Optimization and Machine Learning Frameworks for Complex Network Analysis

Daehan Won

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

Reading Committee:

W. Art Chaovalitwongse, Chair

Thomas Grabowski

Archis Ghate

Adrian KC Lee

Onur Seref

Program Authorized to Offer Degree:
Industrial and Systems Engineering

University of Washington

Abstract

Optimization and Machine Learning Frameworks for Complex Network Analysis

Daehan Won

Chair of the Supervisory Committee:
Professor W. Art Chaovalitwongse
Industrial and Systems Engineering

Networks are all around us, and they may be connections of tangible objects in the Euclidean space such as electric power grids, the Internet, highways systems, etc. Among the wide range of areas in the network analysis, finding critical component in the large scale complex networks is one of the most challenging but fascinating problem in the network analysis. Analytical approaches of finding critical components have been widely studied and extensively used to investigate and provide meaningful characterizations of the intrinsic dynamics and properties of complex structures in networked systems.

The objective of this thesis is to build novel mathematical models for finding critical components and connectivity patterns in complex networks that may reveal hidden, yet insightful, information for the investigation of underlying dynamics of the networks.

In particular:

1. I propose mixed integer programming (MIP) models to seek k-Cardinality Tree (KCT) which address the finding critical components problem. I proposed seven variations of MIP models that are based on connected component constraints and subtour elimination constraints. Through the investigation of polyhedral structures and experimental results, the best performance model has been chosen and then we compared it with state of the art algorithm in the literature.
2. I expand our scope to find critical components in the labeled networks. I design two mathematical programming model to determine k-sized critical component including

the most informative edges to classify the networks. As a first step, we develop mixed integer programming (MIP) model for finding critical components in the networked data classification. Due to the computationally intractability on the large scaled data, I built a branch-and-cut algorithm based on the Benders decomposition.

3. I also build a mixed integer nonlinear programming (MINLP) model based on the support vector machine (SVM) formulation. Rather than solving this MINLP directly, an efficient iterative algorithm combining with multiple kernel learning is proposed.

To demonstrate the utility of the proposed models and solution approaches, synthetic networks and brain functional connectivity networks are used as case points in this thesis. Through the extensive experiments on both data sets, proposed approaches achieve impressive scalability and comparable or even better performance rather than the state-of-the-art methods. On human brain networks, the approaches are used to detect informative regions of interests (ROIs) and their connectivity patterns that may be useful in detecting people who are risk of developing neurological diseases.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The work in this thesis could not never have been accomplished without the aluable help, assistance, guidance, and efforts of a lot of people. First of all, I would like to render my heartfelt appreciation to my prinicipal supervisor, mentor, boss, and friend Dr. Chaovalitwongse for instilling in me the qualities of being a good engineer as well as scientist. His passion and brilliant advisory have been major driving forces through my graduate career at the University of Washington.

I would like to thank thesis committee members Dr. Grabowski, Dr. Ghate, Dr. Lee, and Dr. Seref for their guidance over the years. Also, I wish to thank members past and present of the TeamArt: Dr. Chou, Dr. Wang, Dr. Kampa, Danica, George, Amm, and Hasan. Dr.Chou and Dr.Wang helped me to start my new life at this university as a graduate student. I express my grateful gratitutde to them for their kind advice and crucial contribution. Without George Presnyakov, I would not start my studies with extremely complicated brain data. With his patient advice and brilliant efforts, my thesis could include the substantial researches on the brain networks. Dr. Kampa brough me to the new area: machine learning. He taugh me everything Without Hasan Manzour, this thesis would not have been so completed. I would like to thank his great knowledge on the math. modeling and valuable comments on the techical writings. Also, I wish to thank two former visiting scholars from China, Yulian and Sally.

Many thanks go in particular to Dr.Tsai, who is a great scientist and physician on the neurology at the Harborview Medical Center. Collaboration with him was a great research experience and he gave a beautiful insight on the neurology and brain science. Also, his valuable advices on the writing and research design were the veriest asset for the entire thesis.

I thank the members in the Integrate Brain Imaging Center (IBIC), UW Medicine. Par-

ticularly, I am much indebted to Liza L. Young for her support. She has been a tower of strength in IBIC. Also, I gratefully thanks to Dr. Madhyastha and Dr.Askren for their advice in the discussion and kind help in the brain research.

During the years on the U.S., without my sincere friend Donghyo Alex Min, I would not be survived. I would like to express my gratitude to him. Your friendship is irreplaceable. Also, I wish to thank Yooin Rho, who is the special heroine in the Ph.D. life, for providing unfailing support, continuous encouragement and putting up my craziness. Finally, this thesis would not be started without the unforgettable encouragement from Dr.Lee. Without his encouragement and helps, I would be kept in home and miss this great opportunity.

The last, and surely the most, I would like to express my sincere gratitude to my parents and brother, for their consistent love and support throughout my entire life. Without them, I am nothing. This thesis would not be possible without them.

Chapter 1

INTRODUCTION

Biological systems, chemical interactions, social networks, the Internet, and the neural system of the human brain are only a few examples of items composed by a large number of highly interconnected dynamic units. The first step toward capturing the global and local properties of connected systems is to represent them as general networks. In particular, as symbolic representations. Graphs have been commonly used as frameworks for modeling system components as nodes, with connections between components as edges [77, 176].

With the rapid development of technology, large-scale networks with complicated structural connectivity, which have been called "complex networks", have emerged in various applications, such as social networks, bioinformatics, and human brain connectivity. The explosive growth in the size of this data has brought substantial challenges, as well as opportunities for network science.

Analyzing large-scale networks gives rise to critical issues, such as data storage, scalability issues for analytical algorithms, and the difficulty of understanding the intrinsic characteristics of a network.

Despite these challenges, such complex networked data have also brought great opportunities for data analysis, such as the chance to reveal underlying characteristics of networks, the possibility of improving stability and general performance, and the discovery the hidden knowledge within this massive data.

In this thesis, I present novel frameworks for complex network analysis, while introducing the concept of **optimization frameworks** to understand the intrinsic dynamics and underlying structure of complex networks. Specifically, I have focused on determining the **critical components** of a single network and group-labeled networks and this has provided insight into the intrinsic dynamics and properties of complex structures in networked systems.

1.1 Researches on Networks

Basically, research on networks can be summarized into three categories: network optimization, the study of global attributes, and machine learning.

1.1.1 Network Optimization

Historically, the study of networks has been mainly the domain of graph theory. Since its birth in 1736, when the Swiss mathematician, Leonhard Euler, published the solution to the königsberg Bridge Problem (consisting of finding a round trip that traversed each of the bridges of the Prussian city of königsberg exactly once), graph theory has witnessed many exciting developments and has provided answers to a series of practical questions such as what the maximum flow per unit time is from source to sink in a network of pipe lines; how to color the regions of a map using the minimum number of colors so that neighboring regions receive different colors or how to assign n jobs to n people with maximum total utility [23].

Solving problems based on graph theory has been called *"network optimization"*. This research has focused on finding objective structures, establishing strategies, and deducing other solutions in terms of a problems characteristics by satisfying the given objectives and constraints of the problem.

Mathematical optimization (alternatively, optimization or mathematical programming) is the selection of a best solution with regard to some criteria from a set of available alternatives. As a branch of optimization, network optimization seeks to find the optimal solution of a problem when the problem structure is composed of network representation. Network optimization has been used to solve various practical network-structured optimization problems, such as the shortest path problem and problems involving assignments, max-flow, transportation, optimal trees, and the Traveling Salesman Problem, all of which constitute the most common class of optimization problem.

Aside from their wide usage in practice, methodologically, the ties between linear programming and combinatorial optimization can be traced to the representation of the constraint polyhedron as the convex hull of its extreme points. When a network is involved, however,

these ties become much stronger because the extreme points of the polyhedron are integer and represent solutions of combinatorial problems. Because of this structure and also because of their intuitive character, network models provide ideal vehicles for explaining many of the fundamental ideas which are present in common optimization problems [20].

One representative network optimization problem is the network flow problem. The network flow problem is one of the most important and most frequently encountered class of optimization problems. It arises naturally in the analysis and design of large systems, such as communications, logistics, and transportation. It can also be used to model important classes of combinatorial problems. e.g., job assignments, issues concerning matching and the Traveling Salesman Problem. Basically, the network flow problem consists of source and sink points, combined with several routes that connect these points and that are used to transfer the source to the sink. These routes contain intermediate transshipment points. Often, the source, sink, and transshipment points can be modeled by the nodes of a graph, and the routes can be modeled by the paths of the graph. Furthermore, there may be multiple types of commodities delivered via these routes. There are also some constraints concerning the characteristics of the route, such as capacity, and some cost and revenue issues associated with using particular routes. Such problems can be modeled as network optimization problems, whereby we try to find routes that involve the minimized cost or maximized revenue, while satisfying transshipment conditions from source to sink points.

The max-flow problem is based in the boundary of the network flow problem. Its objective is to move as much flow as possible from source to sink, while observing capacity constraints. More precisely, in solving this problem, we want to find a flow vector that makes the divergence of all nodes other than the source and sink equal to zero, while maximizing the divergence of the source. The max-flow problem arises in many practical situations, such as calculating the throughput of a highway system or a telecommunication network.

In addition to the flow conservation and bounds, there exist network optimization problems that involve identifying a subset of networks that satisfy a set of constraints with minimum or maximum total cost. Such a problem has been called a “network design problem”. In particular, there may be constraints that couple the flows of different edges, and there may also be integer constraints involving these flows. For example, the flow of the edge can

be either 0 or 1. There are popular combinatorial optimization problems like this, such as the Traveling Salesman Problem (TSP) [10]. This problem refers to a salesman who wants to find a minimum cost tour that visits each of N given cities exactly once and to then return to the city where he started. Intuitively, to convert this to a network flow problem, we associate a node with each city and we introduce an edge with an edge cost. The optimal tree problem is one of the examples of network design problems based on integer programming optimization.

The minimum spanning tree problem is the most representative optimal tree problem. As well as the minimum spanning tree problem, the k-cardinality tree problem can also be a network optimization problem [74]. This seeks to find the subtree of a given graph with exact k edges and minimum edge weights. There are a number of variations of optimal tree problems aside from these two. The Steiner Tree Problem involves finding a tree that includes the subset S of its nodes and has a minimum total weight of edges in the founded tree [98].

1.1.2 Global Attributes in Complex Network

Aside from network optimization, scientists have to cope with structural issues, such as characterizing the topology of complex architecture, revealing the unifying principles at the basis of real networks, and developing models that mimic the growth of a network and reproduce its underlying structural properties. To that end, much research has been done on network analysis and this has provided meaningful insight into networks and also information that may be applied to other areas.

In the 1990s, with the introduction of the Internet, it became possible for us to observe the dynamics and evolution of this large-scale network. This new opportunity generated fresh interest in analyzing common patterns in real networks. These studies led to the conclusion that different kinds of networks share some macroscopic statistical properties. For example, when studying the worldwide web, protein interaction, Facebook, and several other real networks, it has been noticed that all of them have a similar degree of distribution and a similar community structure [32].

The massive comparative analysis of networks from different areas has produced plenty of results. The first problem though is structural. The research on complex networks began with the effort to define new concepts and a determination to characterize the topology of networks. This has led to remarkable results involving the identification of a series of unifying principles and statistical characteristics common to most of the real networks.

The node degree is the first relevant result, which is the number of directed connections to other nodes (i.e., the number of neighborhood nodes). In a real complex network, the degree distribution, $P(k)$, is defined as the probability that a node selected uniformly at random has degree k and significantly deviates from the Poisson distribution expected for a random graph and, in many cases, exhibits a power law (scale-free) tail with an exponent r ranging from 2 to 3. Besides this, real networks are defined by node degree correlations, by having a relatively short path between any two nodes (e.g., a small world network property), and by specific motifs.

Since there is a pressing need to reflect *real world* networks accurately, these findings have initiated a revival of modeling which involves compensating for the lack derived from the limitations of graph theory approaches. It has been shown that engaged architecture offers important results for network robustness and responses to external attacks or perturbations. At the same time, it reveals the possibility of empirically studying the dynamical behavior of large assemblies of systems interacting via complex topologies.

This allows us to investigate the evidence that points to the important role played by network topology in determining the emergence of collective dynamical behavior, e.g., synchronization or in governing the main attributes of relevant processes that take place in complex networks, such as the spreading of epidemics, information, and rumors. Considering these emerging research interests in the world of complex network analysis, there are a number of review papers and books already in existence [182, 7, 142, 202, 35].

Differing from deterministic graph theory-based approaches, this work provides approximation, since it means translating the interaction between two dynamical units, which is usually dependent on time, space, and many more other details, into a simple binary variable: the existence of a link between two corresponding nodes. Although the graph representation step is based on the approximation scheme, it provides a simple, but still

powerful representation of the system.

Recently, the growing availability of databases and the optimized configuration of computing facilities have provided better machinery to explore the topological properties of several networks from the real world. This allows us to investigate the topology of interactions in various systems as diverse as communications, social networks, and biological systems. From these investigations, we have observed that most real networks are characterized by the same topological properties, such as, for instance, relatively small characteristic path lengths, high clustering coefficients, fat-tailed shapes in their degree distributions, degree correlations, and the presence of motifs and community structures, despite a difference of inheritance in these networks (detailed description of these characteristics will be presented in Chapter 2). All of these observations imply that real networks are quite far from the regular lattices and random graphs which have been used as the standard models in mathematical graph theory. Furthermore, this research has allowed us to understand evolutionary processes, such as the topology of a network, and the design processes of models that reflect the significant properties we have empirically observed. In Chapter 2, we will briefly discuss the modeling process. In this section, we will present the most important topological properties, such as the small world network and scale-free networks. More detailed descriptions are offered by [31].

The study of dynamical processes over real networks has revealed the existence of shortcuts, thus increasing communication speed between distant nodes. Lets suppose the presence of regular, hyper-cubic lattices H, where $dim(H) = d$. The average number of nodes ($=N$) one has to pass by so as to arrive at an arbitrarily selected node involves following the lattice size as $N^{1/d}$. Conversely, in most of the real networks, there is a relatively short path between any two nodes. The small world network property represents this feature and it is defined by an average shortest path length L that depends logarithmically on the network size N [202].

Furthermore, this is a mathematical property of some network models, such as, for instance, in random graphs. The small world network property of real networks is often associated with the presence of clustering coefficients, denoted by high values of clustering coefficients (see Chapter 2 for details). For this reason, [202] proposed to define a small world network

as a network having both a small L and a high clustering coefficient C.

In 1999, Albert et al. started to draw a map of the Internet and observed the surprising result that only a few pages possess the majority of links, whereas most pages are only sparsely connected. This was totally different from the conventional expectation that the degree distribution of nodes would be localized around an average value [17]. Based on the fact that they found the degree distribution was following a power law, such as $P(k) \sim Ak^{-r}$, they denoted this as a scale-free network.

Since the observations of Albert et al.[17], many researchers have empirically analyzed many real world networks with this scale-free property. As a result, r is the degree exponent and this ranges between 2 and 3 (precisely $1.9 \leq r \leq 2.5$). This factor is derived by counting the number of edges per node and plotting the results by increasing degrees within a chart with a log-log-scale (the slope of the resulting straight line is r).

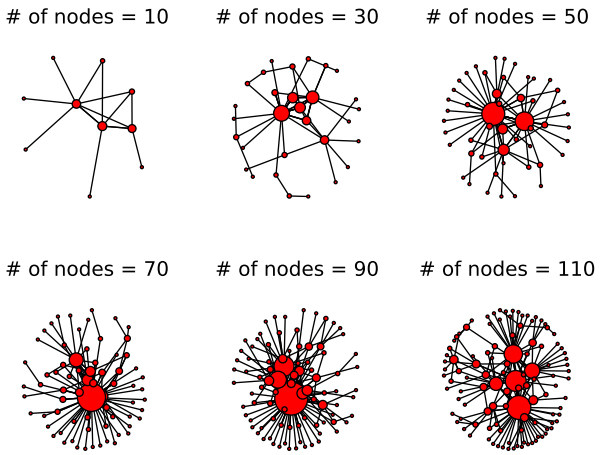


Figure 1.1: Scale free network generation process. The volume size of the node is directly proportional to the degree of the node

1.1.3 Machine Learning on the Networks

In the machine learning community, there has been a lot of interests in recent years in learning about networked data. Machine learning approaches to networks (*aka* graph mining) can be categorized into the following subgroups: (i) Network clustering (unsupervised learning)this is the task of grouping networks into clusters by means of similarity measures. Various ways have been investigated in order to represent the structural information of networks, , such as patterning common subnetworks [43], calculating the median of a set of networks, applying k-means clustering [94], and developing spectral feature vectors for network representation [128]. (ii) Network classification (supervised learning) corresponds to the task of predicting the label (=class) of a given network. This is similar to network optimization and standard analysis of networks, and it has a wide application range from chemical compound informatics (e.g., having compounds that are toxic or non-toxic as targets) and bio-informatics (e.g., classifying proteins into different families and classifying tissue samples), to telecommunication networks (e.g., classifying customers based on their conversation patterns) and social networks (e.g., classifying users based on their feeds on Twitter, Facebook, and so on) [102, 120, 72]. This learning model in itself does not differ from the existing classification models. Therefore, most proposed schemes have dealt with the extraction of features from the networked data.

One of the most popular extraction techniques is the use of kernels. These have been widely used due to their ability to represent data as a symmetric, positive semi-definite matrix which contains a pairwise similarity between all pairs of networks in the population. Using the kernel between networks was firstly proposed by Gärtner et al. [82] and this was based on random walks or cycles. Additionally, several topology-based kernels have been developed, e.g., shortest paths [33], subtrees[161], and cycles[95].

A common point of view concerning these kernels is that their features are composed of counts of label sequences produced by network traversal. Thus, the essential difference between the kernels lies in how networks are traversed and which methodologies are involved in calculating them.

In mathematics, it is possible to consider a kernel based on a more complicated structure.

However, considering all possible subnetworks is hard [82]. Recently, subgraph mining has come into the spotlight due to its high performance rate. In considering subnetworks as features, the basic idea is to mine frequent subnetworks and employ them as features for classification, e.g., with SVM [61]. [61] Scientists have conducted experiments on chemical compound data based on the assumption that substructure patterns are good indicators of belonging to one particular label. Popular methods for subnetwork pattern mining are gSpan [208] and FFSM [96]. However, all of them involve expensive computation since the number of frequent subnetworks is extremely huge, especially in large, complex networks.
Aside from the kernel method, global attributes also can be used as features for classification. We can construct a feature vector by means of calculating topological attributes [120]. This technique is underpinned by the premise that networks that belong to the same group should have similar topological and group attributes. The most prominent advantage of this technique is its easiness to implement and fast computation time.
In contrast to the above two methodologies based on an unweighted network which only considers binary connectivity, weights on edges can be represented as a feature vector by a concentration of weights. This method is especially useful for homogeneous structure networks with different weights, such as human brain connectivity networks [121]. Note that we are mainly focusing in this thesis on network classifications where all given networks have homogeneous structures with different edge weights.

1.2 Networks Applications

In this section, I provide a brief information of two representative networks which may involve large scale of networked data with complicated structures: sensor networks and human brain networks. Due to the structural complexity of the data, both of them have been regarded as complex networks [1, 162].

1.2.1 Sensor Network

Here, I present a brief information to the history and current state of the art with regard to sensor networks. The origins of the research on sensor networks(SN) (or wireless sensory network, WSN) can be traced back to the Distributed Sensor Networks (DSN) program at

the Defense Advanced Research Projects Agency (DARPA) at around 1980. Due to the high operation cost and large size of sensors in early SN, its applications were limited to specific usage.

Recent advances in computing, communication technology have caused a significant shift in SN researches and brought it closer to achieving the original vision. In the new wave of sensor network research, networking techniques and networked information processing suitable for highly dynamic ad-hoc environments and resource-constrained sensor nodes have been the focus. Further, the sensor nodes have been much smaller in size and much cheaper in price, and thus many new applications of sensor networks such as environment monitoring, vehicular sensor network and body sensor network have emerged.

Currently, SN has been viewed as one of the most important technologies for the 21st century. Countries such as China have involved SNs in their national strategic research programs [145].

Components of SN

Basically, a SN consists of spatially distributed sensor *nodes*. In a SN, each sensor node can perform some processing and sensing jobs independently. In Further, sensor nodes *communicate* with each other in order to forward their acquired information to a central processing unit or hub location. Basically, each sensor node consists of several hardware components: a radio transceiver, a processor, memories, power source and sensors.

- Processor: the functionality of a processor is to schedule tasks, process collected data, and control the functionality of other components
- Transceiver: a transceiver is addressing the communication of a sensor node. It includes a choice option of transitioning data types such as radio frequency, laser, infrared etc.
- Memory: a memory is composed of two type: in-chip flash memory and random access memory (RAM) of external memory.
- Power Source: due to the power consumption by various sensing activities, power can be stored in batteries or capacitors. Usually, battery have been preferred.

- Sensors: it is a hardware device that produces a *measurable* response signal to a change in a physical condition such as temperature, humidity and any specific status.

Network Functioning

A SN is a network consisting of numerous sensor nodes with sensing, wireless communications and computing capabilities. These sensor nodes are scattered in an unattended environment (i.e. sensing field) to sense the physical world. The sensed data can be collected by a few sink nodes which have accesses to infra-structured networks. Finally, end users (i.e., network operators) can remotely fetch the sensed data by accessing infra-structured networks. According to the Alyildiz et al. [6], basic protocol of SN is composed of following five layers: application, transport, network, data linkage, and physical. The physical is responsible for frequency selection, carrier frequency generation, signal detection, modulation and data encryption. The data linkage is responsible for the multiplexing of data streams, data frame detection, medium access and error control. The network layer takes care of routing the data supplied by the transport layer. The network layer design in WSNs must consider the power efficiency, data-centric communication, data aggregation, etc. The transportation layer helps to maintain the data flow and may be important if SNs are planned to be accessed through the Internet or other external networks. Depending on the sensing tasks, different types of application software can be set up and used on the application layer.

Applications

The original aim of SN development is usage on the military area. As the costs for sensor nodes and communication networks have been reduced, many other potential applications including those for practical purposes have emerged [6, 53].

- Environment monitoring: it can be used for animal tracking, forest surveillance, pollution level detection, and weather forecasting.
- Healthcare monitoring: SN can be embedded into a hospital or medical institution to track and monitor patients and all medical resources.
- Industrial Sensing: In a manufacturing (or processing) plant, equipment failure cause

unplanned downtime and it induces serious cost defects. A SN in the machine make it economically feasible to monitor the "malfunctioning" machines and to ensure safe operation.

- Traffic Control: SN have been used for traffic monitoring and controlling. At many roads, there are either overhead or buried sensors to detect vehicles and to control the traffic lights.
- Security: SN can be used for infrastructure security and preventing terrorism applications. Critical buildings and facilities (e.g, power plant) have to be protected from external invasions. SN consists of video, acoustic, and other sensor nodes can be deployed around these facilities.

1.2.2 Brain Networks

Neural assemblies (i.e., local connections of neurons) are considered to be largely distributed and linked to form a Web-like structure of the brain [195]. The resulting networks, which is sparse, is able to coordinate and integrate distributed brain activities in a neural process. The interaction of neuron organizations and neural functions was revealed out by [178]. The most important fact that is addressed whether the structure of neural connection is related to brain functions. Even though there exist a huge number of neurons and their communications in human brain cortex, the brain organization is constrained by optimizing principles of resource allocation and constraint minimization [110]. Especially, in the course of the brain evolution, the number of neurons has increased, whereas their connections have become indirect.

Connectivity would lead to minimized cost of interconnection between neural regions, thus yielding an efficient communication. Since a brain has the complicated structure, there has been an explosion of interest in exploring and understanding the intrinsic brain networks. As the intrinsic networks vary over time, neurons dynamics change continuously. This plasticity makes neurons changing their connection continuously or establishing new connections according to underlying needs. This dynamic connection between different cortical regions would emerge the global dynamics. According to recent studies, a functional connection

(time dependent) and not only a simultaneous activation of the population of neurons, is a fundamental mechanism for the large scale network integration problem. By means of this mechanism, local population of synchronized neural assemblies are integrated functionally at various spatial and temporal scales as a comprehensive neural process. Thus, human brain is an example of a complex system that can be described as an active network of neuronal elements (i.e., neurons) in convert with a complicated structural network [162].

Recently, quantitative approaches based on a graph theory have been widely applied to analyze brain networks to investigate their topological structures and underlying dynamics. So as to explore such a complex network, the structure of the brain and the *interaction* (i.e., *connectivity*) of brain functions need to be clearly defined. There exist various methods to observe the anatomy of the brain and there are also variant ways to determine brain connectivity.

Brain Connectivity

Generally, the brain connectivity is referred as a pattern and the patterns are categorized as three types: anatomical links (anatomical connectivity), statistical dependencies (functional connectivity) and causal interactions (effective connectivity) between distinct units within a nervous system [78].

Firstly, anatomical (or structural) connectivity is a pattern defined by structural connections such as synapses or fiber pathways. While the anatomical connectivity shows a robust status against external and internal stimulations, the other connectivities encounter various stimulations caused by spontaneous or task-based activities.

Secondly, functional connectivity, unlike anatomical connectivity, is a pattern representing the *correlations* of spatially distributed regions in a temporal domain. Such patterns of dynamic interactions are measured by statistical dependencies (e.g., cross-correlation values) from time series signal data [78]. The signal data can be obtained by various neuroimaging methods such as functional magnetic resonance imaging (fMRI), electroencephalographic (EEG), magnetoencephalographic (MEG), local field potentials (LFP), positron emission tomography (PET), and multielectrode array (MEA) [32].

Lastly, causal (or effective) connectivity is another pattern of causal effects among neural elements, often inferred on the basis of temporal consecutiveness. Commonly, it may be regarded as the union of anatomical and functional connectivity, as it describes networks of directional effects of one neural element (or specified region) over another. Basically

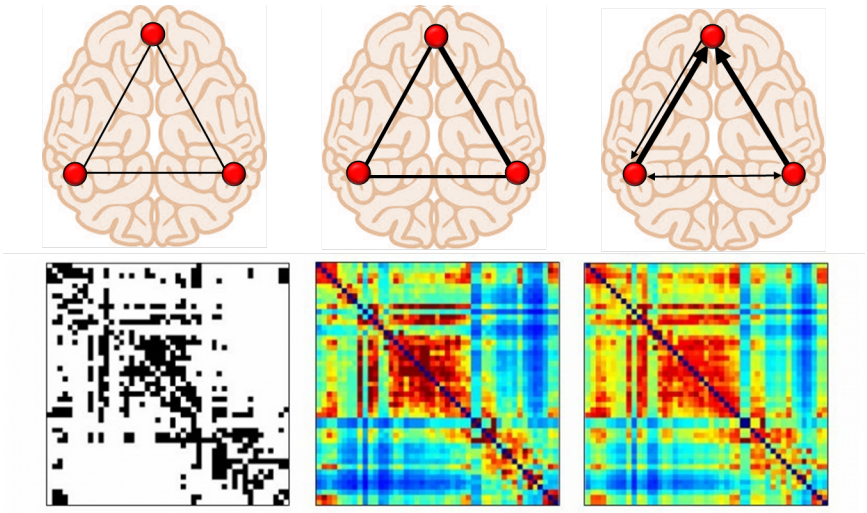


Figure 1.2: Three modes of brain connectivity. Figures at the top display structural connectivity (fiber pathways), functional connectivity (correlations), and effective connectivity (information flow) among four brain regions in macaque cortex. Matrices at the bottom show binary anatomical connections (left), symmetric mutual information (middle) and non-symmetric transfer entropy (right).

brain connectivity patterns can be represented in graph or matrix format (see Figure 1.2). Anatomical brain connectivity builds a sparse and directed graph. The graph can be weighted, with weights representing connection densities or binary, with binary elements indicating the presence or absence of a connection. Functional brain connectivity builds a full symmetric matrix, with each of the elements encoding statistical similarity or proximity between two system elements (neurons, recording sites, voxels). Such matrices may be *thresholded* to yield binary undirected graphs, with the setting of the threshold controlling the degree of sparsity. Effective brain connectivity provides a full non-symmetric matrix. Applying a threshold to such matrices yields binary directed graphs.

A common denominator to the different kinds of brain connectivity is the idea that, in

all instances, the system can be described as a (structural or functional) network. The idea that the brain is a network-like system is far from being novel in neuroscience, but attempts to examine these properties under formal mathematical network theories are a recent endeavor.

Bran Network Analysis

Various ways described in Section 1.1.2 allow us to analyze the brain network similar to what we have done in general complex network analysis. The analysis of neural connectivity matrices has exhibited the small world network property [181, 97, 193, 198, 150]. In different cortical connectivity matrices, significant correlations were observed between the path length or the clustering coefficient and the degree of each brain region. It is worth to notice that in most of cases the networks are considered as unweighted. However, neural system can be described as weighted networks. In this case, the framework of small world network has suggested the hypothesis of minimization of connection between brain regions.

Together with the small world property, some cortical networks display a multi-cluster organization which is related to a scale free network [193]. Moreover, some preliminary studies suggest that there is a similar behavior between when some nodes int the cortical network are removed and scale free network. However, it is too early to conclude that brain network has scale free network property by statistical assessment due to the small size of the data sets [97]. Recently, local connectivity patterns in neural connections have been identified by motifs analysis [68]. The local organization of neural links significantly displays some motifs. Within the cortical connectivity, a distinction has been made between functional and structural motifs. Structural motifs refer to neural elements connected by anatomical connections, whereas functional motifs consider the possible combinations of nodes and links within the structural motifs.

Researches who have attempted to assess a complex topology from the functional connectivity have shown relatively slow progress compared to anatomical connectivity researches [135, 78, 181, 166]. The first reason is a difficulty for a unified definition of the connectivity. Due to the volume conduction in brain tissue, it has different spatial and temporal resolution

scales of recorded dynamics. Second is the drawbacks induced by use of functional correlations to asses brain connectivity patterns. As we mentioned above, functional connectivity is temporal correlation between brain activities recorded by several techniques. Correlation results definitely depend on the correlation calculation methods. Neural oscillators are essentially complex nonlinear systems, hence a nonlinear character of their interactions is expected. Therefore, appropriate statistical methods are necessary to deal with the possible nonlinearities in the time series captured by neuroimaging techniques. Therefore, in order to assess a significant complex topology in brain connectivity, a careful statistical study would be necessary. As functional connections are strongly time dependent, the correlation methods need to capture the fluctuations in the connections. Moreover, functional correlations between some brain areas are very sensitive to slow frequencies (< 0.1 Hz), or to physiological noise sources as the respiratory and cardiac activities.

Functional connectivity in fMRI

Nevertheless the difficulties in researches of functional connectivity network, the functional network analysis has shown noticeable achievements as time goes on [166, 2, 3, 193, 167, 198, 18, 129]. Especially, fMRI has been widely used in many research studies as a basic tool to investigate the brain functional connectivity and perform brain functional mapping. Without invasive experiments, fMRI obtains sequence of individual MRI images that shows the level of Blood Oxygen Level Dependent (BOLD) signals over time. Similar to network analysis based on the anatomical connectivity, functional connectivity analysis by use of fMRI has been done. The analysis of connectivity in fMRI suggested a scale free network in brain functional networks [65]. Furthermore, various network analysis has applied to functional networks to reveal the underlying structure of functional network in brain.

Resting state fMRI (rsfMRI or r-fMRI) has emerged as a reproducible and unbiased way of measuring the functional connectivity between different regions of the brain [129, 86, 199, 87, 59, 197, 73]. Even in the absence of stimulus or explicit tasks (i.e., resting state), recent studies have identified multiple correlated networks in the brain associated with various functions [59]. The most easily detected and consistent of these is the default mode

network (DMN). The reproducibility of this network across patients and studies makes it a target of interests in functional connectivity research. Many studies have shown that disruption/alteration of DMN structure have been associated with neurological disorders such as Alzheimer's disease (AD) [217], schizophrenia [197], and dementia [73, 152]. Functional connectivity mapping using r-fMRI has been shown to be one of the most important methods for evaluation neurological integrity of many cognitive networks [165].

1.3 Challenges and Opportunities

In this section, I comprehensively describe the challenges of the aforementioned network research methodologies. First of all, I will explore the issues arising from large-scale data (*aka* big data) which are common challenges for complex network research. The main characteristic of big data is its massive volume size and high dimensionality, which results in many critical issues, such as problems concerning data storage, scalability, and difficulties of understanding the data [69]. In spite of these challenges, large-scale data have also brought great opportunities for modern data analysis, such as the chance to improve general performance and to discover subtle knowledge.

Specifically, for networked data studies, the concurrent challenges are as follows:

- In network optimization, many problems for solving an optimization scheme are NP-hard or have comparable difficulties. This is especially true when a network design problem involves subnetwork identification that satisfies a set of constraints with minimum or maximum total weights. This is a well-known combinatorial optimization problem and NP-hard in nature [5]. Due to the complexity of computation, the computational cost of this network optimization problem will be drastically increased, along with the data size. Considering the fact that an optimization problem based on complex networks generally involves a huge size of constraints and variables, developing effective solution approaches still remains challenging.

- In global attribute analysis, similar to the case of the network optimization, scalability is the most critical issue for complex, structured network data. Moreover, as the data size increases, the accumulation noise effect becomes more pronounced for complex

networks with high dimensional data [69]. Also, existing methodologies for defining topological attributes have been limited capability to capturing local dynamics (e.g., only a few components have changed) [143, 112].

- In machine learning circles, the main issue is how to convert discrete network objects into numerical features for effective learning. Adapting existing feature extraction methods directly to networked data is fundamentally non-trivial. Basically, networks are semi-structured data. So there is no obvious choice of features in networks. Furthermore, limited progress has been made by applying kernel functions for feature extraction because developing well-designed kernels is not a simple task. Also, the connection between kernel space and the original network space is not clear. Even if one extracts features from the kernel successfully, there still exists the high dimensionality problem.

- In addition to the fundamental challenges of kernel methods, most of the feature extraction methods are based on unweighted networks which focus on characterizing binary connectivity structures (i.e., connected vs. disconnected). Therefore, it is impossible to capture subtle distinctions in these connectivities, such as their status as being weakly connected or strongly connected. Since most of the existing methods are not applicable when all observations have same network structure (e.g., all networks are complete networks with different weights), researchers have used a filtering (thresholding) method which removes the edges whose weights are smaller than the predetermined threshold in order to obtain heterozygous connectivity structures. However, this method leads to serious drawbacks, such as there is no clear evidence that the removed edges are not informative. Thus, when using this method, one has to rely on prior domain knowledge of given networks and the resulting structures are highly dependent on the employed test metrics [164, 189].

Nevertheless, complex networked data has brought great opportunities for the network science community.

- As a kind of big data, it help us to understand systems more precisely and to possess

abundantly detailed information to describe them. Furthermore, it provides important information which was hidden by complex connectivity representations. For example, if one aims to establish the most cost-effective delivery schedule based on a traffic network, that network contains a great deal of detailed logistical information that will be helpful for deciding upon the most reliable and optimized schedule.

- The structure of data is important for understanding the intrinsic property of data. However, this subtle information in the data can be obtained accurately only if enough data are provided. For example, a change in local connectivity patterns in human brain networks is significantly determined only if these networks contain enough data collecting points (nodes). Also, the acquired networked data might be generated from multiple sources with slight dynamics. These changes are discovered only if the data size is large enough.
- The large volume and high dimensionality of networked data is helpful to improve stability and generalization. In general, with more features, a better description regarding research objectives can be achieved.

1.4 The Motivation and Roadmap of this Thesis

Critical components within a complex network or between complex networks can provide us insightful into the intrinsic dynamics and properties of complex structures in networked systems. For example, in August 2003, North America suffered a blackout affecting two million people in eight U.S. states, with an estimated economic loss of between 7 and 10 billion US dollars. The blackout started with the failure of a few power stations [75]. If one was able to identify such vulnerable components at the early stage, the restoration process could begin immediately and the loss caused by the blackout could be reduced [64].
Mathematically, critical components can be found by solving cardinality constrained network design problems with the employment of the following constraint:

$$\|G_s\| \leq k \tag{1.1}$$

where G_s is a resulting critical component and $\|.\|$ indicates size of networks e.g., the number of nodes or edges. A parameter k denotes the size of networked to be designed. Based on

the domain knowledge, G_s can be represented by various network topologies such as clique, tree, or quasi-clique.

As a tool for complex network analysis, the identification of critical components via optimization frameworks can produce insights which help us investigate the topological structures of complex networks. For example, finding a maximum clique, which is one of the classic network optimization problems, has been used to detect cohesive subgroups in large social networks [16]. Also, recent studies have revealed that large-scale social and information networks contain hierarchical or tree-like structures. This result suggests that the core tree structure is crucial to understanding connectivity in networks [4, 141].

As a formal representation, (1.1) can be transformed to the ℓ_0-norm, such as $\|\mathbf{x}\|_0 \leq k$, where $\mathbf{x}$ is a selection indicator vector for nodes or edges. In general, the resultant optimization problem with a ℓ_0-norm is NP-hard [37].

The critical components in a single network can provide meaningful information concerning the classification of networked data within a group of labeled networks. Given a series of networks, where every single network has a corresponding label, identification of core components can provide clues to help us unveil the underlying data structure. For example, in human brain networks, such critical components allow us to detect the most important regions of the brain related to certain neurological diseases, such as regions in the motor cortex for Parkinson's disease and regions in the default mode network for Alzheimer's disease. Moreover, along with feature selection in learning tasks with complex networks, selecting critical components consists of finding the most relevant features regarding a given output and is very effective for improving predictive, as well as computational performance.

By exploiting optimization frameworks for finding critical components, the goal of this thesis is to conduct complex network analysis on three topics, namely:

- Exact solution approaches for k cardinality tree
- Node selection of networked data classification via mathematical programming
- Convex optimization for group feature selection of networked data

In Chapter 3, as one of the approaches to finding critical components, I develop mixed integer programming (MIP) mathematical models to find a subtree in the network. In particular,

the goal there is to find a k-cardinality tree (KCT), which is the subtree with a fixed cardinality of nodes, while minimizing or maximizing the sum of the edge costs in the subtree. It is a classic network optimization problem and also regarded as a combinatorial optimization problem which generalizes the Minimum Spanning Tree (MST) problem. Originally, this problem was firstly introduced by Fischetti et al. [74] and it has gained considerable interest in recent years due to its various applications. We modulate the constraints of the problem in two phases: the connected component module and cycle elimination. Based on the modulation, we then propose seven variations of MIPs. Also, I extend the issue of finding critical components to a group of networks with corresponding label information. KCT is employed as a tool for feature extraction and an extracted tree for each network is used as a measurable characteristic of the original complex network. Functional connectivity of human brain are introduced as a case point for this approach.

In Chapter 4, I extend the issue of finding critical components to a group of networks with corresponding label information. Motivated by feature selection, I propose a new MIP which aims to provide a node selection scheme involving networked data classification. In contrast with conventional feature selection-i.e., edge selection in the networked data I propose to select nodes that are regarded as indirect, but more informative features of the networked structure. Due to NP-hardness and the big-M constraints of the model, I develop a branch-and-cut algorithm based on Bender's decomposition.

In Chapter 5, in spite of the novelty of the proposed mathematical model and solution approaches in the Chapter 4, the overall computational times are shown to be not satisfactory for practical usage. Therefore, I formulate a mixed integer nonlinear programming (MINLP) problem based on the SVM formulation. In order to resolve computational intractability, it is relaxed into a convex quadratically constrained linear programming (QCLP) model and solved by an iterative algorithm.

1.5 Contributions

The major contributions of this thesis are summarized as follows:

Table 1.1: Overview of this thesis

	Chapters		
	3	4	5
Component Topology	k sized tree	k sized subnetwork	
Math. Model	MIP	MIP	MINLP
Solution Approach	Exact (global optimal) solution		Heuristic
Algorithm	Branch and Bound	Branch and Cut	Iterative algorithm
Instance	Single weighted network G	Labeled weighted networks; $\{G_i, y_i\}_{i=1}^n$	

1.5.1 Exact Solution Approaches for the k-Cardinality Tree Problem

Given a connected, undirected and weighted graph $G = (V, E)$ with a positive integer, k, the k-Cardinality Tree Problem (KCTP) seeks to find a minimal cost subtree of graph G with exact k edges ($k - 1$ nodes). The KCTP has been proven to be NP-hard, and there have been several exact and heuristic approaches to this in the literature. Existing methods are either inefficient or too complicated to implement with varying degrees of performance. In this study, I, therefore:

- Propose new MIP formulations which are composed of two submodules: cardinality restriction and cycle elimination.
- Provide polyhedral comparisons which are conducted in terms of the tightness of the corresponding polyhedron for each formulation. This provides a theoretical prospective for the modules which has wide applicability for various network design problems.
- Apply the Reformulation and Linearization Technique (RLT) in order to strengthen this formulation.
- Propose formulations that are compact and easy to implement, whereas the state-of-art method (using a branch-and-cut-based algorithm [174]) includes complicated preprocessing tasks and pricing steps in the whole process.
- Propose a model from the computational results that outperforms the current cutting-edge algorithm for analyzing complex networks, especially denser networks.
- Employ KCT as a tool for the complex network analysis. It can be used as a feature

regarding as a marker for the classification.

1.5.2 Node selection of networked data classification via mathematical programming

In this study, I focus on a specialized version of networked data classification, which involves classifying data where every observation is constructed by a common network structure with different edge weights. Thus, under a conventional classification scheme, the collection of such edge weights may be regarded as a feature vector for each network object. Therefore, to select features is equivalent to selecting edges and it is hard to reflect connectivity structure from the given data and find meaningful information. In this study, I, therefore:

- Present a new mathematical formulation to classify the networked data and improve computational efficiency by employing the branch-and-cut algorithm based on Bender's decomposition.
- Propose a new feature selection scheme: "node-based feature selection" to overcome over-fitting which generally deteriorates predictive performance, as well as leading to computational inefficiency with a reflection of the implicit networked structure in the multidimensional data.
- Verify our new methods through comparison with conventional feature selection schemes and provide a new perspective through our methods.

1.5.3 Convex Optimization for Group Feature Selection of Networked Data

In this study, I expand node selection of networked data to general group feature selection. This features a collection of edge weights and feature "groups" that are a collection of nodes in the network. The proposed mathematical model entails a ℓ_0-norm regularization, which is NP-hard due to its non-convex properties, but which provides a better predictive performance than ℓ_1-norm regularization.

- To overcome the computational challenge of the ℓ_0 norm regularization in group feature selection, I propose a mathematical model with ℓ_0 norm based on the support vector machine (SVM) formulation and to develop an efficient computational technique to solve the proposed model.

- This technique is based on a convex relaxation to address the high dimensional data by solving the proposed quadratically constrained linear programming (QCLP) problem which is convex. Since there are an exponential number of constraints in the QCLP, we propose an iterative algorithm to solve it. The proposed technique finds the most informative B-sized groups of features iteratively by using an approximation algorithm. Once B feature groups are found, our technique determines the feature weight, $\mathbf{w}$, that achieves the optimal trade-off between the maximization of the hyperplane distance. This process automatically goes through the re-training process for selected features and that helps to overcome the "selection-bias" which has been a critical drawback in the ℓ_1 norm-based feature selection method.
- I apply the proposed group feature selection to the networked data classification, regarding a node as a group. As a consequence, the selection problem was achieved by solving a network optimization problem, specifically the "k heaviest subnetwork problem". Therefore, the proposed technique is based on combining machine learning methodologies with network optimization throughout the entire process.
- I apply this technique to both synthetic and real data sets in the context of networked data classification. Through this application phase, I show that the proposed technique is able to reveal new perspectives concerning the feature selection of networked data.

1.6 Organization of Thesis

The organization of this thesis is described as follows:
Chapter 2 gives a detailed introduction to related studies on the key three topics and current issues. Chapter 3 presents the exact solution approach for the k cardinality tree problem. Chapter 4 provides a new mathematical model for a node selection scheme in networked data classification. Specifically, the branch-and-cut algorithm based on the Bender's decomposition is proposed in order to solve the problem efficiently. Chapter 5 presents a convex relaxation approach for the group feature selection problem which is a generalized version of node selection. Nonlinear mixed integer programming is developed and solved by an iterative algorithm. Chapter 6 concludes this thesis and discusses possible future work.

Chapter 2

RELATED STUDIES

In this section, we provide various methodologies for network optimization and network analysis for complex network. Furthermore we present a brief review of the classification and networks classification based on the optimization framework. Since the network optimization involves concepts and principles of optimization including linear and integer programmings, we presume the readers have basic backgrounds about that. Moreover, as we mentioned at Chapter 1, network optimization and analysis encompass preliminary principles about graph theory. Graph theory is the natural framework for the exact mathematical treatment of networks and, formally a network can be represented as a graph. Here, we establish terminology and basic notions about graphs first.

A directed graph $G = (N, A)$ consists of a set N of nodes and a set A of pairs of distinct nodes from N arcs. The number of nodes and arcs are denoted by N and A respectively. An arc (i, j) is represented as on ordered pair and is to be distinguished from the pair (j, i). If we define arc (i, j),we can say that (i, j) is outgoing from node i and incoming to node j; we also say that j is an outward neighbor of i and that i is an inward neighbor of j.We say that arc (i, j) is incident to i and to j, and that i is the start node and j is the end node of the arc. The degree of a node i is the number of arcs that are incident to i. If a graph contains all possible arcs, we can say it a complete (aka, fully connected) graph.

We note that much of the literature of graph theory distinguishes between directed graphs where an arc (i, j) is an ordered pair to be distinguished from arc (j, i), and undirected graphs where an arc is associated with a pair of nodes regardless of order. For more precise description, we use a terminology an edge $\{i, j\}$ instead of arcs (i, j) where $\{i, j\}$=$\{j, i\}$ and a set of edges is denoted by E. Therefore, an undirected graph is represented as $G = (N, E)$.

A *path* in a directed graph G is a sequence of nodes $(n_1,\ n_2,\ \ldots\ ,n_k)$ with $k \geq 2$ and corresponding sequence of $k-1$ arcs such that ith arcs in the sequence is either (n_i, n_{i+1}) or

(n_{i+1}, n_i). In detail, a path is said to be 'forward' if all arcs are on the forward directions, otherwise we said it a 'backward' path.

A *cycle* is a path for which the start and end nodes are the same. A path is said to be simple if it contains no repeated arcs and no repeated nodes, except that the start and end nodes could be the same (in which case the path is called a simple cycle). A Hamiltonian cycle is a simple forward cycle forward cycle that contains all the nodes of the graph. Some authors use the term "walk" to refer to a path and term "path" to refer to a simple path.

If a graph does not contain any simple cycles, it is said to be *acyclic*. A graph is said to be *connected* if for each pair of nodes i and j, there is a path starting from i to ending at j. It is said to be *strongly connected* if for each pair of nodes.

A graph $G' = (N', A')$ is a *subgraph* of graph $G = (N, A)$ if $N' \subset N$ and $A' \subset A$. A *tree* is a connected acyclic graph. A *spanning tree* of graph G is a subgraph of G, that is a tree and contains all the nodes of G. It can be shown that a subgraph is an spanning tree if and only if it is connected and it contains $N - 1$ arcs.

A *clique* is a subset of nodes such that every two nodes in the subset are connected by an edge. It is equivalently called a complete subgraph. Cliques are one of the basic concepts of graph theory and are used in many other problems and construction on graphs.

2.1 Methodologies: Network Optimization

From Section 1.1, we examined that network optimization problems involve graphs corresponding to the structures of the networks. We introduce basic concepts of flow which can be measure the quantity flow through each arcs. Mathematically, the flow of an arc (i, j) is a scalar (real number), which we usually denote by x_{ij}. It is convenient to allow negative as well as positive values for flow. In applications, a negative arc flow indicates that whatever is represented by the flow (material, electric current, etc.), moves in a direction opposite to the direction of the arc.

Given a graph $G = (N, A)$, a set of flows $\{x_{ij} | \forall (i, j) \in A\}$ is considered as a flow vector. The divergence (or conservation) vector y asscociated with a flow vector x is the N dimensional

vector.

$$y_i = \sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji}, \quad \forall i \in N. \tag{2.1}$$

Thus, y_i is the total flow starting from node i less the total flow arriving at node i; it can be regarded as the divergence of node i. The flow vector x that we will consider often be constrained to range from lower to upper bound of the form:

$$l_{ij} \leq x_{ij} \leq u_{ij}, \quad \forall (i,j) \in A.$$

2.1.1 The Minimum Cost Flow Problem

By means of the predefined flow variables and divergence variables, we can formulate the linear programming (LP) version optimization problem; minimum cost flow problem such that it is to find a set of arc flows while minimizing a linear cost function, subject to the constraints that they produce a given divergence vector and they lie with in given bound;

$$\begin{aligned} \min \quad & \textstyle\sum_{(i,j)\in A} c_{ij}x_{ij} \\ s.t. \quad & \\ & \textstyle\sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji} = s_i \quad \forall i \in N, && (2.2) \\ & l_{ij} \leq x_{ij} \leq u_{ij} \qquad\qquad \forall (i,j) \in A. && (2.3) \end{aligned}$$

where c_{ij} is the cost occurred when the flow is passing by arc (i,j). l_{ij} and u_{ij} is the lower and upper bounds of (i,j) respectively. s_{ij} is the supply of node i when $s_{ij} \geq 0$. Otherwise, it is called the demand of i. We refer to constraints (2.2) and (2.3) as the conservation of flow constraints and the capacity constraints, respectively. If a flow vectors satisfies both constraints, it is called feasible flow.

For a typical application of the minimum cost flow problem, consider the nodes as locations (cities, warehouses, or factories) where a certain product is produced or consumed. Consider the arcs as transportation links between the locations, each with transportation cost c_{ij} per unit transported. The problem then is to move the product from the production points to the consumption points at minimum cost while observing the capacity constraints of the transportation links. Moreover, the minimum cost network flow problem has many

applications that are well beyond the transportation as will be seen from the special versions. Basically, the following special versions play a central role in the theory and applications of networks.

2.1.2 The Shortest Path Problem

Let suppose that each arc (i, j) of a graph is weighted a cost c_{ij}, and suppose that we define the cost of a forward path to be the sum of the costs of its arcs. Given a pair of nodes, the shortest path problem is to find a forward path that connects these nodes with minimum cost. Similarly, it is possible to cast the problem of finding a shortest path from node s to node t as the following minimum cost flow problem.

$$\begin{aligned} \min \quad & \textstyle\sum_{(i,j)\in A} c_{ij}x_{ij} \\ s.t. \quad & \\ & \textstyle\sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji} = \begin{cases} 1 & i = s, \\ -1 & i = t, \\ 0 & \text{otherwise} \end{cases}, \qquad (2.4) \\ & x_{ij} \geq 0 \qquad \forall (i,j) \in A. \qquad (2.5) \end{aligned}$$

To see this, let associate with any forward path P from s to t the flow vector x with components given by

$$x_{ij} = \begin{cases} 1 & (i,j) \in P \\ 0 & \text{otherwise} \end{cases} \qquad (2.6)$$

Then x is a feasible for problem and the cost of x is equal to the path P's length. Thus, if a vector x of the form (2.6) is an optimal solution, the corresponding path P is shortest path equivalently.

From the above LP, we can observe that the solution of LP is the shortest path in G with starting node s to ending node t. Apart from solution approach in terms of LP, there exist various algorithmic ways to solve the problem. Here, we introduce one famous algorithmic way: Dijkstra's algorithm.

Dijkstras algorithm works by first setting $d(s) = 0$ and $d(u) = \infty$ for all nodes $s \neq u$. The node s is a starting node and node u is all other nodes $\in N$. And then, we select the node u repeatedly with the smallest distance $d(u)$, and relaxing its edges, i.e. setting $d(v) = min(d(v); d(u) + w(u, v))$, where v is a neighbor node of u and $w(u, v)$ is the weight (or cost) of the arc (u, v). In the following pseudo-code, we assume that all nodes have been initialized to unvisited, $d(u) = \infty$, and $d(s) = 0$.

Algorithm 1: Dijkstras algorithm

Result: Shortest path P

```
Create queue Q and empty set P;
Q ← {s};
while Q ≠ do
    u ← argmin_{u∈Q} d(u);
    Q ← Q − {u};
    visited(u) ← true;
    for each v ∈ adj(u) do
        d(v) ← min(d(v), d(u) + w(u, v));
        if visited(u)==false then
            Q ← Q ∪ {v};
        end
    end
end
P = Q;
```

2.1.3 The Maximum Flow Problem

In the maximum flow problem, we have a graph with two special nodes: the source , denoted by s, and the sink, denoted by t. The objective of this problem is to move as much flow as possible from source s into sink t while satisfying the capacity constraints. Similar to the shortest path problem, we formulate the LP as a special version of the minimum cost flow problem by assigning cost c_{ij} as 0 to $\forall(i, j) \in A$ and by employing an artificial arc (t, s)

with cost -1. The LP formulation is given by:

$$\begin{aligned}
\max \quad & x_{ts} \\
s.t. \quad & \\
& \textstyle\sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji} = 0 \qquad \forall i \in N \text{ with } i \neq s \text{ and } i \neq t, \\
& \textstyle\sum_{\{j|(s,j)\in A\}} x_{sj} = \sum_{\{i|(i,t)\in A\}} x_{it} = x_{ts}, \\
& l_{ij} \leq x_{ij} \leq u_{ij} \qquad \forall (i,j) \in A \text{ with } (i,j) \neq (t,s).
\end{aligned}$$

Even though we can formulate the max-flow problem as LP, central to the max-flow problem is the max-flow/min-cut theorem, which is one of the most insightful and brilliant theorem of network optimization. Before explanation of the theorem, in here, we should introduce the concept of 'cuts' first.

A cut Q in a graph $G = (N, A)$ is a partition of the N into two non-empty subsets, a set S and its complement $\overline{S}$. We introduce the notation

$$Q = \left[S, \overline{S}\right]$$

For a cut $Q = \left[S, \overline{S}\right]$, we use the following notation

$$Q^+ = \{(i,j) \in A | i \in S, j \notin S\}, Q^- = \{(i,j) \in A | i \notin S, j \in S\},$$

and we note that Q^+ and Q^- are the sets of forward and backward arcs of the cut, respectively.

Given a flow vector x, the flux across a nonempty cut $Q = \left[S, \overline{S}\right]$ is defined to be the total net flow coming out of S, i.e.,

$$F(Q) = \sum_{\{(i,j)\in Q^+\}} x_{ij} - \sum_{\{(j,i)\in Q^-\}} x_{ij}, \quad \forall i \in N. \tag{2.7}$$

Recall from the (2.1),

$$y_i = \sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji}, \quad \forall i \in N,$$

We can observe that $F(Q) = \sum_{i \in S} y_i$. Analogously, given lower and upper bounds l_{ij} and u_{ij} for each arc (i, j), the capacity of a cut Q is:

$$C(Q) = \sum_{\{(i,j) \in Q^+\}} u_{ij} - \sum_{\{(j,i) \in Q^-\}} l_{ij}, \quad \forall i \in N. \tag{2.8}$$

Obviously, for any capacity-feasible flow vector x, the flux $F(Q)$ across Q is no larger than the cut capacity $C(Q)$. If $F(Q) = C(Q)$, then Q is said to be a saturated cut with respect to x; the flow of each forward (backward) arc of such a cut must be at its upper (lower) bound.

Recap the max-flow problem, where we want to maximize the divergence out of s over all capacity-feasible flow vectors having zero divergence for all nodes other than s and t. Given any such flow and any cut Q separating s from t, the divergence out of s is equal to flux across Q while it is smaller than the $C(Q)$. Therefore, it there exist a feasible flow, we have

$$\text{Maximum Flow} \leq C(Q). \tag{2.9}$$

2.1.4 The Multi-Commodity Flow Problem

Multi-Commodity flow problems involve several flow types or commodities, which simultaneously use the network and are coupled through either the arc bounds or through the cost function. For example, in Internet data transfer networks the commodities are the size of different classes of traffic (image, text, db-quarry, etc.) that involve different origin-destination pairs. Thus there is a separate commodity per class of traffic and origin-destination pair.

Given a directed graph $G = (N, A)$ with a set of ordered node pair (i_k, j_k), $k = 1, ..., K$, regarded as origin-destination (OD) pairs. The nodes i_k and j_k are referred to as the origin and destination of the OD pair. For each OD (i_k, j_k), we are given a r_k that is its input amount of traffic (or commodity). The objective is to divide each r_k among the many paths from the origin i_k to the destination j_k in a way that the resulting total arc flow pattern minimizes a cost function.

Let denote x_{ij}^k the flow on arc (i, j) of OD pair (i_k, j_k), flow conservation constraints are given by:

$$\sum_{\{j|(i,j)\in A\}} x_{ij}^k - \sum_{\{j|(j,i)\in A\}} x_{ji}^k = \begin{cases} r_k & i = i_k, \\ -r_k & i = j_k, \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in N.$$

for each $k = 1, ..., K$. Furthermore, the flows x_{ij}^k are required to be non-negative, and possibly to satisfy additional constraints, such as bound constraints. The cost function often has the form

$$f(x) = \sum_{(i,j)\in A} f_{ij}(y_{ij})$$

where f_{ij} is a function of y_{ij} which is the total flow of arc (i, j) such that

$$y_{ij} = \sum_{k=1}^{K} x_{ij}^k$$

2.1.5 The Minimum Spanning Tree Problem

Given a graph $G = (N, E)$ and a weight w_{ij} for each edge $\{i, j\}$, Minimum spanning tree (MST) problem is to find a spanning tree $G_T = (N_T, E_T)$ with minimum sum of edge weights. Note that the direction of arc is useless according to the definition of MST problem. Therefore, we will use edge $\{i, j\}$ as an indicator the connection between two nodes.

Basically, we can formulate the problem as Integer Programming model with decision variable x_{ij} given by:

$$x_{ij} = \begin{cases} 1 & \text{edge } \{i, j\} \in E_T, \\ 0 & \text{otherwise,} \end{cases}$$

The constraints of the IP require to restrict that the edges in E_T form a tree. According to the definition of spanning tree, solution tree must have $N - 1$ edges with keeping connected.

A possible IP of MST is given:

$$
\begin{aligned}
\min \quad & \textstyle\sum_{\{i,j\}\in E} w_{ij}x_{ij} \\
s.t. \quad & \\
& \textstyle\sum_{\{i,j\}\in E} x_{ij} = |N| - 1, && (2.10)\\
& \textstyle\sum_{\{i,j\}\in (S,S)} x_{ij} \leq |S| - 1, \quad \forall S \subseteq N, && (2.11)\\
& x_{ij} \in \{0,1\} \qquad\qquad \forall \{i,j\} \in E. && (2.12)
\end{aligned}
$$

where (S, S) is an edge set such that every edge in (S, S) go from a node in the set S to another node in the set S. (2.11) is the cycle elimination constraints in E_T. From the above formulation, we can observe following facts.

First, LP relaxation has integral extreme points. Second, Even though the LP relaxation has integral extreme points, this does not imply that we can solve the MST problem in polynomial time due to the exponential number of constraints. Hence, we can establish the following strategy. Relax the problem without (2.11). Given the solution, find which of the relaxed constraints are violated and add them. Resolve the problem including these constraints until no constraints is violated. This process is the basic step of 'branch and cut'. Detail description of the 'branch-and-cut' algorithm is [206]. Since we can separate the inequalities (2.11) in polynomial time, it follows that we can solve problem in polynomial time.

Asides from the IP, there exist other polynomial time algorithms to solve the problem like Dijkstra's algorithm in the shortest path problem. Here, we introduce one representative algorithm: Kruskal algorithm. As a first step, sort the edge in non-decreasing order of weight.

Algorithm 2: Kruskal's Algorithm

Result: Minimum spanning tree T

```
k ← 1;
i ← 1;
E_T ← ∅;
while i ≤ |N| do
    let e_k = {i, j};
    if i and j are not connected in (N, E_T) then
        E ← E ∪ {i, j};
        T ← T ∪ {i} ∪ {j};
    end
end
```

From the Algorithm 2, we can obtain the MST with $O(|E| + |N| \log |N|)$ complexity. Detail proof is available in [206].

2.1.6 Network Flow Algorithms

Typically, network optimization problems are not able to be solved analytically. In usual, they need to be addressed computationally with one of available algorithms. Until the above section, we have observed that network flow problems could be formulated by LP (or MIP). However, at Section 2.1.2, there exist algorithmic ways to solve the problem while yield same optimal solution with determined complexity. This section provides a broad categorization of the various classes of algorithms for network optimization problems. The network structure can be exploited to speed up the solution by using either an adaptation of a general purpose algorithm such as the simplex method, or by using a specialized network optimization algorithm [5]. In practice, network optimization problems can often be solved hundreds and even thousands of times faster than general linear or convex programs of comparable dimension. The algorithms the problems can be grouped in three main categories:

- Primal cost improvement: We try to iteratively improve the cost to its optimal value by constructing a corresponding sequence of feasible flows.

- Dual cost improvement: We define a problem related to the original network flow problem, called the dual problem, whose variables are called prices. We then try to iteratively improve the dual cost to its optimal value by constructing a corresponding sequence of prices.Dual cost improvement algorithms also iterate on flows, which are related to the prices through a property called complementary slackness.
- Auction: We generate a sequence of prices in a way that is similar to real-life auctions. Strictly, there is no primal or dual cost improvement here, although we will show that auction can be viewed as an approximate dual cost improvement process.

All of the preceding types of algorithms can be used to solve both linear and convex network problems (although the structure of the given problem may favor significantly the use of some types of methods over others). Detail descriptions of those algorithms are too broad and extensive. Details are in [20].

2.1.7 k Cardinality Constrained Network Design Problem

Given a network $G = (N, E)$ and a weight w_{ij} for each edge $\{i, j\}$, cardinality constrained network design problem is to find a sub network $G_s = (N_s, E_s)$ which is obtained by solving $G_s = \arg\min\{c(G_s) : |G_s| \leq k,\ G_s \in \mathcal{T}\}$, where $|.|$ indicates a size of sub network G_s (e.g., the number of edges and nodes) and $\mathcal{T}$ is a topology regularizer for a specific structure of G_s such as tree, clique and quasi-clique etc. Depending on the objective function $c(.)$ and a regularizer $\mathcal{T}$, the problems have been varied as followings.

- If $c(.)$ is a sum of the edge weights and $\mathcal{T}$ is a clique, the problem is equivalent to solve the maximum weight k clique problem (MWkCP) which is well known NP-hard problem. Basically, it is formulated by quadratic programming (QP) and has been solved by unconstrained QP [8] and linearization technique [47]. When all edge weights are equal, it is reduced to the just k clique problem which is NP-completeness and solved by $O(n^{0.792k})$ algorithm [196].
- If $c(.)$ is a sum of the edge weight and there is no specific $\mathcal{T}$, the problem is equivalent to solve the k heaviest sub network problem which is also known as NP-hard. It is also known as *k-cluster problem* and *dense k-subgraph problem* if all edges have the

same weight. Many solution methods have been developed to tackle this problem [130, 22, 34].

- If $c(.)$ is a sum of the edge weights, and $\mathcal{T}$ is a tree, the problem is equivalent to solve the k cardinality tree problem (KCTP) which is NP-hard. There exist various solution approaches including heuristics [27, 67, 106, 138, 30], approximation algorithm [84] and exact solution approaches [174, 52, 157]. Instead of sum of edge weight, if $c(.)$ is a sum of node weights, it is transformed to the k cardinality node weighted tree problem [40].

Nevertheless of broad applications and continuous needs from the various fields, computational challenges are still remaining due to the NP-hardness of the problems.

2.2 Methodologies: Global Topological Attributes of Complex Network

In this section, it is intended to serve as a brief account of the recent developments in the characterization and modeling of the topological properties of a network. We shall introduce to the analysis of the properties observed in real networks, and provide the reader with a brief review of the models motivated by the empirical observations.

2.2.1 Degree Distribution

The degree k_i of a node i is the number of edges incident with the node such as $k_i = \sum_{j \in N} x_{ij}$ where x_{ij} is a connection indicator whether there exist a edge (i, j) or not.

The most basic topological characterization of a graph G can be obtained in terms of the degree distribution $P(k)$, defined as the probability that a node chosen uniformly at random has degree k or, equivalently, as the fraction of nodes in the graph having degree k. Information on how the degree is distributed among the nodes of a undirected network can be obtained either by a plot of $P(k)$,or by the calculation of the moments of the distribution. The n-moment of $P(k)$ is defined as: $\langle k^n \rangle = \sum_k k^n P(k)$. The first momentum is the mean degree of G.

The degree distribution completely determines the statistical properties of uncorrelated networks. However, as we shall see, a large number of real networks are correlated in the

sense that the probability that a node of degree k is connected to another node of degree, say k' depend on k. Let introduce the conditional probability $P(k'|k)$, being defined as the probability that a link from a node of degree k points to a node of degree k'. Although the degree correlation are formally characterized by $P(k'|k)$, the direct evaluation of the conditional probability gives noisy results because of their finite size N. To overcome this problem, average nearest neighbor degree of i is defined by:

$$k_{nn,i} = \frac{1}{k_i} \sum_{j \in N_i} k_j = \frac{1}{k_i} \sum_{j=1}^{N} x_{ij} k_j$$

where N_j is the set of first neighbors of i. Using the above equation, one can get the average degree of the nearest neighbors of nodes with degree k, denoted as $k_{nn}(k)$.

2.2.2 Shortest Path Length, Efficiency

Shortest paths play an important role in the transport and communication within a network. From Section 2.1.2, we can obtain a shortest path from node i to j and we denote the length as d_{ij}. A measure of the typical distance between two nodes in the graph is given by the average shortest path length L as following:

$$L = \frac{1}{N(N-1)} \sum_{i,j \in N, i \neq j} d_{ij}$$

A problem with L is that it diverges if there are disconnected components in the G. One possibility is to consider the harmonic mean of geodesic length, and to define the so-called efficiency of G [203].

$$E = \frac{1}{N(N-1)} \sum_{i,j \in N, i \neq j} \frac{1}{d_{ij}}$$

Such a quantity if an indicator of the traffic capacity of a network and avoids the divergence of L, since any couple of nodes belonging to disconnected components of the graph yields a contribution equal to zero in the formula ($\frac{1}{d_{ij}} \approx 0$).

2.2.3 Clustering

Clustering, also known as transitivity, is a typical property of acquaintance networks, where two individuals with a common friend are likely to know each other. In G, transitivity means

the present of high number of triangles. This can be represented as T of the graph as the relative number of transitive triples:

$$T = \frac{3 \times \text{number of triangles in } G}{\text{number of connected triples of nodes in } G}$$

An alternative possibility is to introduce the clustering coefficient C, a measure introduced by Watts and Strogatz in [203] is defined as follows. A local clustering coefficient c_i is first introduced, expressing how likely $x_{jk} = 1$ for two neighbors j and k of node i. Its value is obtained by counting the actual number of edges (denoted by e_i) in G_i (See Section 2.2.2). The local clustering coefficient c_i is defined as the ratio between e_i and $\frac{k_i(k_i-1)}{2}$, the maximum possible number of edges in G_i [203]:

$$c_i = \frac{2 \times e_i}{k_i(k_i-1)} = \frac{\sum_{j,k \in N} x_{ij} x_{jk} x_{ki}}{k_i(k_i-1)}$$

The clustering coefficient of the G is then given by the average of c_i over all the nodes in G:

$$C = \frac{1}{|N|} \sum_{i \in N} c_i$$

By definition, $0 \leq c_i \leq 1$ and $0 \leq C \leq 1$ respectively [203].

2.2.4 Motifs

A motif M is a pattern of interconnections occurring either in a undirected or in a directed graph G at a number significantly higher than in randomized versions of the graph. As a pattern of interconnections, M is usually described as a connected (undirected or directed) n-node graph which is a subgraph of G. The research of the significant motifs in a graphs is based on matching algorithms counting the total number of occurrences of each n-node sized subgraph M in the original graph and in the randomized ones. The statistical significance of M is then described by the Z-score, defined as:

$$Z_M = \frac{n_M - \langle n_M^{rand} \rangle}{\sigma_{nM}^{rand}}$$

where n_M is the number of times (frequency) the subgraph M appears in G and $\langle n_M^{rand} \rangle$ and $\langle \sigma_M^{rand} \rangle$ are mean and standard deviation of the number of appearance in the randomized network respectively.

2.2.5 Small-Worldness

As we mentioned the property of small world network at Section 1.1.2, small world network is known to have high clustering coefficient C, but low average shortest path length L, compared to a random graph. The small worldness, δ is defined by:

$$\delta = \frac{\frac{C_G}{C_R}}{\frac{L_G}{L_R}},$$

where C_R is the clustering coefficient of random graph rewired in order to preserve the degree distribution of G, and L_R is the average shortest path length of such a random graph. Similarly, C_G and L_G is the clustering coefficient and the average shortest path length of G respectively. A network is generally regarded as "small world" if $\delta > 1$ [203, 97].

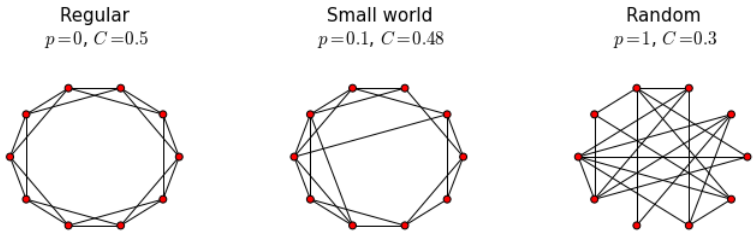


Figure 2.1: Watts-Strogatz model [203] for a small world network. A ring lattice become a small world after a random wiring with probability p. Continues rewiring edges the lattice's structure until a random graph. C is the clustering coefficient for each network

2.2.6 Robustness

As a measure of robustness, we looked at the resistance of network to fragmentation after removal of nodes either in random or in decreasing order of their degree. Suppose that there are M fragments in the network, i.e., M subgraphs that connected with each other but disconnected from each other. Resistance to such fragmentation is calculated as:

$$R = \frac{\sum_{j=1}^{M} N_j(N_j - 1)}{N(N-1)}$$

2.3 Methodologies: Classification and Feature Selection

Classification is a fundamental machine learning task whereby rules are developed for the allocation of independent objects to groups. Classic examples of applications includes medical diagnosis, economics, marketing, finance, and so on [118, 155, 209]. Basically, data are collected concerning objects with known group membership. This *training data* is used to develop rules for the classification of future objects with unknown group membership.
The classification methodologies can be categorized two approaches: (i) statistical inference based on the Bayesian frameworks and (ii) mathematical programming approaches based on the optimization frameworks. In this thesis, we will focus on the latter approach, (ii) mathematical programming approach, which is non-parametric and does not make strict assumption in terms of the distribution of the data analyzed.

2.3.1 Classification via Mathematical Programming

Mathematical programming approaches for classification have been developed in the last two decades. Since the pioneering work of [92] and [76], various formulations have been introduced as well as performance improvements. In mathematical programming approaches, the model can determine the coefficients of discriminant functions for different objectives. Most of the literature about mathematical programming approach is concerned with either using mathematical programming to determine the coefficients of linear discriminant functions or with support vector machines (SVM).

2.3.2 Linear Programming Classification

Linear programming (LP) approaches to determine the weights of linear discriminant functions (aka "linear classifier") has been widely investigated [76, 83, 105]. Since Freed and Glover proposed two linear programming (LP) approaches to solve the classification problem in discriminant analysis [76], many researchers have studied the variant of LP for the classification problem. Among the existing mathematical programming formulations, minimizing the sum of deviations (MSD) model has been received great interests due to the reliability and efficiency [147].

We are given n examples(or samples, observations, subjects) of objects and let n_1 and n_2 represents the number of subjects in two groups i.e., controls and patients respectively. i.e., $n = n_1 + n_2$. Let S denote the set of all subjects and let S_1 and S_2 are representing each class respectively (e.g., S_1=Controls vs. S_2=Patients). Thus we have $S = S_1 \cup S_2$. For each one, say the ith one, a description of its features in terms of m dimensional vector $\mathbf{w}^i = [w_1^i, w_2^i, ..., w_m^i]$

A *linear classifier* is defined in terms of an m-dimensional vector $\mathbf{x} = [x_1, x_2, ..., x_m]$ and a scalar x_0 and operate as follows. Let assume that there exist value x_0, x_j, $j = 1, ..., m$, such that (2.13) and (2.14) in the following hold if the subjects are clearly classified:

$$x_0 + \sum_j x_j w_j^i < 0 \quad \forall i \in S_1 \tag{2.13}$$

$$x_0 + \sum_j x_j w_j^i > 0 \quad \forall i \in S_2 \tag{2.14}$$

A linear classifier makes decision on the basis of a linear combination of the different features. The objective is to use the available samples in order to design a "good" classifier. In real world problems, it is hard to find a classifier separating all objects perfectly. If all subjects are not clearly separable, the objective of our model is to minimize the amount of violation of (2.13) and (2.14). Therefore, we introduce continuous variable ϵ_i which measure the violation of the following equation for each subject i:

$$x_0 + \sum_{j=1}^{m} x_j w_j^i + \epsilon_i = 0 \tag{2.15}$$

The amount of ϵ_i can denote the class of the subject such that if $i \in S_1$, then $\epsilon_i > 0$. Otherwise, $\epsilon_i < 0$ for $i \in S_2$. If we concern only the separation between two classes, then an equation (2.15) is the classification function itself. Once the weights x_0, x_j are obtained, these weights can be used as parameter values to classify new incoming samples.

Next, in order to make a "good" classifier, one need to estimate appropriate weights which separate the samples as correctly as possible. Therefore, we construct optimization problem, which is a form of linear programming (LP), to estimate the weights in order to fit a resulting classification model. In the constructed LP for consistency of the analogy, we use same x_0, x_j as classification weights. Furthermore, in order to estimate $\epsilon_s \in \mathbb{R}$, two non-negative

variables z_-^s, z_+^s, $\forall s \in S$ are introduced. Since the goal of the LP is to minimize the violation induced by determined linear classifier, we formulate the LP as follows:

$$\text{Minimize} \quad \sum_{i \in S_1} z_+^i + \sum_{i \in S_2} z_-^i \tag{2.16}$$

s.t.

$$x_0 + \sum_{j=1}^{m} x_j w_j^i + z_-^i - z_+^i = -\sigma \qquad \forall i \in S_1, \tag{2.17}$$

$$x_0 + \sum_{j=1}^{m} x_j w_j^i + z_-^i - z_+^i = \sigma \qquad \forall i \in S_2, \tag{2.18}$$

$$x_0, \quad x_j \quad \text{free} \qquad j = 1, ..., m, \tag{2.19}$$

$$z_+^i, z_-^i \geq 0 \qquad \forall i \in S. \tag{2.20}$$

In Constraints (2.17) and (2.18), σ is a constant and any σ can be used actually. Usually $\sigma = 1$ has been used for convenient usage. Since z_-^i, z_+^i are non-negative variables, at most one of them has a value. These variables represent how far each subject is separated by the estimated classifiers. Thus, if a subject is correctly classified then, $z_-^i - z_+^i + \sigma > 0$ for $i \in S_1$ and $z_-^i - z_+^i - \sigma < 0$ for $i \in S_2$. Therefore, z_+^i and z_-^i represent empirical errors of the classification eventually. Note that we employ positive σ in order to prevent the trivial solution where $x_0 = 0$ and $x_j = 0$ for all j. Since we use $\sigma > 0$, a gap of classification is equal to 2σ obviously.

2.3.3 Support Vector Machine

A support vector machine (SVM) is one of the mathematical programming approach based on the nonlinear optimization problem [56]. It has been extensively studied and has become popular method in many fields in recent years.

SVM incorporates the concept of structural risk minimization by determining a separating hyperplane that not only maximizes the margin separating the two classes of data but also minimizes a quantity measuring the misclassification error. We herein provide a brief mathematical background of SVM. Given empirical data $\{\mathbf{x}_i, y_i\}_{i=1}^n$ with input feature pattern $\mathbf{x}_i \in \mathbf{X}$ and binary label $y_i \in \{+1, -1\}$, SVM finds a linear hyperplane $h(\mathbf{x}) = \mathbf{w}^\intercal \mathbf{x} + b$ that separates the positive from the negative samples with the largest soft-margin. To construct

this optimal hyperplane, one solves the following mathematical optimization problem:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \quad \text{s.t. } y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) + \xi_i \geq 1 \quad i = 1, ..., n, \tag{2.21}$$

where ξ_i are slack variables for soft margin and C is a hyper-parameter between the margin space and the prediction error.

2.3.4 Feature Selection

High dimensionality of real world data, in both sample space and feature space, can hinder efficiency and efficacy of classification methods. Specifically, the *curse of dimensionality* leads to not only computational inefficiency for processing the data but also diminished classification performance, also known as *over-fitting* [89]. Fortunately, for many high dimensional data sets, most of the features are irrelevant to the output. Under this scenario, feature selection, also known as the variable selection, is to compress the original data set by selecting the most relevant subset of features based on some criteria.

Feature selection is a fundamental task in machine learning community. A limited sample size in real world data presents new challenges to traditional classification algorithms as they tend to overfit the prediction model and lack a generalization power when training on a dataset with the number of features far larger than the sample size ($m \gg n$ problem). The concept of parsimony (Occam's razor) is often invoked to have a minimal number of features, leading to simpler models, better generalization and easier interpretation. The main challenge of feature selection is thus to identify the least number of features, sub-set of the original set of features, that best address the over-fitting issue and improve prediction performance [89]. For this reason, feature selection has received a great deal of attention over the last decade (see, [88]) to deal with these challenges.

In general, existing feature selection methods can be classified into three categories, namely filter methods, wrapper methods, and embedded methods. The distinction of these categories is based on how feature selection and model training are integrated. *Filter methods* select features based on a statistical criterion (e.g., Chi-squared test, information gain) that scores each feature independently and removes features with the scores worse than a pre-determined threshold [89]. These methods are univariate and heuristic in nature, and

thus can reduce the computational complexity. However, they do not take into account the interaction between selected variables and chosen predictor and ignore correlations and complementary information that exist between features. *Wrapper methods* use deterministic (e.g., forward and backward passes) and/or stochastic (e.g., hill-climbing) search to find and evaluate subsets of features based on their usefulness to improve model accuracy. In other words, wrapper methods utilize search algorithms [113] as a black box and evaluate different combinations of features according to their predictive power [89]. However, the search process can be computationally intractable as the number of feature grows, and the problems is NP-hard [9]. Thus, the most commonly used wrapper method is based on greedy heuristics to add or remove one feature to/from the model at a time, known as forward and backward selection, respectively [111]. SVM with recursive feature elimination (SVM-RFE) is a widely used wrapper method where feature with the smallest weight in the trained model is removed iteratively [90]. *Embedded methods* aim to reduce the computation time of wrapper methods by incorporating the feature selection as part of the model training process to determine which features best contribute to the accuracy of the model. Regularization is the most commonly used embedded method, which performs feature selection as part of the model training by selecting the subset of features that best contributes to the accuracy of the model. Regularization introduces a penalty term (aka loss function) to the objective function to bias the model toward lower complexity and induce the model's sparsity (less features being selected).

2.3.5 Feature Selection with Cardinality Constraint

Feature selection is fundamentally an optimization problem, which can be mathematically defined as follows. Given input matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ of n samples with m features, label vector $\mathbf{y} \in \{+1, -1\}^n$, and the number of features k to be selected from m features, the feature selection problem is to determine the optimal weight vector $\mathbf{w} \in \mathbb{R}^m$ of m features that minimizes the expected generalized loss function $\mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y})$. There have been many variants of the loss functions $\mathcal{L}(.)$ introduced in the literature, such as least square loss in linear regression [91, 89], logistic loss in logistic regression [127, 144], and hinge loss in

support vector machines [144, 127, 187, 44]. The mathematical programming formulation of the feature selection problem is given by

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}), \text{ s.t. } \quad \|\mathbf{w}\|_0 \leq k, \tag{2.22}$$

where $\|\mathbf{w}\|_0 = \sum_{j=1}^{m} I_{\{w_j \neq 0\}}$ is the ℓ_0 norm to count the number of nonzero weights of $\mathbf{w}$ [46]. This feature selection problem has been shown to be NP-hard and computationally intractable [9].

As an alternative representation of (2.22), regularization technique (*aka sparse learning* [14]) tries to find a sparse prediction model by penalizing the cardinality (a number of nonzero elements) of the support of the weight vector $\mathbf{w}$ with a form of Lagrangian version of (2.22). Regularized model can be mathematically defined by $\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \gamma \|\mathbf{w}\|_0$, where $\gamma \in \mathbb{R}$ is a regularization parameter [89, 188]. This ℓ_0 norm regularization model has been shown to be NP-hard due to the non-convex ℓ_0 norm, $\|\mathbf{w}\|_0$ term in the objective function. Insightful theoretical properties of the regularization approach has been explored from a theoretical point of view in [215, 85, 169]. Moreover, [169] points out that (2.22) is *preferable* over the regularization technique in terms of superior properties of the resulting parameter.

To efficiently solve this regularization problem, researchers have proposed several convex relaxation approaches by replacing the ℓ_0 norm with ℓ_1 ($\|\mathbf{w}\|_1 = \sum_{i=1}^{m} |w_i|$) or ℓ_2 ($\|\mathbf{w}\|_2^2 = \sum_{j=1}^{m} |w_j|^2$) norm. Among the most notable techniques, least absolute shrinkage and selection operator (LASSO) was proposed as a sparse regression model with ℓ_1 norm regularization [190], and regularized support vector machines (RSVM) was also proposed as a classification model with ℓ_1 hinge loss [79, 144]. Although these techniques can reduce the computational effort, the use of penalty term does not explicitly control the number of features to be selected as it is subject to the scaling of features [38, 79, 201].

Nevertheless its computational difficulties, sparsity control as a feature selection is crucial for large-scale learning due to the following reason. For large and complex data, noise existence in the data becomes even worse due to the accumulation effect of noises for the learning model [69]. In the extreme case, the accumulated noise may totally dominate the true

patterns and dramatically degrades the learning performance. Sparsity control can resolve such challenge through selecting the most relevant ($\approx$ informative) features regarding a given output. As a result, it can be effective for high dimensional data to reduce the noise accumulation effect.

2.3.6 Limitations of ℓ_1 and ℓ_2 norm Methods

As we mentioned at Section 2.3.5, almost methods in the literature are based on either ℓ_2 or ℓ_1 norm as an approximation of ℓ_0 norm. Naturally, ℓ_2 norm is commonly used in regression (i.e., output y_i is continuous), $\|\mathbf{w}\|_2^2 = \sum_{j=1}^{m} |w_j|^2$, which is called ridge regression model. Although ℓ_2 norm can shrink all coefficients toward zero, it cannot guarantee a sparse solution (a small number of non-zero weights). In the regression, least absolute shrinkage and selection operator (LASSO) was proposed to impose an ℓ_1 norm penalty, which can induce both continuous shrinkage and variable selection simultaneously. After the seminal work in [190], many researchers have replaced ℓ_0 norm with ℓ_1 norm in logistic loss or hinge loss (see, [38, 79, 144, 201]). Despite computational efficiency of ℓ_1 norm, it has a critical drawback on "selection bias" because it is a by-product approach of sparse solution [88, 214]. A recent comparison study of LASSO [190] and forward stepwise regression [89], which is a greedy surrogate of ℓ_0 norm regularization, indicates that the ℓ_1 norm never outperforms the ℓ_0 norm by more than a constant factor, and in some cases, the ℓ_1 norm is much worse than the ℓ_0 norm [124]. In addition, fine-tuning the penalty parameter of the ℓ_1 norm regularization can be very challenging. On one hand, if we choose a large regularization parameter γ, minimizing $\gamma\|\mathbf{w}\|_1$ induces a very sparse $\mathbf{w}$, and very few features would be selected. However, the risk of poor prediction is also increased. On the other hand, if we choose a small γ, then the model would provide better fitted results that reduce the loss function values. However, the resulting model might not be sparse. For this reason, ℓ_1 norm methods cannot simultaneously provide the model sparsity and a de-bias solution to the model [124, 214]. Furthermore, according to the empirical tests in [124], the bias becomes more pronounced as the data size increases.

2.3.7 Group Feature Selection

Real world networked systems (e.g., biological systems, transportation systems, financial systems, energy systems, and cyber-physical systems) often possess an intrinsic relationship among features/sensors, known as feature structure. Some feature structures that have been studied in the literature include group (disjoint or overlapping) [101], hierarchy [109], and spatial smoothness [191]. In feature selection with group structure (group feature selection), features in one group can be selected if and only if the feature group is selected. In general, a feature can be assigned to multiple groups simultaneously (overlapping group structure). In this case, group feature selection can be accomplished by either one of the following two approaches: (1) selected features are defined by the union of selected groups [99] or (2) selected features are defined by the complement of the union of non-selected groups [101, 212, 119]. In this paper, we focus on the latter approach, which can be mathematically defined as follows. Suppose that m features are organized by p groups, $G = \{G_1, G_2, ..., G_p\}$, and each group contains an index set of feature supports belonging to the g-th group, $G_g \subset \{1, ..., m\}$, $g = 1, ..., p$. The group feature selection problem can be formulated as

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}), \text{ s.t. } \quad \|\mathbf{w}_{G_g}\|_{2,0} \leq B, \tag{2.23}$$

where $\mathbf{w}_{G_g} \in \mathbb{R}^{|G_g|}$ is the weight vector of $\mathbf{w}$ related to G_g and $\|\mathbf{w}_{G_g}\|_{2,0}$ is a modified $\ell_{2,0}$ norm $= \sum_{g=1}^{p} I(\|\mathbf{w}_{G_g}\|_2 \neq 0)$ to count the cardinality of selected features groups [13]. Regularization version of group feature selection can be defined by $\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \gamma \|\mathbf{w}_{G_g}\|_{2,0}$. This $\ell_{2,0}$ norm regularization of group feature selection is NP hard due to $\|\mathbf{w}_{G_g}\|_{2,0}$ [14]. To solve the group feature selection, convex relaxations based on ℓ_1 norm have been introduced. As an extension of the LASSO, Group LASSO was proposed as a convex optimization problem for group feature selection [213]. The mathematical formulation of group feature selection can be defined as follows. Suppose that m features are put into p non-overlapping groups $G = \{G_1, G_2, ..., G_p\}$ with m_g the number of features in group G_g. For brevity, we denote $\mathbf{w}_{G_g}$ as the coefficient of features corresponding to the gth group. Then, the formulation of group lasso is given by $\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \gamma \sum_{g=1}^{p} \sqrt{m_g} \|\mathbf{w}_{G_g}\|_2$, where $\sqrt{m_g}$ accounts for the varying group sizes, and $\|\mathbf{w}_{G_g}\|_2 = \sqrt{\sum_{j \in G_g} w_j^2}$ is the Euclidean norm [213]. This model

resembles the ℓ_1 norm at the group level. In fact, if G is the set of singletons, the model reduces to the standard LASSO. Group LASSO has been applied to several research and real life domains (see, e.g., [213, 132, 163]), and a wide variety of formulations and solution approaches of group LASSO have been presented in the literature [99, 101, 212, 186].

In many domains of interest, the instances are connected via a set of edges/links, thus forming a network. For example, research papers are connected by citations, telephone accounts are linked by calls, and brain connections are connected via different regions. Several classification methods have been developed for networked data in different domains, including web pages (e.g., [192]), computational biology (e.g., [115]), and human brain network (e.g., [180]). Feature selection of networked data can be thus considered to be a special case of group feature selection. Although the technique developed in this study is motivated by networked data, it can be generalized to group feature selection problems.

2.4 Methodologies: Networked Data Classification

The network classification can be defined as follows: There is a data of networks $G_i \in \mathcal{S}$ with $i = 1, ..., n$. Each network $G_i = (V_i, E_i)$ is given as a set of nodes and edges respectively. Finally, each network G_i has a corresponding label $y_i \in \mathcal{C}$ where $\mathcal{C}$ is the set of $|\mathcal{C}|$ classes of labels. In this thesis, we will only focus on the binary classification, i.e., $\mathcal{C} = \{+1, -1\}$. Along the same vein as the standard classification, the aim of network classification is to learn a classifier $f : \mathcal{D} \rightarrow \mathcal{C}$ that predicts the label for any unlabeled network.

2.4.1 Feature Extraction in the Networked Data Classification

The one of the challenges in the networked data classification is how to convert discrete network objects into numeric features for effective classification. Existing classification researches on the networked data can be divided into three following schemes.

First of all, graph kernel methods have been widely used to the classification in the machine learning community. The graph kernel enable us to represent the data as $n \times n$ symmetric, positive semi-definite kernel matrix $\mathbf{K} = \{\kappa(G_i, G_j)\}_{i,j=1}^{n}$ which measure the pair-wise similarity (or dissimilarity). Once obtained the kernel matrix $\mathbf{K}$, we are able to apply various classification approaches (e.g., SVM) to learn the networked data classifier. There

exists several studies to develop efficient and effective kernel κ such as the methods based on random walks [82], shortest paths [33], subtrees[161], and cycles[95]. Although those kernels have provide insightful features, there are still remaining challenges, especially for large sized instances, i.e., the complexity of computation for existing kernel methods are extremely *high*.

Secondly, it is possible to use global topology attributes for each G_i [120]. Instead of relying on fixed "patterns", computed global topological attributes, which are mentioned at Section 2.2, can be used as a feature vector. For example, one can calculate efficiency of G_i, $E(G_i)$ and clustering coefficient $C(G_i)$ and apply the SVM with a feature vector $\mathbf{x}^i = [E(G_i), E(C_i)]$ for ith network. Using global attributes can substantially reduce the computational cost [120].

Finally, it is usually called *'a bag of edges'*, where the networks are treated as a collection/bag of edges. A bag of independent edges through multivariate classification methods. Contrast from the usage of global attributes, it can reflect the entire connection weights simultaneously. Another novelty of this method is to extract informative sub-network through the feature selection techniques. Since each feature is one to one matched to its corresponding edge, we can construct a sub-network with selected features. We will handle it at the following Section 2.4.2.

2.4.2 Feature Selection in the Networked Data Classification

In the Section 2.4, we briefly introduce the methods for feature extraction from the given networks. Next step is to apply feature selection motivated by general feature selection in Section 2.3.5. Contrast from extensive studies in terms of feature extraction on the networked data, feature selection has been less studied. In using kernel method, typical approach is to use filtering method based on the frequency of the subnetworks. It aims to collect most frequently appealing subnetworks [208].

If we use the global attributes, selecting features is to pick the most informative attributes and does not differ from the standard feature selection. This method cannot determine a subnetwork as selected features.

If we transplant the feature selection scheme into the networked data classification, it would be equivalent to the edge selection problem since there exists one-to-one matching between an edge and a feature respectively. As a result, we can select a set of edges via conventional feature selection scheme. In addition, since the input data is composed of a specific networked structure, it can be regarded as one of the structural feature selection. When the data have complex structures, it is desirable to mine such structure for better prediction. Unlike to the extensive investigations for the general feature selection, feature selection researches on the networked data is at an opening stage. In the literature, several researches on the human brain connectivity networks has been done. [104] applied a wrapper feature selection method; SVM-RFE on the features made by random work graph kernel with mild cognitive impairment (MCI) and edge selection has been applied to the autism spectrum disorder(ASD) data [100, 51].

Chapter 3

EXACT SOLUTION APPROACHES FOR THE K-CARDINALITY TREE PROBLEM

3.1 Introduction

Let $G = (V, E)$ be a connected undirected graph with vertex set V, edge set E, edge cost function c: $E \rightarrow \mathbb{R}$. Let $T = (V_T, E_T) \subseteq G$ represent a tree structure. Then, the k-cardinality tree problem (KCTP) is to find a minimum cost tree with exactly k edges, which can be expressed by

$$T^* \in \arg\min_{\forall T \subseteq G} \{c(T) : |E_T| = k\}.$$

, where $c(T) = \sum_{e \in E_T} c(e)$.

KCTP was first introduced by Fischetti et al. who have also proven KCTP to be NP-hard when $2 \leq k \leq |V| - 2$ for arbitrary k as part of the input [74]. KCTP reduces to finding the minimum cost edge when $k = 1$ and the minimum spanning tree problem (MSTP) when $k = |V| - 1$, which can be solved efficiently [114, 156]. KCTP has been shown to be polynomially solvable if input graph G is a tree [27]. KCTP has been extensively studied in literature and applied to many practical applications in oil-field leasing, facility layout, open pit mining, matrix decomposition, quorum-cast routing, and telecommunication fields [29].

A number of heuristic approaches have been proposed to solve KCTP. Ehrgott et al. proposed a greedy approach based on dynamic programming (DP) [67]. Blum revisited the DP approach and employed Kruskal's and Prim's algorithms to solve the MSTP in the DP framework [27]. Meta-heuristic methods have also been applied to KCTP using tabu search [106, 24], evolutionary algorithms [25], variable neighborhood search [138, 30], and ensemble metaheuristics [28]. In addition to heuristic approaches, approximation algorithms have also been developed [11, 26, 80, 81], which are mainly based on the primal-dual theory proposed by [84] for the prize-collecting steiner tree problem.

Fischetti et al. introduced the first exact solution approach through integer programming based on general subtour elimination constraints (GSEC) requiring exponential number of constraints [74]. Ehrgott and Freitag solved the GSEC formulation using a branch-and-cut approach on graph instances with up to only 30 vertices [66]. As a follow-up work, Chimani et al. extended the formulation in [74] by adding symmetry breaking constraints that substantially reduced the search space for the branch-and-cut algorithm [52].

Simonetti et al. introduced a branch-and-bound algorithm using a rooted version of KCTP, where a predetermined vertex was chosen to be the root of the tree, and the algorithm was iterated over all vertices as root vertices [175]. Quintão et al. proposed two mixed integer programming (MIP) formulations [157]: one using multi commodity flow (MCF) constraints to enforce connectivity and another using lifted Miller-Tucker-Zemlin (MTZ) constraints to break subtours [62]. Their computational results showed that the MCF formulation had a stronger LP relaxation bound compared to that of the MTZ formulation; however, it required longer CPU times to obtain the LP relaxation. Therefore, they suggested Lagrangian relaxation for the MCF formulation while using a branch-and-bound algorithm to solve the MTZ formulation. More recently, Simonetti et al. introduced three new families of valid inequalities on top of the formulation in [52], two of which being facet defining constraints for the convex hull of feasible KCTP solutions [174]. Compared to other formulations, the approach in [174] produced the best computational results, especially for grid type graphs which are known as the crux of KCTP.

In this study, I propose new mixed integer programming (MIP) formulations to solve the KCTP. Our formulations are decomposed into two parts: the first is to find k cardinality constrained connected components with at most one cycle (subtour), and the secondary part is to remove such cycles. In the first part, three MIP formulations are introduced in terms of the number of artificial vertices to be added to G. For the secondary part, I use either single commodity flow (SCF) constraints or lifted MTZ constraints to break the cycles in the connected components. Following the reformulation-linearization technique (RLT) applied to the asymmetric traveling salesman problem in [173], I transform our MTZ constraints accordingly to obtain tighter linear relaxation bounds.

The formulations developed in our study have similarities to those in [157], however I present

computational evidence that the modifications with the artificial vertices in the first part provide several computational advantages. In particular, I compare the performance of our algorithms with other exact solution approaches: two formulations in [157] and a formulation in [174]. The formulations in [157] can be solved by a branch-and-bound algorithm while the formulation in [174] can be solved by a branch-and-cut algorithm. We should note that the formulations and the solution methods of [52] and [174] are quite similar; therefore, in this study I include comparisons of our formulations to only the latter since it outperforms the former. We report competitive results for higher edge densities and comparable results for lower edge densities, where our formulations may be preferable due to their ease of implementation using any commercial MIP solvers as opposed to complicated implementation required for the branch-and-cut algorithm in [174].

The rest of this study is organized as follows. In Section 3.2, I demonstrate the modular development of our formulations. In Section 3.3, the application of RLT to lifted MTZ constraints is proposed. Computational results from our formulations are compared to those from other formulations in Section 3.5. Finally, conclusions are drawn in Section 3.6

3.2 Mathematical Formulations

In this section, we present the modular development of our MIP formulations for the KCTP, including connections to other formulations. We first introduce the notation used in the study as follows. Given an undirected graph $G = (V, E)$, an edge $e \in E$ is defined as $\{i, j\}$ with endpoints of vertices i and j. For a given digraph $D = (V, A)$, I denote an arc $a \in A$ whose start vertex is i and end vertex j by (i, j). We also define $\delta^+(S) = \{(i, j) \in A | i \in S, j \notin S\}$ and $\delta^-(S) = \{(i, j) \in A | i \notin S, j \in S\}$ for outgoing and incoming arcs of set S respectively. Similarly, if set S contains single vertex i, we use $\delta^+(i)$ and $\delta^-(i)$ instead of $\delta^+(\{i\})$ and $\delta^-(\{i\})$ respectively.

The input for KCTP are an undirected graph $G = (V, E)$, an integer number $k \leq |V|-1$, and an edge cost function c. We transform an input instance graph G into a digraph $D = (V, A)$ to develop our formulations. An arc set A is a set of pairs of arcs obtained from a given edge set E such that $A = \{(i, j), (j, i) | \{i, j\} \in E\}$. Edge costs are also transformed into arc costs $c_{(i,j)} = c_{(j,i)} = c_{\{i,j\}}$. Given a digraph $D = (V, A)$ with a root vertex $s \in V$, the

feasible solution of the KCTP, $T = (V_T, A_T)$ has to satisfy following properties [52]:

- T has exactly k arcs,
- T has exactly $k + 1$ vertices,
- there is a di-path from s to v, where $\forall v \in V_T$, and
- indegree of each $v \in V_T \setminus \{s\}$ is equal to 1.

Considering these properties, I can identify two necessary conditions what the MIP formulation has to satisfy. First, the feasible solution of MIP, $T = \{V_T, A_T\}$ has to be consistent. That is, if an arc $(i, j) \in A_T$, then vertices i and $j \in V_T$. Second, T should not involve any cycle; that is, there should be a unique dipath from s to each $v \in V_T \setminus \{s\}$. Hence, I propose two constraint modules in this study: a primary module for finding connected components and a secondary module for subtour (cycle) elimination. We provide three different constraint sets for the primary module in Section 3.2.1 and two different constraint sets to break subtours in Sections 3.2.2 and 3.2.3. Finally, I propose a RLT approach to improve the MTZ formulation.

3.2.1 Connected Components with at Most One Cycle

Let $x_a \in \mathbb{B}$ be a binary variable that denotes whether arc $a \in A$ is selected or not, $y_i \in \mathbb{B}$ be a binary variable that denotes whether vertex $i \in V$ is selected or not, and $z_i \in \mathbb{B}$ be a binary variable that denotes whether vertex $i \in V$ is the root vertex or not. Now, I define the following set of constraints, which produce connected components with desirable properties.

$$X_{CC}: \quad \sum_{a \in A} x_a = k, \tag{3.1}$$

$$\sum_{i \in V} y_i = k, \tag{3.2}$$

$$\sum_{i \in V} z_i = 1, \tag{3.3}$$

$$\sum_{a \in \delta^-(i)} x_a = y_i, \qquad \forall i \in V, \tag{3.4}$$

$$z_i + y_i \leq 1, \qquad \forall i \in V, \tag{3.5}$$

$$x_{(i,j)} + x_{(j,i)} \leq (y_i + z_i), \qquad \forall (i, j), \in A. \tag{3.6}$$

$$x_a, y_i, z_i \in \mathbb{B}, \qquad \forall i \in V, \quad \forall a \in A. \tag{3.7}$$

Constraint (3.1) requires that exactly k arcs are selected. Constraints (3.2) and (3.3) enforce that exactly k ordinary vertices and one root vertex are selected, respectively. Constraints (3.4) require the in-degree of selected arcs at an ordinary vertex i to be equal to 1 due to feasibility property 3.2 if vertex i is selected, and 0 otherwise. Constraints (3.5) enforce that vertex i cannot be selected as the root vertex and an ordinary vertex at the same time. Constraints (3.6) require that only one of the two arcs between vertices can be selected if both vertices are selected as root or ordinary vertices.

Instead of choosing a vertex among the vertices in V, an artificial vertex can be introduced as the root vertex, which is indexed as vertex 0. [52] proposed a KCTP reformulation defined over a digraph $\hat{D} = (\hat{V}, \hat{A})$, where $\hat{V} = V \cup \{0\}$ and $\hat{A} = A \cup \{(0, i) : i \in V\}$ with $\{c_{(0,i)} = 0 : i \in V\}$. This change implies an increase in the cardinality of the tree to $k + 1$ due to the fact that one of the added arcs $(0, i)$ connected to the artificial vertex has to be selected. Removal of the selected arc $(0, i)$ from the solution would imply a k-cardinality tree of G, where vertex i becomes the actual root vertex in T. Constraints pertaining to the new augmented graph $\hat{D} = (\hat{V}, \hat{A})$ are given as follows.

$$
\begin{aligned}
\hat{X}_{CC}: \quad & \text{constraint (1)}, & & \\
& \textstyle\sum_{i \in V} y_i = k + 1, & & (3.8)\\
& \textstyle\sum_{a \in \delta^-(i)} x_a + x_{(0,i)} = y_i, & \forall i \in V, & (3.9)\\
& \textstyle\sum_{a \in \delta^+(0)} x_a = 1, & & (3.10)\\
& x_{(i,j)} + x_{(j,i)} \leq y_i, & \forall (i,j) \in A, & (3.11)\\
& x_a, y_i \in \mathbb{B}, & \forall i \in V, \quad \forall a \in A. & (3.12)
\end{aligned}
$$

Constraint (3.8) requires that any feasible solution must contain $k + 1$ vertices. Constraint (3.9) requires in-degree of any ordinary vertex to be 1 if the vertex is also selected, and 0 otherwise. Constraint (3.10) enforces that one arc connected to vertex 0 must be selected. Constraints (3.11) enforce only one of the two arcs between two vertices to be selected if both of the vertices are selected, and none otherwise.

[157] adopt a formulation where the vertex selector variables for the vertices are implicitly replaced by arcs from another artificial vertex $\acute{0}$ (in addition to vertex 0), which are used to

"unselect" vertices. Then, the new digraph $\overline{D} = \{\overline{V}, \overline{A}\}$ can be defined as $\overline{V} = V \cup \{0, \acute{0}\}$ and $\overline{A} = A \cup (\acute{0}, 0) \cup \{(0, i) : i \in V\} \cup \{(\acute{0}, i) : i \in V\}$. Corresponding constraint set is given as follows.

$$
\begin{aligned}
\overline{X}_{CC}: \quad & \text{constraint (3.1)}, \\
& \text{constraint (3.10)}, \\
& \textstyle\sum_{a \in \delta^+(\acute{0})} x_a = n - k, && (3.13)\\
& \textstyle\sum_{a \in \delta^-(i)} x_a = 1, \qquad \forall i \in V \cup \{0\}, && (3.14)\\
& x_{(i,j)} + x_{(j,i)} \leq 1, \qquad \forall (i,j) \in A, && (3.15)\\
& x_{(\acute{0},i)} + x_{(i,j)}, \leq 1, \qquad \forall (i,j) \in A. && (3.16)\\
& x_{(\acute{0},0)} = 1 && (3.17)\\
& x_a \in \mathbb{B}, \qquad \forall a \in \overline{A}. && (3.18)
\end{aligned}
$$

Constraint (3.13) requires that $n-k$ arcs emanating from vertex $\acute{0}$ are selected, which point at unselected vertices. Constraint (3.14) requires the in-degree of all vertices in V and also vertex $\{0\}$ to be equal to 1. Constraints (3.15) enforce only one of the two arcs between any two vertices can be selected. Constraints (3.16) enforce that in any feasible tree if $i \in V$ is connected to $\acute{0}$ then vertex i cannot be connected to other vertices and vice versa.
In the following two propositions, I show the equivalence of the three constraint sets, including the integrality constraints, and I present a structural property that they share.

Proposition 3.2.1 *There exists one-to-one correspondence mapping for integer feasible solution within each constraint sets.*

Proof Let $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ be a feasible solution of X_{CC}. Based on the $(\mathbf{x}, \mathbf{y}, \mathbf{z})$, let construct a solution of $\hat{X}_{CC}$; $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ as followings. Firstly, define $\hat{x}_a = x_a$, $\forall a \in A$. Next, let $\hat{x}_{(0,i)} = 1$ for $z_i = 1$ and $\hat{x}_{(0,i)} = 0$ otherwise $(z_i = 0)$, $\forall i \in V$. Finally, let $\hat{y}_i = y_i + z_i$, $\forall i \in V$.
In Constraints (3.8), $\sum_{i \in V} \hat{y}_i = \sum_{i \in V} y_i + \sum_{i \in V} z_i = k + 1$. In Constraints (3.9), for $i \in \{i \mid z_i = 0\}$, we have $\sum_{a \in \delta^-(i)} \hat{x}_a + \hat{x}_{(0,i)} = \sum_{a \in \delta^-(i)} x_a = y_i = \hat{y}_i$. Similarly, for $i \in \{i \mid z_i = 1\}$, we have $\sum_{a \in \delta^-(i)} \hat{x}_a + \hat{x}_{(0,i)} = \sum_{a \in \delta^-(i)} x_a + \hat{x}_{(0,i)} = z_i = \hat{y}_i$ because $\sum_{a \in \delta^-(i)} x_a = 0$ when $z_i = 1$ due to Constraints (3.4) and (3.5). In Constraints (3.10),

$\sum_{a\in\delta^+(0)} x_a = \sum_{i\in V} z_i = 1$ Constraints (3.11) is easily derived from Constraints (3.5) and (3.6). Thus, we can confirm that $(\hat{x}_a, \hat{y}_i)$ can be a feasible solution of $\hat{X}_{CC}$.

Following the same line the case of $\hat{X}_{CC}$, let construct $(\overline{x}_a, \forall a \in \overline{A})$ for $\overline{X}_{CC}$. Define $\overline{x}_a = x_a, \forall a \in A$. For all $i \in V$, $\overline{x}_{(0,i)} = 1$ for $z_i = 1$ and $\overline{x}_{(0,i)} = 0$ otherwise. Similarly, $\overline{x}_{(\acute{0},i)} = 1$ for $y_i = z_i = 0$ and $\overline{x}_{(\acute{0},i)} = 0$ when either $z_i = 1$ or $y_i = 1$. Simply, let define $\overline{x}_{(\acute{0},0)} = 1$.

In Constraint (3.13), we have $\sum_{a\in\delta^+(\acute{0})} \overline{x}_a = n - \sum_{i\in V} y_i = n - k$. In Constraint (3.10), $\sum_{a\in\delta^+(0)} \overline{x}_a = \sum_{i\in V} z_i = 1$. In Constraint (3.14), for $i \in \{i \mid y_i = 1, \forall i \in V\}$, we have $\sum_{a\in\delta^-(i)} \overline{x}_a = \sum_{a\in\delta^-(i)} x_a = y_i = 1$. When $i = \{0\}$, $\sum_{a\in\delta^-(i)} \overline{x}_a = \overline{x}_{(\acute{0},0)} = 1$. For Constraints (3.16), if $\overline{x}_{(\acute{0},i)} = 1$ on a vertex i, we have $\overline{x}_{(i,j)} = 0$ due to the Constraint (3.6) and the definition of $\overline{x}_{(\acute{0},i)}$; $\overline{x}_{(\acute{0},i)} = 1 \leftrightarrow y_i = z_i = 0$. Otherwise, $\overline{x}_{(\acute{0},i)} + \overline{x}_{(i,j)} = \overline{x}_{(i,j)} \leq 1$. Thus, any $\overline{x}_a, \forall a \in \overline{A}$ satisfy Constraints (3.16). Therefore, $(\overline{x}_a, \forall a \in \overline{A})$ can be a feasible solution of $\overline{X}_{CC}$. ∎

Proposition 3.2.2 *Feasible solutions in X_{CC}, $\hat{X}_{CC}$ and $\overline{X}_{CC}$ are composed of connected components with at most one cycle.*

Proof Note that two vertex cycles are not possible in X_{CC} due to constraint (3.6). Because of constraints (3.4), which implies that every vertex in $|V|$ can have only one parent vertex, one can start with any arbitrary vertex and trace back to the root vertex or itself within the connected component. Let C be a connected component that satisfies X_{CC}. Assume that there are two cycles $W_1 \in C$ and $W_2 \in C$. Let $W_1 \cap W_2 \neq \emptyset$; then there is at least one vertex in $W_1 \cap W_2$, whose in-degree is 2. Thus, it contradicts constraint (3.4). Let $W_1 \cap W_2 = \emptyset$; then there is a directed path between $W_1$and W_2, which means that a vertex in W_1 or W_2 has in-degree of 2, which again contradicts constraint (3.4). The proofs for $\hat{X}_{CC}$ and $\overline{X}_{CC}$ follows from Proposition 3.2.1. ∎

Proposition 3.2.2 implies that the connected components are either directed trees or they contain a single cycle that can be broken to construct a directed tree due to the cardinality constraints. This property may facilitate finding KCTP solutions after inserting connectivity and subtour elimination constraints to X_{CC},$\hat{X}_{CC}$, and $\overline{X}_{CC}$. We implement these

additional constraints in two different ways: conservation of flow using a single commodity flow formulation and Miller-Tucker-Zemlin subtour elimination constraints.

3.2.2 Single Commodity Flow (SCF) Constraints

In this section, we propose Single Commodity Flow (SCF) constraints to ensure the connectivity, which would break cycles in the connected components from X_{CC} and produce feasible KCTP solutions. The main idea of SCF is to send a total of k units of flow from a root vertex to k ordinary vertices in the solution, where each ordinary vertex has a unit demand. Connectivity is satisfied through flow balance constraints. Since only k arcs can be selected from A, the resulting solution is constructing a k-cardinality tree
Let variable f_a denote a flow value on arc a. Then the following constraints establish connectivity through conservation of flow in a SCF problem.

$$S_{SCF}: \quad \sum_{a\in\delta^+(i)} f_a \le k(y_i + z_i), \qquad \forall i \in V, \tag{3.19}$$

$$\sum_{a\in\delta^-(i)} f_a - \sum_{a\in\delta^+(i)} f_a = y_i - kz_i, \qquad \forall i \in V, \tag{3.20}$$

$$f_a \le kx_a, \qquad \forall a \in A. \tag{3.21}$$

Constraints (3.19) require that out-flow from a vertex is at most k. Constraints (3.20) enforce that k units of flow are generated at the selected root vertex and one unit of flow is absorbed at a selected ordinary vertex. Constraints (3.21) enforce that at most k units of flow on an arc if the arc is selected, and 0 otherwise. Similar to S_{SCF}, the following set of constraints based on the single commodity flow can be applied to $\hat{X}_{CC}$.

$$\hat{S}_{SCF}: \quad \sum_{a\in\delta^+(i)} f_a \le ky_i, \qquad \forall i \in V, \tag{3.22}$$

$$\sum_{a\in\delta^-(i)} f_a - \sum_{a\in\delta^+(i)} f_a = y_i, \qquad \forall i \in V, \tag{3.23}$$

$$f_a = (k+1)x_a, \qquad \forall a \in \delta^+(0), \tag{3.24}$$

constraints (3.21).

In Constraints (3.22) and (3.23), we rewrite Constraints (3.19) and (3.20) by excluding the root vertex selector variable z_i due to the artificial root vertex 0. Constraints (3.24) require that the selected artificial arc will carry $k+1$ units of flow to be distributed to the selected ordinary vertices.

In $\overline{X}_{CC}$, we only use arc variables x_a. Therefore, the flow conservation constraints for $\overline{X}_{CC}$ is given by

$$\overline{S}_{SCF}: \quad \sum_{a\in\delta^-(i)} f_a - \sum_{a\in\delta^+(i)} f_a = 1, \qquad \forall i \in V, \tag{3.25}$$

$$\sum_{a\in\{\delta^+(\acute{0})\setminus(\acute{0},0)\}} f_a = n-k-1, \tag{3.26}$$

$$\text{constraints (3.21)},$$

$$\text{constraints (3.24)}.$$

Constraints (3.25) require that one unit of flow is absorbed by each vertex in V. Constraint (3.26) ensures that the unit flow demanded by the $n-k-1$ vertices not in the solution tree are supplied by the artificial vertex $\acute{0}$, whereas the demand for the selected vertices is supplied through artificial vertex 0 as in constraint (3.24).

The feasible set of solutions for KCTP for the three different variations satisfy the constraint sets $X_{CC}\cap S_{SCF}$, $\hat{X}_{CC}\cap\hat{S}_{SCF}$ and $\overline{X}_{CC}\cap\overline{S}_{SCF}$, respectively. The cost of the tree is the sum of the arc costs included in the tree, i.e., $\sum_{a\in A} c_a x_a$. Thus we can state the three variations of the SCF formulation of KCTP as follows.

$$K_{SCF} \;:\; \min\{\sum_{a\in A} c_a x_a : (x,y,z,f)\in X_{CC}\cap S_{SCF}\} \tag{3.27}$$

$$\hat{K}_{SCF} \;:\; \min\{\sum_{a\in A} c_a x_a : (x,y,f)\in \hat{X}_{CC}\cap \hat{S}_{SCF}\} \tag{3.28}$$

$$\overline{K}_{SCF} \;:\; \min\{\sum_{a\in A} c_a x_a : (x,f)\in \overline{X}_{CC}\cap \overline{S}_{SCF}\} \tag{3.29}$$

3.2.3 Lifted Miller-Tucker-Zemlin Constraints

One of the most well known subtour elimination constraints was proposed by [60] to solve the traveling salesman problem (TSP). However, this approach requires an exponential number of constraints. [137] proposed Miller-Tucker-Zemlin (MTZ) constraints for subtour elimination by introducing a vertex level variable that represents the number of arcs in the path from a start vertex to the current vertex. However, MTZ constraints are known to produce a weak LP relaxation [117, 146]. [62] proposed lifted MTZ constraints, which provide an improved LP relaxation compared to the original MTZ constraints.

In this subsection, I adopt the lifted MTZ constraints for subtour elimination in three versions of our formulation. We first introduce vertex level variables $\{u_i : i \in V\}$ that represent the number of arcs from the root vertex to vertex i in any feasible tree, which can be interpreted as the depth of a vertex. The resulting lifted MTZ constraints are as follows.

$$
\begin{aligned}
S_{MTZ}: \quad & u_i \le k y_i, && \forall i \in V, && (3.30)\\
& u_i \le k(1 - z_i), && \forall i \in V, && (3.31)\\
& k x_{(i,j)} + (k-2) x_{(j,i)} + u_i - u_j \le k - 1, && \forall (i,j) \in A, && (3.32)\\
& u_i \in \mathbb{R}^+, && \forall i \in V.
\end{aligned}
$$

Constraints (3.30) and (3.31) restrict the depth of a vertex to k or less. Constraint (3.31) set the depth of the root vertex to 0. Constraints (3.32) are the lifted MTZ constraints that set proper depth values to vertices in the tree, which prevent cycles.

We can apply MTZ constraints to $\hat{X}_{CC}$ and $\overline{X}_{CC}$ with minor modifications in terms of the structures of them. $\hat{S}_{MTZ}$ and $\overline{S}_{MTZ}$,which are the lifted MTZ constraints for $\hat{X}_{CC}$ and $\overline{X}_{CC}$, are given as follows.

$$
\begin{aligned}
\hat{S}_{MTZ}: \quad & u_i \le (k+1) y_i, && \forall i \in V, && (3.33)\\
& (k+1) x_{(i,j)} + (k-1) x_{(j,i)} + u_i - u_j \le k, && \forall (i,j) \in A, && (3.34)\\
& (k+1) x_{(i,j)} + u_i - u_j \le k, && \forall (i,j) \in \hat{A} \setminus A, && (3.35)\\
& u_0 = 0, && && (3.36)\\
& u_i \in \mathbb{R}^+, && \forall i \in V.
\end{aligned}
$$

$$
\begin{aligned}
\overline{S}_{MTZ}: \quad & (k+3) x_{(i,j)} + (k+1) x_{(j,i)} + u_i - u_j \le k + 2, && \forall (i,j) \in A, && (3.37)\\
& (k+3) x_{(i,j)} + u_i - u_j \le k + 2, && \forall (i,j) \in \hat{A} \setminus A, && (3.38)\\
& u_{\acute{0}} = 0, && && (3.39)\\
& u_i \in \mathbb{R}^+, && \forall i \in V.
\end{aligned}
$$

Constraints (3.34) and (3.37) are the MTZ constraints for arcs in A. Constraints (3.35) and (3.38) are the MTZ constraints for the arcs in $\hat{\delta}^+(0)$ and $\hat{\delta}^+(\acute{0})$. Constraints (3.36) and (3.39) ensure that the root vertex has 0 depth.

Reformulations of KCTP with MTZ constraints (K_{MTZ},$\hat{K}_{MTZ}$, and $\overline{K}_{MTZ}$) can then be given as follows.

$$K_{MTZ} \quad : \quad \min\{\sum_{a\in A} c_a x_a : (x,y,z,u) \in X_{CC} \cup S_{MTZ}\} \tag{3.40}$$

$$\hat{K}_{MTZ} \quad : \quad \min\{\sum_{a\in A} c_a x_a : (x,y,u) \in \hat{X}_{CC} \cup \hat{S}_{MTZ}\} \tag{3.41}$$

$$\overline{K}_{MTZ} \quad : \quad \min\{\sum_{a\in A} c_a x_a : (x,u) \in \overline{X}_{CC} \cup \overline{S}_{MTZ}\} \tag{3.42}$$

3.2.4 Symmetry Breaking Constraints

A symmetry breaking constraint proposed by [174] significantly reduces the search tree in branch-and-bound or branch-and-cut type algorithms, which can be adopted to our formulations as follows.

$$\textstyle\sum_{j>i} x_{(0,j)} + y_i \leq 1, \qquad \forall i \in V, \tag{3.43}$$

$$\textstyle\sum_{j>i} x_{(0,j)} + (1 - x_{(\acute{0},i)} - x_{(0,i)}) \leq 1, \qquad \forall i \in V. \tag{3.44}$$

Constraints (3.43) impose that the index of the root vertex is the smallest of the indices of the selected vertices, which correspond to one of the $k+1$ symmetric solutions. Similar to Constraints (3.43), Constraints (3.44) forces the root vertex to start with smallest index among all asymmetric feasible solutions. Note that if both $x_{(\acute{0},i)}$ and $x_{(0,i)}$ are equal to zero, the second component on the left-hand-side of Constraint (3.44) implies that vertex i is a child vertex as opposed to the root vertex in the feasible tree. The first component is the sum of the arcs with index $j > i$ that determines the root vertex, and it is equivalent to the first component of (3.43). These constraints are generic to all KCTP implementations except for K_{SCF} and K_{MTZ} due to the absence of the artificial root vertex $\{0\}$. Therefore, I include Constraints (3.43) to both $\hat{K}_{SCF}$ and $\hat{K}_{MTZ}$ and Constraints (3.44) to both $\overline{K}_{SCF}$ and $\overline{K}_{MTZ}$, respectively.

3.2.5 Polyhedral Study

In this section, I study the polyhedral characteristics of our formulations. In the following, let P_{SCF}, $\hat{P}_{SCF}$, and $\overline{P}_{SCF}$ be the polyhedrons corresponding to the LP-relaxation for

K_{SCF}, $\hat{K}_{SCF}$, and $\overline{K}_{SCF}$ respectively. Analogously, P_{MTZ}, $\hat{P}_{MTZ}$, and $\overline{P}_{MTZ}$ be the polyhedrons corresponding to the LP-relaxation for K_{MTZ}, $\hat{K}_{MTZ}$, and $\overline{K}_{MTZ}$.

Each polyhedron P includes all feasible solutions for each formulation with replacement of integer properties to the continuous properties. e.g., $P_{SCF} = \{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathbb{R}_{+}^{|A|+2|V|} | \text{Const. } (3.1) - (3.5), \text{Const. } (3.19) \text{ - } (3.21),\ 0 \leq x_a, y_i, z_i \leq 1\}$.

Additionally, let $z_{LP}(.)$ denote the optimal solution value of the LP-relaxation of a given model. For example, $z_{LP}(K_{SCF}) = \min \sum_a c_a x_a$ where $x_a \in P_{SCF}$. Therefore, the objective value of the optimal solution within the relaxed formulation which might contains fractional values is a lower bound for the optimal integer solution in its original formulation containing integer property (e.g., $z(P_{SCF}) \leq z(K_{SCF})$). We consider a stronger formulation when its optimal LP solution gives a tighter lower bound than the other. For example, suppose there exist two MIP formulations K_1 and K_2 on the same problem (both of them have same optimal solution, $z(K_1) = z(K_2)$) and their corresponding LP relaxations are P_{K_1} and P_{K_2}. Then, $P_{K_1} \subseteq P_{K_2}$ if and only if $z(P_{K_1}) \geq z(P_{K_2})$. We can consider the K_1 stronger since its LP-solution gives a tighter lower bound than K_2.

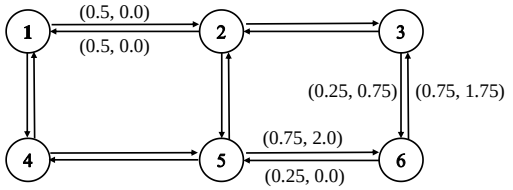
Proposition 3.2.3 *Both formulation K_{SCF} and $\hat{K}_{SCF}$ have same LP relaxation tightness.*

Proof Equivalence of LP relaxation can be proved by: $\hat{P}_{SCF} = \Pi_1(P_{SCF})$, where Π is a projection of P_{SCF} onto the $\hat{P}_{SCF}$, i.e., $(\mathbf{x}^*,\mathbf{y}^*,\mathbf{z}^*,\mathbf{f}^*) \overset{\Pi}{\rightarrow} (\hat{\mathbf{x}}^*, \hat{\mathbf{y}}^*, \hat{\mathbf{f}}^*)$ where $(\mathbf{x}^*,\mathbf{y}^*,\mathbf{z}^*,\mathbf{f}^*)$ and $(\hat{\mathbf{x}}^*, \hat{\mathbf{y}}^*, \hat{\mathbf{f}}^*)$ are feasible points of P_{SCF} and $\hat{P}_{SCF}$ respectively.

Prove $\Pi_1(P_{SCF}) \subseteq P(\hat{P}_{SCF})$ first. Consider a projection Π such that $\hat{x}_a^* = x_a^*$, $\forall a \in A$. $\hat{x}_{(0,i)}^* = z_i^*, \forall i \in V$. $\hat{y}_i^* = y_i^* + z_i^*$. Similarly, $\hat{f}_a^* = f_a^*, \forall a \in A$. $\hat{f}_{(0,i)}^* = (k+1)z_i^*, \forall i \in V$. First of all, constructed values onto the space of $\hat{P}_{SCF}$ $(\hat{\mathbf{x}}^*,\hat{\mathbf{y}}^*,\hat{\mathbf{z}}^*,\hat{\mathbf{f}}^*)$ satisfy Constraints in $\hat{X}_{CC}$ by Proposition 3.2.1. Next, $\hat{\mathbf{f}}^*$ satisfy Constraints (3.22) since $\hat{y}_i^* = y_i^* + z_i^*$. $\sum_{a \in \delta^-(i)} \hat{f}_a^* - \sum_{a \in \delta^+(i)} \hat{f}_a^* = \sum_{a \in \delta^-(i) \setminus \{(0,i)\}} f_a^* + \hat{f}_{(0,i)}^* - \sum_{a \in \delta^+(i)} f_a^* = y_i^* - kz_i^* + (k+1)z_i^* = y_i^* + z_i^* = \hat{y}_i^*$. Constraints (3.24) is already defined in the above transformation. Thus, any points projected from the feasible points in P_{SCF} are also feasible in $\hat{P}_{SCF}$, i.e.,$P(K_{SCF}) \subseteq P(\hat{K}_{SCF})$

Prove $\hat{P}_{SCF} \subseteq \Pi_1(P_{SCF})$. Given any feasible points $(\hat{\mathbf{x}}^*, \hat{\mathbf{y}}^*, \hat{\mathbf{f}}^*)$ in $\hat{P}_{SCF}$, we can simply obtain: $x_a^* = \hat{x}_a^*, \forall a \in A$. $z_i^* = \hat{x}_{(0,i)}^*, \forall i \in V$. $y_i^* = \hat{y}_i^* - \hat{x}_{(0,i)}^*$. $f_a^* = \hat{f}_a^*, \forall a \in A$. Similarly,

constructed ($\mathbf{x}^*$,$\mathbf{y}^*$,$\mathbf{z}^*$) satisfy Constraints in X_{CC}. For $\mathbf{f}^*$, it satisfies Constraints (3.19) because $y_i^* + z_i^* = \hat{y}_i^*$. In Constraints (3.20), $\sum_{a\in\delta^-(i)} f_a^* - \sum_{a\in\delta^+(i)} f_a^* = \sum_{a\in\delta^-(i)} \hat{f}_a^* - \sum_{a\in\delta^+(i)} \hat{f}_a^* - \hat{f}_{(0,i)}^* = \hat{y}_i^* - \hat{f}_{(0,i)}^* = (y_i^* - z_i^*) - (k+1)z_i^* = y_i^* - kz_i^*$. Thus, from any feasible points in $\hat{P}_{SCF}$, I can obtain feasible points in P_{SCF}. Hence, both P_{SCF} and $\hat{P}_{SCF}$ are polytope-wise equivalent. ∎



i	$(\overline{x}^*_{(0,i)}, \overline{f}^*_{(0,i)})$	$(\overline{x}^*_{(\acute{0},i)}, \overline{f}^*_{(\acute{0},i)})$
1	(0, 0)	(0.5, 1)
2	(0.25, 1)	(0.25, 0)
3	(0, 0)	(0.25, 0)
4	(0, 0)	(1, 1)
5	(0.75, 3)	(0, 0)
6	(0, 0)	(0, 0)

Figure 3.1: A sample instance for proof of Proposition 3.2.4. Arc costs $c_{(1,2)} = c_{(3,6)} = c_{(5,6)} = 1$ and remaining costs are equal to 10. Note that the solution in the figure is generated by the case of $k = 3$.

Proposition 3.2.4 *K_{SCF} is stronger than $\overline{K}_{SCF}$, i.e., $z_{LP}(K_{SCF}) \geq z_{LP}(\overline{K}_{SCF})$.*

Proof Prove $\Pi_2(P_{SCF}) \subset \overline{P}_{SCF}$ first. Based on the feasible points of $P(K_{SCF})$, ($\mathbf{x}^*$,$\mathbf{y}^*$,$\mathbf{z}^*$,$\mathbf{f}^*$), we define a projection Π_2 such that $\overline{x}_a = x_a$, $\overline{x}^*_{(0,i)} = z_i^*$ and $\overline{x}^*_{(\acute{0},i)} = 1 - (y_i^* + z_i^*)$. From the Proposition 3.2.1, $\overline{\mathbf{x}}^*$ satisfy the constraints in $\overline{X}_{CC}$. For $\overline{\mathbf{f}}^*$, let $\overline{f}_a^* = f_a^*$, $\forall a \in A$, $\overline{f}^*_{(0,i)} = (k+1)z_i^*$, and $\overline{f}^*_{(\acute{0},i)} = \overline{x}^*_{(\acute{0},i)}$.

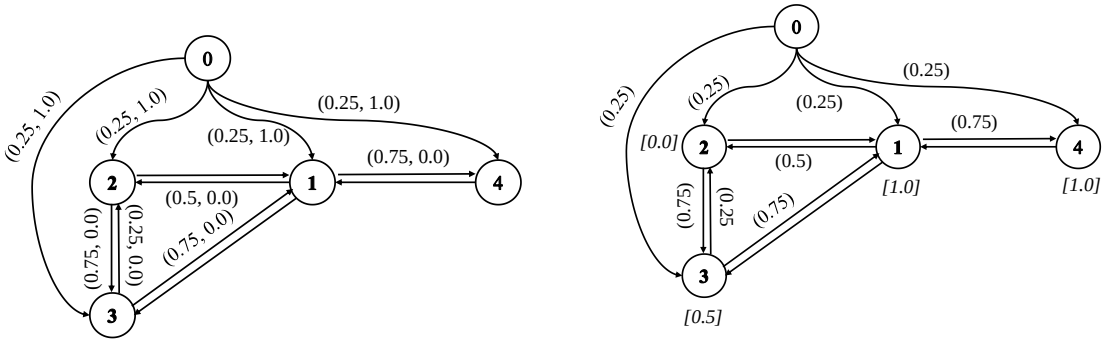
In Constraints (3.25), $\sum_{a\in\delta^-(i)} \overline{f}_a^* - \sum_{a\in\delta^+(i)} \overline{f}_a^* = \sum_{a\in\delta^-(i)\cap A} \overline{f}_a^* + \overline{f}^*_{(0,i)} + \overline{f}^*_{(\acute{0},i)} - \sum_{a\in\delta^+(i)} \overline{f}_a^* = \sum_{a\in\delta^-(i)\cap A} f_a^* - \sum_{a\in\delta^+(i)} f_a^* + \overline{f}^*_{(0,i)} + \overline{f}^*_{(\acute{0},i)} = y_i^* - kz_i^* + (k+1)z_i^* + 1 - (y_i^* + z_i^*) = 1$.

In Constraint (3.26), $\sum_{a\in\{\delta^+(\acute{0})\setminus(\acute{0},0)\}} \overline{f}_a^* = \sum_{a\in\{\delta^+(\acute{0})\setminus(\acute{0},0)\}} \overline{x}^*_{(\acute{0},i)} = \sum_{i\in V}(1 - (y_i^* + z_i^*)) = n - \sum_{i\in V}(y_i^* + z_i^*)) = n - k - 1$.

On the other hand, a traditional argument is introduced in order to identify fractional solution feasible for $\overline{K}_{SCF}$, but infeasible for K_{SCF}. In Figure 3.1, assume a grid graph on 6 vertices. On the the left panel of Figure 3.1, a feasible solution $(\overline{x}_a^*, \overline{f}_a^*)$ for all $a \in A$. For all $a \in \overline{A} \setminus A$, we make a table for $(\overline{x}_a^*, \overline{f}_a^*)$ at the right panel of Figure 3.1. These values are

feasible for $\overline{K}_{SCF}$. However, for K_{SCF}, there is no possible y_i and z_i satisfying Constraints (3.6), Constraints (3.2), and Constraint (3.3) simultaneously. By use of $\overline{x}^*_a$ in Figure 3.1, every corresponding nodes i of arc a whose $\overline{x}^*_a > 0$ has to satisfy $\overline{x}^*_{(i,j)} + \overline{x}^*_{(i,j)} \leq (y^*_i + z^*_i)$ according to Constraints (3.6). Thus, we have $z_i + y_i = 1$ for $i \in \{1, 2, 3, 5, 6\}$. Thus, we have $\sum_{i \in V}(y^*_i + z^*_i) = 5$ which contradicts to Constraints (3.2) and Constraint (3.3). ∎

Remark $P_{SCF} = \hat{P}_{SCF} \subseteq \overline{P}_{SCF}$. Thus, $z_{LP}(K_{SCF}) = z_{LP}(\hat{K}_{SCF}) \geq z_{LP}(\overline{K}_{SCF})$



(a) Proposition 3.2.5

(b) Proposition 3.2.8

Figure 3.2: A sample instance for proofs of Proposition 3.2.5 and 3.2.8. Arc costs are all equal to 1 except $c_{(1,4)} = 10$

Proposition 3.2.5 *Constraints* (3.43) *and* (3.44) *strengthen* $\hat{K}_{SCF}$ *and* $\overline{K}_{SCF}$ *respectively.*

Proof For notational convenience, let $\hat{P}^s_{SCF}$ denote the LP relaxation polyhedron including the symmetry breaking constraints (3.43). Consider Constraints (3.43) first. Obviously, any feasible solution in $\hat{P}^s_{SCF}$ is feasible in $\hat{P}_{SCF}$.

One the other hand, we use a same argument used in the proof of Proposition 3.2.4. In Figure 3.2(a), note that feasible $\hat{y}^*_i = 1$ for all $i \in \{1, 2, 3, 4\}$ by Constraints (3.9). Hence, we only have $x_{(0,1)} = 1$ and $\hat{x}^*_{(0,2)} = \hat{x}^*_{(0,3)} = \hat{x}^*_{(0,4)} = 0$ to satisfy symmetry breaking constraints (3.43). Thus, the feasible solution of $P(\hat{K}_{SCF})$ in Figure 3.2(a) is infeasible in $\hat{P}^s_{SCF}$.

Along the same line, let $\overline{P}^s_{SCF}$ be a polyhedron including Constraints (3.44). From the feasible solution $(\hat{\mathbf{y}}^*, \hat{\mathbf{x}}^*, \hat{\mathbf{f}}^*)$ of $\hat{K}_{SCF}$ in Figure 3.2(a), we construct a feasible solution of

$\overline{K}_{SCF}$ such that $\overline{x}^*_a = \hat{x}^*_a$ and $\overline{f}^*_a = \hat{f}^*_a$, $\forall a \in \overline{A} \cap \hat{A}$. Simply we set $\overline{x}^*_{(\acute{0},i)} = \overline{f}^*_{(\acute{0},i)} = 0$. Similar to the case of $P'(\hat{K}_{SCF})$, constructed $(\overline{\mathbf{x}}^*, \overline{\mathbf{f}}^*)$ is infeasible of $P'(\overline{K}_{SCF})$. For example, when $i = 1$ in Constraint (3.44), $\overline{x}^*_{(0,2)} + \overline{x}^*_{(0,3)} + \overline{x}^*_{(0,4)} + 1 - \overline{x}^*_{(\acute{0},1)} - \overline{x}^*_{(0,1)} = 0.75 + 1 - 0 - 0.25 = 1.5 > 1$. Thus, $(\overline{\mathbf{x}}^*, \overline{\mathbf{f}}^*)$ is not feasible in $\overline{P}^s_{SCF}$. ∎

Similar to the case of above three formulations based on the SCF, I compare the polyhedral strength within three MTZ based formulations: K_{MTZ}, $\hat{K}_{MTZ}$, and $\overline{K}_{MTZ}$.

Proposition 3.2.6 *Both formulation K_{MTZ} and $\hat{K}_{MTZ}$ have same LP relaxation tightness.*

Proof Prove $P_{MTZ} \subseteq \hat{P}_{MTZ}$ first. Let ($\mathbf{x}^*$,$\mathbf{y}^*$,$\mathbf{z}^*$,$\mathbf{u}^*$) be an optimal solution of LP relaxation K_{MTZ}. Then, we have $\hat{x}^*_a = x^*_a$, $\forall a \in A$. $\hat{x}^*_{(0,i)} = z^*_i$, $\forall i \in V$. $\hat{y}^*_i = y^*_i + z^*_i$. Every ($\mathbf{x}^*$,$\mathbf{y}^*$,$\mathbf{z}^*$) satisfy the constraints in $\hat{X}_{CC}$. Thus, it is sufficient to show that we are always able to find a $\mathbf{u}^*$ satisfying MTZ constraints in $\hat{S}_{MTZ}$. For an arc $a = (i, j) \in A$, we have following linear inequalities:

$$\hat{u}_i \leq (k+1)(y^*_i + z^*_i) \tag{3.45}$$

$$\hat{u}_j \leq (k+1)(y^*_j + z^*_j) \tag{3.46}$$

$$\hat{u}_i - \hat{u}_j \leq k - (k+1)x^*_{(i,j)} - (k-1)x^*_{(j,i)} \tag{3.47}$$

$$\hat{u}_i \geq \max\{(k+1)z^*_i - k, 0\} \tag{3.48}$$

$$\hat{u}_j \geq \max\{(k+1)z^*_j - k, 0\} \tag{3.49}$$

First of all, $\max\{(k+1)z^*_i - k, 0\} \leq (k+1)(y^*_i + z^*_i)$ is established. Thus, for i, we have $\max\{(k+1)z^*_i - k, 0\} \leq \hat{u}_i \leq (k+1)(y^*_i + z^*_i)$. Next, we can confirm that there exist feasible $\mathbf{u}^*$ unless the above linear inequalities are infeasible. The above inequalities are infeasible if and only if $k - (k+1)x^*_{(i,j)} - (k-1)x^*_{(j,i)} < \max\{(k+1)z^*_i - k, 0\} - (k+1)(y^*_j + z^*_j)$. However, according to the Constraints (3.6), we have $k-(k+1)x^*_{(i,j)}-(k-1)x^*_{(j,i)}-(\max\{(k+1)z^*_i - k, 0\} - (k+1)(y^*_j + z^*_j)) = 2k - (k+1)(x^*_{(i,j)} + x^*_{(j,i)} - y^*_j) + (k+1)(z^*_i - z^*_j) - 2x^*_{(j,i)} \geq 0$. Note that $k \geq 1$ always. Thus, we always find a feasible $\mathbf{u}^*$ based on the ($\mathbf{x}^*$,$\mathbf{y}^*$,$\mathbf{z}^*$).

Next, prove $\hat{P}_{MTZ} \subseteq P_{MTZ}$. First $x^*_a = \hat{x}^*_a$, $\forall a \in A$. $z^*_i = \hat{x}^*_{(0,i)}$, $\forall i \in V$. $y^*_i = \hat{y}^*_i - \hat{x}^*_{(0,i)}$ is

feasible in X_{CC}. Secondly, we have the following inequalities:

$$u_i \leq \min\{k(\hat{y}^*_i - \hat{x}^*_{(0,i)}), k(1 - \hat{x}^*_{(0,i)})\} \tag{3.50}$$

$$u_j \leq \min\{k(\hat{y}^*_j - \hat{x}^*_{(0,j)}), k(1 - \hat{x}^*_{(0,j)})\} \tag{3.51}$$

$$u_i - u_j \leq k - 1 - k\hat{x}^*_{(i,j)} - (k-2)\hat{x}^*_{(j,i)} \tag{3.52}$$

$$\hat{u}_i \geq 0 \tag{3.53}$$

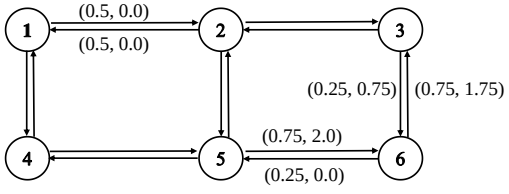
$$\hat{u}_j \geq 0 \tag{3.54}$$

Similarly, the above linear inequalities are infeasible if and only if $k-1-k\hat{x}^*_{(i,j)}-(k-2)\hat{x}^*_{(j,i)} < -\min\{k(\hat{y}^*_j - \hat{x}^*_{(0,i)}), k(1 - \hat{x}^*_{(0,j)})\}$.

Suppose $\min\{k(\hat{y}^*_j - \hat{x}^*_{(0,j)}), k(1 - \hat{x}^*_{(0,j)})\} = k(\hat{y}^*_j - \hat{x}^*_{(0,j)})$ first, then $k - 1 - k\hat{x}^*_{(i,j)} - (k-2)\hat{x}^*_{(j,i)} + k(\hat{y}^*_j - \hat{x}^*_{(0,j)}) = k(y^*_j - \hat{x}^*_{(j,i)}) - \hat{x}^*_{(i,j)}) + k - k\hat{x}^*_{(0,j)} + 2\hat{x}^*_{(j,i)} = k(y^*_j - \hat{x}^*_{(j,i)} - \hat{x}^*_{(i,j)}) + k - k\hat{x}^*_{(0,i)} - 1 + 2\hat{x}^*_{(j,i)} = k(y^*_j - \hat{x}^*_{(j,i)} - \hat{x}^*_{(i,j)} - \hat{x}^*_{(0,j)}) + k - 1 + 2\hat{x}^*_{(j,i)} \geq 0$ due to the Constraints (3.9) and (3.11).

Otherwise, when $\min\{k(\hat{y}^*_j - \hat{x}^*_{(0,j)}), k(1 - \hat{x}^*_{(0,j)})\} = k(1 - \hat{x}^*_{(0,j)})$, then
$k-1-k\hat{x}^*_{(i,j)}-(k-2)\hat{x}^*_{(j,i)}+k(1-\hat{x}^*_{(0,j)}) = k(2-\hat{x}^*_{(i,j)}-\hat{x}^*_{(j,i)}-\hat{x}^*_{(0,j)})+2\hat{x}^*_{(j,i)}-1 \geq 0$ due to the Constraints (3.9) and (3.11). Thus, both formulation have equally strong LP-relaxation.

∎



i	$\overline{x}^*_{(0,i)}$	$\overline{x}^*_{(\hat{0},i)}$	$\overline{u}^*_i$
1	0	0.5	1.0
2	0.25	0.25	1.0
3	0	0.25	5.0
4	0	1	1.0
5	0.75	0	4.0
6	0	0	4.5

Figure 3.3: A sample instance for proof of Proposition 3.2.6. $c_{(i,j)}$ is a edge cost of an edge (i,j). $k = 3$

Proposition 3.2.7 *K_{MTZ} is stronger than $\overline{K}_{MTZ}$, i.e., $z_{LP}(K_{MTZ}) \geq z_{LP}(\overline{K}_{MTZ})$.*

Proof Based on the feasible solution of P_{MTZ}; ($\mathbf{x}^*$, $\mathbf{y}^*$, $\mathbf{z}^*$, $\mathbf{u}^*$), we transform $\overline{x}_a = x_a$, $\overline{x}^*_{(0,i)} = z^*_i$ and $\overline{x}^*_{(\acute{0},i)} = 1 - (y^*_i + z^*_i)$. From the Proposition 3.2.1, $\overline{\mathbf{x}}^*$ satisfy the constraints in $\overline{X}_{CC}$. In contrast to the proof of 3.2.6, which is shown by general feasibility of the linear inequalities for $\overline{u}^*_i$, let consider a projection $\overline{u}^*_i = u^*_i + 2(y^*_i + z^*_i)$.

In Constraints (3.37), $(k+3)x^*_{(i,j)} + (k+1)x^*_{(j,i)} + u^*_i + 2(y^*_i + z^*_i) - u^*_j - 2(y^*_j + z^*_j) = kx^*_{(i,j)} + (k-2)x^*_{(j,i)} + u^*_i - u^*_j + 3(x^*_{(i,j)} + x^*_{(j,i)}) + 2(y^*_i + z^*_i) - 2(y^*_j + z^*_j)$.

Since $kx^*_{(i,j)} + (k-2)x^*_{(j,i)} + u^*_i - u^*_j \leq k-1$ and $3(x^*_{(i,j)} + x^*_{(j,i)}) + 2(y^*_i + z^*_i) - 2(y^*_j + z^*_j) \leq 5(y^*_i + z^*_i) - 2(y^*_j + z^*_j) \leq 3$, thus we can find feasible $\overline{u}^*_i$ satisfy Constraints (3.37) through the mapping.

In Constraints (3.37), $(k+3)z^*_j - u^*_j - 2(y^*_j + z^*_j) = (k+1)z^*_j - u^*_j - 2y^*_j \leq k+2$.

On the other hand, similar to the process for the proof of Proposition 3.2.4, we provide an example which contains a feasible solution of $\overline{P}_{MTZ}$, but infeasible for P_{MTZ}.

In Figure 3.3, it is equivalent instance used in Figure 3.1. We obtain a same feasible solution $\overline{\mathbf{x}}^*$ of $\overline{P}_{MTZ}$ as the solution of $\overline{P}_{SCF}$. On the right panel, we make a table for $(\overline{x}^*_a, \overline{u}^*_i)$ as a reference. Equivalently to the Proposition 3.2.3, $\overline{\mathbf{x}}^*$ is infeasible while violating Constraints (3.6), Constraints (3.2), and Constraint (3.3) in X_{CC} ∎

Remark $z_{LP}(K_{MTZ}) = z_{LP}(\hat{K}_{MTZ}) \geq z_{LP}(\overline{K}_{MTZ})$

Proposition 3.2.8 *Constraints* (3.43) *and* (3.44) *strengthen* $\hat{K}_{MTZ}$ *and* $\overline{K}_{MTZ}$ *respectively.*

Proof Along the same vein as the proof of Proposition 3.2.5, feasible solution $\hat{x}^*_a$ making all $\hat{y}^*_i = 1$ cannot satisfying the symmetry breaking constraints (3.43). Also, we derive the feasible solution of $\overline{K}_{CC}$ from the solution in Figure 3.2(b) such that let set $\overline{x}^*_a = \hat{x}^*_a$ and $\overline{x}^*_{(0,i)} = \hat{x}^*_{(0,1)}$. By Constraint (3.14), we can calculate $\hat{x}^*_{(\acute{0},1)}$ as a result. We omit the verification of For $i = 1$, in Constraint (3.44), $\overline{x}^*_{(0,2)} + \overline{x}^*_{(0,3)} + \overline{x}^*_{(0,4)} + 1 - \overline{x}^*_{(\acute{0},1)} - \overline{x}^*_{(0,1)} = 0.75 + 1 - 0 - 0.25 = 1.5 > 1$ so we can verify that $\overline{\mathbf{x}}^*$ is feasible for $\overline{K}_{MTZ}$, but strictly violates the symmetry breaking constraints. ∎

3.3 A Reformulation-Linearization Technique Approach

Shearli and Driscoll introduced tighter relaxations of the MTZ constraints for the asymmetric TSP using the reformulation-linearization technique (RLT) [173]. They started with a nonstandard restatement of MTZ constraints that involved nonlinear product terms and then applied a specialized version of the RLT [170, 171], exploiting inherent special structures as in [172]. Most of the development in their work can be adapted for $\overline{K}_{MTZ}$ with the following reformulation.

$$\overline{K}_{MTZ2}: \quad \text{Minimize} \quad \textstyle\sum_{a\in A} c_a x_a$$

subject to:

$$x \in \overline{X}_{CC}$$

constraints (3.39),

$$u_j x_{(i,j)} = (u_i + 1)x_{(i,j)}, \qquad \forall (i,j) \in \overline{A}, \tag{3.55}$$

$$u_0 = 1, \tag{3.56}$$

$$1 \le u_i \le k + 2, \qquad \forall i \in V, \tag{3.57}$$

Nonlinear constraints (3.55) are subtour elimination constraints. For vertices $\{0, \acute{0}\}$ in $\overline{K}_{MTZ2}$, I can observe $u_j x_{(\acute{0},j)} = x_{(\acute{0},j)}$ and $u_j x_{(0,j)} = 2x_{(0,j)}$ from Constraints (3.39) and (3.56), respectively. Since there are k arcs in the feasible tree, the upper bound of u_i is $k + 2$. These bounds are given in Constraints (3.57).

Next, I reformulate $\overline{K}_{MTZ2}$ by generating additional implied constraints and linearizing nonlinear terms by substitution of distinct variables that enforce the nonlinear relationships.

3.3.1 Reformulation Phase

In the reformulation phase, I construct additional sets of constraints via steps (R1)–(R4) stated below.

(R1) Using the enforcement in-degree constraints (3.14), I construct the following valid

inequalities:

$$u_i \left(\sum_{a \in \delta^-(i)} x_a - 1 \right) = 0, \quad \forall i \in N \cup \{0\}. \tag{3.58}$$

(R2) For each i, I multiply the inequality $(u_i - 1) \geq 0$ of Constraint (3.57) by (i) $x_{(i,j)} \geq 0$ for each $(i,j) \in A$ and by (ii) $(1 - x_{(i,j)} - x_{(j,i)}) \geq 0$ for each $(i,j) \in A$. Note that these multiplications yield inequalities of type (i):

$$(u_i - 1)x_{(i,j)} \geq 0, \quad \forall (i,j) \in A, \tag{3.59}$$

and type (ii):

$$(u_i - 1)(1 - x_{(i,j)} - x_{(j,i)}) \geq 0, \quad \forall (i,j) \in A \tag{3.60}$$

Observe that the inequalities used as multipliers in (ii) are the two vertex cycle elimination constraints. Since $x_{(i,j)} + x_{(j,i)} \leq 1$ and $x_{(i,j)}, x_{(j,i)} \geq 0$ imply $0 \leq x_{(i,j)} \leq 1$ and $0 \leq x_{(j,i)} \leq 1\}$, though not vice-versa, the use of factors $(1 - x_{(i,j)} - x_{(j,i)})$ generates potentially tighter relaxations than that obtained using the simple bound constraints $(1 - x_{(i,j)})$ and $(1 - x_{(j,i)})$ as studied generally in [172].

(R3) Similar to (R2), using the upper bound conditions in Constraints (3.57), I construct valid inequalities type (i):

$$(k + 2 - u_i)x_{(i,j)} \geq 0, \quad \forall (i,j) \in \hat{A}, \tag{3.61}$$

and type (ii):

$$(k + 2 - u_i)(1 - x_{(i,j)} - x_{(j,i)}) \geq 0, \quad \forall (i,j) \in A. \tag{3.62}$$

(R4) For each vertex i which are all non-root vertex and have $u_i \geq 3$, I construct the following valid inequalities:

$$(u_i - 3)(1 - x_{(\acute{0},i)} - x_{(0,i)}) \geq 0, \quad \forall i \in N \tag{3.63}$$

Note that when $x_{(\acute{0},i)} = 1$ or $x_{(0,i)} = 1$, these constraints are trivial valid inequalities. Likewise, $x_{(\acute{0},i)} = 0$ and $x_{(0,i)} = 0$, since I must then have $3 \leq u_i \leq k + 2$, these constraints are again valid inequalities.

3.3.2 Linearization Phase

In the linearization phase, I linearize $\overline{K}_{MTZ2}$ along with new classes of constraints from (R1) to (R4) generated above by using the substitution

$$y_{(i,j)} = u_i x_{(i,j)} \text{ and } z_{(i,j)} = u_j x_{(i,j)}, \quad \forall (i,j) \in \hat{A}. \tag{3.64}$$

Note that Constraints (3.55) together with (3.57) imply,

$$z_{(i,j)} = x_{(i,j)} + y_{(i,j)}, \quad \forall (i,j) \in \hat{A}. \tag{3.65}$$

Subsequently, $z_{(i,j)}$ in Constraint (3.64) can be eliminated by substitution using Constraint (3.65), and it finally yields the following formulation:

K_{RLT} :

Minimize $\sum_{a \in A} c_a x_a$

subject to:

$$x \in \overline{X}_{CC} \tag{3.66}$$

$$\sum_{(i,j) \in \hat{A}} y_{(i,j)} + 1 = u_j, \qquad \forall j \in N, \tag{3.67}$$

$$y_{(i,j)} \geq x_{(i,j)}, \qquad \forall (i,j) \in A, \forall i \in N \cup \{0\}, \tag{3.68}$$

$$(k+1)x_{(i,j)} \geq y_{(i,j)}, \qquad \forall (i,j) \in A, \forall i \in N \cup \{0\}, \tag{3.69}$$

$$\begin{aligned}(k+1)x_{(i,j)} + (k+2)x_{(j,i)} + u_j \\ \leq y_{(i,j)} + y_{(j,i)} + k + 2, \qquad \forall (i,j) \in A,\end{aligned} \tag{3.70}$$

$$(k+1)x_{(i,j)} + u_j \leq y_{(i,j)} + k + 2, \qquad i \in \{\acute{0}, 0\},\ j \in N, \tag{3.71}$$

$$y_{(i,j)} + y_{(j,i)} \leq x_{(j,i)} + u_j - 1, \qquad \forall (i,j) \in A, \tag{3.72}$$

$$y_{(i,j)} \leq u_j - 1, \qquad i \in \{\acute{0}, 0\},\ j \in N, \tag{3.73}$$

$$u_j + 2x_{(\acute{0},j)} + x_{(0,j)} \geq 3, \qquad \forall j \in N, \tag{3.74}$$

constraints (3.39),(3.56), and (3.57)

Under the linearization in Eq. (3.64) and the substitution Eq. (3.65), the reformulation step (R1) yields Constraint (3.67). Step (R2)(i) and inequalities (R3)(i) provide lower and upper bounding restrictions in Constraints (3.68) and (3.69), respectively. Similarly, (R2)(ii) and (R3)(ii) produce lower and upper bounding restrictions in Constraints (3.70) and (3.72), respectively. Since there does not exist arc (j, i), where $j \in N$ and $i \in \{\acute{0}, 0\}$, we distinguish Constraints (3.71) and (3.73) from Constraints (3.70) and (3.72) separately in terms of arcs' condition. The set of constraints generated at step (R4) produce bounding inequalities given in Constraint (3.74). Note that the new variable $y_{(i,j)}$ represents the depth of the arc (i, j) in the tree from the root vertex if it appears in the solution, 0 otherwise.

The interpretation of each constraint is thereby evident by considering a possible $0-1$ value for the accompanying x variable. For instance, consider Constraints (3.70) and (3.72), which are lower and upper bounds for $y_{(i,j)} + y_{(j,i)}$, respectively and suppose that $x_{(i,j)} = 1$ and $x_{(j,i)} = 0$ for $(i, j) \in A$. Then Constraints (3.70) and (3.72) reduce to $y_{(i,j)} + y_{(j,i)} = u_j - 1$. Similarly, for (i, j) where $i \in \{\acute{0}, 0\},\ j \in N$, $y_{(i,j)} = u_j - 1$ is established by Constraints (3.71) and (3.73). The interpretation of y variables confirms this relationship since we must have $y_{(j,i)} = 0$ due to Constraint (3.69). According to the original definition of $y_{(i,j)}$ in Constraint (3.64), $y_{(i,j)} = u_i$ when $x_{(i,j)} = 1$. Consequently, it leads to the valid equation: $u_i = u_j - 1$.

We next show the validity of the K_{RLT} formulation and also show that $\overline{K}_{MTZ}$ is a surrogate relaxation of $\overline{K}_{MTZ}$, which implies that K_{RLT} yields a tighter LP relaxation than that of $\overline{K}_{MTZ}$. We further discuss the conditions under which the constraints of K_{RLT} are tighter than the constraints of $\overline{K}_{MTZ}$.

Proposition 3.3.1 *The formulation K_{RLT} is a valid model for KCTP and $\overline{K}_{MTZ}$ is a surrogate relaxation of K_{RLT}.*

Proof The validity of the constraints described in K_{RLT} follows by construction and arguments presented in the reformulation phase of the RLT process. Therefore, it is sufficient to show that Constraints (3.66)-(3.74) imply the constraints of $\overline{K}_{MTZ}$ in the continuous sense. Note that Constraints (3.66) are common. Hence, it remains to show that Constraints (3.37) and (3.38) are implied by Constraints (3.67)-(3.74). For any arc $(i, j) \in A$, let us examine

Constraints (3.70), (3.72), and (3.15). By swapping i and j in the first, multiplying the second by -1, and finally surrogating all of them, Constraint (3.37) can be obtained. The case of arc (i, j), $i \in \{\acute{0}, 0\}, j \in N$, yields the same equivalence to Constraint (3.37) similarly if we replace arc (j, i) with 0. This result shows that $\overline{K}_{MTZ}$ is a surrogate relaxation of K_{RLT}. ∎

Fixing all other variables, the relaxed feasible region for $x_{(i,j)}$ and $x_{(j,i)}$ would be a unit square as shown in Figure 3.4. We next provide a detailed comparison of the feasible region defined by Constraints (3.75) and (3.76) to the region defined by Constraint (3.77) within this unit square. By using the equality $y_{(i,j)} = u_i x_{(i,j)}$ from Constraint (3.64), we first rearrange Constraint (3.70) (after swapping i and j), and Constraint (3.72) as in Constraints (3.75) and (3.76), respectively. As a result, we obtain

$$x_{(i,j)} + \tfrac{k+1-u_j}{k+2-u_i} x_{(j.i)} \leq 1, \tag{3.75}$$

$$\tfrac{u_i}{u_j-1} x_{(i,j)} + x_{(j,i)} \leq 1. \tag{3.76}$$

Similarly, we rearrange Constraint (3.37) as in Constraint (3.77) and obtain

$$\tfrac{k+3}{k+2+u_j-u_i} x_{(i,j)} + \tfrac{k+1}{k+2+u_j-u_i} x_{(j,i)} \leq 1. \tag{3.77}$$

For $u_j \geq u_i + 1$, it is easy to check that the region defined by Constraint (3.15), which is $x_{(i,j)} + x_{(j.i)} \leq 1$, is tighter than the region defined by Constraints (3.75) and (3.76), and also tighter than the region defined by Constraint (3.77). Note that Constraint (3.15) is common to both K_{RLT} and $\overline{K}_{MTZ}$.

Next we show that for $u_j \leq u_i - 1$, the feasible region defined by Constraints (3.75) and (3.76) is strictly tighter than the region defined by Constraint (3.77), which are shown as the quadrilateral $ABCD$ (gray area) and the triangle AEF (light gray), respectively in Figure 3.4.

Proposition 3.3.2 *For $u_j \leq u_i - 1$, Constraints* (3.75) *and* (3.76) *provide a strictly tighter relaxation than Constraint* (3.77)*.*

Proof It is sufficient to show that vertices B, C and D of the feasible region defined by Constraints (3.75) and (3.76) satisfy Constraint (3.77), with at least one of them strictly

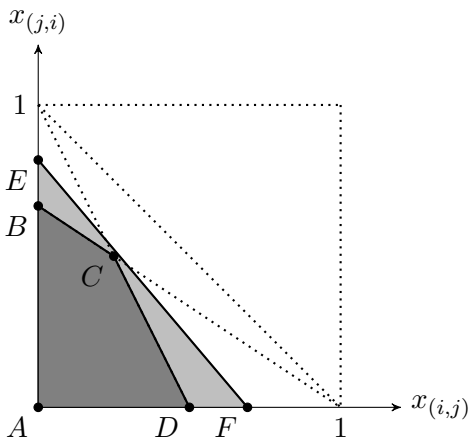


Figure 3.4: The LP relaxation for K_{RLT} is tighter than the LP relaxation for $\overline{K}_{MTZ}$.

satisfying Constraint (3.77). For notational convenience, we let $u_i = u$, and $u_j = u - t$, where $t \geq 1$. In Table 3.1, we present the coordinates of the points in Figure 3.4, and the values obtained when they are plugged in the left-hand-side (LHS) of Constraint (3.77). It is clear that point A strictly satisfies Constraint (3.77), and points E and F are on the constraint line of Constraint (3.77). It is also easy to observe that the expressions in square brackets in LHS of Constraint (3.77) column are strictly greater than 0 when $t > 1$ for points B, C and D, thus strictly satisfying Constraint (3.77). For $t = 1$, points B and C satisfy Constraint (3.77) at equality, however point D strictly satisfies Constraint (3.77), completing the proof. ∎

3.4 Applications to Networked Data Classification

As I mentioned at Chapter 1, extracted k cardinality tree T^* can be represented as a critical component of the complex network which represent the "core" connectivity with restricted number of connection.

In the network data classification, which is to determine the classifying rule which induce the group level difference, KCT can be a representative feature for each network instance. In specific, given $\{G_i, l_i\}_{i=1}^n$ where n is the total number of instances and each G_i has its corresponding label $l_i \in L$, I set simple but insightful hypothesis that the group leveled

Table 3.1: Coordinates of points given in Figure 3.4 and the values obtained when they are plugged in the left-hand-side (LHS) of constraint (3.77)

point	$x_{(i,j)}$-coord	$x_{(j,i)}$-coord	LHS of (3.77)
A	0	0	0
B	0	$\frac{k+2-u}{k+1-u+t}$	$1-\left[\frac{(t-1)(u-t)}{(k+2-t)(k+1-u+t)}\right]$
C	$\frac{(t-1)(u-t-1)}{(k+2)(t+1)-2u}$	$\frac{(t+1)(k+2-u)}{(k+2)(t+1)-2u}$	$1-\left[\frac{(t-1)(t+1)}{((k+2)(t+1)-2u)(k+2-t)}\right]$
D	$\frac{u-t-1}{u}$	0	$1-\left[\frac{(t+1)(k+3-u)}{(k+2-t)u}\right]$
E	0	$\frac{k+2+u_j-u_i}{k+1}$	1
F	$\frac{k+2+u_j-u_i}{k+3}$	0	1

difference, i.e., L, might be described by the alteration of connectivity patterns among G_i. This hypothesis has been widely used in the functional brain network analysis such that the normal structure of functional brain network might be somehow disrupted in neuropsychiatric disease. Therefore, the alteration/disruption can potentially be used as a bio-marker of abnormal brain eventually. As an example, a minimum spanning tree (MST) was introduced to demonstrate disrupted modularity of connectivity of default mode network (DMN) in the functional brain network, which is identified networks in the resting state of human brain [197].

However, even though connectivity among the nodes in the network may be sensitive to abnormal changes, MST might not provide the most insightful information on the network changes because it had to contain all nodes in the network in which some nodes might be irrelevant. In specific, the study of DMN, the regions of DMN are mostly defined based on entire study subject group and the actual regions for individual subject can vary in size, shape and location. Therefore, instead of considering all nodes, I apply KCT as a representative connectivity of the core area in the large scale complex network.

As a framework for this purpose, I can extract a T_i from G_i with predetermined cardinality k and use the average weights $w(T_i)$ in the T_i as a measurable characteristic of connectivity patterns in G_i. Then, I evaluate the discrimination power between group based on

$\{w(T_i), l_i\}_{i=1}^{n}$. Thus, KCT can be regarded as a tool for *feature extraction* of univariate classification in the networked data classification.

3.5 Computational Results

All our formulations were coded in $C\#$ using IBM ILOG CPLEX 12.5 MIP solver and tested on an Intel Xeon E5335 2.00 GHz with 4GB of RAM. For every test instance, computation time was restricted as 1 hour (3,600 seconds). Only single thread processing was used. All parameters used were from the default settings of CPLEX.

In this section, I conduct the test as following two directions. First, I focus on the evaluation of computational efficiency. Through the extensive experiments. I validate the computation performance among the proposed MIPs first. Then, I compare the selected MIPs with the state-of-art methods which have been developed. Second, I apply the KCT for the networked data classification. Specifically, I employ the KCTP as a network analysis technique using functional connectivity human brain networks collected from fMRI. Through the experiments, I aim to characterize transition stages between healthy brain stage to cognitive decline.

3.5.1 Results on Small Public Benchmark Data Set

First, we present the computational results of our models on benchmark problem instance. The first set of KCTP test instances was provided in [67], consisting of 120 connected graphs with 10, 20, and 30 vertices and varying numbers of edges (with a maximum of 268). Subsequently, 35 KCTP instances of 4-regular graphs were provided in [24] with varying numbers of vertices. However, all these graph instances were extremely easy to solve with recent methods and therefore not effective in differentiating true performances of different algorithms [28, 52]. More recently, KCTP test instances of four different graph types were provided in [28]. The first group of graphs included grid type graphs with different number of vertices, which were identified with the suffix *bb*. The second group contained hard 4-regular graphs from [24]. The third group included Steiner tree problem (STP) instances with varying edge densities. Finally, the fourth group was associated with Leighton graphs. Detailed descriptions of these data sets can be found in [28]. In this study, to compare the

computational efficiency of all seven formulations of our KCTP (K_{MTZ}, $\hat{K}_{MTZ}$, $\overline{K}_{MTZ}$, K_{SCF}, $\hat{K}_{SCF}$, $\overline{K}_{SCF}$ and K_{RLT}), we applied our formulations on the benchmark datasets by [28] using 10 equidistantly distributed cardinality values between 2 and $|V| - 2$.

Table 3.2 compares the CPU times for our formulations on the small benchmark instances, where k represents cardinality and $r = \frac{2|E|}{|V|(|V|-1)}$ edge density. Additionally, the second column, named 'Type', represents the type of graph, which are specified by [28]. In general, the results show that K_{SCF} and $\hat{K}_{SCF}$ are faster than other models. Among all four types of test instances, grid instances were reported to be the most difficult instances [52, 29, 39]. $\hat{K}_{SCF}$ outperforms all other models. K_{SCF} comes second with similar performance to $\hat{K}_{SCF}$. However, $\overline{K}_{SCF}$ shows poor performance regardless of graph type.

Table 3.2: CPU time comparisons with the models: selected instance of the benchmark data set. The bold faced measures with $*$ indicate the best performing model for that instance

Instances	Type	k	r	CPU time (in seconds)						
				K_{MTZ}	$\hat{K}_{MTZ}$	$\overline{K}_{MTZ}$	K_{SCF}	$\hat{K}_{SCF}$	$\overline{K}_{SCF}$	K_{RLT}
steine5	sparse	500	0.0010	540.14	80.10	54.94	54.93	**32.67***	3600.00	314.41
steind5	sparse	400	0.0025	371.95	11.39	15.97	18.87	**5.34***	3600.00	22.82
g1000-4-01	regular	20	0.0040	10.48	10.63	9.92	**2.09***	5.55	3600.00	22.53
g1000-4-05	regular	100	0.0040	27.80	12.05	16.24	**3.50***	7.93	3600.00	15.45
g400_4_05	regular	40	0.0100	3.21	3.88	4.21	**1.21***	1.50	3600.00	4.40
bb45×5_1	grid	40	0.0159	1128.79	190.14	321.46	201.56	**13.00***	3600.00	2257.80
bb15×15_2	grid	20	0.0167	39.16	3.07	5.54	6.73	**1.97***	654.52	20.86
steind15	dense	400	0.0100	3600.00	50.03	59.65	102.35	**14.77***	3600.00	335.516
steinc15	dense	400	0.0200	3600.00	63.24	282.49	3.83	**3.29***	3600.00	162.85
lec450_15a	dense	405	0.0809	3600.00	**14.68***	45.33	70.99	49.16	1321.54	45.33

Table 3.3 shows the optimality status results for all of the 190 instances in the benchmark data set. In this table, '*opt*' denotes the number of instances for which the model proved optimality; '*tlo*' denotes that time limit was reached but the current solution was optimal; '*tlf*' denotes that time limit was reached while the current solution was not optimal but feasible; finally, '*na*' denotes that no feasible solution was found within the time limit. A

Table 3.3: Solution status counts for all benchmark instances

	$\hat{K}_{MTZ}$	K_{MTZ}	$\hat{K}_{SCF}$	K_{SCF}	$\overline{K}_{MTZ}$	$\overline{K}_{SCF}$	K_{RLT}
opt	152	103	175	162	142	42	134
tlo	15	66	4	14	17	21	24
tlf	13	12	10	10	22	15	24
na	10	9	1	4	9	112	8

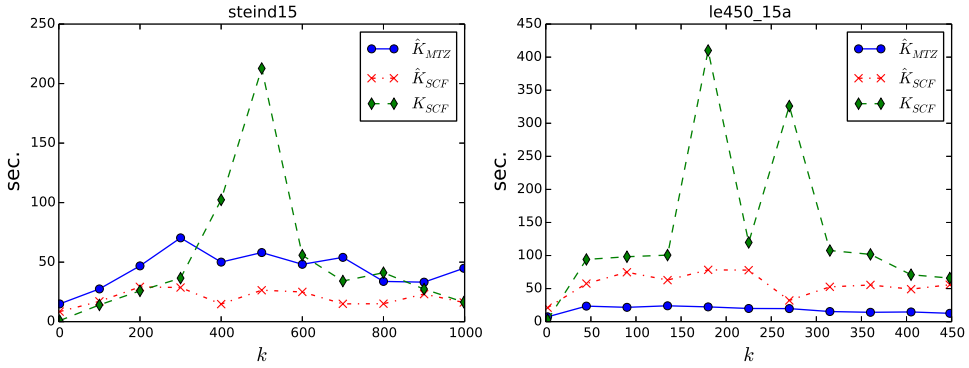


Figure 3.5: CPU time (in seconds) of $\hat{K}_{MTZ}$, $\hat{K}_{SCF}$, K_{SCF} for dense graphs 'steind15' and 'le450_15a'.

great majority of the 190 instances were solved to optimality by all models except for $\overline{K}_{SCF}$ and more than 150 instances out of 190 were solved to provable optimality by our three models: $\hat{K}_{MTZ}$, K_{SCF}, and $\hat{K}_{SCF}$.

In Figure 3.5, we present a plot of CPU times (y-axis) versus an increasing cardinality (x-axis) for the three best performing models based on Tables 3.2 and 3.3, namely, $\hat{K}_{MTZ}$, $\hat{K}_{SCF}$, K_{SCF}. The results in the figure are based on two instances, 'steind15' (left) and 'le450_15a' (right), which are identified as dense graphs in [28]. In 'steind15', $\hat{K}_{SCF}$ shows the best overall performance and K_{SCF} is faster than $\hat{K}_{MTZ}$ except for two specific cardinality values. In 'le450_15a', $\hat{K}_{MTZ}$ is the fastest, followed by $\hat{K}_{SCF}$ and K_{SCF}.

Aside from the CPU time, we also present LP relaxation results of our models in Table 3.4. The reported optimality gap was calculated by $\frac{Z_{IP}-Z_{LP}}{Z_{IP}}$, where Z_{IP} was the best integral solution and Z_{LP} was the LP relaxation value. Table 3.4 shows that $\hat{K}_{SCF}$ has

the smallest optimality gap. It means $\hat{K}_{SCF}$ has a tighter LP relaxation than the other models. In general, there is not much difference between models based on X_{CC} and $\hat{X}_{CC}$. However, models based on $\overline{X}_{CC}$ show relatively worse relaxation results, which implies that constraints based on X_{CC} and $\hat{X}_{CC}$ provide tighter bounds than $\overline{X}_{CC}$. We also observe similar results between $\overline{K}_{MTZ}$ and K_{RLT}, supporting Proposition 3.3.1.

Table 3.4: Optimality gap and LP relaxations in the benchmark data set. 'N/A' indicates that the instance cannot be found any integer feasible solution within the time limit.

Instances	K_{MTZ}		$\hat{K}_{MTZ}$		$\overline{K}_{MTZ}$		K_{SCF}		$\hat{K}_{SCF}$		$\overline{K}_{SCF}$		K_{RLT}	
	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}
steine5	0.0047	9226.78	0.0015	9256.11	0.5202	4460.07	0.0017	9254.17	0.0002	9268.17	N/A	4448.00	0.5202	4448.00
steind5	0.0060	8633.00	0.0007	8679.00	0.3062	6026.00	0.0021	8666.80	0.0001	8684.23	N/A	6030.13	0.3062	6026.00
g1000-4-01	0.0304	255.00	0.0279	255.67	0.7643	62.00	0.0116	259.96	0.0000	263.00	0.7993	62.00	0.7643	62.00
g1000-4-05	0.0086	1633.82	0.0010	1646.38	0.5158	798.00	0.0026	1643.77	0.0010	1646.38	0.5158	798.00	0.5158	798.00
g400_4_05	0.0239	654.00	0.0100	663.28	0.5716	287.00	0.0104	663.06	0.0073	665.08	0.5716	287.03	0.5716	287.00
bb45×5_1	0.2412	527.33	0.2412	527.33	0.7410	180.00	0.2119	547.74	0.1787	570.83	0.7345	184.49	0.7410	180.00
bb15×15_2	0.2242	237.22	0.2190	238.18	0.8142	59.00	0.2006	247.62	0.1534	250.57	0.8127	59.00	0.8127	59.00
steind15	0.0058	2431.80	0.0029	2439.00	0.2416	1856.22	0.0026	2439.53	0.0013	2442.78	N/A	1855.00	0.2416	1855.00
steinc15	0.0048	3554.00	0.0017	3565.00	0.0266	3481.75	0.0008	3568.22	0.0003	3569.82	N/A	3476.00	0.0266	3476.01
lec450_15a	0.0086	624.00	0.0014	628.00	0.0137	592.00	0.0017	627.59	0.0013	628.25	0.0093	593.12	0.0108	592.52

We present comparative results between our top three models, the branch-and-cut algorithm by [174] and the branch-and-bound algorithm by [157]. All of them were implemented under the same computational configuration. In Table 3.5, solution status results are given (similar to Table 3.3), where the branch-and-cut algorithm in [174] is denoted as S_{BnC} and the two branch-and-bound algorithms in [157] are identified as Q_{MCF} and Q_{MCFR} respectively.

S_{BnC} shows the best results in all categories, followed by our models with $\hat{K}_{SCF}$ having the best result among our models. The performance of Q_{MCF} and Q_{MCFR} proposed by [157] are relatively poor.

Table 3.5: Solution status counts for all instances for comparison with other exact solution algorithms.[157, 174]

	$\hat{K}_{MTZ}$	K_{SCF}	$\hat{K}_{SCF}$	S_{BnC}	Q_{MCF}	Q_{MCFR}
opt	152	162	175	190	23	98
tlo	15	14	4	0	3	64
tlf	13	10	10	0	34	10
na	10	4	1	0	130	18

3.5.2 Results on Dense Graphs

The publicly available benchmark datasets lack instances with high edge density. Therefore, we randomly generated 40 different random graphs with various vertice size and edge density. Edge costs were drawn uniformly from $[1, 100]$. Graph sizes and densities were set as the combinations of $|V| \in \{50, 100, 150, 200, 500\}$ and $r \in \{0.25, 0.50, 0.75, 0.99\}$, respectively. Cardinalities were determined in the interval $k \in \{\frac{|V|}{5}, \frac{|V|}{2}\}$. Tables 3.6 shows CPU time of the generated instances. From the table, $\hat{K}_{MTZ}$ solves 6 instances out of 8 faster than others whereas K_{MTZ} solves 2 instances faster than others. K_{SCF} and $\overline{K}_{SCF}$ fail to solve one and K_{RLT} fails to solve two instances to optimality. A detailed account of solution status is given in Table 3.8.

Similar to Table 3.4, we present optimality gap results for selected instances in Table 3.9 and 3.10. From the table, one can observe the same order of tightness of LP relaxation as in Table 3.4. In the same manner as in Section 3.5.1, we choose three models; $\hat{K}_{MTZ}$, K_{MTZ}, $\overline{K}_{MTZ}$, which have shown significant computational performance out of our seven models. Interestingly, we observe that models with MTZ constraints were selected in contrast to the results from the benchmark data set.

Table 3.11 and 3.12 shows the CPU time comparison between our selected models and other algorithms of the selected instances in dense graphs. Also, Table 3.13 presents optimality status counts (similar to Table 3.5), where all algorithms reach optimality in all instances except for Q_{MCF}. Figure 3.6 plots average CPU times of $\hat{K}_{MTZ}$ and S_{BnC} over test instances of 200 vertices ($|V| = 200$) with a cardinality of 100 ($k = 100$) and 500 vertices ($|V| = 500$)

Table 3.6: CPU time comparisons of models on selected dense graphs. The bold faced measures with $*$ indicate the best performing model for that instance.

$\|V\|$	r	k	CPU Time (in seconds)						
			K_{MTZ}	$\hat{K}_{MTZ}$	$\overline{K}_{MTZ}$	K_{SCF}	$\hat{K}_{SCF}$	$\overline{K}_{SCF}$	K_{RLT}
100	0.25	20	0.37	**0.31***	1.17	1.10	0.68	24.23	1.67
100	0.25	50	0.79	**0.60***	0.81	0.77	0.92	12.18	1.26
150	0.5	30	4.26	**3.15***	10.43	6.71	7.92	379.08	17.53
150	0.5	75	**2.02***	4.20	8.89	10.84	20.60	283.30	17.80
200	0.75	40	5.90	**5.11***	10.83	19.01	29.59	635.91	44.54
200	0.75	100	**7.13***	7.27	12.73	21.37	32.17	931.72	56.22
500	0.99	100	290.47	**114.85***	417.90	3600.00	1699.93	3600.00	3600.00
500	0.99	250	565.42	**100.45***	604.50	1076.61	1876.15	3600.00	3600.00

with a cardinality of 250 ($k = 250$).

In the figure, the performance of $\hat{K}_{MTZ}$ was comparable to that of S_{BnC} in smaller graph instances ($|V| = 200$, $k = 100$). However, it is very important to note that the CPU times of $\hat{K}_{MTZ}$ were much more stable than those of S_{BnC}, suggesting our $\hat{K}_{MTZ}$ to be a more robust choice of solution approach. More importantly in larger graph instances ($|V| = 500$, $k = 250$), $\hat{K}_{MTZ}$ notably outperformsS_{BnC}, using only half of CPU times.

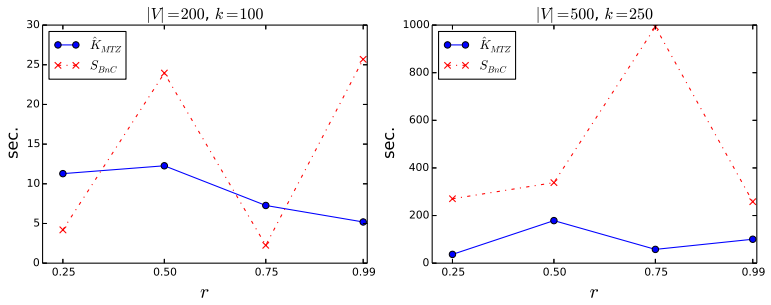


Figure 3.6: CPU time of $\hat{K}_{MTZ}$, S_{BnC} for $|V| = 200$, $k = 100$ and $|V| = 500$, $k = 250$.

Table 3.7: CPU time comparisons of models on selected dense graphs (ER). The bold faced measures with $*$ indicate the best performing model for that instance.

$\|V\|$	p	k	CPU Time (in seconds)						
			K_{MTZ}	$\hat{K}_{MTZ}$	$\overline{K}_{MTZ}$	K_{SCF}	$\hat{K}_{SCF}$	$\overline{K}_{SCF}$	K_{RLT}
100	0.25	20	0.82	**0.66***	2.28	2.02	1.24	15.01	3.27
100	0.25	50	1.10	**0.75***	1.97	1.81	1.50	21.75	3.44
150	0.5	30	5.04	**3.82***	9.78	6.21	8.10	348.12	25.33
150	0.5	75	6.09	**5.25***	12.35	13.01	12.60	339.53	31.06
200	0.75	40	**7.18***	7.90	18.64	18.17	15.69	601.47	57.23
200	0.75	100	11.16	**10.07***	22.10	19.09	23.65	897.20	89.13
500	0.99	100	311.57	**134.44***	390.08	3154.08	2122.15	3600.00	3398.05
500	0.99	250	450.63	**178.56***	789.64	3600.00	3029.34	3600.00	3600.00

Table 3.8: Solution status counts for all dense graph instances. See Section 3.5.1 for the descriptions of '*opt*', '*tlo*', '*tlf*', and '*na*'.

	$\hat{K}_{MTZ}$	K_{MTZ}	$\hat{K}_{SCF}$	K_{SCF}	$\overline{K}_{MTZ}$	$\overline{K}_{SCF}$	K_{RLT}
opt	80	80	78	65	80	70	75
tlo	0	0	0	0	0	3	1
tlf	0	0	2	0	0	5	1
na	0	0	0	5	0	2	3

3.5.3 Results on Large graphs

We compare the computational performance of $\hat{K}_{MTZ}$ and S_{BnC} on large graph instances with $|V| \in \{500, 600, 800\}$. The number of edges in the generated graphs was controlled by the edge densities $r \in \{0.25, 0.50, 0.75, 0.99\}$. For each setting ($|V|$, r), we generated 10 random instances (realizations) with different edge costs, resulting in a total of 120 graphs generated. For notational convenience, we introduce the term "relative cardinality, k_{rel}, which is a fraction of $|V|$, and varied as $\{0.1, 0.25, 0.40, 0.50\}$. For example, suppose that a graph G is generated by ($|V| = 500$, $r = 0.25$) and k_{rel} is set to 0.1. Then, G is constructed by 31,187 edges ($|E| = \lfloor \frac{r \times |V| \times (|V|-1)}{2} \rfloor$) with 500 vertices and the cardinality k for KCTP

Table 3.9: Optimality gap and LP relaxations in the Dense graphs. 'N/A' indicates that the instance cannot find any integer feasible solution within the time limit.

$\|V\|$	r	k	K_{MTZ}		$\hat{K}_{MTZ}$		$\overline{K}_{MTZ}$		K_{SCF}		$\hat{K}_{SCF}$		$\overline{K}_{SCF}$		K_{RLT}	
			GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}
100	0.25	20	0.0200	49.00	0.0200	49.00	0.4200	29.00	0.0200	49.00	0.0200	49.00	0.4200	29.00	0.4200	29.00
100	0.25	50	0.0127	155.00	0.0000	157.00	0.0573	148.00	0.0007	156.89	0.0000	157.00	0.0573	148.00	0.0573	148.00
150	0.5	30	0.0299	33.55	0.0257	34.10	0.1429	30.00	0.0326	33.86	0.0126	34.56	0.1429	30.00	0.1429	30.00
150	0.5	75	0.0102	97.00	0.0053	97.48	0.0816	90.00	0.0054	97.47	0.0051	97.50	0.0816	90.00	0.0816	90.00
200	0.75	40	0.0000	60.00	0.0000	60.00	0.0000	60.00	0.0000	60.00	0.0000	60.00	0.0000	60.00	0.0000	60.00
200	0.75	100	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00
500	0.99	100	0.0000	150.00	0.0000	150.00	0.0000	150.00	N/A	150.00	0.0000	150.00	0.0000	150.00	N/A	150.00
500	0.99	250	0.0000	250.00	0.0000	250.00	0.0000	250.00	0.0000	250.00	0.0000	250.00	0.0000	250.00	N/A	250.00

is 50 ($=|V| \times k_{rel}$). Note that we increase the time limit up to 2 hours (7,200 seconds) due to the large scale of the generated graph instances.

Table 3.14 presents the number of instances in terms of the solution status, which is similar to those in Tables 3.3, 3.5, 3.8, and 3.13. From the table, we observe that the number of optimal instances ('*opt*' in Table 3.14) decreases when the size of graph instances increases ($|V|$ and r). Furthermore, in the case of $r = 0.25$, both models successfully found the optimal solutions except for only one case (S_{BnC} at $|V| = 800$). However, the difference of '*opt*' between the two models gradually increases with an increasing value of r (i.e., the graph becomes denser). Especially, for the case of $|V| = 800$ and $r = 0.99$ across all k_{rel} (the last four columns in Table 3.14), out of the 40 test instances $\hat{K}_{MTZ}$ found feasible solutions of 35 instances ($\sum opt = 26$, $\sum tlf = 9$), whereas S_{BnC} found feasible solutions of only 18 instances ($\sum opt = 6$, $\sum tlf = 12$).

To provide a more in-depth comparison between $\hat{K}_{MTZ}$ and S_{BnC}, Figure 3.7 reports the instance counts of the solution status and relative solution comparison when both models share the same status. As the solution status for each model can be categorized into three levels (refer to Table 3.14), we can build scenarios in terms of their status as follows. First, we define a set of generated graphs I ($|I| = 10$) for each setting of ($|V|$, r), which is composed of three disjoint subsets in terms of the solution status. For $\hat{K}_{MTZ}$, we define

Table 3.10: Optimality gap and LP relaxations in the ER Dense graphs. 'N/A' indicates that the instance cannot find any integer feasible solution within the time limit.

$\|V\|$	r	k	K_{MTZ}		$\hat{K}_{MTZ}$		$\overline{K}_{MTZ}$		K_{SCF}		$\hat{K}_{SCF}$		$\overline{K}_{SCF}$		K_{RLT}	
			GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}	GAP	Z_{LP}
100	0.25	20	0.0714	39.00	0.0190	41.20	0.3333	28.00	0.0388	40.37	0.0190	41.20	0.3333	28.00	0.3333	28.00
100	0.25	50	0.0370	130.00	0.0148	133.00	0.1111	120.00	0.0203	132.26	0.0148	133.00	0.1111	120.00	0.1064	120.64
150	0.5	30	0.0211	36.22	0.0181	36.33	0.1892	30.00	0.0205	36.24	0.0181	36.33	0.1892	30.00	0.1892	30.00
150	0.5	75	0.0183	107.00	0.0092	108.00	0.0642	102.00	0.0017	108.82	0.0000	109.00	0.0642	102.00	0.0642	102.00
200	0.75	40	0.0000	40.00	0.0000	40.00	0.0000	40.00	0.0000	40.00	0.0000	40.00	0.0000	40.00	0.0000	40.00
200	0.75	100	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00
500	0.99	100	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00	0.0000	100.00
500	0.99	250	0.0000	250.00	0.0000	250.00	0.0000	250.00	N/A	250.00	0.0000	250.00	0.0000	250.00	N/A	250.00

a set $I^K = I^K_{opt} \cup I^K_{tlf} \cup I^K_{na}$. Similarly, we define a set $I^S = I^S_{opt} \cup I^S_{tlf} \cup I^S_{na}$ for S_{BnC}. Recall that status *opt*, *tlf*, and *na* are representations of optimality, time limit feasibility, and non-availability of a solution (i.e., *out-of-memory* error and no solution within time limit), respectively. For comparison measurements, we consider two factors: t_i is the CPU time for instance i and v_i is a solution value for instance i. We use the term (t^K_i, v^K_i) for $\hat{K}_{MTZ}$ and (t^S_i, v^S_i) for S_{BnC}, respectively. Based on the given notations, we build 6 scenarios to count comparison results between the two models.

- Scenario 1 (S_1): In this scenario, we count the number of instances, at which $\hat{K}_{MTZ}$ **outperforms** S_{BnC}. If both models solve instance i within the time limit (i.e., finding an optimal solution), we simply compare t^K_i and t^S_i. Otherwise, if $\hat{K}_{MTZ}$ finds an optimal solution whereas S_{BnC} does not (i.e., time limit feasible or failure to find the solution), we regard that $\hat{K}_{MTZ}$ outperforms S_{BnC}. Thus, we are able to count up the number of instances for scenario 1: $|S_1|$, where $S_1 = \{i \mid t^K_i < t^S_i,\ \forall i \in I^K_{opt} \cap I^S_{opt}\} \cup \{i \mid \forall i \in I^K_{opt} \cap (I^S_{tlf} \cup I^S_{na})\}$.

- Scenario 2 (S_2): In this scenario, we also count the number of instances, at which $\hat{K}_{MTZ}$ **outperforms** S_{BnC}. When both models find a feasible solution for instance i but are terminated by the time limit, we compare v^K_i and v^S_i. We count the number of

Table 3.11: CPU time comparison with other exact solution algorithms [157, 174]. The bold faced measures with $*$ indicate the best performing model for that instance.

$\|V\|$	r	k	CPU time (in seconds)					
			$\hat{K}_{MTZ}$	K_{MTZ}	$\overline{K}_{MTZ}$	S_{BnC}	Q_{MCF}	Q_{MCFR}
100	0.25	20	**0.35***	0.37	1.17	0.45	1203.49	1.69
100	0.25	50	0.60	**0.49***	0.81	0.70	1475.38	2.96
150	0.5	30	**3.15***	4.26	10.43	5.34	3600.00	13.20
150	0.5	75	4.20	**2.02***	8.89	4.52	3600.00	5.59
200	0.75	40	**5.11***	6.90	10.83	7.71	3600.00	31.82
200	0.75	100	**6.17***	7.13	12.73	6.25	3600.00	278.35
500	0.99	100	**114.85***	290.47	417.90	130.18	3600.00	858.46
500	0.99	250	**100.45***	565.42	604.50	258.14	3600.00	1066.27

instances where $v_i^K < v_i^S$. Thus, the number of instance for scenario 2 can be obtained by: $|S_2|$, where $S_2 = \{i \mid v_i^K < v_i^S,\ \forall i \in I_{tlf}^K \cap I_{tlf}^S\} \cup \{i \mid \forall i \in I_{tlf}^K \cap I_{na}^S)\}$.

- Scenario 3 (S_3): In contrast to scenario 1, this scenario is when S_{BnC} **outperforms** $\hat{K}_{MTZ}$. The number of instances for this scenario is equal to $|S_3|$, where $S_3 = \{i \mid t_i^S < t_i^k,\ \forall i \in I_{opt}^S \cap I_{opt}^K\} \cup \{i \mid \forall i \in I_{opt}^S \cap (I_{tlf}^K \cup I_{na}^K)\}$.

- Scenario 4 (S_4): In contrast to scenario 2, this scenario is also when S_{BnC} **outperforms** $\hat{K}_{MTZ}$. In this scenario, we count the number of instances, where S_{BnC} finds a better solution than that of $\hat{K}_{MTZ}$, and denote it by $|S_4|$, where $S_4 = \{i \mid v_i^S < v_i^K,\ \forall i \in I_{tlf}^S \cap I_{tlf}^K\} \cup \{i \mid \forall i \in I_{tlf}^S \cap I_{na}^K)\}$.

- Scenario 5 and 6 (S_5, S_6): These two scenarios are when both models have a tie in performance. In scenario 5, we count when both models find the same feasible solution. We denote the set of instances for scenario 5 as $S_5 = \{\ i \mid v_i^K = v_i^S,\ \forall i \in (I_{tlf}^K \cap I_{tlf}^S)\}$. In scenario 6, we consider the cases, where both models fail to find a solution. Hence, the set for scenario 6 is defined as: $S_6 = \{i \mid \forall i \in I_{na}^K \cap I_{na}^S\}$.

Table 3.12: CPU time comparison with other exact solution algorithms [157, 174] of ER data set. The bold faced measures with $*$ indicate the best performing model for that instance.

$\|V\|$	p	k	CPU time (in seconds)					
			$\hat{K}_{MTZ}$	K_{MTZ}	$\overline{K}_{MTZ}$	S_{BnC}	Q_{MCF}	Q_{MCFR}
100	0.25	20	0.66	0.82	2.28	0.75	1104.59	2.01
100	0.25	50	0.75	1.10	1.97	1.03	1565.20	5.12
150	0.5	30	3.82	5.04	9.78	4.12	3600.00	14.50
150	0.5	75	5.25	6.09	12.35	6.05	3600.00	7.79
200	0.75	60	7.90	7.18	18.64	8.89	3600.00	25.28
200	0.75	100	10.07	11.16	22.10	10.89	3600.00	103.64
500	0.99	150	134.44	311.57	390.08	180.08	3600.00	771.35
500	0.99	250	178.56	450.63	789.64	223.84	3600.00	1521.62

Table 3.13: Solution status count comparison with other exact solution algorithms. [157, 174]

	$\hat{K}_{MTZ}$	K_{MTZ}	$\overline{K}_{MTZ}$	S_{BnC}	Q_{MCF}	Q_{MCFR}
opt	80	80	80	80	25	75
tlo	0	0	0	0	0	0
tlf	0	0	0	0	18	5
na	0	0	0	0	37	0

In Figure 3.7, we compare the performance between two models according to $|V|$. For $V = 500$, S_{BnC} outperforms $\hat{K}_{MTZ}$ for most cases. For $V = 600$, results are mixed and it is not clear which model is superior. For $V = 800$, we can clearly observe that $\hat{K}_{MTZ}$ outperforms S_{BnC} at $r = 0.75$ and $r = 0.99$ except two cases; ($r = 0.25$, $k_{rel} = 0.10$) and ($r = 0.50$, $k_{rel} = 0.25$).

Figure 3.8 depicts bar graphs of average CPU times of 10 iterations for each ($|V|$, r) separately, where the error bars indicate the standard error. Each sub-figure presents a result for a different relative cardinality value. The first number in the x-axis labels represents

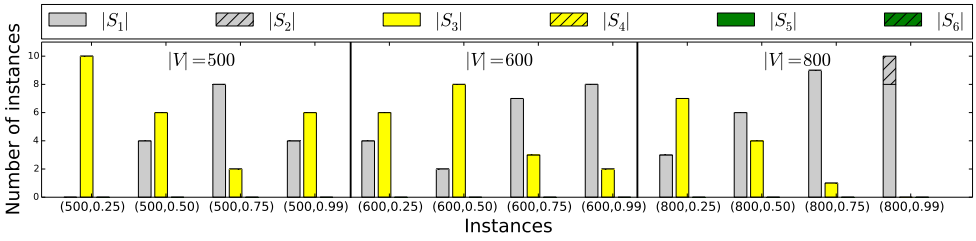
Table 3.14: Solution status count for large graphs. '*opt*' is the number of instances which is optimal. '*tlf*' is the number of instances which is terminated by the time limit while providing a feasible solution. '*na*' is the number of instances which is not able to find any solution. Also, '*na*' includes *out-of-memory* error cases. The three numbers in parentheses refer to '*opt*', '*tlf*', and '*na*' in sequential order.

Condition	k_{rel}	Instances												
		$\|V\|$	500				600				800			
		r	0.25	0.50	0.75	0.99	0.25	0.50	0.75	0.99	0.25	0.50	0.75	0.99
opt/tlf/na	0.10	$\hat{K}_{MTZ}$	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(8,2,0)
		S_{BnC}	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(9,0,1)	(8,0,2)	(7,0,3)	(10,0,0)	(5,4,1)	(7,1,2)	(2,4,4)
	0.25	$\hat{K}_{MTZ}$	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(9,1,0)	(10,0,0)	(10,0,0)	(9,1,0)	(9,1,0)
		S_{BnC}	(10,0,0)	(9,0,1)	(10,0,0)	(8,1,1)	(10,0,0)	(8,1,1)	(10,0,0)	(6,4,0)	(10,0,0)	(8,1,1)	(2,5,3)	(2,7,1)
	0.40	$\hat{K}_{MTZ}$	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(9,1,0)	(10,0,0)	(10,0,0)	(9,1,0)	(5,3,2)
		S_{BnC}	(10,0,0)	(10,0,0)	(7,2,1)	(9,0,1)	(10,0,0)	(10,0,0)	(6,2,2)	(7,1,2)	(10,0,0)	(2,7,1)	(7,1,2)	(1,0,9)
	0.50	$\hat{K}_{MTZ}$	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(10,0,0)	(8,0,2)	(9,1,0)	(10,0,0)	(10,0,0)	(9,1,0)	(4,3,3)
		S_{BnC}	(10,0,0)	(10,0,0)	(9,0,1)	(9,1,0)	(10,0,0)	(9,1,0)	(6,4,0)	(9,0,1)	(9,1,0)	(7,2,1)	(1,0,9)	(1,1,8)

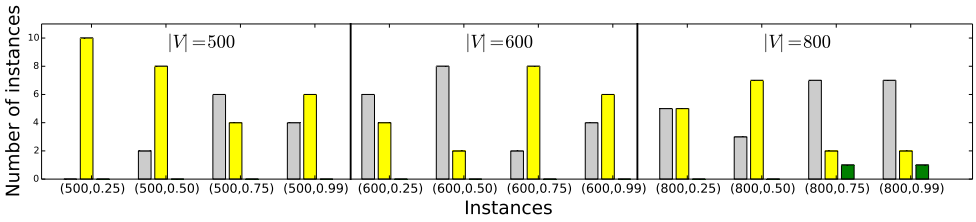
the number of vertices $|V|$ and the second number represents edge density r. We made an assumption that the instances that do not fit in the allocated memory would take longer to solve than those that fit in the memory and cannot be solved optimality within the time limit. Thus, instances that induced *out-of-memory* error were treated to failure of finding solution within the time limit. From these plots, it is clear that $\hat{K}_{MTZ}$ is faster than S_{BnC} in the majority of the instances, especially as the graph size increases. From the error bars, we can conclude that $\hat{K}_{MTZ}$ has a more robust performance when compared to S_{BnC}. For example, at $|V| = 600$, $r = 0.5$ and $k_{rel} = 0.10$, S_{BnC} is faster in 8 out of 10 instances, but average CPU time is longer than $\hat{K}_{MTZ}$.

3.5.4 Detection Cognitive Decline within Default Mode Network via KCTP

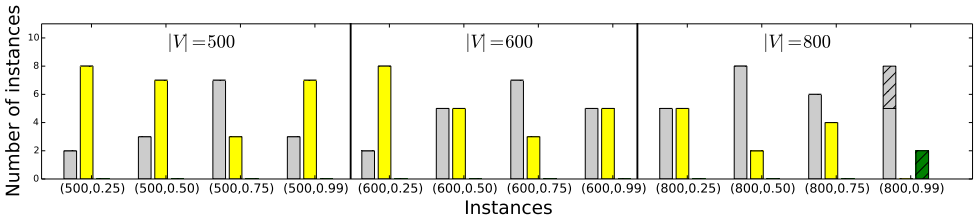
Mild cognitive impairment has been defined as an intermediate stage between normal age-related changes and dementia [149]. To prevent from the delayed detection, there is a need for quantitative method that can be used to assess brain function before overt cognitive decline. In this section, I use KCT as a tool to investigate if the strength and structure of brain connectivity within functional regions are altered and associated with cognitive



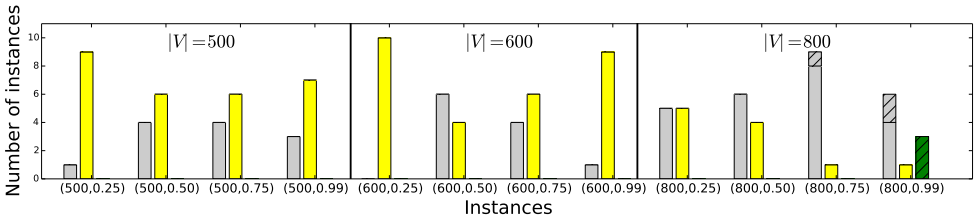
(a) $k_{rel} = 0.10$



(b) $k_{rel} = 0.25$



(c) $k_{rel} = 0.40$

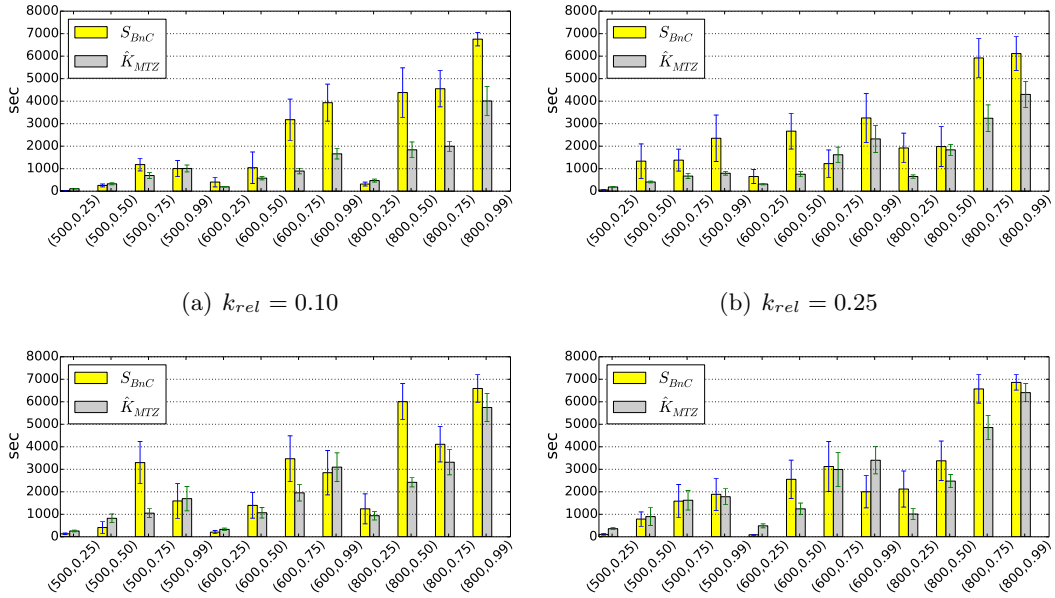


(d) $k_{rel} = 0.50$

Figure 3.7: Comparison of the solution status for $\hat{K}_{MTZ}$ and S_{BnC} over different graph sizes, densities (first and second number in x-axis labels) and relative cardinalities (different plots). Each bar is the number of instances within its respective scenario.

decline.

In this section, I use the functional connectivity networks collected from fMRI which allowed



(a) $k_{rel} = 0.10$ (b) $k_{rel} = 0.25$

(c) $k_{rel} = 0.40$ (d) $k_{rel} = 0.50$

Figure 3.8: Comparison of CPU times for $\hat{K}_{MTZ}$ and S_{BnC} (in seconds) over different graph sizes, densities (first and second number in x-axis labels) and relative cardinalities (different plots).

researchers to define alteration in large scale neuronal networks that may be related to abnormal cognitive changes. In particular, I focus on the default mode network (DMN), which include multiple spatially distinct regions in the resting state of human brain. Many researches have shown that DMN have been shown to associated with neurological disorders [152, 41, 73].

In contrast to conventional studies of the DMN focusing on examining coarse functional connectivity between spatially distinct DMN regions, called *inter connectivity* between DMN regions, I focus on a local level network of functional connectivity within individual DMN regions (*intra connectivity*) rather than limiting oneself to a global level network among DMN regions since the role of disrupted local connectivity is still unexplored.

Data Set

I used fMRI data of 29 subjects collected from the Seattle Longitudinal Study (SLS). Among 29 subjects, 11 subjects were classified a priori as having cognitive declining and remain 18 subjects are stable. For each subject's fMRI data, specific regions of DMN were obtained. Defined six local regions are:

- Region-1: (Bilateral) Medical Frontal Cortex (MFC)
- Region-2: (Bilateral) Posterior Cingulate Cortex (PCC)
- Region-3: Left Dorsal Parietal Cortex (lDPC)
- Region-4: Left Medial Temporal Lobe (lMTL)
- Region-5: Right Dorsal Parietal Cortex (rDPC)
- Region-6: Right Medial Temporal Lobe (rMTL)

After DMN regions were identified, a local connectivity graph was constructed for each DMN region based on the functional connectivity matrix of all voxel pairs with in the region. Thus, it is represented by N by N matrix, where N is the number of voxels in the region. A pairwise voxel connectivity is measured by correlation coefficient for each BOLD signals obtained from the voxel. Since we consider all pairwise of voxels, constructed networks are all *fully connected* within N voxels.

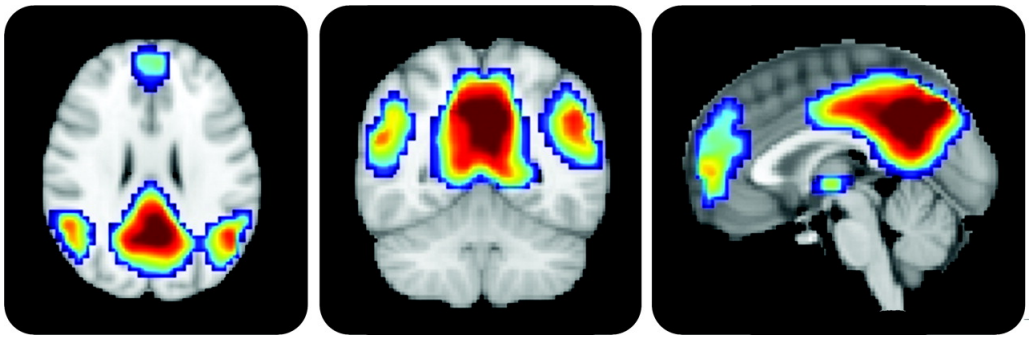


Figure 3.9: Example of default mode network (DMN) [151]

For each of the six DMN regions, Table 3.15 shows the range, average, and standard deviation of the number of voxels (=nodes) in the subject space across all subjects.

Table 3.15: The range and the average and standard deviation of the problem sizes across different subjects.

Statistics	DMN Regions of Interest (ROIs)					
	MFC	PCC	lLPC	lMTL	rLPC	rMTL
Range [Min–Max]	[719–1062]	[538–929]	[222–317]	[93–128]	[310–479]	[110–189]
Average ± std	878 ± 83	716 ± 94	271 ± 25	109 ± 9	401 ± 38	147 ± 18

Discriminating Power

First of all, we employ $\hat{K}_{MTZ}$ as the main model for this approach since it has shown the best performance from the computational experiments at Section 3.5.1, 3.5.2 and 3.5.3. The cardinality K was fixed as a percentage of the number of voxels to be included from the total number of voxels in the region because every subject has a different size of network for each region. I fixed the value of K to 10%, 25%, 50%, 75%. Note that when $K = 100\%$, the KCT is equivalent to the MST.

Basically, to compare the solution to the KCT problem across subjects, a percent average correlation per edge (i.e., dividing the objective function value by the number of edges in KCT times 100) was reported and referred as the *percent normalized objective function (PNOF) value*.

To investigate which regions played a significant role in separating subject group into "decliner" and "non-decliner", for each region I isolated the PNOFs of the two groups. Table 3.16 reports the p-values of $\hat{K}_{MTZ}$ method. Note that two DMN regions, MFC and PCC, are excluded since they can be solved by $\hat{K}_{MTZ}$ within a time limit due to their larger size. The bold faced number in the Table 3.16 represent p-value ¡0.05, and one can observe that the KCT solutions show more discriminating power when $K\%$ is larger, i.e., more voxels are included. The result in Table 3.16 can conclude that left and right MTL regions are the key DMN regions that are significantly altered by cognitive decline and likely to be used as a bio-marker of executive decliner.

Table 3.16: Comparison of p-value for decliner vs. non-decliner instances using $\hat{K}_{MTZ}$

DMN regions	$K\%$				
	10%	25%	50%	75%	100%
lLPC	0.6393	0.5362	0.4148	0.3899	0.3994
lMTL	0.0898	**0.0373**	**0.0251**	**0.0106**	**0.0082**
rLPC	0.9724	0.9160	0.7493	0.6484	0.6180
rMTL	0.3412	**0.0469**	**0.0270**	**0.0217**	**0.0264**

3.6 Summary

In this chapter, we propose a modular framework to formulate the KCTP. We formulate seven MIP models to find connected components and eliminate cycles in these components. For cycle elimination, we use SCF constraints and lifted MTZ constraints. We also introduce a RLT formulation based on the MTZ constraints to obtain tightened relaxation bounds. To illustrate the usefulness of our proposed MIP models, we present computational results on public benchmark datasets and randomly generated graphs. We show that SCF-based formulations perform better on benchmark datasets and exhibit tighter LP relaxation comparing with others. The random graph instances that we generated are designed to be dense, on which MTZ-based models perform better. However, SCF-based models still have tight relaxation bounds on random graph instances.

We also compare our fastest formulations, all of which are polynomial in size, to other polynomial and exponential size exact solution models that are known to be the state-of-the-art methods to solve KCTP. The model with an exponential number of constraint in [174], which is solved by a branch-and-cut algorithm, outperforms others on the benchmark datasets, however the polynomial-sized formulations in [157] do not show significant advantages compared to our formulations. On dense graph instances, the proposed MTZ-based models outperform all other models. We also perform a comparison between the best MTZ-based model and the exponential-sized model proposed in [174] on large random graphs.

Our formulation not only performs better in overall, but provides more robust results with smaller variation of CPU time. Thus, we can conclude that the contribution of this study is to present both theoretical and computational results of our framework as a method to obtain simple-to-implement, low-overhead formulations to solve KCTP with relatively good performance, especially in larger and denser graphs.

In addition, from the results in the human brain networks analysis, KCTP could be used a feature extraction technique for the classification of networked data having various size. As one of the real world applications, identification of local connectivity strengths and configurations could provide a noninvasive bio-marker for brain health, and aid in the assessment of neuroprotective strategies. Computational method via KCTP can be considered as a essential first step to develop useful tools for system neuroimaging that can be employed and tested a new bio-marker of cognitive decline. Moreover, this approach will enable the methodical uncovering of abnormal alterations in brain function and bring informative insight into mechanism of brain disease.

Chapter 4

NODE SELECTION OF NETWORKED DATA CLASSIFICATION VIA MATHEMATICAL PROGRAMMING

4.1 Introduction

Knowledge discovery in complex data is an essential problem in the data science including machine learning, data mining and data analytics. Networked data, a discrete structure, have been widely applied in various areas such as bio-informatics, social network analysis and human brain connectivity networks. One of the analytic tools on the networked data is data classification. It is to develop a pattern to separate the group labeled networks. Networked data classification have a wide variety of applications and decomposed into several specific domain in terms of the structure of the networks, algorithmic approaches, and preliminary knowledge. For example, [115] represented chemical compounds as network structures and classified those structures with respect to toxic vs. non-toxic and active vs. inactive in an anti cancer screening process. As we mentioned at Section 2.4.1, feature extraction on networked data classification can be divided into two categories:

- (1) The first one is based on topological characteristics of given networks. It also branches into two sub-categories: one is using various topological measurements based on graph theory such as clustering coefficient, modularity, centrality and so on. These measurements have been used to capture global pattern in connected networks. It has been widely applied to real world problems e.g., text mining [153], websites [192] and human brain networks for detecting alteration of specific neurological disorders [180]. The other is called as graph mining which investigates patterns of subgraphs in the given networked data. This approach has been applied to text classification, computer vision, software engineering and computational biology [189, 103, 115, 164].
- (2) The second one is usually called as a bag of edges, where each network is transformed to a collection of edges. Such transformation allows us to apply standard mul-

tivariate classification methods such as regression, decision trees and support vector machine. Particularly, this method has been applied for brain networks classification [112].

In (1), regardless of successful achievements for the approaches, it has several drawbacks. First, it is less sensitive to local changes (e.g., only a few edges are altered) [112]. Second, it is not directly applicable when all observations have same network structure e.g., all observations are represented by complete networks (e.g., human brain networks). In this case, in order to obtain heterozygous connectivity structures among the networks, researchers have used a filtering method: to fix a threshold $\varepsilon \in \mathbb{R}$ and remove the links whose weight is lower than ε to zero. Once the filtered network is obtained, various topological metrics are calculated in terms of the aim of research. However, there is no clear evidence that removed edges are not informative. Finally, we have to rely on the preliminary knowledge in terms of the test metrics. Moreover, in the graph mining approaches, it is inappropriate to apply large sized datasets because enumerating all (or even partial) subgraphs is computationally difficult and its discriminate power is not generally stable for various types of the networks [164, 189].

In the large scale complex networks, there might exist the most discriminative subnetwork that contains critical (=valuable) information related to group difference. For instance, among large scale chemical compound networks, there exist a subnetwork which discriminate the toxic and non-toxic of original networks while distinction from noisy (i.e., non-informative) subnetworks.

In the machine learning community, to identify the most informative subset of data is conducted by feature selection (For detail, recall Section 2.3.4). However, it is hard to determine the subnetwork by means of features made by (1) due to the structural information loss after the feature extraction phase. For example, once we obtain a connectivity pattern within subgraphs from the connectivity structure, how can we obtain a particular subnetwork from the connectivity pattern? The subset of connectivity pattern can be a partial information thus it cannot represent the original networks anymore.

In contrast from (1), since a feature is exactly matched to an edge separately, we can infer the informative subgraph through the feature selection easily while preserving original connectivity pattern. Therefore, in this chapter, we are focusing on the determining critical component of the whole networks based on (2).

In detail, if we transplant the feature selection scheme into the networked data classification, it would be equivalent to the edge selection problem since there exists one-to-one matching between an edge and a feature respectively. As a result, we can select a set of edges via conventional feature selection scheme. However, we suggest different perspective for the selection problem in this chapter. Given a certain network structure, it is almost certain that features' relevance to the label might be affected by a set of nodes instead of individual edges. Thus, an extracted sub network by feature selection can be denser when there exist a few informative nodes for classification. Therefore, it is important not only to find relevant/informative edges but also to find relevant/informative nodes. Hence, we propose a new feature selection scheme which is based on the selection of nodes for which are corresponding to the informative edges. To model our proposing method, we present a mixed integer programming (MIP) model for node selection, which aims to minimize the classification error with restriction of total number of nodes in the resulting sub-network instead of penalizing or restricting total number of edges.

In order to handle with computational difficulty for the constructed MIP, a branch-and-cut algorithm based on the benders decomposition is introduced. It decomposes the original problem into two sub-problems: one is to decide sub network with use of integer variables and the other is to calculate minimum classification error based on the determined subnetwork inherited from the first sub-problem. We conduct computational experiments with two kinds of network datasets: synthetic dataset and human brain networks datasets constructed by functional magnetic resonance imaging (fMRI). From the experimental results, we demonstrate that our approaches outperform the state-of-the-art feature selection algorithms for the networked classification problems. Also, the selected nodes can provide intrinsic information to reveal the complex structure of the given networked data sets.

The remainder of this chapter is organized as follows: Section 4.2, we provide related studies for this chapter including benders decomposition and networked data classification problem.

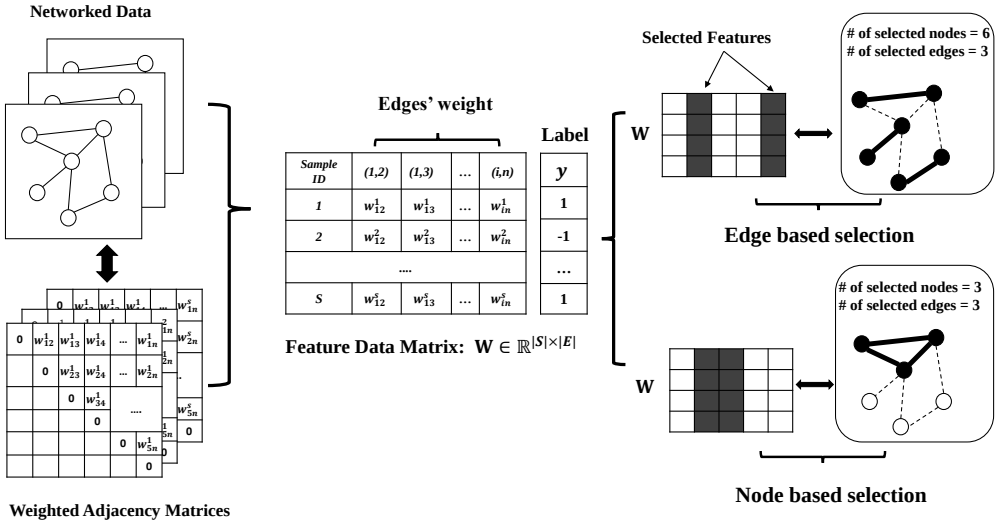


Figure 4.1: An illustration of the networked data classification with node selection

We formulate the problem with mathematical programming in Section 4.3 and propose the benders decomposition approach in Section 4.4. Section 4.5 presents the experimental results. Finally, we conclude with some remarks and directions for future research.

4.2 Related Studies

In this section, we briefly introduce background knowledge involved in this chapter. Firstly, we introduce several networked data classification researches especially on the human brain networks. And then, we provide a concise terminology of the benders decomposition.

4.2.1 Networked data classification for human brain networks

Among the various applications on this problem, the application on human brain networks becomes an emerging are in this field. In the last decade, the use of advanced tools deriving from statistics, signal processing, and statistical physics, has significantly improved our understanding of the brain function. Notably, connectivity-based methods have had a prominent role in characterizing normal brain organization as well as alteration due to

various disorder of brains. By measuring the magnitude of temporal dependence between regional activities - using functional modalities such as functional magnetic resonance imaging (fMRI), electroencephalography (EEG), or magnetoencephalography (MEG) - functional connectivity patterns describe how N different brain regions interact with each other. The resulting $N \times N$ multivariate relationships lead to an interconnected representation of the brain that can be conveniently treated as a network.

Based on the constructed brain networks, the two approaches mentioned at Section 4.1 have been extensively applied to investigate functional changes of the brain under various circumstances. Especially, the first approach, topological investigation has been widely studied to determine the networked characteristic of human brain with neurological disorder. Given all the pairs of regions lead to $N \times N$ relationships that can be represented in a complete weight matrix $W_{N\times N}$, containing all the pairwise functional connectivity measure $w_{(i,j)}$ corresponding to the weighted links between regions i and j. After that, filtering method is applied: to fix a threshold $\varepsilon \in \mathbb{R}$ and remove the links whose weight $w_{(i,j)}$ is lower than ε to zero. Once the filtered graph is obtained, various metric are calculated in terms of the aim of research. Finally, the structural difference between subjects (or groups) might be observed.

For example, "small-worldness" was measured on the human brain networks to denote the topological alteration/disruption in the Alzheimer's disease patients [185] and Schizophrenia [126]. Also [200] showed that clustering coefficient of children with ADHD significantly increased rather than aged volunteers. Detail information for topological studies are well described in the two survey papers [180, 42].

On the contrary from the efforts for topological characteristics investigations, the second approach has been focused on the achievement of accurate prediction under given connectivity data. Most of the researches are implication of collected brain networks with use of the stat-of-the-art classification approaches. Based on the collections of edges weights, researches have pursued to reveal the specific patterns could be distinguished from the labeled group such as patients vs. healthy control , children vs. adult, and task related vs. resting state [155, 48, 139, 57, 216].

4.2.2 Benders Decomposition

Benders decomposition is a solution method for solving large-scale optimization problems. Instead of considering all decision variables and constraints of a large-scale problem simultaneously, the benders decomposition partitions the problem into multiple smaller problems. Since computational difficulty of optimization problems increases significantly with the number of variables and constraints, solving these smaller problems iteratively can be more efficient than solving a single large problem wholly [19].

In Benders decomposition, a first-stage master problem is solved for a subset of variables, and the values of the remaining variables are determined by a second-stage subproblem given the values of the first-stage variables. If the subproblem determines that the proposed first-stage decisions are infeasible, then one or more constraints are generated and added to the master problem, which is then re-solved. In this manner, a series of small problems are solved instead of a single large problem, which can be justified by the increased computational resource requirements associated with solving larger problems.

Mathematically, let consider the following problem:

$$\min\{c^T x + d^T y |\ Ax \geq b,\ Tx + Qy \geq g,\ x \in \mathbb{Z}_+^n,\ y \in \mathbb{R}_+^t\} \tag{4.1}$$

Benders' decomposition works as follows. An artificial variable $\gamma = d^T y$ is introduced along with a lower bound $\overline{\gamma}$ and the master problem relaxation is give by:

$$\min\{c^T x + \gamma |\ Ax \geq b,\ \gamma \geq \overline{\gamma},\ x \in \mathbb{Z}_+^n\} \tag{4.2}$$

Assume that (4.2) can be solved and find the optimal solution (x^*, γ^*). Note that x^* is integer. The optimal solution is used to define the following dual slave problem:

$$\max\{\pi^T (g - Tx^*) |\ \pi^T Q \leq d^T,\ \pi \geq 0\} \tag{4.3}$$

If the dual slave problem (4.3) is unbounded, an unbounded extreme ray $\overline{\pi}$ is selected, and the benders' feasibility cut (*feas-cut*); $\overline{\pi}^T (g - Tx^*) \leq 0$ is added to the master, which is solved again. Otherwise, let $(z^*, \overline{\pi})$ be the optimal value and the optimal vertex of the dual slave problem respectively. If $z^* \leq \gamma^*$ then the feasibility of the current solution (x^*, γ^*) is established and it would be an optimal for the problem (4.1). If not (i.e., $z^* > \gamma^*$), the

optimality cut (*opt-cut*); $\overline{\pi}^T(g - Tx^*) \leq \gamma$ is added to the current master problem. The overall procedure is to repeat those process iteratively.

4.3 Mathematical Model

Let n denote the number of subjects or observations in the sample and let n_1 and n_2 represents the number of subjects in two groups e.g., controls vs. patients respectively. Let S denote the set of all observations and let S_1 and S_2 are subset of all observations corresponding to each group respectively (e.g., S_1=Healthy controls vs. S_2=Patients). We assume that there are no overlapped observations ($S_1 \cap S_2 = \emptyset$), thus we have $S = S_1 \cup S_2$. For each s-th observation has a data profile which is represented as a networked data $G(N, E)$ with node set N and edge set E. For each subject s, every edge (i, j) in E has a weight $w^s_{(i,j)}$ and it is also regraded as a feature expression of the connection (i, j).
Based on the linear programming approaches of classification (see Section 2.3.2), we have:

$$P_c : \qquad \min \quad \textstyle\sum_{s\in S_1} z^s_+ + \sum_{s\in S_2} z^s_- \tag{4.4}$$

subject to:

$$x_0 + \sum_{(i,j)\in E} x_{(i,j)} w^s_{(i,j)} + z^s_- - z^s_+ = -\sigma \qquad \forall s \in S_1, \tag{4.5}$$

$$x_0 + \sum_{(i,j)\in E} x_{(i,j)} w^s_{(i,j)} + z^s_- - z^s_+ = \sigma \qquad \forall s \in S_2, \tag{4.6}$$

$$x_0, \quad x_{(i,j)} \quad \text{free} \qquad \forall (i, j) \in E, \tag{4.7}$$

$$z^s_+, z^s_- \geq 0 \qquad \forall s \in S. \tag{4.8}$$

In (4.5) and (4.6), σ is a constant and any σ can be used basically. In here, we set $\sigma = 1$ in our implementation for convenient usage. Since z^s_-, z^s_+ are non-negative variables, at most one of them has a value. These variables represent how far each subject is separated by the learned classifiers. Hence, if a subject is correctly classified then, $z^s_- - z^s_+ + \sigma > 0$ for $s \in S_1$ and $z^s_- - z^s_+ - \sigma < 0$ for $s \in S_2$. z^s_- and z^s_+ represent empirical errors of the classification eventually. Note that we employ positive σ in order to prevent the trivial solution where $x_0 = 0$ and $x_{(i,j)} = 0$ for all $(i, j) \in E$. Since we use $\sigma > 0$, a gap of classification is equal to 2σ obviously.

To prevent from the "curse of dimensionality", one can apply the feature selection scheme to the P_c by pursuit of the sparse $\mathbf{x}$ where $\mathbf{x} = [x_1\, x_2 \ldots x_{|E|}]$. Based on the Section 2.3.4, we can add the sparsity constraint $\|\mathbf{x}\|_0 \leq k$ to P_c. However, as we mentioned at Section 4.1, we consider different perspective of the feature selection scheme. Although distinctive edges can affect the classification actually, we presume that such distinction on edges are induced by a set of nodes. In other words, there exist a set of nodes corresponding to the subject's class. Thus, we limit the number of nodes instead of the number of edges to determine relevant / informative nodes as well as edges.

Let introduce binary variable $\mathbf{y} = [y_1\ y_2\ \ldots\ y_N]$ indicating whether the node i is selected ($y_i = 1$) or not ($y_i = 0$). Additionally, another binary variable $e_{(i,j)} \in \{0, 1\}$ is introduced, which denotes that edge (i, j) is selected in the classifier, i.e., estimated parameter if $x_{(i,j)} \neq 0$, then $e_{(i,j)} = 1$. Otherwise $e_{(i,j)} = 0$ This modified mixed integer programming (MIP) model P_c^R is stated formally as follows:

$$P_c^R: \quad \text{Minimize} \quad \textstyle\sum_{s\in S_1} z_+^s + \sum_{s\in S_2} z_-^s \tag{4.9}$$

subject to:

$$x_0 + \sum_{(i,j)\in E} x_{(i,j)} w_{(i,j)}^s + z_-^s - z_+^s = -\sigma \qquad \forall s \in S_1, \tag{4.10}$$

$$x_0 + \sum_{(i,j)\in E} x_{(i,j)} w_{(i,j)}^s + z_-^s - z_+^s = \sigma \qquad \forall s \in S_2, \tag{4.11}$$

$$x_{(i,j)} \leq M e_{(i,j)} \qquad \forall (i,j) \in E \tag{4.12}$$

$$-x_{(i,j)} \leq M e_{(i,j)} \qquad \forall (i,j) \in E \tag{4.13}$$

$$\sum_{(i,j)\in adj(j)} e_{(i,j)} \leq D_j y_j \qquad \forall j \in N, \tag{4.14}$$

$$\textstyle\sum_{n\in N} y_n \leq k \tag{4.15}$$

$$x_0, \quad x_{(i,j)} \quad free \qquad \forall (i,j) \in E, \tag{4.16}$$

$$z_+^s, z_-^s \geq 0 \qquad \forall s \in S \tag{4.17}$$

$$e_{(i,j)},\ y_n \in \{0, 1\} \qquad \forall (i,j) \in E, \quad \forall n \in N \tag{4.18}$$

Constraints (4.10)) and (4.11) are equivalent to (4.5)) and (4.6). (4.12) and (4.13) are coupling constraints; if $y_{(i,j)} > 0$ or $y_{(i,j)} < 0$ then $e_{(i,j)} = 1$. $M > 0$ is a big-M constant

that is sufficiently large. In some cases, $y_{(i,j)}$ for some $(i,j) \in E$ may become arbitrarily large in absolute value, i.e., $|y_{(i,j)}| \to \infty$. In such case, (4.12) and (4.13) also play a normalization role. In (4.14), $D_j = \min\{d_j, k-1\}$ where d_j is a degree of node j. Thus, if an edge (i,j) is selected, its corresponding node should be selected. An edge (i,j) is selected in the classification model only if $y_{(i,j)} \neq 0$ in the solution of P_c^R. Let $E^{'}$ be a set of edges which are in the solution of P_c^R such that $E^{'} = \{(i,j)|x_{(i,j)} \neq 0\}$. From the solution of P_c^R, the value of estimated parameters are obtained and following classification functions can be constructed:

$$c_s = x_0 + \sum_{(i,j)\in E^{'}} x_{(i,j)} w^s_{(i,j)} \tag{4.19}$$

If $c_s < 0$ which is calculated by (4.19), a s-th subject is classified as a class 1 (i.e., $s \in S_1$) . Otherwise, it is classified as a class 2 ($s \in S_1$) if $c_s > 0$ and we define that $c_s = 0$ is not classified case.

The most common algorithm to MIP is a branch and bound, in which bounds are typically derived from the linear (continuous) relaxation of the model. The efficiency of branch and bound is directly related to the tightness of those bounds. In the P_c^R, two continuous variables (z^s_- and z^s_+) and one integer variable (y_n) are included in the objective function. In the linear relaxation of P_c^R, it is unnecessary to consider tightness of bounds for continuous variables in the linear relaxation. However, y_n will be assigned the value: $\frac{\sum_{(i,n)\in E} x_{(i,n)}}{M(|N|-1)}$ assuming that $x_{(i,n)}$ is in the solution having nonzero value. Even with conservative choices of M, theses positive values for the relaxed variables are too close to zero to provide tight objective bounds. Hence, nodes in the search tree are difficult to fathom, the search procedure deeper into the tree, and for even modest data sizes the procedure takes too long to complete.

4.4 Benders Decomposition Approach

An important concern regarding constructing and solving optimization problems is that the amount of memory and the computational effort required to solve such problems growing significantly with the number of variables and constraints. The traditional approach such as general branch-and-bound involves making all decision variables simultaneously by solving

a monolithic optimization problem, quickly becomes intractable as the number of variables and constraints increases.

Multi-stage optimization algorithms, such as the benders decomposition [19], have been developed as an alternative solution approach to alleviate those difficulties. Differently from the traditional algorithms, these approaches divide the solving process into several stages. In benders decomposition, a first-stage master problem is solved for a subset of variables, and the values of remaining variables are determined by a second stage sub-problem given the values of the first stage variables. If the sub-problem determines that the proposed first stage decisions are infeasible, the additional constraint is generated and inserted to the master problem, which is then repeated to solve iteratively. Following this manner, a number of small problems are solved instead of solving single large size problem, which can be justified by the increased computational burden associated with that.

In this section, we apply the Benders decomposition to our P_c^R. First, we select a subset of nodes and edges in the G in a master problem, and then find a classifier which minimize the classification errors. Let us first reformulate the problem in terms of the node and edge variables y_n, $e_{(i,j)}$:

$$MP_c^R: \qquad \text{Minimize} \quad t \tag{4.20}$$

subject to:

$$\sum_{(i,j)\in E} e_{(i,j)} \leq D_j y_j \qquad \forall j \in N, \tag{4.21}$$

$$\sum_{n\in N} y_n \leq k \tag{4.22}$$

$$\sum_{(i,j)\in E} g_{(i,j)} e_{(i,j)} + h \leq 0 \qquad \forall g, h \in \mathcal{F}, \tag{4.23}$$

$$\sum_{(i,j)\in E} g_{(i,j)} e_{(i,j)} + h \leq t \qquad \forall g, h \in \mathcal{O}, \tag{4.24}$$

$$e_{(i,j)},\, y_n \in \{0,1\} \qquad \forall (i,j) \in E, \quad \forall n \in N \tag{4.25}$$

Variable t stand for a surrogate of the objective contribution $\sum_{s\in S_1} z_+^s + \sum_{s\in S_2} z_-^s$. Set $\mathcal{O}$ contains optimality cuts which correct underestimation of the objective value by t. Set $\mathcal{F}$ contains feasibility cuts that eliminate binary solutions x and e for which the subproblem

is infeasible. At the initial stage, we set $\mathcal{O}$ and $\mathcal{F}$ as empty: $\mathcal{O} = \mathcal{F} = \emptyset$. We describe how to determine g and h of added cut constraints in $\mathcal{F}, \mathcal{O}$ as followings.

Given feasible solution of MP_c^R; $(\hat{e}, \hat{y})$, we can find the minimum classification error for the corresponding feasible network, if one exist, by solving the following linear program:

$$SP_c^R(\hat{e}): \quad \text{Minimize} \quad \textstyle\sum_{s\in S_1} z_+^s + \sum_{s\in S_2} z_-^s \tag{4.26}$$

subject to:

$$x_0 + \sum_{(i,j)\in E} x_{(i,j)} w_{(i,j)}^s + z_-^s - z_+^s = -\sigma \qquad \forall s \in S_1, \tag{4.27}$$

$$x_0 + \sum_{(i,j)\in E} x_{(i,j)} w_{(i,j)}^s + z_-^s - z_+^s = \sigma \qquad \forall s \in S_2, \tag{4.28}$$

$$x_{(i,j)} \le M\hat{e}_{(i,j)} \qquad \forall (i,j) \in E \tag{4.29}$$

$$-x_{(i,j)} \le M\hat{e}_{(i,j)} \qquad \forall (i,j) \in E \tag{4.30}$$

$$x_0, \quad x_{(i,j)} \quad \text{unrestricted} \qquad \forall (i,j) \in E, \tag{4.31}$$

$$z_+^s, z_-^s \ge 0 \qquad \forall s \in S \tag{4.32}$$

Let α_s, be an optimal dual multiplier associated with constraints (4.27) and (4.28). Similarly, let $\beta_{(i,j)}^+$ and $\beta_{(i,j)}^-$ be dual multiplier corresponding to (4.29) and (4.30). Then, the dual problem of $SP_c^R(\hat{e})$ is:

$DSP_c^R(\hat{e})$:

$$\text{Maximize:} \quad \sigma\Big(\sum_{s\in S2} \alpha_s - \sum_{s\in S_1} \alpha_s\Big) + M \sum_{(i,j)\in E} \hat{e}_{(i,j)}(\beta_{(i,j)}^+ - \beta_{(i,j)}^-) \tag{4.33}$$

subject to:

$$\sum_{s\in S} \alpha_s = 0 \tag{4.34}$$

$$\sum_{s\in S} \alpha_s w_{(i,j)}^s + \beta_{(i,j)}^+ - \beta_{(i,j)}^- = 0 \qquad \forall (i,j) \in E \tag{4.35}$$

$$\alpha_s^- \le 1 \qquad \forall s \in S_1 \tag{4.36}$$

$$-\alpha_s^+ \le 1 \qquad \forall s \in S_2 \tag{4.37}$$

$$\alpha_s^- \quad free \qquad \forall s \in S_1, \tag{4.38}$$

$$\alpha_s^+ \quad free \qquad \forall s \in S_2, \tag{4.39}$$

$$\beta_{(i,j)}^-, \beta_{(i,j)}^+ \le 0 \qquad \forall (i,j) \in E \tag{4.40}$$

Our decomposition strategy first relaxes MP_c^R and solves it to optimality which yields elements of a k sized subgraph ($\hat{e}$,$\hat{y}$). If SP_c^R has a feasible solution ($\hat{z}$, $\hat{x}$), then ($\hat{e}$,$\hat{y}$,$\hat{z}$, $\hat{x}$) corresponds to an optimal classifier. On the other hand, if SP_c^R is infeasible, then the dual formulation of SP_c^R, DSP_c^R is unbounded according to primal-dual theorem. Let ($\hat{\alpha}$, $\hat{\beta}$) be an extreme ray of DSP_c^R such that $\sigma(\sum_{s\in S2} \hat{\alpha_s}^+ - \sum_{s\in S_1} \hat{\alpha}_-^s) + M \sum_{(i,j)\in E} \hat{e}_{(i,j)}(\hat{\beta}^+_{(i,j)} - \hat{\beta}^-_{(i,j)}) > 0$. Then, all $\hat{e}$ vectors that are feasible must satisfy

$$\sigma(\sum_{s\in S2} \hat{\alpha_s}^+ - \sum_{s\in S_1} \hat{\alpha}_-^s) + M \sum_{(i,j)\in E} \hat{e}_{(i,j)}(\hat{\beta}^+_{(i,j)} - \hat{\beta}^-_{(i,j)}) \leq 0 \tag{4.41}$$

Otherwise (i.e., when DSP_c^R is an optimal), add the following optimality cut

$$\sigma(\sum_{s\in S2} \hat{\alpha_s}^+ - \sum_{s\in S_1} \hat{\alpha}_-^s) + M \sum_{(i,j)\in E} \hat{e}_{(i,j)}(\hat{\beta}^+_{(i,j)} - \hat{\beta}^-_{(i,j)}) \leq t \tag{4.42}$$

to the current master problem MP_c^R. Then, we re-solve updated MP_c^R in the next iteration to obtain new incumbent solutions. Therefore, the generic procedure of the benders decomposition approach is described at Algorithm 6.

4.4.1 Branch and Cut Algorithm

In contrast to early works on the benders decomposition based on the Algorithm 3, with the development optimization solvers, recent researches on the benders decomposition works have been employed a branch-and-cut approach to solve the problem with reducing waste of computational time [15, 36].

In the Algorithm 3, although it have been performed decomposition problems successfully, each time we add a new cut to the master problem, and we have to solve it anew. The new cuts may change the structure of the solution tree, we would spend time revisiting candidate solutions that we have already eliminated at the earlier node.

In order to avoid this wasted repetition, we present an alternative strategy that is to solve master problem only once. We aim at embedding the generation of cuts into the branch and cut framework solving the master problem MP_c^R. Thus, it enables us to avoid considerable rework. Recent studies by [15, 36] proved that using a branch-and-cut algorithm could save computational time significantly compared to the generic algorithm; Algorithm 3.

Algorithm 3: Generic Benders Decomposition Algorithm

Data: Problem instances G_s

Result: Solutions: x^*, y^*, z^*, e^*

Construct MP_c^R, SP_c^R, and DSP_c^R ;
while $\hat{t} > \bar{t}$ **do**
    $\hat{e}$, $\hat{x}$, $\hat{t} \leftarrow$ solve MP_c^R;
    if $SP_c^R(\hat{e}, \hat{x})$ *is infeasible* **then**
        get an extreme ray $\hat{\alpha}$, $\hat{\beta} \leftarrow$ solve DSP_c^R;
        add a cut $(\hat{\alpha}, \hat{\beta})$ to the MP_c^R; `/* add a feasibility cut */`
    else
        $\bar{t} \leftarrow$ obj. value of SP_c^R;
        get an extreme ray $\hat{\alpha}$, $\hat{\beta} \leftarrow$ solve DSP_c^R;
        if $\bar{t} > \hat{t}$ **then**
            add an optimality cut $(\hat{\alpha}, \hat{\beta})$ to the MP_c^R; `/* add an optimality cut */`
        end
    end
end

It is an important fact that adding many cuts early in the branch-and-bound tree aids to reduce the unnecessary visiting of infeasible nodes. However, if we include too many cuts in the early stage, it would deteriorate the computational performance to solve the linear programming relaxation for each node in the branch-and-bound tree obviously. Hence, we decide whether or not to add a cut only at integer nodes. The proposed branch-and-cut algorithm is described in Algorithm 4.

4.5 Computational Results

In this section, we evaluate our two proposed algorithms: the formulation P_c^R and branch and cut approach based on the benders decomposition with comparison of the state-of-the-art methods. For notational convenience, we denote the branch and bound approach for solving P_c^R as `MIP` and the branch and cut approach as `BD` respectively. We demonstrate the performance of our proposed algorithms via a series of experiments on both synthetic and real data.

All our methods were coded in Python 2.7 using IBM ILOG CPLEX 12.4 solver and tested on an Intel Xeon E5335 2.00 GHz with 4GB of RAM. For every test instance, computation time was restricted as 1 hour (3,600 seconds). Only single thread processing was used.

For `MIP`, all parameters have been kept to their default values. Basically, CPLEX explore the branch and cut tree with the dynamic search automatically. For `BD`, since the model does not need to contain all constraints explicitly, we turned off the option about the dual presolving. Also, we prohibited the CPLEX's dynamic search because it might interrupt our own cut generation procedure.

4.5.1 Data Description

Synthetic Datasets. Two different scale of networks are considered; one is a set of complete graphs with $|N| = \{20, 30, 90\}$ and the others are created by Erdös-Rényi random graph model with $|N| = \{80\}$ and four edge creation probabilities $p = \{0.10, 0.25, 0.50, 0.75\}$. Based on the network structure (N, E), feature weights (i.e., edge weights) for each observation are sampled from i.i.d. Normal distribution $N(0, 1)$. We fixed the total number of observation for training/test set to 150. Finally, we were able to obtain input data matrix

Algorithm 4: Branch and Cut Benders Decomposition Algorithm

Data: Problem instances G_s

Result: Solutions: x^*, y^*, z^*, e^*

Construct MP_c^R, SP_c^R, and DSP_c^R ;
$T = \{o\}$ where o has empty branching constraints;
$UB = +\infty$;
while $T \neq \emptyset$ **do**
    choose a node $o^{'}$ from T;
    $T \leftarrow T \setminus \{o^{'}\}$;
    $\hat{e}$, $\hat{x} \leftarrow$ optimal solution of linear relaxation of MP_c^R at node $o^{'}$;
    $\hat{t} \leftarrow$ optimal obj. value of MP_c^R at node $o^{'}$;
    if $\hat{t} < UB$ **then**
        if $\hat{x}, \hat{e}$ *is integral* **then**
            if $SP_c^R(\hat{e}, \hat{x})$ *is infeasible* **then**
                get an extreme ray $\hat{\alpha}$, $\hat{\beta} \leftarrow$ solve DSP_c^R;
                add a cut ($\hat{\alpha}$, $\hat{\beta}$) to the MP_c^R; `/* add a feasibility cut */`
            else
                $\hat{y}$, $\hat{z} \leftarrow$ solve SP_c^R $\bar{t} \leftarrow$ obj. value of SP_c^R;
                if $\hat{t} < \bar{t}$ **then**
                    add an cut ($\hat{\alpha}$, $\hat{\beta}$) to the MP_c^R; `/* add an optimality cut */`
                else
                    $UB \leftarrow \hat{t}$; `/* update new upper bound */`
                    $x^* \leftarrow \hat{x}$, $e^* \leftarrow \hat{e}$, $y^* \leftarrow \hat{y}$, $z^* \leftarrow \hat{z}$; `/* update current solutions */`
                end
            end
        end
        if *cuts are added* **then**
            $T \leftarrow T \cup \{o^{'}\}$;
        end
        if $\hat{e}, \hat{x}$ *are fractional* **then**
            branching T;
        end
    end
end

$W \in \mathbb{R}^{150\times|E|}$, where $W = [w_1^T, w_2^T, ..., w_{150}^T]^T$ and w_i was a collection of edge weight for observation i.

To determine the class of each observation, a sparse vector $\alpha \in \mathbb{R}^{|E|}$ is generated, which contributes the class level $l \in \{-1, 1\}$. Given the set of nodes B which were randomly selected from N, nonzero elements of α were determined by induced subgraph from B. Each weight of nonzero element in α was also sampled from i.i.d. Normal distribution $N(0, 1)$.

After that, by use of determined α, the class l is determined by $l = sign(W\alpha)$. For evaluation our methods, we created testing data sets separately. Thus, under the same configuration of (N, E), we had $W_{training}$ and $W_{testing}$. Also, the l was figured out such that $l_{training} = sign(W_{training}\alpha)$ and $l_{testing} = sign(W_{testing}\alpha)$. The detail information of the synthetic datasets is described at Table 4.1.

Table 4.1: Synthetic Datasets Description

Instances	Types	$\|N\|$	$\|E\|$	$\|B\|$	$\|E^B\|$
CO(20)	Complete Graph	20	190	5	10
CO(30)		30	435	5	10
CO(90)		60	4,005	10	45
ER(0.10)	Random Graph	80	317.90±14.08	8	6.00±0.89
ER(0.25)			785.20±17.32		9.50±1.44
ER(0.50)			1566.93±15.65		15.00±2.50
ER(0.75)			2371.08±18.65		21.70±2.01

Real Datasets. With the recent advanced in neuroimaging technology, the research on brain network becomes has been widely studied. One of the popular research on the brain network analysis is to emphasize constructing networks among brain regions and detecting alterations in connectivity related to brain disease. In this study, we conduct classification task on two neurological disorders (Parkinson's Disease (PD) and Alzheimer's Disease (AD)) brain network datasets created by fMRI. Note that we only considered data sets conducted under resting-state experiment (i.e., task free experiment). One of the distinctive

characteristic of the brain networks, in contrast from conventional networked data, is that nodes and edges are not given but need to extract from the neuroimaging data in terms of the scope of individual research. Thus, Harvard-Oxford atlas (HOA) was used to divide the whole brain into a number of regions (=nodes) [107, 136, 131]. Based on the defined regions, we collected 4D (3D $\times$ time) matrix from fMRI data. We assumed that all regions are interconnected and each connection strength (=edge weight) was calculated by correlation coefficient between two ROIs' time series. As a result, we finally were able to obtain fully connected weighted networks. Since HOA is composed of 96 ROIs, constructed networks have 96 nodes and 4560 edges. For notational convenience, we hereafter denote the data set of Parkinson's Disease as PD and Alzheimer's Disease as ADNI respectively. In terms of the observation size for each dataset, PD contains 25 healthy controls and 21 patients respectively (total 46 observations) and ADNI contains 21 healthy controls and 21 patients respectively (total 42 observations).

4.5.2 Initial Heuristic

One of the key factor for the proposed method is to determine the node cardinality k properly. If k is large, the classifier may completely or almost completely separate the groups in the training data and the constructed model may have many alternative optimal solutions. However, one possible problem with such an optimal solution if over-fitting, i.e., memorizing the pattern in the training data, when too many features (=edges) are in the resulting classifier. Otherwise, i.e., k is too small, we might miss informative nodes/edges accidentally. Thus, a possible strategy is to select the large k first and reduce the k gradually until the fixed k.

The remaining task is how to set the start k. Therefore, we propose the greedy heuristic determine the initial k. Firstly, we calculate individual edge's score in terms of its relevance to the class label l. After that, since we only consider the number of nodes, edges induced by the nodes in the current solution are having the first priority to be selected. In formal, let T be a set of selected edges, $N(T)$ be a set of corresponding nodes in T. Then we have a set of candidate $H(T) = \{(i,j)|i \in N(T), j \in N(T), (i,j) \in E \setminus T\}$. Finally, we solve the

P_c iteratively with $T \cup \{(i,j)\}$ where an edge $(i,j) \in H(T)$ and select edges which reduce the objective value (i.e., classification error) of P_c. Recall that parameter y is restricted by M in P_c^R. Thus, we replace the range of y from unrestricted to $|y| \leq M$ and denote the modified P_c as P_c'.. If there is no more edges to reduce the objective value, then we expand the search space by the order of edge score. A pseudocode of the algorithm is described in Algorithm 5

As an edge score, we introduce the Fisher distance which measure each edge's importance:

$$\theta_{(i,j)} = \frac{|S_1|(\bar{w}^1_{(i,j)} - \overline{w}_{(i,j)})^2 + |S_2|(\bar{w}^2_{(i,j)} - \overline{w}_{(i,j)})^2}{\sum_{s \in S_1}(w^s_{(i,j)} - \bar{w}^1_{(i,j)})^2 + \sum_{s \in S_2}(w^s_{(i,j)} - \bar{w}^2_{(i,j)})^2}, \tag{4.43}$$

where $\bar{w}^1_{(i,j)} = \frac{1}{|S_1|}\sum_{s \in S_1} w^s_{(i,j)}$, $\bar{w}^2_{(i,j)} = \frac{1}{|S_2|}\sum_{s \in S_2} w^s_{(i,j)}$ and $\overline{w}_{(i,j)} = \frac{1}{(|S|)}\sum_{s \in S} w^s_{(i,j)}$. Generally, (4.43) have been widely used to screen the features for subset selection as a preliminary process [183, 89]. For example, [183] selected features (=genes) whose Fisher distance were greater than 0.3 before conducting a classification process.

Algorithm 5: Heuristic Algorithm for Finding the Best Subnetwork

Data: Problem instances G_s, edge score θ, stopping criteria ϵ

Result: Solutions: T^*, $|N(T^*)|$

```
Choose the start edge e_0 = arg max_(i,j) θ ;
Append the start edge e_0 to the solution edge set, T = {e_0} ;
obj_best ⟵ ∞;
while z > ε do
    sort current search space H(T) by θ;
    for e ∈ H(T) do
        obj_curr ⟵ P'_c(T ∪ {e});
        if obj_curr < obj_best then
            T ⟵ T ∪ {e};
            obj_best ⟵ obj_crnt;
        end
    end
    construct new search space H(T);
    update N(T)
end
```

Note that the Algorithm 5 should be terminated when $z \leq \epsilon$. We set $\epsilon = 10$ as a default value for practical use. From Algorithm 5, we obtain a $|N(T^*)|$ sized sub-network whose classification error is less than ϵ. Therefore, we employ $|N(T^*)| - 1$ as the upper bound of k in P_c^R for the computational experiments as follows.

4.5.3 Comparison Methods

For evaluation task, we compare our proposed methods; `MIP` and `BD` with two state-of-the-art feature selection methods, including ℓ_1 regularized support vector machine (ℓ_1-`SVM`) and ℓ_1 regularized logistic regression (ℓ_1-LR). Given networked data for each i-th instance (W_i, l_i), where W_i is a collective vector of edge weights and l_i is a class label, ℓ_1-SVM and ℓ_1-LR are defined as following optimization problems:

$$\ell_1 - \texttt{SVM}: \quad \min_{\mathbf{y}} \|\mathbf{y}\|_1 + C \sum_{i=1}^{n} \max(0,\ 1 - l_i(\mathbf{y}^T W_i))^2 \tag{4.44}$$

$$\ell_1 - \texttt{LR}: \quad \min_{\mathbf{y}} \|\mathbf{y}\|_1 + C \sum_{i=1}^{n} \log(1 + \exp(-l_i(\mathbf{y}^T W_i))) \tag{4.45}$$

, where C is a hyper-parameter to control the trade-off between classification error and regularization phase. If we replace the ℓ_1 regularizer $\|\mathbf{y}\|_1$ to ℓ_2 norm $\|\mathbf{y}\|_2$, then (4.44) and (4.45) are equivalent to the standard SVM and logistic regression eventually. Among the various algorithms to solve (4.44) and (4.45) such as interior point method, generalized linear model with elastic net (GLMNET) and so on [70, 211]. LIBlinear, which employed the coordinate descent algorithm to solve the non-smooth optimization problem, has shown the best performance in terms of classification efficiency [211]. Therefore, we include the LIBlinear solver for comparison. Moreover, we take the standard SVM and LR classifier as the baseline which include the all features for classification task. Both two standard classifier were solved by LIBlinear solver as well as ℓ_1 regularization methods. Note that we keep the all parameters of LIBlinear as default.

4.5.4 Performance Evaluation: Synthetic Dataset

First, we evaluate within our models in terms of computational efficiency. In Table 4.2, CPU time spent for solving the synthetic instances are described. Also, since we limit the

CPU time up to 3,600 seconds, we calculate the number of instances which can be solved optimality within the time limit. From the column of CPU time, BD solve the instances faster than the MIP in overall. Also, comparing with the number of instances proving optimality, BD outperforms MIP on the instance CO(90) with $k = 5$ and $k = 10$.

Table 4.2: CPU time and number of instances solved optimality on the complete graphs in synthetic dataset

Instacnces	k	CPU time (sec.)		# of opt. instances	
		BD	MIP	BD	MIP
CO(20)	4	35.10±15.92	70.06±24.62	10/10	10/10
	5	18.58±3.02	34.26±2.50	10/10	10/10
	6	9.56±5.77	25.82±15.31	10/10	10/10
CO(30)	4	1354.29±345.75	2661.70±150.20	10/10	10/10
	5	250.16±80.55	450.45±101.92	10/10	10/10
	6	202.39±50.07	377.02±25.02	10/10	10/10
CO(90)	5	3475.10±67.25	3600.00	2/10	0/10
	10	2266.74±500.29	3458.54±64.62	7/10	2/10
	15	45.94±9.01	366.64±50.25	10/10	10/10
ER(0.10)	6	3494.60±159.14	3540.10±146.09	5/10	3/10
	8	2781.33±261.12	3214.70±379.52	9/10	6/10
	10	1137.20±299.93	1834.29±356.38	10/10	10/10
ER(0.25)	6	3600	3600	0/10	0/10
	8	3487.33±194.12	3553.33±147.68	3/10	1/10
	10	2511.20±591.14	2884.02±356.38	10/10	10/10
ER(0.50)	6	3600	3600	0/10	0/10
	8	3561.33±161.12	3600.00	2/10	0/10
	10	3193.31±259.74	3488.10±121.88	5/10	3/10
ER(0.75)	6	3600.00	3600.00	0/10	0/10
	8	3166.74±500.29	3571.54±91.62	3/10	1/10
	10	3430.20±299.93	3586.11±59.77	2/10	1/10

In addition, we can observe that CPU time is increasing with the node size of instances. It is obvious that large size instances deteriorate the computational efficiency. Interestingly, we also observe that CPU time is affected by the k. Thus, the smaller k take much time to solve the instances. Recall that our objective is to minimize the deviation from the separating hyperplane for two groups. Hence, if we set the large k, we can have more chance to include edges to reduce the sum of deviation. It is a line with a general case of the conventional classification problems. If we consider more variables to classify the datasets, the training efficiency might be increased.

From the Table 4.2, we can also found out that there exist several instances where both of our algorithms fail to find the optimal solution. Although we cannot find the optimal solution, it is worthwhile to compare obtained time limit feasible solutions with each other. Let $Z^{\star}_{BD}$ and $Z^{\star}_{MIP}$ be the time limit feasible solution of `BD` and `MIP` respectively. Then, we count the number of instances in terms of solution comparison and present at the following Table 4.3.

Table 4.3: Solution quality comparison for instances whose solution status are time limit feasible

	CO(90)		ER(0.10)		ER(0.25)		ER(0.50)			ER(0.75)		
k	5	10	6	8	6	8	6	8	10	6	8	10
$z^{\star}_{BD} > z^{\star}_{MIP}$	0	0	1	1	2	1	2	0	0	0	1	0
$z^{\star}_{BD} = z^{\star}_{MIP}$	0	2	2	5	0	0	0	0	3	0	0	1
$z^{\star}_{BD} < z^{\star}_{MIP}$	10	8	7	4	8	9	8	10	7	10	9	9

From Table 4.3, it can be seen that `BD` provide a good solution rather than `MIP`. The solution quality, i.e., which one provide lower objective value, is highly related to the classification performance of the model since the objective value ($Z^{\star}_{BD}$ or $Z^{\star}_{MIP}$) actually represents the training efficiency of the input dataset. Thus, the model providing lower objective value is more likely to construct good classifier eventually. Of course, the lower value cannot always guarantee the good classifier due to risk of the over-fitting.

Once we solve the instances, then we can obtain the classifier function (4.19). With use of (4.19), we conduct classification test based on the generated test datasets.

In Table 5.3, we describe the classification accuracy with different node cardinality k on the synthetic datasets. To compare the classification performance fairly, once we obtain the solution from our models, we extract the same number of features from ℓ_1-SVM and ℓ_1-LR by manual adjustment $C \in [10^{-4}, 10^{-1}]$ in (4.44) and (4.45) since it is impossible to select desired number of edges and nodes exactly through ℓ_1-SVM and ℓ_1-LR while our methods can find k nodes exactly. Note that $|N|$ in the Table 4.4 represent a predetermined cardinality k.

Table 4.4: Test results on the synthetic dataset

Instances	Solution Size								Classification Accuracy					
	$\|E\|$				$\|N\|$									
	BD	MIP	ℓ_1-SVM	ℓ_1-LR	BD	MIP	ℓ_1-SVM	ℓ_1-LR	BD	MIP	ℓ_1-SVM	ℓ_1-LR	SVM	LR
CO(20)	6.0	6.0	6.0	6.0	4.0	4.0	6.0	5.0	**0.88**	**0.88**	0.84	0.83	0.67	0.68
	10	10	10	10	5	5	11	12	**0.94**	**0.94**	0.86	0.86		
	15	15	15	15	6	6	17	17	**0.89**	**0.89**	0.84	0.85		
CO(30)	6	6	6	6	4	4	6	5	**0.81**	**0.81**	0.83	0.84	0.65	0.66
	10	10	10	10	5	5	11	12	**0.95**	**0.95**	0.86	0.87		
	15	15	15	15	6	6	17	16	**0.91**	**0.91**	0.84	0.88		
CO(90)	10	10	10	10	5	5	17	19	0.62	0.64	**0.70**	0.67	0.52	0.54
	45	45	45	45	10	10	59	54	**0.88**	0.82	0.68	0.64		
	75	75	75	75	15	15	74	73	0.67	0.62	**0.71**	0.69		
ER(0.10)	5.3	5.5	5.2	5.2	6.0	6.0	8.2	8.4	**0.89**	0.88	0.87	0.85	0.65	0.64
	7.5	7.2	7.3	7.3	8.0	8.0	11.8	11.1	**0.93**	**0.93**	0.91	0.90		
	12	11.1	11.5	11.5	10.0	10.0	13.5	12.1	**0.90**	**0.90**	**0.90**	0.88		
ER(0.25)	6.2	6.5	6.3	6.2	6.0	6.0	9.0	11.5	0.84	0.85	**0.88**	0.87	0.59	0.61
	9.9	9.1	9.6	9.5	8.0	8.0	18.2	19.1	**0.91**	0.90	0.86	0.86		
	16.8	17.3	16.5	17.1	10.0	10.0	24.0	21.3	0.82	0.83	**0.86**	0.85		
ER(0.50)	7.8	7.9	7.9	7.9	6.0	6.0	8.3	7.1	**0.78**	0.76	**0.78**	0.75	0.56	0.58
	15.2	14.5	16.2	15.4	8.0	8.0	24.7	28.2	**0.90**	0.89	0.82	0.81		
	23.1	22.4	22.4	23.9	10.0	10.0	28.1	26.0	**0.89**	0.88	0.80	0.79		
ER(0.75)	13.1	11.5	12.1	12.1	6.0	6.0	19.1	16.5	**0.73**	0.72	0.65	0.66	0.59	0.59
	24.5	26.3	24.5	24.5	8.0	8.0	33.1	32.4	**0.87**	0.82	0.79	0.78		
	37.6	38.2	38.1	38.0	10.0	10.0	48.3	44.2	**0.81**	0.72	0.79	0.79		

From Table 4.4, we observe that ours outperform than conventional feature selection meth-

ods (ℓ_1-SVM & ℓ_1-LR) as well as the baseline classifiers. Note that BD and MIP have equivalent classification accuracy on CO(20) and CO(30) because the optimal solution of both two methods are same. However, they have different accuracy on CO(90) due to the fact that they have different time limit feasible solution. The quality of the time limit feasible solutions for both our methods can be observed at Table 4.3 more clearly.

In the result of CO(90), we can observe that ours perform lower than the comparison methods at $k = 5$ and $k = 15$. This can be explained that when $k = 5$, we might ignore informative edges because the total number of edges should be less than 10 ($=\binom{5}{2}$). However, when $k = 15$, the over-fitting can be occurred because too many edges are included in the resulting function. Thus, those two cases imply that we should select the k carefully to obtain a satisfactory result.

From the results in Table 4.4, the performance of ℓ_1 methods gradually becomes worse along with the increasing of the edge creation probability from 0.10 to 0.75. According to Table 4.1, the edge creation probability is directly related to the edge density of the graphs. Thus, we conclude that conventional feature selection algorithms cannot perform better than ours under the dense graphs. From the result of ER(80,0.5) and ER(80,0.75), we can observe that ours accuracy are decreasing at some degree. This is caused by the declining the number of instances solved optimality in Table 4.2.

4.5.5 Performance Evaluation: Real Dataset

On the contrary from the experiments on the synthetic datasets, there is no clear distinction between testing and training data in real datasets. Therefore, we employ cross-validation technique to estimate how accurately our models will perform in practice. Since we do not have abundant observation size, we use the 'leave-one-out' cross validation method to validate classification models. In this method, we leave one observation out in turn and use the other $n-1$ observations to set up our models; BD and MIP. We then use the estimated classification function to classify the observations that is left out. This process is repeated n times for all observations. Finally, we count the number of misclassified observations and calculate the cross validation rate. First, we compare the performance in terms of CPU

time for our proposed method by describing on Table 4.5.

Table 4.5: CPU time and number of instances solved optimality on the real dataset

Instacnces	k	CPU time (sec.)		# of opt. instances	
		BD	MIP	BD	MIP
PD	5	3501.28±233.57	3576.21±162.55	4/46	1/46
	6	3230.25±525.49	3451.03±351.17	19/46	8/46
	7	129.74±22.20	446.95±160.05	46/46	46/46
	8	81.74±27.05	267.20±187.83	46/46	46/46
ADNI	5	3513.56±197.98	3600.00	6/42	0/42
	6	3184.77±526.15	3485.94±328.43	22/42	8/42
	7	179.13±154.88	513.91±396.13	42/42	42/42
	8	90.25±37.61	324.31±74.19	42/42	42/42

Table 4.6: Solution quality comparison for instances whose solution status are time limit feasible on the real datasets

	PD		ADNI	
k	5	6	5	6
$z^\star_{BD} > z^\star_{MIP}$	3	2	0	2
$z^\star_{BD} = z^\star_{MIP}$	1	3	0	4
$z^\star_{BD} < z^\star_{MIP}$	42	41	42	36

In the same vein as Table 4.3, we compare the quality of the time limit feasible solutions for BD and MIP. Clearly, BD provides better solution than BD.

In Table 5.5, we evaluate our models with the state-of-the-art methods on the real dataset. Differently from the synthetic data set, we cannot estimate the proper upper bound of k. Instead of relying on empirical trials, we employ the Algorithm 5 described in Section 4.5.2.

Once we run the Algorithm 5 on the real dataset, we obtain subnetworks whose node size are 9 equivalently. However, since we set the ϵ as the lower value, it is more likely to occur the over-fitting. Thus, the cardinality k for real dataset test is varied; $k \in \{5, 6, 7, 8\}$. If we tune the algorithm more wisely, we might obtain a smaller network with proper objective value. We leave it for the future direction. Note that the runtime of the Algorithm 5 is less than a minute.

Table 4.7: Test results on real dataset: PD and ADNI

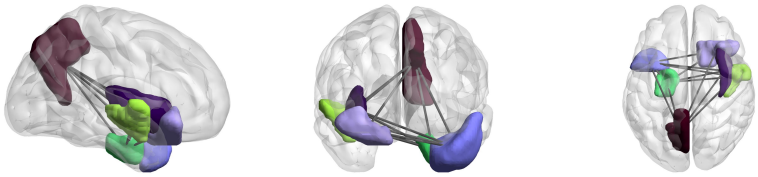
Instances	Size								Accuracy					
	$\|E\|$				$\|N\|$									
	BD	MIP	ℓ_1-SVM	ℓ_1-LR	BD	MIP	ℓ_1SVM	ℓ_1-LR	BD	MIP	ℓ_1-SVM	ℓ_1-LR	SVM	LR
PD	10	10	10	10	5	5	16.78	16.50	**0.61**	0.58	0.58	0.56	0.58	0.56
	15	15	15	15	6	6	23.14	21.55	**0.70**	0.68	0.61	0.58		
	21	21	21	21	7	7	29.78	27.87	**0.65**	0.63	0.61	0.54		
	28	28	28	28	8	8	35.82	33.37	0.57	0.57	**0.59**	**0.59**		
ADNI	10	10	10	10	5	5	13.55	14.90	**0.61**	0.57	0.52	0.57	0.59	0.57
	15	15	15	15	6	6	20.65	20.90	**0.68**	0.65	0.55	0.52		
	21	21	21	21	7	7	25.82	26.49	**0.63**	**0.63**	0.57	0.57		
	28	28	28	28	8	8	30.94	32.77	**0.59**	0.52	**0.59**	0.55		

From Table 4.7, our models outperform the ℓ_1 methods except for $k = 8$. From the classification accuracy, we can infer that $k = 8$ occurs over-fitting problem. Among the various k, $k = 6$ for both data shows the best classification accuracy. It implies that we might miss the informative edges when $k < 6$ and include irrelevant/non-informative edges when $k > 6$. Comparing with the standard methods(`SVM` and `LR`), if the number of selected edges are small ($|E|$=10), the accuracy is worse than SVM and LR with considering all edges. However, if we increase k and include more edges, the accuracy is better than SVM and LR, which verifies the importance of the feature selection scheme on real datasets.

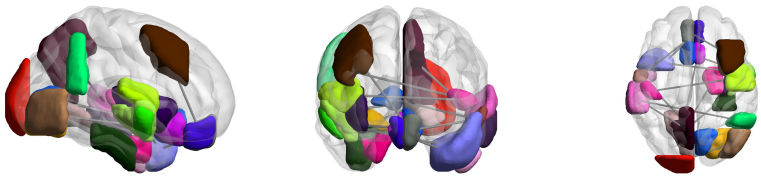
4.5.6 Interpretability Enhancements via Node based Feature Selection

From the previous Sections 4.5.4 and 4.5.5, our proposed methods, thus the node based selection scheme, outperform the conventional feature selection methods through the tests on the synthetic and real datasets. It means the node selection scheme is more effective

to find informative features to classify the observations. Recall that another fundamental objective of the feature selection is proving better understanding of the underlying structural difference between the grouped observations. Therefore, we present the selected subnetworks through two different selection schemes on the real datasets.



(a) Node based feature selection

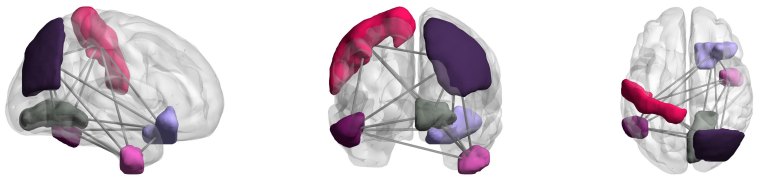


(b) Conventional feature selection

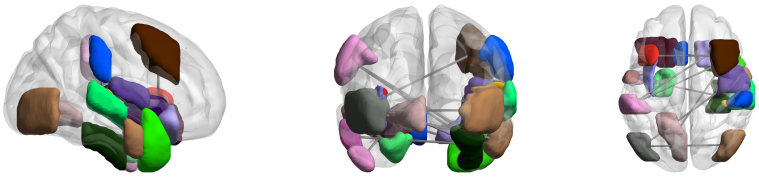
Figure 4.2: Selected sub-networks on PD dataset: Node based feature selection ($|N| = 6$, $|E| = 15$) vs. Conventional feature selection ($|N| = 22$, $|E| = 15$)

Two different subnetworks are depicted in Figure 4.2. Figure 4.2(a) is defined from the node selection scheme `BD` and Figure 4.2(b) is defined from the conventional feature selection scheme ℓ_1-`SVM` respectively. Note that both of them contain same number of edges (features) but different number of nodes. Basically, the regions of the brain have been regarded as highly associated with the neurological disorders. Thus, defining related regions is also important as well as defining related connectivity. For example, the selected nodes in Figure 4.2; 'Temporal pole', 'Precuneous', 'Parahippocampal Gyrus', and 'Orbitofrontal' have been reported that they were significantly related to Parkinson's Disease from the longitudinal functional connectivity study [63]. Besides, recent evidence suggest that one of the selected node;'Insular' activity may be a "precursor" for cognitive processes related

to the task at hand. In addition, it is highly connected with the central executive network, which is associated with Parkinson's disease, as well [168, 134, 54]. In contrast, we cannot find exact relatedness to Parkinson's Disease from some regions in Figure 4.2(b). Thus, we can infer that conventional feature selection scheme might select noisy/irrelevant features and it might not be a reflective of the Parkinson's Disease as a consequence.



(a) Node based feature selection



(b) Conventional feature selection

Figure 4.3: Selected sub-networks on ADNI dataset: Node based feature selection ($|N| = 6$, $|E| = 15$) vs. Conventional feature selection ($|N| = 24$, $|E| = 15$)

In Figure 4.3, out of 6 selected regions, ours select two 'Inferior Temporal Gyrus' which are highly related to semantic memory. According to the literature, Alzheimer's disease and Dementia has been characterized by a patient's inability to integrate semantic memories [45]. Moreover, 'Orbitofrontal' also has been examined the connectivity to Alzheimer's disease [194] and 'Lingual gyrus' is a brain structure that play a role in analysis of logical condition (i.e., logical order of events) and encoding visual memories [133]. The remaining two are 'Postcentral Gyrus' and 'Occipital cortex' which are in charge of sensory reception and visual stimulus respectively. From the selected regions, we can derive various symptoms from the Alzheimer's disease patients. For example, if the disruption is occurred between

temporal area and occipital area, then the patients might have a trouble to recognize the familiar faces and may seem to forget who a familiar person is. Thus, associated connections with 'Occipital Cortex' are also playing an important role to investigate the Alzheimer's disease. Sensory disorder has been often found from the Alzheimer's disease patients [58]. In Figure 4.2(b) and Figure 4.3(b), relatively many nodes are selected with inaccurate classification accuracy comparing with ours. A part of selected nodes in Figure 4.2(b) and Figure 4.3(b) seem to be related to the corresponding diseases. However, we can observe that the fewer nodes can provide more clear view points to understand the selected features. In this study, we will not handle with the detail interpretation of the specific brain functions in terms of neurological pathways since it is beyond of our scope. Instead, we support that node based selection scheme would be beneficial for networked data feature selection in terms of not only predictive performance but also analyzing intrinsic structure of the data.

4.6 Summary

In this chapter, we present a mathematical programming model for the networked data classification. Compared to the extensive researches based on the mathematical programming approaches for the ordinary data classifications, the researches on the networked classification have not paid much attention. Furthermore, considering high dimensionality of the feature domain is less investigated as well. Therefore, based on the presented mathematical model, we propose a node selection model as a new feature selection scheme while reflecting the structure of the data. Through our new proposed scheme, in contrast to the conventional selection scheme such as ℓ_1-SVM or ℓ_1-LR, our proposed scheme allows us to reveal the underlying structure of the data with node based and it should be beneficial when one needs to investigate informative or irrelevant nodes in given data. Additionally, ours can provide much clear view points rather than the conventional methods for the data analysis. Due to the computational intractability of the proposed MIP, we apply the branch-and-cut algorithm based on the benders decomposition. As a result, we can improve the computational efficiency with providing better solution quality. From the conduced experiment on the synthetic datasets, we can observe that ours outperform the state-of-the-art feature selection methods under the conventional feature selection scheme when the relevant features

are affected by a set of nodes. Even the instances becomes denser, the degree of superiority of ours are increasing. Moreover, from the experiments on the real datasets, ours still outperform the conventional approaches. It implies that the status of the networks might be affected by nodes rather than individual edges. Through the results, we can also confirm that selected nodes representing regions in the human brain are pathologically meaningful and it will be beneficial for future researches to investigate the configuration of various neurological disorders via brain connectivity studies.

Chapter 5

CONVEX OPTIMIZATION FOR GROUP FEATURE SELECTION OF NETWORKED DATA

5.1 Introduction

A common goal of machine learning is to estimate an explicit function/model that maps data points of feature vector from the training set to an output target such that the model can be used to make predictions on new, unseen data points in the testing set. Feature selection is a related problem associated with classification, and it often arises when dealing with superfluous information in real world data. High dimensionality of real world data, in both sample space and feature space, can hinder efficiency and efficacy of classification methods. Specifically, the *curse of dimensionality* leads to not only computational inefficiency for processing the data but also diminished classification performance, also known as over-fitting [89].

Motivated by applications in real life networked systems, there is a need of new prediction models that can incorporate the knowledge about the structure of features (i.e., group feature selection) and control the sparsity (size) limit on selected features and their groups (i.e., ℓ_0 norm). In this chapter, we present an efficient computational technique based on mathematical optimization to solve the support vector machine (SVM) group feature selection problem with ℓ_0 norm constraint, denoted as ℓ_0 *norm Group-Regularized SVM (L0GRSVM)*. This problem uses the squared hinge loss in SVM as the loss function $\mathcal{L}(.)$ and the ℓ_0 norm constraint on the number of selected feature groups. Specifically, we extend the group lasso problem introduced in [213] by substituting the $\ell_{2,1}$ regularization term with the $\ell_{2,0}$ norm to prevent undesirable properties of the $\ell_{2,1}$ norm [124] while using the mathematical programming structure of SVM. Due to the combinatorial nature of the $\ell_{2,0}$ norm, we apply a convex relaxation to the $\ell_{2,0}$ norm and construct a new formulation based on quardratically constrained linear program (QCLP). There are an exponential number of

constraints in the QCLP; therefore, we propose an iterative algorithm to efficiently solve the QCLP. Our proposed technique finds the most informative groups of features (B) iteratively by using a computationally efficient approximation algorithm. Once B feature groups are found, our technique determines the feature weight $\mathbf{w}$ that achieves the optimal trade-off between the maximization of the hyperplane distance ($\|\mathbf{w}\|_2^2$) and the minimization of the prediction loss as measured by the loss function in the multiple kernel learning (MKL) problem [158]. Consequently, our technique automatically goes through the re-training process for selected features that helps overcome the bias problem, which is common drawback of the ℓ_1 norm based methods. We apply our technique to both synthetic and real-life datasets that are in the context of networked data classification. We focus on identifying important network nodes (feature groups) instead of the conventional setting that considers edge (feature) selection.

The real-life dataset of complex networked system is from brain connectivity data, where connectivity strengths between two brain regions are features and brain regions are groups of features. The prediction model is trained and a group of brain regions is selected to classify subjects (samples) to normal controls or brain diseases (Parkinson's disease vs. Control and Autistic spectrum disorder vs. Control). Empirical comparison of our technique with state-of-the-art feature selection methods, including ℓ_1-SVM, ℓ_1-Logistic Regression (LR) and group LASSO, demonstrates our technique outperforms these methods. Specifically, our technique yields sparser solutions with competitive predictive power (i.e., classification performance) or better classification performance with similar sparsity of the solutions. In addition, because our technique can control the number of features to be selected while incorporating of knowledge about the structure of features, our technique provides more accurate prediction results and yields more interpretable results that help neurologists obtain a better understanding of systems neuroscience in neurological diseases.

The rest of this chapter is organized as follows. Section 5.2 presents a solution approach for the proposed mathematical model. We conduct computational experiments in Section 4.5, and conclude this study in Section 5.4.

5.2 Solution Approach: Convex Relaxation and Iterative Algorithm

In this section, we first present a mixed integer non-linear programming (MINLP) formulation of ℓ_0 norm Group-Regularized SVM (L0GRSVM), which uses $\ell_{2,0}$ norm group feature selection with SVM structure. Without loss of generality, we model the L0GRSVM for networked data. Since the problem is non-convex, we propose a convex relaxation method to reformulate the problem as a quadratically constrained linear programming (QCLP). We develop an iterative algorithm to efficiently solve the QCLP to obtain an accurate prediction model.

5.2.1 ℓ_0 norm Group-Regularized SVM (L0GRSVM)

Given empirical data $\{\mathbf{x}_i, y_i\}_{i=1}^n$ of n observations, each with m features that are organized by p groups, $G = \{G_1, G_2, ..., G_p\}$, each containing an index set of feature group members, binary label $y_i \in \{+1, -1\}$ and a limit on the number of selected feature groups B, integrating the SVM model in (2.21) and the group feature selection model in (2.23) gives rise to a natural formulation of L0GRSVM, which is given by

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) + \xi_i \geq 1 \quad \forall\, i = 1, ..., n, \\ & \|\mathbf{w}_{G_g}\|_{2,0} \leq B \quad \forall\, g = 1, 2, ..., p, \end{aligned} \tag{5.1}$$

where ξ_i are slack variables for soft margin, C is a hyper-parameter between the margin space and the prediction error, $\mathbf{w}_{G_g} \in \mathbb{R}^{|G_g|}$ is the weight vector of $\mathbf{w}$ related to G_g and $\|\mathbf{w}_{G_g}\|_{2,0}$ is a modified $\ell_{2,0}$ norm $= \sum_{g=1}^{p} I(\|\mathbf{w}_{G_g}\|_2 \neq 0)$ to count the cardinality of selected features groups.

One can reformulate the group feature selection SVM formulation in (5.1) by mapping the input data to a lower dimension $\mathbf{x}_i \rightarrow (\mathbf{x_i} * \mathbf{s})$, where $\mathbf{s} = [s_1, ..., s_m] \in \{0, 1\}^m$ is a vector indicating the binary selection of features and the operator $*$ denotes an element-wise product. The use of this indicator vector is very common in feature selection (e.g., kernel tuning or kernel selection [159, 160, 184] and SVM feature selection [207, 187]).

L0GRSVM can thus be reformulated as a MINLP, given by

$$\begin{aligned} \min_{\mathbf{w},b,\xi,\mathbf{s},\mathbf{v}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\intercal(\mathbf{x}_i * \mathbf{s}) + b) + \xi_i \geq 1 \quad i = 1, ..., n, \\ & |\mathbf{v}\|_0 \leq B, \\ & s_j \leq v_g \quad \forall j \in G_g \quad \forall g = \{1, ..., p\}, \end{aligned} \tag{5.2}$$

where $\mathbf{v}$ is a p-dimensional vector, $\mathbf{v} = [v_1, v_2, ..., v_p] \in \{0, 1\}^p$, indicating the binary selection of feature group ($v_g = 1 \rightarrow g$th group is selected). The $\|\mathbf{v}\|_0 \leq B$ constraint stipulates that at most B feature groups can be selected whereas the $s_j \leq v_g$ constraint implies that if the weight of at least one feature in a group is non-zero, then the feature group must be selected. The following proposition proves that the two formulations in (5.1) and (5.2) have the same objective function value, which indicates the training prediction performance.

Proposition 5.2.1 *Problems* (5.1) *and* (5.2) *have the same objective value.*

Proof Let $(\hat{\mathbf{w}}, \hat{\xi}, \hat{b})$ and $\mathcal{L}(\hat{\mathbf{w}}, \hat{\xi})$ refer to the optimal solution of Problem (5.1) and its corresponding objective value, respectively. Similarly, let $(\overline{\mathbf{w}}, \overline{\mathbf{s}}, \overline{\mathbf{v}}, \overline{\xi}, \overline{b})$ and $\mathcal{L}(\overline{\mathbf{w}}, \overline{\xi})$ represent the optimal solution and its corresponding objective value of Problem (5.2), respectively. We can prove that $\mathcal{L}(\hat{\mathbf{w}}, \hat{\xi}) = \mathcal{L}(\overline{\mathbf{w}}, \overline{\xi})$ by contradiction.

If $\mathcal{L}(\hat{\mathbf{w}}, \hat{\xi}) < \mathcal{L}(\overline{\mathbf{w}}, \overline{\xi})$, then consider a feasible solution of Problem (5.2) as follows: $\mathbf{w}^t = \hat{\mathbf{w}}$, $s_j^t = I_{w_j \neq 0}$, $v_g^t = I_{\mathbf{w}_{G_g} \neq 0}$, $\xi^t = \hat{\xi}$ and $b^t = \hat{b}$ for all $j = 1, ..., m$ and $g = 1, ..., p$. We can easily verify that $(\mathbf{w}^t, \mathbf{s}^t, \mathbf{v}^t, \xi^t, b^t)$ satisfies the constraints given in Problem (5.2). Thus, we have $\mathcal{L}(\hat{\mathbf{w}}, \hat{\xi}) = \mathcal{L}(\mathbf{w}^t, \xi^t) < \mathcal{L}(\overline{\mathbf{w}}, \overline{\xi})$. It contradicts the optimality of the solution $(\overline{\mathbf{w}}, \overline{\mathbf{s}}, \overline{\mathbf{v}}, \overline{\xi}, \overline{b})$.

Conversely, if $\mathcal{L}(\hat{\mathbf{w}}, \hat{\xi}) > \mathcal{L}(\overline{\mathbf{w}}, \overline{\xi})$, we can construct a solution $\mathbf{w}^t = \overline{\mathbf{w}}$ and $\xi^t = \overline{\xi}$ which satisfies $\|\mathbf{w}_{G_g}^t\|_{2,0} \leq B$. Note that if $\overline{v}_g = 0$, then all the $\overline{w}_j$, $j \in G_g$ have to be zero which is equivalent to $\|\overline{\mathbf{w}}_{G_g}\|_2 = 0$. As a result, we have $\mathcal{L}(\hat{\mathbf{w}}, \hat{\xi}) > \mathcal{L}(\mathbf{w}^t, \xi^t) = \mathcal{L}(\overline{\mathbf{w}}, \overline{\xi})$, which contradicts the optimality of $(\hat{\mathbf{w}}, \hat{\xi}, \hat{b})$. ∎

5.2.2 L0GRSVM for Networked Data

In the networked data, each observation i can be represented as an undirected weighted network structure $\{N, E, \mathbf{c}_i\}_{i=1}^n$ where N, E represent the set of nodes and edges, and the

edge weight (e.g., connectivity strength) between two nodes is defined by $\mathbf{c_i} : E \rightarrow \mathbb{R}^{|E|}$. Each observation is associated with a label, denoted by $\{y_i \in \{-1, 1\}\}_{i=1}^{n}$. Note that all observations have the same network structure (N, E) with different edge weights. Thus, the classification of the networked data aims to learn a decision hyperplane from all observation $f(\mathbf{x_i}) = \mathbf{w}^\intercal \mathbf{x_i} + b$, $\forall\, i = 1, 2, ..., n$ where $\mathbf{x_i} = \{c(i, j) \,|\, \forall (i, j) \in E\}$ is the input data. In this conventional setting, feature selection is regarded as edge selection.

In this study, *a node in the network is regarded as a group*. Therefore, a group/node includes all of the edges connected to the node. Thus, group selection leads to node selection. In this setting, a group set G is defined as $G = \{G_1, G_2, ..., G_{|N|=p}\}$ including $|N|$(=the number of nodes) groups and each $G_g \subset \{1, ..., |E| = m\}$ contains an index set of features (=edges) linked to gth group (=gth node). Considering the network structure, we have $G_g = \{(i, g)| \quad \forall (i, g) \in Adj(g)\}$, where $Adj(g)$ is an adjacent edge set of node g. It is worth noting that a feature (=edge) is assigned to two corresponding nodes, simultaneously. Therefore, the problem is an overlapping group feature selection with $\|\mathbf{v}\|_0 \leq B$. We can extend the L0GRSVM formulation in (5.2) for networked data as:

$$
\begin{aligned}
\min_{\mathbf{w},b,\xi,\mathbf{s},\mathbf{v}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \\
\text{s.t.} \quad & y_i(\mathbf{w}^\intercal(\mathbf{x}_i * \mathbf{s}) + b) + \xi_i \geq 1 \quad i = 1, ..., n, \\
& \sum_{g=1}^{p} v_g \leq B, \\
& s_j \leq v_g, \quad \forall i \in Adj(g), \quad \forall g = \{1, ..., p\}.
\end{aligned}
\tag{5.3}
$$

5.2.3 Convex Relaxation

In Problem (5.3), the non-zero elements of the optimal solution ($\mathbf{s}$, $\mathbf{v}$) constitute a sub-network that contains at most B nodes. We define such a sub-networks as B sized network. Let define group (node) domain by $\mathcal{V} = \{\mathbf{v}| \sum_{g=1}^{p} v_g \leq B\}$ and feature (edge) domain by $\mathcal{E}(\mathcal{V}) = \{\mathbf{s} \,|\, s_j \leq v_g,\, \forall j \in Adj(g),\, \forall g \in \{1, ..., p\},\, \mathbf{v} \in \mathcal{V}\}$. Consequently, Problem (5.3) can

be rewritten as

$$\begin{aligned} \min_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{2}\sum_{i=1}^{l}\xi_i^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\intercal(\mathbf{x}_i * \mathbf{s}) + b) + \xi_i \geq 1 \quad i = 1, ..., n. \end{aligned} \tag{5.4}$$

This problem is extremely hard to solve due to the combinatorial nature of $\mathcal{E}(\mathcal{V})$. Nevertheless, once $\mathbf{s}$ is determined, the inner problem of (5.4) becomes a standard SVM, which can be solved efficiently. Inspired by [122], we introduce a convex relaxation to deal with the computational intractability of Problem (5.4).

Assuming $\mathbf{s} \in \mathcal{E}(\mathcal{V})$ is predefined and given, we can rewrite the dual of inner problem (5.4) as

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\|\sum_{i=1}^{l}\alpha_i y_i(\mathbf{x}_i * \mathbf{s})\|_2^2 - \frac{1}{2C}\boldsymbol{\alpha}^\intercal\boldsymbol{\alpha} + \sum_{i=1}^{n}\alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0, \quad i = 1, ..., n, \\ & \alpha_i \geq 0, \quad i = 1, ..., n, \end{aligned} \tag{5.5}$$

where $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_n]$ is a dual vector for the constraints in Problem (5.4). We define a domain of $\boldsymbol{\alpha}$, $A = \{\boldsymbol{\alpha}|\boldsymbol{\alpha}^\intercal\mathbf{y} = 0, \forall\alpha_i \geq 0, i = 1, ..., n\}$ and define the objective function of Problem (5.5) with (-1) multiplication as: $h(\boldsymbol{\alpha}, \mathbf{s}) = \frac{1}{2}\|\sum_{i=1}^{l}\alpha_i y_i(\mathbf{x}_i * \mathbf{s})\|_2^2 + \frac{1}{2C}\boldsymbol{\alpha}^\intercal\boldsymbol{\alpha} - \sum_{i=1}^{n}\alpha$. Thus, Problem(5.4) can be rewritten as:

$$\min_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} \max_{\boldsymbol{\alpha}\in A} -h(\boldsymbol{\alpha}, \mathbf{s}). \tag{5.6}$$

We derive the lower bound of Problem (5.6) by applying the minimax relaxation and exchanging $\min_{\mathbf{s}\in\mathcal{E}(\mathcal{V})}$ and $\max_{\boldsymbol{\alpha}\in A}$ [123, 108] and obtain

$$\max_{\boldsymbol{\alpha}\in A} \min_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} -h(\boldsymbol{\alpha}, \mathbf{s}) = \min_{\boldsymbol{\alpha}\in A} \max_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} h(\boldsymbol{\alpha}, \mathbf{s}). \tag{5.7}$$

Solving Problem (5.7) directly is computationally intensive. Thus, we introduce an additional bounding variable $\eta \in \mathbb{R}$ [93, 187, 122, 50, 71] and reformulate Problem (5.7) as a quadratically constrained linear problem (QCLP) problem, which is given by

$$\text{QCLP}: \min_{\boldsymbol{\alpha}\in A, \eta\in\mathbb{R}} \eta \quad \text{s.t.} \quad \eta \geq h(\boldsymbol{\alpha}, \mathbf{s}) \quad \forall\mathbf{s} \in \mathcal{E}(\mathcal{V}). \tag{5.8}$$

The QCLP in (5.8) is convex in terms of $\boldsymbol{\alpha}$ for each $\mathbf{s}$. However, it is still difficult to solve due to the exponential number of constraints in (5.8). Therefore, we develop an iterative algorithm in the next subsection to efficiently handle with the exponential number of constraints.

5.2.4 Iterative Algorithm

Inspired by [50], we develop an iterative algorithm to solve the QCLP in (5.8). The proposed algorithm consists of two main steps: (i) computing an intermediate SVM solution with a restricted subset of constraints, and (ii) updating the subset of constraints according to the obtained intermediate SVM solution. The algorithm belongs to a family of algorithms for solving general QCLP problems known as the exchange methods in which the constraints are exchanged at each iteration.

From the QCLP in (5.8), we formate the *restricted mater problem (RMP)* based on a restricted subset of constraints, called *localized set* F_l. The RMP is given by

$$\text{RMP}: \min_{\boldsymbol{\alpha}\in A, \eta\in\mathbb{R}} \eta \quad \text{s.t.} \quad \eta \geq h(\boldsymbol{\alpha}, \mathbf{s}_t) \quad \forall \mathbf{s}_t \in F_l. \tag{5.9}$$

The solution to this RMP in (5.9) yields an intermediate sub-optimal solution $(\eta, \boldsymbol{\alpha})$ to the QCLP in (5.8). Our algorithm iteratively solve this RMP by adding a new constraint of B-sized subnetwork with the maximum violation until a stopping criterion is met. A new constraint is generated by solving the *Constraint Generation Problem (CGP)*, which is to find a new B-sized subnetwork, $\mathbf{s}_{l+1} \in \mathcal{E}(\mathcal{V})$, that maximizes $h(\boldsymbol{\alpha}, \mathbf{s}_{l+1})$.

This procedure of our iterative algorithm is outlined in Algorithm 6. Considering each $\mathbf{s}_l$ represents a B-sized subnetwork, the algorithm is to find the B-sized sub networks iteratively and the composition of all determined subnetworks will be the final solution. Thus, the total number of nodes (groups) is bounded by lB after l iterations.

Solving the RMP

Given a set of B-sized subnetworks F_l, we can solve Problem (5.9) as follows. Define X_t as a subset of X denoted by $\mathbf{s}_t \in F_l$. The KKT conditions of Problem (5.9) implies that the

Algorithm 6: Iterative Algorithm for the QCLP in (5.8)

$\boldsymbol{\alpha}^0 \leftarrow C$;
$F_0 = \emptyset \leftarrow$ localized set;
$l = 1 \leftarrow$ iteration step;
repeat
    Find the most violated constraint $\mathbf{s}_l$ based on $\boldsymbol{\alpha}^{l-1}$ using CGP ;
    $F_l \leftarrow F_{l-1} \cup \{\mathbf{s}_l\}$;
    $\boldsymbol{\alpha}^l, \eta^l \leftarrow$ Solve the RMP;
    $l \leftarrow l + 1$;
until *convergence*;

Lagrangian multiplier $\boldsymbol{\lambda}$ must satisfy $\sum_{\mathbf{s}_t \in F_l} \lambda_t = 1$. Therefore, solving the RMP (5.9) is equivalent to solve the following optimization problem:

$$\max_{\boldsymbol{\alpha} \in A} \min_{\boldsymbol{\lambda} \in Q} -\sum_{\mathbf{s}_t \in F_l} \lambda_t h(\boldsymbol{\alpha}, \mathbf{s}) = \min_{\boldsymbol{\lambda} \in Q} \max_{\boldsymbol{\alpha} \in A} -\frac{1}{2}(\boldsymbol{\alpha} * \mathbf{y})^\intercal (\sum_{\mathbf{s}_t \in F_l} \lambda_t X_t X_t^\intercal + E)(\boldsymbol{\alpha} * \mathbf{y}), \tag{5.10}$$

where Q is a domain for Lagrangian multiplier $\boldsymbol{\lambda}$ such that $Q = \{\boldsymbol{\lambda} \mid \sum_{\mathbf{s}_t \in F_l} \lambda_t = 1\}$ and $E = \frac{1}{C} I$, where I is an n-dimensional identity matrix. Note that we can interchange the operators $\max_{\boldsymbol{\alpha} \in A}$ and $\min_{\boldsymbol{\lambda} \in Q}$ because it is concave in $\boldsymbol{\alpha}$ and convex in $\boldsymbol{\lambda}$, respectively [177].

Note that Problem (5.10) is equivalent to the multiple kernel learning (MKL) problem [116], where the kernel matrix to be learned is a convex combination of the base kernel matrices $\{X_t X_t^\intercal \mid \mathbf{s}_t \in F_l\}$. There are several approaches in the literature to solve the MKL (e.g., [116, 160, 148]). Our algorithm uses the *SimpleMKL* approach, which solves the non-smooth optimization problem via sub-gradient method [160].

Solving the CGP

To find the most violated constraint $\mathbf{s}$ at each iteration step, we propose the following procedure to solve the CGP. Recall that the feasibility of each constraint is measured by the corresponding value of η. Therefore, the constraint that yields the largest η can be regarded as the most violated constraint. Hence, it can be calculated as $\mathbf{s}^* = \text{argmax}_{\mathbf{s} \in \mathcal{E}(\mathcal{V})} h(\boldsymbol{\alpha}, \mathbf{s})$

with given $\boldsymbol{\alpha}$. Consequently, the CGP can be formulated as

$$\max_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} \quad \frac{1}{2}\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i * \mathbf{s})\|_2^2 + \frac{1}{2C}\alpha^\intercal\alpha - \sum_{i=1}^{n}\alpha_i. \tag{5.11}$$

Proposition 5.2.2 *Problem* (5.11) *is NP-hard.*

Proof In Problem (5.11), $\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i*\mathbf{s})\|_2^2 = \|\sum_{j=1}^{m}(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i)*\mathbf{s}\|_2^2$. Let $(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i) = \boldsymbol{\tau} \in \mathbb{R}^m$. Therefore, each non-negative τ_j can be considered as a non-negative weight on j-th feature (=edge) s_j. Recall that a set of edges must be involved in the B-sized subnetwork (i.e., $\forall\, \mathbf{s} \in \mathcal{E}(\mathcal{V})$). Hence, with removal of the constant term $\frac{1}{2C}\boldsymbol{\alpha}^\intercal\boldsymbol{\alpha} - \sum_{i=1}^{n}\alpha_i$, Problem (5.11) can be reformulated as

$$\begin{aligned} \max \quad & \frac{1}{2}\sum_{j=1}^{m}\tau_j^2 s_j^2 \\ \text{s.t.} \quad & s_j \le v_g \quad \forall j \in Adj(g), \quad \forall g = 1, ..., p \\ & \sum_{i=1}^{p} v_g \le B. \end{aligned} \tag{5.12}$$

The above formulation in (5.12) is equivalent to *the heaviest B subgraph problem* (HSP), which is well-known NP-hard problem [55]. We can thus conclude that the HSP can be reduced to Problem (5.11). ∎

The HSP determines a subset of B nodes such that total edge weight of the subgraph induced by such B nodes is maximized. It is also known as *B-cluster problem* and *dense B-subgraph problem* if all edges have the same weight. Many solution methods have been developed to tackle this problem (e.g., [130, 22]). However, existing methods in the literature can solve only small instances. Recently, a preprocessing routine to reduce the size of a given network while retaining a $(1 + \frac{1}{B})$-approximation for HSP problem have been proposed [34]. Our iterative algorithm also uses this routine to solve Problem (5.11).

5.2.5 Analysis of Convergence

In this section, we analyze the convergence property of Algorithm 6. Assume the current iteration step is l. At the l-th iteration, a new constraint $\mathbf{s}_l$ below can be found via the

CGP derived by $\boldsymbol{\alpha}^{l-1}$

$$h(\boldsymbol{\alpha}^{l-1}, \mathbf{s}_l) = \max_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} h(\boldsymbol{\alpha}^{l-1}, \mathbf{s}). \tag{5.13}$$

As a result, the new constraint set F_l is given by $F_l = F_{l-1} \cup \{\mathbf{s}_l\}$.

Define $ub^l = \min_{1\le k\le l} h(\boldsymbol{\alpha}^{k-1}, \mathbf{s}_k)$ and $(\eta^l, \boldsymbol{\alpha}^l)$ as the intermediate solution pair to Problem (5.9), which is obtained by

$$\boldsymbol{\alpha}^l = \operatorname*{argmin}_{\boldsymbol{\alpha}\in A} \left(\max_{\mathbf{s}_t\in F_l} h(\boldsymbol{\alpha}, \mathbf{s}_t) \right), \tag{5.14}$$

$$\eta^l = \max_{1\le k\le l} h(\boldsymbol{\alpha}^l, \mathbf{s}_k) = \min_{\alpha\in A} \Big(\max_{1\le k\le l} h(\boldsymbol{\alpha}, \mathbf{s}_k) \Big). \tag{5.15}$$

For notational convenience, we introduce lb^l such that $\eta^l = lb^l$. We obtain

$$lb^l = \min_{\boldsymbol{\alpha}\in A} \max_{1\le k\le l} h(\boldsymbol{\alpha}, \mathbf{s}_k), \tag{5.16}$$

$$ub^l = \min_{1\le k\le l} h(\boldsymbol{\alpha}^{k-1}, \mathbf{s}_k) = \min_{1\le k\le l} \Big(\max_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} h(\boldsymbol{\alpha}^k, \mathbf{s}) \Big). \tag{5.17}$$

Based on the definitions of lb^l and ub^l, we present the following two lemmas to prove that Algorithm 6 converges to the optimal solution.

Lemma 5.2.3 *Suppose $(\eta^{opt}, \boldsymbol{\alpha}^{opt})$ is an optimal solution pair of Problem* (5.8)*. Inequality $lb^l \le \eta^{opt} \le ub^l$ holds at the l-th iteration. As iteration step increases, the sequence $\{lb^l\}$ is monotonically increasing and $\{ub^l\}$ is monotonically decreasing.*

Proof Firstly, we have the optimal solution $\eta^{opt} = \min_{\boldsymbol{\alpha}\in A} \max_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} h(\alpha, \mathbf{s})$ from Problem (5.8). For any feasible $\boldsymbol{\alpha}$ in Problem (5.8), we have $\max_{\mathbf{s}\in F_l} h(\boldsymbol{\alpha}, \mathbf{s}) \le \max_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} h(\boldsymbol{\alpha}, \mathbf{s})$ under $F_l \subseteq \mathcal{E}(\mathcal{V})$. With consideration of $\boldsymbol{\alpha}$, we obtain:

$$\min_{\boldsymbol{\alpha}\in A} \max_{\mathbf{s}\in F_l} h(\boldsymbol{\alpha}, \mathbf{s}) \le \min_{\boldsymbol{\alpha}\in A} \max_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} h(\boldsymbol{\alpha}, \mathbf{s}).$$

By Eq. (5.16), we have $lb^l \le \eta^{opt}$.

On the other hand, from Problem (5.13), $\mathbf{s}_k = \operatorname{argmax}_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} h(\boldsymbol{\alpha}^{k-1}, \mathbf{s})$ for $k = 1, ..., l$. Thus, $\{(\boldsymbol{\alpha}^{k-1}, h(\boldsymbol{\alpha}^{k-1}, \mathbf{s}_k))\}_{k=1}^l$ is a set of feasible solutions to Problem (5.8). Since $(\boldsymbol{\alpha}^{opt}, \eta^{opt})$ is the optimal solution to Problem (5.8), any feasible solution $h(\boldsymbol{\alpha}^{k-1}, \mathbf{s}_k))$ satisfies $\eta^{opt} \le h(\boldsymbol{\alpha}^{k-1}, \mathbf{s}_k)$, for $k = 1, ..., l$. This follows from Eq. (5.17) and finally we have $\eta^{opt} \le ub^l$.

Localized set F_l is monotonically expanding with increasing iteration step l. Therefore, the sequence $\{lb^l\} = \{\eta^l\}$ is also monotonically increasing. Along the same line, the sequence $\{ub_l\}$ is monotonically decreasing. ∎

According to the Lemma 5.2.3, the convergence can be traced by the gap between lb^l and ub^l. For example, if this gap is less than a predetermined tolerance, the Algorithm 6 is terminated.

Lemma 5.2.4 *Assume that our algorithm converges at the l-th step. The intermediate solution $(\eta^{l-1}, \boldsymbol{\alpha}^{l-1})$ is the global optimal solution to Problem* (5.8).

Proof Once the algorithm is terminated at the l-th iteration, there is no more update from the $(l-1)$-th step to l-th step. Thus, $F_l = F_{l-1}$ and $\boldsymbol{\alpha}^l = \alpha^{l-1}$. A new constraint $\mathbf{s}_l$ at the l-th iteration is given by $\mathbf{s}_l = \text{argmax}_{\mathbf{s}\in\mathcal{E}(\mathcal{V})}\, h(\boldsymbol{\alpha}^l)$. From Problem (5.13), we have $h(\boldsymbol{\alpha}^{l-1}, \mathbf{s}_l) = \max_{\mathbf{s}\in\mathcal{E}(\mathcal{V})}\ h(\boldsymbol{\alpha}^{l-1}, \mathbf{s})$ and it is equivalent to $h(\boldsymbol{\alpha}^{l-1}, \mathbf{s}_l) = \max_{\mathbf{s}\in F_l}\ h(\boldsymbol{\alpha}^{l-1}, \mathbf{s})$ as the new constraint $\mathbf{s}_l \in F_l = F_{l-1}$. Following from Eqs. (5.16) and (5.17), we obtain

$$\begin{aligned} h(\boldsymbol{\alpha}^{l-1}, \mathbf{s}_l) = \max_{\mathbf{s}\in\mathcal{E}(\mathcal{V})} h(\boldsymbol{\alpha}^{l-1}, \mathbf{s}) = \max_{\mathbf{s}\in F_{l-1}} h(\boldsymbol{\alpha}^{l-1}, \mathbf{s}) = \eta^{l-1} \\ ub^{l-1} = \min_{1\le k\le l} h(\boldsymbol{\alpha}^{k-1}, \mathbf{s}_k) \le \eta^{l-1} \end{aligned} \tag{5.18}$$

Since we already proved that $lb^l \le \eta^{opt} \le ub^l$ in Lemma 5.2.3, we obtain $\eta^l = lb^l$. Consequently, we conclude that $lb^{l-1} = ub^{l-1} = \eta^{l-1} = \eta^{opt}$ and $(\eta^{l-1}, \boldsymbol{\alpha}^{l-1})$ is the optimal solution to Problem (5.8) ∎

5.3 Computational Results

In this study, we carried out several experiments to compare the performance of our feature selection technique, L0GRSVM, with several state-of-the-art methods, in terms of classification accuracy and interpretability of the results. All computations were implemented on a Window Server 2012 machine with Intel Xeon E5335 eight core processor @ 2.00 GHz, 24 GB of RAM. All our methods were implemented by MATLAB R2013a.

5.3.1 Experimental Setup

The baseline ℓ_1-norm feature selection methods used for our performance comparison were ℓ_1-SVM and ℓ_1-logistic regression, which are among the most commonly used embedded feature selection methods. We used the coordinate descent algorithm, developed in [211], to solve these ℓ_1-norm feature selection methods. To evaluate the performance of the proposed L0GRSVM technique, we compared it with the group lasso (group regularized logistic regression) based on the fast iterative shrinkage-thresholding algorithm (FISTA), implemented in the SLEP package [125]. To compare the group lasso in the classification tasks, we substituted the sum of the least squares with the logistic loss. Due to the computational issues in ℓ_1-norm methods, we assumed $w_j = 0$ if $|w_j|/\max_j(|w_j|) \leq 1 \times 10^{-4}$. Also, we set all $w_j = 0$ if $\max_j(|w_j|) \leq 1 \times 10^{-5}$ [44]. With respect to termination condition, we applied a relative tolerance as follows: $\{z(l) - z(l-1)\}/z(0) \leq \delta$ where $z(l)$ is the objective value of (5.10) at the lth iteration. Note that $z(l)$ is monotonically decreasing.

In this section, we use the following terminology as we report the results. We refer to Algorithm 6 for L0GRSVM as networked group feature selection **'NF'**, two standard feature selection methods as **'ℓ_1-SVM'** and **'ℓ_1-LR'** (logistic regression), respectively, and the group regularized logistic regression [125] as **'GR-LR'**. We also included both standard SVM and logistic regression without feature selection components as the baseline of our analysis, and they are denoted by by **'S-SVM'** and **'S-LR'** respectively.

5.3.2 Datasets

In this section, we provide detailed information of the synthetic dataset that we randomly generated and the real dataset from neuroimaging studies.

Synthetic Datasets

We generated two sets of large-scale networks as follows. The first set of networks is based on a complete network (CN) with 150 nodes and 11,175 edges. The second set of networks is based on the Erdös-Rényi random network model [1] (ER) with 350 nodes and 12,388

[1]Sources are available at http://strategic.mit.edu/

edges (the probability of edge creation = 0.2). We then used i.i.d. Gaussian distribution $\mathcal{N}(0,1)$ to assign random weights to edges. We replicated 2,000 networks in this fashion and constructed two data matrices $\mathbf{X}_{CN} \in \mathbb{R}^{2,000\times 11,175}$ and $\mathbf{X}_{ER} \in \mathbb{R}^{2,000\times 12,388}$.

To compare the performance of group feature selection in different network topologies, we constructed two types of sub-networks: *dense (group-structured) networks* and *sparse(non group structured) networks*. These sub-networks actually represent the selected features in the networked data. To construct the dense sub-network, we firstly selected 10% of nodes from each $\mathbf{X}_{CN}$ and $\mathbf{X}_{ER}$. We denoted selected nodes in these sub-networks by N^T ($|N^T| = 15$ for $\mathbf{X}_{CN}$ and $|N^T| = 35$ for $\mathbf{X}_{ER}$). Then, we extracted all of the edges associated with N^T, denoted by E^T, and assigned random edge weights to them (i.e., $w_{(i,j)} \sim \mathcal{N}(0,1)$, $\forall (i,j) \in E^T$ and $w_{(i,j)} = 0$, $\forall (i,j) \notin E^T$). Note that the constructed network is dense since $|N^T| \ll |E^T|$. Because the edges corresponding to $\mathbf{w}$ were defined by a set of nodes, the relevant features were thus organized by a set of groups. Thus, this characteristic of dense networks can be an strong indicator of **presence of group structure**.

To construct sparse networks, 300 edges from both synthetic datasets (i.e., CN and ER) were randomly chosen. The set of selected edges was denoted by E^T. We then selected all the nodes, N^T, associated with E^T and assigned random weight to them $w_{(i,j)} = c(i,j) \sim \mathcal{N}(0,1)$, similar to dense networks. In this construction, corresponding edges in $\mathbf{w}$ were unrelated to the certain groups. Thus, this characteristic of dense networks can be an strong indicator of **absence of group structure**.

For each $\mathbf{w}$, output label was produced by $\mathbf{y} = sign(\mathbf{X}\mathbf{w})$. The characteristics of the constructed synthetic data sets are described in Table 5.1.

Table 5.1: Network characteristics of synthetic datasets.

Instances	Scale	$\|N\|$	$\|E\|$	$\|N^T\|$	$\|E^T\|$	Group structure
SN-150	CN	150	11,175	15	105	Yes
SE-150	CN			149	300	No
SN-350	ER	350	12,388	35	80	Yes
SE-350	ER			289	300	No

Real Datasets

Brain functional networks have been extensively investigated in recent years in search for biomarkers of neurological diseases [179, 204]. Researches have studied connectivity in the brain network by means of various neuroimaging modalities (e.g., electroencephalography (EEG), functional magnetic resonance imaging (fMRI), and positron emission tomography (PET)). In this chapter, we focus on the brain networks, collected by fMRI, from two neurological diseases: Parkinson's Disease (PD) and Autism Spectrum Disorders (ASD). To define nodes in these networked data, Power's functional coordinate system in the brain was used to locate functional nodes in the brain functional network [154]. Based on Power's functional nodes, for all node pairs, the blood-oxygen-level-dependent (BOLD) time series from each node used to calculate the strength of functional connectivity by means of Pearson's correlation coefficient. The resulting correlation coefficients were used as the weights of edges in our netwok, i.e., connectivity strengths between two nodes. Note that brain network is a complete network. Detail information of these real datasets are described in Table 5.2. The objective of analyses of these real datasets was to classify whether subjects were from the normal control group or the disease group (i.e., PD vs. Control, ASD vs. Control).

Table 5.2: Real Datasets Statistics.

Instance	$\|N\|$	$\|E\|$	Size (# of Controls/ # of Patients)	Disease
PD	264	34,716	46(25/21)	Parkinson's Disease
ASD	264	34,716	360(180/180)	Autism Spectrum Disorder

5.3.3 Algorithm Convergence and Sensitivity Analysis

We investigated the convergence of Algorithm 6 in terms of its input parameter B on the synthetic data SN-150. In addition, we performed sensitivity analysis on hyper-parameter C in (5.3). In Figure 5.1, we empirically demonstrated convergence of Algorithm 6 with respect to $B \in \{5, 10, 15\}$ with $C = 10$. The termination criterion was $\delta = 1.0 \times 10^{-3}$.

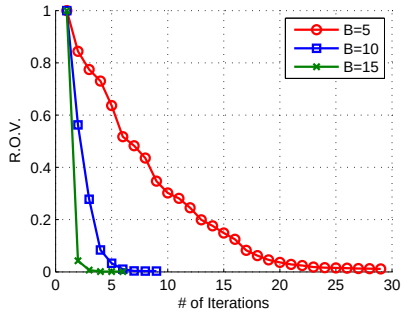
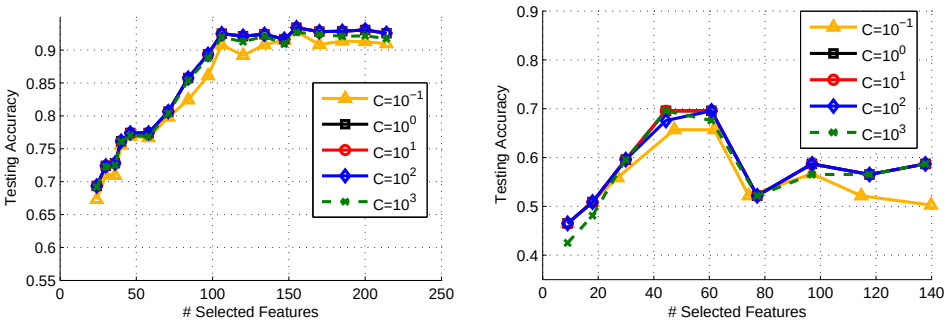


Figure 5.1: Convergence of our algorithm on synthetic data SN-150 with $B \in \{5, 10, 15\}$ in terms of Relative Objective Value (R.O.V.)

Figure 5.1 illustrates that the algorithm converges with similar objective values for all of B. This implies that choosing any B could achieve the convergence. However, the number of iterations to reach convergence varies. The larger B, the fewer iterations are required for the algorithm to converge. This is because increasing B leads to a large number of features to be included at each iteration resulting in a fewer number of the CGPs. However, we also note that some irrelevant features might be included unexpectedly, when B is too large.

We also conducted a sensitivity analysis on hyper-parameter C in Eq. (5.3), which generally represents the trade-off between the loss function and the marginal space for the separating hyperplane in SVM. We varied the value of $C \in \{10^{-1}, 10^0, 10^1, 10^2, 10^3\}$, and performed the analysis on two datasets: SN-150 from the synthetic datasets and PD from the real datasets. The corresponding testing accuracies and the number of selected features as a function of C are illustrated in Figure 5.2 for both datasets. A small value for $C = 10^{-1}$ yielded a poor accuracy compared to the others. Similarly, we observe the poor accuracy when C is too large $C = 10^3$ for the PD data may be due to over-fitting. Except for these two C values, the other cases of $C \in \{10^0, 10^1, 10^2\}$ yielded comparable results. These preliminary results illustrated in Figure 5.2 supports that our method is robust with respect to C, and suggested the good range for the value of C to be $[10^0, 10^2]$.



(a) SN-150 (b) PD

Figure 5.2: Sensitivity analysis of the hyper-parameter C for the Algorithm 6 on two data sets: SN-150 and PD. Variation of C is $C \in \{10^{-1}, 10^0, 10^1, 10^2, 10^3\}$

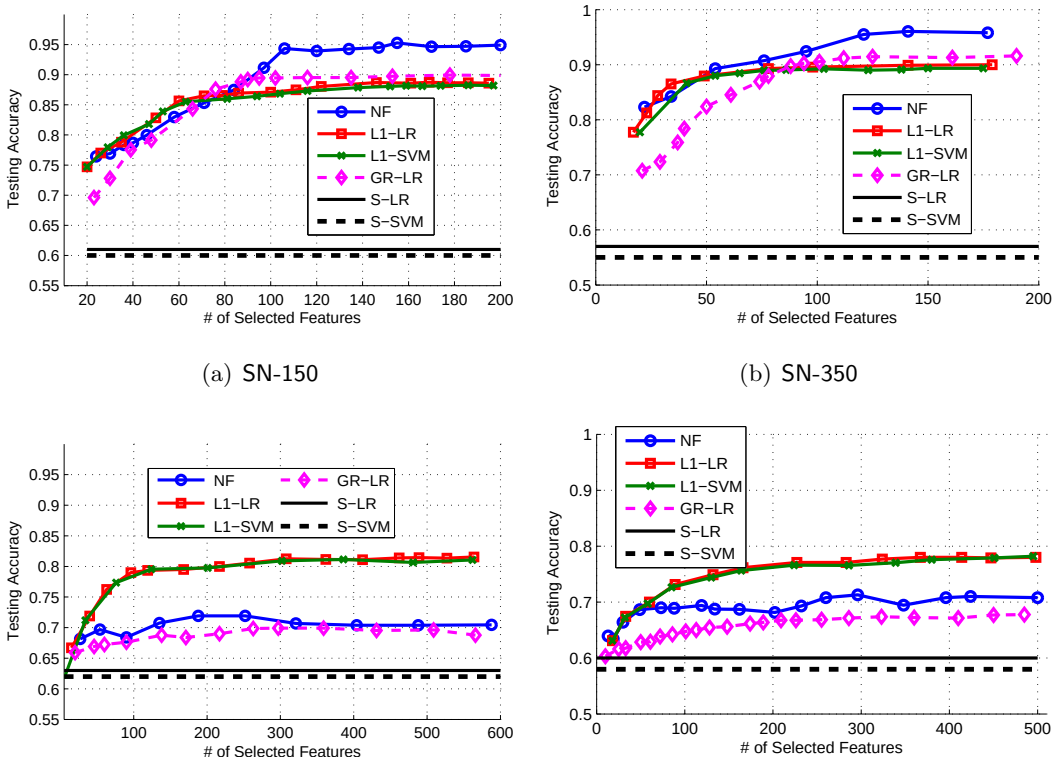
5.3.4 Experimental Results on Synthetic Datasets

We compared the performance of our technique, NF, with other methods including ℓ_1-SVM and ℓ_1-LR, GR-LR, S-SVM and S-LR in terms of testing accuracies and sparsity of the solutions (i.e., number of node selected). For NF, parameter C was set to be 10, the termination criterion was $\delta = 1.0 \times 10^{-3}$, and parameter B was varied to achieve different numbers of selected features and groups. For ℓ_1-SVM, ℓ_1-LR, and GR-LR, the regularization parameter γ was varied. Note that the regularization parameters in ℓ_1-LR, ℓ_1-SVM, and GR-LR were carefully fine-tuned to achieve comparable numbers of features and groups with those by NF. Table 5.3 provides detail information about how the regularization parameters were set in each method.

Figure 5.3 depicts testing accuracies of NF and all other baseline methods on all four synthetic datasets. In datasets with presence of group structure, SN-150 in Figure 5.3(a) and SN-350 in Figure 5.3(b), NF outperforms all other baseline methods, providing the highest classification accuracy. In datasets with absence of group structure, SE-150 in Figure 5.3(c) and SE-350 Figure 5.3(d), as expected, both baseline regularization methods, ℓ_1-LR and ℓ_1-SVM, outperform group regularization methods, NF and GR-LR, and baseline classification methods without regularization, S-LR and S-SVM. It is, however, important to note that NF still outperforms the baseline group feature selection method, GR-LR, and

Table 5.3: Ranges of regularization parameter settings: B in NF, γ for $\mathbf{w}$ in ℓ_1-LR and ℓ_1-SVM, γ for $\sum_{g=1}^{p} \|\mathbf{w}_{G_g}\|_2$ in GR-LR

Instances	NF	ℓ_1-LR	ℓ_1-SVM	GR-LR
SN-150	$B \in \{3,...,10\}$	$\gamma = [0.01,0.03]$	$\gamma = [0.003,0.008]$	$\gamma = [4.55,4.60]$
SN-350	$B \in \{3,...,15\}$	$\gamma = [0.01,0.03]$	$\gamma = [0.003,0.008]$	$\gamma = [4.50,4.60]$
SE-150	$B \in \{3,...,12\}$	$\gamma = [0.01,0.045]$	$\gamma = [0.002,0.010]$	$\gamma = [4.30,4.65]$
SE-350	$B \in \{3,...,20\}$	$\gamma = [0.01,0.045]$	$\gamma = [0.002,0.010]$	$\gamma = [4.25,4.65]$

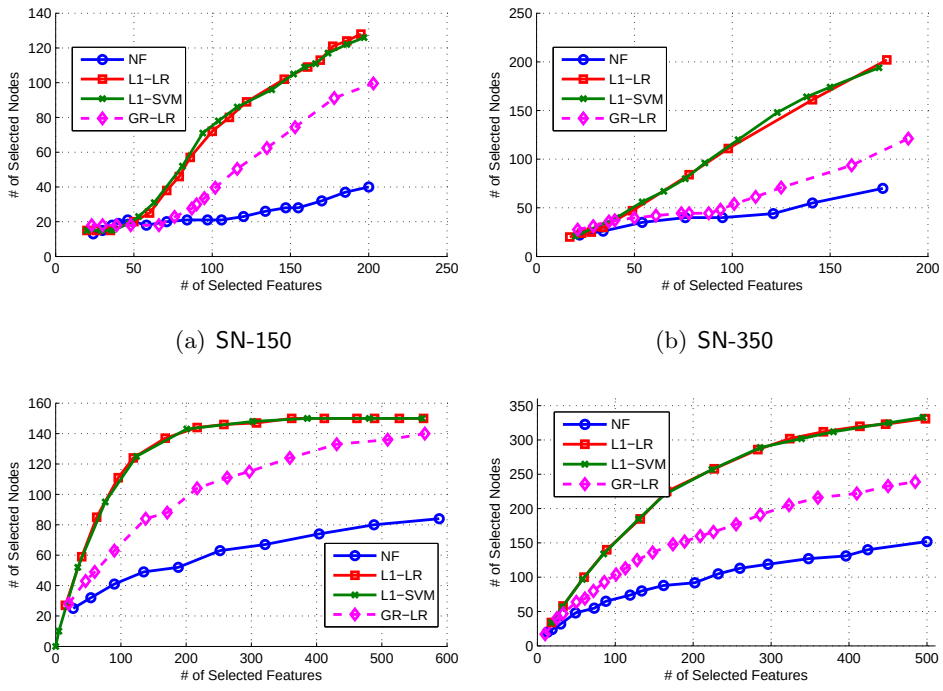


(a) SN-150 (b) SN-350

(c) SE-150 (d) SE-350

Figure 5.3: Comparison of testing accuracies achieved by NF, baseline group feature selection method (GR-LP), baseline feature selection methods (ℓ_1-LR and ℓ_1-SVM), baseline classification methods (S-LR and S-SVM) on synthetic datasets

other baseline classification methods, S-LR and S-SVM. This observation is in line with other research studies (e.g.,[211]) and reinforces the idea that incorporating group structure of data (if any) in a model explicitly can significantly enhance the testing accuracy of a model. On the other hand, on the datasets with absence of group structure, SE-150 and SE-350, both group feature selection methods produce inferior results to the standard feature selection methods. Based on all around results as shown in Figure 5.3, it is clear that NF provides better results in terms of testing accuracies in comparison to GR-LR, and is the most accurate method when applied to the datasets with presence of group structure.



(a) SN-150 (b) SN-350

(c) SE-150 (d) SE-350

Figure 5.4: Comparison of the sparsity of group features achieved by NF, baseline group feature selection method (GR-LP), baseline feature selection methods (ℓ_1-LR and ℓ_1-SVM), baseline classification methods (S-LR and S-SVM) on synthetic datasets

In Figure 5.4, we illustrate the group sparsity of solutions (i.e., the number of selected feature groups) obtained from NF and other baseline methods with respect to the number of selected features. In all datasets, NF was able to achieve the highest group sparsity.

Especially in datasets with presence of group structure, SN-150 in Figure 5.4(a) and SN-350 in Figure 5.4(b), not only does NF provide the highest group sparsity (i.e., using the least number of feature groups) but it also yield the most accurate results (as shown in Figure 5.3). From the figure, NF and GR-LR provide group sparse solutions whereas other feature selection methods (ℓ_1-LR and ℓ_1-SVM) failed to achieve group sparsity. When compared with GR-LR, the proposed NF achieved a much better solution quality in terms of group sparsity and classification performance.

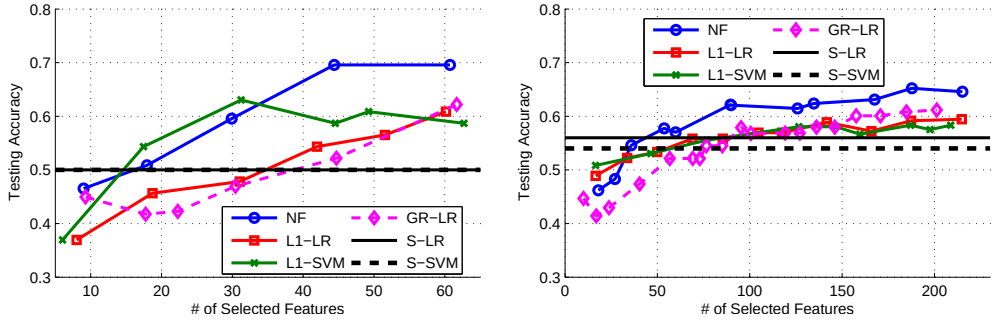
5.3.5 Experiments on Real Datasets

In this section, experimental results on two real data sets are presented. Similar to experiments on synthetic datasets, we evaluated the performance in terms of testing classification accuracy and group sparsity of the solution. To evaluate the proposed classification technique and all baseline methods, we used a k-fold cross validation, in which the original data was randomly partitioned into k equal sized subsamples. From these k subsamples, a single subsample was kept as the validation data for testing the classification model, and the remaining $(k-1)$ subsamples were used as training data. The cross-validation process was repeated k times meaning that each of the k subsamples used exactly once as the validation data. Since there are only 46 data samples in PD dataset, we applied a leave-one-out cross validation, $k = n = 46$, in which a single sample was as the testing set and the remaining $(n - 1 = 45)$ samples as the training set, and the process repeated n times so that all data samples were tested. In ASD dataset, as there are $n = 360$ samples, we applied a 10-fold cross validation.

We also set $C = 10$ and $\delta = 1.0 \times 10^{-3}$, which was the same setting as in the experiments on Synthetic datasets. The details for all parameters are described at Table 5.4.

Table 5.4: Input parameters for real datasets experiments

Instances	NF	ℓ_1-LR	ℓ_1-SVM	GR-LR
PD	$B \in \{3,...,7\}$	$\gamma = [0.60,0.70]$	$\gamma = [0.05,0.75]$	$\gamma = [2.02,2.03]$
ASD	$B \in \{3,...,15\}$	$\gamma = [0.50,1.85]$	$\gamma = [0.10,1.50]$	$\gamma = [2.01,2.03]$

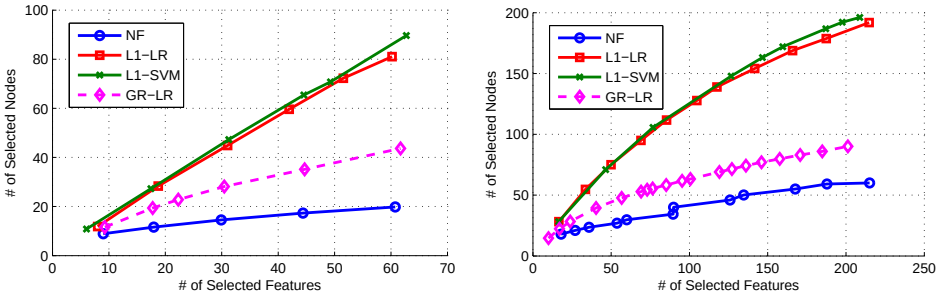


(a) PD (b) ASD

Figure 5.5: Comparison of the sparsity of group features achieved by NF, baseline group feature selection method (GR-LP), baseline feature selection methods (ℓ_1-LR and ℓ_1-SVM), baseline classification methods (S-LR and S-SVM) on two real datasets, PD and ASD

Figure 5.5 depicts testing accuracies of NF and all other baseline methods on both real datasets, PD and ASD, as a function of the number of selected features. For PD dataset in Figure 5.5(a), testing accuracies of all methods increase as the number of selected features increases from 1 to 50. The accuracies of all methods decrease and plateau when the number of selected features is larger than 50 (not shown in the figure), which is one would expect due to overfitting. As the number of samples in PD data is only 46, the number of selected features should not be larger than 50. However, it is important to note that NF outperforms all other baseline methods, providing the highest classification accuracy. For ASD dataset in Figure 5.5(b), testing accuracies of all methods increase as the number of selected features increases. Because the number of samples in ASD data is 360, all methods did not encounter the overfitting issue. Similar to the PD result, NF outperforms all other baseline methods, providing the highest classification accuracy. We also note that the testing accuracy of GR-LR was better than the baseline feature selection methods, ℓ_1-LR and ℓ_1-SVM, and other baseline classification methods, S-LR and S-SVM.

Figure 5.6 illustrates the group sparsity of solutions (i.e., the number of selected feature groups) obtained from NF and other baseline methods with respect to the number of selected features on both real datasets. Similar to synthetic datasets, not only does NF provide the highest group sparsity (i.e., using the least number of feature groups) but it also yield the



(a) PD (b) ASD

Figure 5.6: Comparison of the sparsity of group features achieved by NF, baseline group feature selection method (GR-LP), baseline feature selection methods (ℓ_1-LR and ℓ_1-SVM), baseline classification methods (S-LR and S-SVM) on two real datasets, PD and ASD

most accurate results. We also note that, as expected, GR-LR also produced group sparse solutions but not as sparse as the ones obtained from NF, which may be due to the solution bias induced by $\ell_{2,1}$-norm minimization.

5.3.6 Scientific Interpretation of Node Selection on Real Datasets

One of the goals of feature/group selection is to improve interpretability of the underlying complex data. Taking into account the specific domain knowledge of given data, the selected features can give useful information for understanding data, e.g., informative connectivities in the human brain networks on autism spectrum disorder [49]. In this section, we investigate the features and groups of features selected by each of the methods tested in this chapter. Based on these selections, we examined how they form subnetworks, which may provide valuable insights to a greater understanding of disrupted brain connectivities due to the brain diseases.

Figure 5.7 illustrates subnetworks, in terms of nodes and edges, that were obtained from group feature selection methods and baseline feature selection methods. In the figure, two dimensional coordinates (x, y) for each node were randomly generated with the Uniform distribution $x, y \sim \mathcal{U}(0, 150)$. Coordinates of 'Ground-truth' nodes in N^T were $\bar{x}, \bar{y} \sim \mathcal{U}(0, 15)$. Let the size of network for each method be (# of nodes, # of edges). Then,

NF=(21,104), GR-LR=(27,112), ℓ_1-SVM=(71,105),ℓ_1-LR=(69,109). From the figure, it can be observed that relevant features and groups are located at the south west corner of the network. Such an observation is much more prominent in NF results. If SN-150 was a transportation or sensory network, then one can easily identify the root cause of congestions or malfunctioning sensors from the resulting sub-networks obtained from NF.

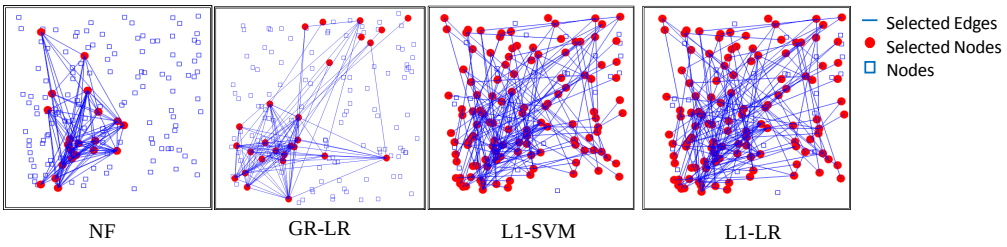


Figure 5.7: Illustration of the selected sub-networks on SN-150 with the best performing sub-networks for NF, baseline group feature selection method (GR-LP), baseline feature selection methods (ℓ_1-LR and ℓ_1-SVM)

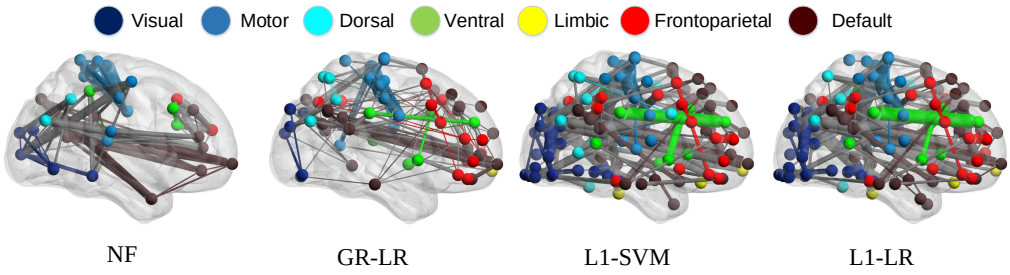


Figure 5.8: Illustration of the selected subnetworks obtained from NF, baseline group feature selection method (GR-LP), baseline feature selection methods (ℓ_1-LR and ℓ_1-SVM) on ASD dataset

Figure 5.8 illustrates three dimensional coordinates (x, y, z) of selected edges and nodes, which are derived from [154]. All nodes are categorized by 7 functional networks defined in [210]. Each node is colored differently in terms of its functional network group. The size of selected subnetworks are (94 edges, 34 nodes) for NF, (101 edges, 63 nodes) for GR-LR, (96 edges, 118 nodes) for ℓ_1-SVM, (94 edges, 115 nodes) for ℓ_1-LR. It is worth noting that, in

the literature, it has not been shown that any one of these functional networks is a predictor of the disease onset. Based on the selected subnetworks, it is clear that disruptions in brain network are not localized in any of functional networks. Nevertheless, the selected feature groups (nodes) and features (edges) by NF provide a much more insightful interpretation of brain network disruptions. Sparser feature groups (nodes) and features (edges) also help researcher identify a few key localized regions in the brain that may be disrupted in ASD subjects. For example, we can observe that there exist strong connections between occipital and frontal areas through the default mode networks (brown colored nodes). These findings, which are discovered only by the proposed NF technique, are supported by previous ASD studies [49, 12].

5.4 Summary

In this chapter, we modified the group lasso problem by substituting the $\ell_{2,1}$ regularization term with the $\ell_{2,0}$ norm to avoid the undesirable properties of the $\ell_{2,1}$ norm while using the mathematical programming structure of SVM. Due to the non-convexity of the original problem, we applied a convex relaxation and reformulated the model as a convex QCLP, which includes an exponential number of constraints. To deal with inefficiency of constraint size, we developed a new iterative algorithm that can guarantee to converge with an optimal solution. This iterative algorithm computes an intermediate SVM solution with a redirected number of constraints and update the subset in an iterative fashion based on the solution of intermediate SVM. Our technique for the group feature selection based on $\ell_{2,0}$ norm can overcome the biased solution of $\ell_{2,0}$-norm on the high dimensional data and has shown a successful extension of the group selection application through the node selection in the networked data. Our technique can also be extended to ultra-high dimensional data. Computational results performed on synthetic and real datasets showed promising results, which can be summarized as follows:

1. Experiments results on synthetic datasets showed that if relevant features contain a group structure, the proposed technique can significantly outperform the standard feature selection methods. Also, we observed superior performance of our technique compared to group lasso in terms of testing accuracy (predictive power) and the group

sparsity of the solution (number of nodes selected).

2. Experimental results on both Parkinson's Disease (PD) and Autism Spectrum Disorder (ASD) datasets showed that our proposed technique provides a group sparse solution with the highest testing accuracy.
3. The illustration of the results of synthetic and real datasets confirmed that our technique provides more interpretable results (i.e., a few number of nodes with more meaningful structure) compared to other methods.

Chapter 6

SUMMARY AND FINAL REMARKS

In this thesis, I proposed optimization frameworks, including not only the standard network optimization problem, but also machine learning problems to find critical components which contain insights for analysis of large-scale complex networks.

For this purpose, I formulated mathematical programming problems which had many applications, especially to human brain connectivity networks, and were of considerable theoretical interest because they generalized several well-known network optimization problems.

My thesis was motivated by the need not only to build new models for identifying critical components, but also for developing efficient solution approaches to handle the large-scale data which have so far jeopardized the general performance of optimization frameworks. As a consequence, I devoted my attention to understanding the structural properties of complex networks and exploiting these properties to develop MIP/MINLP models with computationally efficient solution approaches.

In particular, I designed:

1. MIP models for the k-cardinality tree problem (KCTP). These formulations find a minimum weighted k-sized subtree in the network. Among them, I selected a computationally efficient model by means of theoretical and empirical studies.
2. A MIP formulation for node selection in networked data classification. This is solved by a branch-and cut-algorithm based on the Bender's decomposition in order to improve computational efficiency.
3. A MINLP formulation for a group selection problem. This model generalized the group variable selection as proposed by [213]. This can be applied to the node selection problem in networked data classification.

In Chapter 3, seven different compact formulations for KCTP were presented. KCTP is

one of the network optimization problems that involves finding a critical component in the network which keeps connectivity with use of the least edges. Due to the NP-hardness of the original problem, various solution approaches have been proposed to resolve computational intractability. Recently, a branch-and-cut algorithm has been proposed by [174] and has been shown to outperform all other approaches, especially for grid-shaped networks which are generally sparse. However, this algorithm was hard to implement for practical usage since it required several complicated tuned algorithms. Furthermore, its performance was not effective when working with dense networks since the algorithm generated a huge size of constraints to maintain connectivity on such networks. This motivated us to develop new models which work well for dense, complex networks, as well as being easy to implement. Through theoretical studies and computational experiments on the benchmark data sets and randomly generated dense data sets, the proposed MTZ-based model outperformed all other models on the dense networks. We also performed a comparison between the selected model and the branch-and-cut model proposed by [174] on large random networks. Our formulation not only performed better overall, but also provided more robust results with smaller variations in CPU time.

As a next step, I wish to expand my research toward finding the critical components in a group of labeled networks. Specifically, when working on the classification of networks, I will assume that there exists a core subnetwork which is composed of informative features. Detecting this component will be helpful for describing group level distinctions in data sets, while discarding non-informative subnetworks. First of all, I employed KCT as a feature extraction tool for the network analysis. In specific, I obtained the averaged connectivity of the extracted KCT for each network instance and did a univariate statistical analysis based on the extracted values by means of KCT. This approach is also in the line with filtering feature selection method.

Using the KCT is one of the application of feature selection in the network classification. To produce generalized models, I planed to develop modified optimization frameworks based on the feature selection problem which has been used for general data classification. With this objective in mind, the specific goal of the frameworks I proposed was to select a k-sized subnetwork which contained the most relevant features and would thus contribute to reduc-

ing the misclassification rate. Simultaneously, to select a k-sized subnetwork was equivalent to discarding noisy information which deteriorates general learning performance, as well as diminishing the understanding of the underlying structure of large-scale networked data sets. Through the proposed models, the aims of this proposed framework were simplification of the model to make it easier to interpret by future users, a reduction in computation time, and the enhancement of generalization. Considering the general context of the feature selection, this framework can be regarded as a wrapper method which select the best subset of features in the whole possible combinations.

In terms of methodological approaches, I first formulated MIP to identify the k-sized subnetwork, which minimizes the misclassification rate of the induced k-sized subnetwork. Since the formulated MIP was NP-hard in the nature of its combinatorics, the Bender's decomposition approach was used to decompose the original problem into master and slave problems. Instead of a generic algorithm for solving the Bender's decomposition problem, a branch-and-cut algorithm was developed to find a global optimal solution by using decomposed problems. As a result, the proposed algorithm outperformed a branch-and-bound algorithm which has been a general solution procedure for MIP in terms of computation speed and predictive power when dealing with synthetic and real data sets.

Regardless of the successful improvement of computational efficiency through the research discussed in Chapter 4, it was still necessary to improve computational efficiency for practical usage in terms of large-scale networked data sets. Therefore, I proposed a new mathematical formulation for node selection based on the SVM as it was computationally efficient and well-suited to problems such as high dimensional data classification. The resulting optimization problem was still NP-hard due to the existence of integer variables and a ℓ_0-norm, so I relaxed the original problem into convex QCLP and solved the relaxed QCLP by means of a cutting plane scheme. Unlike the MIP presented in Chapter 4, this new proposed technique could not find an exact k-sized subnetwork optimally, but it did find an optimal combination of several k-sized subnetworks. Compared with the state of the art feature selection algorithms, the developed technique achieved competitive or even better performance in terms of generalized predictions.

Also, it provided a tractable computational time, in spite of containing a ℓ_0-norm, which

made the optimization problem NP-hard. Thus, it achieved computational efficiency and great predictive performance simultaneously, whereas existing ℓ_1-norm based methods suffer from "biased" results.

I view this thesis as synthesizing concepts from two important areas: mathematical programming (i.e., optimization problems) and machine learning (i.e., data classification). Specifically, I introduce the network optimization framework as a feature selection scheme for the data classification problem which is a core part of machine learning. The collaboration of concepts from both these areas provided crucial direction to the development of our approach. From the viewpoint of machine learning, I focused on how to construct an efficient model to select informative features concerning the classification of structured data. Simultaneously, from the viewpoint of optimization, I focused on how to develop solution approaches to solve the constructed machine learning problem in a tractable computation time. These collaborations are the first of such approaches involving networked data classification. With the development of technologies of data collection and processing, the need for networked data classification will increase drastically.

The future work of this study will focus on the following directions:

In Chapter 3, the proposed methods for KCTP showed how we tackled the problem of dense networks with a large number of constraints and variables in terms of size of their edges. From the literature on KCTP, various heuristic methods, such as dynamic algorithm [27], were developed. In future, I will use this solution as the primary heuristic aim of the MIP. I will set the solution of the heuristic algorithm as the initial solution of the branch-and-bound procedure for the MIP, thus reducing the solution search space which is extremely large due to the huge problem size and I thus will obtain a shorter computational time.

In Chapter 4, I proposed a MIP to select the most informative k-sized subnetwork of large-scale complex networks. The selected nodes and their corresponding edges were regarded as the most relevant subnetwork to describe group level distinctions in given networks. The proposed MIP contained inherited drawbacks caused by the big-M method which has been previously used to restrict the upper bound of feature weights in the resulting classifier. This big-M method has been known to deteriorate the computational efficiency of the MIP in general. A recent study of the subset selection problem by Bertsimas et al. presented

an approximation method to find the appropriate values of the bound [21]. Motivated by this approximation method, I will estimate the bound of feature weights in the proposed MIP and this will be helpful for providing a reliable solution and significantly reducing the computation.

In Chapter 5, I will focus on developing an exact solution approach extended from the current heuristic solution approach. As the computational time will be still tractable for practical usage, the new approach will show better predictive performance with providing exact k sized component at a time.

Concerning the application of this formulation to complex networks, it is worth mentioning that the hierarchical structure of networks plays a new role in the context of complex network learning. For example, I depicted the selected critical component in Figure 5.8 where each node was colored in terms of their predetermined functionality, as defined by Yeo et al. [210]. Based on the predetermined characteristic of each node, the next step is to consider the sparsity of such functionality, e.g., how to select critical components having the same functionality. Indeed, considering this sparsity of the components will be very important. There is no clear evidence that every edge of the critical component is informative for classification. Therefore, in this case, it will be necessary to decompose the current cardinality constraint $G_s \leq k$ into $V(G_s) \leq k_1$ and $E(G_s) \leq k_2$, respectively, where $V(G_s)$ represents the number of nodes in G_s and $E(G_s)$ represents the number of edges in G_s.

BIBLIOGRAPHY

[1] Mouhamed Abdulla. *On the fundamentals of stochastic spatial modeling and analysis of wireless networks and its impact to channel losses*. PhD thesis, Concordia University, 2012.

[2] Sophie Achard and Ed Bullmore. Efficiency and cost of economical brain functional networks. *PLoS computational biology*, 3(2):e17, 2007.

[3] Sophie Achard, Raymond Salvador, Brandon Whitcher, John Suckling, and ED Bullmore. A resilient, low-frequency, small-world human brain functional network with highly connected association cortical hubs. *The Journal of Neuroscience*, 26(1):63–72, 2006.

[4] Aaron B Adcock, Blair D Sullivan, and Michael W Mahoney. Tree-like structure in large social and information networks. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1–10. IEEE, 2013.

[5] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows. Technical report, DTIC Document, 1988.

[6] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *Communications magazine, IEEE*, 40(8):102–114, 2002.

[7] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[8] Bahram Alidaee, Fred Glover, Gary Kochenberger, and Haibo Wang. Solving the maximum edge weight clique problem via unconstrained quadratic programming. *European Journal of Operational Research*, 181(2):592–597, 2007.

[9] Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1):237–260, 1998.

[10] David Applegate, Robert Bixby, William Cook, and Vasek Chvátal. *On the solution of traveling salesman problems*. Rheinische Friedrich-Wilhelms-Universität Bonn, 1998.

[11] S. Arora and G. Karakostas. A (2+ϵ)-approximation algorithm for the k-mst problem. *In Proc. of the eleventh annual ACM-SIAM symposium on Discrete algorithms (SODA'00)*, pages 754–759, 2000.

[12] Michal Assaf, Kanchana Jagannathan, Vince D Calhoun, Laura Miller, Michael C Stevens, Robert Sahl, Jacqueline G O'Boyle, Robert T Schultz, and Godfrey D Pearlson. Abnormal functional connectivity of default mode sub-networks in autism spectrum disorder patients. *Neuroimage*, 53(1):247–256, 2010.

[13] Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. Convex optimization with sparsity-inducing norms. *Optimization for Machine Learning*, pages 19–53, 2011.

[14] Francis Bach and Guillaume Obozinski. Sparse methods for machine learning theory and algorithms. *ECML/PKDD Tutorial*, 2010.

[15] Lihui Bai and Paul A Rubin. Combinatorial benders cuts for the minimum tollbooth problem. *Operations research*, 57(6):1510–1522, 2009.

[16] Balabhaskar Balasundaram, Sergiy Butenko, and Illya V Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59(1):133–142, 2011.

[17] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[18] R Baumgartner, C Windischberger, and E Moser. Quantification in functional mag-

netic resonance imaging: fuzzy clustering vs. correlation analysis. *Magnetic Resonance Imaging*, 16(2):115–125, 1998.

[19] Jacques F Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.

[20] Dimitri P Bertsekas and Athena Scientific. Network optimization: Continuous and discrete models. *Interfaces-Providence-Institute of Management Sciences*, 28:73–75, 1998.

[21] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. arXiv/1507.03133, 2015.

[22] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $o(n^{\frac{1}{4}})$ approximation for densest k-subgraph. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 201–210. ACM, 2010.

[23] Norman Biggs, E Keith Lloyd, and Robin J Wilson. *Graph Theory 1736-1936*. Oxford University Press, 1976.

[24] Maria J. Blesa, M. Josep Blesa, Fatos Xhafa, and Jordi Girona. A c++ implementation of tabu search for k-cardinality tree problem based on generic programming and component reuse. In *GCSE Young Researchers Workshop 2000 (Part of the Second International Symposium on Generative and Component-based Software Engineering) October 9-12*, pages 648–652. Net.ObjectDaysForum, 2000.

[25] M.J. Blesa, P. Moscato, and F. Xhafa. A memetic algorithm for the minimum weighted k-cardinality tree subgraph problem. In *In Proceedings of the Metaheuristics International Conference MIC'2001*, volume 1, pages 85–90, Porto, Portugal, 2001.

[26] A. Blum, R. Ravi, and S. Vempala. A constant factor approximation algorithm for the k-mst problem. *In ACM Symposium on Theory of Computing*, pages 442–448, 1996.

[27] C. Blum. Revisiting dynamic programming for finding optimal subtrees in trees. *European Journal of Operational Research*, 177(1):102–115, 2007.

[28] C. Blum and M. J. Blesa. New metaheuristic approaches for the edge-weighted k-cardinality tree problem. *Computers & Operations Research*, 32:1355–1377, 2005.

[29] C. Blum and M. Ehrgott. Local search algorithms for the k-cardinality tree problem. *Discrete Applied Mathematics*, 128(2-3):511–540, 2003.

[30] Christian Blum and Maria Blesa. Combining ant colony optimization with dynamic programming for solving the k-cardinality tree problem. In *Computational Intelligence and Bioinspired Systems*, volume 3512 of *Lecture Notes in Computer Science*, pages 25–33. Springer Berlin Heidelberg, 2005.

[31] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(45):175 – 308, 2006.

[32] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.

[33] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[34] Steffen Borgwardt and Felix Schmiedl. Threshold-based preprocessing for approximating the weighted dense k-subgraph problem. *European Journal of Operational Research*, 234(3):631–640, 2014.

[35] Stefan Bornholdt, Heinz Georg Schuster, and John Wiley. *Handbook of graphs and networks*, volume 2. Wiley Online Library, 2003.

[36] Quentin Botton, Bernard Fortz, Luis Gouveia, and Michael Poss. Benders decomposition for the hop-constrained survivable network design problem. *INFORMS journal on computing*, 25(1):13–26, 2013.

[37] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[38] Paul S Bradley and Olvi L Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, volume 98, pages 82–90, 1998.

[39] J. Brimberg, D. Urošević, and N. Mladenović. Variable neighborhood search for the vertex weighted k-cardinality tree problem. *European Journal of Operational Research*, 171(1):74–84, 2006.

[40] Jack Brimberg, D Urošević, and N Mladenović. Variable neighborhood search for the vertex weighted k-cardinality tree problem. *European Journal of Operational Research*, 171(1):74–84, 2006.

[41] Samantha J Broyd, Charmaine Demanuele, Stefan Debener, Suzannah K Helps, Christopher J James, and Edmund JS Sonuga-Barke. Default-mode brain dysfunction in mental disorders: a systematic review. *Neuroscience & biobehavioral reviews*, 33(3):279–296, 2009.

[42] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186–198, 2009.

[43] Horst Bunke, Pasquale Foggia, C Guidobaldi, and Mario Vento. Graph clustering using the weighted minimum common supergraph. In *Graph based representations in pattern recognition*, pages 235–246. Springer, 2003.

[44] Antoni B Chan, Nuno Vasconcelos, and Gert RG Lanckriet. Direct convex relaxations of sparse svm. In *Proceedings of the 24th international conference on Machine learning*, pages 145–153. ACM, 2007.

[45] Dennis Chan, Nick C Fox, Rachael I Scahill, William R Crum, Jennifer L Whitwell, Guy Leschziner, Alex M Rossor, John M Stevens, Lisa Cipolotti, and Martin N Rossor. Patterns of temporal lobe atrophy in semantic dementia and alzheimer's disease. *Annals of neurology*, 49(4):433–442, 2001.

[46] Chieng-Yi Chang. Dynamic programming as applied to feature subset selection in a

pattern recognition system. In *Proceedings of the ACM annual conference-Volume 1*, pages 94–103. ACM, 1972.

[47] Wanpracha Chaovalitwongse, Panos M Pardalos, and Oleg A Prokopyev. A new linearization technique for multi-quadratic 0–1 programming problems. *Operations Research Letters*, 32(6):517–522, 2004.

[48] Sandeep Chaplot, LM Patnaik, and NR Jagannathan. Classification of magnetic resonance brain images using wavelets as input to support vector machine and neural network. *Biomedical Signal Processing and Control*, 1(1):86–92, 2006.

[49] Colleen P Chen, Christopher L Keown, Afrooz Jahedi, Aarti Nair, Mark E Pflieger, Barbara A Bailey, and Ralph-Axel Müller. Diagnostic classification of intrinsic functional connectivity highlights somatosensory, default mode, and visual regions in autism. *NeuroImage: Clinical*, 8:238–245, 2015.

[50] Jianhui Chen and Jieping Ye. Training svm with indefinite kernels. In *Proceedings of the 25th international conference on Machine learning*, pages 136–143. ACM, 2008.

[51] Veronika Cheplygina, David MJ Tax, Marco Loog, and Aasa Feragen. Network-guided group feature selection for classification of autism spectrum disorder. In *Machine Learning in Medical Imaging*, pages 190–197. Springer, 2014.

[52] Markus Chimani, Maria Kandyba, Ivana Ljubić, and Petra Mutzel. Obtaining optimal k-cardinality trees fast. *Journal of Experimental Algorithmics (JEA)*, 14:5:2.5–5:2.23, 2009.

[53] Chee-Yee Chong and Srikanta P Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[54] Leigh Christopher, Connie Marras, Sarah Duff-Canning, Yuko Koshimori, Robert Chen, Isabelle Boileau, Barbara Segura, Oury Monchi, Anthony E Lang, Pablo Rusjan, et al. Combined insular and striatal dopamine dysfunction are associated with executive deficits in parkinsons disease with mild cognitive impairment. *Brain*, page awt337, 2013.

[55] Derek G Corneil and Yehoshua Perl. Clustering and domination in perfect graphs. *Discrete Applied Mathematics*, 9(1):27–39, 1984.

[56] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[57] R Cameron Craddock, Paul E Holtzheimer, Xiaoping P Hu, and Helen S Mayberg. Disease state prediction from resting state functional connectivity. *Magnetic resonance in Medicine*, 62(6):1619–1628, 2009.

[58] JeffreyL Cummings. Dementia: the failing brain. *The Lancet*, 345(8963):1481–1484, 1995.

[59] JS Damoiseaux, SARB Rombouts, F Barkhof, P Scheltens, CJ Stam, Stephen M Smith, and CF Beckmann. Consistent resting-state networks across healthy subjects. *Proceedings of the national academy of sciences*, 103(37):13848–13853, 2006.

[60] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, pages 393–410, 1954.

[61] Mukund Deshpande, Michihiro Kuramochi, Nikil Wale, and George Karypis. Frequent substructure-based approaches for classifying chemical compounds. *Knowledge and Data Engineering, IEEE Transactions on*, 17(8):1036–1050, 2005.

[62] M. Desrochers and G. Laporte. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27–36, 1991.

[63] Kim TE Olde Dubbelink, Diederick Stoffers, Jan Berend Deijen, Jos WR Twisk, Cornelis J Stam, Arjan Hillebrand, and Henk W Berendse. Resting-state functional connectivity as a marker of disease progression in parkinson's disease: A longitudinal meg study. *NeuroImage: Clinical*, 2:612–619, 2013.

[64] Sarah Dunn and Sean M Wilkinson. Identifying critical components in infrastructure networks using network topology. *Journal of Infrastructure Systems*, 19(2):157–165, 2012.

[65] Victor M Eguiluz, Dante R Chialvo, Guillermo A Cecchi, Marwan Baliki, and A Vania Apkarian. Scale-free brain functional networks. *Physical review letters*, 94(1):018102, 2005.

[66] M. Ehrgott and J. Freitag. K tree/k subgraph: a program package for minimal weighted k-cardinality. *European Journal of Operational Research*, 1(93):214–225, 1996.

[67] M. Ehrgott, J. Freitag, H.W. Hamacher, and F. Maffioli. Heuristics for the k-cardinality tree and subgraph. *Asia-Pacific Journal of Operational Research*, 14(1):87–114, 1997.

[68] Damien A Fair, Alexander L Cohen, Jonathan D Power, Nico UF Dosenbach, Jessica A Church, Francis M Miezin, Bradley L Schlaggar, and Steven E Petersen. Functional brain networks develop from a local to distributed organization. *PLoS computational biology*, 5(5):e1000381, 2009.

[69] Jianqing Fan, Fang Han, and Han Liu. Challenges of big data analysis. *National science review*, 1(2):293–314, 2014.

[70] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[71] Shu-Cherng Fang, Chih-Jen Lin, and Soon-Yi Wu. Solving quadratic semi-infinite programming problems by using relaxed cutting-plane scheme. *Journal of computational and applied mathematics*, 129(1):89–104, 2001.

[72] Hongliang Fei and Jun Huan. Structure feature selection for graph classification. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 991–1000. ACM, 2008.

[73] Massimo Filippi, Federica Agosta, Elisa Scola, Elisa Canu, Giuseppe Magnani, Alessandra Marcone, Paola Valsasina, Francesca Caso, Massimiliano Copetti, Giancarlo Comi, et al. Functional network connectivity in the behavioral variant of frontotemporal dementia. *Cortex*, 49(9):2389–2401, 2013.

[74] M. Fischetti, W. Hamacher, K. Jornsten, and F. Maffioli. Weighted k-cardinality trees: Complexity and polyhedral structure. *Networks*, 24:11–21, 1994.

[75] US-Canada Power System Outage Task Force, Spencer Abraham, Herb Dhaliwal, R John Efford, Linda J Keen, Anne McLellan, John Manley, Kenneth Vollman, Nils J Diaz, Tom Ridge, et al. *Final report on the August 14, 2003 blackout in the United states and Canada: causes and recommendations*. US-Canada Power System Outage Task Force, 2004.

[76] Ned Freed and Fred Glover. Linear programming and statistical discrimination the lp side*. *Decision Sciences*, 13(1):172–175, 1982.

[77] Pascal Fries. A mechanism for cognitive dynamics: neuronal communication through neuronal coherence. *Trends in cognitive sciences*, 9(10):474–480, 2005.

[78] Karl J Friston. Functional and effective connectivity in neuroimaging: a synthesis. *Human brain mapping*, 2(1-2):56–78, 1994.

[79] Glenn Fung and Olvi L Mangasarian. Data selection for support vector machine classifiers. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 64–70. ACM, 2000.

[80] N. Garg. A 3-approximation for the minimum tree spanning k vertices. *In Proc. of the 37th Annual Symposium on Foundations of Computer Science(FOCS'96)*, pages 302–309, 1996.

[81] N. Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. *In Proc. of the thirty-seventh annual ACM symposium on Theory of computing (STOC'05)*, pages 396–402, 2005.

[82] Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*, pages 129–143. Springer, 2003.

[83] Willy Gochet, Antonie Stam, V Srinivasan, and Shaoxiang Chen. Multigroup discriminant analysis using linear programming. *Operations Research*, 45(2):213–225, 1997.

[84] Michel X Goemans and David P Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[85] Eitan Greenshtein et al. Best subset selection, persistence in high-dimensional statistical learning and optimization under l1 constraint. *The Annals of Statistics*, 34(5):2367–2386, 2006.

[86] Michael Greicius. Resting-state functional connectivity in neuropsychiatric disorders. *Current opinion in neurology*, 21(4):424–430, 2008.

[87] Michael D Greicius, Ben Krasnow, Allan L Reiss, and Vinod Menon. Functional connectivity in the resting brain: a network analysis of the default mode hypothesis. *Proceedings of the National Academy of Sciences*, 100(1):253–258, 2003.

[88] Wei Guan, Alex Gray, and Sven Leyffer. Mixed-integer support vector machine. In *NIPS Workshop on Optimization for Machine Learning*, 2009.

[89] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[90] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[91] Jiuqi Han, Zhengya Sun, and Hongwei Hao. l 0-norm based structural sparse least square regression for feature selection. *Pattern Recognition*, 48(12):3927–3940, 2015.

[92] David J Hand. Discrimination and classification. *Wiley Series in Probability and Mathematical Statistics, Chichester: Wiley, 1981*, 1, 1981.

[93] Rainer Hettich and Kenneth O Kortanek. Semi-infinite programming: theory, methods, and applications. *SIAM review*, 35(3):380–429, 1993.

[94] Adel Hlaoui and Shengrui Wang. Median graph computation for graph clustering. *Soft Computing*, 10(1):47–53, 2006.

[95] Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 158–167. ACM, 2004.

[96] Jun Huan, Wei Wang, and Jan Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 549–552. IEEE, 2003.

[97] Mark D Humphries, Kevin Gurney, and Tony J Prescott. The brainstem reticular formation is a small-world, not scale-free, network. *Proceedings of the Royal Society B: Biological Sciences*, 273(1585):503–511, 2006.

[98] Frank K Hwang, Dana S Richards, and Pawel Winter. *The Steiner tree problem*. Elsevier, 1992.

[99] Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pages 433–440. ACM, 2009.

[100] Wasifa Jamal, Saptarshi Das, Ioana-Anastasia Oprescu, Koushik Maharatna, Fabio Apicella, and Federico Sicca. Classification of autism spectrum disorder using supervised learning of brain connectivity measures extracted from synchrostates. *Journal of neural engineering*, 11(4):046019, 2014.

[101] Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. Structured variable se-

lection with sparsity-inducing norms. *The Journal of Machine Learning Research*, 12:2777–2824, 2011.

[102] David Jensen and Jennifer Neville. *Data mining in social networks*. na, 2003.

[103] Chuntao Jiang, Frans Coenen, Robert Sanderson, and Michele Zito. Text classification using graph mining-based feature extraction. *Knowledge-Based Systems*, 23(4):302–308, 2010.

[104] Biao Jie, Daoqiang Zhang, Chong-Yaw Wee, and Dinggang Shen. Structural feature selection for connectivity network-based mci diagnosis. In *Multimodal Brain Image Analysis*, pages 175–184. Springer, 2012.

[105] Erich A Joachimsthaler and Antonie Stam. Mathematical programming approaches for the classification problem in two-group discriminant analysis. *Multivariate Behavioral Research*, 25(4):427–454, 1990.

[106] Kurt Jörnsten and Arne Løkketangen. Tabu search for weighted k-cardinality trees. *Asia Pacific Journal of Operational Research*, 14:9–26, 1997.

[107] DN Kennedy, N Lange, N Makris, J Bates, J Meyer, and VS Caviness. Gyri of the human neocortex: an mri-based analysis of volume and variance. *Cerebral Cortex*, 8(4):372–384, 1998.

[108] Seung-Jean Kim and Stephen Boyd. A minimax theorem with applications to machine learning, signal processing, and finance. *SIAM Journal on Optimization*, 19(3):1344–1367, 2008.

[109] Seyoung Kim, Eric P Xing, et al. Tree-guided group lasso for multi-response regression with structured sparsity, with an application to eqtl mapping. *The Annals of Applied Statistics*, 6(3):1095–1117, 2012.

[110] Christof Koch and Gilles Laurent. Complexity and the nervous system. *Science*, 284(5411):96–98, 1999.

[111] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997.

[112] Xiangnan Kong and Philip S Yu. Brain network analysis: a data mining perspective. *ACM SIGKDD Explorations Newsletter*, 15(2):30–38, 2014.

[113] Guy Kortsarz and David Peleg. On choosing a dense subgraph. In *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pages 692–701. IEEE, 1993.

[114] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[115] Hugo Kubinyi. Drug research: myths, hype and reality. *Nature Reviews Drug Discovery*, 2(8):665–667, 2003.

[116] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

[117] André Langevin, François Soumis, and Jacques Desrosiers. Classification of travelling salesman problem formulations. *Operations Research Letters*, 9(2):127–132, 1990.

[118] Eva K Lee and Tsung-Lin Wu. Classification and disease prediction via mathematical programming. In *Handbook of Optimization in Medicine*, pages 1–50. Springer, 2009.

[119] Seunghak Lee and Eric P Xing. Screening rules for overlapping group lasso. *arXiv preprint arXiv:1410.6880*, 2014.

[120] Geng Li, Murat Semerci, Bülent Yener, and Mohammed J Zaki. Effective graph classification based on topological and label attributes. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(4):265–283, 2012.

[121] Kaiming Li, Lei Guo, Jingxin Nie, Gang Li, and Tianming Liu. Review of methods for functional brain connectivity detection using fmri. *Computerized Medical Imaging and Graphics*, 33(2):131–139, 2009.

[122] Yu-Feng Li, James T Kwok, Ivor W Tsang, and Zhi-Hua Zhou. A convex method for locating regions of interest with multi-instance learning. In *Machine learning and knowledge discovery in databases*, pages 15–30. Springer, 2009.

[123] Yu-Feng Li, Ivor W Tsang, James T Kwok, and Zhi-Hua Zhou. Tighter and convex maximum margin clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 344–351, 2009.

[124] Dongyu Lin, Emily Pitler, Dean P Foster, and Lyle H Ungar. In defense of ℓ_0. In *Workshop on Feature Selection,(ICML 2008)*, 2008.

[125] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009.

[126] Yong Liu, Meng Liang, Yuan Zhou, Yong He, Yihui Hao, Ming Song, Chunshui Yu, Haihong Liu, Zhening Liu, and Tianzi Jiang. Disrupted small-world networks in schizophrenia. *Brain*, 131(4):945–961, 2008.

[127] Zhaosong Lu and Yong Zhang. Penalty decomposition methods for ℓ_0-norm minimization. *Technical Reports*, 2010.

[128] Bin Luo, Richard C Wilson, and Edwin R Hancock. Spectral feature vectors for graph clustering. In *Structural, syntactic, and statistical pattern recognition*, pages 83–93. Springer, 2002.

[129] Liangsuo Ma, Binquan Wang, Xiying Chen, and Jinhu Xiong. Detecting functional connectivity in the resting brain: a comparison between ica and cca. *Magnetic resonance imaging*, 25(1):47–56, 2007.

[130] Elder Magalhães Macambira and Cid Carvalho De Souza. The edge-weighted clique problem: valid inequalities, facets and polyhedral computations. *European Journal of Operational Research*, 123(2):346–371, 2000.

[131] Nikos Makris, James W Meyer, Julianna F Bates, Edward H Yeterian, David N Kennedy, and Verne S Caviness. Mri-based topographic parcellation of human cerebral

white matter and nuclei: Ii. rationale and applications with systematics of cerebral connectivity. *Neuroimage*, 9(1):18–45, 1999.

[132] James McAuley, Ji Ming, Darryl Stewart, and Philip Hanna. Subband correlation and robust speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 13(5):956–964, 2005.

[133] John Mendoza and Anne Foundas. *Clinical neuroanatomy: A neurobehavioral approach*. Springer Science & Business Media, 2007.

[134] Vinod Menon and Lucina Q Uddin. Saliency, switching, attention and control: a network model of insula function. *Brain Structure and Function*, 214(5-6):655–667, 2010.

[135] David Meunier, Renaud Lambiotte, Alex Fornito, Karen D Ersche, and Edward T Bullmore. Hierarchical modularity in human brain functional networks. *Hierarchy and dynamics in neural networks*, 1:2, 2009.

[136] James W Meyer, Nikos Makris, Julianna F Bates, Verne S Caviness, and David N Kennedy. Mri-based topographic parcellation of human cerebral white matter: I. technical foundations. *Neuroimage*, 9(1):1–17, 1999.

[137] C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulations and travelling salesman problem. *J. Assoc. Computation*, pages 326–329, 1960.

[138] Nenad Mladenović and Dragan Urošević. Variable neighborhood search for the k-cardinality tree. In *Metaheuristics*, pages 481–500. Kluwer Academic Publishers, 2004.

[139] Janaina Mourão-Miranda, Arun LW Bokde, Christine Born, Harald Hampel, and Martin Stetter. Classifying brain states and determining the discriminating activation patterns: support vector machine on functional mri data. *NeuroImage*, 28(4):980–995, 2005.

[140] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

[141] Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

[142] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[143] CSA Nextmedia. Social networks overview: Current trends and research challenges. *European Commission Information Society and Media*, 2010.

[144] Andrew Y Ng. Feature selection, ℓ_1 vs. ℓ_2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[145] Lionel M Ni, Yunhao Liu, and Yanmin Zhu. China's national research project on wireless sensor networks. *Wireless Communications, IEEE*, 14(6):78–83, 2007.

[146] Manfred Padberg and Ting-Yi Sung. An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming*, 52(1-3):315–357, 1991.

[147] Panos M Pardalos and H Edwin Romeijn. *Handbook of optimization in medicine*, volume 5. Springer Science & Business Media, 2009.

[148] EY Pee and Johannes O Royset. On solving large-scale finite minimax problems using exponential smoothing. *Journal of Optimization Theory and Applications*, 148(2):390–421, 2011.

[149] Ronald C Petersen. Clinical practice. mild cognitive impairment. *The New England journal of medicine*, 364(23):2227, 2011.

[150] Jeffrey R Petrella. Use of graph theory to evaluate brain networks: a clinical tool for a small world? *Radiology-Radiological Society of North America*, 259(2):317, 2011.

[151] JR Petrella, FC Sheldon, SE Prince, VD Calhoun, and PM Doraiswamy. Default mode network connectivity in stable vs progressive mild cognitive impairment. *Neurology*, 76(6):511–517, 2011.

[152] Michela Pievani, Willem de Haan, Tao Wu, William W Seeley, and Giovanni B Frisoni. Functional network disruption in the degenerative dementias. *The Lancet Neurology*, 10(9):829–843, 2011.

[153] Grigory Pivovarov and Sergei Trunov. Clustering and classification in text collections using graph modularity. *arXiv preprint arXiv:1105.5789*, 2011.

[154] Jonathan D Power, Alexander L Cohen, Steven M Nelson, Gagan S Wig, Kelly Anne Barnes, Jessica A Church, Alecia C Vogel, Timothy O Laumann, Fran M Miezin, Bradley L Schlaggar, et al. Functional network organization of the human brain. *Neuron*, 72(4):665–678, 2011.

[155] True Price, Chong-Yaw Wee, Wei Gao, and Dinggang Shen. Multiple-network classification of childhood autism using functional connectivity dynamics. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014*, pages 177–184. Springer, 2014.

[156] R. C. Prim. On the shortest spanning subtree of a graph and the traveling salesman problem. *The Bell System technical journal*, 36(6):1398–1401, 1957.

[157] Frederico Quintão, Alexandre Salles da Cunha, Geraldo R. Mateus, and Abilio Lucena. The k-cardinality tree problem: Reformulations and lagrangian relaxation. *Discrete Applied Mathematics*, 158(12):1305 – 1314, 2010.

[158] Alain Rakotomamonjy. Variable selection using svm based criteria. *The Journal of Machine Learning Research*, 3:1357–1370, 2003.

[159] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. More efficiency in multiple kernel learning. In *Proceedings of the 24th international conference on Machine learning*, pages 775–782. ACM, 2007.

[160] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

[161] Liva Ralaivola, Sanjay J Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093–1110, 2005.

[162] S Ramón y Cajal. Histology of the nervous system of man and vertebrates. *Oxford Univ. Press, New York*, 1995.

[163] Nikhil Rao, Christopher Cox, Rob Nowak, and Timothy T Rogers. Sparse overlapping sets lasso for multitask learning and its application to fmri analysis. In *Advances in neural information processing systems*, pages 2202–2210, 2013.

[164] Saif Ur Rehman, Asmat Ullah Khan, and Simon Fong. Graph mining: A survey of graph mining techniques. In *Digital Information Management (ICDIM), 2012 Seventh International Conference on*, pages 88–92. IEEE, 2012.

[165] Baxter P Rogers, Victoria L Morgan, Allen T Newton, and John C Gore. Assessing functional connectivity in the human brain by fmri. *Magnetic resonance imaging*, 25(10):1347–1357, 2007.

[166] Amy Krain Roy, Zarrar Shehzad, Daniel S Margulies, AM Kelly, Lucina Q Uddin, Kristin Gotimer, Bharat B Biswal, F Xavier Castellanos, and Michael P Milham. Functional connectivity of the human amygdala using resting state fmri. *Neuroimage*, 45(2):614–626, 2009.

[167] Raymond Salvador, John Suckling, Martin R Coleman, John D Pickard, David Menon, and ED Bullmore. Neurophysiological architecture of functional magnetic resonance images of human brain. *Cerebral cortex*, 15(9):1332–1342, 2005.

[168] William W Seeley, Vinod Menon, Alan F Schatzberg, Jennifer Keller, Gary H Glover, Heather Kenna, Allan L Reiss, and Michael D Greicius. Dissociable intrinsic connectivity networks for salience processing and executive control. *The Journal of neuroscience*, 27(9):2349–2356, 2007.

[169] Xiaotong Shen, Wei Pan, Yunzhang Zhu, and Hui Zhou. On constrained and regularized high-dimensional regression. *Annals of the Institute of Statistical Mathematics*, 65(5):807–832, 2013.

[170] Hanif D Sherali and Warren P Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete. Math*, 3:411–430, 1990.

[171] Hanif D Sherali and Warren P Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discrete Appl. Math*, 52:83–106, 1994.

[172] Hanif D Sherali, Warren P Adams, and Patrick J Driscoll. Exploiting special structures in constructing a hierarchy of relaxations for 0-1 mixed integer problems. *Operations Research*, 46(3):396–405, 1998.

[173] Hanif D Sherali and Patrick J Driscoll. On tightening the relaxations of miller-tucker-zemlin formulations for asymmetric traveling salesman problems. *Operations Research*, 50(4):656–669, 2002.

[174] L. Simonetti, A.S. da Cunha, and A. Lucena. Polyhedral results and a branch-and-cut algorithm for the k-cardinality tree problem. *Mathematical Programming*, pages 1–28, 2012.

[175] L. Simonetti, F. Protti, Y. Frota, and C.C. Souza. New branch-and-bound algorithms for k-cardinality tree problems. *Electrical Notes in Discrete Mathmatics*, 37:27–32, 2011.

[176] Wolf Singer. Neuronal synchrony: a versatile code for the definition of relations? *Neuron*, 24(1):49–65, 1999.

[177] Maurice Sion et al. On general minimax theorems. *Pacific J. Math*, 8(1):171–176, 1958.

[178] Constantino Sotelo. Viewing the brain through the master hand of ramon y cajal. *Nature Reviews Neuroscience*, 4(1):71–77, 2003.

[179] Olaf Sporns. *Networks of the Brain*. MIT press, 2011.

[180] Cornelis J Stam. Modern network science of neurological disorders. *Nature Reviews Neuroscience*, 15(10):683–695, 2014.

[181] Cornelis J Stam and Jaap C Reijneveld. Graph theoretical analysis of complex networks in the brain. *Nonlinear biomedical physics*, 1(1):3, 2007.

[182] Steven H Strogatz. Exploring complex networks. *Nature*, 410(6825):268–276, 2001.

[183] Minghe Sun and Momiao Xiong. A mathematical programming approach for gene selection and tissue classification. *Bioinformatics*, 19(10):1243–1251, 2003.

[184] Zhaonan Sun, Nawanol Ampornpunt, Manik Varma, and Svn Vishwanathan. Multiple kernel learning and the smo algorithm. In *Advances in neural information processing systems*, pages 2361–2369, 2010.

[185] Kaustubh Supekar, Vinod Menon, Daniel Rubin, Mark Musen, and Michael D Greicius. Network analysis of intrinsic functional brain connectivity in alzheimer's disease. *PLoS computational biology*, 4(6):e1000100, 2008.

[186] Mingkui Tan, Ivor W Tsang, and Li Wang. Towards ultrahigh dimensional feature selection for big data. *The Journal of Machine Learning Research*, 15(1):1371–1429, 2014.

[187] Mingkui Tan, Li Wang, and Ivor W Tsang. Learning sparse svm for feature selection on very high dimensional datasets. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1047–1054, 2010.

[188] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, page 37, 2014.

[189] Marisa Thoma, Hong Cheng, Arthur Gretton, Jiawei Han, Hans-Peter Kriegel, Alex Smola, Le Song, Philip S Yu, Xifeng Yan, and Karsten M Borgwardt. Discriminative frequent subgraph mining with optimality guarantees. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3(5):302–318, 2010.

[190] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[191] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[192] Fujio Toriumi, Isamu Okada, Hitoshi Yamamoto, Hirohiko Suwa, Kiyoshi Izumi, and Yasuhiro Hashimoto. Classification of social network sites based on network indexes and communication patterns. In *Proceedings of International Workshop on Social Web Mining Co-located with IJCAI*, 2011.

[193] Martijn P van den Heuvel, Cornelis J Stam, Maria Boersma, and HE Hulshoff Pol. Small-world and scale-free organization of voxel-based resting-state functional connectivity in the human brain. *Neuroimage*, 43(3):528–539, 2008.

[194] Gary W Van Hoesen, Josef Parvizi, and Ching-Chiang Chu. Orbitofrontal cortex pathology in alzheimer's disease. *Cerebral Cortex*, 10(3):243–251, 2000.

[195] Francisco Varela, Jean-Philippe Lachaux, Eugenio Rodriguez, and Jacques Martinerie. The brainweb: phase synchronization and large-scale integration. *Nature reviews neuroscience*, 2(4):229–239, 2001.

[196] Virginia Vassilevska. Efficient algorithms for clique problems. *Information Processing Letters*, 109(4):254–257, 2009.

[197] James A Waltz, Zuzana Kasanova, Thomas J Ross, Betty J Salmeron, Robert P McMahon, James M Gold, and Elliot A Stein. The roles of reward, default, and executive control networks in set-shifting impairments in schizophrenia. *PloS one*, 8(2):e57257, 2013.

[198] Jinhui Wang, Liang Wang, Yufeng Zang, Hong Yang, Hehan Tang, Qiyong Gong, Zhang Chen, Chaozhe Zhu, and Yong He. Parcellation-dependent small-world brain functional networks: A resting-state fmri study. *Human brain mapping*, 30(5):1511–1523, 2009.

[199] Jinhui Wang, Xinian Zuo, and Yong He. Graph-based network analysis of resting-state functional mri. *Frontiers in systems neuroscience*, 4, 2010.

[200] Liang Wang, Chaozhe Zhu, Yong He, Yufeng Zang, QingJiu Cao, Han Zhang, Qiuhai Zhong, and Yufeng Wang. Altered small-world brain functional networks in children with attention-deficit/hyperactivity disorder. *Human brain mapping*, 30(2):638–649, 2009.

[201] Lifeng Wang and Xiaotong Shen. On ℓ_1-norm multiclass support vector machines. *Journal of the American Statistical Association*, 102(478), 2007.

[202] Duncan J Watts. *Small worlds: the dynamics of networks between order and randomness*. Princeton university press, 1999.

[203] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442, 1998.

[204] Shih-Jen Weng, Jillian Lee Wiggins, Scott J Peltier, Melisa Carrasco, Susan Risi, Catherine Lord, and Christopher S Monk. Alterations of resting state functional connectivity in the default network in adolescents with autism spectrum disorders. *Brain research*, 1313:202–214, 2010.

[205] Christian Windischberger, Herbert Langenberger, Thomas Sycha, Edda M Tschernko, Gabriele Fuchsjäger-Mayerl, Leopold Schmetterer, and Ewald Moser. On the origin of respiratory artifacts in bold-epi of the human brain. *Magnetic resonance imaging*, 20(8):575–582, 2002.

[206] L. A. Wolsey. *Integer Programming*. Wiley New York, 1998.

[207] Zenglin Xu, Rong Jin, Jieping Ye, Michael R Lyu, and Irwin King. Non-monotonic feature selection. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1145–1152. ACM, 2009.

[208] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In

Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on, pages 721–724. IEEE, 2002.

[209] Paul R Yarnold, Robert C Soltysik, and Gary J Martin. Heart rate variability and susceptibility for sudden cardiac death: An example of multivariable optimal discriminant analysis. *Statistics in medicine*, 13(10):1015–1021, 1994.

[210] BT Thomas Yeo, Fenna M Krienen, Jorge Sepulcre, Mert R Sabuncu, Danial Lashkari, Marisa Hollinshead, Joshua L Roffman, Jordan W Smoller, Lilla Zöllei, Jonathan R Polimeni, et al. The organization of the human cerebral cortex estimated by intrinsic functional connectivity. *Journal of neurophysiology*, 106(3):1125–1165, 2011.

[211] Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *The Journal of Machine Learning Research*, 11:3183–3234, 2010.

[212] Lei Yuan, Jun Liu, and Jieping Ye. Efficient methods for overlapping group lasso. In *Advances in Neural Information Processing Systems*, pages 352–360, 2011.

[213] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[214] Cun-Hui Zhang and Jian Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *The Annals of Statistics*, pages 1567–1594, 2008.

[215] Cun-Hui Zhang, Tong Zhang, et al. A general theory of concave regularization for high-dimensional sparse estimation problems. *Statistical Science*, 27(4):576–593, 2012.

[216] Yudong Zhang, Shuihua Wang, Genlin Ji, and Zhengchao Dong. An mr brain images classifier system via particle swarm optimization and kernel support vector machine. *The Scientific World Journal*, 2013, 2013.

[217] David C Zhu, Shantanu Majumdar, Igor O Korolev, Kevin L Berger, and Andrea C Bozoki. Alzheimer's disease and amnestic mild cognitive impairment weaken con-

nections within the default-mode network: a multi-modal imaging study. *Journal of Alzheimer's Disease*, 34(4):969–984, 2013.

Appendix A

APPENDIX

A.1 Abbreviations

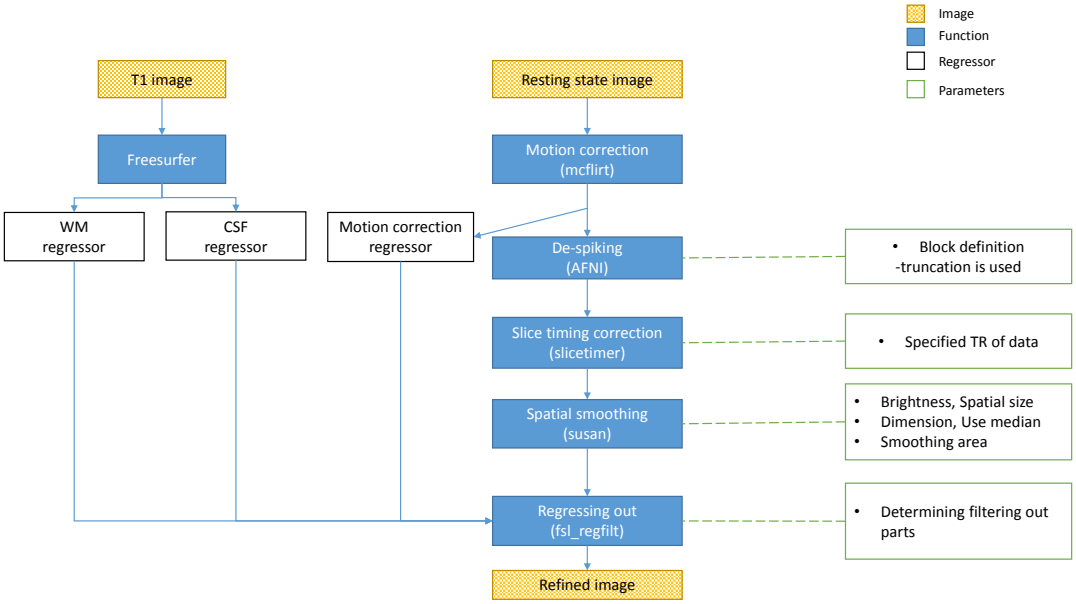
AD	Alzheimer's Disease
ASD	Autism Spectrum Disorder
BnB	Branch-and-Bound
BnC	Branch-and-Cut
fMRI	Functional Magnetic Resonance Imaging
KCTP	k Cardinality Tree Problem
LP	Linear Programming
LR	Logistics Regression
MIP	Mixed Integer Programming
MINLP	Mixed Integer Non Linear Programming
MST	Minimum Spanning Tree
PD	Parkinson's Disease
SVM	Support Vector Machine
ROI	Region of Interest

Table A.1: Introduced abbreviations in this thesis

A.2 fMRI Data Preprocessing Pipeline

No matter the design, multiple volumes (made from multiple slices) have been acquired in time. Before getting data out, one need to make sure the signal from each voxel contains the right temporal and spatial information. Thus, in general, researches have preprocessed acquired fMRI data to collect data as clean as possible. I depict a general pipeline for preprocessing at following Figure A.1. In Figure A.1, WM and CSF indicates 'White Matter'

Figure A.1: Preprocessing pipeline for fMRI data



and 'Cerebrospinal Fluid' respectively. TR means 'Repetition time'.

- **Regressing out:** From the image atlas (i.e., T1 image), we can obtain three regressors in terms of WM, CSF, and Motion corrections. Signal changes in the WM and CSF primarily reflect non-neural fluctuations such as scanner instabilities, subject motion, and physiological artifacts (e.g. respiration and cardiac effects) [205]. These signals are largely independent from the BOLD signal fluctuations recorded in gray matter, and may introduce temporal coherences that lead to an overestimation of functional connectivity strength. Thus, in order to collect fine signals, one need to estimate

WM and CSF and exclude from the original data successfully. Similarly, unexpected motion such as accidental head movement during an experiment should be excluded from the data. By means of three regressors, one can remove noisy signals from the data.

- **De-spiking:** It is removal of spikes which are made by brightness changes due to the instability of electrical signals in the scanner. In general, they have appeared in the image with regular stripe patterns or dots. As scanning technologies develop, the occurrence frequency of spikes is decreasing, but when they are occurred unexpectedly, they can give negative effects to the analysis.

- **Slice timing Correction:** One of the important step of preprocessing of fMRI data is the proper treatment of differences in the recording time of each slices. The problem caused by different slice scanning time originates from the fact that a functional map of a whole brain is not covered but a series of successively measured two dimensional (2D) slices. For example, suppose that an image volume is collected by 30 slices and each volume TR is 3 seconds. And then, the recorded time of last slice is measured almost 3 seconds later than the data of the first slice.

- **Spatial smoothing:** It is a process such that data points are being averaged with their neighborhoods. Being averaged effects are similar to a low pass filtering meaning that high frequencies are excluded from the data while enhancing low frequencies. Those smoothing result that the sharp edges of fMRI images become to be blurred and also spatial correlation is more pronounced.