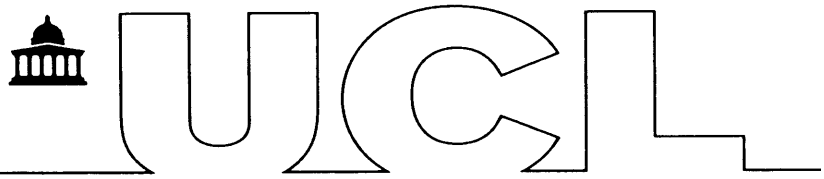


Optimisation-based Approaches for Data Analysis

by

Gang Xu

A thesis submitted for the degree of

Doctor of Philosophy

of

University College London

Department of Chemical Engineering

University College London

London WC1E 7JE

May 2008

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

I, Gang Xu, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Recent advances in science and technology promote the generation of a huge amount of data from various sources including scientific experiments, social surveys and practical observations. The availability of powerful computer hardware and software offers easier ways to store datasets. However, more efficient and accurate methodologies are required to analyse datasets and extract useful information from them. This work aims at applying mathematical programming and optimisation methodologies to analyse different forms of datasets. The research focuses on three areas including data classification, community structure identification of complex networks and DNA motif discovery.

Firstly, a general data classification problem is investigated. A mixed integer optimisation-based approach is proposed to reveal the patterns hidden behind training data samples using a hyper-box representation. An efficient solution methodology is then developed to extend the applicability of hyper-box classifiers to datasets with many training samples and complex structures.

Secondly, the network community structure identification problem is addressed. The proposed mathematical model finds optimal modular structures of complex networks through the maximisation of network modularity metric. Communities of medium/large networks are identified through a two-stage solution algorithm developed in this thesis.

Finally, the third part presents an optimisation-based framework to extract DNA motifs and consensus sequences. The problem is formulated as a mixed integer linear programming model and an iterative solution procedure is developed to identify multiple motifs in each DNA sequence. The flexibility of the proposed motif finding approach is then demonstrated to incorporate other biological features.

Acknowledgments

I take this opportunity to thank everyone who made the completion of this thesis possible.

First, I would like to thank my supervisor Dr Lazaros Papageorgiou for introducing me to the fascinating world of mathematical programming and optimisation theory. His excellent supervision, contribution and guidance throughout my PhD studies are more than appreciated.

Financial support from the Overseas Research Students (ORS) Award Scheme, the Centre for Process Systems Engineering, Royal Academy of Engineering and UCL Graduate School is also gratefully acknowledged.

Many thanks to Prof. David Bogle and Prof. Eric Fraga, for valuable helps, comments and discussions. Special appreciation goes to Dr. Sophia Tsoka for her always friendly and resourceful helps of my research.

All the past and present members of the Centre for Process Systems Engineering, UCL Department of Chemical Engineering for their understanding and support.

My partner and best friend, Shao Nan, thank you so much for sharing my life all these years, your patience, love and understanding are my most prized possessions.

Finally, I would like to thank my parents Xu zhenli and Yang liying for providing me with their unconditional love and constant support.

Table of Contents

PART III: DNA SEQUENCE ANALYSIS

List of Figures

List of Tables

Chapter 1

Introduction

Learning from data has been considered as one of the most promising research avenues to recognise patterns and extract knowledge. The generation of complex datasets requires more efficient and accurate data analysis methodologies to find knowledge/patterns. Mathematical programming techniques and optimisation theory have been recognised as a very fertile research area with a wide range of applications in science, engineering and business. Mathematical programming has also made a vital contribution to the field of process systems engineering since the pioneering work of Prof. Roger Sargent (1977). Due to the availability of a number of modelling software and the latest developments of solution algorithms together with the upgrade of computing facilities, mathematical programming techniques have been applied extensively to build up and solve models to analyse real world systems from the optimisation point of view.

1.1 Research Developments and Challenges of Data Analysis in the 21st Century

The 21st century is undoubtedly a data explosive era. Scientific and technological advances together with the development of computer hardware and software significantly accelerate the generation, collection and storage of data from a variety of sources including research experiments, social survey, financial markets and so on. Since data is the fundamental form of information that needs to be managed, mined and interpreted to extract knowledge, discovering patterns from data becomes one of the major challenges of the information age.

During the last 30 years, data analysis has attracted research communities from different disciplines due to the increasing demands of data mining and pattern discovery from both academia and industry. The research developments of data analysis have mainly been driven by the following two areas:

- *Information Technology Developments*

  Current development of computer hardware especially the availability of large volume hard disks and their drastically dropping costs make the storage of massive data possible and cheap. Internet and parallel distributed computing facilities play an important role as an infrastructure to provide a global connection force so that datasets can be shared publicly and analysed simultaneously. On the other hand, database models and software tools facilitate data collection and storage in a structural and efficient way (Haughton *et al.*, 2003).

- *Research Developments of Data Analysis Methodologies*

  The explosive growth of data and databases has generated an urgent need for novel methodologies that automatically and intelligently transform data into knowledge. The research advances of data analysis, data mining, pattern recognition and machine learning are contributed by not only mathematicians and computer scientists, but also engineers from both academia and industry. Diversified data analysis demands from engineering, banking, bioinformatics, healthcare etc. have driven the researchers from different backgrounds to

develop novel data analysis methodologies (Bishop, 2006). The underlying knowledge extracted through those methodologies has successfully facilitated people's understanding of complex systems and decision making.

However, we are inevitably facing a number of research challenges of data analysis:

- *The Increase of Data Scales and Complexity*

  The success of data collection and storage dramatically increase the number of available datasets and the dimensions associated with each sample. Such advances inevitably increase the complexity of extracting information from data. Therefore, analysing data in a more efficient way by extending current data analysis methodologies is crucially important to researchers from various disciplines.

- *Development of Novel Data Analysis Approaches to New Data Types*

  Today, most data analysis methods focus on data with real-value matrix forms and continuous distributions. However, datasets with non-conventional types have been rapidly accumulated including data samples distributed in disjointed regions, network topology datasets, sequence/string datasets and time series datasets. Traditional data analysis approaches usually fail to achieve satisfactory results on them. The development of novel methodologies for such datasets has become a promising research avenue.

1.2 Data Analysis Methodologies

Datasets collected from various sources may have many different forms and reveal distinct information. Therefore, suitable methodologies need to be developed to tackle particular data analysis tasks. Most datasets collected have a matrix form which includes a number of data samples (rows) and several distinct attributes/features (columns). For example, the data collected from the business markets include massive financial information of companies (samples). Each company is characterised by a number of separated features such as share price, total assets, free cash flow, profit margins etc.

Recently, network datasets have gradually been collected and reconstructed. Network topological data contains entities included in the system studied and the connections among them, it is considered as one of the most representative frameworks for complex systems. Analysing such data provides a precious opportunity to interpret their statistical, topological and organisational properties.

Another form of data especially generated due to the developments of molecular biology and availability of high throughput technology is sequence data such as DNAs, RNAs and proteins. A sequence data consists of a finite set of alphabet letters linked sequentially so as to complete some functions alone or with other sequences. Uncovering the hidden information using sequence data analysis methodologies is critical for biologists to understand the fundamental biology of organisms and reveal evolutionary relations of different species.

Data analysis and knowledge discovery are rapidly evolving areas lying are at the intersection of several disciplines including statistics, engineering, operation research and computer science. These involve the extraction and discovery of implicit and potentially useful patterns/knowledge directly from available data. Among a number of specific tasks of data mining, *Clustering* and *Classification* are two typical topics. *Clustering* involves partitioning of a dataset without any qualitative information (class membership) into subsets (clusters) so that samples in each cluster are close enough and share some common traits accordingly. Clustering is an unsupervised procedure where the final partition of data relies only on the distance metric of each pair of data sample. Typical algorithms include hierarchical clustering, k-means clustering, fuzzy c-means clustering, quality threshold clustering and graph theoretical approaches (Duda *et al.*, 2001). *Classification* is a supervised learning procedure from data with known category labels and the prediction of new samples into known patterns. Data classification usually comprises two procedures: training and testing. In the training stage, classification functions are generated to separate samples of known class membership into different groups using the attribute values associated with each training sample. A new sample is then classified into one of those classes by comparing its discriminant scores derived from classification functions in the testing phase. Linear/Quadratic statistical discriminant analysis was first proposed undertaking particular assumptions on group distributions. Recently, novel classification and learning methodologies such as neural networks (Jain and Nag,

1995), support vector machines (Cortes and Vapnik, 1995), decision trees (Quinlan, 1986; Breiman *et al.*, 1984) and mathematical programming approaches (Freed and Glover, 1981a, 1981b, 1986; Gehrlein, 1986; Ryoo, 2006; Sueyoshi, 2006) have been developed to solve various practical classification challenges including financial risk evaluation, protein secondary structure prediction and process fault detection/diagnosis.

Recently, network data analysis has attracted more and more attention from various research communities since many complex systems such as Internet, social relations and biological systems can be represented as network models. Statistical analysis has reflected a number of structural and topological properties of different forms of networks including small world effects, degree distribution and high network transitivity (Barabasi and Albert, 1999; Newman, 2003; Boccaletti *et al.*, 2006). Other methods have been developed to investigate detailed network properties such as community structures (Newman and Girvan, 2004; Duch and Arenas, 2005; Newman, 2006), network design and robustness issues (Dartnell *et al.*, 2005; Estrada, 2006; Deutscher *et al.*, 2006; Paul *et al.*, 2006).

Sequence data analysis plays an important role in various research fields particularly in bioinformatics. Given a collection of sequences, quite a few computational software tools including BLAST (Altschul, 1990), BLAST2 (Tatusova and Madden, 1999), CLUSTAL W (Thompson *et al.*, 1994) together with statistical and combinatorial sequence analysis frameworks have been developed to arrange the primary sequences of DNAs, RNAs, or proteins and identify regions of similarity. Highly reserved patterns obtained that may lead to the discovery of functional, structural, or evolutionary relationships among those sequences.

According to the description above, there exists a clearly identified need to develop optimisation-based data analysis frameworks to facilitate pattern recognition and knowledge discovery in various fields and potentially lead to significant understanding of practical complex systems investigated.

1.3 Optimisation-based Methodologies for Data Analysis

Mathematical programming refers to the study of problems in which one seeks to minimise or maximise a real objective function by systematically designing

continuous and integer optimisation variables and building mathematical frameworks. It provides a precious opportunity to investigate many real world systems by formulating mathematical models and solve them from the optimisation point of view. The maximum/minimum solutions of optimisation problems usually provide a guidance to optimally allocate natural/human recourses, design business strategies, identify stable molecular configurations and so on.

However, expensive computational resources have hindered wider and deeper applications of mathematical programming to various research fields and industries until recently when computer hardware become fairly cheap and the availability of modelling software and efficient solution algorithms. There have been seen truly potentials to apply mathematical programming techniques to enormous fields around the world such as engineering, planning and scheduling, business and finance, chemistry and biology etc.

According to the systematic elaboration by Williams (1999), building mathematical programming models usually is motivated by the following three reasons:

- To gain an insight into the problem. The actual exercise of building a mathematical model often reveals relationships that were not apparent previously. As a result, greater understanding of the problem is achieved.

- To identify non-obvious solutions. Having built a model it is then possible to analyse it mathematically and help suggest course of actions that might not otherwise be obvious.

- To investigate extreme aspects of the problem. Computational experiments can be undertaken when it is not possible or desirable to conduct an experiment in real-life and provide us with useful information concerning the problem under investigation.

A number of mathematical programming models have been developed in the area of data analysis and data mining. Most mathematical frameworks proposed focus on typical data mining problems such as unsupervised clustering and supervised classification problems. Hansen and Jaumard (1997) reviewed recent advances of cluster analysis and highlighted a number of key mathematical programming-based methodologies for various clustering problems. Linear programming (LP) models

have first been proposed to tackle data classification problems (Freed and Glover, 1986; Mangasarian, 1997). Mixed integer programming (MIP) models later have been addressed where binary variables were introduced to indicate whether the sample is correctly or incorrectly classified (Gehrlein, 1986; Stam and Joachimsthaler, 1990; Wilson, 1996). More complex classification models such as piecewise linear and hyper-box classifiers have recently been developed as MILP formulations (Glen, 2005; Ryoo, 2006; Uney and Turkay, 2006). Comparing with other computational methodologies, the key advantage of developing data analysis models using mathematical programming techniques is obvious: it is very straightforward to implement classification and clustering models through standard modelling tools and very few parameters are required. Linear or nonlinear optimisation models can be developed via a series of algebraic equations and the relationships between data and models can be expressed through logical constraints.

Various network models have been developed using mathematical programming. Typical network path problems such as Minimum Spanning Trees, Shortest Path and Travelling Salesman and Arc Routing problems can be formulated as mathematical programming models (Eiselt and Sandblom, 2000). Practical network flows and design problems such as supply chain networks (Tsiakis *et al.*, 2001; Gjerdrum *et al.*, 2001), heat exchanger networks (Zamora and Grossmann, 1998; Shivakumar and Narasimhan, 2002) and biological networks (Burgard and Maranas, 2001; Lin *et al.*, 2003) can also be investigated using mathematical programming. Particularly, Palsson and his research group proposed Flux Balance Analysis (FBA) (Varma and Palsson, 1993), which provided a linear programming (LP) framework for modelling metabolic networks and studying the metabolic capabilities of an organism. Moreover, mathematical programming techniques have been used to investigate topological characteristics of networks. Integer programming (IP) models were developed to find cliques in networks (Balasundaram *et al.*, 2005). Dartnell *et al.* (2005) proposed an LP-based approach to study the network robustness under random and direct attacks.

There have been some attempts to apply mathematical programming to analyse sequence data. Meneses *et al.* (2004, 2005) discussed typical problems in string selection and comparison. A number of MILP models and heuristics were developed to identify sub strings from a group of sequences using different objective functions. A general integer programming-based approach was proposed to find repeated

patterns/motifs of DNA and protein sequences (Zaslavsky and Singh, 2006). The proposed model was flexible and robust so as to cope with several variants of the motif finding problem. Kingsford *et al.* (2006) formulated a compact mathematical programming model for DNA motif finding problems. A set of constraints were added to tighten the linear relaxation of the proposed MIP model and an efficient separation algorithm was developed to reduce the computational efforts. Lee *et al.* (2006) presented a novel graph-theoretical approach for representing a wide variety of sequence analysis problems and developed two MILP models to solve such problems. Since those models have proven to be computationally intensive, a heuristic algorithm, introduced herein for multiple sequence alignment, overcomes such challenges and is capable of returning good sequence alignments within reasonable computational time. Later, an MILP formulation was proposed to tackle the global piecewise protein sequence alignment problem (McAllister *et al.*, 2007). Not only does the proposed formulation guarantee the identification of the global optimal alignment, but also it provided a rank-ordered list of pairwise alignments to incorporate other biological functions.

Apart from mathematical formulations described above, meta-heuristic or stochastic search methods have also been applied to solve data analysis problems; examples include Genetic Algorithms (GA), Simulated Annealing (SA) and Tabu Search (TS). Meta-heuristics mainly invoke search procedures all over the feasible region and intensify the search in some promising areas. Meta-heuristics cannot easily be trapped in local optimal solutions while are computationally costly due to their slow convergence. Meta-heuristics can be classified into population-based methods and point-to-point methods. In the latter methods, the search invokes only one solution at the end of each iteration from which the search will start in the next iteration. On the other hand, the population-based methods invoke a set of many solutions at the end of each iteration. Below, we highlight the principles of genetic algorithms as an example of population-based methods, and simulated annealing as examples of point-to-point methods.

Genetic algorithms (GA) (Booker *et al.*, 1989) are meta-heuristic methods inspired by a number of evolutionary biology mechanisms such as inheritance, mutation, selection and crossover. GA is typically implemented as a simulation where a population of abstract representations (called chromosomes) of candidate solutions (called

individuals) to an optimisation problem evolves toward better solutions. As an approach to global optimisation, GA has been applied to solve various data classification problems. Sharpe and Glover (1999) proposed efficient GA-based approaches for data classification. Peng (2003) combined GA and support vector machines (SVM) to identify the types of cancer molecules. Finally, fuzzy-rule-based classifiers were developed using GAs, which extracted the optimal parameters of the fuzzy classifier including fuzzy membership functions and the size and structure of fuzzy rules (Zhou and Khotanzad, 2007).

Simulated Annealing (SA) (Kirkpatrick *et al.*, 1983) calculates the probability of accepting a solution which is worse than the reference solution to a temperature-like parameter based on an analogy of metal cooling processes. One of the most successful applications of SA to data analysis involves the community structure identification of network data. Guimera and Amaral (2005) applied SA to identify functional modules in metabolic networks of twelve organisms from three different superkindoms by maximising their modularity values. The same optimisation methodology was also applied to analyse and benchmark social networks (Medus *et al.*, 2005) where a trade-off between quality of solutions and computational requirements was noted.

1.4 Aims and Objectives

Deeply motivated by the promise of better understanding and enhanced problem-solving capabilities offered by mathematical programming, the aim of our work is *to propose optimisation-based approaches for data analysis.* Our objectives are specifically to develop a number of mathematical programming frameworks and efficient solution methodologies to analyse different forms of data.

In order to achieve our goal, the following areas will be addressed:

- *Data Classification using Mixed Integer Optimisation*

  This work involves the development of a mathematical programming framework for the data classification problem using hyper-box classifiers. A number of data samples are used in the training process and each sample is characterised by a number of independent attributes together with a class label. The dimensions and locations of each hyper-box are obtained after training

representing the pattern of data samples. The class membership of a new testing sample is predicted from the training outcome. Later, an efficient two-stage solution algorithm is introduced to tackle classification of more training samples and complex structures.

- *Network Community Structure Identification through Mathematical Programming*

  This area deals with the development of a optimisation-based framework to automatically identify community structures in complex networks through network modularity maximisation. In order to accommodate the combinatorial nature of community structure identification of medium/large size networks, efficient solution approaches are proposed to achieve near-optimal solutions within reasonable computational times.

- *Motif Identification from DNA Sequences using Mixed Integer Programming*

  This topic is to develop a novel mathematical programming formulation to identify consensus sequences and motifs from a collection of DNA sequences. The proposed optimisation framework determines not only the positions and contents of motifs inside each DNA sequence but also the content of the consensus sequence. The total matching scores between the consensus sequence and motifs are maximised.

The problems described in the thesis are formulated as Mixed Integer Linear Programming (MILP), Mixed Integer Quadratic Programming (MIQP) or Mixed Integer Non-Linear Programming (MINLP) optimisation models. The General Algebraic Modelling System (GAMS, Brooke, *et al.* 2003), a specially designed modelling language coupled with many linear, nonlinear and mixed integer programming solvers are used to obtain the solutions of the resulting models.

1.5 Thesis Outline

The rest of the thesis is structured in three parts as follows: Part I addresses the problem of data classification and consists of Chapters 2 and 3. Part II tackles the problem of network community structure identification and comprises of Chapters 4

and 5. Part III involves the DNA motif finding problem which is included in Chapter 6.

Chapter 2 presents a mixed integer optimisation approach for data classification problems. In the training stage, the classification problem is formulated as a mixed integer linear programming (MILP) mathematical model capturing correctly classified training samples using hyper-boxes. An iterative solution approach is then proposed to allow multiple boxes for each single class of data samples. In the testing part, the distances between each new sample with unknown class membership and all established hyper-boxes are calculated and testing samples are allocated to the nearest hyper-box. Finally, the testing performances of the proposed optimisation-based approach are compared with other linear/nonlinear classifiers.

In order to extend the applicability of optimisation-based hyper-box classifiers to large scale datasets and complex data structures, an efficient two-stage decomposition algorithm is proposed in Chapter 3. In this first stage, all training samples are split into a number of disjoint regions without considering their class memberships. Training data samples in each region are then completed using the optimisation-based approach developed in Chapter 2. The computational results indicate that the proposed decomposition scheme is able to accommodate more demanding data classification tasks with very competitive testing performances.

In Chapter 4, a mathematical programming-based approach to identify the optimal community structures of complex networks based on the maximisation of a network modularity metric is proposed. The overall problem is formulated as an MIQP model, which can then be solved to global optimality using a standard branch-and-bound procedure. Special symmetry-breaking constraints are incorporated to eliminate equivalent solutions. Additional features such as minimum/maximum module size and balancing among modules can easily be incorporated in the model. The applicability of the proposed optimisation-based approach is demonstrated by a number of illustrative network examples.

The mathematical model proposed in Chapter 4 is able to obtain global optimal solutions for small size networks. Chapter 5 presents a two-stage optimisation-based solution approach to find community structures for medium/large networks with good quality network modularity values. An iterative solution procedure is then developed

to find finer modular structures with high resolution. The applicability of the proposed approach is finally demonstrated by a number of synthetic and real networks. The computational results are finally compared with other community structure identification approaches in the literature.

An MILP model is developed in Chapter 6 to identify the consensus sequence of a set of DNA sequences together with the DNA motifs of each sequence. The proposed mathematical model determines the content of a consensus sequence and motif locations of a collection of DNA sequences so as to maximise the total similarity scores between the consensus subsequence and all sequence motifs. The predicted motif contents are compared to the real DNA motifs identified through biological experiments and the prediction accuracies of the proposed model are finally compared with two other computational methodologies in the literature.

Finally, Chapter 7 summarises the main contributions of the thesis and provides recommendations for further research work.

PART I

OPTIMISATION-BASED

DATA CLASSIFICATION

Chapter 2

A Mixed Integer Optimisation Model for Data Classification

In recent years, data mining and machine learning have become increasingly important since the collection and storage of data are much easier and less expensive. Data classification involves the automatic discrimination of patterns from data and prediction of group memberships of unknown samples. Computational methodologies for data classification have widely been applied to extract knowledge of many real systems. In this chapter, a mixed integer linear programming (MILP) model is proposed for general data classification problems using a hyper-box representation. This representation is particularly suitable for capturing disjoint data regions. The objective function used is the minimisation of the total number of misclassified data samples. An iterative solution procedure is then developed to assign potential multiple boxes to each single class so as to improve training and testing performances. Finally, the applicability of the proposed approach is demonstrated through a number of illustrative examples.

2.1 Introduction and Literature Survey

Data classification is one of the fundamental problems in data mining and machine learning. It deals with the identification of patterns and the assignment of new samples into known groups. During the training process, classification functions are generated to separate samples of known class membership into different groups using all attribute values associated with each sample. A new sample is then classified into one of those classes by comparing its discriminant scores derived from classification functions. Different classification models have been successfully applied in a wide range of fields including financial aspects (Becerra-Fernandez *et al.*, 2002; Glen, 1999; Sueyoshi, 2004; Zopounidis and Doumpos, 2002), fault diagnosis and quality control (Chiang *et al.*, 2004; Purintrapiban and Kachitvichyanukul, 2003; Shin *et al.*, 2005), flow regime identification (Tarca *et al.*, 2004; Trafalis *et al.*, 2005), and protein secondary structure prediction (Ramnarayan *et al.*, 2008; Turkay *et al.*, 2005). Generally, major classification methodologies include statistical methods, neural networks, support vector machines, decision trees and mathematical programming approaches. Next, a brief description of each methodology is provided.

Statistical Methods: The development of statistical classification models can be tracked back to linear discriminant analysis (LDA) where the ratio of between groups and within group variances is maximised. Previous statistical discriminant methods were reviewed by McLachlan (1992). The conventional statistical discriminant analysis methods usually undertake assumptions on group distributions. Sometimes linear and quadratic discriminant functions can obtain very promising results; it is also true that many datasets do not satisfy such distribution assumptions. Apart from statistical discriminant analysis, other statistical methods have also been proposed such as logistic regression and k-nearest neighbour (k-NN) approaches (Hand, 1997). Logistic regression is a statistical method where the logistic transformation of a linear function of the training samples is considered as the logarithm of the odds of class membership. k-NN methods estimate the class membership based on k closest training samples. The number of nearest neighbours, k, must be specified before the training.

Neural Networks: Neural network (NN) approaches have been applied extensively by various researchers to solve different problems of data classification and pattern recognition (Jain and Nag, 1995; Markham and Ragsdale, 1995; We, 2002; Autio *et*

al., 2007). It should be mentioned that a min-max neural networks classifier using fuzzy sets as pattern classes was described (Simpson, 1992). *N*-dimensional fuzzy set hyper-boxes were defined by minimum and maximum points with a corresponding membership function. The boundary points are determined through fuzzy min-max learning algorithms. An efficient fuzzy partition of the feature space to generate fuzzy if-then rules for pattern classification was addressed (Mandal, 1997). Overlapping hyper-boxes were used to decompose the whole feature space. Simpson's work was later generalised and extended to combine the supervised and unsupervised learning with a single learning algorithm (Gabrys and Bargiela, 2000). The boundaries of each distinct class were represented by hyper-boxes and their sizes were adjusted through the learning process. As demonstrated by Funahashi (1989) and Hornik *et al.* (1989), NN can approximate unknown nonlinear functions to any degree of accuracy without any assumptions of the data distribution. It is so flexible that no prior specifications of discriminant functions are needed. A problem associated with NN is that the global optimality of NN solutions is not guaranteed. Also, NN is considered as a black box and people can not find any relevance of input variables after the training. All we can obtain are the optimised weights between the nodes hidden in the network structure.

Support Vector Machines: Support vector machines (SVM), which are based on the statistical learning theory developed by Vapnik and his group (1995), have been considered as one of the most promising approaches for two-class classification problems (Yajima, 2005). Given a series of training samples which belong to two distinct categories, SVM tend to find an optimal separating hyper-plane so as to maximise the margin between them. In order to improve the training performance, nonlinear mapping is adopted from the input space to a higher dimensional feature space by using several kernel functions. The overall problem is formulated as a quadratic programming (QP) problem. SVM have been applied extensively to the area of image recognition (Je *et al.*, 2003), text categorisation (Kaufman, 1999; Platt, 1999), protein folding recognition (Ding and Dubchak, 2001) and so on. However, the main difficulty when using SVM is the selection of optimal kernel functions and their parameters before training. Also, SVM is originally designed for two-class classification problems. The optimal design for multi-class SVM classifiers is a further area for research (Navia-Vazquez, 2007; Zhong and Fukushima, 2007).

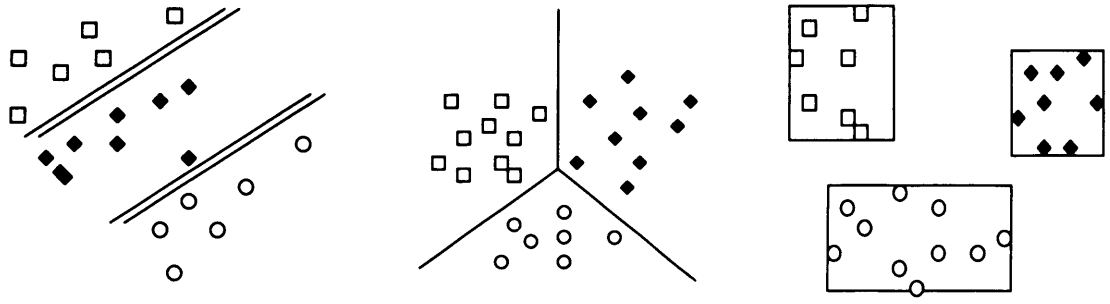
Decision Trees: Decision trees (DT) create a discriminant tree that recursively splits training samples into disjointed subsets until either no further splitting can produce significant differences or the subsets are small enough. There are a number of available models including ID3 (Quinlan, 1986) and CART (Breiman *et al.*, 1984). ID3 methods propose a top-down irrevocable strategy to construct a decision tree and the attribute with the minimum entropy is selected to split the tree. In CART techniques, the expected cost of misclassifications is minimised. DT techniques have been considered as important pattern classification tools and were widely used in knowledge discovery from manufacturing systems (Koone *et al.*, 1997, Markham *et al.*, 1998); control chart patterns detection (Guh, 2005); protein cellular localisation prediction (Lorena and de Carvalho, 2007) and so on. However, one of the disadvantages of DT lies in its instability due to the hierarchical nature of the classification process. The effect of an error on the top level of a tree can be propagated down to other branches. The other issue of using DT methods is the difficulty of designing optimal tree structures since the classification performance heavily depends on how well the tree is constructed.

Mathematical Programming: Mathematical programming (MP) techniques have also been proposed to various data classification problems. Comparing with other approaches, MP methods are straightforward to implement through standard modelling tools and very few parameters are required during training processes. Linear or nonlinear classifiers (see Figure 2.1) can be designed via a series of algebraic equations and the relationships between training samples and classifiers can be expressed through logical constraints. More importantly, the patterns hidden from data can be extracted mathematically from the coefficients of the classifiers. The development of MP discriminant analysis methods was largely stimulated by the work of Freed and Glover (1981a, 1981b, 1986). In the simplest form of MP discriminant analysis, a discriminant function, which partitions a training sample with known membership into the specified group, is generated using linear programming (LP) models (Bajgier and Hill, 1982; Freed and Glover, 1986; Glover, 1990; Lam and Moy, 1996, 1997). In LP models, maximisation of the minimum deviation (MMD) and minimisation of the sum of deviations (MSD) are the two most commonly used optimisation criteria. Different LP formulations and goal programming models were recently compared by Bal *et al.* (2006). Extending LP models, the number of

misclassifications can, however, be minimised directly in mixed integer programming (MIP) models where a binary variable for each training sample was introduced to indicate whether the sample is correctly classified (Gehrlein, 1986; Stam and Joachimsthaler, 1990; Wilson, 1996). Recently, a series of non-parametric discriminant analysis approaches called DEA-DA (Data Envelopment Analysis-Discriminant Analysis) for two-class and multi-class data classification problems were proposed (Sueyoshi, 2006). DEA-DA models provided a set of parallel linear discriminant functions to determine group memberships. Glen (2001) presented an MILP model to maximise the classification accuracy for two-group classification problems. Later, piecewise linear classifiers were applied to approximate nonlinear discriminant functions to improve the classification performance (Glen, 2005; Ryoo, 2006). Novel MILP models were developed to solve classification problems with multiple classes (Lee and Wu, 2007). The resulting optimisation-based framework was able to incorporate heterogeneous types of inputs and high dimensional data transformation. Finally, Uney and Turkay (2006) proposed an MILP model using hyper-box classifiers. The relationships among discrete variables in the model were converted to the equivalent integer constraints using boolean algebra. The proposed optimisation-based approach in this chapter shares similar concepts with the work of Uney and Turkay (2006). Both approaches adopt a hyper-box representation to enclose the correctly classified training samples. Comparing with the work of Uney and Turkay (2006), we develop a completely different MILP formulation which involves significantly fewer binary/continuous optimisation variables and constraints. Comparing with LP models and statistical approaches, MIP-based methods may obtain better performances, while the existence of binary variables make MIP models solve datasets involving a relatively small number of training samples.

In addition to the standard MP-based models, several two-stage models have been developed. Stam and Ragsdale (1992) proposed a two-stage approach where training samples that were difficult to classify were initially identified and more details were analysed in the second stage. The proposed two-stage approach was tested on simulated two-group problems using both MSD and minimisation of misclassifications as the second stage objective, with the results suggesting the approach to be particularly suitable for discriminant problems with outlier contaminated data. Later, Sueyoshi (2001) developed another two-stage MIP

approach. In his approach, the overlapping region was identified in the first stage and all samples which belong to the overlapping region were reclassified in the second stage. However, one limitation of Sueyoshi's method is that it considers two-class problems only. Finally, Glen (2006) compared the performance of several two-stage approaches using simulated and real datasets.



Parallel hyper-planes Linear classifiers Hyper-boxes

Figure 2.1 Discriminant classifiers

In this chapter, an MILP model for data classification problems is presented. Hyper-boxes are adopted to capture the discrete regions of the training samples. Special constraints are introduced to avoid overlapping of boxes that belong to different classes. An iterative solution algorithm is then proposed to improve the training and prediction accuracy by allowing multiple boxes for each class. The rest of the chapter is structured as follows: in the next section, typical mathematical programming-based classification models are summarised. Section 2.3 states the problem of data classification and section 2.4 presents a novel mathematical formulation for the general data classification problem. An iterative solution algorithm is proposed in section 2.5 and a testing procedure is described in section 2.6. In section 2.7, one motivated example is presented in detail to illustrate the applicability of the proposed optimisation-based approach. Five real examples and two synthetic datasets are selected in section 2.8 to perform an extensive computational comparison between our approach and other competing literature methods. Finally, some concluding remarks are made in section 2.9.

2.2 Summary of Typical Mathematical Programming-based Classifiers

In this section, we summarise two typical mathematical programming models for data classification problems. Both approaches proposed linear discriminant classifiers using MILP representations.

2.2.1 An MILP Model for Linear Discriminant Classifiers (Gehrlein, 1986)

Consider a general data classification problem with C classes and S samples. Each sample s is characterised by the values of M independent attributes. A linear discriminant classifier model was proposed by Gehrlein (1986) through an MILP representation. First, the linear discriminant function of class c is mathematically formulated as:

$$Y_c = \alpha_c^0 + \sum_m \alpha_{cm} \cdot A_{sm} \tag{2.1}$$

where A_{sm} is the value of training sample s on attribute m. α_c^0 and α_{cm} are discriminant function coefficients and are treated as continuous variables. The following MILP model is proposed to discriminate training samples from different classes using the above classification function (equation 2.1).

$$\text{Min} \sum_s W_s \tag{2.2}$$

subject to

$$\alpha_c^0 + \sum_m \alpha_{cm} \cdot A_{sm} - (\alpha_{c'}^0 + \sum_m \alpha_{c'm} \cdot A_{sm}) + U \cdot W_s \geq \varepsilon \quad \forall s, m, c_s, c' \neq c_s \tag{2.3}$$

$$\alpha_c^0, \alpha_{cm} : \textit{unrestricted} \; ; \; W_s \in \{0,1\}$$

where c_s is the class sample s belongs and U is a suitable upper bound. Binary variable W_s is used here to indicate whether training sample s is correctly classified. Constraint 2.3 shows the proposed linear discriminant function separate sample s belonging to class c from other classes c' at least by a value of ε if this sample is

correctly classified ($W_s = 0$). The objective function adopted in this model is minimisation of total misclassified training samples.

After training, the testing score of testing sample s on each discriminant function belonging to class c can be calculated by $\alpha_c^{0*} + \sum_m \alpha_{cm}^* \cdot A_{sm}$ where α_c^{0*} and α_{cm}^* are the optimal classification function parameters obtained during the training. Testing sample s will be classified to one of classes c^* for which $c^* = \arg\max_c (\alpha_c^{0*} + \sum_m \alpha_{cm}^* \cdot A_{sm})$

2.2.2 An MILP Model for Parallel Hyper-planes Classifiers (Sueyoshi, 2006)

Recently, Sueyoshi (2006) developed parallel classification hyper-planes using mixed integer optimisation. The general MILP formulation for classification with multiple groups is summarised below:

$$\text{Min} \sum_s W_s \tag{2.4}$$

subject to

$$\sum_m (\lambda_m^+ - \lambda_m^-) \cdot A_{sm} - w_c + U \cdot W_s \geq 0 \quad \forall s \in c_s, c = 1,...,C-1 \tag{2.5}$$

$$\sum_m (\lambda_m^+ - \lambda_m^-) \cdot A_{sm} - w_{c-1} - U \cdot W_s \leq -\varepsilon \quad \forall s \in c_s, c = 2,...,C \tag{2.6}$$

$$\sum_m (\lambda_m^+ + \lambda_m^-) = 1 \tag{2.7}$$

$$\varepsilon \cdot \zeta_m^+ \leq \lambda_m^+ \leq \zeta_m^+ \qquad \forall m \tag{2.8}$$

$$\varepsilon \cdot \zeta_m^- \leq \lambda_m^- \leq \zeta_m^- \qquad \forall m \tag{2.9}$$

$$\zeta_m^+ + \zeta_m^- \leq 1 \qquad \forall m \tag{2.10}$$

$$\sum_m (\zeta_m^+ + \zeta_m^-) = M \tag{2.11}$$

w_c: unrestricted; $\zeta_m^+, \zeta_m^-, W_s \in \{0,1\}$; $\lambda_m^+, \lambda_m^- \geq 0$

In this model, the linear discriminant function for class c is formulated as:

$$Y_c = \sum_m (\lambda_m^+ - \lambda_m^-) \cdot A_{sm} - w_c \tag{2.12}$$

It should be noted that the classification hyper-plane of each class has the same weight ($\lambda_m^+ - \lambda_m^-$) with different coefficient w_c. Here, the normalisation technique proposed by Glen (1999) is incorporated in Equations (2.7) to (2.11). Constraint (2.7) enforces that all λ_m^+ and λ_m^- have the values between 0 and 1. For each attribute, two binary variables, ζ_m^+ and ζ_m^+, are introduced in equations (2.8-2.10) so that only one of λ_m^+ and λ_m^- have non-zero values. Finally, equation (2.11) illustrates that all weights are used to construct the linear classifier.

Overall, training samples with C classes are separated by the optimal values of w_c^* and $(\lambda_m^{+*} - \lambda_m^{-*})$. After training, a new testing sample, s, is classified as:

If $\sum_m (\lambda_m^{+*} - \lambda_m^{-*}) \cdot A_{sm} \geq w_{c_1}^*$ then sample s belongs to class c_1.

If $w_c^* \leq \sum_m (\lambda_m^{+*} - \lambda_m^{-*}) \cdot A_{sm} \leq w_{c-1}^* - \varepsilon$ then sample s belongs to class c ($c = c_2, ..., C-1$).

If $\sum_m (\lambda_m^{+*} - \lambda_m^{-*}) \cdot A_{sm} \leq w_{C-1}^* - \varepsilon$ then sample s is classified into class C.

2.3 Problem Statement

According to the review in section 2.1, the main task of data classification is to design a type of classifier to differentiate data with unique class labels and the total number of misclassifications should be minimised. The classifier structure needs to be fixed and the parameters of discriminant function will be identified during the training process. After training, the class membership of several new samples will be predicted based on the classifiers derived during the training process.

The prototype of our model is based on the MILP mathematical formulation for the process plant layout problem proposed by Papageorgiou and Rotstein (1998) where process facilities were simplified to rectangular boxes within two dimensional space and the optimal facilities positioning was determined so as to minimise the total connection cost of the process flowsheet. Our approach is based on the previous MILP representation by extending it to cover M dimensions (where M is the number of attributes). The specific patterns of training data are captured by hyper-boxes with M dimensions. Each box is characterised by its centroid position and dimensions. Linear constraints are used to avoid overlapping among hyper-boxes belonging to different classes. The final objective function is the total number of misclassified samples, which is minimised.

The overall problem investigated can be stated as follows:

Given:

Training data of S samples with M attributes

Classification of training data into one of C classes

Determine:

The optimal number of hyper-boxes

The optimal centroid position and dimensions of hyper-boxes.

so as to

Minimise the total number of misclassified samples

2.4 Mathematical Formulation

In this section, a rigorous mathematical model for data classification problems is presented. It is first assumed that *only* one hyper-box is adopted to enclose the training samples for each class. The above assumption is relaxed in section 2.5 where an iterative solution procedure based on multiple boxes is introduced in order to improve the training and testing accuracy.

The indices and parameters associated with the data classification problem are listed below:

Indices

s	Sample ($s=s_1, s_2, \ldots, S$)
m	Attribute ($m=m_1, m_2, \ldots, M$)
i, j	Hyper-box ($i,j=1,2, \ldots, N$)
i_s	Hyper-box which sample s belongs to

Parameters

A_{sm}	Value of sample s on attribute m
ε	Minimum distance between boxes that belong to different classes

The formulation is based on the following key variables:

Binary variables

E_s	1, if sample s is included in the corresponding hyper-box; 0 otherwise
Y_{ijm}	0, if box i and j do not overlap each other on attribute m; 1 otherwise

Continuous variables

LE_{im}	Length of hyper-box i on attribute m
X_{im}	Central coordinate of hyper-box i on attribute m

2.4.1 Objective Function

The objective function used here is the minimisation of the total number of misclassified samples.

$$\min \sum_{s} (1 - E_s) \tag{2.13}$$

2.4.2 Hyper-box Enclosing Constraints

In this model, hyper-boxes are used to enclose correctly classified samples. Binary variables, E_s, are introduced to describe if sample s is classified correctly within the corresponding box. Therefore, a sample s is correctly classified if the following two conditions are active at the same time:

$$A_{sm} \geq X_{im} - \frac{LE_{im}}{2} \qquad \forall s, i_s, m \tag{2.14}$$

$$A_{sm} \leq X_{im} + \frac{LE_{im}}{2} \qquad \forall s, i_s, m \tag{2.15}$$

These enclosing conditions can mathematically be modelled in a mixed-integer linear form:

$$A_{sm} \geq X_{im} - \frac{LE_{im}}{2} - U(1 - E_s) \qquad \forall s, i_s, m \tag{2.16}$$

$$A_{sm} \leq X_{im} + \frac{LE_{im}}{2} + U(1 - E_s) \qquad \forall s, i_s, m \tag{2.17}$$

where U is a suitable upper bound. Note that both constraints (2.14-2.15) are active only if sample s is correctly classified (*i.e.* $E_s = 1$).

2.4.3 Non-overlapping Constraints

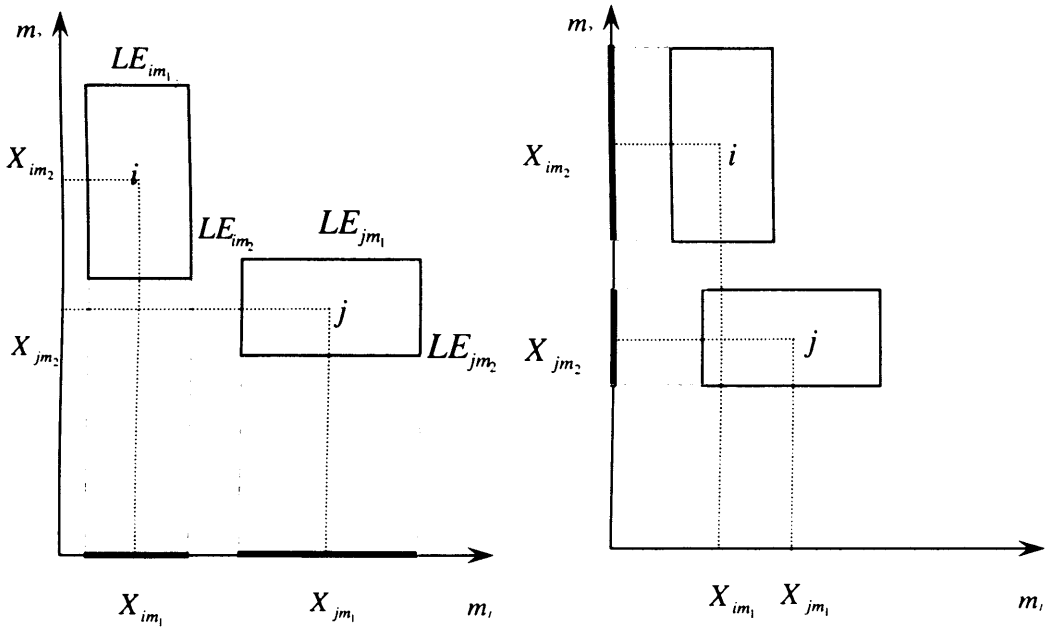
Since hyper-boxes represent the unique pattern of corresponding classes, any two boxes that belong to different classes can not share the same position. If a region in the attribute space is covered by two boxes that represent different classes, it is possible that some new data samples are allocated to more than one class. The prototype of non-overlapping constraints comes from the MILP formulation of process plant layout problems proposed by Papageorgiou and Rotstein (1998) where N process facilities were considered as rectangular boxes (i or j) within two dimensional space (m_1 and m_2). The conditions of any two facilities occupying the same physical location are prohibited by activating *at least* one of the following four inequality constraints:

$$X_{im_1} - X_{jm_1} \geq \frac{LE_{im_1} + LE_{jm_1}}{2} \qquad \forall i = 1, ..., N-1, j = i+1, ..., N \qquad (2.18)$$

$$X_{jm_1} - X_{im_1} \geq \frac{LE_{im_1} + LE_{jm_1}}{2} \qquad \forall i = 1, ..., N-1, j = i+1, ..., N \qquad (2.19)$$

$$X_{im_2} - X_{jm_2} \geq \frac{LE_{im_2} + LE_{jm_2}}{2} \qquad \forall i = 1, ..., N-1, j = i+1, ..., N \qquad (2.20)$$

$$X_{jm_2} - X_{im_2} \geq \frac{LE_{im_2} + LE_{jm_2}}{2} \qquad \forall i = 1, ..., N-1, j = i+1, ..., N \qquad (2.21)$$



(a) (b)

Figure 2.2 Non-overlapping of hyper-boxes

The non-overlapping conditions are clearly depicted in Figure 2.2. For instance, in case (a) of Figure 2.2, inequality (2.19) is active while in case (b) of Figure 2.2, inequality (2.20) is active. These non-overlapping conditions can mathematically be modelled by introducing a binary variable, Y_{ijm}, together with the following constraints:

$$X_{im_1} - X_{jm_1} + U \cdot Y_{ijm_1} \geq \frac{LE_{im_1} + LE_{jm_1}}{2} \qquad \forall i, j \neq i \qquad (2.22)$$

$$X_{im_2} - X_{jm_2} + U \cdot Y_{ijm_2} \geq \frac{LE_{im_2} + LE_{jm_2}}{2} \qquad \forall i, j \neq i \qquad (2.23)$$

where U is a suitable upper bound. On dimension m, the non-overlapping condition of facility i and j is active when Y_{ijm} has a value of zero. Finally, overlapping between any pair of facility boxes is avoided by forcing *at least* one of conditions (2.18) to (2.21) to be active by:

$$Y_{ijm_1} + Y_{jim_1} + Y_{ijm_2} + Y_{jim_2} \leq 3 \qquad \forall i = 1, \ldots, N-1, j = i+1, \ldots, N \qquad (2.24)$$

Next, we extend the non-overlapping conditions from two dimensions to M dimensions, which will be used for data classification problems. Similar to the non-overlapping inequalities described above (conditions 2.18 to 2.21), the overlapping of boxes i and j on attribute m is avoided by satisfying the following condition:

$$X_{im} - X_{jm} \geq \frac{LE_{im} + LE_{jm}}{2} + \varepsilon \qquad \forall m, i, j \neq i \qquad (2.25)$$

where ε is defined as a small positive number in condition (2.25) to prevent two boxes from sharing the same border on attribute m. Thus, the non-overlapping conditions of boxes i and j can be enforced by:

$$X_{im} - X_{jm} + U \cdot Y_{ijm} \geq \frac{LE_{im} + LE_{jm}}{2} + \varepsilon \qquad \forall m, i, j \neq i \qquad (2.26)$$

It should be added that boxes i and j are considered as non-overlapped when condition (2.25) is satisfied for *at least* one dimension:

$$\sum_{m=1}^{M} (Y_{ijm} + Y_{jim}) \leq 2M - 1 \qquad \forall i = 1, \ldots, N-1, j = i+1, \ldots, N \qquad (2.27)$$

Overall, the MILP optimisation model for the **M**ulti-**C**lass data classification **P**roblem (MCP) is summarised as follows:

[Problem MCP]

$$\min \sum_{s} (1 - E_s)$$

subject to

Hyper-box enclosing constraints: (2.16)-(2.17)

Non_overlapping constraints: (2.26)-(2.27)

$E_s, Y_{ijm} \in \{0,1\}; LE_{im} \geq 0; X_{im}$: unrestricted

2.5 An Iterative Solution Algorithm

In section 2.4, an MILP model for the data classification problem has been proposed. As already mentioned, only one hyper-box for each class is first introduced to enclose the maximum number of correctly classified samples. In this section, an iterative solution procedure is proposed to allow multiple boxes for each class so as to improve the training and testing performance. After solving the single level MILP (MCP), new boxes are assigned to any misclassified samples (*i.e.* $E_s = 0$) during previous iterations and the modified MILP model with more hyper-boxes is then solved. The algorithm will terminate when the objective functions of two successive iterations have the same value. It should be noted that when a new box is added, the non-overlapping constraints (2.26) and (2.27) and the corresponding Y_{ijm} binary variables are generated *only* for those boxes which belong to different classes. Therefore, we do allow potential overlapping of boxes of the same class but not between different classes. The following sets and scalars are defined for the description of the iterative algorithm:

Sets

H = Set of boxes that belong to the same class

Δ = Set of misclassified samples

Scalars

N = Number of boxes used

C = Number of class

The proposed iterative solution algorithm allowing **Multi**ple **H**yper-**B**oxes (Algorithm Multi-HB) for each single class during training can be outlined as follows:

[Algorithm Multi-HB]

STEP 1: Initialise $\Delta = \phi$, $H = \phi$, $N=C$.

STEP 2: Solve single level MILP (MCP, see section 2.4).

STEP 3: Identify samples outside hyper-boxes ($E_s = 0$). Update Δ.

STEP 4: Add one more box for each class to samples in Δ. Update N, i_s, H.

STEP 5: Formulate new MCP problem with more added boxes. Non-overlapping constraints and variables are generated for $i, j \notin H$ only.

STEP 6. Solve the modified single level MILP (MCP) using updated N boxes.

STEP 7: If the objective function values of two successive iterations are the same, STOP; otherwise, go to STEP 3.

2.6 Testing Procedure

After the training process, the main patterns from the data are established. The other important task for any classification methods is the ability to perform a successful prediction based on the training outcome. According to our hyper-box approach, some

new samples will be assigned to one of the existing hyper-boxes so as to identify their class memberships. The basic idea of the testing scheme is that the unclassified sample should be assigned to the nearest box. The variables used in the testing phase are listed below:

LB_{im} Lower bound of box i on attribute m

UB_{im} Upper bound of box i on attribute m

$DIST_{sim}$ Distance between sample s and hyper-box i on attribute m

DSI_{si} Distance between sample s and hyper-box i

First, the lower and upper bound of each hyper-box is calculated:

$$LB_{im} = X_{im} - \frac{LE_{im}}{2} \qquad \forall i, m \qquad (2.28)$$

$$UB_{im} = X_{im} + \frac{LE_{im}}{2} \qquad \forall i, m \qquad (2.29)$$

The distance between sample s and box i on attribute m, $DIST_{sim}$, is defined to be:

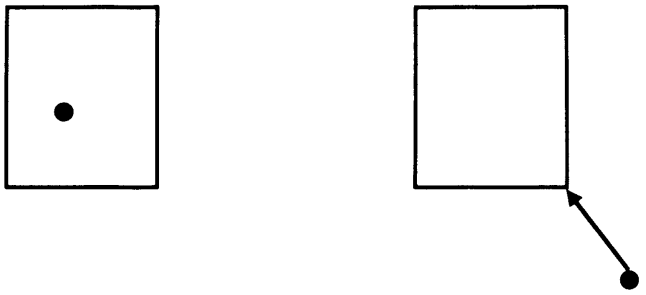
$$DIST_{sim} = \max(0, A_{sm} - UB_{im}, LB_{im} - A_{sm}) \qquad \forall s, i, m \qquad (2.30)$$

So, the distance between testing sample s and hyper-box i, DSI_{si}, is defined to be:

$$DSI_{si} = \sqrt{\sum_{m=1}^{M} DIST_{sim}^2} \qquad \forall s, i \qquad (2.31)$$

Figure 2.3 shows the actual calculation of DSI_{si} in the two-dimensional space. According to equations (2.30) and (2.31), if a sample lies within the boundaries of a box on all attributes (see case *a* in Figure 2.3), the distance between the sample and the box is zero. The membership of the testing sample is identified directly as the class that is represented by the hyper-box enclosing the sample. If a testing is outside all existing boxes (see case b in Figure 2.3), all DSI_{si} values are positive. Sample s

will then be allocated into one of the existing hyper-boxes, i^* , for which $i^* = \arg\min_i \{DSI_{si}\}$.



case a case b

Figure 2.3 Calculation the distance between sample s and box i

2.7 Motivating Example

In this section, we introduce a motivated example to show how the proposed approach captures patterns of training samples. This example involves 32 data points belonging to four classes. Each sample is characterised by two independent attributes and samples of classes 1 and 4 have disjoint regions. We choose 28 of the total 32 samples to construct training hyper-boxes (see filled points in Figure 2.4) and the rest is used for testing (see void points in Figure 2.5). After the training stage, It can clearly be seen that the proposed optimisation-based approach perfectly capture all training samples using six hyper-boxes (see Figure 2.5) through three iterations. After training, the proposed testing method is used to predict the class membership of all 4 testing samples. Two testing samples are directly assigned to classes 2 and 4 as they are already included the corresponding hyper-boxes obtained. The other two testing samples are outside all resulting hyper-boxes and are classified into the nearest box.

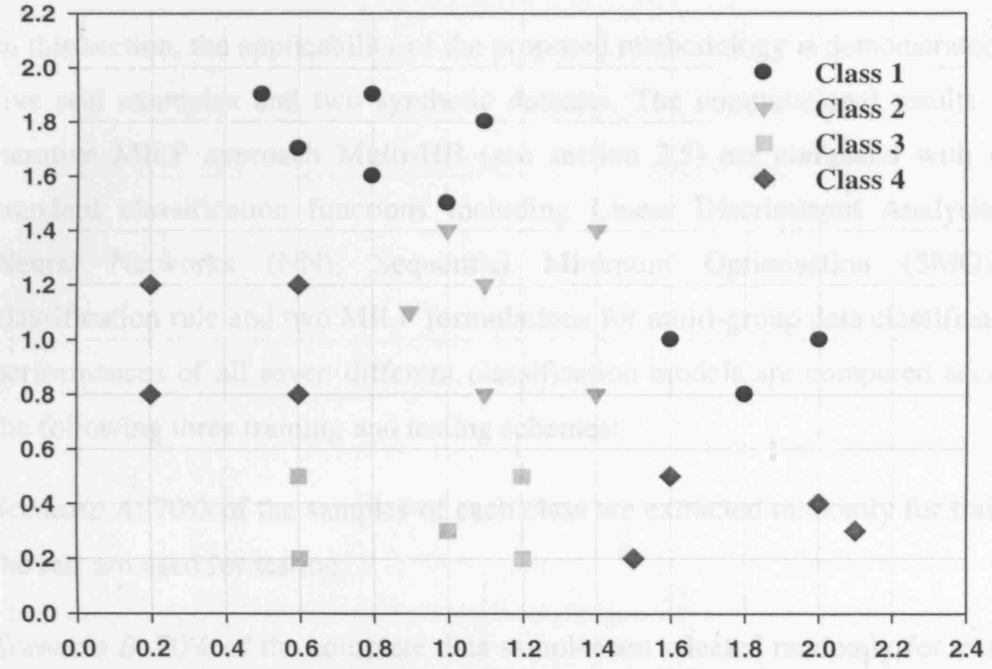


Figure 2.4 Graphical representation of training samples for the motivated example

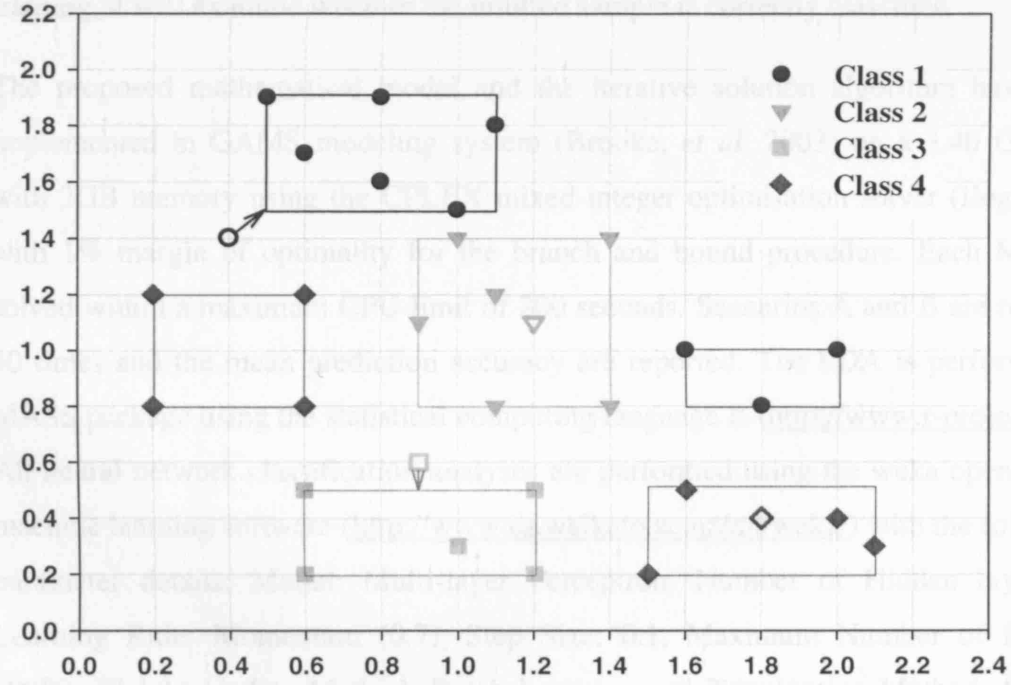


Figure 2.5 Hyper-boxes enclosing training samples for the motivated example

2.8 Computational Results

In this section, the applicability of the proposed methodology is demonstrated through five real examples and two synthetic datasets. The computational results from the iterative MILP approach Multi-HB (see section 2.5) are compared with six other standard classification functions including Linear Discriminant Analysis (LDA), Neural Networks (NN), Sequential Minimum Optimisation (SMO), OneR classification rule and two MILP formulations for multi-group data classification. The performances of all seven different classification models are compared according to the following three training and testing schemes:

Scenario A: 70% of the samples of each class are extracted randomly for training and the rest are used for testing.

Scenario B: 70% of the complete data samples are selected randomly for training and testing is applied to the remaining samples.

Scenario C: leave-one-out scheme. This scheme drops one sample from the whole set of samples for testing and the remaining set of samples are used for training. After training, it will examine whether the omitted sample is correctly classified.

The proposed mathematical model and the iterative solution algorithm have been implemented in GAMS modeling system (Brooke, *et al.* 2003) on a 3.40 GHz PC with 2GB memory using the CPLEX mixed-integer optimisation solver (Ilog, 2006) with 1% margin of optimality for the branch and bound procedure. Each MILP is solved within a maximum CPU limit of 200 seconds. Scenarios A and B are repeated 50 times and the mean prediction accuracy are reported. The LDA is performed by MASS package using the statistical computing language R (http://www.r-project.com). All neural network classification analyses are performed using the weka open source machine learning software (http://www.cs.waikato.ac.nz/ml/weka/) with the following parameter details: Model: Multi-layer Perceptron, Number of Hidden layers: 2, Learning Rule: Momentum (0.7), Step Size: 0.1, Maximum Number of Epochs: 10000, Weight Update Method: Batch Learning and Termination Method: Cap the number of epochs. Two other methods, SMO and OneR, are also performed using their default settings in the weka package. SMO trains support vector machines using sequential minimal optimisation algorithms (Platt, 1999). Multi-class problems are

solved through a pairwise classification scheme. The OneR approach builds a single level decision tree, learns classification rules from each attribute of the training samples and selects the rule with the smallest error rate (Witten and Frank, 2005). The two MILP model presented here for comparison are linear discriminant classifiers (Gehrlein, 1986) and parallel hyper-planes (Sueyoshi, 2006).

2.8.1 Real Datasets

The applicability of the proposed approach is demonstrated by five literature examples. The first small example was initially used by Nath and Jones (1988). This dataset consists of four financial ratios (Cash Flow to Total Debt; Net Income to Total Assets; Current Assets to Current Liabilities; Current Assets to Net Sales) of 21 bankrupt firms and 25 non-bankrupt firms. The second example introduced by Sueyoshi (2006) is related to the corporate bankruptcy in US electric power industry. This dataset contains 61non-default firms (class 1) and 22 default firms (class 2). The financial performance of all firms is characterised by 13 independent attributes (Cash to Total Assets; Net Working Capital to Total Assets; Sales to Total Assets; Long-term Debt to Total Assets; Shareholder Equity to Total Assets; Net Income to Total Assets; Retained Earnings to Total Assets; Return on Equity; Market to Book Ratio; Beta; Price over Earnings; Earnings per Share; Share Price). It should be mentioned that the non-default firms are currently providers of US electricity market. All the default firms have experienced their bankruptcy from 1996 to 2002. Data of both examples are listed in Tables 2.1 and 2.2.

Apart from the above two financial datasets, three more examples (Iris, Glass and *E. coli*) with multiple groups are selected from UCI machine learning database (http://kdd.ics.uci.edu). The third example used is the famous iris data (Fisher, 1936), which consists of 3 classes of 50 instances each, where each class refers to a type of iris plant. The class membership of each sample is characterised by four independent factors: sepal length, sepal width, petal length and petal width. One class is linearly separable from the other two and the remaining two classes are not linearly separable. The fourth example involves 214 training samples and considers the identification of six different types of glass based on 10 composite measurements for each sample. The last example includes an *E. coli* dataset with 336 protein sequences labelled to 8 classes of localisation sites. Two redundant attributes are deleted from the original

dataset and the rest 5 attributes are used for training and testing. The prediction performances for the proposed five examples under three different scenarios (A, B and C) are summarised in Table 2.4. The best performance for each scenario is indicated in bold.

Table 2.1 Financial ratios of bankrupt and non-bankrupt firms (Example 1)

Class 1- Bankrupt Firms					Class 2- Non-bankrupt Firms				
	Values of Financial Ratios					Values of Financial Ratios			
Firm	CF/TD	NI/TA	CA/CL	CA/NS	Firm	CF/TD	NI/TA	CA/CL	CA/NS
1	-0.4485	-0.4106	1.0865	0.4526	22	0.5135	0.1001	2.4871	0.5368
2	-0.5633	-0.3114	1.5134	0.1642	23	0.0769	0.0195	2.0069	0.5304
3	0.0643	0.0156	1.0077	0.3978	24	0.3776	0.1075	3.2651	0.3548
4	-0.0721	-0.0930	1.4544	0.2589	25	0.1933	0.0473	2.2506	0.3309
5	-0.1002	-0.0917	1.5644	0.6683	26	0.3248	0.0718	4.2401	0.6279
6	-0.1421	-0.0651	0.7066	0.2794	27	0.3132	0.0511	4.4500	0.6852
7	0.0351	0.0147	1.5046	0.7080	28	0.1184	0.0499	2.5210	0.6925
8	-0.0653	-0.0566	1.3737	0.4032	29	-0.0173	0.0233	2.0528	0.3484
9	0.0724	-0.0076	1.3723	0.3361	30	0.2169	0.0779	2.3489	0.3970
10	-0.1353	-0.1433	1.4196	0.4347	31	0.1703	0.0695	1.7973	0.5174
11	-0.2298	-0.2961	0.331	0.1824	32	0.1460	0.0518	2.1692	0.5500
12	0.0713	0.0205	1.3124	0.2497	33	-0.0985	-0.0123	2.5029	0.5778
13	0.0109	0.0011	2.1495	0.6969	34	0.1398	-0.0312	0.4611	0.2643
14	-0.2777	-0.2316	1.1918	0.6601	35	0.1379	0.0728	2.6123	0.5151
15	0.1454	0.0500	1.8762	0.2723	36	0.1486	0.0564	2.2347	0.5563
16	0.3703	0.1098	1.9941	0.3828	37	0.1633	0.0486	2.3080	0.1978
17	-0.0757	-0.0821	1.5077	0.4215	38	0.2907	0.0597	1.8381	0.3786
18	0.0451	0.0263	1.6756	0.9494	39	0.5383	0.1064	2.3293	0.4835
19	0.0115	-0.0032	1.2602	0.6038	40	-0.3330	-0.0854	3.0124	0.4730
20	0.1227	0.1055	1.1434	0.1655	41	0.4785	0.0910	1.2444	0.1847
21	-0.2843	-0.2703	1.2722	0.5128	42	0.5603	0.1112	4.2918	0.4443
					43	0.2029	0.0792	1.9939	0.3018
					44	0.4746	0.1380	2.9166	0.4487
					45	0.1661	0.0351	2.4527	0.137
					46	0.5808	0.0371	5.0594	0.1268

CF/TD=Cash Flow/Total Debt; NI/TA=Net Income/Total Assets; CA/CL=Current Assets/Current Liabilities; CA/NS=Current Assets/Net Sales

Table 2.2 Financial performance of non-default and default firms (Example 2)

Firm	C/TA	WC/TA	S/TA	LTD/TA	SE/TA	NI/TA	RE/TA	ROE	MK/BK	BETA	PE	EPS	PRICE
(CLASS 1) Non-default firms													
1	0.03	-0.01	0.25	0.53	0.15	0.01	0.08	8.43	1.70	1.77	16.35	0.88	16.35
2	0.00	-0.16	0.93	0.29	0.25	0.04	0.10	16.57	1.67	0.32	9.08	3.74	36.22
3	0.07	0.06	0.47	0.28	0.35	0.04	0.13	11.24	1.86	-0.11	15.09	1.70	25.20
4	0.02	-0.02	0.44	0.39	0.31	0.03	0.13	9.66	1.50	-0.02	12.65	2.30	30.36
5	0.01	-0.07	0.43	0.27	0.34	0.05	0.17	14.20	1.73	0.02	12.82	3.46	42.30
6	0.01	-0.12	1.43	0.21	0.17	0.02	0.07	12.19	1.67	-0.11	19.26	3.11	43.53
7	0.04	0.05	2.43	0.29	0.50	0.00	0.03	7.94	0.86	0.38	4.72	1.21	13.26
8	0.10	0.13	3.43	0.00	0.42	0.12	-0.11	29.49	1.06	0.64	3.42	2.60	9.40
9	0.02	-0.22	4.43	0.25	0.31	0.05	0.15	17.18	1.79	0.20	8.12	3.45	33.84
10	0.07	0.04	5.43	0.33	0.14	0.03	0.06	21.29	1.77	0.73	7.33	2.11	16.79
11	0.09	0.09	6.43	0.31	0.40	0.00	0.13	0.49	1.02	0.26	17.04	0.08	16.70
12	0.11	0.14	7.43	0.18	0.46	0.04	0.14	10.24	1.46	0.14	17.11	3.11	43.47
13	0.01	-0.09	8.43	0.29	0.24	0.04	0.11	15.04	1.83	-0.19	12.57	2.78	33.43
14	0.01	-0.11	9.43	0.35	0.29	0.04	0.19	14.31	2.03	0.08	15.47	1.56	21.97
15	0.01	-0.05	10.43	0.40	0.11	-0.03	-0.06	-17.51	1.61	0.42	70.68	-2.53	24.03
16	0.01	-0.21	11.43	0.22	0.21	0.04	0.03	29.87	1.72	0.15	5.34	4.27	24.49
17	0.02	-0.05	0.57	0.32	0.35	0.04	0.31	12.49	1.35	-0.03	12.60	3.21	40.36
18	0.01	-0.12	0.28	0.19	0.29	0.01	0.11	2.14	1.07	0.21	9.76	0.52	26.55
19	0.01	-0.06	0.31	0.34	0.30	0.02	0.03	6.50	1.98	0.10	19.33	2.17	60.10
20	0.00	0.01	0.28	0.51	0.20	0.05	0.21	26.20	3.71	0.12	14.15	1.81	24.08
21	0.00	-0.02	0.40	0.37	0.16	-0.05	0.24	-30.27	1.78	0.22	-16.75	-2.75	18.93
22	0.02	-0.02	0.41	0.39	0.24	0.02	0.10	7.17	1.51	-0.16	24.53	2.15	41.94
23	0.01	-0.02	1.23	0.25	0.27	0.04	0.13	15.60	2.44	-0.06	14.65	2.58	39.26
24	0.11	-0.06	0.31	0.34	0.09	0.03	0.04	73.41	4.49	-0.17	-1.34	7.37	15.10
25	0.02	0.02	0.47	0.36	0.27	0.04	0.16	14.63	1.65	0.26	11.07	1.30	14.50
26	0.01	-0.08	0.30	0.35	0.30	0.01	0.05	3.88	1.59	-0.14	27.27	0.59	21.00
27	0.06	0.05	0.52	0.34	0.30	0.03	0.14	10.53	1.27	0.25	12.75	1.61	18.99
28	0.01	-0.02	0.37	0.28	0.30	0.03	0.14	9.43	1.15	-0.02	11.50	3.18	39.11
29	0.02	-0.02	0.43	0.37	0.24	0.04	0.03	17.21	1.98	-0.06	13.60	4.42	47.88
30	0.01	-0.06	0.21	0.31	0.21	0.02	0.04	8.85	1.64	-0.06	12.49	2.85	34.98
31	0.00	-0.12	0.49	0.28	0.36	0.04	0.18	12.98	1.66	-0.01	13.06	4.63	56.40
32	0.02	-0.01	0.86	0.23	0.34	0.03	0.02	10.72	1.05	-0.31	13.32	1.93	18.65
33	0.05	-0.45	0.20	0.13	0.11	0.01	0.02	11.59	1.55	0.04	12.91	3.19	40.28
34	0.04	0.00	0 35	0.24	0.30	0.04	0.25	12.26	1.13	-0.09	8.80	3.33	29.56
35	0.04	-0.06	1.38	0.26	0.24	0.02	0.03	10.24	1.16	0.13	9.00	1.65	16.02
36	0.01	-0.12	0.54	0.33	0.20	0.01	0.05	6.10	1.37	0.17	22.81	1.03	23.06
37	0.01	0.01	0.67	0.22	0.22	0.02	0.07	12.56	1.10	0.45	9.09	1.97	17.63
38	0.02	-0.12	1.62	0.32	0.22	0.02	0.04	11.20	1.45	-0.11	13.72	2.03	21.05
39	0.03	-0.06	0.23	0.61	0.17	0.02	0.05	11.86	1.42	0.13	11.07	1.36	15.50
40	0.01	-0.11	0.60	0.26	0.25	0.00	0.06	-0.19	1.88	0.13	186.88	-0.05	44.85
41	0.16	0.14	0.64	0.20	0.43	0.03	-0.03	25.22	1.89	-0.09	-1.98	3.00	19.24
42	0.00	-0.05	0.57	0.33	0.31	0.04	0.13	13.10	1.42	-0.07	10.62	3.86	41.85
43	0.01	0.01	0.80	0.33	0.34	0.05	0.14	14.81	1.05	0.39	6.85	3.83	27.95
44	0.00	-0.02	0.78	0.37	0.34	0.04	0.14	8.96	1.30	-0.03	7.23	1.51	22.57
45	0.08	0.04	0.46	0.44	0.15	0.01	0.08	9.10	2.19	0.35	8.34	1.16	34.85
46	0.00	0.00	0.41	0.46	0.29	0.03	0.10	9.02	1.59	-0.03	14.72	2.65	45.03
47	0.01	-0.12	0.33	0.41	0.16	0.03	0.07	18.44	2.13	-0.01	11.37	3.67	42.19
48	0.02	-0.03	0.61	0.38	0.27	0.02	0.01	8.31	1.38	0.10	10.84	1.31	21.89
49	0.01	-0.11	1.51	0.19	0.22	0.03	0.10	13.39	1.17	0.16	13.46	3.17	26.52
50	0.01	0.00	0.61	0.38	0.34	0.03	0.07	8.91	1.67	0.07	16.35	2.00	37.60
51	0.03	0.00	0.44	0.34	0.30	0.07	0.16	24.57	1.37	-0.02	4.97	5.15	27.83
52	0.01	-0.04	0.56	0.41	0.21	0.01	0.00	1.76	0.89	0.22	376.25	0.34	15.05
53	0.01	-0.05	0.34	0.28	0.27	0.04	0.15	14.02	2.21	-0.35	16.36	1.62	25.35

Table 2.2 (continued)

Firm	C/TA	WC/TA	S/TA	LTD/TA	SE/TA	NI/TA	RE/TA	ROE	MK/BK	BETA	PE	EPS	PRICE
54	0.02	-0.17	0.39	0.27	0.32	0.05	0.19	15.40	1.90	0.02	11.66	2.26	26.24
55	0.03	-0.08	0.66	0.38	0.19	0.02	0.04	10.57	1.55	-0.19	13.71	3.12	47.15
56	0.13	0.09	0.20	0.37	0.30	0.03	-0.04	9.48	0.95	1.41	11.69	0.57	4.56
57	0.05	-0.02	0.58	0.27	0.27	0.03	0.12	11.87	1.47	0.18	12.89	4.21	51.30
58	0.08	-0.04	0.53	0.29	0.16	0.02	-0.09	14.33	1.48	0.30	9.83	1.83	18.19
59	0.01	0.01	0.47	0.39	0.25	0.03	0.15	10.14	1.27	-0.01	13.19	1.78	22.56
60	0.02	0.03	0.93	0.25	0.27	0.03	0.13	10.84	1.69	-0.03	14.28	2.75	36.55
61	0.02	-0.07	0.52	0.42	0.22	0.03	0.09	12.60	1.54	-0.01	12.06	2.28	27.74
(CLASS 2) Default firms													
62	0.03	-0.10	0.96	0.72	0.27	-0.28	-0.28	-91.47	0.22	1.13	-0.46	-1.93	0.84
63	0.03	-0.10	0.19	0.01	0.20	0.01	-0.73	-6.36	0.01	-0.22	0.08	-0.23	0.04
64	0.02	-0.05	0.69	0.17	0.33	-0.41	-1.10	-125.38	0.77	0.69	-2.31	-1.07	1.50
65	0.12	0.05	0.53	1.40	-0.57	-0.10	-2.26	20.17	-0.18	0.92	30.83	-1.54	0.93
66	0.02	-0.28	0.43	0.86	-0.22	-0.48	-0.67	220.16	1.99	0.29	-0.67	-10.21	4.00
67	0.03	-0.08	0.34	0.50	0.00	-0.07	-0.06	-3720	0.91	0.82	-4.00	-4.65	4.52
68	0.12	0.15	0.49	1.01	-0.15	-0.41	-0.32	237.02	0.23	-0.71	5.40	-9.67	0.69
69	0.06	0.02	0.45	0.39	0.33	0.01	0.02	4.47	0.37	0.69	5.03	0.47	3.88
70	0.02	0.45	2.89	0.13	0.18	0.01	0.05	8.66	6.12	0.58	61.57	1.22	83.13
71	0.00	-0.18	7.82	0.22	0.03	-0.01	-0.55	-22.62	5.05	0.04	20.62	-0.54	15.05
72	0.00	0.05	0.40	0.99	-0.15	-0.47	-0.66	313.70	0.33	0.86	-0.35	-12.65	2.63
73	0.03	-0.17	0.71	0.18	0.26	-0.13	-0.11	-48.87	0.61	1.29	-1.55	-2.30	3.56
74	0.04	0.14	0.48	0.72	-0.53	0.02	-1.02	-2.90	-0.16	1.33	-0.10	0.15	0.81
75	0.19	0.32	1.19	0.00	0.63	-1.21	-1.76	-193.51	0.12	2.15	-0.19	-6.38	0.74
76	0.02	0.22	1.49	0.06	0.49	-0.21	-0.59	-42.86	0.80	0.69	-0.97	-0.86	0.81
77	0.06	0.31	0.74	0.15	0.07	-0.16	-0.09	-314.34	1.78	0.69	5.40	-9.29	18.25
78	0.04	-0.17	0.55	0.93	-0.20	-0.29	-0.68	142.04	-2.01	0.25	-1.05	-1.74	0.86
79	0.03	-1.10	0.13	0.00	-0.21	-0.12	-0.74	59.18	-0.01	0.69	-0.02	-1.08	0.02
80	0.01	0.01	0.41	0.50	0.11	-0.13	-0.11	-111.42	0.35	0.51	-0.93	-6.60	3.60
81	0.02	-0.47	1.08	0.35	-0.01	-0.06	-0.09	166.68	-12.3	1.72	-1.36	-3.01	3.19
82	0.01	-0.45	0.43	0.23	-0.08	-0.22	-0.33	274.42	0.85	0.10	-1.05	-1.95	0.88
83	0.01	-0.71	0.14	0.00	0.15	0.01	-0.10	9.91	0.23	0.72	4.83	0.24	0.58

C/TA: Cash/Total Assets; NWC/TA: Net Working Capital/Total Assets; S/TA: Sales/Total Assets; LTD/TA: Long-term Debt/Total Assets; SE/TA: Shareholder Equity/Total Assets; NI/TA: Net Income/Total Assets; RE/TA: Retained Earnings/Total Assets; ROE: Return on Equity; MK/BK: Market/Book Ratio; BETA: Beta; PE: Price over Earnings; EPS: Earnings per Share; PRICE: Share Price

Table 2.3 Bounds of hyper-boxes for Example 1

Hyper-boxes &Classes	CF/TD		NI/TA		CA/CL		CA/NS	
	LB	UB	LB	UB	LB	UB	LB	UB
Box1-class1	-0.56	0.12	-0.41	0.11	0.33	1.68	0.16	0.95
Box2-class2	-0.33	0.58	-0.09	0.14	1.80	5.06	0.13	0.69
Box3-class2	0.14	0.48	-0.03	0.09	0.46	1.24	0.19	0.26

Table 2.4 Computation results for real datasets

Examples	Models	Scenario A	Scenario B	Scenario C
Example 1 (Firm 1)	Our work	**83.23%**	**81.83%**	**86.96%**
	Gehrlein (1986)	82.00%	81.69%	**86.96%**
	Sueyoshi (2006)	72.77%	69.53%	76.09%
	SMO	77.89%	77.56%	71.74%
	OneR	79.38%	81.38%	80.43%
	LDA	81.43%	76.28%	84.78%
	NN	78.81%	79.53%	71.74%
Example 2 (Firm 2)	Our work	91.33%	90.92%	91.57%
	Gehrlein (1986)	85.17%	86.67%	81.93%
	Sueyoshi (2006)	89.46%	89.41%	89.16%
	SMO	**95.25%**	**93.16%**	**95.18%**
	OneR	93.25%	92.55%	92.77%
	LDA	89.68%	90.24%	90.36%
	NN	90.31%	90.82%	91.56%
Example 3 (Iris)	Our work	95.33%	95.16%	97.33%
	Gehrlein (1986)	93.51%	93.64%	94.00%
	Sueyoshi (2006)	91.42%	91.91%	90.67%
	SMO	96.09%	96.67%	96.67%
	OneR	94.31%	94.22%	92.67%
	LDA	**96.84%**	**97.06%**	**98.00%**
	NN	95.13%	95.51%	96.67%
Example 4 (Glass)	Our work	**64.84%**[b]	**62.75%**[b]	**65.89%**[b]
	Gehrlein (1986)	57.16%[b]	56.68%[b]	56.07%[b]
	Sueyoshi (2006)	44.34%[a]	45.34%[a]	43.48%[a]
	LDA	61.07%	60.56%	64.95%
	SMO	55.38%	55.84%	54.67%
	OneR	55.85%	55.12%	55.14%
	NN	61.43%	59.97%	59.35%
Example 5 (*E. coli*)	Our work	**85.48%**[a]	**85.34%**[a]	**85.42%**[b]
	Gehrlein (1986)	79.20%[b]	78.98%[b]	79.17%[b]
	Sueyoshi (2006)	56.00%[b]	53.54%[b]	52.38%[b]
	SMO	81.25%	82.35%	82.14%
	OneR	64.08%	64.82%	64.88%
	LDA	84.81%	85.19%	85.12%
	NN	76.97%	75.95%	74.40%

a: some of the MILPs solved to optimality within 200 seconds; b: none of the MILPs solved to optimality within 200 seconds

Table 2.3 shows dimensions (lower and upper bounds) of hyper-boxes for each class as designed by our proposed approach using the complete dataset of example 1 for training. Three hyper-boxes successfully capture the patterns of bankrupt and non-bankrupt firms and only three samples are misclassified. The testing accuracy of the proposed approach is then evaluated and compared with six other classification methods over three testing scenarios (scenario A, B and C) listed above. According to the computational results (see Table 2.4), our hyperbox-based approach obtains the best prediction accuracy under all scenarios for three out of all five datasets studied. In the second example, our approach lists as the third best methods out of all six literature methodologies. In the Iris dataset (example 3), our method obtains slightly worse results than LDA, SMO and NN but better than OneR and the other two mathematical programming approaches.

2.8.2 Synthetic Datasets

In this section, two synthetic datasets are generated using different uniform distribution functions to evaluate the performance of our approach. In the first synthetic example, 100 samples with two attributes are generated belonging to two classes with 50 samples per class. Particular characteristics of the proposed datasets are that two classes of data are well separated but samples of each class distribute within different disjoint regions. The second synthetic dataset consists of 200 samples, 100 points for each class (50% of all training samples overlap). Table 2.5 provides details for the distribution of all synthetic data samples and Figure 2.6 and 2.7 graphically show both synthetic examples used in our study.

Table 2.5 Data distribution of synthetic examples

Data	Samples	Attribute 1	Attribute 2	Class membership
Synthetic Example 1	$s_1 - s_{25}$	U(1,2)	U(1,3)	Class1
	$s_{26} - s_{50}$	U(2,3)	U(2,4)	Class1
	$s_{51} - s_{75}$	U(1,2)	U(3,4)	Class 2
	$s_{76} - s_{100}$	U(2,3)	U(1,2)	Class2
Synthetic Example 2	$s_1 - s_{100}$	U(1,2)	U(1,2)	Class1
	$s_{101} - s_{200}$	U(1.5,2.5)	U(1,2)	Class2

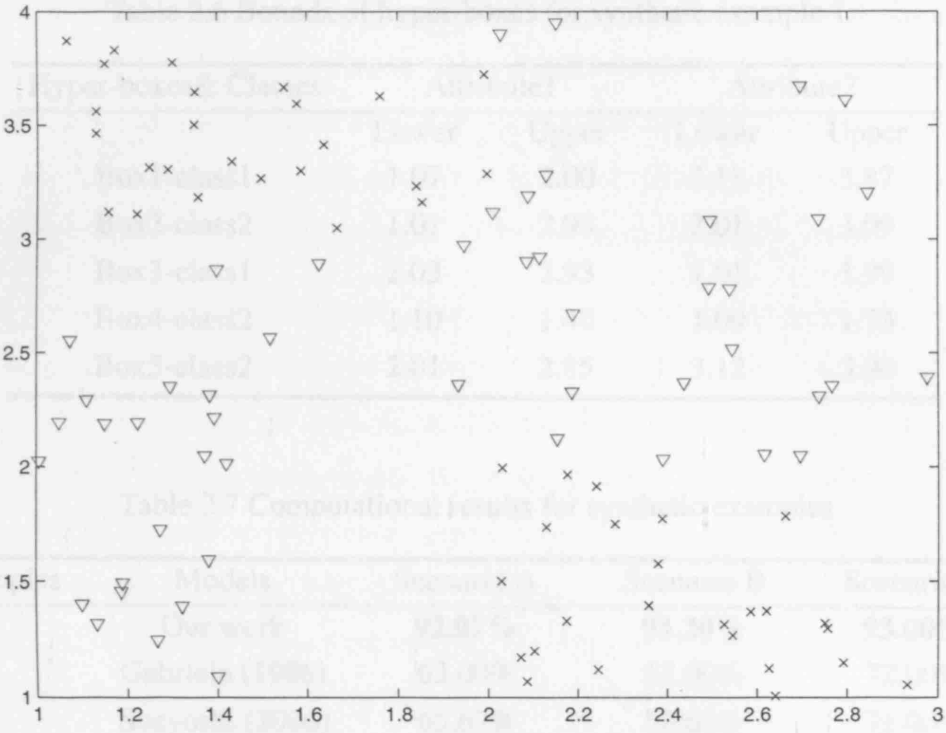


Figure 2.6 Synthetic example 1: Disjoint Data

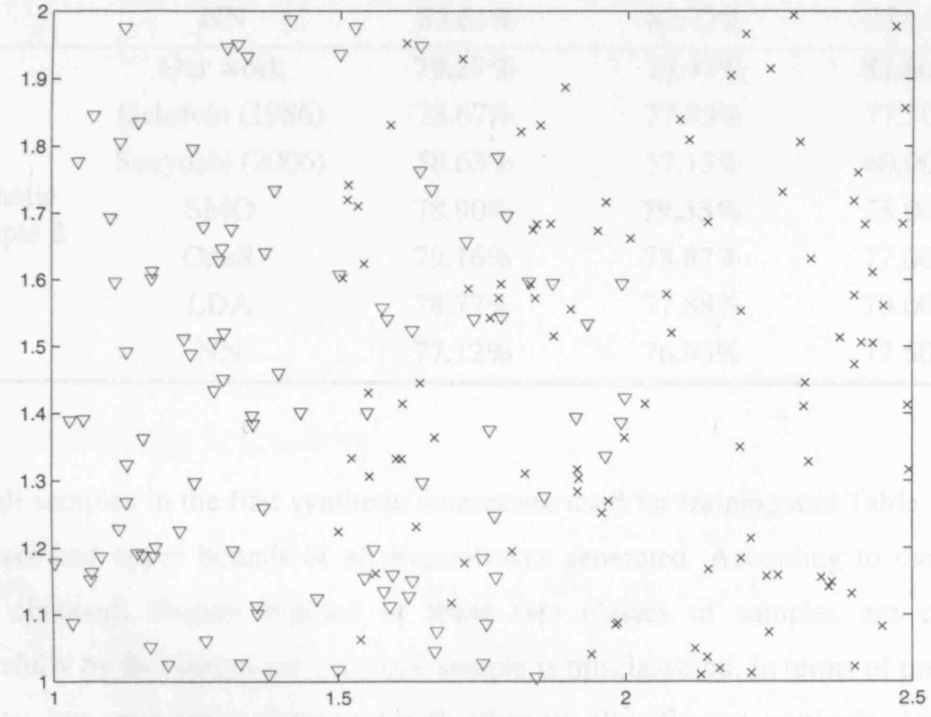


Figure 2.7 Synthetic example 2: Overlapped Data

Table 2.6 Bounds of hyper-boxes for synthetic example 1

Hyper-boxes& Classes	Attribute1		Attribute2	
	Lower	Upper	Lower	Upper
Box1-class1	1.07	2.00	3.11	3.87
Box2-class2	1.01	2.98	2.01	3.09
Box3-class1	2.03	2.93	1.01	1.99
Box4-class2	1.10	1.40	1.09	1.73
Box5-class2	2.01	2.85	3.12	3.90

Table 2.7 Computational results for synthetic examples

Examples	Models	Scenario A	Scenario B	Scenario C
Synthetic Example 1	Our work	**92.07%**	**91.20%**	**93.00%**
	Gehrlein (1986)	63.00%	62.00%	72.00%
	Sueyoshi (2006)	65.67%	63.60%	71.00%
	SMO	49.60%	48.30%	82.00%
	OneR	74.53%	73.67%	75.00%
	LDA	43.27%	44.20%	43.00%
	NN	83.61%	82.73%	87.00%
Synthetic Example 2	Our work	**79.27%**	78.47%	**81.00%**
	Gehrlein (1986)	78.67%	77.93%	77.50%
	Sueyoshi (2006)	58.63%	57.13%	60.00%
	SMO	78.90%	**79.33%**	78.00%
	OneR	79.16%	78.87%	77.00%
	LDA	78.77%	77.88%	79.00%
	NN	77.12%	76.93%	77.50%

First, all samples in the first synthetic dataset are used for training and Table 2.6 lists the lower and upper bounds of all hyper-boxes generated. According to the hyper-boxes obtained, disjoint regions of these two classes of samples are captured successfully by five boxes and only one sample is misclassified. In terms of prediction accuracy, our approach is compared with other six classification methods. According to the computational results obtained in Table 2.7, the proposed methodology outperforms other approaches over all three scenarios in the first synthetic example. In the second synthetic example, we show how our approach can solve not only disjoint

datasets but also examples with severe overlap characteristics. In this dataset, the distributions of each class of data samples partially overlap each other. Under the three testing scenarios, it can be seen that our method still gets the best performance among all seven classifiers over scenarios A and C. On scenario B, our approach achieves the third best prediction over all methodologies listed (see Table 2.7).

Finally, we adopt the following scoring scheme to evaluate the testing performance of each methodology over all seven examples studied (five real and two synthetic datasets). For each example and particular scenario, the method with the highest testing accuracy is given seven points; the second highest approach obtains six points and so on. The average score of each approach over all examples is reported in Table 2.8 and the highest score for each scenario is in bold. It can clearly be seen that our approach outperforms the other six methods in terms of prediction accuracy over all three testing scenarios.

Table 2.8 Average scores of each approach for all three scenarios

Approaches	Scenario A	Scenario B	Scenario C
Our work	**6.43**	**6.00**	**6.57**
Gehrlein (1986)	3.29	3.43	3.86
Sueyoshi (2006)	1.57	1.57	2.00
SMO	4.14	4.71	4.57
OneR	4.14	4.14	3.71
LDA	4.43	4.00	5.00
NN	4.00	4.14	4.29

2.9 Concluding Remarks

In this chapter, a rigorous mixed integer optimisation model for data classification problems has been proposed. In the training part, hyper-boxes are constructed to enclose the samples which belong to the same class. The optimal location and dimension of each box is determined by an MILP model so as to minimise the total number of misclassifications. An iterative solution procedure has been proposed to allow multiple boxes for each class. In the testing stage, the memberships of some new data samples have been identified by calculating the distances between testing samples to all existing hyper-boxes established in the training part. The unclassified

sample is then assigned to the closest box. Finally, the applicability of the methodology has been demonstrated through five literature examples and two synthetic datasets. The computational results indicate that our approach is competitive in terms of prediction accuracy when compared with other alternative classification methodologies.

As mentioned in several papers (Adem and Gochet, 2006; Gallagher *et al.*, 1997; Glen, 1999; Stam and Joachimsthaler, 1990), the existence of binary variables makes MIP-based classification models difficult to achieve optimality. Similarly to other MIP representations, our approach can only solve problems with relatively few training samples. In the next chapter, an efficient solution approach is developed to tackle datasets with more training instances and complex data structures.

Chapter 3

A Two-stage Optimisation-based Solution Approach for Data Classification

In this chapter, an efficient two-stage solution approach for data classification problems is developed using mixed integer optimisation. In the first stage, all training samples are initially partitioned into a number of disjoint regions without considering their class labels. In the second stage, the hyper-box based approach (Multi-HB) proposed in Chapter 2 is then applied to each partitioned region to complete training. This treatment decomposes all training samples into a number of independent batches thus reducing significantly the computational burden of approach (Multi-HB) and extending the applicability of hyper-box classifiers to larger and more complex data mining tasks.

3.1 Introduction

In Chapter 2, hyper-box classifiers have been developed to solve data classification problems using mixed integer optimisation. The applicability of the proposed optimisation approach (Multi-HB) has been demonstrated through a number of synthetic and real datasets. The computational results indicated that this approach is very competitive in terms of prediction accuracy when compared with other literature approaches. It is widely accepted that the existence of binary variables makes MIP-based classifiers difficult to achieve optimality. A number of mathematical programming-based heuristics have been developed to reduce the computational efforts when solving the corresponding MIP models. Stam and Ragsdale (1992) proposed a two-stage method where training samples that are more difficult to classify are identified in the first stage and analysed in more detail in the second stage. The computational results suggested that the approach is particularly suitable for discriminant problems with outlier contaminated data. Sueyoshi (2001) developed a modified two-stage heuristic and tested it on both real and synthetic datasets. This modified two-stage method outperformed standard LP-based discriminant analysis models (Freed and Glover, 1986) and the two-stage method by Stam and Ragsdale (1992). Recently, an extensive computational comparison of both two-stage mathematical programming approaches was performed by Glen (2006). Different objective functions such as minimisation of the sum of deviations (MSD) and maximisation of classification accuracy (MCA) are used to test the prediction performances of both approaches using synthetic and real datasets. The computational results showed MP-based methods are competitive when comparing with other methodologies such as Linear Discriminant Analysis (LDA). It is also noted that a single technique does not produce the best prediction under all data conditions. Different methods should therefore be considered in developing classification models, with the most appropriate method chosen for a particular problem.

In Chapter 2, an iterative solution approach, Multi-HB, was proposed to solve general data classification problems. This approach involves solving a series of MILP models by allowing possible assignment of multiple hyper-boxes for each single class so as to identify hidden patterns behind training datasets. Similar to other MIP-based methodologies, the proposed approach, Multi-HB, in Chapter 2 can only solve

problems with relatively few training samples. Such computational difficulties usually exist in MIP-based classifiers due to the existence of binary variables (Adem and Gochet, 2006; Gallagher *et al.*, 1997; Glen, 1999; Stam and Joachimsthaler, 1990). The aim of this chapter is to propose efficient solution algorithms based on hyper-box classifiers for general data classification problems so as to tackle data classification cases with more training samples and complex structures without compromising significantly the prediction accuracy. The basic idea is to develop an efficient decomposition scheme to split training samples into a number of batches. Each training batch involves fewer samples thus reducing the computational burden when using the Multi-HB approach.

This chapter is structured as follows: in the next section, two mixed integer linear programming (MILP) models are proposed to partition all training samples into two non-overlapped regions through the most separable attribute. An iterative decomposition scheme is developed in section 3.3 to split training samples further into multiple disjoint sub regions. Section 3.4 applies hyper-box classifiers to each partitioned region identified during the first stage to complete training using Multi-HB. Section 3.5 illustrates the applicability of the proposed solution approach through a number of synthetic and real datasets. Finally, some concluding remarks are made in section 3.6.

3.2 A Mathematical Model for Partitioning Training Samples

In order to reduce the computational efforts needed in the training stage of Multi-HB approach (see Chapter 2), a two-stage solution algorithm is proposed in this chapter. First, a decomposition scheme is developed to partition all training samples involved into a number of disjoint, non-overlapped regions without considering their class memberships. In the second stage, hyper-box classifiers are then applied to each region to complete training using Multi-HB.

In order to achieve the goal of the first stage, mathematical programming techniques are applied first to allocate all samples into two regions. An iterative splitting procedure is then developed in the next section to further partition all training samples into potentially multiple non-overlapping regions.

Here, a mixed integer linear programming (MILP) model (Bi-Split) is formulated to partition all samples into two non-overlapped regions. Two hyper-boxes with M dimensions are used to enclose all training samples. Non-overlapping constraints are introduced to avoid these two regions occupying the same position. Balancing constraints are proposed to avoid the existence of regions enclosing too many or too few samples. The sets, parameters and variables involved in the proposed model are listed below:

Indices

s	Sample ($s=s_1, s_2, \ldots, S$)
m	Attribute ($m=m_1, m_2, \ldots, M$)
m_f	Most separable attribute
i, j	Regions

Set

F	Feature selected for partition
Δ	Samples needed for partition

Parameters

A_{sm}	Value of sample s on attribute m
ε	Minimum distance between different regions.
LB_m	Lower bound of partitioned regions
UB_m	Upper bound of partitioned regions

The formulation is based on the following key variables:

Binary variables

E_s	1, if sample s is included in disjoint region 1; 0, if sample s is enclosed in disjoint region 2.

Y_{ijm} 0, if regions i and j do not overlap each other on attribute m; 1 otherwise

Continuous variables

LE_{im} Length of region i on attribute m

X_{im} Central coordinate of region i on attribute m

3.2.1 Data Enclosing Constraints

First, all samples should be enclosed either in regions i_1 or i_2. Therefore, the following constraints:

$$A_{sm} \geq X_{i_1,m} - \frac{LE_{i_1,m}}{2} \qquad \forall s \in \Delta, m \qquad (3.1)$$

$$A_{sm} \leq X_{i_1,m} + \frac{LE_{i_1,m}}{2} \qquad \forall s \in \Delta, m \qquad (3.2)$$

or

$$A_{sm} \geq X_{i_2,m} - \frac{LE_{i_2,m}}{2} \qquad \forall s \in \Delta, m \qquad (3.3)$$

$$A_{sm} \leq X_{i_2,m} + \frac{LE_{i_2,m}}{2} \qquad \forall s \in \Delta, m \qquad (3.4)$$

must be satisfied. These enclosing conditions (3.1-3.4) can mathematically be modelled in a mixed integer linear form:

$$A_{sm} \geq X_{i_1,m} - \frac{LE_{i_1,m}}{2} - U(1-E_s) \qquad \forall s \in \Delta, m \qquad (3.5)$$

$$A_{sm} \geq X_{i_2,m} - \frac{LE_{i_2,m}}{2} - U \cdot E_s \qquad \forall s \in \Delta, m \qquad (3.6)$$

$$A_{sm} \leq X_{i_1,m} + \frac{LE_{i_1,m}}{2} + U(1-E_s) \qquad \forall s \in \Delta, m \qquad (3.7)$$

$$A_{sm} \le X_{i_2,m} + \frac{LE_{i_2,m}}{2} + U \cdot E_s \qquad \forall s \in \Delta, m \qquad (3.8)$$

where binary variable E_s is used to indicate if sample s is allocated into region i_1 and U is a suitable upper bound. When binary variable E_s has a value of 1, constraints (3.1) and (3.2) are active and sample s is allocated to region i_1. Sample s is in region i_2 when E_s has a zero value. It should be noted that the above constraints (3.5-3.8) guarantee every sample in the training set is included in either region i_1 or i_2.

3.2.2 Non-overlapping Constraints

In order to guarantee that both regions do not overlap each other, the following inequality condition needs to be satisfied on *at least* one of M attributes:

$$X_{im} - X_{jm} \ge \frac{LE_{im} + LE_{jm}}{2} + \varepsilon \qquad \forall m, i, j \ne i \qquad (3.9)$$

where ε is defined as a small positive number in condition (3.9) to prevent two regions from sharing the same border on attribute m. The above non-overlapping condition can be modelled as the following mixed integer linear form:

$$X_{im} - X_{jm} + U \cdot Y_{ijm} \ge \frac{LE_{im} + LE_{jm}}{2} + \varepsilon \qquad \forall m, i, j \ne i \qquad (3.10)$$

where binary variable Y_{ijm} has a value of zero when regions i_1 and i_2 are non-overlapped on attribute m. Overlapping between these two regions is avoided by enforcing Y_{ijm} to the value of zero on *at least* one dimension:

$$\sum_{m=1}^{M} (Y_{ijm} + Y_{jim}) \le 2M - 1 \qquad \forall i < j \qquad (3.11)$$

In order to reduce the computational demands of the model, the above non-overlapping constraints (3.10 and 3.11) can be activated only on a number of selected attributes, F. Therefore inequality constraints (3.10) and (3.11) can be rewritten as:

$$X_{im} - X_{jm} + U \cdot Y_{ijm} \ge \frac{LE_{im} + LE_{jm}}{2} + \varepsilon \quad \forall m \in F, i, j \ne i \qquad (3.12)$$

$$\sum_{m\in F}(Y_{ijm}+Y_{jim})\le 2|F|-1 \qquad \forall i<j \tag{3.13}$$

where $|F|$ denotes the cardinality of set F. In this chapter, we only select the most separable feature, m_f, to perform the partition (see section 3.2.4 to select the most separable feature). Therefore, constraint (3.13) can be eliminated and the non-overlapping constraint can be simplified as:

$$X_{i_1m_f}-X_{i_2m_f}\ge\frac{LE_{i_1m_f}+LE_{i_2m_f}}{2}+\varepsilon \tag{3.14}$$

3.2.3 Boundary Conditions

The lower and upper bounds of regions i_1 and i_2 should not exceed particular boundary conditions:

$$X_{im}-\frac{LE_{im}}{2}\ge LB_m \qquad \forall i,m \tag{3.15}$$

$$X_{im}+\frac{LE_{im}}{2}\le UB_m \qquad \forall i,m \tag{3.16}$$

where LB_m and UB_m denote the lower and upper bounds of sub region i. These boundaries will be updated as the minimum/maximum values of the parent region of i_1 and i_2. Such an iterative algorithm is applied later in this chapter (see section 3.3) to decompose all training samples into multiple regions. Setting appropriate boundaries of each region guarantees all partitioned regions are not overlapped each other.

3.2.4 Balancing Constraints

Next, we describe how the maximum number of samples included in each region is enforced. Developing such constraints is able to balance the number of samples enclosed in each region thus avoiding the existence of regions including too many or too few samples. Since binary variable E_s is introduced to indicate if sample s is enclosed in region i_1, $\sum_s E_s$ denotes the total number of samples in region i_1 and $\sum_s(1-E_s)$ indicates the number of samples in region i_2. The following balancing

constraints enforce that the number of training samples enclosed in each region are less than *N_S*:

$$\sum_s E_s \le N\_S \tag{3.17}$$

$$\sum_s (1 - E_s) \le N\_S \tag{3.18}$$

Here, *N_S* is a user-defined parameter which should be greater than 50% of the total number of samples involved to guarantee all samples are included. In this chapter, *N_S* is set as 2/3 of all partitioned samples. Overall, the proposed MILP model to partition all training samples into two disjoint regions is to find an feasible solution (lower and upper bounds of each region) so as to satisfy:

[Bi-Split]

Data enclosing constraints: (3.5)-(3.8)

Non-overlapping constraint: (3.14)

Boundary conditions: (3.15) and (3.16)

Balancing constraints: (3.17) and (3.18)

$E_s, Y_{ijm} \in \{0,1\}; LE_{im} \ge 0$

In the Bi-Split model, we aim at finding a feasible solution satisfying all constraints listed above. This can be achieved by optimising a pseudo objective function (i.e. maximise *Obj* where *Obj* is fixed to a constant) under all constraints required. It is also noted that non-overlapping constraint (3.14) are only active on the most separable feature, m_f. Next, we propose an MILP model to find such an attribute. Recently, Frank and Rubin (2003) described an MILP formulation for feature selection and data transformation based on maximisation of pairwise class separability. Here, a simplified MILP formulation is proposed below to identify the most separable attribute among all classes using the same optimisation objective. The indices, parameters and variables used in the proposed MILP model (MILP-FS) are listed below:

Indices

m Attribute ($m=m_1, m_2, \ldots, M$)

c, c' Class ($c = c_1, c_2 \ldots C$)

Parameters

$B_{cc'm}$ Separability of samples in classes c and c' on attribute m

r The total number of selected features

Binary variables

EY_m 1, if attribute m is selected; 0 otherwise

The objective function is maximisation of the separability of all pairs of classes on all attributes:

$$\text{Max} \quad \sum_{c}\sum_{c'}\sum_{m} B_{cc'm} \cdot EY_m \tag{3.19}$$

Binary variable EY_m is used to indicate if feature m is selected. Selecting at most r features can mathematically be modelled as:

$$\sum_{m} EY_m \le r \tag{3.20}$$

In this chapter, parameter r is set as 1 to find the most separable feature, m_f. The term $B_{cc'm}$ in the objective function denotes the separability function between classes c and c' on attribute m and is defined by:

$$B_{cc'm} = \tanh\left(c \cdot \frac{|\mu_{cm} - \mu_{c'm}|}{\sigma_{cm} \cdot \sigma_{c'm} \Big/ \sigma_{cm} + \sigma_{c'm}}\right) \qquad \forall c, c', m \tag{3.21}$$

where μ_{cm} and σ_{cm} are the mean value and standard deviation of samples belonging to classes c on attribute m. It should be added that model MILP-FS is resolved for all

training samples in every partition level described in section 3.3 to obtain the most separable attribute for each region to be partitioned.

3.3 An Iterative Bi-Partition Procedure

In this section, an iterative partition procedure is introduced to enclose all training samples into multiple sub regions using the MILP models proposed in section 3.2. The algorithm starts by including all samples into two regions using models Bi-Split and MILP-FS. If the number of samples inside a sub region is more than a user-defined parameter, N_P, both models (Bi-Split and MILP-FS) are used to partition that region further into two smaller sub regions. The algorithm stops when the number of enclosed samples in each sub region is less than N_P. Figure 3.1 illustrates the detailed partition procedure where all samples are included in five regions. First, all training samples are partitioned into two regions (R_1 and R_2). Since both regions contain more than N_P samples (indicated as white boxes at level 1 of Figure 3.1), R_1 is further decomposed in to regions R_{11} and R_{12} while R_2 is partitioned into R_{21} and R_{22}. It is noted that the lower and upper bounds of sub regions R_{11} and R_{12} are the minimum and maximum values of their parent region R_1 while the boundaries of region R_2 are set as the boundary conditions of R_{21} and R_{22}. Such treatment can guarantee that all current partitioned regions (R_{11}, R_{12}, R_{21} and R_{22}) do not overlap each other. Sub regions R_{11}, R_{12} and R_{22} are highlighted as grey boxes as they contain fewer than N_P samples thus no further partition is required. Region R_{21} is then partitioned one level further to guarantee all disjointed regions meet the termination criterion. Figures 3.2 to 3.4 demonstrate graphically how two classes of samples are partitioned into five sub regions indicated in Figure 3.1. Solid lines in Figures 3.4 indicate final partitions while partitions of the previous levels are highlighted as dash lines.

For each particular partition level, disjoint regions i_1 and i_2 are independent so that further partitioning of one region will not affect the other regions. For example, partitioning region R_1 into sub regions R_{11} and R_{12} in Figure 3.1 does not influence the partition procedure of region R_2. Therefore, the proposed iterative bi-partition procedure can also be parallised by simply distributing each disjoint region into different machines so as to accomplish the partition.

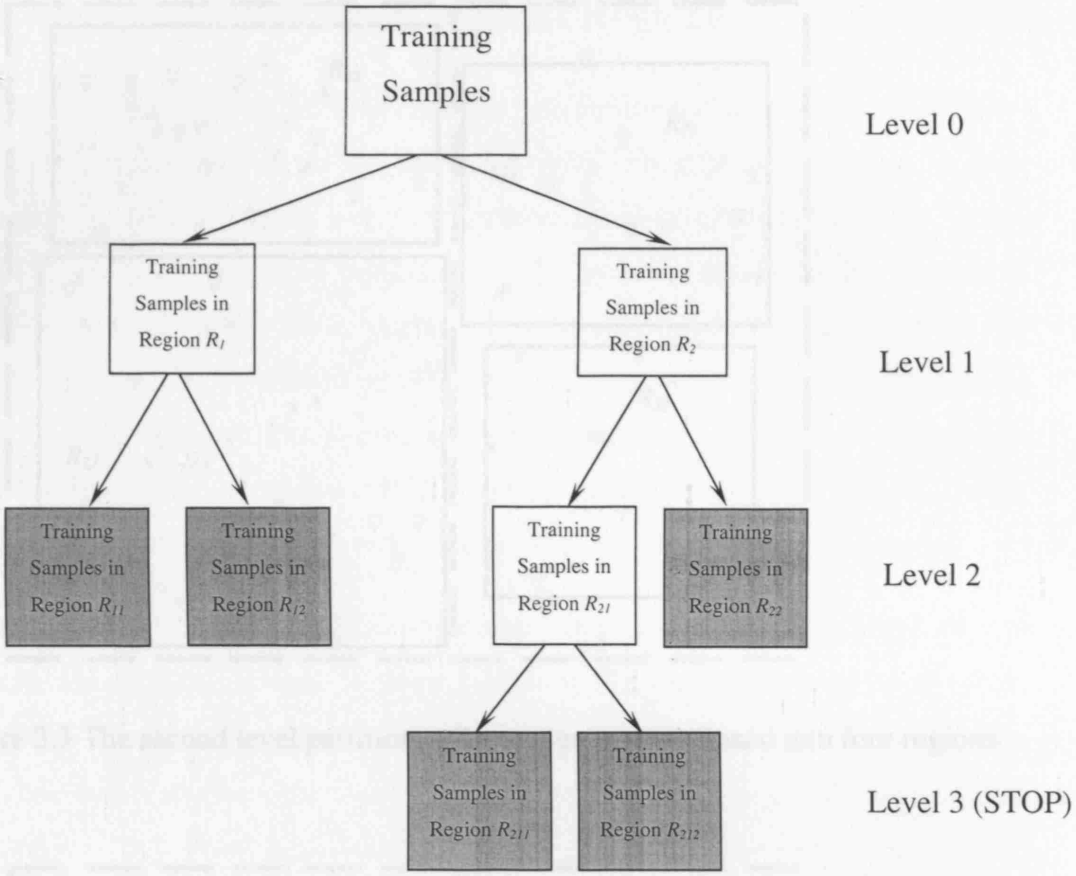


Figure 3.1 Schematic representation of splitting procedure.

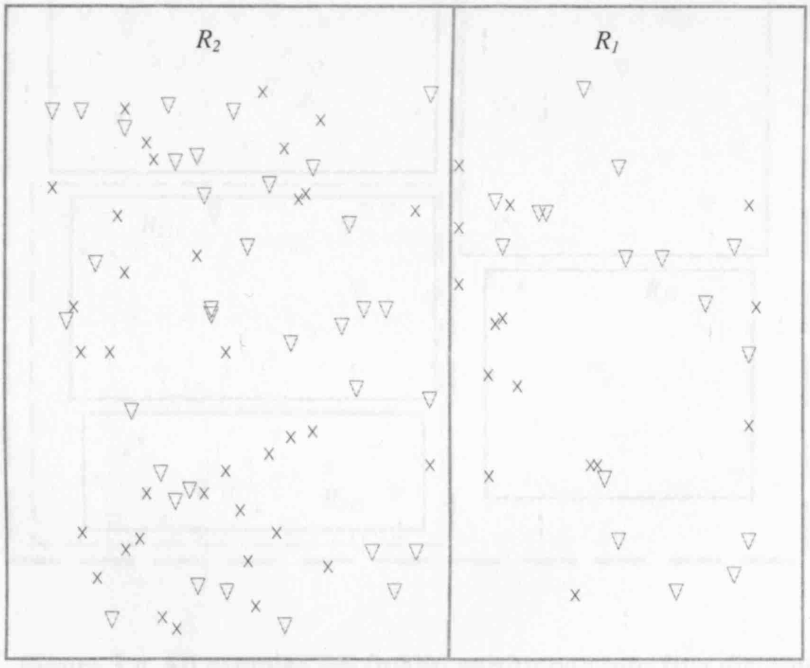


Figure 3.2 The first level partitions: all training samples are included in two regions

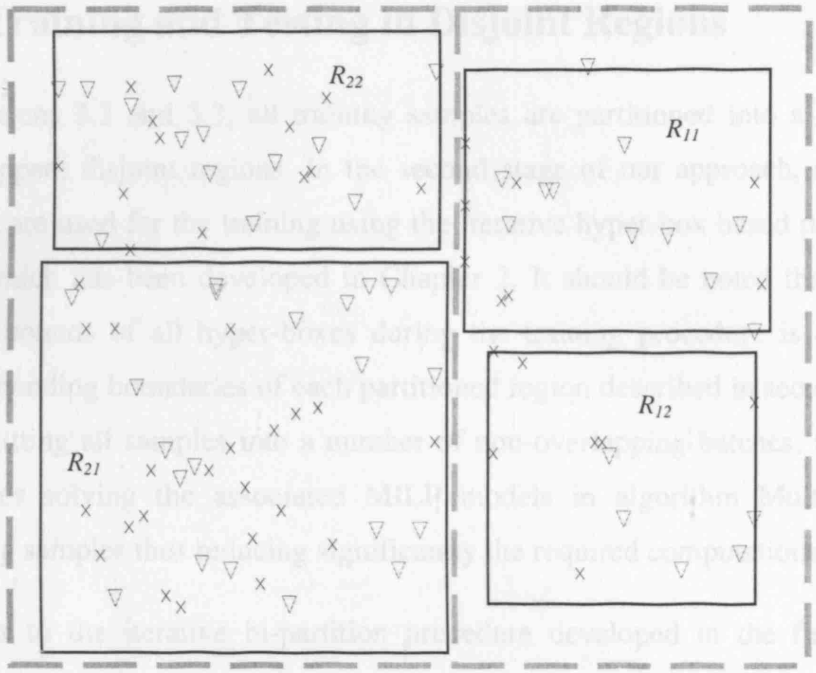


Figure 3.3 The second level partition: all samples are partitioned into four regions

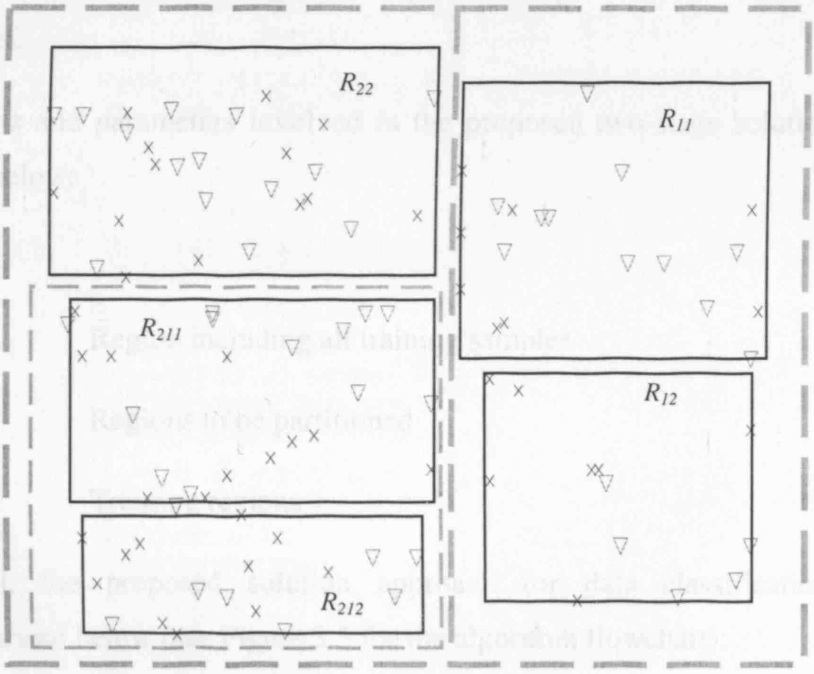


Figure 3.4 All samples are finally partitioned into five disjoint regions

3.4 Training and Testing in Disjoint Regions

In sections 3.2 and 3.3, all training samples are partitioned into a number of non-overlapped, disjoint regions. In the second stage of our approach, samples in each region are used for the training using the iterative hyper-box based procedure, Multi-HB, which has been developed in Chapter 2. It should be noted that the lower and upper bounds of all hyper-boxes during the training procedure is enforced by the corresponding boundaries of each partitioned region described in sections 3.2 and 3.3. By splitting all samples into a number of non-overlapping batches, the training task involves solving the associated MILP models in algorithm Multi-HB on fewer training samples thus reducing significantly the required computational resources.

Similar to the iterative bi-partition procedure developed in the first stage of our approach, the training of each disjoint region is also independent of each other. Hyper-boxes can be obtained through training samples in each partitioned regions using Multi-HB. Therefore, the training task can be completed in parallel on different computers to finish training of each partition region without considering other data samples.

The sets and parameters involved in the proposed two-stage solution algorithm are listed below:

Sets

I	Region including all training samples
Θ	Regions to be partitioned
Φ	Training regions

Overall, the proposed solution approach for data classification problems is summarised below (see Figure 3.5 for the algorithm flowchart):

[Algorithm MILP-decomp]

STEP 1: Initialisation $\Theta = I$. $\Phi = \emptyset$.

STEP 2: Solve MILP-FS to find the most separable attribute for each region in Θ.

STEP 3: Solve Bi-Split to partition each region in Θ into two disjoint regions i_1 and i_2. Update Θ by replacing parent region i by sub regions i_1 and i_2.

STEP 4: Move all regions in Θ with no more than N_P samples in Φ. If $\Theta = \emptyset$, go to STEP 5; otherwise, go to STEP 2.

STEP 5: Solve MILP-HB to each derived region in Φ to obtain hyper-box classifiers.

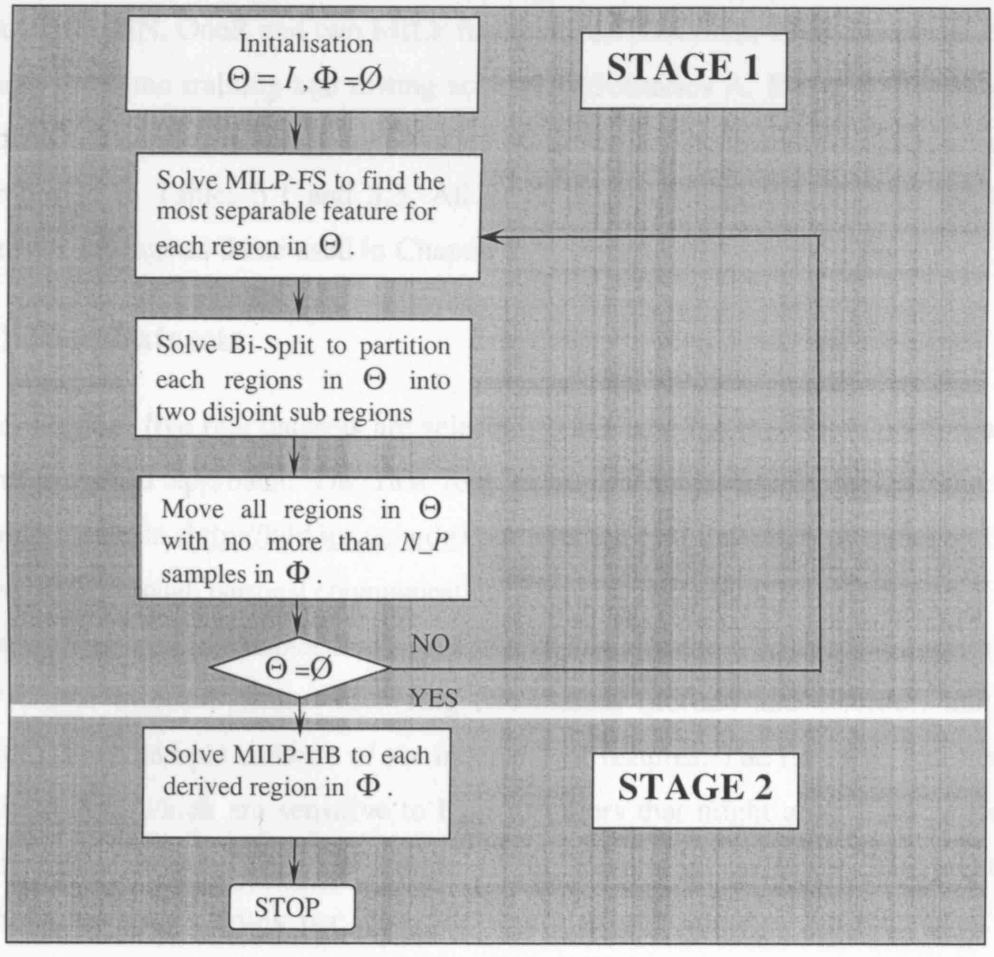


Figure 3.5 Flowchart of the proposed two-stage data classification algorithm

After training, the same testing scheme introduced in Chapter 2 is applied to predict the class membership of any new sample. The distances between testing sample s to all hyper-boxes obtained from the training procedure are calculated and sample s will then be allocated to the nearest existing hyper-box.

3.5 Computational Results

In this section, the applicability of the proposed two-stage approach (MILP-decomp) is demonstrated via a number of synthetic and real examples. All mathematical models and solution algorithms are implemented in GAMS modelling system (Brooke, *et al.*, 2003) on a 3.40 GHz PC with 2GB memory using the CPLEX mixed-integer optimisation solver with 1% margin of optimality for the branch and bound procedure. Each MILP model is solved within 200 seconds. In our two-stage solution algorithm, different values of N_P (50 and 100) are tried and the corresponding prediction accuracies are reported respectively. Moreover, the prediction performances of the proposed approach are compared with six different classification models including LDA, SMO, NN, OneR and two MILP formulations (Gehrlein, 1986; Sueyoshi, 2006) through the same training and testing scenarios (Scenarios A, B and C) described in Chapter 2. Scenarios A and B are repeated 50 times and the mean prediction accuracy are reported in Tables 3.1 and 3.3. All other literature approaches adopt the same parameter settings as those used in Chapter 2.

3.5.1 Real Datasets

In this section, five real datasets are selected to evaluate the prediction performances of the proposed approach. The first four examples are available in UCI machine learning database (http://kdd.ics.uci.edu) and the last real dataset is provided by Dr. P. Angeli, UCL, through personal communication. The first two examples (Glass and *E. coli* datasets) appeared in Chapter 2 are revisited in this chapter. The third example is the BUPA liver disorder dataset involving two classes (normal and disorder) and 345 samples. Each sample consists of six independent features. The first five features are all blood tests which are sensitive to liver disorders that might arise from excessive alcohol consumption. The last feature is the number of half-pint equivalents of alcoholic beverages drunk per day. The fourth dataset contains the records of 768 diabetes patients (two classes). Each sample includes eight features. The last real

dataset reflects the flow patterns of gas-liquid two phase flows in microreactors. All 524 samples are partitioned into three distinct classes (Taylor, Bubbly and Thrun flows) and each sample is characterised by five dimensionless numbers. The prediction performances for the selected five real examples under three different scenarios (A, B and C) are summarised in Table 3.1 and the best performance for each scenario is indicated in bold.

Table 3.1 Comparative prediction accuracies for real datasets

Examples	Models	Scenario A	Scenario B	Scenario C
Glass	MILP-decomp50[d]	**66.00%**	**64.66%**	**67.76%**
	MILP-decomp100	65.78%[a]	64.59% [a]	**67.76%** [a]
	Gehrlein (1986)	57.16% [b]	56.68% [b]	56.07% [b]
	Sueyoshi (2006)	44.34% [b]	45.34% [b]	43.48% [b]
	SMO	61.07%	60.56%	64.95%
	OneR	55.38%	55.84%	54.67%
	LDA	61.07%	60.56%	64.95%
	NN	61.43%	59.97%	59.35%
E. coli	MILP-decomp50	83.76% [a]	84.02% [a]	84.21%[a]
	MILP-decomp100	84.16% [a]	84.04% [a]	84.42%[a]
	Gehrlein (1986)	79.20% [b]	78.98% [b]	79.17% [b]
	Sueyoshi (2006)	56.00% [b]	53.54% [b]	52.38% [b]
	SMO	81.25%	82.35%	82.14%
	OneR	64.08%	64.82%	64.88%
	LDA	**84.81%**	**85.19%**	**85.12%**
	NN	76.97%	75.95%	74.40%
Liver	MILP-decomp50	65.40% [a]	64.66% [a]	66.08% [a]
	MILP-decomp100	**67.75%** [a]	66.85% [a]	69.65%[a]
	Gehrlein (1986)	49.70% [b]	50.25% [b]	48.70%[b]
	Sueyoshi (2006)	64.91% [b]	64.61% [b]	66.67%[b]
	SMO	57.42%	57.90%	57.97%
	OneR	55.89%	56.22%	53.33%
	LDA	65.98%	66.01%	**69.85%**
	NN	67.61%	**67.87%**	67.24%

a: some of MILPs are not solved to optimality; b: all MILPs are not solved to optimality; c: can not return any solution after 10000 seconds; d: the number after MILP-decomp denotes the value of *N_P* used

Table 3.1 (continued)

Examples	Models	Scenario A	Scenario B	Scenario C
Diabetes	MILP-decomp50[d]	69.77% [a]	68.24% [a]	66.80% [a]
	MILP-decomp100	72.38% [a]	71.20% [a]	73.44% [a]
	Gehrlein (1986)	**76.75%** [b]	**76.46%** [b]	76.95% [b]
	Sueyoshi (2006)	63.77% [b]	64.09% [b]	64.98%
	SMO	68.57%	68.24%	68.75%
	OneR	62.91%	61.13%	60.81%
	LDA	76.70%	76.39%	**77.34%**
	NN	N/A[c]	N/A[c]	N/A[c]
Flow	MILP-decomp50	78.89%	79.11%	79.92%
	MILP-decomp100	83.55%	83.10%	83.23%
	Gehrlein (1986)	83.26%	82.45%	83.40%
	Sueyoshi (2006)	57.70%	57.54%	58.26%
	SMO	82.32%	81.51%	82.06%
	OneR	82.01%	81.02%	80.72%
	LDA	83.33%	82.97%	78.81%
	NN	**86.32%**	**87.54%**	**86.06%**

a: some of MILPs are not solved to optimality; b: all MILPs are not solved to optimality; c: can not return any solution after 10000 seconds; d: the number after MILP-decomp denotes the value of *N_P* used

The first two real examples (Glass and *E. coli* datasets) solved by the proposed hyper-box approaches in Chapter 2 are revisited here. Comparing with the iterative method (Multi-HB) in Chapter 2, better prediction accuracies are achieved for the Glass example while slightly worse performance is obtained for *E. coli* example using MILP-decomp. For the following three real examples (Liver, Diabetes and Flow datasets), the iterative hyper-boxes approach (Multi-HB) in Chapter 2 becomes very difficult to solve. In comparison, the proposed solution approach in this chapter is able to obtain very promising prediction accuracies when compared to other competing literature methodologies (see Table 3.1).

It is not surprising that fewer disjoint regions are obtained when partitioning all samples using larger *N_P* values in the first stage of our approach. It is indicated that more samples are to be trained simultaneously in the second stage through hyper-box classifiers (Multi-HB). Therefore, more demanding computational resources are needed when larger *N_P* are used (see Table 3.1).

3.5.2 Synthetic Datasets

In this section, two synthetic datasets are generated to evaluate the performance of our two-stage approach (MILP-decomp). Both datasets consist of 400 samples with two classes and adopt the same data distribution functions introduced in Chapter 2. (see Table 3.2).

Table 3.2 Data distribution of synthetic examples

Data	Samples	Attribute 1	Attribute 2	Class membership
Synthetic Example 1	$s_1 - s_{100}$	U(1,2)	U(1,3)	Class1
	$s_{101} - s_{200}$	U(2,3)	U(2,4)	Class1
	$s_{201} - s_{300}$	U(1,2)	U(3,4)	Class 2
	$s_{301} - s_{400}$	U(2,3)	U(1,2)	Class2
Synthetic Example 2	$s_1 - s_{200}$	U(1,2)	U(1,2)	Class1
	$s_{201} - s_{400}$	U(1.5,2.5)	U(1,2)	Class2

Table 3.3 Comparative results for synthetic datasets

Examples	Models	Scenario A	Scenario B	Scenario C
Synthetic Example 1	MILP-decomp50[d]	98.00%	98.18%	98.50%
	MILP-decomp100	**98.47%**	**98.27%**	**99.00%** [a]
	Gehrlein (1986)	65.55% [b]	64.58% [b]	58.00% [b]
	Sueyoshi (2006)	64.52% [b]	64.67% [b]	73.00% [b]
	SMO	56.00%	53.00%	48.75%
	OneR	70.00%	69.00%	66.00%
	LDA	48.77%	50.45%	51.75%
	NN	80.12%	77.85%	81.25%
Synthetic Example 2	MILP-decomp50	75.50%	73.40%	70.00%
	MILP-decomp100	**78.35%**	**77.97%**	**79.50%**
	Gehrlein (1986)	76.33%	75.43%	73.50%
	Sueyoshi (2006)	60.85%	60.77%	60.25%
	SMO	76.47%	75.88%	77.75%
	OneR	76.66%	76.58%	77.50%
	LDA	75.82%	75.75%	77.25%
	NN	76.93%	76.17%	76.75%

a: some of MILPs are not solved to optimality within 200 seconds; b: all MILPs are not solved to optimality within 200 seconds
d: the number after MILP-decomp denotes the value of *N_P* used

Table 3.3 summarises the comparative results of our approach with other six literature methodologies. The computational results clearly show that our approach MILP-decomp100 achieves the best prediction accuracy over all three testing scenarios. Furthermore, it is noted that our approach obtains very high prediction accuracies (98% to 99%) when classifying samples with disjoint regions due to the hyper-box nature of our approach (see the synthetic example 1 in Table 3.3).

It is widely accepted that no approach can be superior to all other methodologies for all examples examined (Lam and Moy, 2002; Adem and Gochet, 2006). The method which obtains the best prediction accuracy on one dataset may perform badly on the other. The prediction performances are subject to a number of issues including the distribution of data samples, the mathematical structures of classifiers and the parameter values specified. In order to justify the prediction competitiveness of each approach compared in this chapter, the same scoring scheme proposed in Chapter 2 is applied to evaluate the testing performance of each methodology over all seven examples studied (five real and two synthetic datasets). For each example and particular scenario, the method with the highest testing accuracy is given eight points; the second highest approach obtains seven points and so on. The average score of each approach over all examples is reported in Table 3.4 and the best average score for each scenario is highlighted in bold. It can clearly be seen that our approach (MILP-decomp 100) outperforms all other six methods in terms of prediction accuracy over all three testing scenarios.

Table 3.4 Average scores of each approach for all three scenarios

Approaches	Scenario A	Scenario B	Scenario C
MILP-decomp50	5.00	5.00	4.86
MILP-decomp100	**7.29**	**7.14**	**7.14**
Gehrlein (1986)	4.29	4.00	4.29
Sueyoshi (2006)	2.14	2.43	2.71
SMO	4.00	4.29	4.71
OneR	3.29	3.57	3.43
LDA	5.14	5.43	5.71
NN	5.43	5.43	4.86

3.6 Concluding Remarks

In this chapter, an efficient two-stage solution approach (MILP-decomp) for data classification problems has been developed using mixed integer optimisation. In the first stage, an iterative algorithm has been proposed to partition all training samples into a number of non-overlapped, disjoint regions without considering the class membership of each sample. In the second stage, hyper-box classifiers (Multi-HB) have been applied to each region to complete the training task. Such approach involves solving relatively smaller MILP models comparing with Multi-HB thus extending the applicability of mathematical programming-based classification methodologies to large and complex datasets. The prediction accuracy of the proposed solution approach has been compared with other literature methodologies under three different testing scenarios through a number of synthetic and real datasets. It is shown that the proposed two-stage solution algorithm (MILP-decomp100) has successfully achieved competitive predictions within modest computational requirements. It is also noted that no more than 100 samples are included in each partitioned disjoint region is suitable for all cases selected in this chapter to achieve satisfactory prediction accuracies. Finally, we should emphasise that the solution approach described in this chapter has a highly parallel structure which can be further exploited to improve the computational efficiency.

PART II

NETWORK COMMUNITY STRUCTURE IDENTIFICATION

Chapter 4

Finding Community Structures in Complex Networks using Mixed Integer Optimisation

Complex networks are general and flexible to virtually represent any natural/synthetic systems. Many real networks of interest including social, technological and biological networks are found to divide naturally into communities. Such modular structures usually shed light on distinct functionalities. The detection of community structure has been used to reveal the relationships between individual objects and their groupings in networks. Since the quality of partitioning a network into communities was measured by the M*odularity* metric (Newman and Girvan, 2004), the optimal community structure a network can be achieved by optimising its network modularity value. This chapter presents a mathematical programming approach to identify optimal community structures in complex networks based on the maximisation of network modularity metric for partitioning a network into modules. The proposed optimisation-based methodology can be solved to global optimality using standard optimisation software. Special symmetry-breaking constraints are developed to

eliminate equivalent solutions. Additional features such as minimum/maximum module size and balancing among modules can easily be incorporated in the model. The applicability of the proposed approach is demonstrated by a number of illustrative examples.

4.1 Introduction and Literature Survey

Managing complexity is one of the major concerns in current scientific research especially in the area of process systems engineering. Modelling process systems across time and scale inevitably need analysis and management of system complexity. Proper representations of complex systems are crucially important for researchers to gain deeper insights into such systems and propose appropriate modelling frameworks.

Many complex systems such as the Internet, process plant flowsheets, social and biological relations have been represented as networks consisting of a set of nodes joined in pairs by edges to reflect the number of components in the systems and connections among them. Complex networks research can be considered as lying at the intersection among graph theory, statistical mechanics and other physical and engineering sciences. One of the main reasons behind complex networks popularity is their flexibility and generality for representing virtually any natural structure including complex systems evolution undergoing dynamic changes of topology.

The study of complex networks can be traced back to the pioneering works on percolation and random graphs by Flory (1941), Rapoport (1951, 1953, 1957), and Erdos and Rényi (1959, 1960, 1961). Recently, the study of complex networks has attracted more attention of the research community of many disciplines since many real networks have been found to posses characteristics which are not explained by uniformly random connectivity. Instead, networks derived from real data involve community structures, power law degree distributions and hubs. For example, statistical analysis of networks has revealed a number of properties such as small world effects, degree distribution and high network transitivity (Barabasi and Albert, 1999; Newman, 2003; Boccaletti *et al.*, 2006).

Although graph theory is a well-established and developed area in mathematics and has been widely used to study networks, many of the recent developments in complex networks have taken place in various areas such as sociology, biology and physics. Supported by the availability of high performance computers and large data collections, results like the discovery of the scale-free structure of the Internet (Faloutsos *et al.*, 1999) and of the WWW (Barabasi *et al.*, 1999) were of major importance for the increased interest in the new area of complex networks, whose growing relevance has been substantiated by the large number of recent related publications.

Four main types of complex networks include weighted directed, unweighted directed, weighted undirected and unweighted undirected networks. The operation of *symmetry* can be used to transform a directed network into an undirected graph and the operation of *thresholding* can be applied to transform a weighted graph into its unweighted counterpart. In this study, the proposed mathematical frameworks are based on unweighted and undirected networks.

The random graph developed by Rapoport (1951, 1953, 1957) and independently by Erdos and Rényi (1959, 1960, 1961) can be considered the most basic model of complex networks. Starting with N disconnected vertices, the network is constructed by the addition of M edges at random, avoiding multiple and self connections. Furthermore, many real world networks exhibit what is called the *small world* property, i.e. most vertices can be reached from the others through a small number of edges. This characteristic is found, for example, in social networks, where everyone in the world can be reached through a short chain of social acquaintance (Watts, 1999).

Apart from investigation of large-scale statistical properties of networks mentioned above, recent research has focused on the study of detailed properties of networks: network communities. Community structures are often found in various types of networks where the vertices are naturally clustered into tightly connected modules with large number of within-module edges and few inter-module links (see Figure 4.1).

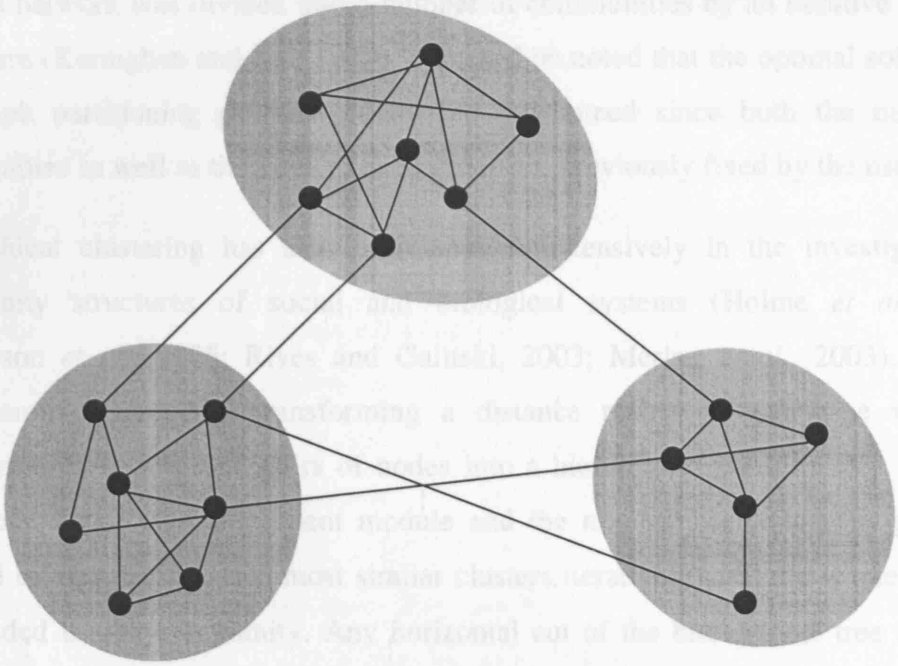


Figure 4.1 A representative network with three communities

The ability to identify and analyse such structures could be of vital importance in practice. For example, groups within the World Wide Web may reveal the thematic relationships of websites on similar topics (Eckmann and Moses, 2002; Flake *et al.*, 2002); modules found in social networks may correspond to different local communities (Girvan and Newman, 2002; Guimera *et al.*, 2003); subgroups in metabolic and cellular networks may reflect distinct functions in biological systems and evolutionary properties of biological molecules and species (Holme *et al.*, 2003; Guimera and Amaral, 2005). Therefore, the modular view of networks provides a clearer understanding on how complex systems are constructed from a number of fundamental components and sheds light into the interactions of such components.

A number of computational approaches have been proposed by various research groups to detect community structures in networks. Traditional methods comprise graph partitioning (Garey and Johnson, 1979) and hierarchical clustering (Fisher, 1996). Graph partitioning deals with the separation of a network into several groups with roughly equal sizes so as to minimise the inter-group communications (Newman, 2004; Boettcher and Percus, 2001). In the area of parallel computing, graph partitioning is applied in order to distribute different tasks to several processors while minimising inter-processor communications. As partitioning a graph is NP-complete (Garey and Johnson, 1979), most heuristic algorithms proposed were bisection-based

where a network was divided into a number of communities by an iterative bisection procedure (Kernighan and Lin, 1970). It should be noted that the optimal solutions to the graph partitioning problem cannot be guaranteed since both the number of communities as well as the sizes of each group are previously fixed by the user.

Hierarchical clustering has also been applied extensively in the investigation of community structures of social and biological systems (Holme *et al.*, 2003; Gustafsson *et al.*, 2006; Rives and Galitski, 2003; Mering *et al.*, 2003). It is an agglomerative procedure transforming a distance matrix of pair-wise similarity measurements between all pairs of nodes into a hierarchical partition tree. Initially, each node forms an independent module and the number of modules is gradually reduced by merging the two most similar clusters iteratively until the whole network is included in one community. Any horizontal cut of the hierarchical tree splits the network into a number of subgroups (see Figure 4.2 for a network of 10 nodes and 3 modules). Although hierarchical clustering does not require any specification of the size or number of modules, it cannot reveal which partition is the best one. Another problem associated with hierarchical clustering lies in its tendency to group only tightly connected nodes in the early stage of clustering because of their strong similarities. However, it cannot always classify nodes with few connectivities correctly since end solution depends on where the agglomerative procedure starts.

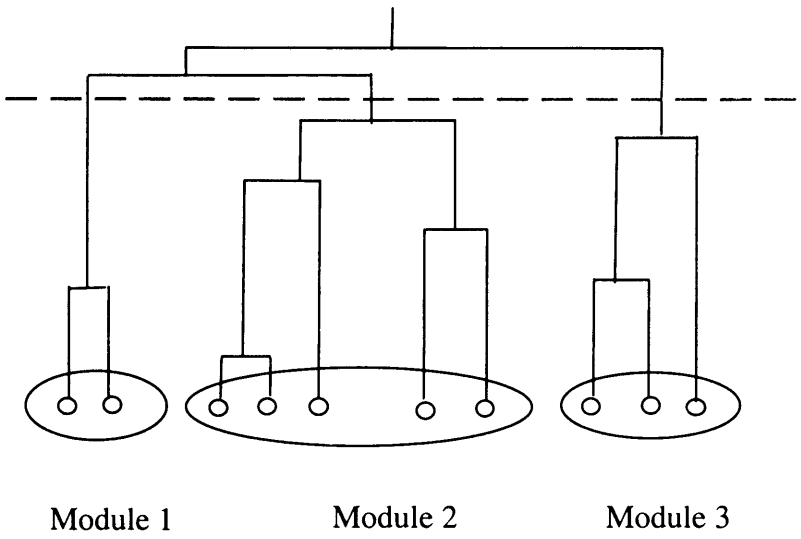


Figure 4.2 A dendrogram of a 10-node network generated by hierarchical clustering

Apart from traditional methodologies proposed above, a number of local algorithms and physical models were applied to detect community structures. First, a set of self-contained local algorithms to detect network communities were proposed (Castellano *et al.*, 2004; Radicchi *et al.*, 2004). The algorithms kept the same level of liability and outperformed other existing approaches with respect to computational costs. Networks were also treated as electric circuits and communities were identified based on notions of voltage drops across networks (Wu and Huberman, 2004). Furthermore, an algorithm based on a modified q-state Potts model was presented (Reichardt and Bornholdt, 2004). Communities are considered as domains with equal spin values near the ground state of the system, which was approximated using Monte Carlo optimisation. Finally, Son *et al.* (2006) developed a random field Ising model to determine the community structure. The ground state problem is equivalent to the maximum flow problems, which can be solved using combinatorial optimisation algorithms.

In more systematic investigations of network properties, the *Modularity* metric (Newman and Girvan, 2004) was introduced as a measure of network partition quality. Network modularity is the fraction of all edges that lie within communities minus the expected value of the same quantity in a graph in which the vertices have the same degrees but edges are placed at random. A modularity value of 0 indicates that the network considered is equivalent to random networks and no obvious community structures are observed; modularity approaching the maximum value of 1, indicates a strong community structure.

Newman and Girvan (2004) developed a series of divisive algorithms to discover community structures, involving the iterative removal of edges with the highest "betweeness" score to split the network into communities. These algorithms were highly effective at discovering community structures for many testing cases at the cost of very high computational resources when analysing large-scale networks. More computationally efficient algorithms were proposed to tackle networks with larger sizes (Clauset *et al.*, 2004; Newman, 2004). Newman proved that network modularity can be rewritten as eigenvectors of a modularity matrix and this expression leads to a spectral algorithm for community detection resulting in higher quality solutions when compared to competing approaches (Newman, 2006).

Since proposing the concept of modularity, the community structure detection problem can be posed as an optimisation task which finds an optimal partition at the maximum value for modularity. Simulated annealing (SA) was first used to identify functional modules in metabolic networks of twelve organisms from three different superkindoms by maximising their modularity values (Guimera and Amaral, 2005). The same optimisation methodology was also applied to analyse and benchmark social networks (Medus *et al.*, 2005) where a trade-off between quality of solutions and computational requirements was noted. Moreover, the applicability of extremal optimisation was demonstrated through a number of test cases of computer-simulated and real networks (Duch and Arenas, 2005).

Recently, Fortunato and Barthelemy (2007) reported the observation that the optimisation of modularity metric has a resolution limit, so submodules smaller than a certain scale in large networks may fail to be detected since the modularity optimisation procedure tends to combine small communities into larger ones. Kumpula *et al.* (2007) showed that the q-state Potts model introduced by Reichardt and Bornholdt (2004) also has a resolution threshold. Both findings raised major concerns of the reliability of modularity optimisation. However, Arenas *et al.* (2007) overcome such problems by proposing a systematic method to discover community structures at different resolution levels using the original modularity concept.

Although the presence of resolution limit of modularity maximisation makes some small modules in large networks invisible, modularity is still one of the most widely accepted metrics to detect community structures. All approaches mentioned above are able to achieve good quality modularity values when partitioning networks of various sizes. However, a major limitation is that global optimality of the solutions cannot be guaranteed. Here, a general mathematical programming formulation for the network community structure identification problem is presented where the objective function considered is maximisation of the modularity value and can be solved to global optimality. More importantly, the proposed optimisation model can easily be extended in the future to detect communities more accurately when alternative measures become available. Other additional features such as minimum/maximum module size and balancing among modules can also be incorporated using mathematical programming to aid accurate detection.

This chapter is structured as follows: the problem statement for network community detection is defined in the next section. Section 4.3 presents an MIQP model to detect community structures of a network with the maximum modularity value. Symmetry breaking constraints are then proposed to avoid redundant equivalent solutions thus reducing the computational requirements significantly. The applicability of the proposed mathematical model is demonstrated in section 4.4 through the use of four network examples and comparisons of the present methodology with other literature approaches. Finally, some concluding remarks are made in section 4.5.

4.2 Problem Statement

Networks are defined by a set of nodes and links connecting them. Each link is undirected and unweighted. Overall, the problem of network community structure identification can be stated as follows:

Given:

- An undirected network consisting of N nodes and L links

Determine:

- Optimal number of modules
- Node-module allocation

So as to:

Maximise the network modularity metric

4.3 Mathematical Formulation

The indices, sets and parameters associated with the mathematical model are listed below:

Indices

n, e	Nodes
l	Links
m, k	Modules

Parameters

N	Total number of nodes
L	Total number of links
M	Total number of modules
d_n	Degree of node n
α	Minimum module size
β	Maximum module size
ε	Maximum size difference between any pair of modules

Sets

S	M most connected nodes
AM_n	Allowed modules for assignment to node $n \in S$
ML_l	Allowable modules for link l
Av_m	Nodes allowed assignment to module m
B_n	Nodes with higher connectivity than node n

The mathematical formulation is based on the following optimisation variables:

Binary variables

E_m	1 if module m exists; 0 otherwise
X_{lm}	1 if link l belongs to module m; 0, otherwise

Y_{nm} 1 if node n belongs to module m; 0, otherwise

Positive continuous variables

L_m Number of links among nodes within module m

D_m Degree of module m

4.3.1 Objective Function

The objective function considered here is the maximisation of the network *modularity* metric as proposed by Newman and Girvan (2004):

$$Q = \sum_m \left[\frac{L_m}{L} - \left(\frac{D_m}{2L} \right)^2 \right] \tag{4.1}$$

4.3.2 Allocation Constraints

Each node should be allocated to exactly one module:

$$\sum_m Y_{nm} = 1 \qquad \forall n \tag{4.2}$$

Link l belongs to module m if both nodes associated with l (*i.e.* nodes n and e) are allocated to module m. This logical condition can be written mathematically as:

$$2X_{lm} \le Y_{nm} + Y_{em} \qquad \forall m, l = \{n, e\} \tag{4.3}$$

Constraint (4.3) can alternatively be disaggregated into two tighter inequalities:

$$X_{lm} \le Y_{nm} \qquad \forall m, l = \{n, e\} \tag{4.4}$$

$$X_{lm} \le Y_{em} \qquad \forall m, l = \{n, e\} \tag{4.5}$$

4.3.3 Definition of L_m and D_m

L_m is defined as the total number of links within module m:

$$L_m = \sum_l X_{lm} \qquad \forall m \tag{4.6}$$

D_m is defined similarly to be equal to the sum of the degrees of nodes allocated to module m:

$$D_m = \sum_n d_n \cdot Y_{nm} \qquad \forall m \qquad (4.7)$$

4.3.4 Additional Constraints

One of the key advantages of using mathematical programming approaches is the ease of accommodating user-defined conditions. Here, a number of additional features are formulated mathematically.

First, we describe how minimum and/or maximum module sizes can be incorporated. A binary variable, E_m, is introduced to determine the existence or not of module m. A degeneracy constraint is proposed to enforce that module m is allowed only when the previous module exists (*i.e.* $E_{m-1} = 1$):

$$E_m \leq E_{m-1} \qquad \forall m = 2,\ldots,M \qquad (4.8)$$

Note that if module *m-1* does not exist (*i.e.* $E_{m-1} = 0$), then module m, does not exist as well (*i.e.* $E_m = 0$). Module m is not empty when the following two constraints are active at the same time:

$$\sum_l X_{lm} \geq \alpha \qquad \forall m \qquad (4.9)$$

$$\sum_l X_{lm} \leq \beta \qquad \forall m \qquad (4.10)$$

The above constraints (4.9) and (4.10) should be activated only if module m exists and therefore, they should be rewritten as:

$$\sum_l X_{lm} \geq \alpha \cdot E_m \qquad \forall m \qquad (4.11)$$

$$\sum_l X_{lm} \leq \beta \cdot E_m \qquad \forall m \qquad (4.12)$$

It is worth mentioning that the above constraints (4.8, 4.11 and 4.12) safeguard that all occupied modules are first ranked to avoid equivalent solutions and then module sizes within prespecified bounds are enforced.

Next, we demonstrate how balancing issues among modules, if required, can easily be accommodated in the current optimisation approach. By balancing, we denote that any two non-empty modules m and k, (*i.e.* $E_m = E_k = 1$) cannot differ by more than a user-defined number of links, ε:

$$|L_m - L_k| \leq \varepsilon \qquad \forall m, k > m \tag{4.13}$$

The above absolute-value inequality can mathematically be written as:

$$L_m - L_k \leq \varepsilon \qquad \forall m, k > m \tag{4.14}$$

$$L_k - L_m \leq \varepsilon \qquad \forall m, k > m \tag{4.15}$$

It should be added that the above constraints are activated only if both modules m and k are selected (*i.e.* $E_m = E_k = 1$). Thus, constraints (4.14) and (4.15) can be rewritten as:

$$L_m - L_k \leq \varepsilon + \beta(2 - E_m - E_k) \qquad \forall m, k > m \tag{4.16}$$

$$L_m - L_k \leq \varepsilon + \beta(2 - E_m - E_k) \qquad \forall m, k > m \tag{4.17}$$

The degeneracy constraint (4.8) indicates that the value of E_m can be forced to 1 when module k is non-empty (*i.e.* $E_k = 1$), so constraints (4.16) and (4.17) can be simplified as:

$$L_m - L_k \leq \varepsilon + \beta(1 - E_k) \qquad \forall m, k > m \tag{4.18}$$

$$L_m - L_k \leq \varepsilon + \beta(1 - E_k) \qquad \forall m, k > m \tag{4.19}$$

Overall, the resulting mathematical model (ModMax) for determining community structures based on the modularity metric for network community identification is formulated as follows:

[ModMax]:

$$\text{Maximise } Q = \sum_m \left[\frac{L_m}{L} - \left(\frac{D_m}{2L} \right)^2 \right]$$

subject to

constraints (4.2, 4.4-4.8, 4.11-4.12, 4.18-4.19).

$$E_m, X_{lm}, Y_{nm} \in \{0,1\};\ L_m, D_m \geq 0$$

4.3.5 Symmetry-Breaking Constraints

It is widely believed that when a set of objects is clustered into a number of modules, any renumbering of the modules generates an equivalent solution (Klein and Aronson, 1991). Specifically, if a network ends up with M optimal communities, there are $M!$ equivalent solutions. Table 4.1 enumerates equivalent solutions for a network example with 8 nodes and 3 modules. Here, two symmetry breaking constraints are proposed to eliminate equivalent solutions and thus to reduce the number of nodes explored during a branch-and bound solution procedure.

Table 4.1 Equivalent solutions for a three-module problem

	Module 1	Module 2	Module 3
Solution 1	n_1, n_2	n_3, n_7, n_8	n_4, n_5, n_6
Solution 2	n_1, n_2	n_4, n_5, n_6	n_3, n_7, n_8
Solution 3	n_4, n_5, n_6	n_1, n_2	n_3, n_7, n_8
Solution 4	n_4, n_5, n_6	n_3, n_7, n_8	n_1, n_2
Solution 5	n_3, n_7, n_8	n_1, n_2	n_4, n_5, n_6
Solution 6	n_3, n_7, n_8	n_4, n_5, n_6	n_1, n_2

Suppose we seek to partition all nodes into M modules. In order to avoid equivalent solutions through the renumbering of modules, each node is allowed to be allocated to one of a particular set of modules, AM_n. First, all nodes are sorted based on their connectivities. For the example shown in Table 4.1, let us assume that n_1 is the most

connected node, n_2 is the second most connected node and so on. The AM_n set is then constructed as: n_1 is allocated to module 1 only; n_2 can be assigned to either module 1 or 2. All other nodes can be allocated to any of the three available modules. Therefore, constraint (4.2) can be rewritten as the following equality:

$$\sum_{m \in AM_n} Y_{nm} = 1 \qquad \forall n \tag{4.20}$$

By activating constraint (4.20), solutions 3 to 6 in Table 4.1 can be eliminated as node n_1 is allocated to module 1. It should be mentioned that similar constraints as in (4.20) have also been reported by Klein and Aronson (1991) in the case of cluster analysis.

Since each node n has its own allowable set of modules (AM_n), link l that connects nodes n and e can be allocated to the modules that appear in both AM_n and AM_e. Here, we define set ML_l (allowable modules for link l) as $AM_n \cap AM_e$, where $l = \{n, e\}$. Consequently, constraints (4.4) and (4.5) can be replaced by:

$$X_{lm} \le Y_{nm} \qquad \forall l = \{n, e\}, m \in ML_l \tag{4.21}$$

$$X_{lm} \le Y_{em} \qquad \forall l = \{n, e\}, m \in ML_l \tag{4.22}$$

$$X_{lm} = 0 \qquad \forall l, m \notin ML_l \tag{4.23}$$

Another logical condition can be imposed by not allowing node n to be allocated to module m (assuming $m \in AM_n$) if all previous nodes e ($e \in B_n \cap Av_{m-1}$) have not been assigned to module m-1 (*i.e.* $\sum_{e \in (B_n \cap Av_{m-1})} Y_{e,m-1} = 0$, then Y_{nm}=0). Note that B_n denotes the set of nodes e with larger number of connections than n and Av_m denotes the set of nodes that can be assigned to module m. Considering the example shown in Table 4.1, we then have: $B_{n_1} = \phi$, $B_{n_2} = \{n_1\}$, $B_{n_3} = \{n_1, n_2\}$; $Av_1 = \{n_1, n_2, ..., n_8\}$, $Av_2 = \{n_2, n_3, ..., n_8\}$, $Av_3 = \{n_3, n_4, ..., n_8\}$ and so on. As a consequence, if nodes n_1 and n_2 are allocated to module 1, then node n_3 should not be placed to module 3; if nodes n_2 and n_3 are allocated to module 2, then node n_4 should not be assigned to module 4; if nodes n_1, n_2 and n_3 are assigned to module 1, then node n_4 is also excluded from module 3 etc.

Based on the above description, the following logical constraint is proposed:

$$Y_{nm} \leq \sum_{e \in (B_n \cap Av_{m-1})} Y_{e\,m-1} \qquad \forall n \geq 3, m = 3..., |AM_n| \tag{4.24}$$

When applying the above constraint to the example shown in Table 4.1, we do not allow node n_3 to be assigned to module 3 as node n_2 appears in module 1 together with node n_1. Thus, solution 2 is eliminated and only solution 1 is feasible.

Symmetry breaking constraints (4.20) and (4.24) avoid all other *M!*-1 equivalent solutions. From our experience, significant computational enhancements are also achieved by only considering the M most connected nodes (defined as set S). Both symmetry breaking constraints are active for all nodes in S:

$$\sum_{m \in AM_n} Y_{nm} = 1 \qquad \forall n \in S \tag{4.25}$$

$$\sum_{m} Y_{nm} = 1 \qquad \forall n \notin S \tag{4.26}$$

$$Y_{nm} \leq \sum_{e \in (B_n \cap Av_{m-1})} Y_{e\,m-1} \qquad \forall n \in S, n \geq 3, m = 3..., |AM_n| \tag{4.27}$$

Overall, the resulting mathematical model (OptMod) for determining community structures based on the modularity metric incorporating the above symmetry breaking constraints for network community identification is formulated as follows:

[OptMod]:

$$\text{Maximise } Q = \sum_{m} \left[\frac{L_m}{L} - \left(\frac{D_m}{2L} \right)^2 \right]$$

subject to

constraints (4.6-4.8, 4.11-4.12, 4.18-4.19, 4.21-4.23, 4.25-4.27).

$$E_m, X_{lm}, Y_{nm} \in \{0,1\};\ L_m, D_m \geq 0$$

The resulting mathematical formulation is a mixed integer quadratic programming (MIQP) model comprising a concave quadratic objective function which is maximised with a set of linear constraints and mixed binary/continuous optimisation variables.

The CPLEX mixed integer optimisation solver (Ilog, 2006) is used to solve the proposed model to global optimality, due to its convexity, through the branch-and-bound procedure (see, for example, Floudas, 1995).

4.4 Computational Results

The proposed mathematical models ModMax and OptMod are applied to four network examples from different research areas. All examples are implemented in GAMS (General Algebraic Modelling System) (Brooke, *et al.* 2003) using the CPLEX mixed integer optimisation solver with 0% margin of optimality and 10000 seconds CPU limit. First, ModMax and OptMod are performed using all selected example networks to demonstrate the efficiency of symmetry breaking constraints. The computational statistics shown in Table 4.2 clear indicate that symmetry breaking constraints save computational resources significantly. Second, the computational statistics and the optimal modularity values obtained by the proposed MIQP OptMod are reported in Table 2. The optimal modularity is then compared with other literature approaches for community structure identification (see Table 4.3). As an alternative, the computational requirements and the best modularity value for each partition from hierarchical clustering are reported. The hierarchical clustering runs are performed by the cluster package using the statistical computing language *R* (www.r-project.org). The community structures for all networks are displayed through the Pajek network analysis program (http://vlado.fmf.uni-lj.si/pub/networks/pajek/). In each figure, dotted lines are used to reveal the modules obtained.

The first example considers a social network compiled by Zachary (1977), who spent two years in observing the social communications between members in a karate club at an American University. Nodes in the network stand for club members and the links reflect the social relations between them (see Figure 4.3). According to the literature (Zachary, 1977), the club naturally split in two smaller communities because of a dispute between the club's administrator (around node 1) and the karate teacher (around node 34). This actual division is visualised in and where squares and circles denote the members of each community. Our approach shows that the optimal partition is found at a modularity value of 0.4198 when splitting the network into four independent modules (see Figure 4.3 for the optimal partition). It can been seen clearly that the optimal partition from the proposed model perfectly reflects the real

community structure (Nodes of modules I and II stand for members around the administrator and modules III and IV belong to the teacher's group). Similar results were produced by hierarchical clustering (Gustafsson *et al.*, 2006), greedy optimisation algorithm (Clauset *et al.*, 2004), simulated annealing (Medus *et al.*, 2005), extremal optimisation (Duch and Arenas, 2005) and the betweenness-based iterative algorithms (Newman and Girvan, 2004) (see Table 4.3). It is noted that hierarchical clustering identifies the same community structures as the optimal partition. The betweenness-based algorithm (Newman and Girvan, 2004) finds five modules with modularity of 0.3724 (see Figure 4.4). Node 10, which is considered as an independent module through that algorithm, should be allocated together with node 34. The betweenness-based algorithm also resulted in one misclassification (node 3) when compared with the actual community structure according to observations by Zachary (1977).

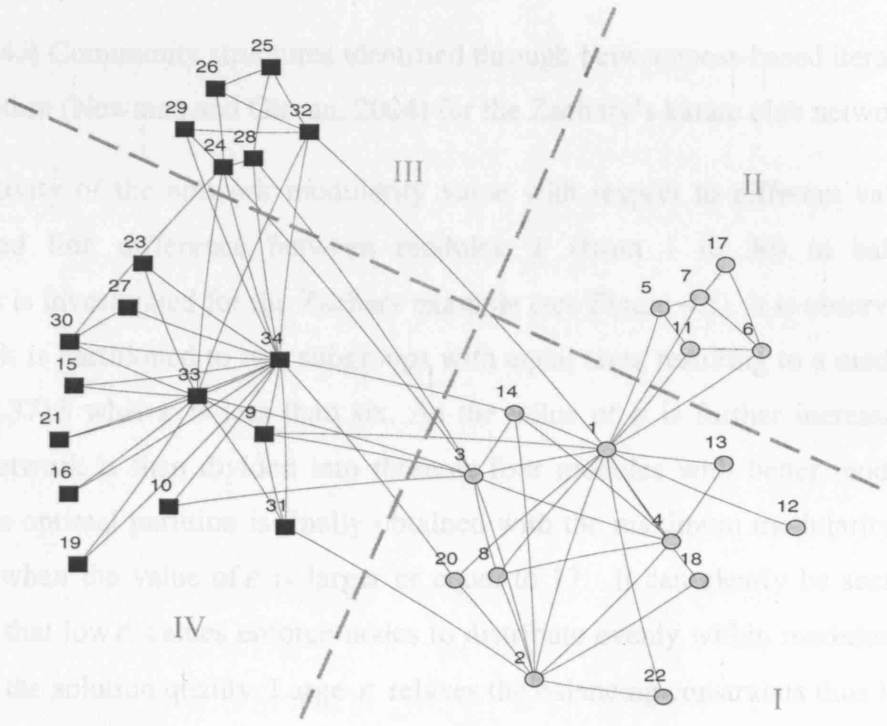


Figure 4.3 Optimal community structure for the Zachary's karate club network using OptMod

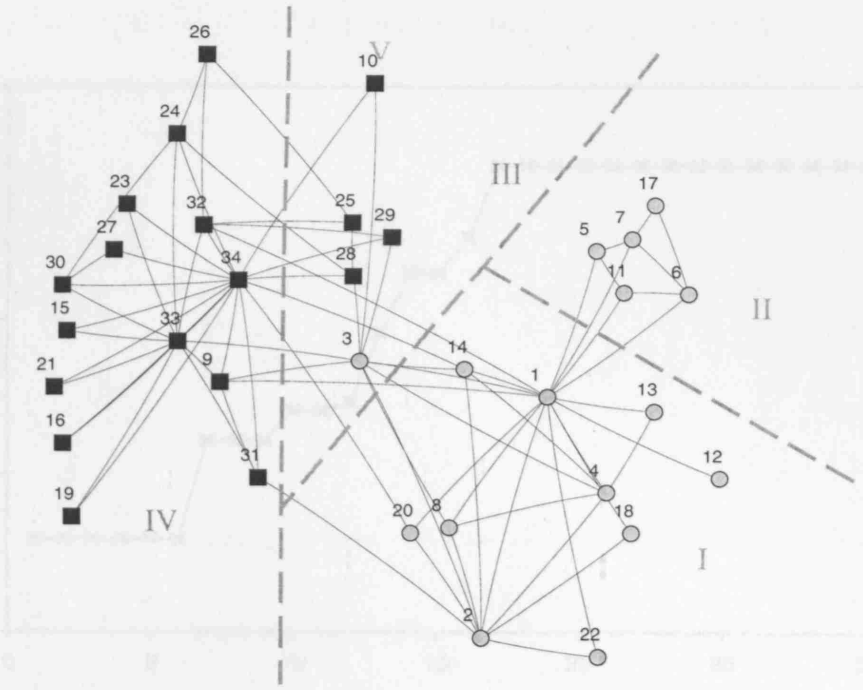


Figure 4.4 Community structures identified through betweenness-based iterative algorithm (Newman and Girvan, 2004) for the Zachary's karate club network

The sensitivity of the network modularity value with respect to different values of user-defined link difference between modules, ε (from 1 to 30) in balancing constraints is investigated for the Zachary example (see Figure 4.5). It is observed that the network is partitioned to two subgroups with equal sizes resulting to a modularity value of 0.3718 when ε is less than six. As the value of ε is further increased, the Zachary network is then divided into three or four modules with better modularity values. The optimal partition is finally obtained with the maximum modularity value of 0.4198 when the value of ε is larger or equal to 17. It can clearly be seen from Figure 4.5 that low ε values enforce nodes to distribute evenly within modules while sacrificing the solution quality. Large ε relaxes the balancing constraints thus leading to the optimal partition achieved by the proposed MIQP model. Consequently, user criteria can prioritise the prevalence of either module size balancing or network partitioning optimality. We note that only this example has been analysed with balancing constraints in order to showcase their use. It is obvious that this type of constraint can be used at will in any other examples as required by the user.

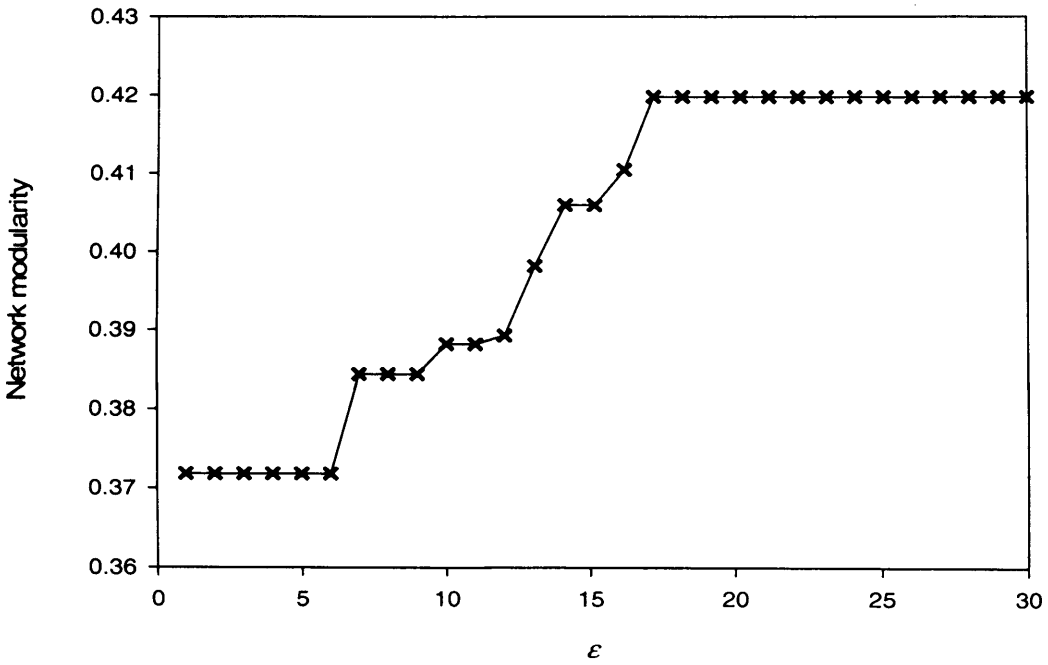


Figure 4.5 Sensitivity of optimal modularity values with parameter ε

In the second example, we present a community of 62 bottlenose dolphins living in Doubtful Sound, New Zealand, constructed by Lusseau (2003) after seven years of field studies. Each node represents a dolphin and the links in the network are identified based on the significantly frequent communications among them. Using this network as input to the proposed MIQP model, five communities are found. Module I and modules II-V reflect the real division observed by Lusseau (2003) with zero misclassification and the MIQP model indicates the existence of four smaller communities in the second group (see Figure 4.6 for the optimal division by our approach; squares and circles denote the actual partition reported by Lusseau). Comparing with the division from hierarchical clustering (see Figures 4.6 and 4.7), the optimal community structure merge groups I and II of Figure 4.6, while partition module IV from hierarchical clustering into two groups (see modules II and IV of Figure 4.6). Hierarchical clustering also results in two misclassifications (nodes 8 and 20) when compared with the real partition. It can be seen from Table 4.3 that all methods partition the dolphin network into five communities. Hierarchical clustering and the betweenness-based iterative algorithms algorithm achieved a modularity value of 0.5084 and 0.5200, respectively. Our approach results in a maximum value of

0.5285, which is 3.80% and 1.61% more efficient than the other two literature approaches.

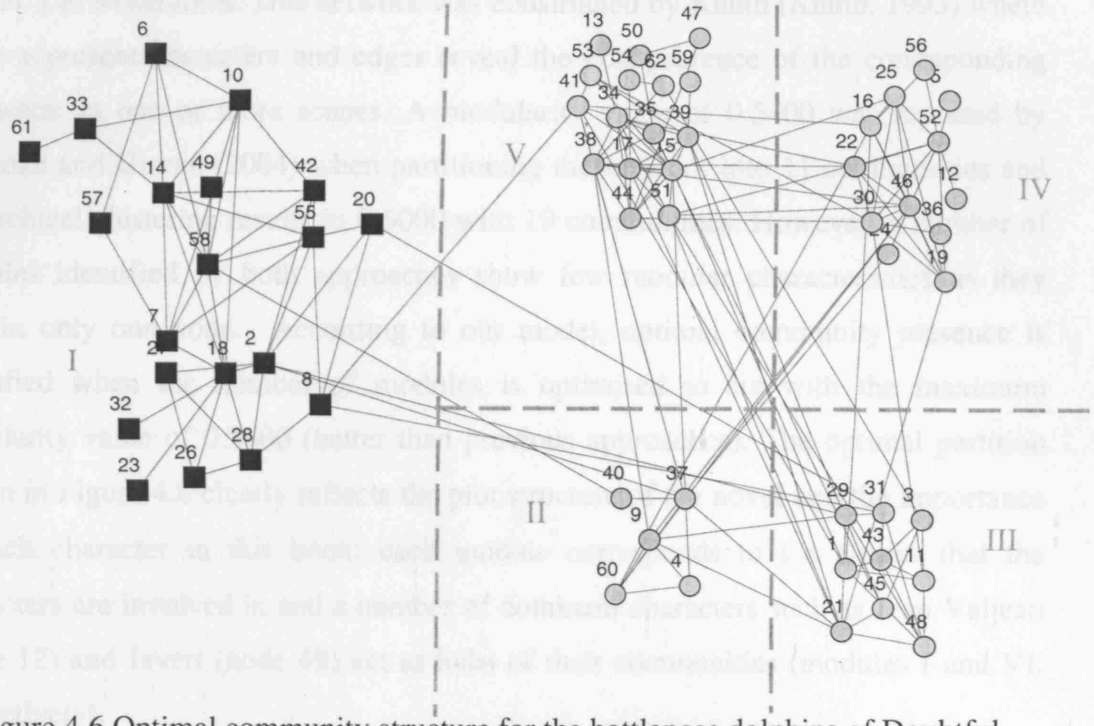


Figure 4.6 Optimal community structure for the bottlenose dolphins of Doubtful Sound using OptMod

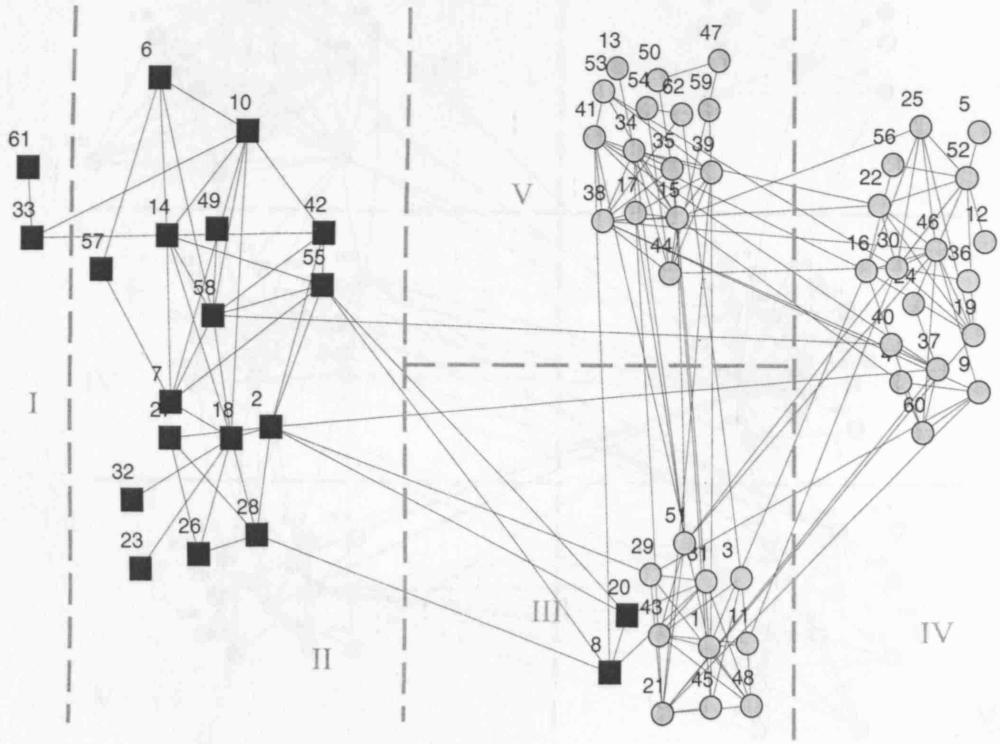


Figure 4.7 Community structures identified through hierarchical clustering for dolphins of Doubtful Sound

The third example considered here is the network showing the connections between major characters in Victor Hugo's novel of crime and redemption in post-restoration France, Les Miserables. This network was constructed by Knuth (Knuth, 1993) where nodes represent characters and edges reveal the coappearence of the corresponding characters in one or more scenes. A modularity value of 0.5400 was reported by Newman and Girvan (2004) when partitioning the network into 11 communities and hierarchical clustering results in 0.5000 with 19 communities. However, a number of modules identified by both approaches show few modular characteristics as they contain only one node. According to our model, optimal community presence is identified when the number of modules is optimised to six with the maximum modularity value of 0.5600 (better than previous approaches). The optimal partition shown in Figure 4.8 clearly reflects the plot structure of the novel and the importance of each character in this book: each module corresponds to the stories that the characters are involved in and a number of dominant characters such as Jean Valjean (node 12) and Javert (node 49) act as hubs of their communities (modules I and VI, respectively).

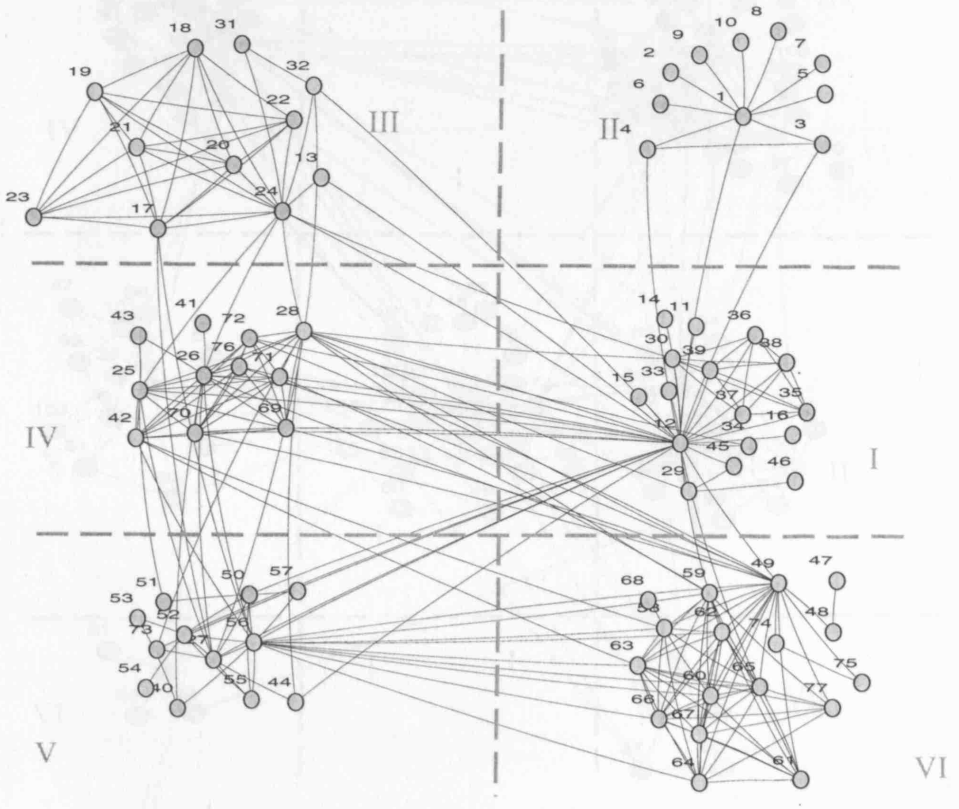


Figure 4.8 Optimal community structures for the Les Miserables network

In the last example, we apply our model to the p53 protein-protein interaction network constructed by Dartnell *et al.* (2005). A list of proteins in this network is included in the appendix. This network involves an annotated protein interaction map in mammalian cell cycle, DNA repair and apoptosis. As a key element in maintaining genomic stability, protein p53 lies in the centre of the network and controls the intra- and intercellular signals with gene transcription. The p53 network consisting of 104 proteins and 226 interactions has been proven to have a scale-free topology (Dartnell *et al.*, 2005) showing that a vast majority of the nodes are poorly connected while few of them act as hubs with a high centrality. Hierarchical clustering found nine modules with modularity of 0.4580. The maximum modularity value (0.5351) is again reported by our model partitioning the p53 network into seven communities (see Figure 4.9). It is not surprising that module I lies in the centre of the network and communicates with all other six modules. Node 68 (protein p53), the most central protein to the network, is included in this module.

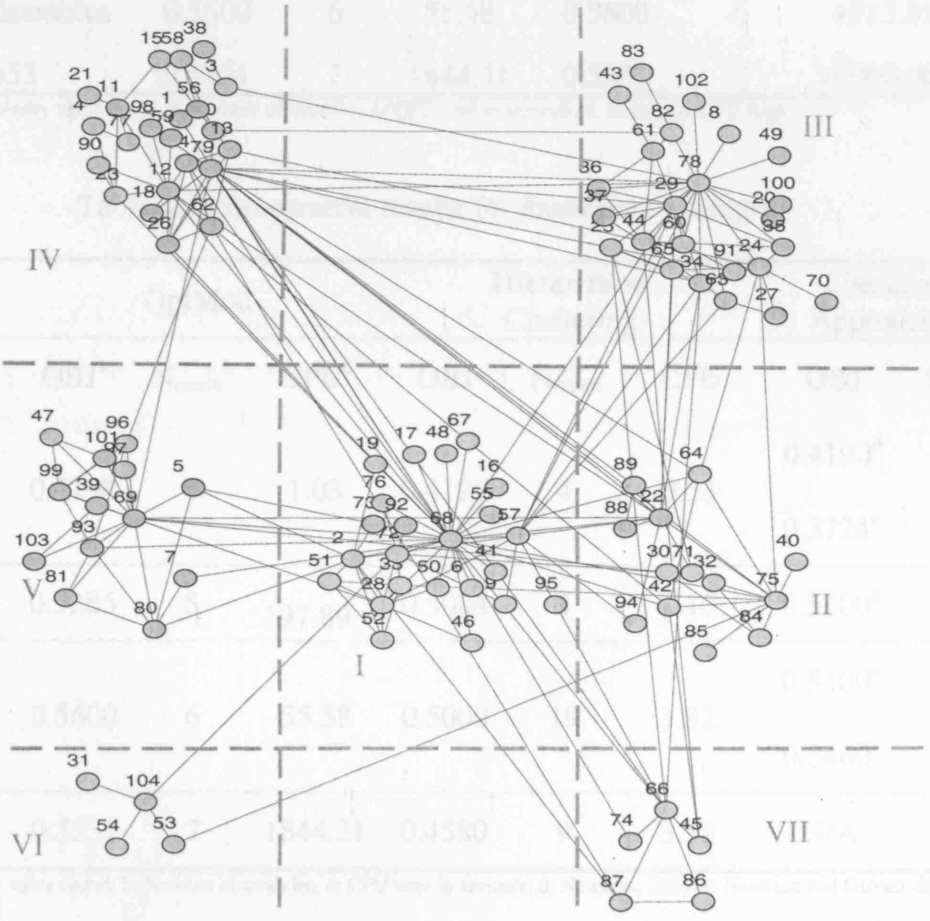


Figure 4.9 Optimal community structure for the p53 protein-protein interaction network

Finally, the power of symmetry-breaking constraints has been revealed in Table 4.2 by comparing the computational resources used for models ModMax and OptMod. As mentioned in section 4.3.5, symmetry-breaking constraints are used to eliminate equivalent solutions of the proposed MIQP model ModMax. It can clearly be seen from Table 4.2 that the MIQP model with symmetry-breaking constraints (model OptMod) is able to achieve the global optimal solution using significantly less computational efforts.

Table 4.2 Computational results of ModMax and OptMod for all illustrative examples

	OptMod			ModMax		
Examples	OBJ[a]	N_{modu}[b]	CPU[c]	OBJ	N_{modu}	CPU
Zachary	0.4198	4	1.03	0.4198	4	4.77
Dolphin	0.5285	5	197.89	0.5285	5	4964.80
Les Miserables	0.5600	6	55.58	0.5600	6	4973.81
p53	0.5351	7	1844.31	0.5351	7	10000.00[d]

a: Best modularity value found; b: Number of modules; c: CPU time in seconds; d: Maximum CPU limit

Table 4.3 Comparative results for illustrative examples

	OptMod			Hierarchical Clustering			Literature Approaches	
Examples	OBJ[a]	N_{modu}[b]	CPU[c]	OBJ	N_{modu}	CPU	OBJ	N_{modu}
Zachary	0.4198	4	1.03	0.4198	4	0.33	0.4190[d]	4
							0.3724[e]	5
Dolphin	0.5285	5	197.89	0.5084	5	1.10	0.5200[e]	5
Les Miserables	0.5600	6	55.58	0.5000	19	1.82	0.5400[e]	11
							0.5460[f]	5
p53	0.5351	7	1844.31	0.4580	9	3.58	N/A	N/A

a: Best modularity value found; b: Number of modules; c: CPU time in seconds; d: Newman, 2006; e: Newman and Girvan, 2004; f: Medus *et al.*, 2005.

4.5 Concluding Remarks

Many social, technical and biological systems can be represented as networks of interacting components. Community structures are usually found in those systems where nodes are naturally divided into subgroups with dense within-module connections. Detection of such structures can be vitally beneficial to the study of various complex systems since nodes within the same module may share similar functional properties and novel patterns or functions can be deducted through the analysis of the interacting modules.

In this chapter, a rigorous MIQP model has been proposed to identify optimal community structure in complex networks. The objective function considered is maximisation of the network modularity proposed previously (Newman and Girvan, 2004). Symmetry breaking constraints have been introduced to avoid the generation of equivalent solutions thus enhancing the computational performance of the proposed model. Our results have shown that global optimal solutions have been achieved for all examples studied. More importantly, the power of mathematical programming is demonstrated by easily incorporating other additional features such as minimum/maximum module size and balancing among modules in the proposed optimisation model. It should be noted that the proposed MIQP model can successfully detect the optimal community structure of small size networks. In this next chapter, a two-stage solution approach is developed to identify accurate modular structures of medium/large networks within reasonable computational times.

Chapter 5

A Two-stage Solution Approach for Network Community Identification using Mathematical Programming

Finding network modules or communities is widely accepted as a major avenue to reveal the underlying properties of complex systems. Since the adoption of the *Modularity* metric to express the modular structure of a network (Newman and Girvan, 2004), a number of computational methodologies for community structure discovery have been developed using modularity maximisation. In Chapter 4, an MIQP model has been developed to identify network communities of small/medium networks. The proposed mathematical model is able to achieve optimal community structures of networks with small/medium sizes. Due to the NP-hard nature of modularity maximisation, finding communities with medium/large networks is computationally

demanding. In this chapter, we present a novel two-stage solution approach to identify network communities using mixed integer optimisation. Computational results show that the proposed algorithm outperforms previous attempts from the literature, being able to tackle large-scale network implementations. Furthermore, in order to overcome the resolution limit of modularity maximisation, a solution procedure is developed to find finer modular structures with high resolution.

5.1 Introduction and Literature Survey

In Chapter 4, the problem of network community structure identification has clearly stated. A simultaneous mixed integer quadratic programming (MIQP) model OptMod has been develop to find the optimal community structure of complex networks based on modularity maximisation. The proposed mathematical programming framework is able to achieve global optimal solutions of small size networks due to the convexity of the model. As modularity optimisation is NP-hard, efficient algorithms to find the maximum modularity values are unlikely to exist. Therefore, most approaches employ heuristics that aim at finding near-optimal solutions within modest computational expense.

The use of mathematical programming for modularity optimisation has been recently reported (Agarwal and Kempe, 2007): an algorithm based on the rounding of a fractional linear programming model and a hierarchical partition through the repeated rounding of a vector programming model to solution were shown to be effective in small to medium size networks. An efficient hierarchical agglomeration algorithm was developed for detecting community structures (Clauset *et al.*, 2004). In this agglomerative procedure, two nodes combine together to form a new cluster if this merge results a maximum modularity increase. This algorithm was later improved by revising the initial hierarchical agglomeration process as random walks so that the computational requirements can be significantly reduced (Pujol *et al.*, 2006). Finally, a near-linear time algorithm has been proposed (Raghavan *et al.*, 2007) where every node is initialised with a unique module label and each node adopts the label that most of its neighbours currently have. After this iterative process, densely connected groups of nodes form a consensus on a unique label to form communities. The algorithm has

been demonstrated to take an almost linear time scale and hence it is computationally less expensive than most current methodologies.

Based on current research development on network community structure identification, crucial considerations in assessing the performance of modularity optimisation approaches are: (i) the scale and optimality handled by modularity optimisation methods and (ii) the resolution limit problem for small-size modules in large networks. These points are discussed further in the next paragraphs and form the main focus of this chapter.

First, there seems to be a trade-off between network size and optimality achieved through modularity optimisation, with methods that guarantee global optimal solutions for modularity maximisation able to operate only in small to medium-sized networks (Xu *et al.*, 2007). For example, divisive algorithms (Newman and Girvan, 2004) and mathematical programming (Agarwal and Kempe, 2007) were found to be prohibitively computationally expensive for large networks. Other methods, such as stochastic optimisation through simulated annealing (Guimera and Amaral, 2005; Medus *et al.*, 2005) and extremal optimisation (Duch and Arenas, 2005) that could be used on large networks, may yield sub-optimal solutions and so may suffer poor performance. In our own work, we have previously reported a rigorous mixed integer quadratic programming (MIQP) formulation to optimise the modularity metric with a set of linear constraints and mixed binary/continuous optimisation variables (Xu *et al.*, 2007). Due to the convexity properties of the model, global optimal solutions are achieved through the standard branch-and-bound procedure with commercial optimisation solvers, but its use is limited to small-medium scale networks due to NP-hardness.

Second, doubts have been raised over the use of modularity optimisation for community detection recently due to the observation that such procedures can reach a resolution limit (Fortunato and Barthelemy, 2007). This effect essentially entails modules smaller that a specific scale not being detected, as the optimisation process combines smaller communities into larger ones so as to achieve better modularity. Some solutions have been suggested through either re-optimising each module without considering the inter-community links (Fortunato and Barthelemy, 2007) or tuning a resolution parameter (Arenas *et al.*, 2007).

Here, we aim to enhance the efficient application of mathematical programming to community structure identification by: (i) extending our previous mathematical programming methodology proposed in Chapter 4 to tackle larger networks and (ii) incorporating methodologies for dealing systematically with the problem of resolution limit through modularity optimisation. In overview, a two-stage solution approach for community identification using mathematical programming is described in the next section, the resolution limit problem of modularity optimisation is addressed in section 5.3 through the introduction of a solution procedure to produce network communities with high resolution and the applicability of the proposed approaches is demonstrated through a number of network examples in section 5.4. Finally, some concluding remarks are made in section 5.5.

5.2 Module Identification via Two-Stage Mathematical Programming

The solution approach proposed in this paper is a two-stage, iterative modularity optimisation procedure, which we call iMod. First, a mixed integer nonlinear programming (MINLP) model (named MINLP_Mod) is formulated to obtain a feasible solution efficiently. An initial partition with a good quality modularity value is selected from a set of MINLP solutions with random starting points. Second, the solution obtained in the first stage is improved through an iterative optimisation procedure employing the model previously developed in Chapter 4, which was shown to be very efficient for community detection in small to medium size networks through a global maximum of the modularity metric (Xu *et al.*, 2007). The overall approach here combining the two aforementioned stages is intended to extend the use of mathematical programming methodologies for larger size networks. The sets and parameters in this approach are defined below.

Indices

n,e	Node
m, m'	Module
k	Major iteration

Sets

I_m	Set of nodes in module m
RE_m	Set of remaining nodes to be released
CN_n	Set of nodes connecting node n
Δ	Set of nodes released

Parameters

N	Total number of nodes
L	Total number of links
M	Total number of modules
IM	Number of runs of the MINLP_Mod in the first stage
N^{max}	Maximum number of runs for MINLP_Mod
N_R	Number of released nodes
MAX^r_m	Maximum number of released nodes for module m

Binary Variables

Y_{nm}	1 if node n is allocated to module m; 0, otherwise
$Y^{\max}_{nm}$	The node-module allocation with the maximum modularity value from the multiple runs of MINLP_Mod

Positive Continuous Variables

L_m	Number of links among nodes within module m
D_m	Degree of module m
Q_k	Network modularity for major iteration k.
Q^{max}	Best modularity value obtained from stage 1

Each stage of the proposed procedure is described in the next sections in more detail.

5.2.1 Stage 1 – Initial Network Partition

Given a network with N nodes and L edges, the modularity metric, Q, of a partition the network into M communities is represented as:

$$Q = \sum_{m} \left[\frac{L_m}{L} - \left(\frac{D_m}{2L} \right)^2 \right] \tag{5.1}$$

where L_m denotes the number of links in module m and D_m is the degree of all nodes in module m. The modularity metric, Q, measures the difference of the fraction of links within communities and the expected fraction values when links are allocated randomly (Newman and Girvan, 2004). The objective function employed here is the maximisation of the network modularity metric shown in equation (5.1). First, each node is allocated to exactly one module:

$$\sum_{m} Y_{nm} = 1 \qquad \forall n \tag{5.2}$$

As previously defined, D_m is equal to the sum of the degrees of nodes allocated to module m:

$$D_m = \sum_{n} d_n \cdot Y_{nm} \qquad \forall m \tag{5.3}$$

A link will be allocated to module m only when both nodes connected by it are in module m. Therefore, the total number of links in module m, L_m, is defined as the following nonlinear constraint:

$$L_m = \sum_{n} \sum_{\substack{e>n \\ e \in CN_n}} Y_{nm} \cdot Y_{em} \quad \forall m \tag{5.4}$$

Overall, the resulting MINLP model (MINLP_Mod) for determining community structures based on the modularity metric maximisation is formulated as:

[MINLP_Mod]

$$\text{Maximise: } Q = \sum_{m} \left[\frac{L_m}{L} - \left(\frac{D_m}{2L} \right)^2 \right]$$

subject to: Constraints (5.2-5.4)

Since global optimality of MINLP models can not be guaranteed, different initial solutions are tested and the best division is selected from a set of candidate solutions. In this approach, we perform N^{max} runs of MINLP_Mod from random initial points

and the node-module allocation and the maximum modularity value is stored for the second stage. We have found that using N^{max}=100 provides a good representation of the solution space.

5.2.2 Stage 2 – Iterative Improvement of Network Partition

Having selected a node-module association with maximum modularity from the previous stage, module allocation is further improved through an iterative fixing and releasing scheme. This stage initially fixes all node-module allocations from stage 1 as binary variables Y_{nm} and nodes in each module are sequentially released while the positions of all other nodes are maintained. After releasing the node-module allocation variable Y_{nm} for all selected nodes, the reduced MIQP model (OptMod) is solved. Comparing the single level MIQP model OptMod, in Chapter 4 to the reduced MIQP models proposed here, the latter involves fewer variables and constraints and can be terminated efficiently. To justify the reason for an iterative reduced MIQP to be preferred over MINLPs we should mention that improved solutions have been achieved by solving a series of reduced MIQP models while no improvements are observed when solving reduced MINLP models iteratively. To avoid releasing too many nodes so that the reduced OptMod model is still difficult to solve, the maximum number of released nodes for module m is set to:

$$MAX_m^r = \frac{U}{Aver_m} \tag{5.5}$$

where $Aver_m$ denotes the average degree in module m without considering the inter-module links and U is a user-defined parameter. Our computational experiments show that the value of U=200 is able to provide satisfactory results for all the examples studied. To illustrate how the improvement procedure works, nodes are first released in m_1, if the number of nodes in module m_1 is greater than $MAX_{m_1}^r$ then nodes in module m_1 will be split into a number of batches and released sequentially. After all nodes in module m_1 are released, the same scheme is applied sequentially to remaining modules until all nodes in the network are released, which completes one round of the major improvement iteration. The same strategy starts again from module

m_1 until no improvement of the modularity value is reported for two successive major iterations. This procedure is schematically represented in Figure 5.1.

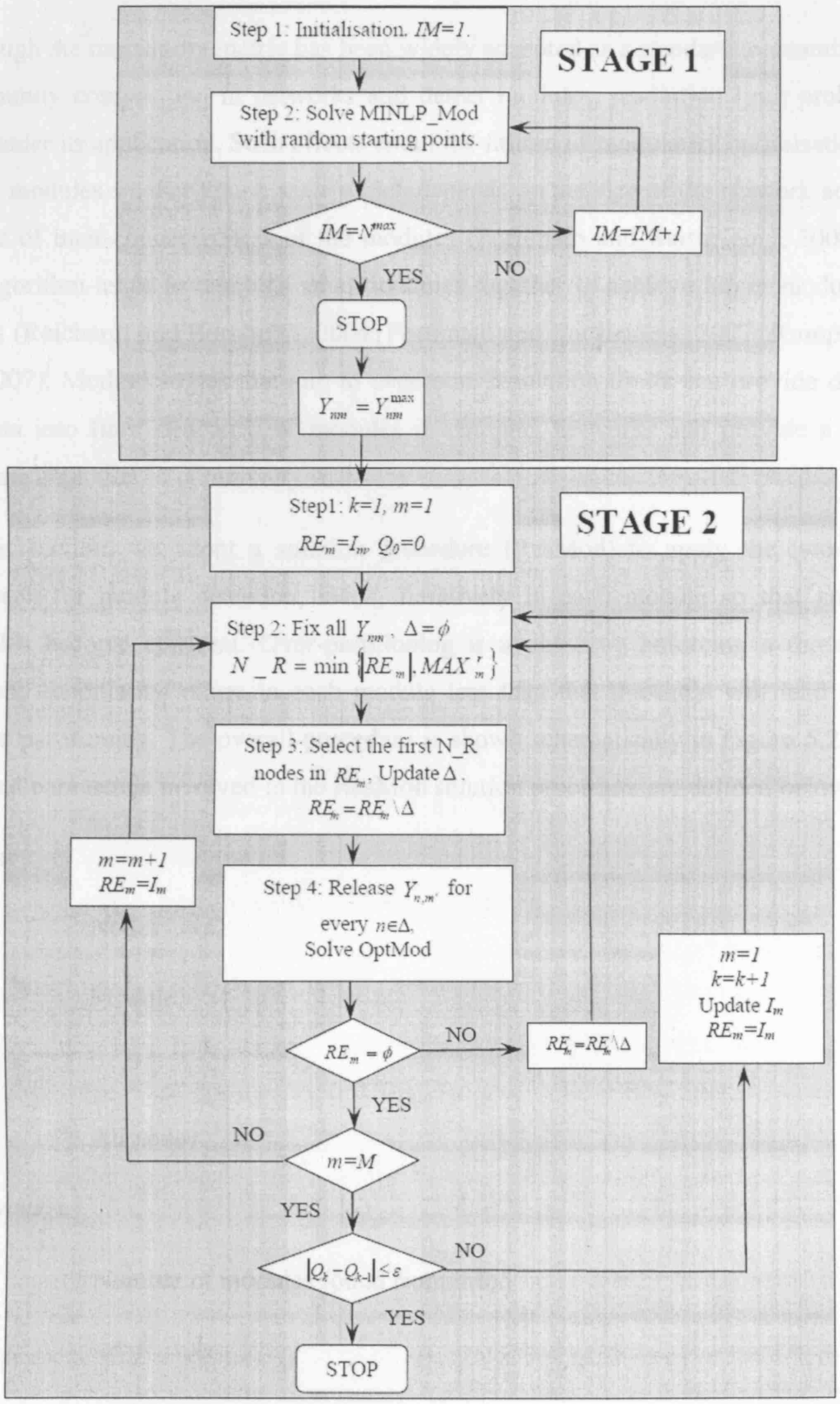


Figure 5.1 Flowchart of the iMod Algorithm

5.3 A Solution Procedure to Correct for Resolution Limitations

Although the modularity metric has been widely accepted as a standard to quantify the community composition in networks and detect modules, resolution limit problems can hinder its application. Such effects entail the failure of modularity optimisation to detect modules smaller than a scale which depends on the size of the network and the degree of inter-connectedness of the modules (Fortunato and Barthelemy, 2007), as the algorithm tends to combine small modules together to achieve larger modularity values (Reichardt and Bornholdt, 2004; Fortunato and Barthelemy, 2007; Kumpula *et al.*, 2007). Methodologies that aim to overcome resolution limits can provide deeper insights into finer structures of modules in complex networks and provide a more accurate depiction of community structure.

In this section, we adopt a solution procedure (ResMod) to apply the two-stage approach for module detection, iMod, iteratively in each module so that smaller modules become apparent. Over-partitioning is avoided by enforcing a threshold, whereby modularity values in each module less than that threshold will lead to no further partitioning. The overall procedure is shown schematically in Figure 5.2. The sets and parameters involved in the ResMod solution procedure are defined below:

Indices

n	Nodes
m	Module

Set

I	All nodes

Parameters

N_M	Number of modules found from iMod

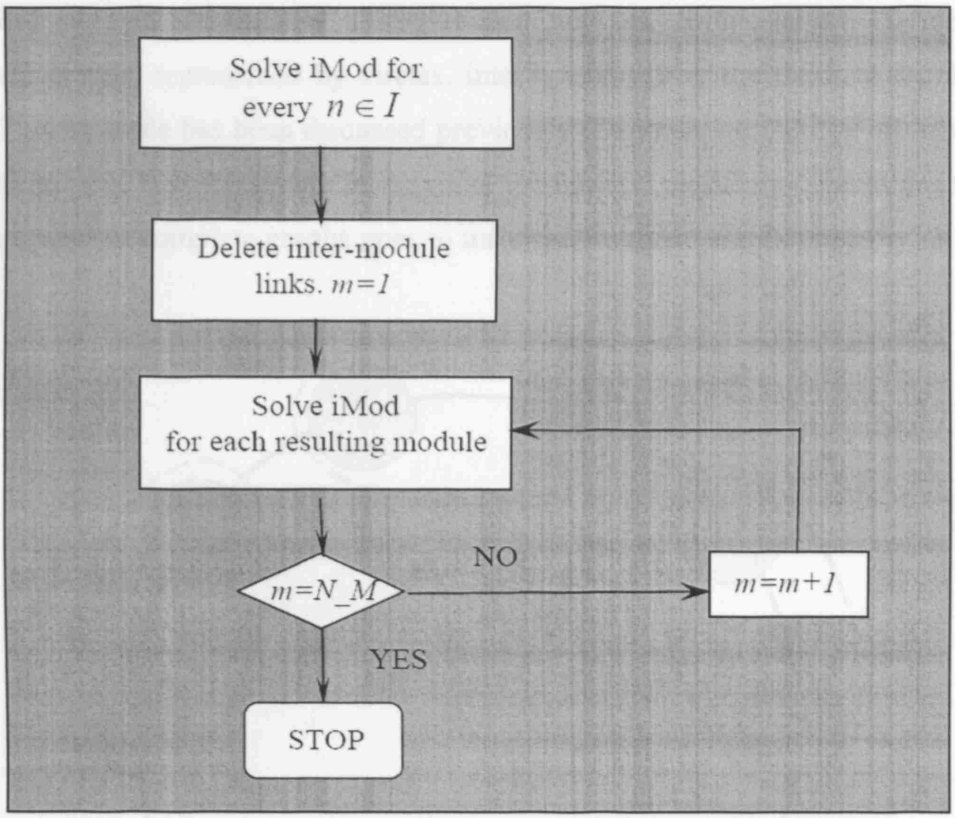


Figure 5.2 Flowchart of the ResMod algorithm

To illustrate how the proposed methodology can be used to overcome resolution limitations when suggesting community structure, two synthetic examples from the literature (Fortunato and Barthelemy, 2007) are used as particularly challenging cases. These network examples are shown schematically in Figures 5.3 and 5.4 and Table 5.1 lists the names, total number of nodes, N, total number of links, L. Median and best modularity values Q are reported after module detection with the iMod methodology (out of ten runs). Modularity (Q_Reso) is shown after accounting for resolution problems with the ResMod approach. Number of modules before and after correcting for resolution is also reported.

Table 5.1 Computational results for simulated network examples

Networks			iMod			ResMod	
Name	N	L	Median Q	Best Q	No. modules	Q_Reso	No. modules
Simu1	30	40	0.6750	0.6750	5	0.6500	10
Simu2	50	40	0.5426	0.5426	3	0.5416	4

The first example (simu1) is a ring-shaped network composed of 10 identical complete graphs, represented by circles, inter-connected by the minimal number of links. This example has been discussed previously (Danon *et al.*, 2005; Fortunato and Barthelemy, 2007) as a test case of maximal modularity: modularity converges to one as the number of complete graphs goes to infinity (Fortunato and Barthelemy, 2007).

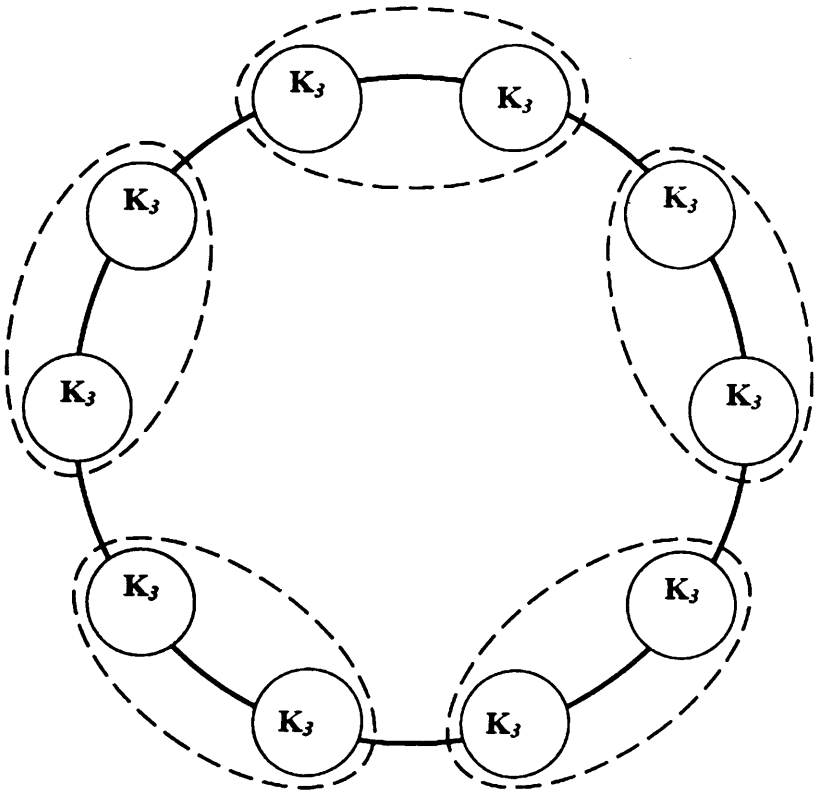


Figure 5.3 Community structures for network example (simu1).

Modularity maximisation using iMod suggests the existence of five modules, in accordance to other approaches. However, when implementing ResMod to correct for resolution limits by optimising each community further without considering the inter-module links, two smaller groups within each module are correctly identified.

The second synthetic example (simu2) comprises four groups, each group is denoted by a circle and consists of completely connected graphs: the two leftmost groups comprise of 20 nodes and the two on the right consist of five nodes each (Fortunato and Barthelemy, 2007). Methods that perform modularity maximisation will tend to merge the two smallest groups; as such a partition will yield the highest modularity value, though without representing the correct community structure. Here, partitioning

the network through iMod and optimising each module through ResMod yields the accurate number of modules and network partitions. Both synthetic examples are rather extreme cases that aim to verify the accurate detection of community structure through iMod and ResMod. The overall results for module detection and resolution correction for the synthetic examples discussed here are shown in Table 5.2.

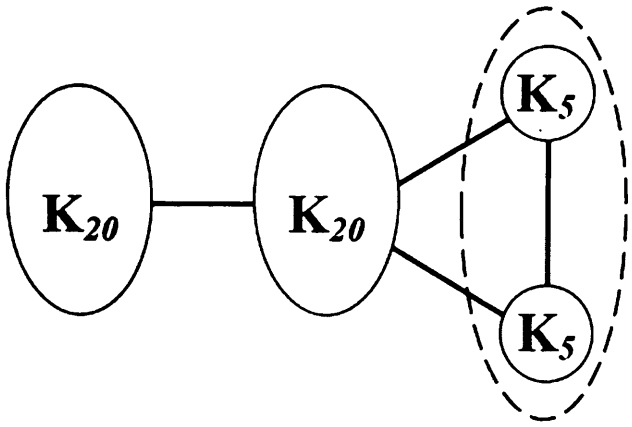


Figure 5.4 Community structures for network example (simu2).

5.4 Results and Discussion

Module detection through iMod and correction for resolution limitations through ResMod are illustrated in this section through a number of network examples. All implementations were performed in GAMS (General Algebraic Modelling System) (Brooke, *et al.* 2003) and models (MINLP and MIQP) are solved using SBB (http://www.gams.com/dd/docs/solvers/sbb.pdf) and CPLEX (Ilog, 2006) mixed integer optimisation solvers within 3600 seconds. Each round of a module detection experiment involves running iMod (incorporating MINLP_Mod and OptMod) ten times and reporting the best and median modularity values, as shown in Table 5.2. ResMod is subsequently used on the partitioned networks to resolve possible resolution problems and identify finer modular structures that may be present. A comprehensive comparison of our approach with other module detection methodologies from the literature is presented to show that the method proposed here is a significant improvement over previous approaches.

Table 5.2 Computational results for comparing the performance of modularity optimisation methodologies across several network examples.

Networks			iMod			GN	DA	EIG	AK	SA
Name	*N*	*L*	Median_Q	Best_Q	No. modules	Q	Q	Q	Q	Q
Zachary	34	78	0.420	**0.420**	4	0.401	0.419	0.419	0.420	-
Dolphin	62	159	0.529	**0.529**	5	0.520	-	-	0.529	-
Les Miseables	77	254	0.560	**0.560**	6	0.540	-	-	0.560	-
P53	104	226	0.535	**0.535**	7	-	-	-	-	-
Jazz	198	2742	0.445	**0.445**	4	0.405	0.445	0.442	0.445	-
E. coli	418	519	0.780	**0.781**	19	-	-	0.766	-	0.752
S. cerevisiae	688	1079	0.768	**0.775**	25	-	-	0.759	-	0.740
C. elegans	453	2025	0.451	**0.453**	9	0.403	0.438	0.435	0.450	-
Email	1133	5451	0.575	**0.580**	9	0.532	0.574	0.572	0.579	-

A number of networks identified from the literature serve as test cases to showcase the efficiency of the computational methodology presented here. Overall, we used eight examples with varying sizes in terms of total number of nodes and links. These cases are inspired from social or biological relationships and represent well-studied cases in network analysis and related algorithm development. The networks describing social interactions are (in ascending number of nodes): the Zachary network of social relationships in an American university club (Zachary, 1977), the communications among dolphins constructed through a field study (Lusseau, 2003), relations among roles in Les Miserables novel (Knuth, 1993), a network of jazz musicians as described through their recordings (Gleiser and Danon, 2003) and a university network of email communication (Guimera *et al.*, 2003). Biological networks assessed in this study are: the p53 protein interaction network (Dartnell *et al.*, 2005), the transcriptional network of the bacterium *Escherichia coli* (Shen-Orr *et al.*, 2002), the transcriptional network of the yeast *Saccharomyces cerevisiae* (Milo *et al.*, 2002) and the network of metabolic reactions of the nematode *Caenorhabditis elegans* (Jeong *et al.*, 2000).

The choice of example networks used here was based on these cases for which module detection through modularity optimisation was previously performed and reported. Even though our main experience and interest is in the analysis of biological networks, we also report the use of social networks for comparison purposes with previous methodologies that have been implemented using in these examples. We would like to mention here an intrinsic difference between social and biological networks: modules in social networks suggest patterns of communication between the underlying parties, whereas as modularity in biological networks is much more complex and can suggest different patterns of evolutionary inheritance of features, genome organisation properties and functional attributes (Ravasz *et al.*, 2002). The analysis of such effects will be the subject of future work.

Similar approaches from the literature for modularity maximisation that we compare our methods against are: an algorithm based on edge-betweenness (GN) (Newman and Girvan 2004) that involves the iterative removal of edges with the highest similarity score to split the network into communities, the eigenvector approach (EIG) where network modularity was rewritten as eigenvectors of a modularity matrix and

lead to a spectral algorithm for community detection (Newman 2006), extremal optimisation (DA) (Duch and Arenas, 2005), simulated annealing (SA) (Fortunato and Barthelemy, 2007) and rounding mathematical programming algorithms (AK) (Agarwal and Kempe, 2007). The results obtained from iMod are compared with these methodologies and the best modularity values are highlighted in bold (Table 5.2). In every example tested, iMod obtained the best value for modularity, as compared to all other approaches.

The first four examples (Zachary, dolphin, Les Miserables and p53 networks) are small in size and can be solved to global optimality through simultaneous MIQP model OptMod proposed in Chapter 4. The computational results are reported in Table 5.2 and clearly show that the iMod algorithm achieves all the optimal solutions for these small-scale examples, demonstrating the favourable results from the proposed approach compared to the other methodologies.

As we reported previously in Chapter 4 and mentioned above, the second stage of our two-stage methodology, OptMod, performs excellently in small to medium size network partitions and can identify a global maximum value of modularity. The benefits of the combination of both stages in the two-stage approach become more apparent as we move to larger examples, which we discuss below. Table 5.2 lists details of five larger networks considered, namely the jazz, *E. coli*, *S. cerevisiae*, *C. elegans* and email networks. Even in large networks, the iMod two-stage process achieved a network partition with the highest value of modularity among all approaches tested.

Further refinement of the community structures proposed is achieved through handling resolution limits of modularity maximisation. Network modules obtained through the iMod process are further partitioned using ResMod ignoring all inter-module links. Finer modular structures are identified if the modularity value of the resulting community is more than a threshold value of 0.3. This threshold value was selected since modularity more than 0.3 indicates strong community structures (Newman 2003). In other words, modules will be further partitioned only if further division still shows strong modular presence (the modularity of each individual module is great than a threshold value). This criterion is implemented to successfully avoid over-partition that may hinder method applicability. The synthetic networks

presented in the previous section have been used as benchmarks to verify that resolution problems are addressed accurately here.

Improved module structures have been detected for the dolphin, p53, *E. coli* and *S. cerevisiae* networks, while no improvement has been detected for the remaining examples (see Table 5.3). Excluding the dolphin network for which a marginal increase to the number of modules was observed after ResMod, all larger size networks have shown a significant increment to the number of modules proposed after the treatment for resolution. As indicated in Table 5.3, modules more than doubled in the p53 and *S. cerevisiae* networks and were four-fold higher in the *E. coli* and *C.elegans* cases. Such wide differences indicate the importance of an accurate procedure in module detection, especially in the cases of larger networks and in biological networks. Future work will concentrate on providing quality control measure in module detection methodologies and thus assessing related computational methodologies not only on the computational efficiency but also on which partitioning relates best to physical significance.

The only other computational methodology that accounts for resolution limitations (Fortunato and Barthelemy, 2007) is the use of simulated annealing for modularity maximisation, followed by the same methodology applied to each module discovered to find out whether any sub-module are identified. Similar results have been reported compared to our methodology: the *E. coli* network was partitioned into 79 modules with a modularity of 0.675 here, compared to 76 modules with modularity of 0.661 (Fortunato and Barthelemy, 2007) and for the yeast network we report 66 modules with modularity of 0.693, as opposed to 57 modules with a total community modularity of 0.677. In both cases, ResMod succeeded in further dividing a higher number of modules and achieved a partition with a better overall modularity score. Furthermore, on the simulated annealing approach there is no indication that small communities obtained from the further partitions still present strong modular structures since it can not be guaranteed that all submodules are not over-partitioned. In contrast, ResMod will only partition modules further if there is indication of strong modular presence (the modularity of each individual module is great than a threshold value), which successfully avoids over-partition.

Table 5.3 Computational comparison of the modularity achieved without correction for resolution problems (iMod) and after accounting for resolution (ResMod) with indication of the number of modules detected in each of these two cases.

Name	iMod Q	No. modules	ResMod Q	No. modules
Zachary	0.420	4	0.420	4
Dolphin	0.529	5	0.504	7
Les Miseables	0.560	6	0.560	6
P53	0.535	7	0.469	16
Jazz	0.445	4	0.445	4
E. coli	0.781	19	0.675	79
S. cerevisiae	0.775	25	0.693	66
C. elegans	0.453	9	0.366	44
Email	0.580	9	0.432	72

5.5 Concluding Remarks

Detecting community structures in complex networks is of crucially importance to reveal the relations between structures and functions of many complex systems. Most existing approaches rely on a quality index, called "modularity" to measure the community structure quality. A general mathematical programming model for network community structure identification has been developed in Chapter 4 to identify the optimal community structure based on modularity maximisation. However, the mathematical framework can only achieve global optimal solutions for small/medium networks. In this chapter, the applicability of mathematical programming to network community detection has been extended by proposing a two-stage solution approach iMod for medium/large networks using mixed integer optimisation. In the first stage, a good initial partition is selected from a number of candidate solutions after solving an MINLP model with different starting points. Furthermore, an iterative procedure has been adopted to improve the solution obtained from the first stage. In order to overcome the resolution limit problem of modularity optimisation, an iterative process ResMod was applied to detect small modules hidden inside each community. The applicability of the proposed methodology has finally

been demonstrated through a number of simulated and real network examples. Comparing with other existing methodologies in the literature, the proposed approach obtained more accurate community structures with higher modularity values. Furthermore, a solution procedure, ResMod, is applied to each network community obtained without considering the inter-community links so that finer community structures become visible. Overall, the proposed optimisation-based approaches in this chapter successfully extend the applicability of mathematical programming techniques in the area of network community structure identification and accurately achieved modular structures of medium/large scale complex networks.

PART III

DNA SEQUENCE ANALYSIS

Chapter 6

Finding DNA Motifs and Consensus Sequences using Mathematical Programming

Discovering motifs involves the identification of sub sequences with high similarities in DNA sequences. Finding such motif is crucially important to identify regulatory regions which usually correspond to protein binding sits in DNA sequences for transcription factors. In this chapter, we propose a general mathematical programming framework for motif finding. The overall problem is formulated as a mixed integer linear programming (MILP) model which determines the contents of a consensus sequence and motifs for all DNA sequences so as to maximise the total matching scores between the consensus sequence and all motifs. An iterative solution procedure is developed to find multiple motifs for each single DNA sequence. The power of mathematical programming to incorporate other biological features is also revealed. The applicability of the proposed approach is demonstrated by a number of

examples and the DNA motifs and consensus sequence found are compared to validated biological motifs and the predicted results of two other computational methodologies.

6.1 Introduction and Literature Survey

A major challenge of post-genomic biology is to understand the mechanisms of regulating the expression of genes. One important step is to identify regulatory elements/motifs, which usually correspond to protein binding sites in DNA sequences for transcription factors. Finding DNA motifs is crucially important for biologists to locate gene regulatory sites, identify drug targets and investigate cell reactions under physiological and pathological conditions. A number of computational methodologies have been proposed to tackle such problems considering different motif representation, scoring functions for motif quality assessment as well as the search procedure. Next, a brief description of major methodologies for DNA motif finding is provided.

Probabilistic Approaches: probabilistic approaches typically maximise the information entropy of the chosen motif instances or the likelihood ratio of the motif model to the background model. The log likelihood ratio and information content are two major commonly used functions. First, Hertz *et al.* (1990) developed a method for identifying consensus patterns in a set of unaligned DNA sequences. Such sequences are known to bind a common protein or to have some other common biochemical functions. The goal of the proposed method is to find the most significant motifs which have the lowest probability of occurring by chance out of all candidate motifs formed from the set of related sequences. Lawrence and Reilly (1990) improved such method by proposing an expectation maximisation (EM) process to find motif locations and contents. Later, the MEME (Multiple Em for Motif Elicitation) algorithm developed by Bailey and Elkan (1995) extends the EM algorithm for identifying motifs in unaligned DNA sequences. MEME expands the range of problems which can be solved using EM and increases the chance of finding good solutions. Sub sequences which actually occur in the DNA sequences are used as starting points for the EM algorithm to increase the probability of finding global optimal motifs. The assumption that each sequence contains exactly one occurrence of the shared motif is relaxed, which allows multiple appearances of a motif to occur in any sequence and permits the algorithm to ignore sequences with no appearance of

shared motifs. Shared motifs can be erased so that several distinct motifs can be found in the same set of sequences. However, EM has a weakness that it cannot escape local optimal solutions since EM-based approaches are gradient descent methods and always converge in a small number of iterations.

Gibbs sampling method is another extensively used motif finding approach. Initially, Lawrence *et al.* (1993) presented a Gibbs sampling algorithm for the DNA and protein local alignment problem. Later, Roth *et al.* (1998) developed the motif finding algorithm, namely AlignACE (Aligns Nucleic Acid Conserved Elements). The AlignACE approach used a maximum a priori log likelihood score, which gauges the degree of overrepresentation and returns a series of motifs as weight matrices that are overrepresented in the input set of DNA sequences. It also takes into account the sequence of the entire genome and highlights those motifs found preferentially in association with the genes under consideration. Moreover, the GLAM (Gapless Local Alignment of Multiple sequences) algorithm proposed by Frith *et al.* (2004) automatically determined the optimal motif length and evaluated the statistical significance of the resulting output. Finally, Thompson *et al.* (2003) developed the Gibbs Motif Sampler software package, containing a number of major Gibbs sampling techniques proposed by Lawrence *et al.* (1993), Liu *et al.* (1995) etc.

Apart from expectation maximisation methods and Gibbs sampling-based approaches, other probabilistic methodologies are also proposed. Hu *et al.* (2006) proposed a clustering-based ensemble algorithm named EMD for de novo DNA motif discovery. EMD approach is able to combine multiple predictions from multiple runs of one or more basic component algorithms proposed from the literature so as to improve the prediction accuracy. Redhat and Bailey (2007) proposed novel discriminative probabilistic algorithm, namely, DEME, to discover DNA and protein motifs. Comparing with other motif finding algorithms, the proposed approach needs two sets of input sequences called "positive set" and "negative set". Local and global search procedures were applied to discriminate negative sequence sets from positive sequences. Finally, Chakravarty *et al.* (2007) developed a novel ensemble learning method, SCOPE. The proposed approach assumed that transcription factor binding sites belong to one of three broad classes of motifs: non-degenerate, degenerate and gapped motifs. SCOPE employs a unified scoring metric to combine the results from three motif finding algorithms each aimed at the discovery of one of these classes of

motifs. Finally, Larsson *et al.* (2007) developed a web-based motif discovery tool, HeliCis, to detect DNA motifs with periodic spacing. By tuning the parameter settings of the tool, colocalised motifs without periodicity or motifs with known or unknown lengths separated by fixed and periodic gaps can be identified.

Recently, Tompa *et al.* (2005) performed a summary of most existing statistical computational tools available for DNA motif finding and provided an extensive runs on a variety of testing examples. This work provided some guidance to users regarding the accuracy of currently available tools through a number of benchmark datasets. They also proposed a number of statistics to assess the prediction performance of all computational tools. Hu *et al.* (2005) developed a comprehensive set of performance measures at the nucleotide, binding site and motif levels and systematically evaluated five motif discovery algorithms using a prokaryotic motif dataset. The computational results indicated that there are number of limitations of current probabilistic approaches. First, the prediction accuracy of most methodologies is still relatively low. Second, the probability-based methods have little flexibility to incorporate other biological features. Third, many popular motif discovery algorithms are based on heuristics which usually result in local optimal solutions rather than global ones. Finally, the performances of these methods are highly subject to user-defined parameter values and starting points in the search space.

Combinatorial Approaches: Compared to the probabilistic approaches, combinatorial methods search motifs by proposing a number of mathematical frameworks to optimise different objectives such as the information content and sum-of-pair matching scores. Combinatorial models are straightforward to implement through standard modeling tools and very few parameters are required. More importantly, it is relatively easy to extend current mathematical models by incorporating other additional conditions and constraints using algebraic equations. A number of mathematical programming models and solution procedures have been developed to solve string selection and comparison problems including the closest string, closest sub string and fastest string problems share many similarities with motif finding. Li *et al.* (2002) showed most combinatorial models for string comparison and motif finding are NP-hard and proposed a polynomial time approximation algorithm to achieve near optimal solution efficiently. Meneses *et al.* (2004) proposed three integer-programming (IP) formulations and a heuristic to solve the closest string problems.

Tight constraints are used to provide upper bounds of optimal solutions. These results showed that the proposed mathematical programming approach is able to solve instances of moderate size to optimality. Later, Meneses *et al.* (2005) have summarised a number of optimisation models and solution approaches for various string comparison problems. Li and Fu (2005) developed an MILP model to tackle the DNA consensus sequence identification problem where the optimisation model determined a consensus sequence with fixed length and the positions of motifs in each DNA sequence so that the total matching scores between the consensus sequence and the DNA motifs are maximised. Zaslavsky and Singh (2006) introduced a combinatorial optimization framework for diverse motif finding problems. The proposed mathematical model combined graph pruning techniques with a novel integer linear programming (ILP) formulation. The objective function is maximisation of total similarity scores of all pairs of DNA motifs. This optimisation-based approach was proved to be flexible and robust enough to model several variants of the motif finding problem. Finally, a more compact mathematical programming model was presented for DNA motif discovery (Kingsford *et al.*, 2006) using the distances between sub sequences come from a limited set of possibilities rather than all pairs of potential motifs.

In this chapter, a general mathematical programming framework is proposed to solve DNA motif finding problems. The overall problem is formulated as a mixed integer linear programming (MILP) model, which determines the content of a consensus sub sequence within a collection of DNA sequences and motifs for each sequence so that the total matching scores between the consensus sequence and each motif are maximised. The key advantages of the proposed mathematical programming framework are: (i) global optimal solutions for the motif and consensus sequence identification problem are guaranteed. (ii) better prediction accuracies are achieved when compared with the other two probabilistic methodologies and (iii) additional biological features can easily be incorporated into current framework thus illustrating the flexibility of our approach for further expansion. In the next section, the general DNA motif finding problem is stated. Section 6.3 presents a novel mathematical formulation for the DNA motif identification problem and section 6.4 presents an iterative algorithm to find multiple motifs for each sequence. In section 6.5, the

applicability of the proposed methodology is demonstrated by a number of illustrative examples. Finally, some concluding remarks are made in section 6.6.

6.2 Problem Statement

Consider a DNA motif finding problem with *TL* DNA sequences. Each DNA string may have different length and consists of A, T, G and C. Motifs represent a set of sub sequences with high similarities inside a set of DNA sequences. The consensus sequence is a sub sequence with same length as those of motifs indicating the most conserved regions of the motifs. Figure 6.1 clearly shows the process of identifying consensus sequence and motifs. In Figure 6.1, we assume that there are three DNA sequences available and each sequence contains eleven positions. The length of the consensus sequence and motifs are fixed as six. The DNA motifs identification problem proposed here aims at finding the highly reserved region for each sequence (see the grey area of each DNA sequence in Figure 6.1) and identifying a consensus sequence representing the feature of all motifs.

DNA sequence 1: TGCGTAAAGTT

DNA sequence 2: GTACGGCGTTA

DNA sequence 3: AGGGCGTTACT

Consensus sequence: GCGTTA

Figure 6.1 Motifs and consensus sequence identified from three DNA sequences

Overall, The DNA motif identification problem can be stated as follows:

Given:

- A collection of DNA sequences
- The length of consensus sequence and motifs

Determine:

- The consensus sequence content
- The motif positions and contents of each DNA sequence

so as to

- Maximise the total matching scores between the consensus sequence and motifs

6.3 Mathematical Formulation

In this section, a rigorous mathematical model for DNA motif finding problems is presented. It is first assumed that *only* one motif will be identified for each DNA sequence. The above assumption is relaxed in section 6.4 to discover multiple motifs for each sequence. The indices, parameters and variables associated with the DNA motif finding problem are listed below:

Indices

l	DNA sequence ($l=l_1, l_2, \ldots, TL$)
p	DNA position ($p=p_1, p_2, \ldots, P$)
s	Motif starting position ($s=s_1, s_2, \ldots, S$)
i	Consensus sequence position ($i=1,2, \ldots, I$)
k	DNA letters (k=A, T, G, C)

Set

SK_{ik}	Set of candidate motifs that have letter k on position i

Parameters

d_{lp}	Letter of position p on sequence l
t_{lsi}	Letter of position i of candidate motif in sequence l starting from position s
I	Motif and consensus sequence length
LE_l	Length of sequence l
m	Iteration number in algorithm MultiMotif
M	Total number of motifs per sequence

Binary variables

Z_{ls} 1 if motif in sequence l starts from position s; 0 otherwise

L_{ik} 1 if position i of consensus sequence is letter k; 0 otherwise

Continuous variables

W_{ik} Matching scores of letter k on position i

6.3.1 Sequence Data Pre-processing

The data pre-processing step generates candidate motifs with length of I from the DNA sequence set. A DNA sequence having LE_l letters can be decomposed into a collection of sub sequences with length of I:

$$t_{lsi} = d_{l\,i+s-1} \qquad \forall l, s, i \qquad (6.1)$$

It can easily be seen that for a DNA sequence with LE_l letters, equation (6.1) can generate LE_l $-I+1$ candidate motifs. Figure 6.2 illustrates the enumeration all candidate motifs of the DNA sequence sets described in Figure 6.1. DNA motifs for each sequence will be selected from the candidate sets through the proposed mathematical model. It is also noted that the data pre-processing stage is also able to generate other potential motifs (e.g. motifs with gaps; reverse complementary motifs) if particular patterns of DNA motifs are required.

Candidate motifs from sequence 1:

TGCGTA GCGTAA CGTAAA

GTAAAG TAAAGT AAAGTT

Candidate motifs from sequence 2:

GTACGG TACGGC ACGGCG

CGGCGT GGCGTT GCGTTA

Candidate motifs from sequence 3:

AGGGCG GGGCGT GGCGTT

GCGTTA CGTTAC GTTACT

Figure 6.2 Enumeration of all candidate motifs

6.3.2 Allocation Constraints

First, only one letter from $\{A,T,G,C\}$ can be allocated into each position in the consensus sequence:

$$\sum_k L_{ik} = 1 \qquad \forall i \qquad (6.2)$$

Second, each motif can only start from one position in each DNA sequence:

$$\sum_s Z_{ls} = 1 \qquad \forall l \qquad (6.3)$$

6.3.3 Definition of W_{ik}

W_{ik} is defined as the total matching scores of letter k on position i between the consensus sequence and all DNA motifs. SK_{ik} is the set include all positions from candidate motifs with letter k:

$$SK_{ik} = \{(l,s) | t_{lsi} = k\}$$

If the potential motif from sequence l starting from position s is selected (i.e. binary variable $Z_{ls} = 1$), also the ith position of the selected motif is letter k, such motif will contribute to the value of W_{ik} :

$$W_{ik} \le \sum_{(l,s) \in SK_{ik}} Z_{ls} \qquad \forall i,k \qquad (6.4)$$

6.3.4 Logical Constraint

The following logical constraint is introduced between binary variables W_{ik} and L_{ik} :

$$W_{ik} \le TL \cdot L_{ik} \qquad \forall i,k \qquad (6.5)$$

where TL denotes the total number of DNA sequences involved and binary variable L_{ik} denotes whether position i of the consensus sequence is letter k. Therefore, the above constraint enforces that W_{ik} will have non-zero value only when position i of the consensus is letter k (i.e. $L_{ik} = 1$).

6.3.5 Objective Function

The objective function used in this model is the maximisation of the total matching scores between the consensus sequence and motifs:

$$\max \quad \sum_i \sum_k W_{ik} \tag{6.6}$$

Overall, the resulting mathematical model (OptMotif) for determining DNA motif and consensus sequences is formulated as follows:

[OptMotif]:

Maximise $\sum_i \sum_k W_{ik}$

subject to

constraints (6.2-6.5).

$Z_{ls}, L_{ik} \in \{0,1\}$; $W_{ik} \geq 0$

The resulting mathematical formulation is a mixed integer linear programming (MILP) model. The CPLEX (Ilog, 2006) mixed integer optimisation solver is used to solve the proposed model to global optimality through the branch-and-bound procedure.

6.4 Searching Multiple Motifs

The optimisation-based mathematical model proposed in the previous section searches only for one motif per DNA sequence. It is also important from the biological point of view to provide alternative motifs. Here, the proposed mathematical model is extended to identify multiple motifs existing in a set of sequences. If more motifs are required per sequence, then an iterative procedure could be employed. After solving the OptMotif model to obtain the consensus sequence and motifs through the optimal values of variables L_{ik} and Z_{ls}, respectively, we then i:) fix the optimal consensus sequence content (i.e. fix the L_{ik} variable to be the values obtained by OptMotif) and ii:) exclude the motifs already found. The reduced MILP model is then solved to find alternative motifs. The flow chart of solution approach, MultiMotif, is described as follows (Figure 6.3):

[Algorithm MultiMotif]

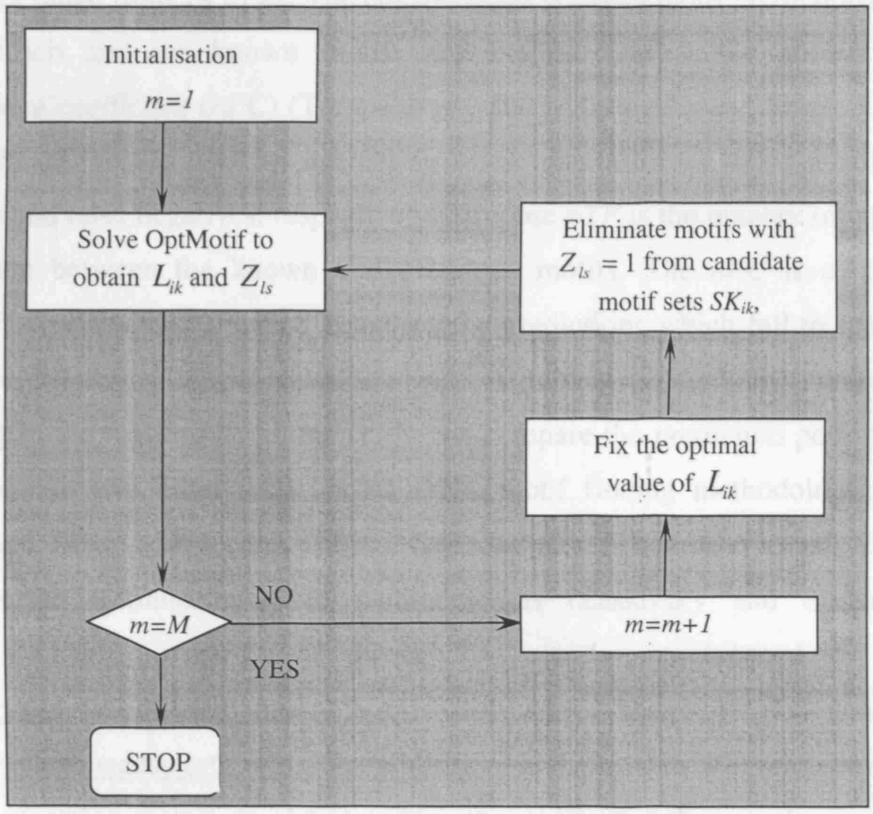


Figure 6.3 Flowchart of Algorithm MultiMotif

6.5 Results and Discussion

In this section, the applicability of the proposed methodology is demonstrated through a number of DNA sequences. The proposed mathematical model and the iterative solution algorithm have been implemented in the GAMS modeling system (Brook *et al.*, 1998) on a 3.40 GHz PC with 2GB memory using the CPLEX mixed-integer optimisation solver with 0% margin of optimality for the branch-and-bound procedure. The computational results from the proposed MILP-based approach are compared with other two motif finding methodologies: MEME and Gibbs Sampler.

6.5.1 DNA Motif Identification

We apply our method to identify the binding sites of six *E.coli* transcription factors. These selected datasets have been used by Zaslavsky and Singh (2006). In addition, the *crp* example provided by Hertz *et al.* (1990) is also selected in this study since it has been extensively investigated by a number of researchers (Hertz *et al.*, 1990; Li

and Fu, 2005). The DNA sequence length varies from 105 to 650 and the motif length parameters range from 14 to 22. The consistencies between motif predictions made by our approach and the known motifs are evaluated using the nucleotide level *performance coefficient* (*nPC*) (Tompa *et al.*, 2005; Zaslavsky and Singh, 2006). Let *nTP*, *nFP*, *nTN*, *nFN* refer to nucleotide level true positives, false positives, true negatives and false negatives, respectively. Suppose *nTP* is the number of nucleotides in common between the known and predicted motifs. The *nPC* is calculated as $nTP/(nTP + nFN + nFP)$, which penalises the predictions which fail to identify any nucleotide belonging to the motif as well as falsely predict any nucleotide not belonging to the real motif. In this study, we compare the prediction performance of our method to two other well known DNA motif finding methodologies MEME (Bailey and Elkan, 1995) and Gibbs Motif Sampler (Thompson *et al.*, 2003) and combinatorial optimisation-based methodologies (Zaslavsky and Singh, 2006). MEME is used through its web-based version (http://meme.sdsc.edu/meme/meme.html). Gibbs Motif Sampler is downloaded from http://bayesweb.wadsworth.org/gibbs/gibbs.html and run with 100 random restarts in a Linux operating system to obtain sufficient sampling of the search space. For all these three approaches, all parameters are kept as their default values. In this study, we search two motifs per DNA sequence (i.e. M=2) using all three approaches and compare them to the real motif available. The closest predicted motif is then selected to calculate the *nPC* value.

Table 6.1 Prediction accuracy comparison (*nPC* value) of all examples

Examples	No. Seq[a]	Motif-Len[b]	MultiMotif	MEME	Gibbs
arcA	11	15	0.521	0.304	0.071
cpxR	7	14	0.500	0.235	0.065
dnaA	6	15	0.875	0.452	0.667
fruR	10	16	0.818	1.000	0.799
metJ	5	16	0.650	0.455	0.571
ntrC	4	17	1.000	0.600	0.850
crp	18	22	0.764[c]	0.788	0.612

a: number of sequences; b: motif length; c: feasible solution within 10,000 seconds

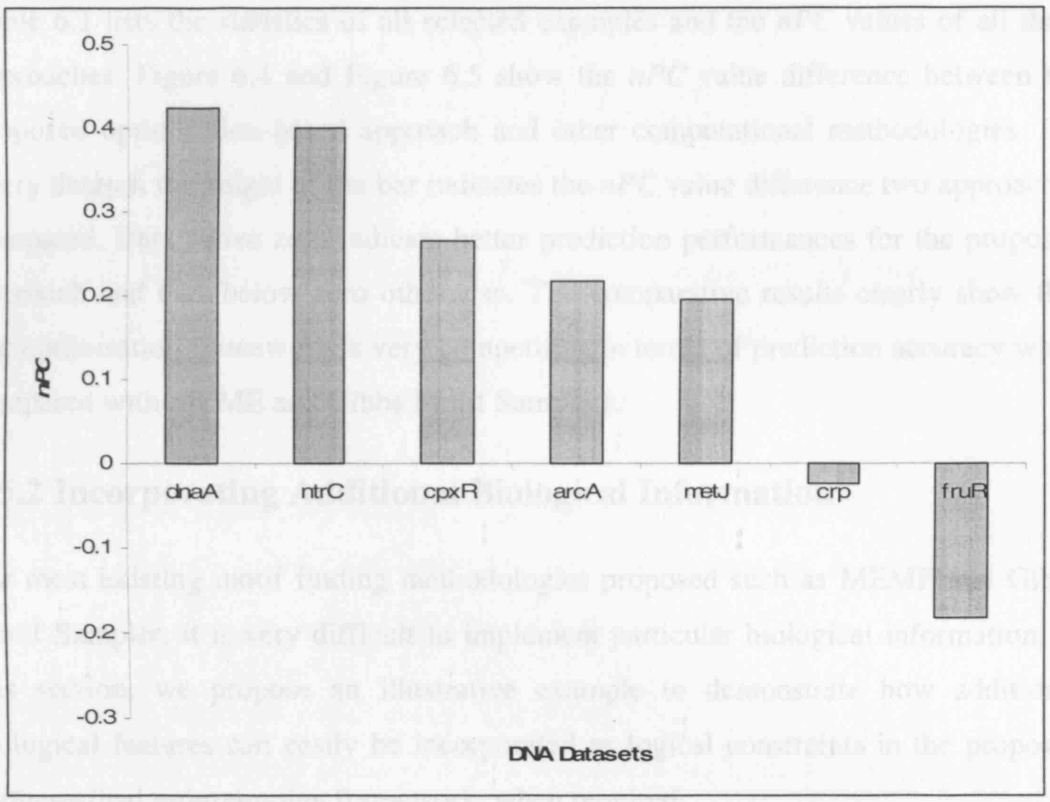


Figure 6.4 Difference of *nPC* values between MultiMotif and MEME

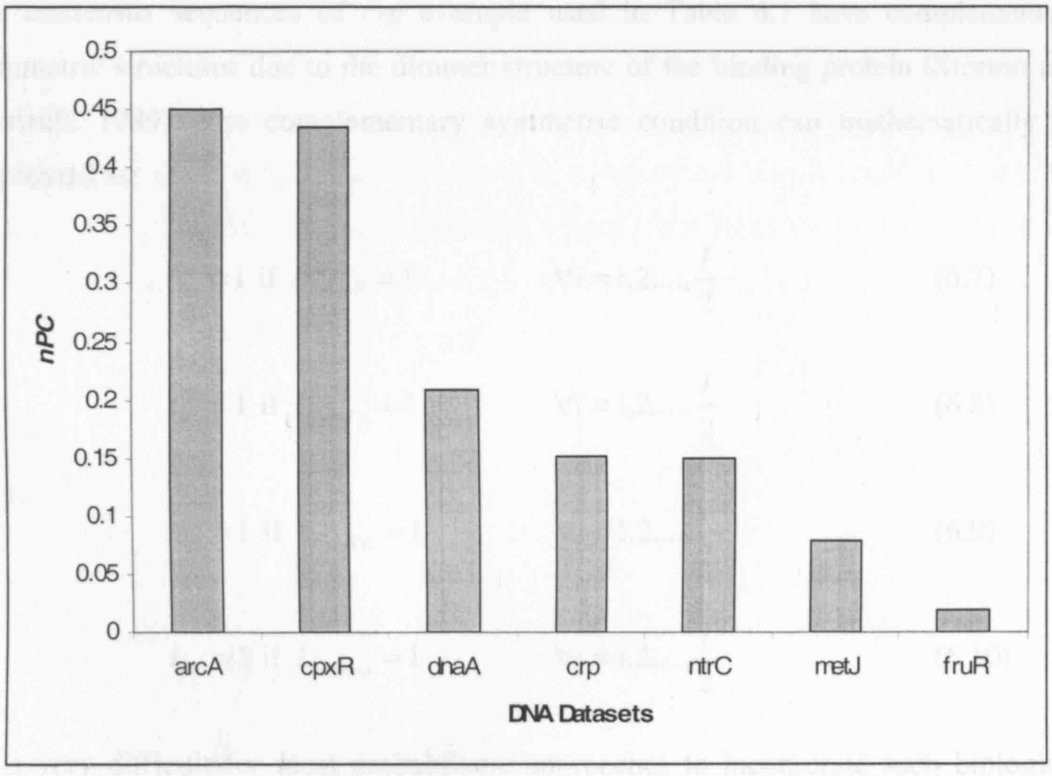


Figure 6.5 Difference of *nPC* values between MultiMotif and Gibbs Motif Sampler

Table 6.1 lists the statistics of all selected examples and the *nPC* values of all three approaches. Figure 6.4 and Figure 6.5 show the *nPC* value difference between the proposed optimisation-based approach and other computational methodologies. For every dataset, the height of the bar indicates the *nPC* value difference two approaches compared. Bars above zero indicate better prediction performances for the proposed approach and bars below zero otherwise. The comparative results clearly show that our optimisation framework is very competitive in terms of prediction accuracy when compared with MEME and Gibbs Motif Samplers.

6.5.2 Incorporating Additional Biological Information

For most existing motif finding methodologies proposed such as MEME and Gibbs Motif Sampler, it is very difficult to implement particular biological information. In this section, we propose an illustrative example to demonstrate how additional biological features can easily be incorporated as logical constraints in the proposed mathematical programming framework, when required.

Take *crp* dataset as an example, biological research from the literature showed that the consensus sequences of *crp* example used in Table 6.1 have complementary symmetric structures due to the dimmer structure of the binding protein (Stormo and Hartzell, 1989). The complementary symmetric condition can mathematically be expressed as:

$$L_{iA} = 1 \text{ if } L_{I-i+1T} = 1 \qquad \forall i = 1,2,...,\frac{I}{2} \qquad (6.7)$$

$$L_{iT} = 1 \text{ if } L_{I-i+1A} = 1 \qquad \forall i = 1,2,...,\frac{I}{2} \qquad (6.8)$$

$$L_{iG} = 1 \text{ if } L_{I-i+1C} = 1 \qquad \forall i = 1,2,...,\frac{I}{2} \qquad (6.9)$$

$$L_{iC} = 1 \text{ if } L_{I-i+1G} = 1 \qquad \forall i = 1,2,...,\frac{I}{2} \qquad (6.10)$$

It is very difficult for most probabilistic approaches to incorporate such biological features. However, additional conditions can easily be accommodated into the proposed optimisation model by adding the following constraint:

$$\alpha \cdot L_{iA} + \beta \cdot L_{iT} + \gamma \cdot L_{iG} + \delta \cdot L_{iC} = \alpha \cdot L_{I-i+1T} + \beta \cdot L_{I-i+1A} + \gamma \cdot L_{I-i+1C} + \delta \cdot L_{I-i+1G}$$
$$\forall i = 1,2,...,\frac{I}{2} \qquad (6.11)$$

where $\alpha, \beta, \gamma, \delta$ are constants and $\alpha \neq \beta \neq \gamma \neq \delta$. According to allocation constraint (6.2), only one of binary variables $L_{iA}, L_{iT}, L_{iG}, L_{iC}$ has the value of 1 and all other three are forced to zero. Therefore, only one term of the left hand side of equation (6.11) has a non-zero value. In order to satisfy constraint (6.11), the term with the same coefficient on the right hand side will be automatically forced to have non-zero values. Therefore, the pair of variables with the same coefficient (e.g. L_{iA} and L_{I-i+1T}) will simultaneously be forced to 1 so as to satisfy equation (6.11).

The extended proposed MILP model (OptMotif-E) incorporating the above logical constraint is used to predict the motifs of example *crp*. Similar to section 6.4.1, we search two motifs per DNA sequence by simply replacing model OptMotif in steps 2 and 3 of algorithm MultiMotif as OptMotif-E and name the new solution algorithm MultiMotif-E. Comparing both motifs obtained to the real motif, the closest one is reported. Table 6.2 lists the starting positions of biologically identified motifs and all predicted motifs from other three approaches. Figure 6.6 graphically shows the actual predicted motif in each DNA sequence. The comparative results listed in Table 6.2 clearly show that MultiMotif-E identifies exactly the same motif locations when compared to the real biological findings after accommodating the complementary symmetric constraint (6.11).

Table 6.2 Comparative results between MultiMotif-E and all other two approaches

Sequences	Known Motif Locations	MultiMotif-E	MEME	Gibbs
cole	17,61	61	60	60
ecoarabop	17,55	55	54	54
ecobglr1	76	76	77	75
ecocrp	63	63	62	62
ecocya	50	50	51	42
ecodeop	7,60	7,60	6	6
ecogale	42	42	23	41
ecoilvbpr	39	39	38	38
ecolac	9,81	9,81	8	8
ecomale	14	14	13	13
ecomalk	29	29	62	53
ecomalt	41	41	42	33
ecoompa	48	48	47	47
ecotnaa	71	71	72	63
ecouxu1	17	17	18	9
pbr-p4	53	53	54	45
trn9cat	84	84	54	N/A
tdc	78	78	77	75

Seq 1 .....ACTGT TTTTTTGATCGTTTTCACAAAA ATGGA.....
Seq 2.....ATTG ATTATTTGCACGGCGTCACACT TTGCT.....
Seq 3.....TTAAT AACTGTGAGCATGGTCATATTT TTATC.....
Seq 4.....TGCAT GTATGCAAAGGACGTCACATTA CCGTG.....
Seq 5.....CAGCA AGGTGTTAAATTGATCACGTTT TAGAC.....
Seq 6.....GTGAA TTATTTGAACCAGATCGCATTA CAGTG.....
Seq 7.....TCCAC TAATTTATTCCATGGCACACTT TTCGC.....
Seq 8.....GTACA AAACGTGATCAACCCCTCAATT TTCCC.....
Seq 9.....GCAAT TAATGTGAGTTAGCTCACTCAT TAGGC.....
Seq 10...GCCAA TTCTGTAACAGAGATCACACAA AGCGA.....
Seq 11...ACGGC TTCTGTGAACTAAACCGAGGTC ATGTA.....
Seq 12...TTTGG AATTGTGACACAGTGCAAATTC AGACA.....
Seq 13....TTCAT ATGCCTGACGGAGTTCACACTT GTAAG.....
Seq 14...CGAAC GATTGTGATTCGATTCACATTT AACAA
Seq 15...GAAAT TGTTGTGATGTGGTTAACCCAA TTAGA.....
Seq 16....ATATG CGGTGTGAAATACCGCACAGAT GCGTA.....
Seq 17...GGCGA AAATGAGACGTTGATCGGCACG
Seq 18....AGTTA ATTTGTGAGTGGTCGCACATAT CCTG.....

Figure 6.6 Motifs identified of *crp* dataset through MultiMotif-E

6.6 Concluding Remarks

DNA motif finding is considered as one of the most important tasks in post genomic era. It focuses on the identification of repeated patterns inside a collection of DNA sequences. There have been quite a few computational methodologies developed to identify DNA motifs over the last 30 years. Most approaches use probabilistic approaches and combinatorial models. However, computational prediction of such regulatory elements is still a complex challenge. In this chapter, a general mathematical programming approach has been proposed to identify the consensus sequence and motifs within a set of DNA sequences. Initially, the pre-processing step generates a collection of potential motifs from all available sequences. Secondly, an optimisation model has been proposed to determine the contents of the consensus sequence and DNA motifs so that the total matching scores between the consensus sub sequence and all DNA motifs are maximised. The overall problem has been formulated as an MILP optimisation model which can be solved to global optimality

using standard branch-and-bound procedure. An iterative solution procedure has been developed to identify multiple motifs for each sequence. The applicability of the proposed approach has been demonstrated through a number of illustrative examples. The prediction performance of our approach has been compared with two other standard DNA motif finding methodologies. The computational results indicate that our approach is very competitive in terms of prediction accuracy when compared with literature computational methodologies. Finally, one additional biological constraint has easily been incorporated in the proposed methodology to improve the motif prediction accuracy thus illustrating the power and flexibility of the proposed mathematical programming framework to accommodate other biological information.

Chapter 7

Conclusions and Future Directions

The aim of the thesis is to *facilitate pattern recognition and knowledge discovery by applying mathematical programming to analyse different forms of data.* Towards this objective, a number of mathematical models and solution procedures have been developed in order to assist researchers from various disciplines to extract patterns and information from data. The key contributions of the thesis are summarised in the next section, while section 7.2 suggests promising new directions for future research work.

7.1 Contributions of the Thesis

7.1.1 Data Classification using Mixed Integer Optimisation

Part I of the thesis concerned with the problem of data classification with two or multiple groups and consists of Chapters 2 and 3. In Chapter 2, a mixed integer optimisation-based approach has been proposed for the general data classification

problem. First, an extensive and comprehensive literature survey was presented in order to familiarise readers with general data classification problems and major research developments particularly mathematical programming-based classification models. Following the literature survey, we focus on the data classification problem using hyper-box classifiers. The overall problem has been formulated as an MILP representation where the boundaries of each distinct class have been determined by one hyper-box classifier so as to enclose the maximum correctly classified training samples and the total misclassified samples are minimised. In order to improve the training and testing performances of the proposed approach, an iterative solution algorithm has been developed to allow multiple hyper-boxes for each single class. In the testing stage, the memberships of some new data samples with unknown class labels have been identified by calculating the distances between testing samples to all existing hyper-boxes established in the training part. The unclassified sample is then assigned to the closest box. Finally, the applicability of the methodology has been demonstrated through five literature examples and two synthetic datasets. The hyper-box representation has been shown to be particularly suitable to classify data samples with disjoint regions. The prediction performance of our approach has been compared with six other standard classification methods over three different testing scenarios. The computational results indicated that our approach is competitive in terms of prediction accuracy when compared with other alternative classification methodologies.

The proposed optimisation-based approach for data classification problems in Chapter 2 was able to complete the training task with a relatively few samples due the existence of binary variables in the MILP models involved. In Chapter 3, an efficient two-stage solution algorithm was developed to extend the applicability of hyper-box classifiers to larger and more complex datasets. In the first stage, an efficient decomposition scheme was introduced to partition all training samples into a number disjoint and non-overlapped regions without considering their class memberships. In the second stage, hyper-box classifiers were applied to complete the training of each partitioned region using the optimisation-based approach proposed in Chapter 2. In order to test the prediction performance of the proposed approach, an extensive computational experiment was carried out using five real datasets and two synthetic examples. These datasets involved large training samples and complex data

distributions and can not be solved efficiently using the mixed integer optimisation-based approach developed in Chapter 2. The computational results revealed that our approach was very competitive when comparing the prediction accuracies of other competing methodologies from the literature. It was also noted that the proposed two-stage solution approach showed a strong parallel structure that can be used by distributing the whole task into different computers so as to significantly improve the computational efficiency.

7.1.2 Network Community Structure Identification

Part II of the thesis involved the problem of network community structure identification and contained Chapters 4 and 5. In Chapter 4, a simultaneous mixed integer quadratic programming (MIQP) model was developed to identify the community structures of small/medium network. First, a comprehensive introduction was performed to indicate how networks represent various real world complex systems. The importance of identifying modular structures of networks to study system functions was also indicated. Recent approaches to find network community structures were then summarised in the literature review section and the major strength and weakness of each approach was pointed out. Since proposing the concept of network modularity (Newman and Girvan, 2004), the community structure detection problem can be posed as an optimisation task which finds an optimal partition at the maximum modularity value. The major disadvantage of all existing approaches based on modularity maximisation is the global optimal solution can not be guaranteed. We then presented a general mathematical programming framework to identify the optimal partition of a network into communities with the maximum modularity metric. The resulting mathematical formulation is an MIQP model comprising a concave quadratic objective function which is maximised with a set of linear constraints and mixed binary/continuous optimisation variables. Due to its convexity, the proposed mathematical model can be solved to global optimality through the branch-and-bound procedure. The solution procedure was further enhanced by developing special symmetry-breaking constraints to eliminate equivalent solutions. Other additional features such as minimum/maximum module size and balancing among modules can easily be incorporated in the proposed mathematical model. The applicability of the proposed optimisation-based approach

has been demonstrated by four network examples. Comparative results with other methodologies from the literature showed that the proposed methodology obtained superior performance while global optimum is guaranteed. The computational results also indicated that small and medium size network examples were solved successfully by employing the proposed model. The achievement of optimal solutions for medium/large networks has proved to be a very computationally demanding task.

The computational limitations of the proposed MIQP model described in Chapter 4 necessitated the development of an efficient solution methodology to tackle medium/large networks. For this reason, a two-stage optimisation-based solution procedure was developed in Chapter 5 to identify community structures with good quality modularity values. In the first stage, a good initial partition is selected from a number of candidate solutions after solving a MINLP model with different starting points. In the second stage, an iterative procedure was adopted to improve the solution obtained from the first stage. In order to overcome the resolution limit problem of modularity optimisation, an iterative process was applied to detect smaller modules hidden inside each community. Finally, the applicability of the proposed solution methodology has finally been demonstrated through a number of simulated and real network examples. Comparing with other existing methodologies in the literature, the proposed approach obtained more accurate community structures with higher modularity values. Moreover, the solution procedure was applied to each network community obtained without considering the inter-community links and the computational results showed finer community structures become successfully visible.

7.1.3 Motif Identification for DNA Sequence Data

The prediction of consensus sub sequences and motifs from a collection of DNA sequence data sets is the third objective of the thesis. In Chapter 6, an MILP model for DNA motif discovery was presented. Recent development of sequence analysis in the area of bioinformatics was initially introduced to bring readers this fascinating research area. A number of mathematical programming approaches for DNA sequence alignment and pattern discovery problems were also highlighted to demonstrate their suitability and flexibility to extract useful information from DNA sequences.

Given a collection of DNA sequences with variable lengths, the overall problem has been formulated as an MILP model, determining the content of the consensus sequence of all sequences and motifs for each sequence so that the total similarity scores between the consensus sub sequence and all motifs identified are maximised. The applicability of the mathematical programming framework was demonstrated via a number of DNA sequence sets selected from the literature. The resulting motifs identified through the proposed optimisation-based approach were compared with real motifs found via biological experiments and the predicted accuracy was calculated. Finally, the prediction performances of our approach were compared with other motif finding methodologies from the literature. The computational results clearly indicated that our method is competitive in terms of prediction accuracy when compared to other existing methodologies.

In this section, the key contributions of the thesis are summarised. This thesis covers three important data analysis topics using mathematical programming techniques including data classification, network community structure identification and DNA sequence analysis. Since data classification is one of major challenges in data mining and pattern recognition, a novel optimisation-based approach has been developed to tackle such problem using hyper-box classifiers. Secondly, optimisation models and solution algorithms have been proposed to identify communities in complex networks thus indicating a very promising research avenue to analyse network topological data through the use of mathematical programming. Finally, an optimisation-based framework has been developed to find DNA motifs and consensus sequences. Overall, the power of mathematical programming techniques to tackle various data analysis problems has been revealed through the competitiveness of the computational results when compared with methodologies from other research communities and the convenience of incorporating other additional features. Next, a number of research directions are recommended for further investigation.

7.2 Recommendations for Future Work

A number of promising future research directions related to the application of optimisation theory and mathematical programming techniques to data analysis are presented in this section. The aim of this section is to provide readers with some future insights in the area of optimisation-based data analysis and pattern discovery as

well as highlight a number of emerging research issues that could benefit from the mathematical modelling frameworks developed in the thesis. Next, we consider those future research issues in detail.

7.2.1 Data Classification through Mathematical Programming

In the first part of the thesis, an optimisation-based approach has been proposed to solve the data classification problem with two or multiple groups. Hyper-box classifiers have been used to discriminate training samples from different classes. The proposed optimisation-based framework successfully captured the patterns of training samples thus illustrating the applicability of mathematical programming techniques in data mining and machine learning. It is also widely accepted that no data classification approaches are superior to all other methods on every dataset (Lam and Moy, 2002; Adem and Gochet, 2006). The method which obtains the best prediction accuracy on one dataset may perform badly on the other. The training and prediction performances are determined by several issues such as the characteristics of datasets, the structures of classifiers and the parameter values specified. The proposed data classification-based framework in this thesis has been shown to particularly be suitable for samples with disjoint regions. It is noted that other approaches such as linear discriminant analysis (LDA) and neural networks (NN) are also achieved very good training and testing performances on some cases. Therefore, developing a hybrid approach combining the strengths of hyper-box classifiers and other discriminant functions is a very interesting direction to tackle classification problems with huge training samples and complex structures.

Hyper-box classifier proposed in Chapter 2 together with linear discriminant classification functions developed by Gehrlein (1986) and Sueyoshi (2006) opened a very promising research avenue to apply mathematical programming techniques to data classification. Recently, piecewise linear classifiers were developed (Glen, 2005; Ryoo, 2006) for only two class classification problems using MILP models. Comparing with the linear classifiers (Gehrlein, 1986; Sueyoshi, 2006), piecewise linear functions use more than one linear classifier to separate data samples with different class labels so as to improve training and testing accuracies. Moreover, Kim and Ryoo (2007a, 2007b) developed MILP models to separate two-class data samples using a finite number of nonlinear and nonconvex discriminant functions, which were

general enough to separate data samples with linear and/or nonlinear boundaries. Stimulated by the above research work, two directions are worth trying in the future. The first direction is to extend the piecewise linear classifier developed by Glen (2005) and Ryoo (2006) to multi-class data classification cases. The other avenue could be the generalisation of the proposed hyper-box classifiers into convex hulls so as to improve the training and testing performances.

7.2.2 Community Structure Identification of Complex Networks

Chapters 4 and 5 proposed optimisation-based approaches to identify community structures of complex networks. This research work demonstrated the power of mathematical programming techniques to investigate the topological characteristics of complex networks. Such frameworks were able to find communities of undirected and unweighted networks. Recently, the availability of various data made more networks directed and weighted so as to approach real properties of complex systems. For example, the weights appeared in social network edges may indicate the communication frequencies between the two nodes linked each other. The directionality of metabolic networks may reflect reversible/irreversible biochemical reactions. The extension of our optimisation-based approaches to general network models could be of crucially importance to identify communities and investigate the properties and functions of such modules more accurately.

The mathematical programming framework developed in Chapters 4 and 5 were capable of finding communities for networks with medium/large sizes. The computational results clearly indicated that the proposed framework identified network modules more accurately when compared to other competing methodologies. Recently, scientific and technological advances made a number of large-scale network data available. For instance, the network of coauthorships between scientists posting preprints on the Condensed Matter E-Print Archive has more than 30,000 nodes. The website connection of Notre Dame University involves more than 325,729 nodes. A number of algorithms were developed to tackle networks with that scale (Claust *et al.*, 2004; Pujol *et al.*, 2006; Raghavan *et al.*, 2007). Although they were efficient from computational point of view, all these approaches suffered from the random nature of the algorithms proposed or poor solution quality. Therefore, the development of more

efficient and accurate optimisation-based approaches for very large networks could be very beneficial to the research community.

The optimisation-based methodologies developed in Chapters 4 and 5 were able to find modular structures of networks with no more than 1200 nodes. Optimising the modularity metric of very large-scale networks become computationally challenging. In the near future, it is very interesting to propose some hybrid approaches for large networks. First, a number of efficient algorithms such as hierarchical clustering can be applied to partition the whole network into a number of small communities (i.e. 500-1000 clusters). Each small cluster can be treated as a pseudo node and are aggregated into larger modules using the proposed optimisation-based frameworks in this thesis. In addition, other modularity improvement techniques such as the fine tuning technique introduced by Newman (2006) can be used as the final postprocessing stage of the multi-stage hybrid approach so as to achieve accurate community structures of large networks.

It is noted that the mathematical models and solution approaches developed in Chapters 4 and 5 provided a general framework to identify network communities using mathematical programming techniques. Our approaches were general enough to incorporate other network partition qualification metric. Recently, Li *et al.* (2008) introduced a novel quantitative function for network community detection. The new metric, named as modularity density, was verified theoretically to be able to avoid the resolution limitations of maximisation of the modularity metric. Moreover, different parameters associated with the modularity density metric were also introduced in this work to enlarge the understanding of complex network topology. In the future, it is very promising to extend our methodologies to accurately identify modular structures and investigate other properties of complex networks.

In the area of network community structure identification described so far, the modules detected are assumed to be non-overlapped, which indicates each node must be allocated into only one distinct module. There are also great interests to detect overlapping/fuzzy communities since quite a few complex systems have modules that partially overlapped each other. Nodes being allocated into more then one community are generally considered as the bridge among distinct modules. Palla *et al.* (2005) proposed a novel approach to detect overlapping clusters in nature and society. Zhang

et al. (2007) applied c-means clustering algorithms to identify overlapping communities in social networks. Finally, Napusz *et al.* (2008) complemented and expanded the concept of overlapping communities and developed an algorithm to detect such structures. The proposed approach allowed each vertex of the graph to belong to multiple communities at the same time. The node-module allocation was determined by exact numerical membership degrees, even in the presence of uncertainty in the data being analysed. All these research work stimulated a very promising direction to apply mathematical programming techniques to identify overlapping/fuzzy community structures of real complex systems.

7.2.3 Finding DNA Motifs and Consensus Sequences using Mathematical Programming

In the sixth chapter of the thesis, a general mathematical programming model was developed to find the consensus sequences and motifs among a collection of DNA sequences. It should be noted that the proposed optimisation model was general to be applied to other sequence datasets. One future direction of this topic is to apply the resulting mathematical model to other forms of sequences.

The computational results shown in Chapter 6 indicated that the proposed MILP model is able to find motifs and consensus sequences with length of up to 18 positions. The identification of longer patterns was hindered by the demanding computational requirements. Recently, Kinsford *et al.* (2006) have developed a combinatorial optimisation approach combining a graph pruning techniques coupled with an integer linear programming (ILP) formulation for diverse motif finding problems. The computational results indicated such approach benefits from its efficiency. Therefore, the second avenue of this topic will be the improvement of our current model by incorporating tighter constraints and efficient heuristics to find longer motifs without sacrificing the solution quality.

It is also indicated that some DNA sequences involve gaps. The number of gap positions also has uncertainties. Li and Fu (2005) developed a mathematical programming approach to identify unknown binding sites with uncertain gaps in DNA sequences. In the near future, the mathematical framework proposed in Chapter 6 can

be extended to solve motif identification problems with variable motif lengths and uncertain gaps.

In this section, a number of research directions associated with optimisation-based data analysis have been introduced. All optimisation-based methodologies proposed in this thesis involve the analysis of static datasets. It is also widely known that the area of data analysis is developing in an unexpected pace especially the generation of dynamic datasets. For example, the operating conditions of process plants vary with time; dynamic evolutions of regulatory networks have been found on all levels in biology. Therefore, the development of methodologies to analyse dynamic datasets is of crucial importance to extract knowledge from dynamic systems. The success of the application of mathematical programming techniques to analyse static datasets shown in this thesis indicates a very promising research opportunity to tackle more complex data analysis tasks from optimisation point of view.

Bibliography

Adem. J. and Gochet, W. (2006) Mathematical programming based heuristics for improving LP-generated classifiers for the multiclass supervised classification problem. *Eur. J. Oper. Res.* **168**, 181-199.

Agarwal, G. and Kempe, D. (2007) Modularity-maximizing network communities via mathematical programming. *arXiv:0710.2533v1.*

Altschul, S. F., Gish, W. and Miller, W. (1990) Basic local alignment search tool. *J. Mol. Bio.* **215**, 403-410.

Arenas, A., Fernandez, A. and Gomez, S. (2007) Multiple resolution of the modular structure of complex networks, *arXiv: physics*/0703218

Autio, L., Juhola, M. and Laurikkala, J. (2007) On the neural network classification of medical data and an endeavour to balance non-uniform data sets with artificial data extension, *Comput. Biol. Med*, **37**, 388-397.

Bailey T. L. and Elkan, C. (1995) Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Mach. Learn.* **21,** 51-80.

Bal, H., Orkcu, H. H. and Celebioglu, S. (2006) An experimental comparison of the new goal programming and the linear programming approaches in the

two-group discriminant problems. *Comput. Ind. Eng.* **50**, 296-311.

Bajgier, S. M. and Hill, A. V. (1982) An experimental comparison of statistical and linear programming approaches to the discriminant problem. *Decision. Sci.* **13**, 604-618.

Balasundaram, B., Butenko. S. and Trukhanov S. (2005) Novel approaches for analyzing biological networks. *J. Global. Optim.* **10**, 23-29.

Barabasi, A. and Albert, R. (1999) Emergence of scaling in random networks. *Science*. **286**, 509-512.

Becerra-Fernandez, I., Zanakis, S. H. and Walczak, S. (2002) Knowledge discovery techniques for predicting country investment risk. *Comput. Ind. Eng.* **43**, 787-800.

Bishop C. M. (2006) Pattern Recognition and Machine Learning. *Springer.*

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M. and Hwang, D.-U. (2006) Complex networks: structures and dynamics, *Phys Rep.* **424**, 175-308.

Boettcher, S. and Percus, A. G. (2001) Optimization with extremal dynamics, *Phys. Rev. E.* **64**, 026114

Booker, L. B., Goldberg, D. E. and Holland, J. H. (1989) Classifier systems and genetic algorithms. *Artif. Intell.* **40**, 235-282.

Breiman, L., Friedman, J., Olshen, R. and Stone, C. J. (1984) *Classification and Regression Trees*. Chapman and Hall, New York.

Brooke, A., Kendrick, D., Meeraus, A. and Raman, R. *GAMS: A user's guide.* (GAMS development Corp. Washington, DC. 2003)

Burgard, A. P. and Maranas, C. D. (2001) Probing the performance limits of the Escherichia coli metabolic network subject to gene additions or deletions. *Biotechnol. Bioeng.* **74**, 364-375.

Castellano, C., Cecconi, F., Loreto, V., Parisi, D. And Radicchi, F. (2004) Self-contained algorithms to detect communities in networks, *Eur. Phys. J. B.* **38**, 311-319.

Chakravarty, A., Carlson, J. M., Khetani, R. S. and Gross, R. H. (2007) A novel ensemble learning method for de novo computational identification of DNA binding sites. *BMC Bioinformatics.* **8**, 249-263.

Chiang, L. H., Kotanchek, M. E. and Kordon, A. K. (2004) Fault diagnosis based on Fisher discriminant analysis and support vector machines. *Comput. Chem. Eng.* **28**, 1389-1401.

Clauset, A., Newman, M. E. J. and Moore, C. (2004) Finding community structure in very large networks, *Phys. Rev. E.* **70**, 066111.

Cortes, C. and Vapnik, V. (1995) Support vector network. *Mach. Learn.* **20**, 273-297.

Danon, L., Diaz-Guilera, A., Duch, J. and Arenas, A. (2005) Comparing community structure identification, *J. Stat. Mech.* P09008.

Dartnell, L., Simeonidis, E., Hubank, M., Tsoka, S., Bogle, I. D. L. and Papageorgriou, L. G. (2005) Robustness of p53 network and biological hackers, *FEBS. Lett.* **579**, 3037-3042.

Deutscher, D., Meilijson, I., Kupiec, M. and Ruppin, E. (2006) Multiple knockout analysis of genetic robustness in the yeast metabolic network *Nat. Genet.* **38**, 993-998.

Ding, C. H. Q., and Dubchak, I. (2001) Multi-class protein folding recognition using support vector machines and neural networks. *Bioinformatics.* **17**, 349-358.

Duch, J. and Arenas, A. (2005) Community detection in complex networks using extremal optimization, *Phys. Rev. E.* **72**, 027104.

Duda, R. O., Hart, P. E. and Strok, D. H. (2001) *Pattern. Recogn.* Second ed., Wiley–Interscience: New York.

Eckmann, J. P. and Moses, E. (2002) Curvature of co-links uncovers hidden thematic layers in the World Wide Web. *Proc. Natl. Acad. Sci. U.S.A.* **99**, 5825-5829.

Eiselt, H. A. And Sandblom, C. L. (2000) Integer programming and network models. Springer.

Erdos, P. and Rényi, A. (1959) On Random Graphs, *Publ. Math,* **6**. 290-297.

Erdos, P. and Rényi, A. (1960) On the Evolution of Random Graphs, *Publ. Math. Inst. Hungar. Acad. Sci.* **5**, 17-61.

Erdos, P. and Rényi, A. (1961) On the Strength of Connectedness of a Random Graph, *Acta Math. Sci. Hung.* **12**, 261-267.

Estrada, E. (2006) Protein bipartivity and essentiality in the yeast protein-protein interaction network. *J. Protein. Res.* **5**, 2177-2184.

Faloutsos, M., Faloutsos, P. and Faloutsos, C. (1999) On power-law relationships of the Internet topology, *Comp. Commun. Rev.* **29**, 251-262.

Fisher, R. (1936) The use of multiple measurements in taxonomic problems. *Ann. Eugenics,* **7**, 179-188.

Fisher, D. (1996) Iterative optimization and simplification of hierarchical clusterings , *J. Artif. Intell. Res.* **4**, 147-179.

Flake, G. W., Lawrence, S. R., Giles C. L. and. Coetzee, F. M. (2002) Self-organization and identification of web communities, *IEEE. Comput.* **35**, 66-71.

Flory, P. J. (1941), Molecular Size Distribution in Three Dimensional Polymers. I. Gelation, *J. Amer. Chem. Soc.* **63**, 3083-3090.

Floudas, C. A. (1995) *Nonlinear and Mixed-Integer Optimisation*; Oxford University Press: New York

Fortunato, S. and Barthelemy, M. (2007) Resolution limit in community detection, *Proc. Natl. Acad. Sci. U.S.A.* **104**, 36-41.

Frank, E; Hall, M; Holmes, G., Kirkby, R., Rfahringer, B., witten, I. H. and Trigg, L. (2005) Weka – A machine learning workbench for Data Mining. *Data Mining and Knowledge Discovery Handbook.* 1305-1314.

Freed, N. and Glover, F. (1981a) Simple but powerful goal programming models for discriminant problems. *Eur. J. Oper. Res. 7*, 44–60.

Freed, N. and Glover, F. (1981b) A linear programming approach to the discriminant problem. *Decision Sci.* **12**, 68-74.

Freed, N. and Glover, F. (1986) Evaluating alternative linear programming models to solve the 2-group discriminant problem. *Decision Sci.* **17**, 151-162.

Frith, M. C., Hansen U., Spouge J. L. and Weng, Z. (2004) Finding functional sequence elements by multiple local alignment. *Nucleic Acids Res.* **32,** 189-200.

Funahashi, K. I. (1989) On the approximate realization of continuous mappings by networks, *Neural. Networ.* **2**, 183-192.

Gabrys, B. and Bargiela, A. (2000) General fuzzy min-max neural network for clustering and classification. *IEEE. T. Neural. Networ.* **11**, 769-783.

Gallagher, R., Lee, E. K. and Patterson, D.A. (1997) Constrained discriminant analysis via 0/1 mixed integer programming. *Ann. Oper. Res.* **74**, 65-88.

Garey, M. R. and Johnson, D. S. (1979) *Computers and intractability, A Guide to the theory of NP-completeness.* W. H. Freeman, San Francisco.

Gehrlein, W. V. (1986) General mathematical programming formulations for

the statistical classification problem. *Oper. Res. Lett.* **5**, 299-304.

Girvan, M. and Newman, M. E. J. (2002) Community structure in social and biological networks, P*roc. Natl. Acad. Sci. U.S.A.* **99**, 7821-7826.

Gjerdrum, J., Shah, N. and Papageorgiou, L. G. (2001) Transfer prices for multi-enterprise supply chain optimization. *Ind. Eng. Chem. Res.* **40**, 1650-1660.

Glen, J. J. (1999) Integer programming methods for normalisation and variable selection in mathematical programming discriminant analysis models. *J. Oper. Res. Soc.* **50**, 1043-1053.

Glen, J. J. (2001) Classification accuracy in discriminant analysis: a mixed integer programming approach. *J. Oper. Res. Soc.* **52**, 328-339.

Glen, J. J. (2005) Mathematical programming models for piecewise-linear discriminant analysis. *J. Oper. Res. Soc.* **56**, 331-341.

Glen, J. J. (2006) A comparison of standard and two-stage mathematical programming discriminant analysis methods. *Eur. J. Oper. Res.* **171**, 496-515.

Gleiser, P. M. and Danon, L. (2003) Community structure in jazz. *Adv. Complex. Syst.* **6**, 565-573.

Glover, F. (1990) Improved linear programming models for discriminant analysis. *Decision. Sci.* **21**, 771-785.

Guh, R. S. (2005) A hybrid learning-based model for on-line detection and analysis of control chart patterns. *Comput. Ind. Eng.* **49**, 35-62.

Guimera, R., Danon, L., Guilera, A. D., Giralt, F. and Arenas, A. (2003) Self-similar community structure in a network of human interactions, *Phys. Rev. E.* **68**, 065103.

Guimera, R. and Amaral, L. A. N. (2005) Functional cartography of complex

metabolic networks, *Nature*. **433**, 895-900.

Gustafsson, M., Hornquist, M. and Lombardi, A. (2006) Comparison and validation of community structures in complex networks, *Physica A*. **367**, 559-576.

Hand, D. J. (1997) Construction and Assessment of classification rules. Chichester, England: Wiley.

Hansen, P. and Jaumard, B. (1997) Cluster analysis and mathematical programming. *Math. Program*. **79**, 191-215.

Haughton, D., Deichmann, J., Eshghi, A., Sayek, S., Teebagy, N. and Topi, H. (2003) A Review of Software Packages for Data Mining. *Am. Stat*. **57**, 290-309.

Hertz, G. Z., Hartzell, G. W. and Stormo, G. D (1990) Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Comput. Appl. Biosci*. **6**, 81-92

Hornik, K., Stinchcombe, M. and White, H. (1989) Multilayer feedforward networks are universal approximators. *Neural. Networ*. **2**, 359-366.

Holme, P., Huss, M. and Jeong, H. (2003) Subnetwork hierarchies of biochemical pathways, *Bioinformatics*. **19**, 532-538.

Hu, J., Li, B. and Kihrar, D. (2005) Limitations and potentioals of current motif discovery algorithms. *Nucleic. Acid. Res*. **33**, 4809-4913.

Hu, J. J., Yang, Y. F. D. and Kihara, D. (2006) EMD: an ensemble algorithm for discovering regulatory motifs in DNA sequences. *BMC Bioinformatics*. **7**, 342-354.

Iannarilli, F. J. and Rubin, P. A. (2003) Feature selection for multiclass discrimination via mixed-integer linear programming. *IEEE. T. Pattern. Anal*, *25*, 779-783.

Ilog, ILOG CPLEX 10.0 User's Manual. (2006).

Jain, B. A. and. Nag, B. N. (1995) Synthetic neural network models for pricing initial public offerings. *Decision Sci.* **26**, 283–302.

Je, M., Kim D. and Bang, S. Y. (2003) Human face detection in digital video using SVM ensemble. *Neural. Process. Lett.* **17**, 239-252.

Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N. and Barabasi, A. L. (2000) The large scale organizatin of metabolic networks. Nature, **407**, 651-654.

Kaufman, L. (1999) Solving the quadratic programming problems arising in support vector classification. In: B. Scholkopf, C. Burges and A. Smola (Eds.), *Advances in Kernel Methods*: *Support Vector Learning*. MIT Press, Cambridge, MA, USA, 147-167.

Kernighan, B.W. and Lin, S. (1970) An efficient heuristic procedure for partitioning graphs, *Bell Syst. Tech. J.* **49**, 291-307.

Kim, K. and Ryoo, H. S. (2007a) Data separation via a finite number of discriminant functions: A global optimization approach. *App. Math. Comput.* **190**, 476-489.

Kim, K. and Ryoo, H. S. (2007b) Nonlinear separation of data via mixed 0–1 integer and linear programming. *App. Math. Comput.* **193**, 183-196.

Kingsford, C., Zaslavsky, E. and Singh, M. (2006) A compact mathematical programming formulation for DNA motif finding. *Lect. Notes. Comput. Sci.* **4009**, 233-245.

Kirkpatrick, S., Gelatt C. D. and Vecchi, M. P. (1983) Optimization by simulated annealing. *Science*. 220, 671-680.

Klein, G. and Aronson, J. E. (1991) Optimal Clusters: a model and methods, *Nav. Res. Log*. **38**, 447-461.

Koonce, D. A., Fang, C. H., and Tsai, S. C. (1997) A data mining tool for learning from manufacturing systems. *Comput. Ind. Eng.* **33**, 27-30.

Knuth, D. E. (1993) *The Stanford graphbase: a platform for combinatorial computing*, Addison-Wesley, Reading, MA.

Kumpula, J. M., Saramaki, J., Kaski, K. and Kertesz, J. (2007) Limited resolution in complex network community detection with Potts model approach, *Eur. Phys. J. B.* **56**, 41-45.

Lam, K. F. and Moy, J. W. (1996) Improved linear programming formulations for the multi-group discriminant problem. *J. Oper. Res. Soc.* **47**, 1526-1529.

Lam, K. F. and Moy, J. W. (1997) An experimental comparison of some recently developed linear programming approaches to the discriminant problem. *Comput. Oper. Res.* **24**, 593-599.

Lam, K. F. and Moy, J. W. (2002) Combining discriminant methods in solving classification problems in two-group discriminant analysis. *Eur. J. Oper. Res.* **138**, 294-301.

Larsson, E., Lindahl, P. and Mostad, P. (2007) HeliCis: a DNA motif discovery tool for colocalized motif pairs with periodic spacing. *BMC Bioinformatics.* **8,** 418-426.

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu J. S., Neuwald, A. F. and Wootton, J. C (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignments. *Science.* **262,** 208-214.

Lawrence, C. E. and Reilly, A. A. (1990) An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins.* **7,** 41-51.

Lee, E. K., Easton, T. and Gupta, K. (2006) Novel evolutionary models and applications to sequence alignment problems. *Ann. Oper. Res.* **148**, 167-187.

Lee, E. K. and Wu, T. L. (2007) Classification and disease prediction via mathematical programming. *Data Mining, Systems Analysis and Optimization in Biomedicine.* **953,** 1-42.

Li, H. L. And Fu, C. J. (2005) A linear programming approach for identifying a consensus sequence on DNA sequences. *Bioinformatics.* **21**, 1838-1845.

Li, M., Ma, B. and Wang, L. (2002) Finding similar regions in many strings. *J. Comput. Syst. Sci.* **65**, 73–96.

Li, Z., Zhang, S., Wang, R. S., Zhang, X. S. and Chen, L. (2008) Quantitative function for community detection. *Phys. Rev. E.* **77**, 036109.

Lin, X. X., Floudas, C. A., Wang, Y. and Broach, J. R. (2003) Theoretical and computational studies of the glucose signalling pathways in yeast using global gene expression data. *Biotechnol. Bioeng.* **84**, 864-886.

Liu J. S., Neuwald, A. F. and Lawrence, C. E. (1995) Bayesian models for multiple local sequence alignment and Gibbs sampling strategies. *J. Amer. Statist. Assoc.* **90,** 1156-1170.

Lorena, A. C. and de Carvalho, A. C. P. L. F. (2007) Protein cellular localization prediction with support vector machines and decision trees. *Comput. Biol. Med.* **37**, 115-125.

Lusseau, D. (2003) The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations – Can geographic isolation explain this unique trait? *Behav. Ecol. Sociobiol.* **54**, 396-405.

Mangasarian, O. L. (1997) Mathematical programming in data mining. *Data. Min. Knowl. Disc.* **1,** 183-201.

Mandal, D. P. (1997) Partitioning of feature space for pattern classification. *Pattern Recognition.* **30**, 1971-1990.

Markham, I. S., Mathieu, R. G. and Wray, B. A. (1998) A rule induction approach for determining the number of kanbans in A just-in-time production system. *Comput. Ind. Eng.* **34,** 717-727.

Markham, I. S. and Ragsdale, C. T. (1995) Combining neural networks and statistical predictions to solve the classification problem in discriminant-analysis. *Decision. Sci.* **26**, 229–242.

McAllister, S. R., Rajgaria, R. and Floudas, C. A. (2007) Global pairwise sequence alignment through mixed-integer linear programming: a template-free approach. *Optim. Method. Softw.* 22, 127-144.

McLachlan, G. L. (1992) Discriminant Analysis and Statistical Pattern Recognition. Wiley, New York.

Medus, A., Acuna, G. and Dorso, C. O. (2005) Detection of community structures in networks via global optimization, *Physica A.* **358**, 593-604.

Meneses, C. N., Oliveira C. A. S. and Pardalos, P. M. (2004) Optimal solutions for the closest-string problem via integer programming, *Informs. J. Comput.* **16**, 419-429.

Meneses, C. N., Oliveira C. A. S. and Pardalos, P. M, (2005) Optimization techniques for string selection and comparison problems in genomics. *IEEE. Med. Bio. Mag.* **24**. 81-87.

Mering, C. V., Zdobnov, E. M., Tsoka, S., Ciccarelli, F. D., Pereira-Leal, J. B., Ouzounis C. A. and Bork, P. (2003) Genome evolution reveals biochemical networks and functional modules, *Proc. Natl. Acad. Sci. U.S.A.* **100**, 15428-15433.

Milgram, S. (1967) The small world problem, *Psy. Today.* **1**, 60-67.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D. and Alon, U. (2002) Network motifs: simple building blocks of complex networks. *Science.*

298, 824-827.

Napusz, T., Petroczi, A., Negyessy, L. and Bazso, F. (2008) Fuzzy communities and the concept of bridgeness in complex networks. *Phys. Rev. E.* **77**, 016107

Nath, R. and Jones, T. W. (1988) A variable selection criterion in the linear programming approaches to discriminant analysis. *Decision. Sci.* **19**, 554-563.

Navia-Vazquez, A. (2007) Compact multi-class support vector machine *Neurocomputing*. **71.** 400-405.

Newman, M. E. J. (2003) The structure and function of complex networks. *SIAM Rev.* **45**, 167-256.

Newman, M. E. J. (2004) Detecting community structure in networks, *Eur. Phys. J. B.* **38**, 321-330.

Newman, M. E. J. and Girvan, M. (2004) Finding and evaluating community structure in networks, *Phys. Rev. E.* **69**, 026113.

Newman, M. E. J. (2004) Fast algorithm for detecting community structure in networks, *Phys. Rev. E.* **69**, 066133

Newman, M. E. J. (2006) Modularity and community structure in networks, *Proc. Natl. Acad. Sci. U.S.A.* **103**, 8577-8582.

Palla, G., Derenyi, I., Farkas, I. and Vicsek, T. (2005) Uncovering the overlapping community structure of complex networks in nature and society. *Nature*. **435**, 814-818.

Papageorgiou, L. G. and Rotstein, G. E. (1998) Continuous-domain mathematical models for optimal process plant layout. *Ind. Eng. Chem. Res.* **37**, 3631-3639.

Paul, G., Sreenivasan, S., Havlin, S. and Eugene, S. H. (2006) Optimization of

network robustness to random breakdowns *Physica A.* **370**, 854-862.

Peng, S. H. (2003) Molecular classification of cancer types from microarray data using the combination of genetic algorithms and support vector machines. *Febs. Lett.* **555,** 358-362.

Platt, J. C. (1999) Fast training of support vector machines using sequential minimum optimization. In: Scholkopf B, Burges C and Smola A (Eds.), *Advances in Kernel Methods: Support Vector Learning.* MIT Press: Cambridge, MA, USA, 185-208.

Pothen, A., Simon, H. and Liou, H. P. (1990) Partitioning sparse matrices with eigenvectors of graphs, *SIAM. J. Matrix. Anal. A.* **11**, 430-452.

Pujol J. M., Bejar, J. and Delgado, J. (2006) Clustering algorithm for determining community structure in large networks. *Phys. Rev. E.* **74**, 016107.

Purintrapiban, U. and Kachitvichyanukul, V. (2003) Detecting patterns in process data with fractal dimension. *Comput. Ind. Eng.* ***45***, 653-667.

Quinlan, J. R. (1986) Induction to decision trees. *Mach. Learn.* **1**, 81-106.

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V. and Parisi, D. (2004) Defining and identifying communities in networks, *Proc. Natl. Acad. Sci. U.S.A.* **101**, 2658-2663.

Raghavan, U. N., Albert, R. and Kumara, S. (2007) Near linear algorithm to detect community structures in large-scale networks. *Phys. Rev. E.* **77**, 036106.

Ramnarayan, K., Bohr, H. G. and Jalkanen, K. J. (2008) Classification of protein fold classes by knot theory and prediction of folds by neural networks: A combined theoretical and experimental approach. *Theor. Chem. Acc.* **119**, 265-274.

Rapoport, A. (1951) Nets with distance bias, *Bull. Math. Biophy.* **13**, 85-91.

Rapoport, A. (1953) Spread of information through a population with socio-structural bias: I. Assumption of transitivity, *Bull. Math. Biophy.* **15,** 523-533.

Rapoport, A. (1957) Contribution to the theory of random and biased nets. *Bull. Math. Biophy.* **19,** 257-277.

Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N. and Barabasi, A. L. (2002) Hierarchical Organization of Modularity in Metabolic Networks. *Science.* **297**, 1551-1555.

Redhead, E. and Bailey T. L. (2007) Discriminative motif discovery in DNA and protein sequences using the DEME algorithm. *BMC Bioinformatics.* **8**, 385-403.

Reichardt, J. and Bornholdt, S. (2004) Detecting fuzzy community structures in complex networks with a Potts model. *Phys. Rev. Lett.* **93**, 218701.

Rives A. W. and Galitski, T. (2003) Modular organization of cellular networks, *Proc. Natl. Acad. Sci. U.S.A.* **100**, 1128-1133.

Roth F. P., Hughes J. D., Estep P. W. and Church, G. M. (1998) Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nat. Biotechnol.* **16,** 939-945.

Ryoo, H. S. (2006) Pattern classification by concurrently determined piecewise linear and convex discriminant functions. *Comput. Ind. Eng.* **51**, 79-89.

Sargent, R. W. H. (1977) The decomposition of systems of procedures and algebraic equations. *In Proc. Biennial Conference on Numerical Mathematics.* Watson, G.A. (ed.), Springer Verlag, Dundee.

Sharpe, P. K. and Glover, R. P. (1999) Efficient GA based techniques for classification. *Appl. Intell.* **11,** 277-284.

Shen-Orr S. S, Milo, R, Mangan, S and Alon, U. (2002) Network motifs in the

transcriptional regulation network of Escherichia coli. *Nat. Genet.* **31**:64-68.

Shin, H. J., Eom, D. H. and Kim, S. S. (2005) One-class support vector machines-an application in machine fault detection and classification. *Comput. Ind. Eng.* **48**, 395-408.

Shivakumar, K. and Narasimhan, S. (2002) A robust and efficient NLP formulation using graph theoretic principles for synthesis of heat exchanger networks. *Comput. Chem. Eng.* **26**, 1517-1532.

Simpson, P. K. (1992) Fuzzy min-max neural networks. 1. Classification. *IEEE T. Neural. Networ.* **3**, 776-786.

Son, S. W., Jeong, H. and Noh, J. D. (2006) Random field Ising model and community structure in complex networks, *Eur. Phys. J. B.* **50**, 431-437.

Stam, A. and Joachimsthaler, E. A. (1990) A comparison of a robust mixed-integer approach to existing methods for establishing classification rules for the discriminant problem. *Eur. J. Oper. Res.* **46**, 113-122.

Stam, A. and Ragsdale, C. T. (1992) On the classification gap in mathematical-programming-based approaches to the discriminant problem. *Nav. Res. Log.* **39**, 545-559.

Stormo, G. D. and Heatzell, G. W. (1989) Identifying protein-binding sites from unaligned DNA fragments. *Proc. Natl. Acad. Sci. U.S.A.* **86**, 1183-1187.

Sueyoshi, T. (2001) Extended DEA-discriminant analysis. *Eur. J. Oper. Res.* **131**, 324-351.

Sueyoshi, T. (2004) Mixed integer programming approach of extended DEA-discriminant analysis. *Eur. J. Oper. Res.* **152**, 45-55.

Sueyoshi, T. (2006) DEA-discriminant analysis: methodological comparison among eight discriminant analysis approaches. *Eur. J. Oper. Res.* **169**, 247-272.

Tarca, L. A., Grandjean, B. P. A. and Larachi, F. (2004) Designing supervised classifiers for multiphase flow data classification. *Chem. Eng. Sci.* **59**, 3303-3313.

Tatusova, T. A. and Madden, T. L. (1999) BLAST 2 SEQUENCES, a new tool for comparing protein and nucleotide sequences, *Fems. Microbiol. Lett.* **174**, 247-250.

Thompson, J. D., Higgins, D. G. and Gibson, T. J (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22,** 4673-4780.

Tompa, M., Li, N., Bailey, T. L., Church, G. M., Moor, B. D., Eskin, E., Favorov, A. V., Frith, M. C., Fu, Y., Kent, W. J., Makeev, V. J., Mironov, A. A., Noble, W. S., Pavesi, G., Pesole, G., Régnier, M., Simonis, N., Sinha, S., Thijs, G., Helden, J. V., Vandenbogaert, M., Weng, Z., Workman, C., Ye, C. and Zhu, Z. (2005) Assessing computational tools for the discovery of transcription factor binding sites. *Nat. Biotechnol.* **23**, 137-144.

Trafalis, T. B., Oladunni, O. and Papavassiliou, D. V. (2005) Two-phase flow regime identification with a multiclassification support vector machine (SVM) model. *Ind. Eng. Chem. Res.* **44**, 4414-4426.

Tsiakis, P., Shah, N. and Pantelides, C. C. (2001) Design of multi-echelon supply chain networks under demand uncertainty. *Ind. Eng. Chem. Res.* **40**, 3585-3604.

Turkay, M., Uney, F. and Yilmaz, O. (2005) Prediction of folding types of protein using mixed integer linear programming. *Proceedings of 15th European Symposium on Computer-aided Process Engineering*, L. Puigjaner and A. Espuna (Eds.) 523-528.

Uney, F. and Turkay, M. (2006) A mixed-integer programming approach to

multi-class data classification problem. *Eur. J. Oper. Res.* **173**, 910-920.

Varma, A. and Palsson, B. O. (1993) Metabolic capabilities of Escherichia coli: II. Optimal growth patterns. *J. Theor. Biol.* **165**, 503-522.

Watts, D. J. (1999) Small worlds: the dynamics of networks between order and randomness Princeton University Press , Princeton.

We, J. M. (2002) Natural discriminant analysis using interactive potts models. *Neural. Comput.* **14**, 689–713.

Williams, H. P. (1999) Model building in mathematical programming. *John Wiley*, New York, USA.

Wilson, J. M. (1996) Integer programming formulations of statistical classification problems. *Omega.* **24**, 681-688.

Wu, F. and Huberman, B. A. (2004) Finding communities in linear time: a physics approach, *Eur. Phys. J. B.* **38**, 331-338.

Xu, G., Tsoka, S. and Papageorgiou, L. G. (2007) Finding Community Structures in Complex Networks using Mixed Integer Optimisation, *Euro. Phys. J. B.* **60**, 231-239.

Yajima, Y. (2005) Linear programming approaches for multicategory support vector machines. *Eur. J. Oper. Res.* **162**, 514-531.

Yuksektepe, F. U., Yilmaz, O. and Turkay, M. (2008) Prediction of secondary structures of proteins using a two-stage method. *Comput. Chem. Eng.* **32,** 78-88.

Zachary, W. W. (1977) Information-flow model for conflict and fission in small-groups, *J. Anthropol. Res.* **33**, 452-473.

Zamora, J. M. and Grossmann. I. E. (1998) A global MINLP optimization algorithm for the synthesis of heat exchanger networks with no stream splits

Comput. Chem. Eng. **22**, 367-384.

Zaslavsky, E. and Singh, M. (2006) A combinatorial optimization approach for diverse motif finding applications. *Algorithm Mol Biol.* **1**, 1-13.

Zhang, S., Wang, R. S. and Zhang, X. S. (2007) Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A*, **374**, 483-490.

Zhou, E. and Khotanzad, A. (2007) Fuzzy classifier design using genetic algorithms. *Pattern. Rrecogn.* **40**, 3401-3414.

Zhong, P. and Fukushima, M. (2006) A new multi-class support vector algorithm. *Optim, Method. Softw.* **21,** 359-372.

Zopounidis, C. and Doumpos, M. (2002) Multi-group discrimination using multi-criteria analysis: Illustrations from the field of finance. *Euro. J. Oper Res.* **139**, 371-389.

Refereed Research Papers from This Thesis

Book Chapter

- Xu, G. and Papageorgiou, L. G. (2006) An MILP model for multiclass data classification, in Bogle, I. D. L., Zilinskas, J. (ed.) Computer Aided Methods in Optimal Design and Operation. *Computers and Operations Research series.* World Scientific, 15-20.

Journal Papers

- Patsiatzis, D. I., Xu, G. and Papageorgiou, L. G. (2005) Layout aspects of pipeless batch plants. *Ind. Eng. Chem. Res.* **44**, 5672-5679.
- Xu, G. and Papageorgiou, L.G. (2007). A construction-based approach to process plant layout using mixed-integer optimization, *Ind. Eng. Chem. Res.* **46**, 351-358.
- Xu, G., Tsoka, S. and Papageorgiou, L. G. (2007) Finding community structures in complex networks using mixed integer optimisation, *Euro. Phys. J. B*, **60**, 231-239.
- Xu, G. and Papageorgiou, L. G. (2007) A mixed integer optimisation model for data classification. Submitted to *Comput. Ind. Eng.*
- Xu, G., Tsoka, S. and Papageorgiou, L. G. Optimisation methodologies for module detection and resolution limitations in complex networks. *In Preparation.*

Conference Papers

- Xu, G. and Papageorgiou, L. G. (2006) An MILP model for multi-class data classification. *Optimal Process Design Workshop.* Vilnius, Lithuania.

- Xu, G. and Papageorgiou, L. G. (2006) An optimisation-based framework for data classification with multiple groups. *Proceedings of 21st European Conference on Operational Research.* Reykjavik, Iceland.

- Xu, G. and Papageorgiou, L. G. (2007) An iterative solution approach to process Plant layout using mixed integer optimisation. *Proceedings of 17th European Symposium on Computer-aided Process Engineering.* Bucharest, Romania.

- Xu, G., Shao, N. and Papageorgiou, L. G. (2007) A mixed integer optimisation approach for data classification with multiple groups. *Proceedings of 17th European Symposium on Computer-aided Process Engineering.* Bucharest, Romania.

- Xu, G. and Papageorgiou, L. G. (2008) Evaluating credit risks of business entities using mixed integer optimisation. *Proceedings of 5th International Conference on Computational Management Science.* London, UK.

Appendix

List of Proteins in p53 Network

node 1	14-3-3
node 2	Abl
node 3	AP2
node 4	APC
node 5	ATM
node 6	Bax
node 7	BRCA1
node 8	C-EBP
node 9	Casp3
node 10	Cdc25A
node 11	Cdc25C
node 12	Cdk1
node 13	Cdk4-6
node 14	Cdk7
node 15	Chk1
node 16	CK1d-k
node 17	CK2
node 18	Cks1
node 19	Crk
node 20	CSB
node 21	C-TAK1
node 22	CycA
node 23	CycB
node 24	CycD
node 25	CycE
node 26	CycH
node 27	DMP1
node 28	DNA-PK
node 29	DP1-2
node 30	Dpase_a
node 31	Dpase_b
node 32	Dpase_d
node 33	dsDNA
node 34	E2F1-2-3
node 35	E2F4
node 36	E2F5
node 37	E2F6
node 38	E-cad
node 39	ERCC1
node 40	FEN-1
node 41	Fos
node 42	Gadd45
node 43	HBP1
node 44	HDAC1
node 45	Histones
node 46	HMG
node 47	HR23B
node 48	JNK
node 49	Jun
node 50	Karp-1
node 51	Ku70
node 52	Ku80
node 53	Ligase_1
node 54	Ligase_3
node 55	MAPK
node 56	Max
node 57	Mdm2
node 58	Myc
node 59	Myt1
node 60	p107
node 61	p130
node 62	p16
node 63	p19ARF
node 64	p21
node 65	p27
node 66	p300
node 67	p36MAT1
node 68	p53
node 69	rpa
node 70	p57
node 71	p68
node 72	PARP
node 73	Paxillin
node 74	pCAF
node 75	PCNA
node 76	PKC
node 77	Plk1
node 78	pRb
node 79	cdk2
node 80	Rad51
node 81	Rad52
node 82	Raf1
node 83	Ras
node 84	Rep_fork
node 85	RF-C
node 86	RHA
node 87	Rpase_2
node 88	Skp1
node 89	Skp2
node 90	SL1
node 91	Sp1
node 92	ssb
node 93	ssDNA
node 94	TAFII250
node 95	TBP
node 96	TFIIH
node 97	U-glyc
node 98	Wee1
node 99	XPA
node 100	XPB
node 101	XPC
node 102	XPD
node 103	XPF
node 104	XRCC1