H. (1979). "Algorithm 39 Clusterwise linear regression." Computing **22**(4): 367-373.

Späth, H. (1981). "Correction to algorithm 39 clusterwise linear regression." Computing **26**(3): 275-275.

Späth, H. (1982). "A fast algorithm for clusterwise linear regression." Computing **29**(2): 175-181.

Spedicato, E. (1994). Algorithms for continuous optimization--the state of the art. Dordrecht, Kluwer Academic.

Tawarmalani, M. and N. Sahinidis (2004). "Global optimization of mixed-integer nonlinear programs: A theoretical and computational study." Mathematical Programming **99**(3): 563-591.

Tawarmalani, M. and N. V. Sahinidis (2002). Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications. Dordrecht, Kluwer Academic.

The MathWorks Inc. (2008). MATLAB 7.7 (Release 2008b). Natick, MA, The MathWorks Inc.

Thurstone, L. L. (1994). "A law of comparative judgement." Psychological Review **101**(2): 266-270.

Torra, V. (2003). Information fusion in data mining. Berlin, Springer-Verlag.

Van Hentenryck, P., I. Lustig, et al. (1999). The OPL optimization programming language. Cambridge, MIT Press.

Vandaele, W. (1978). Participation in illegitimate activities: Ehrlich revisited. Deterrence and incapacitation : estimating the effects of criminal sanctions on crime rates. A. Blumstein, J. Cohen and D. Nagin. Washington, National Academy of Sciences**:** 270–335.

Vapnik, V. N. (2000). The nature of statistical learning theory. New York, Springer.

Velleman, P. F. (2010). DASL, the Data and Story Library.

Waltz, E. and J. Llinas (1990). Multisensor data fusion. Norwood, MA, Artech House.

Wedel, M. (1990). Clusterwise regression and market segmentation. Developments and applications. Wageningen, Landbouwuniversiteit Wageningen. **Doctoral thesis**.

Wedel, M. (1998). GLIMMIX: Simultaneous estimation of latent classes and generalized models within each latent class, User's Manual, version 1.0. Groningen, ProGAMMA.

Wedel, M. and W. S. DeSarbo (1995). "A mixture likelihood approach for generalized linear models." Journal of Classification **12**(1): 21-55.

Zeng, D. and K. Sycara (1998). "Bayesian learning in negotiation." International Journal of Human-Computer Studies **48**(1): 125-141.