COMBINATORIAL AND NONLINEAR OPTIMIZATION METHODS WITH APPLICATIONS IN DATA CLUSTERING, BICLUSTERING AND POWER SYSTEMS

By

NENG FAN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2011

To my parents, family and friends

ACKNOWLEDGMENTS

First and foremost, I would like to thank my supervisor, Dr. Panos M. Pardalos, for his guidance and support. His knowledge, experience, enthusiasm and sense of humor have been truly valuable. Without his help and suggestions, this dissertation cannot be completed.

I thank my dissertation supervisory committee members, Dr. William Hager, Dr. Guanghui Lan, Dr. J. Cole Smith, and Dr. Stanislav Uryasev for providing suggestions at the very beginning of my research and also evaluating my dissertation.

I would like to thank all my collaborators: Altannar Chinchuluun, Qipeng Zheng, Nikita Boyko, Petros Xanthopoulos, Hongsheng Xu, Jicong Zhang, Syed Mujahid, Pando Georgiev, Steffen Rebennack and Ashwin Arulselvan. I am grateful for their efforts and suggestions. I want to thank all my fellow students in Department of Industrial and Systems Engineering at University of Florida, especially friends in the Center for Applied Optimization, for their helpful discussions and their friendship.

I am very grateful to Dr. Feng Pan for bringing me to D-6 group at Los Alamos National Laboratory from August 2010 to May 2011. Especially, the D-6 group and Center for Nonlinear Studies provided me a very great environment to conduct my research in the smart grid and power systems. I want to thank Dr. David Izraelevitz, Dr. Michael Chertkov and Dr. Alexander Gutfraind for their support and discussions.

Of course, I am grateful to my parents for their patience and support. Finally, I would like to thank all my friends during my PhD study in Gainesville, Florida.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure page

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

COMBINATORIAL AND NONLINEAR OPTIMIZATION METHODS WITH APPLICATIONS IN DATA CLUSTERING, BICLUSTERING AND POWER SYSTEMS

By

Neng Fan

August 2011

Chair: Panos M. Pardalos
Major: Industrial and Systems Engineering

With the increasing number of databases appearing in computational biology, biomedical engineering, consumers' behavior survey, social networks, power systems, and many other areas, finding the useful information and discovering the knowledge within these databases are important issues nowadays. Data mining is one of the most important techniques to extract information from large data sets. Data clustering and biclustering are widely used tools in data mining. The graph partitioning problem is a classic combinatorial optimization problem and it is known to be NP-complete.

In the dissertation, for the graph partitioning problem, instead of obtaining two subsets, we use integer programming formulation to obtain more than two subsets directly. Besides its deterministic version, we study the uncertain versions for changes on weights of edges by robust and stochastic optimization approaches. All these models and approaches are for the classic graph partitioning problem, which is stated as graph partitioning by minimum cut in this dissertation.

Besides minimum cut, which is the direct summation of weights of edges connecting different subsets, we also study the graph partitioning problem by Ratio cut and Normalized cut. Both Ratio cut and Normalized cut were introduced for balancing the cardinalities of all subsets, and they are quite useful for data clustering. In the dissertation, we use graph partitioning method to obtain nonoverlap and exclusive clusters. The graph partitioning methods by Ratio cut and Normalized cut for clustering

are formulated as mixed integer nonlinear programs (MINLPs). We prove that the spectral methods, which were widely used for clustering, are solving relaxations of our integer programming models. We also present the semidefinite programming (SDP) relaxations for these integer programming models, and our numerical experiments show that better solutions than spectral methods can be obtained. Since SDP is a subfield of convex optimization concerned with the optimization of a linear objective function over the intersection of the cone of positive semidefinite matrices, and it can be solved efficiently by interior point methods and many other methods. Our SDP relaxations can find many applications in data clustering. Additionally, we present equivalent integer linear programming formulations for our MINLPs, and they can be applied for small data cases to obtain exact solutions. Since a bipartite graph is a special case of a general graph, the bipartite graph partitioning problem can be easily transformed into the graph partitioning problem. For data biclustering, bipartite graph partitioning models with Ratio cut and Normalized cut are also studied from the adapted results of graph partitioning models for clustering.

In the dissertation, we also use graph partitioning methods to form islands in a power grid and formulate these problems as mixed integer nonlinear programs. With these mathematical programming models, optimal formation of islands can be obtained and the different approaches can be compared. Through simulations, experimental results are analyzed and compared to provide insight for power grid islanding.

CHAPTER 1
INTRODUCTION

In this chapter, the motivation and the background for research of the dissertation will be introduced first. Then, the main contribution of the dissertation is stated. After that, the organization of the dissertation is outlined.

1.1 Background

With the amount number of databases appearing in computational biology, biomedical engineering, consumers' behavior survey and social networks, finding the useful information behind these data and grouping the data are important issues nowadays. *Clustering* is a method to classify of objects into different groups, such that the objects in the same group share some common traits or are similar in some sense. It is a method of unsupervised classification, and a common technique for data analysis in many fields, including data mining, machine learning, pattern recognition, image analysis and bioinformatics. Clustering has been widely studied in the past 20 years. The methods for clustering include K-means clustering, fuzzy c-means clustering, and spectral clustering.

However, clustering only groups objects without considering the features of each object may have. In other words, clustering compares two objects by the features that two share, without depicting the different features of the two. A method simultaneously groups the objects and features is called *biclustering* such that a specific group of objects has a special kind group of features. More precisely, a biclustering is to find a subset of objects and features satisfying that these objects are related to those features to some level. Such kind of subsets are called biclusters. Meantime, biclustering does not require objects in the same bicluster to behave similarly over all possible features, but to highly have specific features in this bicluster. Biclustering also has abilities to find hide features and specify them to some subsets of objects. The biclustering problem is

to find biclusters in data sets, and it may have different names such as co-clustering, two-mode clustering in some literatures.

Both clustering and biclustering are performed on data matrices. The data matrix for clustering is a symmetric nonnegative matrix and it is always chosen as a similarity matrix, whose entries denote the similarity of objects. The matrix for biclustering is a rectangular matrix with rows corresponding to objects, columns to features, and entries to the expression level of corresponding features in the objects. This matrix is called expression matrix.

The *Graph Partitioning Problem* (GPP) consists of dividing the vertex set of a graph into several disjoint subsets so that the cut among these disjoint subsets is minimized. The three most widely used cuts for clustering and biclustering are *Ratio cut*, *Normalized cut* and *Minimum cut*. The Minimum cut is the sum weight of the edges with ends in distinct subsets. The Ratio cut is a fraction of the cuts to the sizes of the corresponding subsets, and the Normalized cut is a fraction of the cuts to the weighted degrees of vertices. The Ratio cut and Normalized cut are introduced to balance the the cardinalities of subsets for graph partitioning, and they are used for different purposes of clustering and biclustering.

In the graph partitioning models for clustering, the objects are corresponding to vertices, and the similarity matrix for clustering is chosen as the weight matrix of the graph. In the bipartite graph partitioning model for biclustering, the set of objects corresponds to the first vertex set and the set of features to the other one. The expression matrix is chosen as the biadjacency weight matrix of the bipartite graph. By the graph partitioning method for clustering and biclustering, we obtain nonoverlap and exclusive clusters and biclusters.

The graph partitioning problem is a classic combinatorial optimization problem, and it is known to be NP-complete. With different cuts, the graph partitioning problem can be formulated as an integer program in mathematical programming. For the problem with

Minimum cut, which has been studied for a long time in history, a quadratic programming formulation is always used. For the problem with Ratio cut or Normalized cut, it can be formulated as a mixed integer nonlinear program (MINLP) as modeled in Chapter 3. The MINLP is a difficult class of optimization problems with many applications. In its most general form, a MINLP problem can be formulated as

$$\begin{aligned} \min \quad & g_0(x) \\ s.t. \quad & g_i(x) \leq 0, \quad i = 1, \cdots, m \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, \end{aligned}$$

where $g_i : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear function of x for all $i = 0, 1, \cdots, m$, and sometimes nonconvex. Global optima of MINLP problems can be obtained by linear relaxation for lower bounds. For some class of MINLP problems, some linearization techniques can be used to transform them into equivalent integer linear programming problems.

Recently, optimization methods have found many applications in electric power systems. For example, optimization models are used in these problems: unit commitment for automatic generation control, optimal power flow and economic dispatch, transmission line switching and load shedding for power transmission and distribution, the $N-1$ or $N-k$ contingency analysis for system security, maximum load point and voltage collapse point for system stability, and etc. Some other problems in power systems are still under discussion for reliability, security and stability in order to construct an efficient, economic and reliable smart grid.

1.2 Contributions

The graph partitioning method for clustering by Ratio cut or Normalized cut has been studied for many applications in network analysis, image segmentation, biomedical engineering, text mining and etc. The spectral method is used in both clustering and biclustering. This method uses the eigenvectors and eigenvalues of the Laplacian matrix

of the corresponding graph for clustering or biclustering. However, this method was used directly in data mining without the proof of effectiveness and validity.

In this dissertation, we prove spectral methods are relaxations of our MINLP optimization models for clustering by graph partitioning method with Ratio cut or Normalized Cut. In the meantime, we study the semidefinite programming (SDP) relaxations of these MINLP optimization models. Numerical experiments show that SDP relaxations can obtain better results than spectral methods. We also present equivalent integer linear programming formulations to obtain exact solutions for small data cases. Similarly, models of bipartite graph partitioning by Ratio cut and Normalized are also studied for biclustering. Additionally, all methods we proposed can obtain more than any number of clusters or biclusters directly, while past research results usually obtained just two clusters.

In previous research, the graph partitioning by minimum cut divides the vertex set into subsets of equal or different by 1 cardinalities. Linear, quadratic and semidefinite programming methods are used for such cases. Most of these approaches are still based on relaxations of exact formulations. In this dissertation, considering the cardinality within a loose range and obtaining more than two subsets directly, we present three equivalent zero-one integer linear programming reformulations. For the graph partitioning problem with interval uncertain weights of edges, robust optimization models with two decomposition algorithms are introduced. We also introduce the two-stage stochastic graph partitioning problem and present the stochastic mixed integer programming formulation for this problem with finite explicit scenarios. For solving this problem, we present an equivalent integer linear programming formulation where some binary variables are relaxed to continuous ones. Additionally, for some specific graphs, we present a more simplified linear programming formulation. All formulations for graph partitioning problem with minimum cut, including robust and stochastic optimization

versions, are tested on randomly generated graphs with different densities and different numbers of scenarios.

A power grid island is a self-sufficient subnetwork in a large-scale power system. In weakly connected islands, limited inter-island power flows are allowed. In this dissertation, we use graph partitioning methods to form islands in a power grid and formulate these problems as MINLPs. With these mathematical programming models, optimal formation of islands can be obtained and the different approaches can be compared. Through the simulations, experimental results are analyzed and compared to provide insight for power grid islanding.

1.3 Outline of the Dissertation

In this dissertation, we use graph partitioning based optimization methods for data clustering and biclustering, discuss many aspects of the graph partitioning by minimum cut, and also apply this combinatorial and nonlinear optimization approach in power systems.

Chapter 2 briefly reviews previous methods for clustering and biclustering, as well as some preliminary results for the graph partitioning problem.

Chapter 3 studies quadratically constrained programming approach, semidefinite programming and spectral relaxation methods for clustering and biclustering by graph partitioning method with Ratio cut and Normalized cut.

Chapter 4 introduces several linear and quadratical programming approaches for graph partitioning by minimum cut, and its uncertain forms by robust and stochastic optimization methods.

Chapter 5 uses this combinatorial and nonlinear optimization approach to study the islanding problem in power systems. Many numerical experiments are performed on IEEE test power systems.

Chapter 6 concludes the dissertation and presents some future research directions.

CHAPTER 2
REVIEW OF CLUSTERING, BICLUSTERING AND GRAPH PARTITIONING

In this chapter, literature reviews for both clustering and biclustering will be briefly stated first, and optimization methods will be reviewed with more details. Then, the mathematical programming methods for the graph partitioning problem will be reviewed briefly. Because of relationships between clustering and graph partitioning, some preliminary results for both will be studied in Section 2.3, and these results will be cited throughout Chapter 3 and Chapter 4.

2.1 Review of Data Clustering and Biclustering

As mentioned in 1.1, both clustering and biclustering are performed on data matrices: clustering on similarity matrix and biclustering on expression matrix.

Similarity Matrix. This data matrix has both rows and columns corresponding to a set of samples (objects), with entry measuring the similarity between two corresponding samples. It has same numbers of rows and columns, and it is symmetric. This matrix can be called sample-by-sample matrix. Throughout the dissertation, $W = (w_{ij})_{N\times N}$ is used to denote this matrix with N samples. Since we are using graph partitioning to study clustering, this matrix is corresponding to the weight matrix of a graph and has some requirements as discussed in Definition 2.3.

Note the matrix for clustering can also be dissimilarity matrix with entry denoting the dissimilarity between a pair of samples. There are many similarity measurement functions to compute the (dis)similarity entries [132], such as Euclidean distance, Mahalanobis distance, etc. So the similarity matrix can be computed from the expression matrix.

Expression Matrix. This data matrix has rows corresponding to samples (objects), columns to features, with entry measuring the expression level of a feature in a sample. Each row is called a feature vector of the sample. We can call this matrix as sample-by-feature matrix. Throughout the dissertation, $A = (a_{ij})_{N\times M}$ is used to

denote this matrix with N samples with M features. More details and applications of this matrix for biclustering will be discussed in Sections 3.5 and 4.1.

Sometimes, the matrix is formed from all samples' feature vectors, and the features' level in this sample will be observed directly. Generally we just scale and then put these vectors together to form a matrix if all vectors have the same length, which means they have the same set of features. However, the feature vectors may not conform each other. Under this case, we should add values (may be 0) to vectors with no corresponding features in order to form same-length vectors. In some applications, there are always large set of samples with limited features. Since the developments of biclustering are including some time series models [90, 119], and another kind of time series data are also used in biclustering. This data also can be viewed as stored in a matrix with that rows denote samples while columns from left to right denote observed time points. For some qualitative features or some cases, the data matrix is a kind of sign matrix. Some biclustering algorithms can work under this situation.

Sometimes, before processing algorithms on the matrix, some steps are used, such as normalization, discretization, value mapping and aggression, and the details of these data preparation operations are available at [29].

2.1.1 Review of Clustering

Clustering is the assignment of objects into different groups (called clusters), so that the objects in the same group share some common traits or are similar in some sense. It is a method of unsupervised classification, and a common technique for data analysis in many fields, including data mining, machine learning, pattern recognition, image analysis and bioinformatics.

Clustering has been widely studied in the past 20 years. Most widely used methods for clustering include K-means clustering [88], fuzzy c-means clustering [10], and spectral clustering [101]. A general review of clustering is given by Jain et al. in [68]. This paper presented an overview of pattern clustering methods from a statistical

pattern recognition perspective. The techniques they reviewed include hierarchical clustering algorithms, partition algorithms, mixture-resolving and mode-seeking algorithms, nearest neighbor clustering, fuzzy clustering, artificial neural networks for clustering, evolutionary and search-based approaches for clustering, and etc. They also discussed applications of clustering in image segmentation, object recognition, and information retrieval.

In the survey paper [132], Xu and Wunsch listed similarity and dissimilarity measure for quantitative features, such as Minkowski distance, Eulidean distance, city-block distance, sup distance, Mahalanobis distance, Pearson correlation, point symmetry distance, and Cosine similarity. Algorithms were classified into hierarchical, squared error-based, mixture densities-based, graph-theory based, combinatorial search techniques-based, fuzzy, neural networks-based, and kernel-based clustering methods. They also reviewed some data sets for applications, such as gene expression data, DNA or protein sequences for clustering.

Recently in [7], Berkhin reviewed recent advances methods in clustering, and also discussed clustering of high dimensional data by dimensionally reduction and general issues of clustering, such as assessment of results, choice of appropriate number of clusters, data preparation, proximity measures, and handling outliers.

Graph models are always used for clustering as reviewed in [110]. The spectral clustering method is based on the Laplacian matrix of the graph [47] and sometimes it is called spectral graph partitioning [101]. Two different cuts in graph partitioning are defined for clustering: Ratio cut [56] and Normalized cut [115]. A tutorial on spectral clustering is given by Ding [34]. The spectral method always begins with bipartition of a graph [47, 56, 101, 115] and hierarchical clusters of a multi-way partitioning can be obtained recursively by this approach. The direct multi-way graph partitioning for clustering has been studied in [21] for Ratio cut and [55] for Normalized cut. Recently, a semidefinite relaxation for the multi-way Normalized cut was studied in [131]. In these

papers, the terms “clustering” and “graph partitioning” are not distinguished clearly. In Chapter 3, we are concentrating on the graph partitioning methods by the Ratio cut and Normalized cut and claim that our methods can be used for clustering.

2.1.2 Review of Biclustering

Biclustering, or called co-clustering, two-mode clustering, is an extension of clustering. It allows simultaneous grouping of the objects and features such that the objects in the same group (called bicluster) are highly related to a group of features to some level, and vice versa. Biclustering has found many applications in microarray and gene expression analysis [2, 5, 19, 20, 25, 26, 54, 77, 81, 90, 93, 96, 103, 105, 113, 119, 121, 136, 142], biomedical engineering [16, 77, 98], computational biology [33, 67, 89], text mining [30] and etc. It has been studied recently and several reviews can be found in [18, 38, 89, 122]. Biclustering is related to bipartite graph partitioning [30, 39, 104, 141]. The recent and most widely used methods include Cheng and Church’s algorithm [25, 135, 136], random walk biclustering algorithm [2], order-preserving submatrix algorithm [5], iterative signature algorithm [67], xMotif [93], optimal re-ordering algorithm [33], algorithm based on nonsmooth nonnegative matrix factorization [20, 100], binary inclusion maximal biclustering algorithm (Bimax) [103], spectral biclustering based on bipartite graphs [30, 39, 115], statistical algorithmic method for bicluster analysis (SAMBA) [120, 121], information theoretic biclustering algorithm [31], Bayesian biclustering model [54, 81, 113], cMonkey [105], and etc. Most of biclustering algorithms are unsupervised classification. It does not need to have any training sets to supervise biclustering. But supervised biclustering methods are also useful in some cases of biomedicine applications [17, 18, 96].

Obviously, the objective of biclustering is to find biclusters in data. In clustering, the obtained clusters should have the propositions that the similarities among the samples within each cluster are maximized and the similarities between samples from different clusters are minimized. For biclustering, the samples and features in

each bicluster are highly related. But this does not mean the samples in this bicluster do not have other features, they just have the features in this bicluster more obvious and they still share other features. Thus, in each bicluster, the relations between the samples and features are maximized closer rather than relations between samples (features) from this bicluster and features (samples) from another bicluster. There is no standard for justifying the algorithms. In distinct applications of biclustering, a specific or several objectives should be met so some algorithms are designed to satisfy these requirements. There are some methods are trying to compare different algorithms, and we refer to [89, 103, 106, 142].

Some biclustering algorithms allow that one sample or feature can belong to several biclusters (called overlapping) while some others produce exclusive biclusters. In addition, some algorithms have the property that each sample or feature must have its corresponding bicluster while some others need not to be exhaustive and can allow only find one sub-matrix or several ones from data matrix to form the biclusters.

The first approach to biclustering is “direct clustering of data matrix” by Hartigan [60] in 1972. But the term “biclustering” was famous after Cheng and Church [25] using this technique to do gene expression analysis. After that, many biclustering algorithms are designed in different areas’ applications, such as biological network, microarray data, word-document co-clustering, biomedical engineering, etc., of which the most popular applications are in microarray data and gene expression data.

In 2004, Madeira and Oliveira [89] surveyed the biclustering algorithms for biological data analysis. In this survey, they identified the biclusters into four major classes: biclusters with constant values, with constrant values on rows or columns, with coherent values, and with coherent evolutions. The biclustering structures of a data matrix are classified into nine groups according to algorithms: single bicluster, exclusive row and column biclusters, checkerboard structure, exclusive rows biclusters, exclusive columns biclusters, nonoverlapping biclusters with tree structure, nonoverlapping nonexclusive

biclusters, overlapping biclusters with hierarchical structure, and arbitrarily positioned overlapping biclusters. In addition, they divided the algorithms into five classes: iterative row and column clustering combination, divide and conquer, greedy iterative search, exhaustive bicluster enumeration, and distribution parameter identification. A comparison of these algorithms according to the above three classes was given in this survey.

Another review about biclustering algorithms is by Tanay, Sharan and Shamir in [122] in 2004. In this survey, nine widely used algorithms are reviewed and given with their pseudocodes. One review of biclustering is by Busygin, Prokopyev and Pardalos in [18], and 16 algorithms are reviewed with their applications in biomedicine and text mining. Recently in [38], Fan, Boyko and Pardalos reviewed the backgrounds, motivation, data input, objective tasks, and history of data biclustering. The bicluster types and biclustering structures of data matrix are defined mathematically. Most recent algorithms, including OREO, nsNMF, BBC, cMonkey, etc., are reviewed with formal mathematical models.

Since the development of biclustering algorithms, many softwares are designed to include several algorithms, including BicAT [4], BicOverlapper [109], BiVisu [24], toolbox by R (biclust) [69] and etc. These software or packages allow to do data processing, bicluster analysis and visualization of results, and can be used directly to construct images.

In the toolbox named BicAT [4], it provides different facilities for data preparation, inspection and postprocessing such as discretization, filtering of biclusters according. Several algorithms of biclustering such as Bimax, CC, XMotifs, OPSM are included, and three methods of viewing data including matrix (heatmap), expression and analysis. The software BicOverlapper [109] is a tool for overlapping biclusters visualization. It can use three different kinds of data files of orginal data matrix and resulted biclusters to construct beautiful and colorful images such as heatmaps, parallel coordinates, TRN

graph, bubble map and overlapper. The BiVisu [24] is also a software tool for bicluster detection and visualization. Besides bicluster detection, BiVisu also provides functions for pre-processing, filtering and bicluster analysis. Another software is a package written by R [69], biclust, which contains a collection of bicluster algorithms, such as Bimax, CC, plaid, spectral, xMotifs, etc, preprocessing methods for two way data, and validation and visualization techniques for bicluster results. For individual biclustering software, there are also some package available [18, 122].

2.2 Review of the Graph Partitioning Problem

In this section, the graph partitioning problem actually denotes the graph partitioning problem with minimum cut, which is the direct summation of weights of edges connecting different subsets. This problem is an NP-complete combinatorial optimization problem [51]. Applications of graph partitioning can be found in VLSI design [72], data mining [21, 30, 35, 56, 104, 110, 141], biological or social networks [41, 44], power systems [11, 83, 111, 123, 139], parallel computing [61] and etc.

A very early method for graph partitioning problem is the Kernighan-Lin algorithm [75], which is a heuristic algorithm. There are several surveys [1, 37, 48] regarding heuristic and exact algorithms for graph partitioning. Some heuristic algorithms can compute approximate solutions very fast, for example, spectral methods in [21, 30, 34, 39, 55, 63, 101], multilevel algorithms in [62, 72–74, 117, 127], optimization-based methods in [46], and etc. Software based on heuristic methods include METIS, Chaco, Party, PaToH, SCOTCH, Jostle, Zoltan and HUND as reviewed in [59].

Some mathematical programming methods, including linear programming [14, 42, 84], quadratic programming [42, 57–59], and many semidefinite programming relaxations [3, 70, 71, 84, 129], were also used for graph partitioning.

Some methods ([3, 57, 59, 71]) decompose the vertex set into two subsets, or called bisection, while some ones ([42, 55, 58, 91]) obtain more than two subsets directly. In previous research, the graph is partitioned into equal or different by 1

cardinalities for all subsets by linear programming [84] or semidefinite programming [70, 84]. Some approaches, for example, quadratic [57, 58] and semidefinite [129] programming methods, require the given cardinalities of all subsets.

Through the whole dissertation, all methods can obtain more than two subsets directly. In Chapter 3, we use a loose restriction for graph partitioning and clustering that all cardinalities of subsets take values in the integer range $[C_{min}, C_{max}]$. The cardinalities of subsets are one reason that Ratio cut and Normalized were introduced for graph partitioning.

2.3 Preliminary of Graph Partitioning and Clustering

As discussed in Section 1.2, we are concentrating on direct multi-way clustering. Because of connections between clustering and graph partitioning, our method is based on graph partitioning with respect to different cuts. In this section, we first define the partition matrix for direct multi-way partitioning. The definitions of cuts as well as some other concepts are also studied in the section. At the end of this section, we establish the optimization models of graph partitioning for clustering. Assume that the matrix for clustering is $W = (w_{ij})_{N\times N}$. Throughout this section, we are finding K clusters in the data matrix W as presented in Section 2.1.

Partition Matrix X

Definition 2.1. *The* partition matrix *is defined as a rectangular* $N \times K$ *matrix* $X = (x_{ik})_{N\times K}$*, where* $x_{ik} \in \{0, 1\}$*, row sums are* $\sum_{k=1}^{k} x_{ik} = 1$ *for all* $i = 1, \cdots, N$*, and column sums are* $\sum_{i=1}^{N} x_{ik} \in [C_{min}, C_{max}]$ *for all* $k = 1, \cdots, K$*.*

This matrix includes all decision variables in our graph partitioning models for clustering. The number N is the number of objects to perform clustering and K is the number for multi-way or K-way clustering. The clustering by this matrix X is nonoverlapping and exclusive.

The constraint of the sum $\sum_{k=1}^{k} x_{ik} = 1$ requires to put the ith object into exactly one cluster. The column sum $\sum_{i=1}^{N} x_{ik} := n_k$ defines the size or cardinality constraint of the

kth cluster. Since $x_{ik} \in \{0, 1\}$, the row sums take integer values between the lower size bound C_{min} and upper bound C_{max}. These two bounds are known parameters and can be chosen roughly from $\{1, \cdots, N\}$. More discussions of the cardinality constraint will be stated in Section 4.1.

Remark 1. *All row sums and column sums ensure that each object belongs to exactly one cluster and all objects have corresponding clusters. This is guaranteed by the fact that*

$$\sum_{k=1}^{K} n_k = \sum_{k=1}^{K} \sum_{i=1}^{N} x_{ik} = \sum_{i=1}^{N} \sum_{k=1}^{K} x_{ik} = \sum_{i=1}^{N} 1 = N,$$

which means that n_k can take any integer values in $[C_{min}, C_{max}]$ but their sum is fixed as N. Thus, the partition matrix has exactly N 1's.

Graph Partitioning

Definition 2.2. *Let $G = (V, E)$ be an undirected graph with a set of vertices $V = \{v_1, v_2, \cdots, v_N\}$ and a set of edges $E = \{(v_i, v_j) : \text{ edge between vertices } v_i \text{ and } v_j, 1 \leq i, j \leq N\}$, where N is the number of vertices. The K-*graph partitioning *consists of dividing the vertex set V into K disjoint subsets $V_1, V_2, \cdots, V_K$, i.e., $V_1 \cup V_2 \cup \cdots \cup V_K = V$ and $V_k \cap V_{k'} = \emptyset$ for all $k, k' = 1, 2, \cdots, K, k \neq k'$.*

The graph G is undirected and no loops are allowed. In the dissertation, we assume that each edge of G has nonnegative weights and the vertex has no weight. Always, this is a weighted graph. Let Π be a partition of the graph G, i.e., $\Pi = (V_1, V_2, \cdots, V_N)$, where the subsets $V_1, V_2, \cdots, V_N$ satisfy the Definition 2.2. Now, we construct a bijection between the partition matrix X and the graph partitioning Π.

Let the ith row of X correspond to vertex v_i and the kth column correspond to subset V_k of Π. The entry x_{ik} defines that whether a vertex v_i is in subset V_k if $x_{ik} = 1$ or not if $x_{ik} = 0$. The column sum $\sum_{i=1}^{N} x_{ik}$ is corresponding to the number of vertices in subset V_k. The constraint $\sum_{k=1}^{K} x_{ik} = 1$ of X ensures that vertex v_i can belong to exact one subset, and this also ensures that the subsets are disjoint. Assume that $|V_k| = n_k = \sum_{i=1}^{N} x_{ik}$ for $k = 1, \cdots, K$. Thus we have $\sum_{k=1}^{K} n_k = |V|$, which ensures that

the union of these subsets is V. Usually, the n_ks are unknown before partitioning and we assume that they take integer values in a range as that in the partition matrix. We have transformed the partition matrix into a graph partitioning Π. The reverse direction of the bijection can be constructed easily.

Therefore, the feasible region for the graph partitioning Π into K subsets can be expressed as

$$\mathcal{F}_K = \{(x_{ik})_{N\times K} : x_{ik} \in \{0,1\}, \sum_{k=1}^{K} x_{ik} = 1, C_{min} \leq \sum_{i=1}^{N} x_{ik} \leq C_{max}\}. \tag{2–1}$$

Remark 2. *Generally, the number K can be chosen from the set $\{2, 3, \cdots, N-1\}$. The value $K = 1$ is to put all objects into one cluster and $K = N$ means one cluster for each object. These are trivial cases. For minimum cut (Definition 2.5) of graph partitioning, the determination of K is studied in Section 4.1 by introducing a penalty function.*

Remark 3. *The size or cardinality n_k of the subset V_k is always unknown before partitioning. For minimum cut (Definition 2.5) of graph partitioning, the cardinalities are always chosen as equals N/K [70, 84], pre-given integers [58, 129] or values in a range [42, 43]. In Chapter 3, we use a loose restriction for graph partitioning and clustering that all the cardinalities of subsets take values in the integer range $[C_{min}, C_{max}]$. The cardinalities of subsets are one reason that Ratio cut and Normalized were introduced for graph partitioning. As shown in Section 3.2 and Section 3.3, the relaxations do not need any information of the range for clustering.*

Matrices on Graphs

Definition 2.3. *The* weight matrix *$W = (w_{ij})_{N\times N}$ of the graph $G = (V, E)$ is a symmetric matrix with nonnegative entries. The entry $w_{ij} > 0$ denotes the weight of edge (v_i, v_j) and $w_{ij} = 0$ if no edge (v_i, v_j) exists between vertices v_i and v_j.*

The weighted degree d_i *of the vertex* v_i *is defined as the sum of weights of the edges incident with itself, i.e.,* $d_i = \sum_{j=1}^{N} w_{ij}$*. The* weighted degree matrix *is a diagonal matrix formed by the degrees of all vertices, i.e.,* $D = diag(d_1, d_2, \cdots, d_N)$*.*

The Laplacian matrix L *is defined as the difference of* D *and* W*, i.e.,* $L = D - W$*.*

For undirected graphs G without loops, the matrix W is symmetric and $w_{ii} = 0$ for $i = 1, \cdots, N$. It is the adjacency matrix of G if $w_{ij} = 1$ denotes the existence of edge (v_i, v_j) and otherwise $w_{ij} = 0$. Thus, the matrix W is also called adjacency weighted matrix for G.

The notation *diag* denotes to form a diagonal matrix from the vector, and we also use it in the following as the step to form a vector from the diagonal entries of the matrix.

Cut

Definition 2.4. *The* cut set *of the partition* Π *includes the edges with two ends in different subsets. The cut set between* V_k *and* $V_{k'}$ *is the set of edges* (v_i, v_j)*s with* $v_i \in V_k, v_j \in V_{k'}$ *for* $k \neq k'$*. The* cut *between subsets* V_k *and* $V_{k'}$ *is the sum of weights of the edges in the corresponding cut set, i.e., for* $k \neq k'$*,*

$$cut(V_k, V_{k'}) = \sum_{i,j: v_i \in V_k, v_j \in V_{k'}} w_{ij}.$$

The edge in the cut set is called cut edge. In the definition of cut, we do not require that $(v_i, v_j) \in E$ since $w_{ij} = 0$ if no edge exists. The fact that $cut(V_k, V_{k'}) = cut(V_{k'}, V_k)$ can be easily drawn from the symmetry of the weight matrix W.

In the following, the notation $cut(V_a, V_b)$ is used to denote the sum of weights of edges with one end in V_a and another in V_b, whether V_a and V_b are disjoint or not (the two vertex sets can be the same, or one is a subset of another). For example, the notation $cut(V_1, V_1)$ is the sum of weights of edges with two ends in vertex set V_1, and the fact $cut(V_1, V) = cut(V_1, V_1 \cup V_2 \cup \cdots \cup V_K) = 2cut(V_1, V_1) + cut(V_1, V_2) + \cdots + cut(V_1, V_K)$ is obtained by the following proposition.

Proposition 2.1. *For the graph partitioning $\Pi = (V_1, \cdots, V_K)$, the cuts can be expressed by the partition matrix X in the following:*

$$cut(V_k, V_{k'}) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}x_{ik}x_{jk'}, \quad k \neq k',$$

$$cut(V_k, V_k) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}x_{ik}x_{jk},$$

$$cut(V_k, V) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}x_{ik}.$$

The first one is from the Definition 2.4 of cut. The second one is the sum of weights of the edges with two ends in V_k and each edge contributes once. Assume that $d_{V_k} = \sum_{i:v_i\in V_k} d_i = \sum_{i:v_i\in V_k}\sum_{j=1}^{N} w_{ij}$, thus we have $d_{V_k} = cut(V_k, V)$. If these expressions of cuts present in the matrix forms, we have the following proposition.

Proposition 2.2. *Based on the partition matrix X, we have*

$$X^T WX = \begin{pmatrix} 2cut(V_1, V_1) & cut(V_1, V_2) & \cdots & cut(V_1, V_K) \\ cut(V_2, V_1) & 2cut(V_2, V_2) & \cdots & cut(V_2, V_K) \\ \vdots & \vdots & \ddots & \vdots \\ cut(V_K, V_1) & cut(V_K, V_2) & \cdots & 2cut(V_K, V_K) \end{pmatrix},$$

$$X^T DX = diag(d_{V_1}, \cdots, d_{V_K}),$$

$$X^T LX = \begin{pmatrix} \sum_{k=2}^{K} cut(V_1, V_k) & -cut(V_1, V_2) & \cdots & -cut(V_1, V_K) \\ -cut(V_2, V_1) & \sum_{k=1,k\neq 2}^{K} cut(V_2, V_k) & \cdots & -cut(V_2, V_K) \\ \vdots & \vdots & \ddots & \vdots \\ -cut(V_K, V_1) & -cut(V_K, V_2) & \cdots & \sum_{k=1}^{K-1} cut(V_K, V_k) \end{pmatrix},$$

where W is the weight matrix, D is the weighted degree matrix, and L is the Laplacian matrix.

This proposition can be proved by the facts from Proposition 2.1. Three different cuts of graph partitioning for clustering are presented based on this proposition.

Minimum cut, Ratio Cut and Normalized Cut

Definition 2.5. *For the graph partitioning* $\Pi = (V_1, \cdots, V_K)$*, the* minimum cut *is the sum of weights of the edges with ends in distinct subsets, i.e.,*

$$Mcut = \sum_{k=1}^{K} \sum_{k'=k+1}^{K} cut(V_k, V_{k'}).$$

The Ratio cut *of the graph partitioning* $\Pi = (V_1, \cdots, V_K)$ *is a fraction of the cuts to the sizes of the corresponding subsets, and it is defined as*

$$Rcut = \sum_{k=1}^{K} \sum_{k'=1, k' \neq k}^{K} \left(\frac{cut(V_k, V_{k'})}{|V_k|} \right).$$

The Normalized cut *of the graph partitioning* $\Pi = (V_1, \cdots, V_K)$ *is a fraction of the cuts to the weighted degrees of vertices, and it is defined as*

$$Ncut = \sum_{k=1}^{K} \sum_{k'=1, k' \neq k}^{K} \left(\frac{cut(V_k, V_{k'})}{d_{V_k}} \right).$$

Usually, the term "**graph partitioning problem (GPP)**" refers to finding the partitioning among all possible partitions Π of G with minimum cut. The Ratio cut [56] and Normalized cut [115] were introduced for clustering.

If W is the data matrix to perform clustering and w_{ij} measures the similarity or closeness of the ith and jth objects, we can define the intersimilarity and intrasimilarity of clustering by the graph partitioning $\Pi = (V_1, \cdots, V_K)$ in the following definition.

Intersimilarity and Intrasimilarity

Definition 2.6. *Assume that the partition matrix* X *is performed for clustering of* N *objects with similarity matrix* W*, the* intersimilarity *of clusters by the graph partitioning* $\Pi = (V_1, \cdots, V_K)$ *is the sum of similarities of the objects from different clusters and it can be denoted by*

$$\frac{1}{2} tr(X^T L X).$$

The intrasimilarity *of clusters is the sum of similarities of the objects from the same clusters and it can be denoted by*

$$\frac{1}{2}tr(X^T WX).$$

Here the notation tr denotes the trace of the matrix. As W is a similarity matrix, the graph partitioning problem of minimizing minimum cut finds the best partition such that objects in different groups (subsets) have as least similarity as possible. This is the requirement of clustering. However, as shown in Figure 2-1, the minimum cut always lead to "unnatural bias" [115] for partitioning out small subsets of vertices. This is the reason that Ratio cut and Normalized cut are introduced for clustering. Ratio cut is to balance the cardinalities of each subsets or clusters, while Normalized cut tries to balance the weights among the subsets or clusters. The numerators of fractions in both Ratio cut and Normalized cut measure the similarities among different subsets, and these two cuts should be minimized in graph partitioning for clustering. Besides Ratio cut and Normalized cut, some other cuts are also introduced for clustering, such as min-max cut [35], ICA (Isoperimetric co-clustering) cut [104].

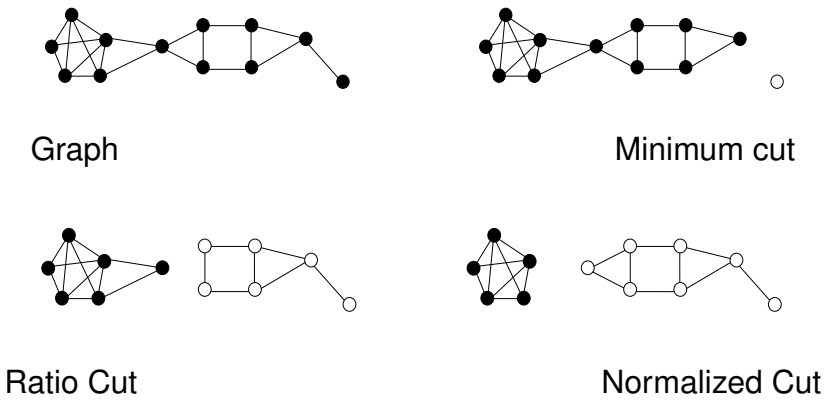


Figure 2-1. Comparisons of Minimum, Ratio and Normalized Cuts

By Definition 2.5 and Definition 2.6, the minimum cut of graph partitioning $\Pi = (V_1, \cdots, V_K)$, and intersimilarity for clustering of W based on the same partition matrix X, are equal to each other, i.e.,

$$Mcut = \frac{1}{2} tr(X^T L X).$$

In clustering, we want to minimize the intersimilarity among these K clusters. Correspondingly, we have the graph partitioning problem to minimize the minimum cut and the formulation is

$$\min_{X \in \mathcal{F}_K} \quad \frac{1}{2} tr(X^T L X). \tag{2–2}$$

Remark 4. *To minimize intersimilarity* $\frac{1}{2} tr(X^T L X)$ *is equivalent to maximize intrasimilarity* $\frac{1}{2} tr(X^T W X)$*. This is obtained from the fact* $tr(X^T L X) + tr(X^T W X) = tr(X^T D X) = \sum_{k=1}^{K} d_{V_k} = \sum_{i=1}^{N} d_i = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}$*, a fixed value for a given matrix* W*. Thus both* $\min \frac{1}{2} tr(X^T L X)$ *and* $\max \frac{1}{2} tr(X^T W X)$ *can be used for graph partitioning by minimum cut in Chapter 4.*

Since a bipartite graph is a special case of a general graph, the bipartite graph partitioning problem can be easily transformed into graph partitioning problem. For biclustering, the bipartite graph partitioning by minimum cut, Ratio cut and Normalized cut is used. For conceptions and notations of biclustering, bipartite graph partitioning, partition matrix, expression matrix, cuts, and etc, we refer to Section 2.1.2, Section 3.5, Section 4.1 and Section 4.2.3.

CHAPTER 3
MULTI-WAY CLUSTERING AND BICLUSTERING BY THE RATIO CUT AND NORMALIZED CUT

In this chapter, we consider the multi-way clustering problem based on graph partitioning models by the Ratio cut and Normalized cut. We formulate the problem using new quadratic models. Spectral relaxations, new semidefinite programming relaxations and linearization techniques are used to solve these problems. It has been shown that our proposed methods can obtain improved solutions. We also adapt our proposed techniques to the bipartite graph partitioning problem for biclustering.

This chapter is organized as follows: In Section 3.1, We present the optimization models for graph partitioning with Ratio cut and Normalized cut; Section 3.2 is a brief review of the spectral relaxation approaches; In Section 3.3, we present the semidefinite programming approaches; Section 3.4 includes the quadratically constrained programming approaches with linearization techniques; In Section 3.5, we discuss the bipartite graph partitioning and biclustering; In Section 3.6, we present the numerical experiments of all proposed algorithms; Section 3.7 concludes the chapter.

3.1 Clustering and Graph Partitioning

Following notations and preliminary results in Section 2.3, we are concentrating on the Ratio cut and Normalized cut of graph partitioning for clustering in this chapter. The graph partitioning with Minimum cut will be discussed in Chapter 4.

Lemma 3.1. *Defining a matrix Y based on partition matrix X as*

$$Y = X \cdot diag(1/\sqrt{n_1}, \cdots, 1/\sqrt{n_K}),$$

where $n_k = \sum_{i=1}^{N} x_{ik}$ for $k = 1, \cdots, K$, the Ratio cut can be expressed in the form of Y and Laplacian matrix L as

$$Rcut = tr(Y^T L Y).$$

Proof. Assume that $P = diag(1/\sqrt{n_1}, \cdots, 1/\sqrt{n_K})$. From the definition of Y, we have the following relations

$$tr(Y^T LY) = tr((XP)^T L(XP)) = tr(P(X^T LX)P).$$

From the Proposition 2.2, we have that

$$tr(Y^T LY) = \sum_{k=1}^{K} \frac{(X^T LX)_{kk}}{n_k} = \sum_{k=1}^{K} \sum_{k'=1, k' \neq k}^{K} \frac{cut(V_k, V_{k'})}{|V_k|},$$

which finishes the proof by the Definition 2.5 for the Ratio cut.

□

This lemma was presented in [21]. Here the notation $(X^T LX)_{kk}$ denotes the entry in the kth row and kth column of the matrix $X^T LX$. Thus, the problem of minimizing Ratio cut of graph partitioning for clustering can be expressed in the matrix form as follows:

$$\begin{aligned} \min \quad & tr(Y^T LY) \\ s.t. \quad & Y = XP, \quad X \in \mathcal{F}_K, \\ & P = diag(\frac{1}{\sqrt{(X^T X)_{11}}}, \cdots, \frac{1}{\sqrt{(X^T X)_{KK}}}). \end{aligned} \tag{3–1}$$

By solving this problem, we can find the graph partitioning $\Pi^* = (V_1^*, \cdots, V_K^*)$ with smallest $Rcut$ by the optimal solution X^*. The assignment of vertex v_i to subset V_k is decided by x_{ik}^*. However, we can use the matrix $Y = (y_{ik})_{N \times K}$ directly to partition the vertex set. If $y_{ik} = 0$, the vertex v_i does not belong to subset V_k; If $y_{ik} > 0$, the vertex v_i belongs to subset V_k. The matrix Y should satisfy some specific constraints, and we present them in the following theorem.

Theorem 3.1. *Y and X are equivalent for graph partitioning with respect to Ratio cut by the following formulation based on Y:*

$$\min \quad tr(Y^T LY) \tag{3–2}$$

$$s.t. \quad Y^T Y = I, \tag{3–3}$$

$$YSe = e, \tag{3–4}$$

$$C_{min} \leq SY^T e \leq C_{max}, \tag{3–5}$$

$$S = diag(s_1, \cdots, s_K),\ Y = (y_{ik})_{N\times K}, \tag{3–6}$$

$$y_{ik} \in \{0, s_k^{-1}\},\ s_k > 0, \tag{3–7}$$

$$i = 1, \cdots, N,\ k = 1, \cdots, K,$$

where e *is a vector of all elements being* 1 *with conform dimension.*

Proof. As shown in Lemma 3.1, the objective (3–2) is to minimize the Ratio cut. We need to show that the constraints (3–4)-(3–7) will present a graph partitioning $\Pi = (V_1, \cdots, V_K)$ as defined in Definition 2.2.

From constraints (3–7), the kth column of Y can be either 0 or a positive number s_k^{-1}. The positiveness of y_{ik} decides whether vertex v_i belongs to subset V_k or not.

If $y_{ik} > 0$, the element $(YS)_{ik}$ in constraints (3–4),(3–5) is 1; Otherwise, if $y_{ik} = 0$, $(YS)_{ik} = 0$. This is obtained by the structures of Y and S in constraints (3–6),(3–7). Thus, YS is a partition matrix. The constraint (3–4) ensures that every row of YS has the sum 1, i.e., every vertex belongs to exactly one subset. The constraint (3–5) guarantees that every column sum of YS is in the range $[C_{min}, C_{max}]$, the size constraints of each subset.

Then, the subsets can be denoted by $V_k = \{v_i : y_{ik} > 0, i = 1, \cdots, N\}$. Assume that $n_k = |V_k|$. Since $Y^T Y = diag(n_1/s_1^2, \cdots, n_K/s_K^2)$, from the constraint (3–3), we have $n_k/s_k^2 = 1$ or $s_k = \sqrt{n_k}$ for all $k = 1, \cdots, K$.

Therefore, we have used the formulation (3–3)-(3–7) to obtain a graph partitioning. The reverse direction can be proved similarly.

□

Remark 5. *In [115], the Normalized association for bipartition of graphs is defined. To extend for multi-way or* K*-way partitioning, the Normalized association for graph*

partitioning Π *is defined as*

$$Nassoc = \sum_{k=1}^{K} \frac{cut(V_k, V_k)}{d_{V_k}}.$$

By Proposition 2.2 and Normalized cut defined in Definition 2.5 for the same partitioning Π *and* X*, we have the relation*

$$2Nassoc + Ncut = K,$$

which is a pre-given parameter. Thus, we can also maximize Normalized association in stead of minimizing Normalized cut, as that for minimum cut.

For Normalized cut of graph partitioning, we have similar results as Ratio cut to define Z in the following lemma. Assume that I is an identity matrix.

Lemma 3.2. *Let the matrix* W' *be* $W' = D^{-1/2}WD^{-1/2}$ *and the matrix* L' *be* $L' = I - W'$*. Defining a matrix* Z *based on partition matrix* X *as*

$$Z = D^{1/2}X \cdot diag(1/\sqrt{d_{V_1}}, \cdots, 1/\sqrt{d_{V_K}}),$$

where $d_{V_k}(k = 1, \cdots, K)$ *is defined in Proposition 2.1, the Normalized cut can be expressed in the form of* Z *and Laplacian matrix* L *as*

$$Ncut = tr(Z^T L' Z).$$

Proof. Assume that $Q = diag(1/\sqrt{d_{V_1}}, \cdots, 1/\sqrt{d_{V_K}})$. From the definition of Z, we have the relations

$$\begin{aligned} Z^T L' Z =& (D^{1/2}XQ)^T (I - D^{-1/2}WD^{-1/2})(D^{1/2}XQ) \\ =& QX^T D^{1/2}(I - D^{-1/2}WD^{-1/2})(D^{1/2}XQ) \\ =& Q(X^T(D - W)X)Q \\ =& Q(X^T LX)Q. \end{aligned}$$

From the Proposition 2.2, we have that

$$tr(Z^T L' Z) = \sum_{k=1}^{K} \frac{(X^T L X)_{kk}}{d_{V_k}} = \sum_{k=1}^{K} \sum_{k'=1, k' \neq k}^{K} \frac{cut(V_k, V_{k'})}{d_{V_k}},$$

which finishes the proof by Definition 2.5 for the Normalized cut.

□

This lemma was presented in [55]. Thus, the program of minimizing Normalized cut of graph partitioning for clustering can be expressed in the matrix form as follows:

$$\begin{aligned} \min \quad & tr(Z^T L' Z) && \text{(3–8)} \\ s.t. \quad & Z = D^{1/2} X Q, \quad X \in \mathcal{F}_K, \\ & Q = diag(\frac{1}{\sqrt{(X^T D X)_{11}}}, \cdots, \frac{1}{\sqrt{(X^T D X)_{KK}}}). \end{aligned}$$

Theorem 3.2. *Z and X are equivalent for graph partitioning with respect to Normalized cut by the following program based on Z:*

$$\begin{aligned} \min \quad & tr(Z^T L' Z) && \text{(3–9)} \\ s.t. \quad & Z^T Z = I, && \text{(3–10)} \\ & ZTe = diag(D^{1/2}), && \text{(3–11)} \\ & C_{min} \leq T Z^T D^{-1/2} e \leq C_{max}, && \text{(3–12)} \\ & T = diag(t_1, \cdots, t_K), && \text{(3–13)} \\ & (D^{-1/2} Z)_{ik} \in \{0, t_k^{-1}\}, t_k > 0, && \text{(3–14)} \\ & i = 1, \cdots, N, k = 1, \cdots, K. \end{aligned}$$

This theorem was presented in [131] and can be proved using the methods in Theorem 3.1. The subset V_k can be denoted by $V_k = \{v_i : z_{ik} > 0, i = 1, \cdots, N\}$. The constraint (3–11) ensures that each vertex can belong to exactly one subset and constraint (3–12) guarantees that each subset has the size in the range $[C_{min}, C_{max}]$. The matrix $D^{-1/2} Z T$ can be considered as a partition matrix.

3.2 Spectral Relaxation Approaches

3.2.1 Spectral Methods

As mentioned in Section 2.1.1, most graph partitioning based clustering methods use the spectral method. We will show that the spectral method of graph partitioning for clustering is a relaxation of our proposed optimization models in Section 3.1. This method is based on spectral graph theory [27] and the eigenvalues and eigenvectors of a matrix are used for partitioning. For multi-way clustering, the spectral methods have been studied in [21] and [55]. In this section, we briefly introduce this method and will compare it with other proposed methods in Section 3.6.

From Theorem 3.1, the decision matrix Y has the property $Y^TY = I$, and similarly, matrix Z in Theorem 3.2 has the property $Z^TZ = I$. In the following, we use spectral relaxations to solve the graph partitioning with Ratio cut and Normalized cut. Let $x_1, x_2, \cdots, x_k$ be the k rows of partition matrix X, i.e., $X = (x_1, x_2, \cdots, x_K)$. Similarly, $Y = (y_1, \cdots, y_K)$ and $Z = (z_1, \cdots, z_K)$.

Therefore, a relaxation form of the program in Theorem 3.1 to minimize Ratio cut is

$$\begin{aligned} \min \quad & tr(Y^TLY) \\ s.t. \quad & Y^TY = I. \end{aligned} \tag{3–15}$$

In addition, by the columns $y_1, \cdots, y_K$, the Ratio cut can be expressed as

$$Rcut = tr(Y^TLY) = \sum_{k=1}^{K} y_k^T L y_k,$$

and the constraints in (3–15) are $y_k^T y_k = 1$ for $k = 1, \cdots, K$.

The dimension of matrix Y^TLY is $K \times K$. By K. Fan's theorem [27], optimal solution to the program (3–15) is eigenvectors $\phi_1, \cdots, \phi_K$ such that $L\phi_k = \lambda_k\phi_k$ with a lower bound $\sum_{k=1}^{K} \lambda_k$ for objective function, i.e.,

$$\sum_{k=1}^{K} \lambda_k \leq \min_{Y\text{: Formulation (3–2)-(3–7)}} Rcut.$$

Roughly, we choose $y_k = \phi_k$ for $k = 1, \cdots, N$ as the solutions for Ratio cut of graph partitioning. Since ϕ_ks are eigenvectors of L, this method is called spectral relaxations.

Similarly, a relaxation form of the formulation in Theorem 3.2 to minimize Normalized cut is

$$\begin{aligned} \min \quad & tr(Z^T L' Z) \\ s.t. \quad & Z^T Z = I. \end{aligned} \tag{3–16}$$

For Normalized cut, we have the following property for the objective function in Theorem 3.2.

Lemma 3.3. *By the partition matrix $X = (x_1, x_2, \cdots, x_k)$, the Normalized cut can be expressed as*

$$Ncut = tr(Z^T L' Z) = \sum_{k=1}^{K} \frac{x_k^T L x_k}{x_k^T D x_k}.$$

Proof. By Lemma 3.2, the matrix L' is $I - W'$, where I can expressed as $D^{-1/2} D D^{-1/2}$ and $W' = D^{-1/2} W D^{-1/2}$. Thus, we have the relations

$$\begin{aligned} Z^T L' Z =& Z^T (I - W') Z = Z^T (D^{-1/2} D D^{-1/2} - D^{-1/2} W D^{-1/2}) Z \\ =& Z^T D^{-1/2} (D - W) D^{-1/2} Z = Z^T D^{-1/2} L D^{-1/2} Z \\ =& (D^{-1/2} Z)^T L (D^{-1/2} Z) = (XQ)^T L (XQ) \\ =& (X (X^T D X)^{-1/2})^T L (X (X^T D X)^{-1/2}) \\ =& ((X^T D X)^{-1/2}) X^T L X ((X^T D X)^{-1/2}), \end{aligned}$$

where $Q = (X^T D X)^{-1/2}$ is from Lemma 3.2. Then,

$$tr(Z^T L' Z) = tr((X^T D X)^{-1} (X^T L X)).$$

□

This result was presented in [131]. By K. Fan's theorem [27] and Lemma 3.3, the optimal solution to this program is eigenvectors $\varphi_1, \cdots, \varphi_K$ from the generalized

eigenvalue problem

$$L\varphi_k = \lambda'_k D\varphi_k,$$

with a lower bound $\sum_{k=1}^{K} \lambda'_k$ for objective function

$$\sum_{k=1}^{K} \lambda'_k \leq \min_{Z\text{: Formulation (3–9)-(3–14)}} Ncut.$$

Similarly, the eigenvectors φ_ks can be chosen as the solutions z_ks for Normalized cut of graph partitioning. Instead of solving this generalized eigenvalue problem, we can solve the eigenvalue problem $D^{-1/2}(D-W)D^{-1/2}\varphi_k = \lambda'_k\varphi_k$.

Algorithm **Spectral Relaxations**

Input: Data Matrix $W = (w_{ij})_{N\times N}$ and Positive Integer K

Output: Partition Matrix $X = (x_{ik})_{N\times K}$ and the Minimized Ratio cut $Rcut$ and Normalized cut $Ncut$ with respect to X

Step 1: Construct the Laplacian matrix L and the weighted degree matrix D from W as defined in Definition 2.3;

Step 2: For Ratio cut, solve the eigenvalue problem $L\phi_k = \lambda_k\phi_k$ to obtain Φ; For Normalized cut, solve the problem $D^{-1/2}(D-W)D^{-1/2}\varphi_k = \lambda'_k\varphi_k$ to obtain Ψ;

Step 3: Use the directional cosine method to obtain X_s from Φ, Ψ for Ratio cut and Normalized cut, respectively; (or use randomized projection heuristic method or clustering rounding to obtain X from Φ, Ψ directly)

Step 4: Obtain the partition matrix X from X_s and compute the corresponding Ratio cut and Normalized cut.

Assume that $\Phi = (\phi_1, \cdots, \phi_K)$ and $\Psi = (\varphi_1, \cdots, \varphi_K)$, where Φ is a relaxed solution for Y and Ψ is a relaxed solution for Z. After solving the eigenvalue problems, we obtain the relaxed solutions Φ and Ψ, which may not be feasible for the original partitioning problem. Next, we present three approaches for obtaining the feasible partition matrix X from Φ, Ψ: directional cosine method [21], randomized projection heuristic method [50]

and clustering rounding. We first present the algorithm for solving graph partitioning for clustering by spectral relaxations in above table.

3.2.2 Relaxed Solutions to Integer Ones

Assume that $X_s = XX^T$. Thus, the entry $(X_s)_{ij}$ denotes whether vertex v_i and v_j are in the same subset ($(X_s)_{ij} = 1$) or not ($(X_s)_{ij} = 0$) for all $i, j = 1, \cdots, N$. Once we have the matrix X_s, the corresponding partition matrix X can be constructed easily. Before presenting the approaches, we first state the following theorem.

Theorem 3.3. *Given the matrices Y, Z as defined in Theorem 3.1 and Theorem 3.2, we have the corresponding matrix X_s as*

$$X_s = D_y YY^T D_y, \tag{3–17}$$

$$X_s = D_z ZZ^T D_z, \tag{3–18}$$

where $D_y = diag(\cdots, \frac{1}{\sqrt{\sum_k y_{ik}^2}}, \cdots)$, $D_z = diag(\cdots, \frac{1}{\sqrt{\sum_k z_{ik}^2}}, \cdots)$, respectively.

Proof. We first prove (3–17). Assume that Y has N row vectors as $yr_1, \cdots, yr_N$. Since D_y is a diagonal matrix, the entry $(X_s)_{ij}$ can be expressed as $\frac{yr_i \cdot yr_j^T}{\sqrt{\sum_k y_{ik}^2 \sum_k y_{jk}^2}}$. As shown in Theorem 3.1, each row has only one none-zero elements. Assume that the ith row yr_i has the non-zero entry $y_{i,k}$ and jth row yr_j has the non-zero entry $y_{j,k'}$. If $k = k'$, we have $(X_s)_{ij} = 1$; otherwise, $(X_s)_{ij} = 0$. By the definition of X_s, we proved (3–17). The proof of Z to X_s is similar.

□

Directional cosine method [21]. Let the matrix Φ be $\Phi = (\phi_{ik})_{N \times K}$. Assume that $\tilde{X}_s$ is the approximate solution of X_s obtained from Φ, by Theorem 3.3, we have

$$\tilde{X}_s = diag(\cdots, \frac{1}{\sqrt{\sum_k \phi_{ik}^2}}, \cdots,) \cdot \Phi\Phi^T \cdot diag(\cdots, \frac{1}{\sqrt{\sum_k \phi_{ik}^2}}, \cdots,),$$

and thus, we have the entry

$$(\tilde{X}_s)_{ij} = \frac{\sum_k \phi_{ik}\phi_{jk}}{\sqrt{(\sum_k \phi_{ik}^2)(\sum_k \phi_{jk}^2)}}.$$

This entry is exactly the cosine of the angle between the ith row and jth row vectors of Φ. By the property of cosine function, $(\tilde{X}_s)_{ij} = 1$ if the two row vectors are in the same direction; $(\tilde{X}_s)_{ij} = 0$ if they are orthogonal to each other. From the approximation matrix $\tilde{X}_s$ to X, using another efficient clustering algorithm, such as K-means clustering on X_s, can fulfill it. In [21], a heuristic method was presented. This method first selects K vertices as the prototypes of the K subsets; the vertices with directional cosine within $\cos(\pi/8)$ of a prototype are added to that prototype's subset; and the remaining vertices are merged into subsets by computing hyperedge cuts from each left vertex and existing subsets.

For Normalized cut, the matrix is approximated by Ψ and the approximation of X_s can be obtained by the equation (3–18).

Randomized projection heuristic method [50]. This method was introduced in [50] for solving the max K-cut problem. Assume that the eigenvectors $\phi'_1, \cdots, \phi'_N$ are normalized row vectors of Φ. The steps for this method include: obtain the approximated solutions $\phi'_1, \cdots, \phi'_N$; choose K random vectors $r_1, \cdots, r_K$; obtain x_{ik} according to which $r_1, \cdots, r_K$ is closest to each ϕ'_i, i.e., $x_{ik} = 1$ if and only if $(\phi'_i)^T r_k \geq (\phi'_i)^T r_{k'}$ for all $k \neq k', k' = 1, \cdots, K$. The steps for Normalized cut are similar.

Clustering rounding. In addition, considering each row of $\tilde{\Phi}$ as a vertex, a clustering method, such as K-means clustering, can be used to obtain the partition matrix X instead of these two methods. This method was used in [131].

In this chapter, we consider the direct K-way partitioning for clustering. Many research papers [30, 115] are based on bipartition or bisection of graphs. The spectral method for bisection uses second smallest eigenvalue and the sign of its corresponding vector is used to partition the vertex set into two subsets. The recursive two-way partitioning is used to obtain multi-way partitioning. However, at each iteration of two-way partitioning, the judgement is needed to check whether further partitioning is

needed. The methods discussed in this chapter are obtaining multi-way partitioning directly.

3.3 Semidefinite Programming Relaxations

3.3.1 Semidefinite Programming Methods

Semidefinite programming (SDP) is a subfield of convex optimization concerned with the optimization of a linear objective function over the intersection of the cone of positive semidefinite matrices with an affine space. A linear primal SDP has the form

$$\begin{aligned} \min \quad & tr(CX) \\ s.t. \quad & tr(A_i X) = b_i, \quad i = 1, \cdots, m, \\ & X \succeq 0, \end{aligned}$$

where $X \succeq 0$ denotes that matrix X is positive semidefinite. In addition, $A \succeq B$ denotes that $A - B$ is positive semidefinite. In the following, We use $A \geq 0$ to denote that every entry of A is nonnegative.

As defined in Section 2.3, the partition matrix is $X \in \mathcal{F}_K$. Assume that $X_s = XX^T$, the objective of minimum cut $tr(X^T LX)$ can be replaced by $tr(LX_s)$. Here we consider several relaxations of X_s based on $\mathcal{F}_K$.

$$\begin{aligned} \mathcal{F}_K =& \{X : Xe = e, X^T e \in [C_{min}e, C_{max}e], x_{ik} \in \{0, 1\}\}, \\ \mathcal{T}_K =& \{X_s : \exists X_s \in \mathcal{F}_K \text{ such that } X_s = XX^T\}, \\ \mathcal{E} =& \{X_s : X_s = X_s^T, diag(X_s) = e, X_s e \in [C_{min}e, C_{max}e]\}, \\ \mathcal{N} =& \{X_s : X_s \geq 0\}, \\ \mathcal{P} =& \{X_s : X_s = X_s^T, X_s \succeq 0\}, \\ \mathcal{C}_K =& \{X_s : X_s = X_s^T, \sum_{i<j, ij \in I} (X_s)_{ij} \geq 1, \forall I \text{ with } |I| = K + 1\}, \\ \mathcal{D} =& \{X_s : X_s = X_s^T, (X_s)_{ij} + (X_s)_{ik} \leq 1 + (X_s)_{jk}, \forall \text{ triples } (i, j, k)\}. \end{aligned}$$

These sets are all reviewed in [70]. The set $\mathcal{T}_K$ is included in the sets $\mathcal{E}, \mathcal{N}, \mathcal{P}, \mathcal{C}_K, \mathcal{D}$. The convex hull of $\mathcal{T}_K$ is equivalent to $\mathcal{F}_K$ for partitioning [70]. The set $\mathcal{C}_K$ denotes the independent set conditions [70], which require that if $X_s \in \mathcal{T}_K$, the graph with adjacency matrix X_s has no independent set of size $K+1$. The set $\mathcal{D}$ is the triangle constraints [70], which mean that if pairs $(v_i, v_j), (v_j, v_k)$ are in the same subset, the pair (v_i, v_k) is also in the subset. The problem of minimizing the minimum cut of graph partitioning has been studied by semidefinite programming based on these sets. These approaches are studied and reviewed in [70, 84, 129].

For the matrix $Y_s = YY^T$, it satisfies $\mathcal{T}_K, \mathcal{N}, \mathcal{P}$ by changing X, X_s to Y, Y_s in these sets, respectively. The $\mathcal{E}$ can be changed to

$$\mathcal{E}_y = \{Y_s : Y_s = Y_s^T, tr(Y_s) = K, Y_s e = e\}.$$

We are proving this result in the following theorem.

Theorem 3.4. *The matrix Y in formulation (3–2)-(3–7) and $Y_s = YY^T$ satisfy the the constraints*

$$tr(Y_s) = K, Y_s e = e.$$

Proof. As shown in Theorem 3.1, the matrix Y can be expressed as $Y = XP$, where P is given in (3–1). Thus, we have the relations:

$$\begin{aligned} Y_s =& YY^T = (XP)(XP)^T = XPPX^T \\ =& X \cdot diag(1/n_1, \cdots, 1/n_K) \cdot X^T \\ =& (x_{ik}/n_k)_{N\times K} X^T. \end{aligned}$$

Therefore,

$$tr(Y_s) = tr(XPPX^T) = \sum_{i=1}^{N}\sum_{k=1}^{K} \frac{x_{ik}}{n_k} x_{ik} = \sum_{k=1}^{K} \frac{\sum_{i=1}^{N} x_{ik}^2}{n_k} = \sum_{k=1}^{K} \frac{n_k}{n_k} = K,$$

and for the part $Y_s e$,

$$Y_s e = (x_{ik}/n_k)_{N\times K} X^T e = (x_{ik}/n_k)_{N\times K}(n_1, \cdots, n_K)^T = (\cdots, \sum_{k=1}^{K}(\frac{x_{ik}}{n_k}\cdot n_k), \cdots)^T = e.$$

□

In addition, the set $conv\{YY^T : Y^T Y = I\}$ is equal to the set

$$\mathcal{O}_K = conv\{YY^T : Y^T Y = I\} = \{Y_s : Y_s = Y_s^T, tr(Y_s) = K, 0 \preceq Y_s \preceq I\}.$$

This result was proved in [94]. For the matrix $Z_s = ZZ^T$, it satisfies $\mathcal{T}_K, \mathcal{N}, \mathcal{P}, \mathcal{O}_K$ and the $\mathcal{E}$ can be changed to

$$\mathcal{E}_z = \{Z_s : Z_s = Z_s^T, tr(Z_s) = K, Z_s diag(D^{1/2}) = diag(D^{1/2})\},$$

by the following theorem.

Theorem 3.5. *The matrix Z in formulation (3–9)-(3–14) and $Z_s = ZZ^T$ satisfy the the constraints*

$$tr(Z_s) = K, Z_s diag(D^{1/2}) = diag(D^{1/2}).$$

This theorem was presented in [131] and can be proved similarly as that in Theorem 3.4.

Remark 6. *Considering these relaxations for Ratio cut and Normalized cut, the size constraints, expressed in the matrix form $Y^T e, Z^T e$, are not included in any relaxed sets. As stated in Remark 3, the range $[C_{min}, C_{max}]$ of the size for each subset can be chosen loosely. In fact, although these constraints are not included in our following SDP relaxations, they are still in a range $[1, N-1]$, which means that no subset is empty. Otherwise, if the kth subset is empty, the size will be $|V_k| = 0$ and the weighted degree will be $d_{V_k} = 0$, which will cause the Ratio cut and Normalized cut to $+\infty$. Similarly, in the spectral relaxations, the range $[1, N-1]$ still satisfy for each subset. In the next section of quadratically constrained programs, if $[C_{min}, C_{max}]$ is chosen as $[1, N-1]$, the size constraints can be dropped from the programs without influencing the solutions.*

In the following, we use these relaxations to construct the semidefinite programming relaxations. The sets $\mathcal{C}_K$ and $\mathcal{D}$ do not work for Ratio cut and Normalized cut of graph partitioning. For minimum cut of graph equipartition, three relaxations are presented in [70], and they are the constraints $\mathcal{P} \cap \mathcal{E}$, $\mathcal{P} \cap \mathcal{E} \cap \mathcal{N}$ and $\mathcal{P} \cap \mathcal{E} \cap \mathcal{N} \cap \mathcal{D} \cap \mathcal{C}_K$. Based on these methods, we also consider the following relaxations for Ratio cut:

$$
\begin{aligned}
&(RSDP1) \quad \min\{tr(LY_s) : Y_s \in \mathcal{P} \cap \mathcal{E}_y\},\\
&(RSDP2) \quad \min\{tr(LY_s) : Y_s \in \mathcal{P} \cap \mathcal{E}_y \cap \mathcal{N}\},\\
&(RSDP3) \quad \min\{tr(LY_s) : Y_s \in \mathcal{P} \cap \mathcal{E}_y \cap \mathcal{N} \cap \mathcal{O}_K\},
\end{aligned}
$$

and for Normalized cut:

$$
\begin{aligned}
&(NSDP1) \quad \min\{tr(L'Z_s) : Z_s \in \mathcal{P} \cap \mathcal{E}_z\},\\
&(NSDP2) \quad \min\{tr(L'Z_s) : Z_s \in \mathcal{P} \cap \mathcal{E}_z \cap \mathcal{N}\},\\
&(NSDP3) \quad \min\{tr(L'Z_s) : Z_s \in \mathcal{P} \cap \mathcal{E}_z \cap \mathcal{N} \cap \mathcal{O}_K\}.
\end{aligned}
$$

The relaxation $(NSDP3)$ has been studied in [131]. The spectral relaxations can be considered as a special form of SDP relaxations which consider only $Y_s, Z_s \in \mathcal{O}_K$.

The most widely used algorithm for SDP is interior point methods. Other methods include Bundle method, augmented Lagrangian method. These methods can be found in [76, 125]. Many codes for solving SDP are available. In this chapter, we use the package `CVX`, a package for specifying and solving convex programs [52, 53].

The solutions Y_s, Z_s from these SDP relaxations may not be feasible for the original partitioning, and we have to transform them into feasible partition matrix Y, Z or X, X_s. In Section 3.2.2, we have discussed three approaches for this transformation. These methods can also be used.

Assume that $\tilde{Y}_s, \tilde{Z}_s$ are obtained from SDP relaxations, they are approximated solutions of Y_s, Z_s. By non-negative matrix factorization [82], $\tilde{Y}$ is obtained from $\tilde{Y}_s = \tilde{Y}\tilde{Y}^T$. The $\tilde{Z}'$ is also obtained from $\tilde{Z}_s = \tilde{Z}'\tilde{Z}'^T$ by SVD and $\tilde{Z} = D^{-1/2}\tilde{Z}'$. Then, the

directional cosine method, randomized projection heuristic or clustering rounding as discussed in Section 3.2.2 are used to obtain $\tilde{X}_s$ from $\tilde{Y}$, $\tilde{Z}$.

3.3.2 Algorithms

The algorithm for graph partitioning by Ratio cut and Normalized cut based SDP is presented in the following.

Algorithm **RSDP/NSDP**

Input: Data Matrix $W = (w_{ij})_{N\times N}$ and Positive Integer K

Output: Partition Matrix $X = (x_{ik})_{N\times K}$ and the Minimized Ratio cut $Rcut$ and Normalized cut $Ncut$ with respect to X

Step 1: Construct the Laplacian matrix L and weighted degree matrix D from W as defined in Definition 2.3;

Step 2: For Ratio cut, solve one of the SDP relaxations $RSDP1$, $RSDP2$, $RSDP3$ to obtain $\tilde{Y}_s$; For Normalized cut, solve one of the SDP relaxations $NSDP1$, $NSDP2$, $NSDP3$ to obtain $\tilde{Z}_s$;

Step 3: Use non-negative matrix factorization (NNMF) to obtain $\tilde{Y}$ by $\tilde{Y}_s = \tilde{Y}\tilde{Y}^T$, and $\tilde{Z}'$ from $\tilde{Z}_s = \tilde{Z}'\tilde{Z}'^T$ by $\tilde{Z} = D^{-1/2}\tilde{Z}'$;

Step 4: Use the directional cosine method, randomized projection heuristic method, or clustering rounding to obtain X_s or X from $\tilde{Y}$, $\tilde{Z}$ for Ratio cut and Normalized cut, respectively;

Step 5: Obtain the partition matrix X from X_s and compute the corresponding Ratio cut and Normalized cut;

Step 6: Compare these results to obtain the best solution.

3.4 Quadratically Constrained Programming Approaches

The formulation (3–1) or the formulation (3–2)-(3–7) for Ratio cut, and the formulation (3–8) or the formulation (3–9)-(3–14) are nonlinear discrete programming. The methods discussed above are all relaxations of these programs. In the following, we

reformulate them as binary quadratically constrained programs based on the decision variables in X.

By the definition of Ratio cut in Definition 2.5 and Proposition 2.2, we can reformulate (3–1) for minimizing Ratio cut of graph partitioning as follows:

$$\begin{aligned} \min \quad & \sum_{k=1}^{K} \sum_{k'=1, k' \neq k}^{K} y_{k,k'} \\ s.t. \quad & y_{k,k'} \Big(\sum_{i=1}^{N} x_{ik} \Big) - \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} x_{ik} x_{jk'} \geq 0, \\ & \qquad k = 1, \cdots, K, \quad k' = 1, \cdots, K, \quad k' \neq k, \\ & \sum_{k=1}^{K} x_{ik} = 1, \quad C_{min} \leq \sum_{i=1}^{N} x_{ik} \leq C_{max}, \\ & x_{ik} \in \{0, 1\}, \quad i = 1, \cdots, N, \quad k = 1, \cdots, K. \end{aligned} \tag{3–19}$$

Here, we assume that $y_{k,k'} = \frac{cut(V_k, v_{k'})}{|V_k|}$, and use the fact $|V_k| = \sum_{i=1}^{N} x_{ik}$ and $cut(V_k, V_{k'}) = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} x_{ik} x_{jk'}$. Similarly, assume that $z_{k,k'} = \frac{cut(V_k, V_{k'})}{d_{V_k}}$. By the definition of Normalized cut in Definition 2.5 and Proposition 2.2 with $d_{V_k} = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} x_{ik}$, we can reformulate (3–8) for minimizing Normalized cut of graph partitioning as follows:

$$\begin{aligned} \min \quad & \sum_{k=1}^{K} \sum_{k'=1, k' \neq k}^{K} z_{k,k'} \\ s.t. \quad & z_{k,k'} \Big(\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} x_{ik} \Big) - \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} x_{ik} x_{jk'} \geq 0, \\ & \qquad k = 1, \cdots, K, \quad k' = 1, \cdots, K, \quad k' \neq k, \\ & \sum_{k=1}^{K} x_{ik} = 1, \quad C_{min} \leq \sum_{i=1}^{N} x_{ik} \leq C_{max}, \\ & x_{ik} \in \{0, 1\}, \quad i = 1, \cdots, N, \quad k = 1, \cdots, K. \end{aligned} \tag{3–20}$$

Both programs (3–19), (3–20) are mixed integer quadratically constrained programs (MIQCP) with linear objective functions. However, the binary constraints $x_{ik} \in \{0, 1\}$ can

be replaced by $x_{ik}(x_{ik}-1) \geq 0$ and $x_{ik}(1-x_{ik}) \geq 0$. These programs become quadratically constrained programs (QCP). In the following, we use linearization techniques on both programs and obtain the equivalent binary integer linear programs.

Let the product $x_{ik}y_{k,k'}$ be $y_{i,k,k'}$, i.e., $y_{i,k,k'} = x_{ik}y_{k,k'}$. Since x_{ik} is binary, this product can be linearized [124] by the following constraints

$$\begin{cases} y_{i,k,k'} \geq y_{k,k'} - u \cdot (1 - x_{ik}), \\ y_{i,k,k'} \geq l \cdot x_{ik}, \\ y_{i,k,k'} \leq y_{k,k'} - l \cdot (1 - x_{ik}), \\ y_{i,k,k'} \leq u \cdot x_{ik}, \end{cases} \tag{3–21}$$

where $l < y_{k,k'}$, $u > y_{k,k'}$ and they can be chosen as sufficient small, sufficient large constants, respectively.

The product of the two binary variables $x_{ik}, x_{jk'}$ can be linearized by $x_{i,j,k,k'}$ using the well known techniques:

$$\begin{cases} x_{i,j,k,k'} \leq x_{ik}, \\ x_{i,j,k,k'} \leq x_{jk'}, \\ x_{i,j,k,k'} \geq x_{ik} + x_{jk'} - 1, \\ x_{i,j,k,k'} \geq 0. \end{cases} \tag{3–22}$$

By the linearization techniques in (3–21) and (3–22), we can reformulate (3–19) for Ratio cut in the following:

$$\begin{aligned} \min \quad & \sum_{k=1}^{K} \sum_{k'=1,k' \neq k}^{K} y_{k,k'} \\ s.t. \quad & \sum_{k=1}^{K} x_{ik} = 1, \quad C_{min} \leq \sum_{i=1}^{N} x_{ik} \leq C_{max}, \\ & \sum_{i=1}^{N} y_{i,k,k'} - \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} x_{i,j,k,k'} \geq 0, \end{aligned} \tag{3–23}$$

Constraints (3–21) and (3–22) ,

$$x_{ik} \in \{0,1\}, \quad i = 1, \cdots, N,$$

$$k = 1, \cdots, K, \quad k' = 1, \cdots, K, \quad k' \neq k.$$

This is a binary integer linear program, where x_{ik} is binary, $y_{k,k'}$, $x_{i,j,k,k'}$, $y_{i,k,k'}$ are (nonnegative) continuous variables. In fact, under these constraints, $x_{i,j,k,k'}$ is still binary.

For the formulation (3–20) for Normalized cut, we can introduce another continuous variables $z_{i,k,k'} = x_{ik} z_{k,k'}$ by the same linearization methods in (3–21). Thus, the formulation (3–20) can be reformulated as a binary integer linear program by $z_{i,k,k'}$, $x_{i,j,k,k'}$ as follows:

$$\min \quad \sum_{k=1}^{K} \sum_{k'=1,k'\neq k}^{K} z_{k,k'} \tag{3–24}$$

$$s.t. \quad \sum_{k=1}^{K} x_{ik} = 1, \quad C_{min} \leq \sum_{i=1}^{N} x_{ik} \leq C_{max},$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} z_{i,k,k'} - \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} x_{i,j,k,k'} \geq 0,$$

Constraints (3–21) (with changes from y to z) and (3–22) ,

$$x_{ik} \in \{0,1\}, \quad i = 1, \cdots, N,$$

$$k = 1, \cdots, K, \quad k' = 1, \cdots, K, \quad k' \neq k.$$

Both formulations (3–23) and (3–24) are binary integer linear programs. It can be decomposed into continuous linear programs and pure integer programs by Benders' decomposition method. The mixed binary integer linear program can also be solved by commercial software, such as CPLEX. The solution of this program is the exact solution for graph partitioning. However, for large scale problems, this method may take long computational time to reach the optimal solution. In practice, the anti-degeneracy constraints [42] are always added to reduce the computational time.

3.5 Biclustering and Bipartite Graph Models

Biclustering simultaneously groups the objects and features. More precisely, biclustering is to find a subset of objects and features satisfying that the objects in this subset are much more highly related to the features in this subset than features from another subset. Biclustering is always performed on a data matrix. Assume that $A = (a_{ij})_{N\times M}$ is the data matrix for biclustering with nonnegative entries, the ith row of this matrix corresponds to the ith object, the jth column corresponds to the jth feature and the entry a_{ij} measures the expression level feature j in object i. Biclustering is to find submatrices of A with specific constraints. In out bipartite graph partitioning models, we are finding K nonoverlap, exclusive, exhaustive submatrices [38] in A. The matrix A is the weight matrix in the bipartite graph model.

The bipartite graph is defined as $G = (V, U, E)$ with vertex sets $V = \{v_1, \cdots, v_N\}$, $U = \{u_1, \cdots, u_M\}$ and edge set $E = \{(v_i, u_j) : \text{ edge between vertices } v_i \text{ and } u_j, 1 \leq i \leq N, 1 \leq j \leq M\}$, where N and M are the numbers of vertices within two sets, respectively. Usually, instead of weight matrix, the biadjacency weight matrix $A = (a_{ij})_{N\times M}$ is given where $a_{i,j}$ is the weight of edge (v_i, u_j). Assume that we still want to obtain K nonempty subsets of both V and U. A partitioning by bipartite graph is $\Pi = (V_1 \cup U_1, \cdots, V_K \cup U_K)$ such that $V_k \cup U_k$ is a union subset and $V_1 \cup V_2 \cup \cdots \cup V_K = V$, $V_k \cap V_{k'} = \emptyset$, $U_1 \cup U_2 \cup \cdots \cup U_K = U$, $U_k \cap U_{k'} = \emptyset$ for all pairs $k \neq k'$. Assume that $n_k = |V_k|$, $m_k = |U_k|$ and $n = (n_1, \cdots, n_K)^T$, $m = (m_1, \cdots, m_K)^T$.

We define two partition matrices $X_v = (x^v_{ik})_{N\times K}$, $X_u = (x^u_{jk})_{M\times K}$ for vertex sets V, U, respectively. The requirements for bipartite graph partitioning is stated in the set

$$\begin{aligned}\mathcal{FB}_K =&\{X_v = (x^v_{ik})_{N\times K}, X_u = (x^u_{jk})_{M\times K} : X_v e = e, X_v^T e = n \in [C_{min}e, C_{max}e],\\ &X_u e = e, X_u^T e = m \in [c_{min}e, c_{max}e]\}.\end{aligned}$$

Let the matrix X be $X = \begin{pmatrix} X_v \\ X_u \end{pmatrix}$. From matrix A, we can construct the corresponding weight matrix $W = \begin{pmatrix} \mathbf{0} & A \\ A^T & \mathbf{0} \end{pmatrix}$, the weighted degree matrix $D_v = diag(Ae)$, $D_u = diag(A^T e)$, and the Laplacian matrix $L = \begin{pmatrix} D_v & -A \\ -A^T & D_u \end{pmatrix}$. The objective of the bipartite graph partitioning by minimum cut is

$$\begin{aligned}\frac{1}{2}tr(X^T LX) =& \frac{1}{2}tr\left(\begin{pmatrix} X_v \\ X_u \end{pmatrix}^T \begin{pmatrix} D_v & -A \\ -A^T & D_u \end{pmatrix} \begin{pmatrix} X_v \\ X_u \end{pmatrix}\right) \\ =& \frac{1}{2}tr(X_v^T D_v X_v + X_u^T D_u X_u - X_v^T A X_u - X_u^T A^T X_v) \\ =& e^T Ae - tr(X_v^T A X_u),\end{aligned}$$

where $tr(X_v^T D_v X_v + X_u D_u X_u) = 2e^T Ae$ can be obtained by the methods used in Proposition 2.2. Thus, as Remark 5, the objective of the bipartite graph partitioning by minimum cut can choose either $\min \frac{1}{2}tr(X^T LX)$ or $\max tr(X_v^T A X_u)$. The formulations for bipartite graph partitioning by Ratio cut and Normalized cut can be constructed similarly as those in (3–1),(3–8) by changing $\mathcal{F}_K$ to $\mathcal{FB}_K$. The partition matrix X has the property as follows:

$$X^T X = \begin{pmatrix} X_v \\ X_u \end{pmatrix}^T \begin{pmatrix} X_v \\ X_u \end{pmatrix} = \begin{pmatrix} n_1 + m_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & n_K + m_K \end{pmatrix},$$

and the matrices D_v, D_u,

$$X^T \begin{pmatrix} D_v & \mathbf{0} \\ \mathbf{0} & D_u \end{pmatrix} X = \begin{pmatrix} d_{V_1} + d_{U_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_{V_K} + d_{U_K} \end{pmatrix}.$$

Assume that $P = X^T X$ and $Q = X^T \begin{pmatrix} D_v & \mathbf{0} \\ \mathbf{0} & D_u \end{pmatrix} X$.

The minimum cut is the sum weight of edges among all distinct union subsets $V_k \cup U_k$. The Ratio cut is the fraction of the cut among distinct union subsets to the sizes of the union subset. The Normalized cut is the fraction of the cut to weighted degrees of vertices in the union subsets. The Ratio cut for bipartite graph partitioning is defined as

$$\begin{aligned} RBcut &= \sum_{k,k'=1,k'\neq k}^{K} \frac{cut(V_k \cup U_k, V_{k'} \cup U_{k'})}{|V_k \cup U_k|} \\ &= \sum_{k,k'=1,k'\neq k}^{K} \frac{cut(V_k, U_{k'}) + cut(V_{k'}, U_k)}{|V_k| + |U_k|} \\ &= \sum_{k,k'=1,k'\neq k}^{K} \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(a_{ij}x_{ik}^{v}x_{jk'}^{u} + a_{ij}x_{ik'}^{v}x_{jk}^{u})}{\sum_{i=1}^{N} x_{ik}^{v} + \sum_{j=1}^{M} x_{jk}^{u}}. \end{aligned} \tag{3–25}$$

The Normalized cut for bipartite graph partitioning is defined as

$$\begin{aligned} NBcut &= \sum_{k,k'=1,k'\neq k}^{K} \frac{cut(V_k \cup U_k, V_{k'} \cup U_{k'})}{d_{V_k} + d_{U_k}} \\ &= \sum_{k,k'=1,k'\neq k}^{K} \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(a_{ij}x_{ik}^{v}x_{jk'}^{u} + a_{ij}x_{ik'}^{v}x_{jk}^{u})}{\sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}x_{ik}^{v} + \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}x_{jk}^{u}}. \end{aligned} \tag{3–26}$$

The quadratically constrained programs for Ratio cut and Normalized cut for bipartite graph partitioning can be constructed according to these definitions of cuts. Considering the union $V_k \cup U_k$ as a subset of $V \cup U$, spectral and SDP relaxations approaches for graph partitioning can be used on bipartite graphs directly. These can be obtained by the properties of $X = \begin{pmatrix} X_v \\ X_u \end{pmatrix}$. Assume that $Y = XP, Z = XQ$, we still have that $RBcut = tr(Y^T L Y)$ and $NBcut = tr(Z^T L' Z)$. We can assume that the first N vertices belong to one vertex set and the remaining M vertices belong to another vertex set. In addition, the size constraints $[C_{min}, C_{max}]$ and $[c_{min}, c_{max}]$ are relaxed into nonempty constraints for each subset, and they can be eliminated without influencing the results.

3.6 Numerical Experiments

In Section 3-5, we have discussed three approaches for graph partitioning by Ratio cut and Normalized cut for clustering. The quadratically constrained program with linearization techniques is the exact method, which can produce exact solutions. However, as we mentioned, this method is computationally expensive. In the following, we use CPLEX to solve several cases with small number of vertices. We first use our proposed methods to check the results mentioned in Fig. 2-1. This graph has 12 vertices and 20 edges, and we want to partition the vertex set into $K = 2$ subsets. By our proposed methods, RSDP1, RSDP2 and RSDP3 can find the best partitioning as shown in Fig. 2-1 with Ratio cut 0.67 while the spectral cannot. For Normalized cut, only NSDP2 and NSDP3 can find best partitioning with minimum Normalized cut 0.22. The QCP approach can find both cuts with minimum values correctly.

In the following, we use Matlab to generate some data sets, and use our relaxations methods, including spectral and SDP relaxations, to obtain approximate results. In Table 3-1, we use QCP to study several cases with small sizes of vertices, and also list the relaxed solutions. The randomly generated weights of edges are either 0 or 1.

Table 3-1. Multi-way clustering results for case $N = 10$

	Ratio Cut (*Rcut*)					Normalized Cut (*Ncut*)				
K	Spectral	RSDP1	RSDP2	RSDP3	QCP	Spectral	NSDP1	NSDP2	NSDP3	QCP
2	6.40	3.75	3.33	4.44	3.33	0.97	0.89	0.73	0.73	0.73
3	10.33	10.60	7.83	9.00	7.83	2.09	2.29	1.59	1.59	1.59
4	18.00	15.00	14.00	12.83	12.83	3.27	3.45	2.92	2.90	2.74
5	21.50	22.33	18.33	18.33	18.33	4.63	4.28	3.92	3.92	3.92

Note: In this table, we present all proposed approaches on graphs with 10 vertices.

From Table 3-1, the QCP approach produces the minimum Ratio cut and Normalized cut. However, as pointed in Section 3.5, this method is computationally expensive. The relaxation methods, NSDP2 and NSDP3 are quite useful for Normalized cut. The RSDP2 and RSDP3 can also work well in most cases.

Table 3-2. Multi-way clustering results for several cases

		Ratio Cut ($Rcut$)				Normalized Cut ($Ncut$)			
N	K	Spectral	RSDP1	RSDP2	RSDP3	Spectral	NSDP1	NSDP2	NSDP3
10	2	5.9651	4.4825	4.4825	4.4825	1.0434	1.0272	0.8955	0.8955
	3	9.0250	9.5688	7.9086	7.9086	2.2872	2.3642	1.9535	1.9535
	4	12.0233	12.1239	10.2214	10.1016	3.5399	3.4026	2.8099	2.9347
	5	22.6727	19.7710	16.7583	18.0279	4.3988	4.6825	4.0243	4.0243
20	2	10.0096	7.6014	7.6014	7.6014	1.0114	1.0871	0.8693	0.8693
	3	20.8108	18.1293	16.5634	17.1563	2.1834	1.9554	1.8213	1.8213
	4	23.6633	29.0600	22.1933	21.2622	3.2677	3.0845	2.6772	2.6772
	5	40.5086	39.7117	36.4658	36.2824	4.3783	4.2096	3.7502	3.7502
30	2	14.8708	16.0350	12.5969	12.5969	0.9484	0.9614	0.8798	0.8798
	3	30.9707	24.6805	24.3050	24.0552	2.1386	2.0779	1.8429	1.8429
	4	44.8108	41.1532	34.4266	35.6052	3.1751	3.0974	2.7678	2.9321
	5	59.1340	63.6217	48.1469	51.8282	4.1201	4.0619	3.7735	3.9155
40	2	20.8233	20.1543	15.6696	15.6696	1.0375	0.9249	0.9060	0.9060
	3	46.6597	41.0729	35.5308	36.2432	2.0809	2.0752	1.8634	1.8599
	4	62.2816	54.8081	51.3613	51.3613	3.1413	2.9849	2.9180	2.8006
	5	84.9918	78.1221	71.5009	73.9993	4.0904	4.1019	3.7695	3.7840
50	2	26.1892	23.3340	21.5288	21.5288	1.0530	1.0239	0.9314	0.9327
	3	49.4853	44.8301	45.1659	41.8663	2.1446	1.9844	1.8452	1.8508
	4	77.3965	75.8846	67.0981	67.6938	3.0348	3.0345	2.8049	2.7914
	5	102.7687	95.7195	87.8784	88.6438	4.0851	4.0305	3.7782	3.8661
60	2	29.6128	28.0421	23.5610	23.5610	1.0081	1.0159	0.9253	0.9240
	3	61.6096	56.9782	53.6327	51.0342	2.0311	2.0338	1.8609	1.8609
	4	85.1127	89.2939	80.1775	80.6099	3.0885	3.0725	2.8195	2.8144
	5	115.4614	102.1540	102.0608	100.1119	4.0484	4.0364	3.8305	3.8458
70	2	35.0627	34.8492	32.2496	32.2496	1.0483	0.9362	0.9138	0.9138
	3	70.0739	65.4819	63.3387	64.2392	1.9915	2.0442	1.8633	1.8742
	4	107.2466	100.9737	91.8401	92.3531	3.0664	3.0263	2.8222	2.8222
	5	143.5539	134.9686	127.5178	126.1204	4.0637	4.0584	3.8014	3.8275
80	2	39.0321	38.4926	35.5522	37.2403	1.0411	0.9396	0.9290	0.9301
	3	79.8124	67.5808	69.0741	68.9926	2.0622	2.0080	1.8772	1.8766
	4	118.0029	114.5897	105.0521	104.8207	3.0781	3.0470	2.8201	2.8206
	5	160.8420	156.5950	140.6994	140.1869	4.0627	4.0479	3.8125	3.8195

Note: In Table 3-2, we perform our methods on the data sets with N vertices and partition them into K subsets. In these computational results, all data sets for weighted matrices are generated by Matlab within the range $[0, 1]$. We use nonnegative matrix factorization method to obtain $\tilde{Y}, \tilde{Z}$ from $\tilde{Y}_s, \tilde{Z}_s$, respectively, with 100 iterations at most. In addition, we use K-means clustering to transform infeasible solutions into feasible solutions.

Table 3-3. Multi-way clustering results for case $N = 100$

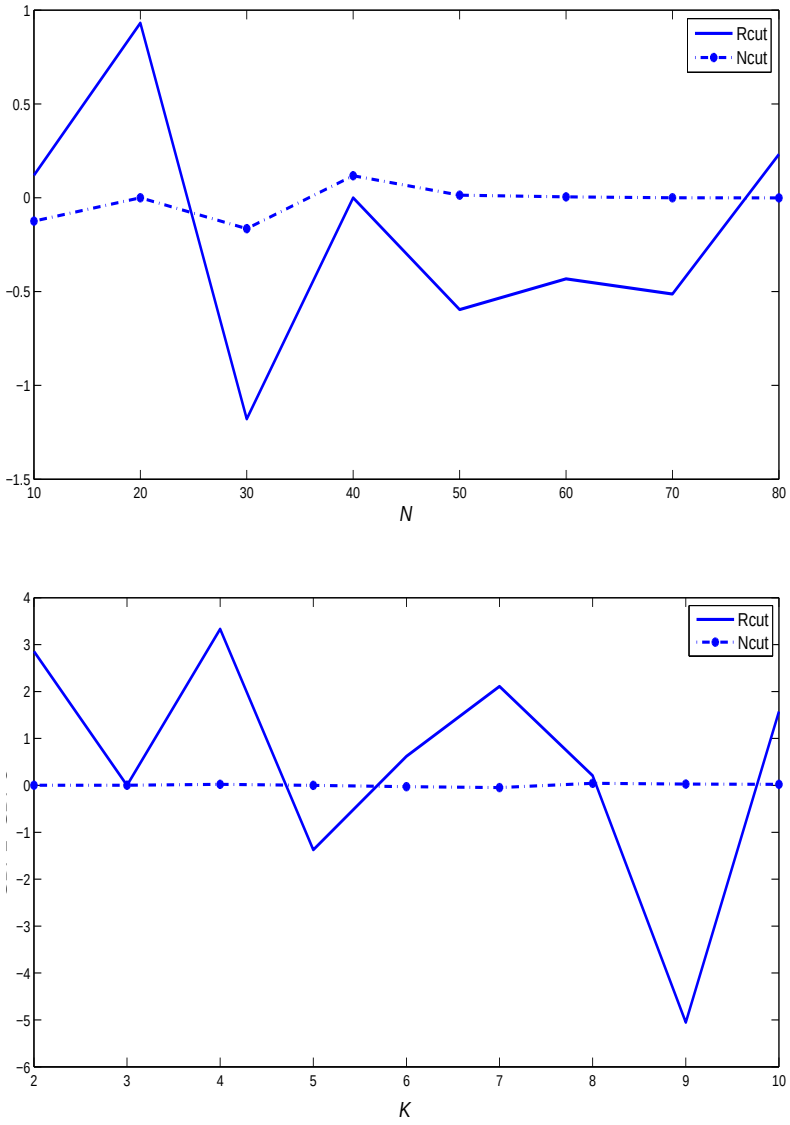
	Ratio Cut ($Rcut$)				Normalized Cut ($Ncut$)			
K	Spectral	RSDP1	RSDP2	RSDP3	Spectral	NSDP1	NSDP2	NSDP3
2	50.6998	47.0588	47.4435	44.5873	0.9843	0.9825	0.9394	0.9376
3	100.5659	102.7191	86.7741	86.7741	2.0243	2.0200	1.8884	1.8865
4	151.2269	143.7174	135.4610	132.1299	2.9901	3.0381	2.8760	2.8530
5	204.8972	194.8411	176.3037	177.6786	4.0602	4.0450	3.8282	3.8290
6	248.4688	251.0488	231.8624	231.2437	5.1010	5.0544	4.8003	4.8283
7	300.8281	286.0585	267.4994	265.3907	6.0430	6.0462	5.7658	5.8131
8	343.8683	338.2903	318.7011	318.4948	7.0807	7.0634	6.7779	6.7330
9	402.2044	377.6955	363.3538	368.4070	8.1043	8.0766	7.7986	7.7713
10	464.7659	421.7296	409.5492	407.9756	9.0762	9.1126	8.7692	8.7478

Note: This table has the same assumptions and use same methods as those in Table 3-2.

From the computational results, the relaxations RSDP1 or NSDP1 can obtain better solutions than spectral relaxations. The RSDP2 and RSDP3 for Ratio cut, and the NSDP2 and NSDP3 for Normalized cut, have the best solutions among these relaxations. However, the RSDP2 and RSDP3 have no clear comparisons. We put the difference of them in Fig. 3-1. From these two figures, NSDP2 and NSDP3 have no big difference for Normalized cut. In [131], the NSDP3 is used for clustering. However, as shown in our practical results, the NSDP2 has less constraints and can be chosen for graph partitioning of Normalized cut.

3.7 Discussion

In this chapter, we present the optimization models for graph partitioning by Ratio cut and Normalized cut. These are mixed integer nonlinear programs. To solve these problems, we reformulate them and use spectral relaxations and semidefinite programming relaxations. In addition, we use quadratically constrained programs to obtain the exact solutions. The proposed methods are quite useful in data clustering. In addition, we discuss three ways to obtain feasible integer solutions from infeasible relaxed solutions: the directional cosine method, randomized projection heuristic method, and clustering rounding. We also use our methods to obtain the bipartite graph partitioning for data biclustering. We have performed our algorithms on different data



Up: Results from Table 3-2 ($K = 4$); Bottom: Results from Table 3-3 ($N = 100$)

Figure 3-1. Differences for Ratio cut and Normalized cut solved by SDP2 and SDP3

sets. It has been shown that the second and the third SDP relaxations can obtain better solutions. The exact QCP models can obtain exact solutions for small data sets.

For quadratically constrained programs, efficient algorithms for large scale data sets are still under discussion. Possible applications of our methods include image segmentation, network analyzing, and text mining. Results of this chapter and Section 2.3 are published in our paper [43].

CHAPTER 4
GRAPH PARTITIONING WITH MINIMUM CUT

The graph partitioning problem is to partition the vertex set of a graph into a number of nonempty subsets so that the total weight of edges connecting distinct subsets is minimized. This refers to graph partitioning with Minimum cut as we discussed in Section 2.3. In this section, all research results are for GPP with Minimum cut. This problem has been studied for a long time, and it is an NP-complete combinatorial optimization problem.

Previous research requires the input of cardinalities of subsets or the number of subsets for equipartition. In Section 4.1, the problem is formulated as a zero-one quadratic programming problem without the input of cardinalities. We also present three equivalent zero-one linear integer programming reformulations. Because of its importance in data biclustering, the bipartite graph partitioning is also studied.

In Section 4.2, robust optimization models with two decomposition algorithms are introduced to solve the graph partitioning problem with interval uncertain weights of edges. The bipartite graph partitioning problem with edge uncertainty is also presented. Throughout this section, we make no assumption regarding the probability of the uncertain weights.

In Section 4.3, we introduce the two-stage stochastic graph partitioning problem and present the stochastic mixed integer programming formulation for this problem with finite explicit scenarios. For solving this problem, we present an equivalent integer linear programming formulation where some binary variables are relaxed to continuous ones. Additionally, for some specific graphs, we present a more simplified linear programming formulation. All formulations are tested on randomly generated graphs with different densities and different numbers of scenarios.

4.1 General Graph Partitioning Problem

Let $G = (V, E)$ be an undirected graph with a set of vertices $V = \{v_1, v_2, \cdots, v_N\}$ and a set of edges $E = \{(v_i, v_j) : \text{edge between vertices } v_i \text{ and } v_j, 1 \leq i, j \leq N\}$, where N is the number of vertices. The weights of the edges are given by a matrix $W = (w_{ij})_{N \times N}$, where $w_{ij}(> 0)$ denotes the weight of edge (v_i, v_j) and $w_{ij} = 0$ if no edge (v_i, v_j) exists between vertices v_i and v_j. This matrix is symmetric for undirected graphs G and is the adjacency matrix of G if $w_{ij} \in \{0, 1\}$.

Let K be the number of disjoint sets that we want to partition into, and $n_1, \cdots, n_K$ be the numbers of vertices within each sets. The general graph partitioning problem can be described as partitioning the vertex set V into K disjoint subsets so that the sum of weights of edges that connect vertices among different subsets is minimized. Methods for graph partitioning problem were reviewed in Section 2.2.

In this section, the cardinalities for each subsets are not required and the number $K(1 < K < N)$ of nonempty subsets is the only required information for possible equal or unequal partitioning. The quadratic and linear programming models are constructed to solve this general graph partitioning problem by algorithms based the exact formulations. Although we claim that the K should be considered as an input for general graph partitioning, we also present the methods for determination of K, and the given cardinalities $n_k(k = 1, \cdots, K)$ with possible changes of level $\hat{n}_k$.

In addition, the bipartite graph $G = (V, U, E)$, consists of two sets V, U of vertices with all edges in E between vertices from different vertex sets. The weights of this graph can be stored in a biadjacency weighted matrix. The bipartite graph partitioning is to partition both V and U into K subsets, respectively. In data mining, biclustering is to partition a data matrix in both rows and columns. In this section, the partitioning for bipartite graph is also formulated in quadratic and three linear programming models based on the results for general graphs.

Constraints and objectives. Based on the graph $G = (V, E)$ described above, let x_{ik} be the decision variables denoting that vertex v_i belongs to kth subset if $x_{ik} = 1$, otherwise $x_{ik} = 0$. Let X be the matrix $X = (x_{ik})_{N\times K}$ and the constraints for general graph partitioning problem are constructed as follows:

1) Exclusive constraints. Every vertex v_i can only belong to exactly one subset of V, and so every row sum of X is 1,

$$\mathcal{R} = \{x_{ik} \in R^{N\times K} : \sum_{k=1}^{K} x_{ik} = 1\}. \tag{4–1}$$

For $i = 1, \cdots, N$, all vertices belong to some subsets, and we have partitioned the vertex set into several subsets.

2) Cardinality (size) constraints. Let the cardinality of the kth subset be $n_k = \sum_{i=1}^{N} x_{ik}$. In order to eliminate the case that all vertices belong to only one subset, every subset should be nonempty ($\mathcal{S}_a$). Some partitioning results require the order of subsets ($\mathcal{S}_b$), and $MaxCard$ denotes the maximum size of each subset ($\mathcal{S}_c$). The constraints $\mathcal{S}_c$ can be considered as an equipartition if $MaxCard$ is chosen as an average cardinality of N/K. In the following models, we always choose the most general case $\mathcal{S}_a$ so that the graph can be partitioned according to its structure. Since we have relaxed the size constraints for graph partitioning, the term "general graph partitioning" is used in this section.

$$\mathcal{S}_a = \{x_{ik} \in R^{N\times K} : n_k \geq 1\}, \tag{4–2a}$$

$$\mathcal{S}_b = \{x_{ik} \in R^{N\times K} : n_1 \geq n_2 \geq \cdots \geq n_k \geq 1\}, \tag{4–2b}$$

$$\mathcal{S}_c = \{x_{ik} \in R^{N\times K} : n_k \leq MaxCard\}. \tag{4–2c}$$

3) Anti-degeneracy constraints. The subset indexes of the vertices can be restricted that the first vertex must belong to first subset, the second vertex must belong either to first or to second subset, and continually, the kth vertex must belong to one of the first k subsets.

$$\mathcal{D} = \{x_{ik} \in R^{N\times K} : x_{ik} = 0, \quad i = 1, \cdots K, \quad k = i + 1, \cdots, K\}. \tag{4–3}$$

With these constraints, the formulations are less degenerated because of the constraints on permutations of vertex assignment.

4) Binary constraints. Every x_{ik} indicates that vertex v_i belongs to kth subset or not.

$$\mathcal{B} = \{x_{ik} \in R^{N\times K} : x_{ik} = 0 \text{ or } 1\}. \tag{4–4}$$

5) Nonnegative constraints. The binary constraints can be relaxed to nonnegative ones which ensure the partitioning of G under some conditions.

$$\mathcal{P} = \{x_{ik} \in R^{N\times K} : x_{ik} \geq 0\}. \tag{4–5}$$

As discussed in Section 2.3, the feasible set (2–1) based on partition matrix for graph partitioning is

$$\mathcal{F}_K = \{(x_{ik})_{N\times K} : x_{ik} \in \{0,1\}, \sum_{k=1}^{K} x_{ik} = 1, C_{min} \leq \sum_{i=1}^{N} x_{ik} \leq C_{max}\},$$

where the cardinality constraint is $C_{min} \leq n_k = \sum_{i=1}^{N} x_{ik} \leq C_{max}$. We have following results for cases of $\mathcal{S}_a, \mathcal{S}_c$: When $C_{min} = 1, C_{max} = N-1$, $\mathcal{F}_K = \mathcal{R} \cap \mathcal{S}_a \cap \mathcal{B}$; When $C_{min} = 1, C_{max} = MaxCard$, we have $\mathcal{F}_K = \mathcal{R} \cap \mathcal{S}_a \cap \mathcal{S}_c \cap \mathcal{B}$.

As defined in Definition 2.5 and Definition 2.6, the Minimum cut is expressed $Mcut = \frac{1}{2}tr(X^T LX)$. As explained in Remark 4, two objects $\min \frac{1}{2}tr(X^T LX)$ and $\max \frac{1}{2}tr(X^T WX)$ are equivalent for graph partitioning by Minimum cut. In the following Section 4.1.1 and Section 4.1.2, both objectives are used for graph partitioning.

Determination of the number K of subsects. For controlling the cardinalities of subsets of V, the constraints $\mathcal{S}_a, \mathcal{S}_b, \mathcal{S}_c$ by notation $n_k = \sum_{i=1}^{N} x_{ik}$ are chosen under different situations. These constraints can be named nonempty constraints, order constraints and maximal cardinality constraints, respectively.

In previous research [57, 70], either $n_1, \cdots, n_K$ are given or the equal partition with given K. In this section, previous formulations require the input of K. However, the determination to input which $K \in \{2, 3, \cdots, N-1\}$ is still open. This section will present one method for determining K.

Let c_k be a binary variable such that $c_k = 1$ if the kth subset of V is nonempty, and $c_k = 0$ otherwise. Thus the cardinality constraints can be expressed as

$$\mathcal{S}_d = \{x_{ik} \in R^{N\times K} : n_k \geq c_k\}. \tag{4–6}$$

The term $\pm\sum_{k=1}^{K} c_k$ can be added to the objective function (add "+" when the original objective is to minimize, and "-" otherwise), in order to require the minimum possible number of subsets, by forcing some of the K clusters to be empty. Thus, K can be chosen as $N-1$ generally. When the constraints $\mathcal{S}_d$ is chosen to control the nonempty subsets, the maximal cardinality constraints $\mathcal{S}_c$ must be chosen at the same time, and nonempty constraints $\mathcal{S}_a$ cannot be chosen.

We have discussed constraints, objective functions and the determination of the number K for graph partitioning by Minimum cut in this section. The rest of Section 4.1 is organized as follows: In section 4.1.1, the quadratic programming model and its relaxations are studied, and in addition, the quadratic models for bipartite graph partitioning are also presented; In section 4.1.2, three linear programming approaches are introduced; In section 4.1.3, numerical experiments for many graphs under different conditions are performed after comparing these approaches; Section 4.1.4 concludes Section 4.1.

4.1.1 Quadratic Programming Approaches

From Remark 4, the objective function can be expressed in the form $\max \frac{1}{2} tr(X^T W X)$. As descriptions above, the constraints (4–1) and (4–4) are prerequisite for graph partitioning problem. In our general case, the constraints (4–2a) are chosen so that the requirement of the cardinality of each subset is not necessarily to be given.

Quadratic programming models and relaxations. The graph partitioning problem can be formulated as a zero-one quadratic program as follows:

$$\begin{aligned} \max \quad & \frac{1}{2} tr(X^T W X) \\ s.t. \quad & x_{ik} \in \mathcal{R} \cap \mathcal{S}_a, \\ & x_{ik} \in \mathcal{B}, \qquad i = 1, \cdots, N, k = 1, \cdots, K. \end{aligned} \tag{4–7}$$

The objective in (4–7) is not standard in a quadratic programming. Denoting $X = (x_{ij})_{N\times K} = (x_1, \cdots, x_k, \cdots, x_K)$, where x_ks are column vectors, $\hat{X} = (x_1^T, x_2^T, \cdots, x_K^T)^T$,

and

$$\hat{W} = \begin{pmatrix} W & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & W & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & W \end{pmatrix},$$

with dimension $(NK) \times (NK)$, where $\mathbf{0}$ is a matrix with all elements being 0 with proper dimension, the standard formulation for the objective in (4–7) is

$$\frac{1}{2} tr(X^T W X) = \frac{1}{2} \sum_{k=1}^{K} x_k^T W x_k = \frac{1}{2} \hat{X}^T \hat{W} \hat{X}. \tag{4–8}$$

Usually, the constrains $\mathcal{D}$ in (4–3) are added to reduce the degeneracy, and (4–2c) are added as a requirement of loose cardinalities, where $MaxCard$ can be chosen loosely (for example, $MaxCard = N/(K-1)$). Let $D = (d_{ij})_{N \times N}$ be a diagonal matrix. By properly choosing the elements d_{ii}, the binary constraints (4–4) can be relaxed to nonnegative ones (4–5) by the following Theorem 4.1 and for the proof details, we refer to [57]. Thus, the following continuous formulation can be obtained.

$$\begin{aligned} \max \quad & \frac{1}{2} tr(X^T (W + D) X) \\ s.t. \quad & x_{ik} \in \mathcal{R} \cap \mathcal{S}_a, \\ & x_{ik} \in \mathcal{P}, \qquad i = 1, \cdots, N, k = 1, \cdots, K. \end{aligned} \tag{4–9}$$

Theorem 4.1. *(Theorem 6.1, [57]) If D is chosen to satisfy $d_{ii} + d_{jj} \geq 2w_{ij}$ for each i and j, then the continuous problem (4–9) has a maximizer contained in $\mathcal{B}$, and hence, this maximizer is a solution of the discrete problem (4–7). Conversely, every solution to (4–7) is also a solution to (4–9). Moreover, if $d_{ii} + d_{jj} > 2w_{ij}$ for each i and j, then every local maximizer for (4–9) lies in $\mathcal{B}$.*

There are many approaches for solving (4–7), including heuristic and exact optimization ones as reviewed in the paper [114] by Sherali and Smith. The heursitc approaches include the rank-two relaxation by Burer et al. [15], the evolutionary

procedure by Lodi et al. [86] and the Tabu Search algorithm by Kochenberger et al. [78]. The exact optimization approaches include the branch-and-bound algorithm by Pardalos and Rodgers [97], and the Lagrangian decomposition algorithm by Chardaire and Sutter [23]. The most recent survey including heuristic and exact optimization approaches can be found for a similar quadratic assignment problem in [87] by Loiola et al. In Section 4.1, CPLEX 11.0 is used to solve the zero-one quadratic programming problem, and CPLEX uses multiple types of most recent algorithms [28].

Quadratic approaches for bipartite graphs. Let $G = (V, U, E)$ be a bipartite graph with vertex sets $V = \{v_1, \cdots, v_N\}$, $U = \{u_1, \cdots, u_M\}$ and edge set $E = \{(v_i, u_j) :$ edge between vertices v_i and u_j , $1 \leq i \leq N, 1 \leq j \leq M\}$, where N and M are the numbers of vertices within two sets, respectively. Usually, instead of weighted matrix, the biadjacency weighted matrix $A = (a_{ij})_{N\times M}$ is given where $a_{i,j}$ is the weight of edge (v_i, u_j), and its corresponding weighted matrix W for G can be constructed as

$$W = \begin{pmatrix} \mathbf{0} & A \\ A^T & \mathbf{0} \end{pmatrix}.$$

By the weighted matrix W and considering $V \cup U$ as the vertex set, the previous formulations can be used. However, some problems (e.g., biclustering, [39]) based on bipartite graph models require partitioning of both vertex set V and U into K disjoint subsets so that the kth subsets of V and also of U as a whole subset. The sum of weights of edges among the union subsets has to be minimized.

Denoting $X = \begin{pmatrix} X_v \\ X_u \end{pmatrix}$, where $X_v = (x^v_{ik})_{N\times K}$ and $X_u = (x^u_{jk})_{M\times K}$ are indicators for vertices from V and U, respectively, the equivalent objective can be expressed as

$$\max \frac{1}{2} tr(X^T W X) = \max tr(X_v^T A X_u). \tag{4–10}$$

The constraints for X_v are still the ones (4–1), (4–2a) and (4–4), while the index should be changed to $j = 1, \cdots, M$ instead of $i = 1, \cdots, N$ for X_u. Let $\mathcal{R}', \mathcal{S}'_a$ and $\mathcal{B}'$ be

the constraints for X_u with changed indexes. The zero-one quadratic program can be formulated as

$$\begin{aligned} \max \quad & tr(X_v^T A X_u) \\ s.t. \quad & x_{ik}^v \in \mathcal{R} \cap \mathcal{S}_a \cap \mathcal{B}, \\ & x_{jk}^u \in \mathcal{R}' \cap \mathcal{S}_a' \cap \mathcal{B}', \\ & \qquad i = 1, \cdots, N, j = 1, \cdots, M, k = 1, \cdots, K. \end{aligned} \tag{4–11}$$

Theorem 4.1 still holds in this case by considering X_v and X_u as submatrices of X, and W is also symmetric. A continuous case of program (4–11) can be obtained by changing $\mathcal{B}, \mathcal{B}'$ to $\mathcal{P}, \mathcal{P}'$ by constructing a diagonal matrix D of dimension $(N+M) \times (N+M)$. In addition, the anti-degeneracy constraints $\mathcal{D}$ can only be put on one of X_v and X_u for bipartite graphs.

4.1.2 Linear Programming Approaches

Direct linear programming models. The objective function

$$\frac{1}{2} tr(X^T W X) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} w_{ij} x_{ik} x_{jk}$$

is nonlinear. Here, two approaches, similarly as presented in [14], are used to transform this nonlinear objective into a linear one.

Let y_{ijk} denote indicator that edge (i, j) with two ends i, j belong to subset k if $y_{ijk} = 1$ or not if $y_{ijk} = 0$. Thus $y_{ijk} = x_{ik} x_{jk}, i, j = 1, \cdots, N, k = 1, \cdots, K$. The linearization is as follows

$$y_{ijk} \leq x_{ik}, y_{ijk} \leq x_{jk}, y_{ijk} \geq x_{ik} + x_{jk} - 1 \text{ and } y_{ijk} \geq 0.$$

Since y_{ijk} is in the objective to maximize and the nonnegativity of w_{ij}, constraints $y_{ijk} \geq x_{ik} + x_{jk} - 1$ and $y_{ijk} \geq 0$ can be eliminated for $w_{ij} \geq 0$. The linear programming

formation for general graph partitioning problem is

$$\begin{aligned} \max \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K} w_{ij}y_{ijk} \\ s.t. \quad & x_{ik} \in \mathcal{R} \cap \mathcal{S}_a \cap \mathcal{B}, \\ & \begin{cases} y_{ijk} \leq x_{ik}, \\ y_{ijk} \leq x_{jk}, \end{cases} \\ & i,j = 1,\cdots,N, k = 1,\cdots,K. \end{aligned} \tag{4–12}$$

The linearized formulation (4–12) introduces K more variables and $2K$ more constraints for each edge comparing with the original one (4–7).

Let z_{ij} be a binary variable such that $z_{ij} = 1$ if the vertices v_i, v_j of edge (v_i, v_j) belongs to the same subset and 0 otherwise. Thus $z_{ij} = \sum_{k=1}^{K} x_{ik}x_{jk}$ and $z_{ij} = 1$ if and only if for some k such that $x_{ik} = x_{jk} = 1$ and all others $x_{ik'} = x_{jk'} = 0$. The linearization of

$$z_{ij} = \sum_{k=1}^{K} x_{ik}x_{jk} \tag{4–13}$$

can be formulated as $z_{ij} \leq 1 + x_{ik} - x_{jk}, z_{ij} \leq 1 - x_{ik} + x_{jk}$, for all i, j, k and all other requirements of this formulations are eliminated for the objective to be maximized. Therefore, we obtain another linear programming formulation as

$$\begin{aligned} \max \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}z_{ij} \\ s.t. \quad & x_{ik} \in \mathcal{R} \cap \mathcal{S}_a \cap \mathcal{B}, \\ & \begin{cases} z_{ij} \leq 1 + x_{ik} - x_{jk}, \\ z_{ij} \leq 1 - x_{ik} + x_{jk}, \end{cases} \\ & i,j = 1,\cdots,N, k = 1,\cdots,K. \end{aligned} \tag{4–14}$$

In practice, the anti-degeneracy constraints $\mathcal{D}$ and the cardinality constraints $\mathcal{S}_c$ can be added to both programs (4–12) and (4–14).

An equivalent linear programming approach. The Laplacian matrix for $G = (V, E)$ with W is defined as $L = diag(\sum_j w_{1j}, \cdots, \sum_j w_{Nj}) - W$. As described in Remark 4, the objective for partitioning the graph G can also be expressed as $\frac{1}{2} tr(X^T L X)$, and the equivalent quadratic programming formulation is

$$\begin{aligned} \min \quad & \frac{1}{2} tr(X^T L X) \\ s.t. \quad & x_{ik} \in \mathcal{R} \cap \mathcal{S}_a \cap \mathcal{B}, \\ & \quad i = 1, \cdots, N, k = 1, \cdots, K. \end{aligned} \tag{4–15}$$

By the methods proposed in [22] and [114], defining $S = (s_{ik})_{N \times K} = (s_1, \cdots, s_K)$, $T = (t_{ik})_{N \times K} = (t_1, \cdots, t_K)$, where s_k, t_k are column vectors, the equivalent linear programming formulation is

$$\begin{aligned} \min \quad & e^T S e \\ s.t. \quad & x_{ik} \in \mathcal{R} \cap \mathcal{S}_a \cap \mathcal{B}, \\ & L x_k - t_k - s_k + Ce = 0, \\ & t_k \leq 2C(e - x_k), \\ & t_{ik}, s_{ik} \in \mathcal{P}, \\ & \quad i = 1, \cdots, N, k = 1, \cdots, K. \end{aligned} \tag{4–16}$$

where the constant $C = 2 \max_{i=1}^{N} \sum_{j=1}^{N} w_{ij}$.

From the methods in [22], the formulation (4–16) is equivalent to (4–15) after subtracting a constant the objective of (4–16) by the following theorem.

Theorem 4.2. *Formulations* (4–15) *and* (4–16) *are equivalent.*

Proof. Let X^0 be an optimal solution of the formulation (4–15). We claim that there exist $T, S(t_{ik} \geq 0, s_{ik} \geq 0)$ for all i, k such that

$$Lx_k^0 - t_k - s_k + Ce = 0, \tag{4–17}$$

$$t_k^T x_k^0 = 0. \tag{4–18}$$

The constraints $t_k \leq 2C(e - x_k^0)$ in (4–16) is equivalent to (4–18) since x_{ik}^0 is binary and C is a positive constant. In the following, we use (4–18) to prove the equivalence.

Since $C = 2\max_{i=1}^{N} \sum_{j=1}^{N} w_{ij} = \|L\|_\infty$, $Lx_k + Ce \geq 0$, and there always exist $T, S(t_{ik} \geq 0, s_{ik} \geq 0)$ for all i, k such that both (4–17) and (4–18) hold. The claim is proved.

The variables S, T can be chosen as S^0, T^0 satisfying (4–17) and (4–18) so that $e^T Se$, the sum of s_{ik} for all i, k, is minimized. We claim that (X^0, S^0, T^0) is an optimal solution for the formulation (4–16).

Multiplying (4–17) by $(x_k^0)^T$, we have

$$(x_k^0)^T Lx_k^0 - (x_k^0)^T s_k + (x_k^0)^T Ce = 0, \tag{4–19}$$

by considering (4–18). Taking the sum over all $k = 1, \cdots, K$, and considering $tr(X^T LX) = \sum_{k=1}^{K} x_k^T Lx_k$, we obtain

$$tr((X^0)^T LX^0) - \sum_{k=1}^{K} (x_k^0)^T s_k^0 + \sum_{k=1}^{K} (x_k^0)^T Ce = 0. \tag{4–20}$$

In addition, $\sum_{k=1}^{K} (x_k^0)^T Ce = e^T Ce = C \cdot N$ is a constant since $x_{ik} \in \mathcal{R}$ in both (4–15) and (4–16). The equation

$$(x_k^0)^T s_k^0 = e^T s_k^0, \tag{4–21}$$

for $k = 1, \cdots, K$ can be proved by contradiction. Assume that for some i, we have $x_{ik}^0 = 0$ and $s_k^0 > 0$, where T^0, S^0 are chosen to minimized $e^T Se$. Defining the vectors $\tilde{t}_k, \tilde{s}_k$ and corresponding $\tilde{T}, \tilde{S}$ such that $\tilde{t}_{ik} = \tilde{t}_{ik}^0 + \tilde{s}_{ik}^0, \tilde{s}_{ik} = 0$ for $j \neq i, \tilde{t}_{jk} = t_{jk}, \tilde{s}_{jk} = s_{jk}$,

in this case, $(X^0, \tilde{T}, \tilde{S})$ satisfies (4–17) and (4–18), and $e^T\tilde{S}e < e^TSe$, a contradiction to the assumption that (X^0, S^0, T^0) is optimal. Thus $(x_k^0)^T s_k^0 = e^T s_k^0$ holds for all $k = 1, \cdots, K$ and $\sum_{k=1}^{K}(x_k^0)^T s_k^0 = e^T S^0 e$, which leads (4–20) to

$$tr((X^0)^T LX^0) = e^T S^0 e - C \cdot N.$$

We have finished the proof that an optimal solution X^0 to (4–15) is also an optimal solution to (4–16) by choosing S^0, T^0 to minimize e^TSe. On the other direction, the proof is similar. □

From Theorem 4.2, the objectives of (4–16) and (4–12), (4–15) have the relation as follows,

$$e^T S^0 e = tr((X^0)^T LX^0) + C \cdot N = -tr((X^0)^T WX^0) + e^T We + C \cdot N,$$

where X^0 is optimal for (4–15) and (X^0, S^0, T^0) is optimal for (4–16).

The programs obtained in (4–12), (4–14) and (4–16) are all binary linear programming. The most widely used method for this kind of programming is branch and bound algorithm besides heuristic approaches, and in CPLEX 11.0 [28], the combined heuristic and exact methods are used.

Linear approaches for bipartite graphs. In equation (4–10), the objective for partitioning bipartite graph $G = (V, U, E)$ with weighted biadjacency matrix A is

$$\max tr(X_v^T AX_u) = \max \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{K} a_{ij}x_{ik}^v x_{jk}^u.$$

Similarly, defining $y_{ijk} = x_{ik}^v x_{jk}^u$, the linear programming formulation for partitioning bipartite graph is as follows,

$$\begin{aligned} \max \quad & \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{K} a_{ij} y_{ijk} \\ s.t. \quad & x_{ik}^v \in \mathcal{R} \cap \mathcal{S}_a \cap \mathcal{B}, \\ & x_{jk}^u \in \mathcal{R}' \cap \mathcal{S}_a' \cap \mathcal{B}', \\ & \begin{cases} y_{ijk} \le x_{ik}^v, \\ y_{ijk} \le x_{jk}^u, \end{cases} \\ & i = 1, \cdots N, j = 1, \cdots M, k = 1, \cdots, K. \end{aligned} \tag{4–22}$$

This formulation (4–22) introduces K more variables and $2K$ more constraints for each edge comparing with the original formulation (4–11). As in the formulation (4–14), defining $z_{ij} = \sum_{k=1}^{K} x_{ik}^v x_{jk}^u$, another linear programming formulation with less variables and constraints is obtained in the following.

$$\begin{aligned} \max \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij} z_{ij} \\ s.t. \quad & x_{ik}^v \in \mathcal{R} \cap \mathcal{S}_a \cap \mathcal{B}, \\ & x_{jk}^u \in \mathcal{R}' \cap \mathcal{S}_a' \cap \mathcal{B}', \\ & \begin{cases} z_{ij} \le 1 + x_{ik}^v - x_{jk}^u \\ z_{ij} \le 1 - x_{ik}^v + x_{jk}^u \end{cases} \\ & i = 1, \cdots N, j = 1, \cdots M, k = 1, \cdots, K. \end{aligned} \tag{4–23}$$

As in (4–16), defining $X_v = (x_{ik}^v)_{N\times K} = (x_1^v, \cdots, x_K^v)$, $X_u = (x_{jk}^u)_{M\times K} = (x_1^u, \cdots, x_K^u)$, $S_v = (s_{ik}^v)_{N\times K} = (s_1^v, \cdots, s_K^v)$, $S_u = (s_{jk}^u)_{M\times K} = (s_1^u, \cdots, s_K^u)$, $T_v = (t_{ik}^v)_{N\times K} = (t_1^v, \cdots, t_K^v)$, $T_u = (t_{jk}^u)_{M\times K} = (t_1^u, \cdots, t_K^u)$ and $D_v = diag(\cdots, \sum_{j=1}^{M} a_{ij}, \cdots)$, $D_u = diag(\cdots, \sum_{i=1}^{N} a_{ij}, \cdots)$, the linear programming formulation for bipartite graph $G =$

(V, U, E) with matrix A is

$$
\begin{aligned}
\min \quad & e^T S_v e + e^T S_u e \qquad (4\text{–}24)\\
s.t. \quad & x_{ik}^v \in \mathcal{R} \cap \mathcal{S}_a \cap \mathcal{B},\\
& x_{jk}^u \in \mathcal{R}' \cap \mathcal{S}_a' \cap \mathcal{B}',\\
& D_v x_k^v - A x_k^u - t_k^v - s_k^v + Ce = 0,\\
& D_u x_k^u - A^T x_k^v - t_k^u - s_k^u + Ce = 0,\\
& t_k^v \le 2C(e - x_k^v),\\
& t_k^u \le 2C(e - x_k^u),\\
& t_{ik}^v, s_{ik}^v \in \mathcal{P},\\
& t_{jk}^u, s_{jk}^u \in \mathcal{P}',\\
& \quad i = 1, \cdots, N, j = 1, \cdots, M, k = 1, \cdots, K.
\end{aligned}
$$

where the constant $C = \max\{\max_i 2\sum_j a_{ij}, \max_j 2\sum_i a_{ij}\}$. The equivalence of (4–24) with original quadratic formulation for bipartite graph can be proved by similar methods in Theorem 4.2.

The constraints $\mathcal{D}$ can also be added to formulations (4–22), (4–23) and (4–24) to reduce the degeneracy on X_v.

4.1.3 Numerical Experiments

Comparisons of two approaches.

For the graph $G = (V, E)$ with N vertices and $|E|$ edges, the discrete quadratic programming (DQ) formulation(4–7), and the linear programming formulations L1 (4–12), L2 (4–14) and L3 (4–16) are compared in Table 1. The number for $\mathcal{R}$ is N while for $\mathcal{S}_a$ is K. For L3(4–16), the number of nonnegative constraints for S, T is not added.

From Table 4-1, the linear formulations introduce variables and constraints to eliminate quadratic variables in the objective, and the formulation (4–14) has less variables and constraints. The numbers of continuous variables and constraints in

Table 4-1. Comparisons of formulations for graph partitioning

Formulation	Objective	#0-1 decisions	#cont. variables	#constraints
DQ(4–7)	quadratic	$N \cdot K$	0	$N + K$
L1(4–12)	linear	$N \cdot K$	$\|E\| \cdot K$	$N + K + 2K \cdot \|E\|$
L2(4–14)	linear	$N \cdot K$	$\|E\|$	$N + K + 2K \cdot \|E\|$
L3(4–16)	linear	$N \cdot K$	$2N \cdot K$	$N + K + 2N \cdot K$

Table 4-2. Computational seconds for general graph partitioning

Graphs	Vertices N	Edges $\|E\|$	Subsets K	L1	L2	L3	DQ
Rand10	10	23	2	0.03	0.02	0.02	0.02
			3	2.01	0.27	0.03	0.08
			4	3.79	0.73	0.25	0.30
Rand20	20	98	2	4.9	0.24	0.06	0.03
Rand40	40	379	2	12.29	5.44	0.15	0.05
Rand100	100	2453	2	3456.69	2796.04	2.17	0.52

the formulations L1 and L2 are related to the number of edges, while the number in formulation L3 is related to the number of vertices. Generally, L1 and L2 can be used for spare weighted matrix W. The cases for bipartite graph partitioning can be analyzed similarly.

Computational results. In this section, all programs are implemented using CPLEX 11.0 [28] via ILOG Concert Technology 2.5, and all computations are performed on a SUN UltraSpace- III with a 900 MHz processor and 2.0 GB RAM. Computational times are reported in CPU seconds.

Graph $G = (V, E)$. In Table 4-2, discrete quadratic and three linear programming approaches are compared. The graphs are randomly generated in C++ with all weights being nonnegative. The gap is default in CPLEX and all approaches obtain the same optimal results. Under the cases of these graphs, the discrete quadratic formulation has the fast computation and the linear formulation (4–16) is the fastest one among three linear formulations though it has the most variables.

Bipartite graph $G = (V, U, E)$. For the case of bipartite graphs, the matrix A is also randomly generated with weights being nonnegative. The gap is default in CPLEX and all approaches obtain the same optimal results. Table 4-3 shows the time

Table 4-3. Computational seconds for bipartite graph partitioning

Graphs	vertices (N, M)	Edges $\|E\|$	K	BL1	BL2	BL3	BDQ
Rand10,10	(10,10)	48	2	0.4	0.22	0.22	0.11
			3	37	4.15	21.75	28.54
			4	373.87	33.51	371.53	1432.63
Rand5,15	(5,15)	40	2	0.38	0.16	0.21	0.10
			3	4.35	0.79	1.64	6.91
			5	0.02	0.05	0.03	0.01
Rand20,20	(20,20)	189	2	40.4	3.55	2.10	1.07

comparisons for discrete quadratic programming formulation BDQ(4–11) and three linear programming formulations, BL1(4–22), BL2(4–23) and BL3(4–24). The results show that the linear program (4–23), which has the fewest variables among three linear programs, is the most efficient one, and it is also better than quadratic formulations for these generated graphs.

4.1.4 Discussion

In this section, a zero-one quadratic programming formulation is used to model the general graph partitioning problem, and the continuous quadratic programming method is also used. Two linear programming approaches can be derived from the zero-one quadratic programming, and another linear approach with introducing some variables in a different way is also presented. We have implemented the algorithms of these formulations for different graphs or networks and compared the computational seconds.

Several formulations and their relaxations to continuous forms may work for different graphs, and choosing which of them to use for a given graph is still in discussion. In addition, for the computational complexity of large graphs, the parallel computing may be useful and algorithms should be designed in such occasions for graph partitioning.

Results of Section 4.1 are based on our paper [42].

4.2 Robust Optimization of Graph Partitioning Involving Interval Uncertainty

Let $G = (V, E)$ be an undirected graph with a set of vertices $V = \{v_1, v_2, \cdots, v_N\}$ and a set of edges $E = \{(v_i, v_j) : \text{edge between vertices } v_i \text{ and } v_j, 1 \leq i, j \leq N\}$, where N is the number of vertices. The weights of the edges are given by a matrix $W = (w_{ij})_{N\times N}$,

where $w_{ij}(>0)$ denotes the weight of edge (v_i, v_j) and $w_{ij} = 0$ if no edge (v_i, v_j) exists between vertices v_i and v_j. This matrix is symmetric for undirected graphs G and is the adjacency matrix of G if $w_{ij} \in \{0, 1\}$.

Assume K is the number of subsets that we want partition V into, and C_{min}, C_{max} are lower and upper bounds of the cardinality of each subset, respectively. Usually, K is chosen from $\{2, \cdots, N-1\}$ and C_{min}, C_{max} can be chosen roughly from $\{1, \cdots, N\}$ such that $C_{min} \leq C_{max}$.

Let x_{ik} be the indicator such that vertex v_i belongs to the kth subset if $x_{ik} = 1$ or not if $x_{ik} = 0$, and y_{ij} be the indicator such that the edge (v_i, v_j) with vertices v_i, v_j are in different subsets if $y_{ij} = 1$ and v_i, v_j in the same subset if $y_{ij} = 0$. Thus, the objective function of graph partitioning to minimize the sum of weights of edges connecting disjoint subsets can be expressed as $\min \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} y_{ij}$ or $\min \sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij} y_{ij}$ because of $w_{ij} = w_{ji}$ and $w_{ii} = 0$ for non-existence of loops. Each vertex v_i has to be partitioned into one and only one subset, i.e., $\sum_{k=1}^{K} x_{ik} = 1$, and the kth subset has the number of vertices in range $[C_{min}, C_{max}]$, i.e., $C_{min} \leq \sum_{i=1}^{N} x_{ik} \leq C_{max}$. The relation between x_{ik} and y_{ij} can be expressed as $y_{ij} = 1 - \sum_{k=1}^{K} x_{ik} x_{jk}$ (similar to (4–13)) and this can be linearized as $-y_{ij} - x_{ik} + x_{jk} \leq 0, -y_{ij} + x_{ik} - x_{jk} \leq 0$ for $k = 1, \cdots, K$ under the objective of minimization. Therefore, the feasible set of deterministic formulation of graph partitioning problem for a graph $G = (V, E)$ with weight matrix W is

$$\mathcal{X} = \left\{ (x_{ik}, y_{ij}) : \begin{array}{l} \sum_{k=1}^{K} x_{ik} = 1,\ C_{min} \leq \sum_{i=1}^{N} x_{ik} \leq C_{max}, \\ -y_{ij} - x_{ik} + x_{jk} \leq 0, \\ -y_{ij} + x_{ik} - x_{jk} \leq 0, \\ x_{ik} \in \{0, 1\},\ y_{ij} \in \{0, 1\}, \\ \quad i = 1, \cdots, N, j = i+1, \cdots, N, k = 1, \cdots, K \end{array} \right\}, \tag{4–25}$$

and the objective function is

$$\min_{(x_{ik},y_{ij})\in\mathcal{X}} \sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}y_{ij} \tag{4–26}$$

The *nominal graph partitioning problem* is to solve the program with the objective (4–26) and the constraints in (4–25) of X. This is a binary integer linear program, which is quite similar to the formulation (4–14). Methods for the nominal graph partitioning problem are reviewed in Section 2.2.

The previous optimization methods are all based on determinate data W and ignore the uncertainty. However, the weights of edges are not always constant and they are uncertain. For example, when analyzing the community structure in a social network [44], the relationship between two members is changing along the time and it is uncertain. Therefore, the graph partitioning problem with uncertain weights of edges should be considered. There are two methods to address data uncertainty in mathematical programming models: stochastic programming and robust optimization. The stochastic programming method always require the known probabilistic distributions of uncertain data, while robust optimization is to optimize against the worst cases by using a min-max objective [8]. Graph partitioning is a combinatorial optimization problem. In the past, robust version of many combinatorial problems have been studied, for example, the robust shortest path [92], the robust spanning tree [133] as well as many other problems in [80].

In this Section 4.2, we follow methods used in [8, 9], which allow some violations and produce the feasible solution with high probability. The uncertainty we address in this section is the interval uncertainty for weight matrix $W = (w_{ij})_{N\times N}$. Each entry w_{ij} is modeled as independent, symmetric and bounded random but unknown distribution variable $\tilde{w}_{ij}$ that takes values in $[w_{ij}-\hat{w}_{ij}, w_{ij}+\hat{w}_{ij}]$. Note that we require $w_{ij} = w_{ji}$ and thus $\hat{w}_{ij} = \hat{w}_{ji}$ for $i, j = 1, \cdots, N$. Assume $\hat{w}_{ij} \geq 0$, $w_{ij} \geq \hat{w}_{ij}$ and $w_{ii} = 0$ for all $i, j = 1, \cdots, N$.

In this section, robust formulations for graph partitioning with uncertain W are discussed and several algorithms will be proposed based on the propositions of formulations. In addition, the cases for bipartite graph partitioning are also studied.

The rest of Section 4.2 is organized as follows: Section 4.2.1 discusses the formulations for the robust graph partitioning problem; In Section 4.2.2, two decomposition methods for solving the robust graph partitioning problem by solving a series of nominal graph partitioning problem are constructed; In Section 4.2.3, the bipartite graph partitioning problem involving uncertainty is discussed; Section 4.2.4 includes the computational results and analysis of these approaches; Section 4.2.5 is the conclusion.

4.2.1 Graph Partitioning with Uncertain Weights

In this section, the robust optimization is to address the uncertainty of weight matrix W with $\tilde{w}_{ij} \in [w_{ij} - \hat{w}_{ij}, w_{ij} + \hat{w}_{ij}]$, where w_{ij} is nominal value of edge (v_i, v_j). Let J be the index set of W with uncertain changes, i.e., $J = \{(i,j) : \hat{w}_{ij} > 0, i = 1, \cdots, N, j = i+1, \cdots, N\}$, where we assume that $j > i$ since W is symmetric. Let Γ be a parameter, not necessarily integer, that takes values in the interval $[0, |J|]$. This parameter Γ is introduced in [8, 9] to adjust the robustness of the proposed method against the level of conservatism of the solution. The number of coefficients w_{ij} is allowed to change up to $\lfloor \Gamma \rfloor$ and another $w_{i_t j_t}$ changes by $(\Gamma - \lfloor \Gamma \rfloor)$. Thus formulation for the *robust graph partitioning problem* can be established as follows:

$$\min_{(x_{ik}, y_{ij}) \in \mathcal{X}} \left(\sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij} y_{ij} + \max_{\left\{ \begin{array}{l} S : S \subseteq J, |S| \leq \Gamma \\ (i_t, j_t) \in J \setminus S \end{array} \right\}} \left(\sum_{(i,j) \in S} \hat{w}_{ij} y_{ij} + (\Gamma - \lfloor \Gamma \rfloor) \hat{w}_{i_t j_t} y_{i_t j_t} \right) \right) \quad (4\text{–}27)$$

Since the value y_{ij} takes value from $\{0, 1\}$, $|y_{ij}|$ in the model [9] is reduced to y_{ij} here. Depending on the chosen of Γ, there are several cases: if $\Gamma = 0$, no changes are

allowed and the problem reduces to nominal problem (4–26); if Γ is chosen as an integer, the maximizing part in (4–27) is $\max_{\{S|S\subseteq J,|S|\leq\Gamma\}}\sum_{(i,j)\in S}\hat{w}_{ij}y_{ij}$; if $\Gamma = |J|$, the problem can solved by Soyster's method [116]. The index set J is equivalent to edge set E if all weights have uncertainty.

As shown in the following theorem, the problem (4–27) can be reformulated as an equivalent binary integer linear programming. The method used in this proof was first proposed in [9].

Theorem 4.3. *The formulation* (4–27) *is equivalent to the following linear programming formulation:*

$$\begin{aligned}
\min \quad & \sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}y_{ij} + \Gamma p_0 + \sum_{(i,j)\in J} p_{ij} && (4\text{–}28)\\
s.t. \quad & p_0 + p_{ij} - \hat{w}_{ij}y_{ij} \geq 0, \quad \forall (i,j)\in J\\
& p_{ij} \geq 0, \quad \forall (i,j)\in J\\
& p_0 \geq 0,\\
& (x_{ik}, y_{ij}) \in \mathcal{X}.
\end{aligned}$$

Proof. For given values $(y_{ij})_{i=1,\cdots,N,j=i+1,\cdots,N}$, the part

$$\left\{ \begin{array}{c} \max \\ S: S\subseteq J, |S|\leq\Gamma \\ (i_t,j_t)\in J\setminus S \end{array} \right\} \left(\sum_{(i,j)\in S} \hat{w}_{ij}y_{ij} + (\Gamma - \lfloor\Gamma\rfloor)\hat{w}_{i_t,j_t}y_{i_t,j_t} \right),$$

in (4–27) can be linearized by introducing z_{ij} for all $(i,j) \in J$ with the constraints $\sum_{(i,j)\in J} z_{ij} \leq \Gamma, 0 \leq z_{ij} \leq 1$, or equivalently, by the following formulation

$$\begin{aligned}
\max \quad & \sum_{(i,j)\in J} \hat{w}_{ij}y_{ij}z_{ij} && (4\text{–}29)\\
s.t. \quad & \sum_{(i,j)\in J} z_{ij} \leq \Gamma,\\
& 0 \leq z_{ij} \leq 1, \quad \forall (i,j)\in J.
\end{aligned}$$

The optimal solution of this formulation should have $\lfloor \Gamma \rfloor$ variables $z_{ij} = 1$ and one $z_{ij} = \Gamma - \lfloor \Gamma \rfloor$, which is equivalent to the optimal solution in the maximizing part in (4–27).

By strong duality, for given values $(y_{ij})_{i=1,\cdots,N, j=i+1,\cdots,N}$, the problem (4–29) is linear and its duality can be formulated as

$$\begin{aligned} \min \quad & \Gamma p_0 + \sum_{(i,j)\in J} p_{ij} \\ s.t. \quad & p_0 + p_{ij} - \hat{w}_{ij} y_{ij} \geq 0, \quad \forall (i,j) \in J \\ & p_{ij} \geq 0, \quad \forall (i,j) \in J \\ & p_0 \geq 0. \end{aligned}$$

Combing this formulation with (4–27), we obtain the equivalent formulation (4–28), which finishes the proof. □

Our algorithm (**MIP**) is based on solving the binary linear program (4–28) directly by CPLEX MIP solver [28]. There are $N \cdot K + \frac{N(N-1)}{2}$ binary decision variables with at most $\frac{N(N-1)}{2} + 1$ continuous variables in this formulation.

4.2.2 Decomposition Methods for Robust Graph Partitioning Problem

Benders decomposition to a nominal problem and a linear program. In the formulation (4–28) for the robust graph partitioning problem, the variables p_0, p_{ij} are continuous while x_{ik}, y_{ij} are binary. For the fixed $\bar{x}_{ik}, \bar{y}_{ij}$, the formulation (4–28) can be reformulated as follows:

$$\begin{aligned} \min \quad & \Gamma p_0 + \sum_{(i,j)\in J} p_{ij} + \sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij} \bar{y}_{ij} \\ s.t. \quad & p_0 + p_{ij} \geq \hat{w}_{ij} \bar{y}_{ij}, \quad \forall (i,j) \in J \\ & p_{ij} \geq 0, \quad \forall (i,j) \in J \\ & p_0 \geq 0. \end{aligned}$$

This is a linear program, and we can obtain its dual problem as follows:

$$\begin{aligned} \max \quad & \sum_{(i,j)\in J} \hat{w}_{ij}\bar{y}_{ij}z_{ij} + \sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}\bar{y}_{ij} \\ s.t. \quad & \sum_{(i,j)\in J} z_{ij} \le \Gamma, \\ & 0 \le z_{ij} \le 1, \quad (i,j) \in J. \end{aligned} \tag{4–30}$$

By Benders decomposition method [6], the program (4–30) presents the subproblem. By solving this subproblem for giving values of $\bar{x}_{ik}, \bar{y}_{ij}$ at the iteration l, we can obtain the values $\bar{z}_{ij}^{(l)}$ and construct the optimality cut

$$z \ge \sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}y_{ij} + \sum_{(i,j)\in J} \hat{w}_{ij}y_{ij}\bar{z}_{ij}^{(l)} \tag{4–31}$$

for the master problem. Therefore, the master problem for Benders decomposition method can be formulated as follows:

$$\begin{aligned} \min \quad & z \\ s.t. \quad & z \ge \sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}y_{ij} + \sum_{(i,j)\in J} \hat{w}_{ij}y_{ij}\bar{z}_{ij}^{(l)}, \quad l = 1, 2, \cdots, L \\ & (x_{ik}, y_{ij}) \in \mathcal{X}. \end{aligned} \tag{4–32}$$

The program (4–30) is always feasible and bounded for any feasible solutions $\bar{x}_{ik}, \bar{y}_{ij}$ from (4–32), and thus, the feasibility cut can be eliminated in the master problem (4–32). Observing (4–30) and (4–32), the master problem is a binary integer linear program with respect to x_{ik}, y_{ij} and the subproblem is a linear program with respect to z_{ij}. Thus, by Benders decomposition method, we decompose the mixed integer program for robust graph partitioning problem into a series of linear programs and mixed binary linear programs. Additionally, the subproblem (4–30) can be easily solved by a greedy algorithm: sorting the coefficients $\hat{w}_{ij}\bar{y}_{ij}$ of z_{ij} for $(i,j) \in J$ in the objective function in

decreasing order; assigning the first $\lfloor \Gamma \rfloor$ corresponding z_{ij}s to be 1 according to this order; and assigning the last z_{ij} to be $\Gamma - \lfloor \Gamma \rfloor$ and all others $z_{ij} = 0$.

Theorem 4.4. *Solving the master problem* (4–32) *at iteration* l *is equivalent to solve* l *nominal graph partitioning problems.*

Proof. At the iteration l of Benders decomposition algorithm, there are l added optimality cuts in the form of (5–7), and this cut is equivalent to

$$z \geq \sum_{(i,j)\in J} (w_{ij} + \hat{w}_{ij}\bar{z}_{ij}^{(l)})y_{ij} + \sum_{(i,j)\notin J} w_{ij}y_{ij}.$$

The right hand side of this cut is in fact the objective of a nominal graph partitioning problem with weights $w_{ij} + \hat{w}_{ij}\bar{z}_{ij}^{(l)}$ for edge $(i,j) \in J$ and w_{ij} for edge $(i,j) \notin J$. Therefore, solving the master problem at iteration l is equivalent to solve l nominal graph partitioning problem and then choose the one with maximum objective. □

The algorithm (**BD**) based on Benders decomposition method is presented in the following table.

Algorithm **BD**

Step 1: Initialization:

$\bar{x}_{ik}, \bar{y}_{ij} :=$ initial feasible solution in X for all i, j, k;

$LB := -\infty$, $UB := \infty$;

Step 2: While there is gap larger than ε between UB and LB, i.e., $UB - LB > \varepsilon$, do the following steps:

Step 2.1: Solve the subproblem (4–30) to obtain point $\bar{z}_{ij}$ for $(i,j) \in J$, and add cut $z \geq \sum_{i=1}^{N}\sum_{j=i+1}^{N} w_{ij}y_{ij} + \sum_{(i,j)\in J} \hat{w}_{ij}y_{ij}\bar{z}_{ij}$ to the master problem (4–32);

$UB := \min\{UB, \sum_{j=i+1}^{N} w_{ij}\bar{y}_{ij} + \sum_{(i,j)\in J} \hat{w}_{ij}\bar{y}_{ij}\bar{z}_{ij}\}$;

Step 2.2: Solve the master problem $\min\{z :$ added cuts, $x_{ik}, y_{ij} \in X\}$;

$LB := \bar{z}$, where $\bar{z}$ is the objective value of master problem;

Step 3: Output the optimal solution x_{ik}^*, y_{ij}^* for all i, j, k.

The Step 1 of this algorithm requires finding a feasible solution. Here, we present a simple method for finding initial solutions of $\bar{x}_{ik}$s: putting vertices $v_1, v_2, \cdots, v_{C_{min}}$ into the first subset, i.e., $\bar{x}_{11} = \bar{x}_{21} = \cdots = \bar{x}_{C_{min},1} = 1$; putting the vertices $v_{C_{min}+1}, v_{C_{min}+2} \cdots, v_{2C_{min}}$ into the second subset, i.e., $\bar{x}_{C_{min}+1,2} = \cdots = \bar{x}_{2C_{min},2} = 1$; repeating these steps until we have $\bar{x}_{K \cdot C_{min},K} = 1$; setting $\bar{x}_{(K \cdot C_{min}+1),K} = 1, \bar{x}_{(K \cdot C_{min}+2),K} = 1, \cdots, \bar{x}_{N,K} = 1$ and all other unassigned $\bar{x}_{ik}$s to be 0. The initial solution for $\bar{y}_{ij}$ can be obtained by $\bar{y}_{ij} = 1 - \sum_{k=1}^{K} \bar{x}_{ik}\bar{x}_{jk}$.

The Benders decomposition method can solve the robust graph partitioning problem by solving a series of nominal graph partitioning problem. However, for solving the master problem at iteration l, it is equivalent to solve l nominal graph partitioning problem. Although the Benders decomposition methods can converge in finite steps, say L, we need to solve $L(L+1)/2$ nominal graph partitioning problems totally. In next section, we present another decomposition method, which can take less computational time in some cases.

Algorithm based on the decomposition of one variable. For all $(i,j) \in J$, let e_l ($l = 1, \cdots, |J|$) be the corresponding value of $\hat{w}_{ij}$ in the increasing order. For example, $e_1 = \min_{(i,j)\in J} \hat{w}_{ij}$ and $e_{|J|} = \max_{(i,j)\in J} \hat{w}_{ij}$. Let $(i^l, j^l) \in J$ be the corresponding index of the lth minimum one, i.e., $\hat{w}_{(i^l,j^l)} = e_l$. In addition, we define $e_0 = 0$. Thus, $[e_0, e_1], [e_1, e_2], \cdots, [e_{|J|}, \infty)$ is a decomposition of $[0, \infty)$.

For $l = 0, 1, \cdots, |J|$, we define the program G^l as follows:

$$G^l = \Gamma e_l + \min_{(x_{ik}, y_{ij})\in\mathcal{X}} \left\{ \sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij} y_{ij} + \sum_{(i,j):\hat{w}_{ij} \geq e_{l+1}} (\hat{w}_{ij} - e_l) y_{ij} \right\}. \tag{4–33}$$

Totally, there are $|J|+1$ of G^ls. In the following theorem, we prove that the decomposition method based on p_0 can solve the program (4–28). The method in the proof was first proposed in [9].

Theorem 4.5. *Solving robust graph partitioning problem* (4–27) *is equivalent to solve* $|J|+1$ *problems* G^l*s in* (4–33) *for* $l = 0, 1, \cdots, |J|$.

Proof. From (4–28) in Theorem 4.3, the optimal solution $(x_{ik}^*, y_{ij}^*, p_0^*, p_{ij}^*)$ satisfies

$$p_{ij}^* = \max\{\hat{w}_{ij} y_{ij}^* - p_0^*, 0\},$$

and therefore, the objective function of (4–28) can be expressed as

$$\begin{aligned} &\min_{\{p_0 \geq 0, (x_{ik}, y_{ij}) \in \mathcal{X}\}} \Gamma p_0 + \sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij} y_{ij} + \sum_{(i,j) \in J} \max\{\hat{w}_{ij} y_{ij} - p_0, 0\} \\ = &\min_{\{p_0 \geq 0, (x_{ik}, y_{ij}) \in \mathcal{X}\}} \Gamma p_0 + \sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij} y_{ij} + \sum_{(i,j) \in J} y_{ij} \max\{\hat{w}_{ij} - p_0, 0\}, \end{aligned} \tag{4–34}$$

where the equality is obtained by the fact y_{ij} is binary in the feasible set $\mathcal{X}$.

By the decomposition $[0, e_1], [e_1, e_2], \cdots, [e_{|J|}, \infty)$ of $[0, \infty)$ for p_0, we have

$$\sum_{(i,j) \in J} y_{ij} \max\{\hat{w}_{ij} - p_0, 0\} = \begin{cases} \sum_{(i,j): \hat{w}_{ij} \geq e_l} (\hat{w}_{ij} - p_0) y_{ij}, & \text{if } p_0 \in [e_{l-1}, e_l], l = 1, \cdots, |J|; \\ 0, & \text{if } p_0 \in [e_{|J|}, \infty). \end{cases}$$

Thus, the optimal objective value of (4–28) is $\min_{l=1,\cdots,|J|,|J|+1}\{Z^l\}$, where

$$Z^l = \min_{\{p_0 \in [e_{l-1}, e_l], (x_{ik}, y_{ij}) \in \mathcal{X}\}} \left(\Gamma p_0 + \sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij} y_{ij} + \sum_{(i,j): \hat{w}_{ij} \geq e_l} (\hat{w}_{ij} - p_0) y_{ij} \right), \tag{4–35}$$

for $l = 1, \cdots, |J|$, and

$$Z^{|J|+1} = \min_{\{p_0 \geq e_{|J|}, (x_{ik}, y_{ij}) \in \mathcal{X}\}} \Gamma p_0 + \sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij} y_{ij}.$$

For $l = 1, \cdots, |J|$, since the objective function (4–35) is linear over the interval $p_0 \in [e_{l-1}, e_l]$, the optimal is either at the point $p_0 = e_{l-1}$ or $p_0 = e_l$. For $l = |J| + 1$, Z^l is obtained at the point $e_{|J|}$ since $\Gamma \geq 0$.

Thus, the optimal value $\min_{l=1,\cdots,|J|,|J|+1}\{Z^l\}$ with respect to p_0 is obtained among the points $p_0 = e_l$ for $l = 0, 1, \cdots, |J|$. Let G^l be the value at point $p_0 = e_l$ in (4–35), i.e.,

$$G^l = \Gamma e_l + \min_{(x_{ik}, y_{ij}) \in \mathcal{X}} \left\{ \sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij} y_{ij} + \sum_{(i,j): \hat{w}_{ij} \geq e_{l+1}} (\hat{w}_{ij} - e_l) y_{ij} \right\}.$$

We finish the proof. □

As shown in Theorem 4.5, $G^{|J|} = \Gamma e_{|J|} + \sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij} y_{ij}$ is the original nominal problem with an added constant. Our Algorithm (**DP0**) is based on this theorem.

Algorithm **DP0**
Step 1: For all $(i, j) \in J$, sort $\hat{w}_{ij}$ in increasing order to obtain $e_0, e_1, \cdots, e_{|J|}$;
Step 2: For $l = 0, 1, \cdots, \|J\|$, solving G^l in (4–33);
Step 3: Let $l^* = \arg\min_{l=0,1,\cdots,\|J\|} G^l$ and obtain the optimal solution $\{x_{ik}^*, y_{ij}^*\} = \{x_{ik}, y_{ij}\}^{l^*}$.

Algorithm (**DP0**) is based on the decomposition of $p_0 \in [0, \infty)$ and each subproblem G^l has the same computational complexity as the nominal graph partitioning problem. Since the nominal graph partitioning problem is NP-complete, from the decomposition algorithm (**DP0**), we can conclude that the robust graph partitioning problem is also NP-complete.

4.2.3 Bipartite Graph Partitioning Involving Uncertainty

The bipartite graph is defined as $G = (V, U, E)$ with vertex sets $V = \{v_1, \cdots, v_N\}$, $U = \{u_1, \cdots, u_M\}$ and edge set $E = \{(v_i, u_j) : \text{ edge between vertices } v_i \text{ and } u_j, 1 \leq i \leq N, 1 \leq j \leq M\}$, where N and M are the numbers of vertices within two sets, respectively. Usually, instead of weighted matrix, the biadjacency weighted matrix $A = (a_{ij})_{N \times M}$ is given where $a_{i,j}$ is the weight of edge (v_i, u_j). In [42, 43], the relations for partitioning between graphs and bipartite graphs have been presented. Assume we still want to obtain K subsets of both V and U, and the cardinality for subsets of V is in the range $[C_{min}, C_{max}]$ and the cardinality for subsets of U is in the range $[c_{min}, c_{max}]$. Let the

constraints of bipartite graph partitioning be a set as follows:

$$\mathcal{Y} = \left\{ (x_{ik}^v, x_{jk}^u, y_{ij}) : \begin{array}{l} \sum_{k=1}^K x_{ik}^v = 1, C_{min} \le \sum_{i=1}^N x_{ik}^v \le C_{max}, \\ \sum_{k=1}^K x_{jk}^u = 1, c_{min} \le \sum_{j=1}^M x_{jk}^u \le c_{max}, \\ -y_{ij} \mp x_{ik}^v \pm x_{jk}^u \le 0, \\ x_{ik}^v, x_{jk}^u, y_{ij} \in \{0,1\}, \\ \quad i = 1, \cdots, N, j = 1, \cdots, M, k = 1, \cdots, K \end{array} \right\},$$

where $x_{ik}^v, x_{jk}^u, y_{ij}$ are the indicators for vertex sets V, U, and edge set E as the same explanations in section 1.

The bipartite graph partitioning problem is formulated as

$$\min_{(x_{ik}^v, x_{jk}^u, y_{ij}) \in \mathcal{Y}} \sum_{i=1}^N \sum_{j=1}^M a_{ij} y_{ij}.$$

Because of its similarity to the graph partitioning problem as discussed in Section 4.2.1, we also consider the uncertain of matrix $\tilde{A}$ where $\tilde{a}_{ij}$ takes values in $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$ for $i = 1, \cdots, N, j = 1, \cdots, M$. The robust optimization for uncertain $\tilde{a}_{ij}$ is as

$$\min_{(x_{ik}^v, x_{jk}^u, y_{ij}) \in \mathcal{Y}} \quad \sum_{i=1}^N \sum_{j=1}^M a_{ij} y_{ij} + \max \left\{ \begin{array}{l} S : S \subseteq J, |S| \le \Gamma \\ (i_t, j_t) \in J \setminus S \end{array} \right\} \left(\sum_{(i,j) \in S} \hat{a}_{ij} y_{ij} + (\Gamma - \lfloor \Gamma \rfloor) \hat{a}_{i_t, j_t} y_{i_t, j_t} \right), \tag{4–36}$$

where $J = \{(i,j) : \hat{a}_{ij} > 0\}$ and $\Gamma \in [0, |J|]$.

As proved in Theorem 4.3, we can obtain the linear formulation as (4–28) for robust bipartite graph partitioning (4–36) similarly as follows:

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} a_{ij}y_{ij} + \Gamma p_0 + \sum_{(i,j)\in J} p_{ij} && (4\text{–}37)\\
s.t. \quad & p_0 + p_{ij} - \hat{a}_{ij}y_{ij} \geq 0, \quad (i,j) \in J\\
& p_{ij} \geq 0, \quad (i,j) \in J\\
& p_0 \geq 0,\\
& (x^v_{ik}, x^u_{jk}, y_{ij}) \in \mathcal{Y}.
\end{aligned}
$$

We omit other methods and algorithms here since the robust optimization for bipartite graph partitioning is quite similar to graph partitioning problems.

4.2.4 Numerical Experiments

In this section, all algorithms (**MIP, BD, DP0**) are implemented using CPLEX 11.0 [28] via ILOG Concert Technology 2.5, and all computations are performed on a SUN UltraSpace- III with a 900 MHz processor and 2.0 GB RAM. Computational times are reported in CPU seconds.

All tested graphs are randomly generated. The density r of a graph is the ratio of the number of edges and the number of possible edges. The uncertain values of $[w_{ij} - \hat{w}_{ij}, w_{ij} + \hat{w}_{ij}]$ are randomly generated. Here we assume $w_{ij} \in \{0, 1\}$ and $0 < \hat{w}_{ij}/w_{ij} < 1$ if $w_{ij} > 0$. In Table 4-4, we assume the cardinality of each subset is in the range $[C_{min}, C_{max}] = [1, N-1]$. The gap in CPLEX is set to be 0.1. All objective values and computational seconds are presented in Table 4-4. From this table, we can find that the algorithm (**DP0**) is the most efficient one, and the algorithm (**BD**) is least efficient. As discussed in Section 4.2.2, the algorithm (**BD**) needs to compute l nominal graph partitioning problems at iteration l, while the algorithm (**DP0**) computes $|J| + 1$ nominal graph partitioning problems totally. Thus, if the Benders decomposition method cannot converges quickly in small number of iterations, it usually takes longer time than the algorithm (**DP0**).

Instead of loose cardinalities, we assume the bounds C_{min}, C_{max} take value around N/K. All objective values and computational seconds for different graphs are presented in Table 4-5 with more conservative cardinality constraints. From this table, we can see in the case of same number of vertices, the computational times increase as the density increases. Comparing results in Table 4-4 and Table 4-4, in the same graph, the case with loose cardinality constraints takes shorter time than the one with conservative bounds.

As discussed in Section 4.2.1, the parameter Γ is introduced in [8, 9] to adjust the robustness of the proposed method against the level of conservatism of the solution. In Fig. 4-1, for the random generated graph with 20 vertices and density 0.4 with 76 uncertain edges, the relationship between the objective values and the values of Γ is presented for obtaining 4 subsets. From this figure, we can see that as the value of Γ increases for considering more uncertainties, the objective values are increasing as well. But when Γ is large enough, the objective value is a constant. Additionally, the values for the cases of conservative cardinalities are larger than corresponding cases of loose cardinalities.

4.2.5 Discussion

In the Section 4.2, we present three algorithms for the robust graph partitioning problem with interval uncertain weights. We first present the formulation for this problem and then give the equivalent mixed integer linear programming formulation. Two decomposition methods, including Benders decomposition method and decomposition on one variable, can solve the robust graph partitioning problem by solving a series of nominal graph partitioning problems. We compare these algorithms on randomly generated graphs with uncertain weights. In this section, a parameter Γ, introduced by [8, 9], is chosen to allow some gap between the optimal value of the exact formulation and the robust solutions. Additionally, we study the bipartite graph partitioning problem involving uncertain weights.

Table 4-4. Computational results and seconds for robust graph partitioning (1)

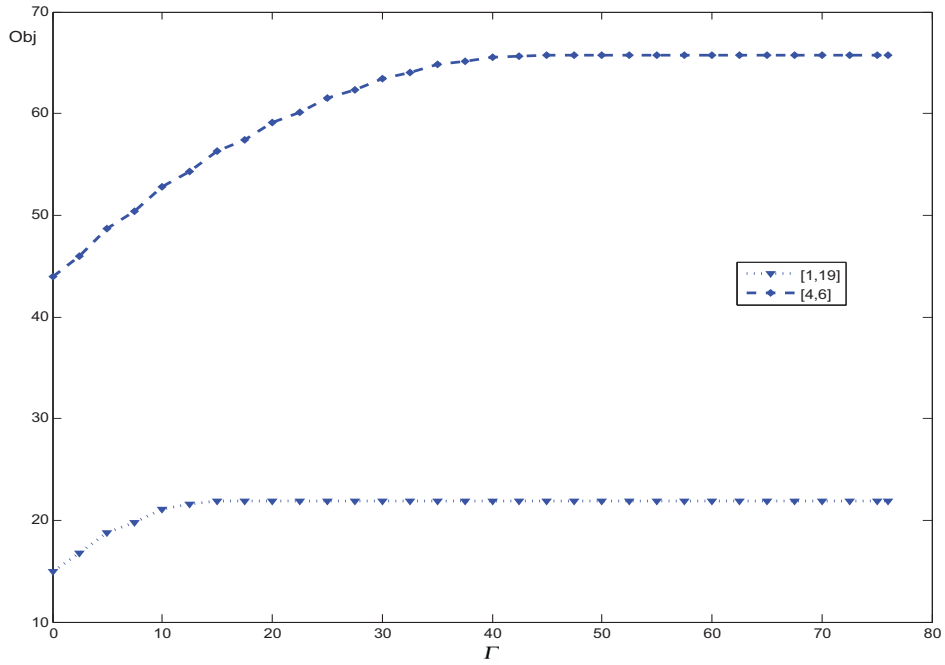
Graphs			Uncertainty		Objective Values			CPU Seconds		
N	r	K	$\|J\|$	Γ	MIP	BD	DP0	MIP	BD	DP0
10	0.1	3	4	2	0	0	0	0.01	0.01	0.01
	0.2		9	5	2.18	2.18	3.27	0.09	0.35	0.06
	0.3		13	7	1.03	1.03	1.67	0.10	0.23	0.05
	0.4		18	9	4.80	4.80	4.80	0.14	0.48	0.22
	0.5		22	11	7.68	7.68	8.59	0.39	0.65	0.20
	0.6		27	14	8.06	8.06	8.06	0.23	0.42	0.26
	0.7		30	15	11.58	11.58	11.58	0.73	1.21	0.37
	0.8		35	18	14.85	14.85	14.85	0.85	0.97	0.46
	0.9		40	20	18.08	18.03	18.29	0.89	4.13	0.28
15	0.1	3	10	5	0	0	0	0.01	0.03	0.01
	0.2		21	11	4.94	4.94	4.98	0.34	1.01	0.28
	0.3		31	16	4.55	4.55	4.55	0.75	0.56	0.39
	0.4		42	21	9.50	9.50	9.50	1.18	5.72	0.43
	0.5		52	26	11.86	11.78	11.86	2.16	2.20	0.60
20	0.1	3	19	10	0	0	0	0.01	0.04	0.03
	0.2		37	19	2.69	2.69	2.64	0.78	1.39	0.43
	0.3		57	29	6.58	6.58	6.58	1.11	1.56	0.70
	0.4		76	38	14.52	14.48	14.52	2.60	2.75	1.06
	0.5		95	48	15.65	15.65	16.25	3.25	11.59	1.48
20	0.1	4	19	10	0	0	0	0.01	0.06	0.02
	0.2		37	19	6.50	6.50	6.74	1.56	6.99	0.92
	0.3		57	29	12.01	12.01	13.11	3.32	9.96	1.54
	0.4		76	38	21.85	21.85	21.85	13.24	41.87	9.07
	0.5		95	48	25.65	25.05	25.65	11.05	21.04	4.74
30	0.1	3	42	21	1.04	1.04	1.46	0.86	0.51	0.07
	0.2		87	44	7.98	7.98	7.98	6.88	4.60	1.84
	0.3		130	65	10.64	10.64	10.64	4.34	7.43	2.39
30	0.1	4	42	21	2.19	2.19	2.29	2.40	6.09	1.30
	0.2		87	44	13.43	13.23	14.24	20.80	33.25	6.71
	0.3		130	65	16.44	16.44	16.44	12.60	20.47	7.49
40	0.1	3	78	39	3.41	3.41	3.41	5.07	3.79	1.07
	0.2		155	78	10.46	10.46	11.39	6.75	13.99	3.83
	0.3		233	117	20.64	20.64	22.00	58.36	134.24	35.57
40	0.1	4	78	39	5.46	5.46	6.89	3.83	11.83	2.49
	0.2		155	78	16.25	16.25	16.25	118.86	303.03	13.35
	0.3		233	117	31.94	31.61	32.23	229.26	>3000	259.06
50	0.1	3	122	61	1.73	1.73	1.73	2.02	1.46	0.12
	0.2		242	121	11.04	11.04	13.05	21.73	23.19	10.85
	0.3		366	183	25.65	25.65	25.65	2088.19	320.45	18.56
50	0.1	4	122	61	3.51	3.51	3.65	5.50	7.82	2.11
	0.2		242	121	19.64	19.94	21.65	1129.04	>3000	940.20
	0.3		366	183	39.71	39.71	42.63	>3000	>3000	>3000
50	0.1	5	122	61	5.43	5.43	5.43	10.80	20.76	4.65
	0.2		242	121	28.13	28.13	31.01	>3000	>3000	>3000

Note: In this table, the the bounds for cardinalities are chosen as $1, N-1$, i.e., $C_{min} = 1, C_{max} = N-1$.

Table 4-5. Computational results and seconds for robust graph partitioning (2)

Graphs			Uncertainty		Cardinality	Objective Values			CPU Seconds		
N	r	K	$\|J\|$	Γ	$[C_{min}, C_{max}]$	MIP	BD	DP0	MIP	BD	DP0
10	0.1	3	4	2	[3,5]	0	0	0	0.01	0.02	0.02
	0.2		9	5		2.55	2.55	2.55	0.20	0.26	0.14
	0.3		13	7		6.29	6.21	6.03	0.17	0.61	0.24
	0.4		18	9		14.01	14.01	14.13	0.59	2.14	0.42
	0.5		22	11		18.97	18.97	19.19	0.80	13.16	0.52
	0.6		27	14		26.22	25.49	26.23	1.41	37.69	0.66
	0.7		30	15		27.65	27.59	28.80	1.93	42.98	0.85
	0.8		35	18		32.08	32.00	34.34	4.93	52.67	1.93
	0.9		40	20		38.52	37.82	39.63	5.01	241.77	3.54
15	0.1	3	10	5	[4,6]	1.31	1.31	1.69	0.07	0.12	0.07
	0.2		21	11		7.65	7.65	7.65	0.73	0.58	0.44
	0.3		31	16		18.67	17.92	18.06	0.87	4.45	0.45
	0.4		42	21		28.75	28.75	30.92	1.59	18.50	0.71
	0.5		52	26		40.58	40.49	40.95	10.26	479.57	1.78
20	0.1	3	19	10	[6,8]	5.35	5.29	5.62	0.41	0.41	0.28
	0.2		37	19		17.59	17.59	17.59	1.51	4.46	0.88
	0.3		57	29		34.27	34.03	34.22	3.54	42.34	2.10
	0.4		76	38		57.54	55.45	56.92	35.65	1358.55	16.83
	0.5		95	48		71.58	71.56	71.56	76.80	>3000	35.58
20	0.1	4	19	10	[4,6]	6.50	6.35	7.70	0.60	2.18	0.55
	0.2		37	19		22.03	21.78	22.10	13.79	115.28	2.98
	0.3		57	29		43.25	43.19	43.84	37.21	>3000	9.00
	0.4		76	38		65.37	65.27	65.27	209.51	>3000	46.65
	0.5		95	48		84.02	83.75	85.89	1190.84	>3000	346.60
30	0.1	3	42	21	[9,11]	13.87	13.87	15.12	1.72	5.26	1.21
	0.2		87	44		47.10	47.08	49.51	15.51	434.36	24.44
	0.3		130	65		86.16	84.56	86.66	358.65	>3000	113.79
30	0.1	4	42	21	[7,9]	17.59	17.58	17.59	5.71	85.38	2.67
	0.2		87	44		59.24	58.72	60.36	754.93	>3000	156.81
	0.3		130	65		106.62	104.48	105.34	>3000	>3000	2222.75

Note: These results are based on same data sets as those in Table 4-4 and same parameters except that $[C_{min}, C_{max}]$.



$(N = 20, r = 0.4, |J| = 76, K = 4, [C_{min}, C_{max}] = [1, 19], [4, 6])$

Figure 4-1. Robust graph partitioning objective values regarding Γ

The graph partitioning problem is NP-complete, and the robust graph partitioning problem is also NP-complete as shown in Section 4.2.2. Two decomposition algorithms, which are solving a series of nominal graph partitioning problems, can be used for further research. For example, the approximative semidefinite programming (SDP) method is useful for nominal graph partitioning problem, and we can combine these decomposition methods and the SDP method to solve large robust graph partitioning problems efficiently.

Results of section 4.2 are based on our paper [44].

4.3 Two-Stage Stochastic Graph Partitioning Problem

Let $G = (V, E)$ be an undirected graph with a set of vertices $V = \{v_1, v_2, \cdots, v_N\}$ and a set of edges $E = \{(v_i, v_j) : \text{edge between vertices } v_i \text{ and } v_j, 1 \leq i, j \leq N\}$, where N is the number of vertices. The weights of the edges are given by a matrix $W = (w_{ij})_{N \times N}$, where $w_{ij} (> 0)$ denotes the weight of edge (v_i, v_j) and $w_{ij} = 0$ if no edge (v_i, v_j) exists between vertices v_i and v_j. This matrix is symmetric for undirected graphs G. Thus, the

edge set can be expressed by $E = \{(v_i, v_j) : w_{ij} > 0, 1 \leq i < j \leq N\}$. In the following we also use $(i, j) \in E$ to denote that $(v_i, v_j) \in E$.

Let K be the number of disjoint subsets that we want to partition V into, and let $n_1, \cdots, n_K$ be the cardinalities of subsets of V and these numbers are assumed to be given before partitioning. Usually, K is chosen from $\{2, \cdots, N-1\}$. As shown in [42, 43], the size n_k for subset k can be loosely chosen in a range. For equal partitioning, n_k is chosen as N/K. In this Section 4.3, we assume K is given and set n_k around the value N/K.

Let x_{ik} be the indicator that vertex v_i belongs to the kth subset if $x_{ik} = 1$ or not if $x_{ik} = 0$, and y_{ij} be the indicator that the edge (v_i, v_j) with vertices v_i, v_j are in different subsets if $y_{ij} = 1$ and v_i, v_j in the same subset if $y_{ij} = 0$. Thus, the objective function of graph partitioning to minimize the sum of weights of the edges between the disjoint subsets can be expressed as $\min \sum_{(i,j)\in E} w_{ij} y_{ij}$. The relation between x_{ik} and y_{ij} can be expressed as $y_{ij} = 1 - \sum_{k=1}^{K} x_{ik} x_{jk}$ (similar to (4–13)).

Therefore, the graph partitioning under weight matrix W and given sizes $n_1, \cdots, n_K$ of subsets can be expressed as

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in E} w_{ij} y_{ij}, \\
s.t. \quad & \sum_{k=1}^{K} x_{ik} = 1, \sum_{i=1}^{N} x_{ik} = n_k, \\
& y_{ij} = 1 - \sum_{k=1}^{K} x_{ik} x_{jk}, \\
& x_{ik} \in \{0, 1\}, y_{ij} \in \{0, 1\}, \\
& i \in V, (i, j) \in E, k = 1, \cdots, K.
\end{aligned}
\tag{4–38}
$$

Since no uncertainty is considered in this model, we usually call this *nominal graph partitioning problem*. As stated in Section 2.2, many mathematical programming methods, including linear programming, quadratic programming, and semidefinite programming,

are used for this problem. As discussed in [44] and Section 4.2, the weights of edges in graph $G = (V, E)$ are always uncertain. In mathematical programming, two methods are always used to deal with such uncertainty. The robust optimization models for graph partitioning, is to find out a best partitioning of vertex set V among all uncertain weights of edges in the worst case. In this section, we will introduce the two-stage stochastic graph partitioning problem with finite explicit scenarios to deal with the uncertainty.

The *set of cut edges* or the *cut set* includes all edges with ends in different subsets after partitioning. The *two-stage stochastic graph partitioning problem* (TSGP) consists of finding a best partition of vertex set in two stages: taking some edges into the set of cut edges in the first stage with certain weights for edges in matrix W; assuming that there are totally S scenarios with weight matrix $C^s = (c_{ij}^s)_{N\times N}$ in scenario s of probability p_s, and the second stage in scenario s is to choose some edges into the set of cut edges for satisfying the requirements of partitioning. The objective is to minimize the total expected weight of edges in the set of cut over all scenarios. Under these requirements and descriptions, we formulate the two-stage stochastic graph partitioning problem as a stochastic mixed integer program (SMIP) [12].

Similarly, many combinatorial optimization problems have been extended to two-stage stochastic forms recently. The two-stage maximum matching problem is studied in [79] by an approximation algorithm, the minimum spanning tree problem is extended to two-stage forms in [49], and etc.

In Section 4.3, we assume the distribution of weights has finite explicit scenarios. The rest of Section 4.3 is organized as follows: Section 4.3.1 presents the model for the two-stage stochastic graph partitioning problem; In Section 4.3.2, we present equivalent integer linear formulations; In Section 4.3.3, we present numerical experiments on randomly generated graphs with different numbers of scenarios; Section 4.3.4 concludes Section 4.3.

4.3.1 The Model of the Two-Stage Stochastic Graph Partitioning Problem

The two-stage stochastic graph partitioning problem can be formally stated as follows: Given a graph $G = (V, E)$ with the first-stage edge weight matrix W and second-stage edge weight matrix $C^s = (c_{ij}^s)_{N\times N}$ for $s = 1, \cdots, S$, and the probability p_s for scenario s, where S is the number of scenarios. The edge set E now is defined as $E = \{(v_i, v_j) : w_{ij} > 0 \text{ or } c_{ij}^s > 0 \text{ for some } s, j > i\}$, which means if $w_{ij} > 0$ or one of $c_{ij}^s > 0$, the edge (v_i, v_j) exists. That is, for edge $(i, j) \in E$, the first-stage weight is w_{ij} and the second stage weight is c_{ij}^s in scenario s with probability p_s. In addition, we are also given the number K of subsets that we want to partition V into and the cardinalities $n_1, \cdots, n_K$ of all subsets.

Remark 7. *The weight matrix C^s for scenario s has the same prefigurements as W: no loops in the graph, i.e., $c_{ii}^s = 0$ for $i = 1, \cdots, N$; symmetrically, i.e., $c_{ij}^s = c_{ji}^s$ for $i, j = 1, \cdots, N$; nonnegativity, i.e., $c_{ij}^s \geq 0$ for $i, j = 1, \cdots, N$.*

Remark 8. *The probability p_s for $s = 1, \cdots, S$ and weight matrices C^s's are defined on a probability space $(\Omega, \mathcal{C}, \mathcal{P})$, where Ω is the sample space and can be chosen as nonnegative real space $\mathcal{R}_+^{N\times N}$, $\mathcal{C}$ is the set of subsets in Ω, and $\mathcal{P}$ is the probability measure. In this problem, we assume finite explicit scenarios.*

The *two-stage stochastic graph partitioning problem* (TSGP) is to find a set of cut edges E^C with the minimum sum of weights so that the subsets satisfy the requirements at each scenario. Assume that E^0 is the set of cut edges chosen in the first-stage, and E^s is the chosen set of cut edges in the second-stage with respect to scenario s for $s = 1, \cdots, S$, the sets have the relations $E^0 \cup E^s$ is the set that can completely separate the vertices into K subsets with the requirement of cardinalities, and $E^0 \cap E^s = \emptyset$. In addition, the cuts $E^0, E^1, \cdots, E^S$ should have the minimum expected sum of weights

$$\sum_{(v_i,v_j)\in E^0} w_{ij} + \sum_{s=1}^{S} p_s \sum_{(v_i,v_j)\in E^s} c_{ij}^s.$$

For example, in Fig. 4-2, the weights for edge $(v_i, v_j) \in E$ at fist stage (w_{ij}) and second stage (c_{ij}^s) for two scenarios ($p_1 = 0.6, p_2 = 0.4$) are shown. The problem is to find 3 subsets with cardinalities as $n_1 = 3, n_2 = 4, n_3 = 5$. By the STGP model, two edges $(2, 4), (7, 12)$ are selected into the cut set at the first stage, while at the second stage, edges $(3, 7), (3, 11), (6, 8)$ are selected in the first scenario $s = 1$ and edges $(4, 7), (5, 7), (6, 7), (8, 9), (8, 11), (8, 12), (10, 11), (11, 12)$ are selected in the second scenario $s = 2$. Three subsets obtained for scenario $s = 1$ are $\{1, 2, 3\}, \{4, 5, 6, 7\}$, $\{8, 9, 10, 11, 12\}$, while three subsets for scenario $s = 2$ are $\{9, 10, 12\}, \{4, 5, 6, 8\}$, $\{1, 2, 3, 7, 11\}$.

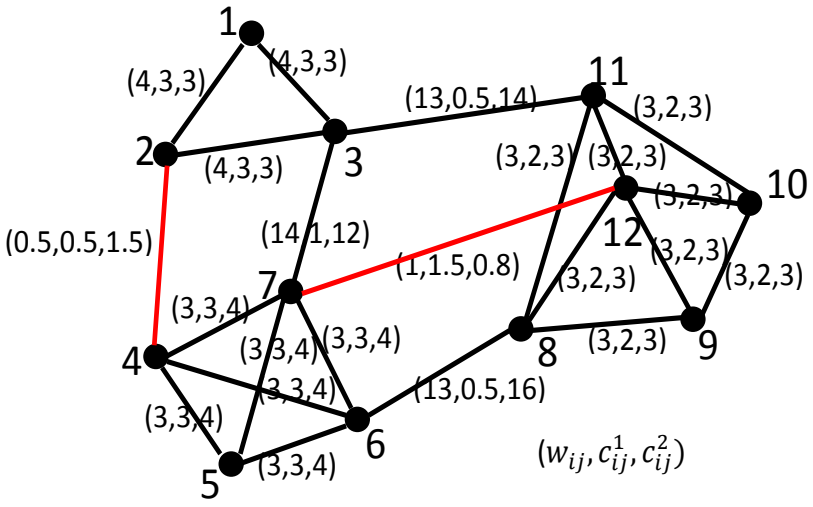


Figure 4-2. An example for two-stage stochastic graph partitioning problem

Assume that $y_{ij} = 1$ denotes that (i, j) is chosen to E^0 and otherwise $y_{ij} = 0$, and $z_{ij}^s = 1$ denotes edge (i, j) is chosen to E^s in scenario s and otherwise $z_{ij}^s = 0$. Let $x_{ik}^s = 1$ denote that the ith vertex belongs to the kth subset and otherwise $x_{ik}^s = 0$. By these decision variables, the two-stage stochastic graph partitioning problem can be

formulated as the following two-stage program:

$$[\textbf{TSGP}] : \min \quad \sum_{(i,j)\in E} w_{ij}y_{ij} + \sum_{s=1}^{S} p_s f(y,s) \tag{4–39}$$

$$s.t. \quad y_{ij} \in \{0,1\}, \tag{4–40}$$

$$i \in V, (i,j) \in E.$$

where for $s = 1, \cdots, S$,

$$f(y,s) = \min_{x,z} \quad \sum_{(i,j)\in E} c_{ij}^s z_{ij}^s \tag{4–41}$$

$$s.t. \quad \sum_{k=1}^{K} x_{ik}^s = 1, \sum_{i=1}^{N} x_{ik}^s = n_k \tag{4–42}$$

$$y_{ij} + z_{ij}^s = 1 - \sum_{k=1}^{K} x_{ik}^s x_{jk}^s, \tag{4–43}$$

$$x_{ik}^s, z_{ij}^s \in \{0,1\}, \tag{4–44}$$

$$i \in V, (i,j) \in E, k = 1, \cdots, K.$$

Next, we first prove this formulation is the correct formulation for two-stage stochastic graph partitioning problem, and then discuss the relaxations of the variables y_{ij}'s and z_{ij}^s's.

Theorem 4.6. *The formulation* (4–39)-(4–44) *is the correct model for the two-stage stochastic graph partitioning problem.*

Proof. From the objective function in (4–39), the decision variables y_{ij} and z_{ij}^s decide whether the edge (v_i, v_j) is included in the set of cut edges for scenario s with respect to the first stage weight w_{ij} and second stage weight c_{ij}^s, respectively. The constraints $x_{ik}^s \in \{0,1\}$ and the constraints (4–42) can guarantee that each vertex belongs to exact one subset and the kth subset has the cardinality n_k in the second stage of scenario s.

Thus, the set of cut edges in the first stage is $E^0 = \{(v_i, v_j) \in E : y_{ij} = 1\}$ and the set of cut edges in the second stage of scenario s is $E^s = \{(v_i, v_j) \in E : z_{ij}^s = 1\}$. We have to prove that $E^0 \cup E^s$ is the set of cut edges and $E^0 \cap E^s = \emptyset$ for any $s = 1, \cdots, S$.

If both vertex v_i and v_j belong to subset k in scenario s, i.e., $x_{ik}^s = x_{jk}^s = 1$ and $x_{ik'}^s = x_{jk'}^s = 0$ for $k' \neq k$, from the constraint (4–43), we have $y_{ij} + z_{ij}^s = 0$. Thus, $y_{ij} = z_{ij}^s = 0$ since both of them are binary. The edge (v_i, v_j) is not in the set of cut edges.

If the vertex v_i belongs to subset k_1 and v_j belongs to subset k_2 of scenario s, i.e., $x_{i,k_1}^s = 1, x_{ik}^s = 0$ for all $k \neq k_1$ and $x_{j,k_2}^s = 1, x_{j,k'}^s = 0$ for all $k' \neq k_2$ where $k_1 \neq k_2$, thus we have $y_{ij} + z_{ij}^s = 1$ from the constraint (4–43). Thus, since both $y_{ij}, z_{ij}^s \in \{0, 1\}$, either $y_{ij} = 1$, which means edge (v_i, v_j) is chosen in cut edges in the first stage, or $z_{ij}^s = 1$, which means (v_i, v_j) is chosen in the cut edges in the second stage of scenario s. Considering all edges, we have proved that $E^0 \cup E^s$ is the set of cut edges and $E^0 \cap E^s = \emptyset$ for any $s = 1, \cdots, S$.

The objective function is to minimize fist stage weight and the expected sum weight of all scenarios $s = 1, \cdots, S$ for edges within the cut. Therefore, we have checked the objective and all constraints, and the program (4–39)-(4–44) correctly formulates the two-stage stochastic graph partitioning problem.

□

Corollary 4.1. *For edge $(v_i, v_j) \in E$, if $w_{ij} > \sum_{s=1}^{S} p_s c_{ij}^s$, we have $y_{ij} = 0$.*

Proof. By contradiction, if $y_{ij} = 1$, which means that edge (v_i, v_j) is chosen into the cut set in the first stage, and the objective with respect to this edge is w_{ij}. However, by choosing $c_{ij}^s = 1$ for all s, the corresponding objective is $\sum_{s=1}^{S} p_s c_{ij}^s$, which is less than w_{ij}, a contradiction to minimizing the objective.

□

However, for the case $w_{ij} = \sum_{s=1}^{S} p_s c_{ij}^s$ for edge $(v_i, v_j) \in E$, if this edge is chosen into the cut set, either $y_{ij} = 0, c_{ij}^s = 1(\forall s)$ or $y_{ij} = 1, c_{ij}^s = 0(\forall s)$ are optimal solutions; if

this edge is not chosen into the cut set, $y_{ij} = 0$. In order to reduce computation time, we make the following assumption without loss of any optimality:

Assumption A For edge $(v_i, v_j) \in E$, if $w_{ij} = \sum_{s=1}^{S} p_s c_{ij}^s$, assume $y_{ij} = 0$.

4.3.2 Equivalent Integer Linear Programming Formulations

In the constraint (4–43), there is a nonlinear term $x_{ik}^s x_{jk}^s$, which always leads to high computational complexity. In this section, we present an approach to linearize this term. Additionally, we prove that some binary variables in the formulation for TSGP can be relaxed to continuous ones.

Lemma 4.1. *By Corollary 4.1 and under* Assumption A*, the decision variables y_{ij}, z_{ij}^s in the two-stage stochastic graph partitioning problem* (4–39)-(4–44) *can be relaxed to be continuous ones such that* $0 \le y_{ij} \le 1, z_{ij}^s \ge 0$.

Proof. In the part E^0, E^s of the proof for Theorem 4.6, in the case of vertices v_i, v_j in the same subset in scenario s, i.e., $y_{ij} + z_{ij}^s = 0$, we have $y_{ij} = z_{ij} = 0$ if $y_{ij}, z_{ij}^s \ge 0$.

In the case of vertices v_i, v_j in different subsets in scenario s, i.e., $y_{ij} + z_{ij}^s = 1$, we discuss in the following three cases:

(a) $w_{ij} > \sum_{s=1}^{S} p_s c_{ij}^s$. By Corollary 4.1, $y_{ij} = 0$ and then $z_{ij}^s = 1$.

(b) $w_{ij} = \sum_{s=1}^{S} p_s c_{ij}^s$. By *Assumption A*, $y_{ij} = 0$ and then $z_{ij}^s = 1$.

(c) $w_{ij} < \sum_{s=1}^{S} p_s c_{ij}^s$. We have $z_{ij}^s = 1 - y_{ij}$. The part in the objective function (4–39) with respect to edge (v_i, v_j) is

$$\begin{aligned}\min(w_{ij} y_{ij} + \sum_{s=1}^{S} p_s c_{ij}^s z_{ij}^s) &= \min(w_{ij} y_{ij} + \sum_{s=1}^{S} p_s c_{ij}^s (1 - y_{ij})) \\ &= \min(w_{ij} - \sum_{s=1}^{S} p_s c_{ij}^s) y_{ij} + \sum_{s=1}^{S} p_s c_{ij}^s.\end{aligned}$$

If $y_{ij} \in \{0, 1\}$, the minimum for this problem is obtained when $y_{ij} = 1$ and $z_{ij}^s = 0$ since $w_{ij} - \sum_{s=1}^{S} p_s c_{ij}^s < 0$. If y_{ij} is relaxed to $y_{ij} \le 1$, the above problem is also optimized at $y_{ij} = 1$ and $z_{ij}^s = 0$.

Summarizing all cases above, we can obtain that under *Assumption A*, the decision variables $y_{ij}, z_{ij}^s \in \{0, 1\}$ in TSGP can be relaxed by $0 \le y_{ij} \le 1, z_{ij}^s \ge 0$. □

For the nonlinear term $x_{ik}^s x_{jk}^s$, by introducing $u_{ijk}^s = x_{ik}^s x_{jk}^s$, we can have following linearization methods.

Lemma 4.2. *The constraint* (4–43) *for edge* $(v_i, v_j) \in E$ *in scenario* s *is equivalent to following linear constraints:*

$$\begin{cases} y_{ij} + z_{ij}^s = 1 - \sum_{k=1}^{K} u_{ijk}^s \\ u_{ijk}^s \leq x_{ik}^s \\ u_{ijk}^s \leq x_{jk}^s \\ u_{ijk}^s \geq x_{ik}^s + x_{jk}^s - 1 \\ u_{ijk}^s \geq 0 \end{cases} \tag{4–45}$$

By Lemma 4.1 and Lemma 4.2, we have the following theorem, which presents the equivalent integer linear programming formulation for TSGP.

Theorem 4.7. *The formulation in* (4–39)-(4–44) *for TSGP under* Assumption A *is equivalent to the following integer linear program in extensive form:*

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} w_{ij} y_{ij} + \sum_{s=1}^{S} p_s \sum_{(i,j)\in E} c_{ij}^s z_{ij}^s \\ s.t. \quad & (4\text{–}42), (4\text{–}45) \\ & x_{ik}^s \in \{0,1\}, 0 \leq y_{ij} \leq 1, z_{ij}^s \geq 0 \\ & i \in V, (i,j) \in E, k = 1, \cdots, K, s = 1, \cdots, S \end{aligned}$$

For some specific case, we can have more simplified formulation for TSGP as shown in the following corollary.

Corollary 4.2. *If* $w_{ij} > 0$ *and* $c_{ij}^s > 0$ *hold for all edge* $(v_i, v_j) \in E$ *for all scenarios, the formulation in* (4–39)-(4–44) *for TSGP is equivalent to:*

$$\min \quad \sum_{(i,j)\in E} w_{ij} y_{ij} + \sum_{s=1}^{S} p_s \sum_{(i,j)\in E} c_{ij}^s z_{ij}^s \tag{4–46}$$

$$s.t. \quad \sum_{k=1}^{K} x_{ik}^s = 1, \sum_{i=1}^{N} x_{ik}^s = n_k \tag{4–47}$$

$$-(y_{ij} + z_{ij}^s) - x_{ik}^s + x_{jk}^s \leq 0 \tag{4–48}$$

$$-(y_{ij} + z_{ij}^s) + x_{ik}^s - x_{jk}^s \leq 0 \tag{4–49}$$

$$x_{ik}^s \in \{0,1\}, 0 \leq y_{ij} \leq 1, z_{ij}^s \geq 0 \tag{4–50}$$

$$i \in V, (i,j) \in E, k = 1, \cdots, K, s = 1, \cdots, S$$

Proof. For edge $(v_i, v_j) \in E$ such that $w_{ij} > 0$ and $c_{ij}^s > 0$ in scenario s, different from Lemma 4.1, if vertices v_i, v_j are in the same subset, i.e., $x_{ik'}^s = x_{jk'}^s = 1$ and $x_{ik}^s = x_{jk}^s = 0$ for all $k \neq k'$, we have $y_{ij} + z_{ij}^s \geq 0$ with considering all k's in (4–48) and (4–49); Similarly, if vertices v_i, v_j are different subsets, we have $y_{ij} + z_{ij}^s \geq 1$ from (4–48) and (4–49). As mentioned in Lemma 4.1, the objective function (4–46) with respect to edge (v_i, v_j) is $\min(w_{ij} y_{ij} + \sum_{s=1}^{S} p_s c_{ij}^s z_{ij}^s)$.

For the case $y_{ij} + z_{ij}^s \geq 0$, we want to prove that $y_{ij} = 0$ and $z_{ij} = 0$ by the formulation (4–46)-(4–50). we have three subcases:

(a) $w_{ij} > \sum_{s=1}^{S} p_s c_{ij}^s$. By Corollary 4.1, $y_{ij} = 0$ and then $z_{ij}^s \geq 0$. Now, $\min(w_{ij} y_{ij} + \sum_{s=1}^{S} p_s c_{ij}^s z_{ij}^s) = \min \sum_{s=1}^{S} p_s c_{ij}^s z_{ij}^s$ should obtain the optimal value at $z_{ij}^s = 0$ since all $c_{ij}^s > 0$.

(b) $w_{ij} = \sum_{s=1}^{S} p_s c_{ij}^s$. By *Assumption A*, $y_{ij} = 0$ and then $z_{ij}^s \geq 0$. This can be proved similarly to above case.

(c) $w_{ij} < \sum_{s=1}^{S} p_s c_{ij}^s$. We have $y_{ij} + z_{ij}^s \geq 0$. $\min(w_{ij} y_{ij} + \sum_{s=1}^{S} p_s c_{ij}^s z_{ij}^s)$ should obtain the optimal value at $y_{ij} = 0, z_{ij}^s = 0$ since $w_{ij} > 0$ and $c_{ij}^s > 0$ for all s.

For all subcases, we have $y_{ij} = 0, z_{ij}^s = 0$ when $y_{ij} + z_{ij}^s = 0$, which is the same as the case $y_{ij} + z_{ij}^s = 0$ in Lemma 4.1.

For the case $y_{ij} + z_{ij}^s \geq 1$, we also discuss three subcases:

(d) $w_{ij} > \sum_{s=1}^{S} p_s c_{ij}^s$. By Corollary 4.1, $y_{ij} = 0$ and then $z_{ij}^s \geq 1$. Now, $\min(w_{ij}y_{ij} + \sum_{s=1}^{S} p_s c_{ij}^s z_{ij}^s) = \min \sum_{s=1}^{S} p_s c_{ij}^s z_{ij}^s$ should obtain the optimal value at $z_{ij}^s = 1$ since all $c_{ij}^s > 0$.

(e) $w_{ij} = \sum_{s=1}^{S} p_s c_{ij}^s$. By *Assumption A*, $y_{ij} = 0$ and then $z_{ij}^s \geq 1$. This can be proved similarly to above case.

(f) $w_{ij} < \sum_{s=1}^{S} p_s c_{ij}^s$. We have $y_{ij} + z_{ij}^s \geq 1$. If $y_{ij} = 0$, $\min(w_{ij}y_{ij} + \sum_{s=1}^{S} p_s c_{ij}^s z_{ij}^s)$ should obtain the optimal value $\sum_{s=1}^{S} p_s c_{ij}^s$ at $z_{ij}^s = 1$ for all s since $z_{ij}^s \geq 1$ and $c_{ij}^s > 0$ for all s; If $y_{ij} = 1$, the objective for edge (v_i, v_j) obtain the optimal value w_{ij} at $z_{ij}^s = 0$ for all s since $z_{ij}^s \geq 0$ and $c_{ij}^s > 0$ for all s; If $0 < y_{ij} < 1$, assume the optima for $\min(w_{ij}y_{ij} + \sum_{s=1}^{S} p_s c_{ij}^s z_{ij}^s)$ is obtained at $y_{ij} = a$ $(0 < a < 1)$, and thus $z_{ij}^s \geq 1 - a$. Then

$$\min(w_{ij}y_{ij} + \sum_{s=1}^{S} p_s c_{ij}^s z_{ij}^s) = aw_{ij} + \min \sum_{s=1}^{S} p_s c_{ij}^s z_{ij}^s$$

with constraints $z_{ij}^s \geq 1-a$ obtains its optimal value $aw_{ij}+(1-a)p_s c_{ij}^s$ at $z_{ij}^s = 1-a$ for all s. Comparing the objective values $\sum_{s=1}^{S} p_s c_{ij}^s$, $aw_{ij}+(1-a)p_s c_{ij}^s$, w_{ij} at $y_{ij} = 0, a, 1$, respectively, we have

$$\sum_{s=1}^{S} p_s c_{ij}^s > aw_{ij} + (1-a)p_s c_{ij}^s > w_{ij}$$

for any $0 < a < 1$, by considering $w_{ij} < \sum_{s=1}^{S} p_s c_{ij}^s$. Therefore, under this subcase, the optimal value is obtained at $y_{ij} = 1$ and $z_{ij}^s = 0$ for all s.

The solutions for the case $y_{ij} + z_{ij}^s \geq 1$ under certain situations are the same as those in Lemma 4.1 when $y_{ij} + z_{ij}^s = 1$.

□

In Corollary 4.2, we require that for any edge $(v_i, v_j) \in E$. $w_{ij} > 0$ and $c_{ij}^s > 0$ for all s. For example, in case (d), if $c_{ij}^s = 0$, the term $\min c_{ij}^s z_{ij}^s$ with $z_{ij} \geq 1$ can obtain the minimum value 0 at any value of nonnegative z_{ij}^s; in case (c), $w_{ij} = 0$ will influence any nonnegative choice of y_{ij}.

Comparing formulations in Theorem 4.7 and Corollary 4.2, we can see that the formulation in Corollary 4.2 reduces a lot of constraints and variables, and it can be

solved more efficiently. Our method for solving TSGP is using CPLEX [28] to solve the binary linear programs in extensive form in Theorem 4.7 and Corollary 4.2.

4.3.3 Numerical Experiments

In this section, we implement all binary linear programs in Theorem 4.7 and Corollary 4.2 using CPLEX 11.0 via ILOG Concert Technology 2.5. All computations are performed on a SUN UltraSpace-III with a 900 MHz processor and 2.0 GB RAM. Computational times are reported in CPU seconds.

To test our algorithms, we generate random graphs with different number of nodes, edges and scenarios. The number of edges is determined by the density r of a graph, which is the ratio of the number of edges and the number of possible edges. All generated weights satisfy $w_{ij}, c_{ij}^s \in [2, 3]$ for $(v_i, v_j) \in E$. The number $N(N-1)r/2$ of decision variables y_{ij}s is related to the number of edges, which is related to N, r; the number $N(N-1)rS/2$ of z_{ij}^ss is determined by N, r, s; and the number $N \cdot K \cdot S$ of x_{ik}^ss is determined N, S, K. In Theorem 4.7, the variables u_{ijk}^ss are introduced, and the number of u_{ijk}^ss is $N(N-1)rKS/2$.

In Table 4-6, the objective values obtained by solving formulations in Theorem 4.7 and Corollary 4.2 are the same while the gap in CPLEX is setting to 0.01.

From Table 4-6, we can see all computational seconds by the formulation in Corollary 4.2 are less than or equal to seconds by the formulation in Theorem 4.7 under the same test cases. As discussed in Section 4.3.2, the formulation in Theorem 4.7 introduces the variables u_{ijk}^ss and this influences the computational complexity. For the graph with $N = 10$ vertices and $S = 2$ scenarios, when the density r is increasing, the computational seconds are increasing as well; for the graph $N = 10, S = 2$ and the same density r, when the number K of subsets increases, the computational seconds increase as well for both formulations. Similarly, when the number S of scenarios and the number N of vertices are increasing, computational seconds increase as well.

4.3.4 Discussion

In this section, we present the two-stage stochastic graph partitioning problem by the stochastic programming approach. This problem is formulated as a nonlinear stochastic mixed integer program, and we also present a linear programming (SMIP) approach for solving this problem by CPLEX. For some cases with specific requirements, we present a more simplified linearization method, which can solve the problem more efficiently. The SMIP problem is very hard to solve, and for further research, more efficient algorithms, such as methods based on Benders decomposition [6], should be designed for solving large-scale two-stage stochastic graph partitioning problems. Results of Section 4.3 are based on our paper [45].

Table 4-6. Computational results and seconds for two-stage stochastic graph partitioning

Graphs			Probability Dist.	Cardinalities	Objective Value	CPU Seconds	
N	r	S	$p_1 + \cdots + p_S$	$n_1 + \cdots + n_K$	Thm 4.7/Cor 4.2	Thm 4.7	Cor 4.2
10	0.1	2	0.6+0.4	5+5	0.00	0.01	0.01
	0.4				13.49	0.10	0.06
	0.7				31.61	0.50	0.33
	1				57.91	17.30	15.84
	0.1	2	0.6+0.4	3+3+4	2.11	0.04	0.03
	0.4				19.72	3.20	2.11
	0.7				43.94	23.04	17.78
	1				76.77	2295.31	1828.23
	0.1	2	0.6+0.4	3+2+3+2	2.11	0.05	0.04
	0.4				22.18	4.95	4.04
	0.7				52.08	170.38	120.70
	1				86.14	5088.86	3583.21
10	0.1	4	0.1+0.2+0.3+0.4	4+3+3	0.00	0.03	0.03
	0.4				20.80	90.62	64.15
	0.1	6	0.05+0.10+0.15+0.20+0.25+0.25	3+3+4	2.21	1.33	0.86
	0.4				19.47	1938.17	774.08
15	0.1	2	0.6+0.4	5+5+5	4.58	0.09	0.08
	0.4				43.42	42.33	23.41
20	0.1	2	0.6+0.4	7+7+6	6.52	0.85	0.42
	0.4				85.22	4354.52	2519.88
25	0.1	2	0.6+0.4	8+8+9	13.95	7.00	1.23
30	0.1	2	0.6+0.4	10+10+10	22.33	11.23	1.44

CHAPTER 5
APPLICATIONS IN POWER SYSTEMS FOR ISLANDING

A power grid island is a self-sufficient subnetwork in a large-scale power system. In weakly connected islands, limited inter-island power flows are allowed. Intentional islanding detection is helpful for the analysis of distributed generation systems connected to a power grid, and also valuable for power system reliability in extreme emergency states. In this chapter, we use graph partitioning methods to form islands in a power grid and formulate these problems as mixed integer programs. Our models are based the optimal power flow model to minimize the load shedding cost. With these mathematical programming models, optimal formation of islands can be obtained and the different approaches can be compared. Through experiment on the IEEE-30-Bus system, computational results are analyzed and compared to provide insight for power grid intentional islanding.

This chapter is organized as follows: Section 5.1 introduces some backgrounds about power grid islanding and some other applications of clustering and graph partitioning in power systems. In Section 5.2, optimal power flow models are reviewed for both AC and DC cases. In Section 5.3, we construct a power grid islanding model based on the DC-OPF, and discuss its variations with minimum island size and inter-island flows. In Section 5.4, our proposed models are applied to the IEEE-30-Bus System Test Case and the results are analyzed and compared for the different models. Section 5.5 summarizes our results.

5.1 Introduction

Recent reports of large scale blackouts in North America, Europe and other countries show that power systems may be subject to outages because of vulnerable operating conditions, such as system limits, weak connections, unexpected events, hidden failures, and human errors. In some other areas, such as many African, South Asian and Latin American countries, the rolling blackout, or load shedding is

a staple of daily life. Besides reasons of insufficient generation capacity and inadequate transmission infrastructure for delivering sufficient power, security and stability issues are vital issues of modern power systems [65]. To improve the reliability of power systems by solving security and stability problems, the method of splitting a large power network into subsystems can present a way by which the remaining systems can operate properly in an acceptable condition, when problems affect some other part of the system. This provides a way to prevent large-scale blackouts. Power grid islanding is one approach of splitting the power network into self-sufficient islands for emergency controls.

On the other hand, with the development of technologies for solar and wind energy resources, more and more distributed generators are added to the centralized generation system. Because of uncertain issues in solar or wind energy, the reliability of a power system becomes even more important than ever, especially in case of failures. Current methods for protecting the decentralized generation power system is “either to disconnect all distributed generators once a fault occurs or to impletment an islanding detection algorithm” ([140]) that detects an islanding situation and initiates disconnection of distributed generators [107].

Therefore, power grid intentional islanding is one promised method to improve the power system reliability for its more flexible control property. The new IEEE Standard 1547-2003 [66] provides general guidelines for design and operation of intentional islands in case of power outages.

There are several approaches to form islands in a power grid. In [85], Liu and Liu presented a review of the main aspects for power system islanding, and they outlined islanding schemes according to graph partitioning, minimal cutset enumeration and generator grouping. Since the topology of the power grid can be denoted by a graph including vertices (buses) connected by edges (transmission lines), graph partitioning is a directly used method to partition the vertex sets into several subsets

[11, 83, 111, 123]. Minimal cutsets are used in partitioning a network into smaller subnetworks [111, 128]. The self-healing strategy [138] can deal with catastrophic events when a power system is approaching an extreme emergency condition by dividing it into small islands with consideration of quick restoration. In [134, 137, 139], the power grid islanding methods are based on slow coherency, which has the capacity to determine sets of generator groups among weak connections. Recent work in controlled islanding used the decision tree method [112] and Pareto optimization [126]. Some other partitioning methods based on matrices were used to partition the power grid, for example, spectral method on power flow Jacobian matrix [123], and spectral and k-means methods on real power flow matrix [99].

Load shedding is always taken to reduce the imbalance of generation and load demand after islanding [36, 85, 138, 140]. In order to be self-sufficient, each island should be balanced. After islanding, islands may have either excess of generation or of load. Excess generation can be controlled by adjusting the output of generators, while some load should be curtailed as load shedding.

Most of the above approaches can only form two islands in a large network, although the procedures can be repeated to form an even number of islands. In [11, 83], approaches for simultaneously obtaining more than two islands are proposed by using graph partitioning. As stated above, islanding a power grid is similar to the graph partitioning problem which is a classic problem in combinatorial optimization and can form multiple partitions in general.

Some optimization approaches ([111, 123, 126, 128]) to power grid islanding relied on heuristic and approximation methods. Without an explicit mathematical programming formulation, the effectiveness of islanding can only be analyzed with approximation ratios or empirically. Although some optimization methods are used for islanding [111, 140] by mathematical programming, they just obtained two islands. Methods in [140] lacked of connectivity constraints, which are crucial for the reliability of islands, and

because of the nonlinear property, they can only work on small cases. Models in [111] were more of graph partitioning and did not consider the load shedding.

In this chapter, we use the Direct Current Optimal Power Flow (DC-OPF) model and graph partitioning approaches to formally formulate power grid islanding problems as mixed integer programs (MIPs). Our model can form multiple islands simultaneously and can be used to form *weakly connected islands* among which inter-island flows are allowed. We adapte the set of measures of inter-island flows from [111, 123] in our formulation. Additionally, connectivity constraints are added to ensure buses, including generators and load buses, within each island are connected. Although our initial formulations result in nonlinear MIPs, we provide the procedures to transform them into linear MIPs. These different models are compared in the simulation experiments. Inter-island flows are viewed as a way to improve power grid performance since they provide a less restrictive solution than complete partition does. However, our simulation results show that inter-island flows do not always help power grid performance and in many cases, completely disconnected islands perform much better than weakly connected islands. The islanding scheme is for intentional and pre-planned islanding for emergency controls. We assume that generators within one island can be synchronous easily.

Different from [140], we add connectivity constraints for each island. Without this kind of constraint, an island may be formed by several parts, and some generation cannot be used other parts of this subsystem. Our weakly connected islanding models allow flows among distinct islands, and this is the real reflection of nowadays' interconnected systems. Additionally, our model can obtain any number of islands directly.

Besides the application of graph partitioning in power systems, clustering is also used in power grids. For example, in [13], clustering is used to find PJM zones for

multiple planning and operations analysis, and this is quite useful for reliability and resource adequacy assessment, as well as identification and mitigation of market power.

5.2 Optimal Power Flow Models

In this section, we state the optimal power flow models. First, we state the notation, parameters and decision variables, used throughout this chapter.

5.2.1 Notations

Indices and Index Sets

- I: set of buses
- $(i, j) \in L$: undirected transmission lines, $i, j \in I$ and $i < j$
  - L': transmission lines with direction, $(i, j), (j, i) \in L'$ if $(i, j) \in L$
- $k \in \{1, \cdots, K\}$: K denotes the number of islands

Let $N = (I, L)$ be an undirected network representing a power grid where I is the set of buses/nodes and L is the set of transmission lines/edges. The direction of the transmission line $l \in L$ is decided by the phase angle differences between its two ends. For any node $i \in I$, it can have generation capacity supplied by generators connected to i and can have loads consumed by consumers connecting to i. For example, node 1 has generation capacity and node 2 has both generation and load in Fig. 5-1.

Parameters

- P_{D_i}: load at bus i
- $P_{G_{i_{max}}}$: generation capacity at bus $i \in I$
- $P_{ij_{max}}$: transmission capacity for line $(i, j) \in L$
- B_{ij}: imaginary part of the admittance matrix w.r.t. line $(i, j) \in L$
- C_{S_i}: penalty cost for load shedding at $i \in I$

Decision variables

- P_{G_i}: power generation at node $i \in I$

- P_{ij}: power flow on line $(i, j) \in L$
- P_{S_i}: load shed at node $i \in I$
- θ_i: phase angle at node $i \in I$
- x_{ik}: $\{0, 1\}$ variable to decide the island belongings, e.g., $x_{ik} = 1$ denotes that node i is in island k
- y_k: variable equal to the total number of buses in subnetwork k
- $y_{ij,k}$: continuous decision variable for line (i, j) in island k to ensure the connectivity of subnetwork k
- z_{ij}: 0 if the end nodes i and j are in different islands and 1 if the end nodes are in the same island.

5.2.2 Power Flow Models

The alternating current optimal power flow (AC-OPF) problem deals with the network flow problem for the AC electric transmission grid to optimality dispatch generation to load consumers subject to network flow constraints and reliability constraints. The flow of electric energy among the power grid follows Kirchhoff's laws. The AC-OPF problem includes a nonlinear objective function and constraints with trigonometric functions. Let Q_{ij} (in MVARs) denote the reactive power flow on line (i, j). The power flow from bus i to j is expressed as $P_{ij} + \sqrt{-1}Q_{ij}$. Let V_i, V_j denote the voltage magnitudes at bus i and j. Let g_{ij}, b_{ij} denote the conductance and the susceptance for line (i, j), respectively, where $g_{ij} = r_{ij}/(r_{ij}^2 + x_{ij}^2)$, $b_{ij} = -x_{ij}/(r_{ij}^2 + x_{ij}^2)$ and r_{ij} is the resistance and x_{ij} is the reactance of line (i, j). By these notations, P_{ij} and Q_{ij} are expressed as

$$P_{ij} = V_i^2 g_{ij} - V_i V_j [g_{ij} \cos(\theta_i - \theta_j) + b_{ij} \sin(\theta_i - \theta_j)]$$

$$Q_{ij} = -V_i^2 b_{ij} - V_i V_j [g_{ij} \sin(\theta_i - \theta_j) - b_{ij} \cos(\theta_i - \theta_j)]$$

The direct current optimal power flow problem (DC-OPF) is an approximation for an underlying AC-OPF problem under the following simplifying restrictions and assumptions:

A.1 The resistance for each line is negligible compared to the reactance and can be set to 0;

A.2 The voltage magnitude at each bus is set to equal to the base voltage V_0;

A.3 The voltage angle difference $\theta_i - \theta_j$ along each line (i, j) is sufficiently small so that $\cos(\theta_i - \theta_j) \approx 1$ and $\sin(\theta_i - \theta_j) \approx \theta_i - \theta_j$.

By some simplifications in [118], we can obtain $P_{ij} = B_{ij}(\theta_i - \theta_j)$, $Q_{ij} = 0$, where $B_{ij} = -b_{ij}$. Usually, one bus is selected as reference bus with specified voltage angle, and we have one more assumption.

A.4 Bus 1 is the reference bus with $\theta_1 = 0$.

For the objective function to minimize the generating cost, it is usually expressed as a quadratic function of the generation output P_{G_i}. Additionally, to maximize the social welfare, we add the load shedding cost $\sum_{i \in I} C_{S_i} P_{S_i}$ to be minimized. Other constraints include the maximum power flow on each line; the power balance at each node, where the served demand at bus i is $(D_i - s_i)$; the maximum generating output; and the limitation of load shedding by the maximum load. Therefore, the DC-OPF problem is modeled as the following linear programming formulation.

$$
\begin{aligned}
\min_{P_G, P_S, P_{ij}, \theta} \quad & \sum_{i \in I} (C_{G_i} P_{G_i} + C_{S_i} P_{S_i}) \\
s.t. \quad & P_{ij} = B_{ij}(\theta_i - \theta_j), \forall (i,j) \in L \\
& P_{G_i} + \sum_{j<i} P_{ji} = (P_{D_i} - P_{S_i}) + \sum_{j>i} P_{ij}, \forall i \in I \\
& -P_{ij_{max}} \leq P_{ij} \leq P_{ij_{max}}, \forall (i,j) \in L \\
& 0 \leq P_{G_i} \leq P_{G_{i\,max}}, \forall i \in I \\
& 0 \leq P_{S_i} \leq P_{D_i}, \forall i \in I
\end{aligned}
$$

Many methods are used for the optimal power flow model, for example, quadratic programming, interior point method and some heuristic methods. These methods are reviewed by [64] and recently by [95].

5.3 Power Grid Islanding Models

5.3.1 Model for Complete Islanding

The DC-OPF model we use is similar to the formulation in [108, 130]. Based on this model, in the following, we present a mathematical program for power grid islanding (**GI-DC-OPF**).

Most current research on power grid islanding [111, 123, 128] deal with a 2-islanding problem by using the $s-t$ graph partitioning approach. In the $s-t$ approach, a power network $N(I, L)$ is partitioned into two subgraphs induced by the node sets I_1 and I_2, $s \in I_1$, $t \in I_2$, and both subgraphs are connected. In this chapter, we consider a general K-islanding problem. Let $\{i^1, i^2, \cdots, i^K\} \subset I$ be a set of K root buses. The K-islanding problem is to separate a power grid into K components $I_1, I_2, \cdots, I_K$ such that $\cup_k I_k = I$, $I_k \cap I_{k'} = \emptyset$ for $k \neq k'$, $i^k \in I_k$, and each component, an induced graph by I_k, is connected.

To keep the formulation less cumbersome, we introduce two indicator functions. The indicator functions are used to show which bus has generation capacity or load. Let $\mathbf{1}_i^g = 1$ for $i \in I$ if $\bar{G}_i > 0$, and 0 otherwise, and let $\mathbf{1}_i^d = 1$ for $i \in I$ if $d_i > 0$, and 0 otherwise. To form K islands, we set $\{i^1, i^2, \cdots, i^K\} \subset I$ to be the root node in each island, i.e., setting $x_{i^k,k} = 1$. Usually, we specify a generator for each island.

The **GI-DC-OPF** model to find K islands is formulated as follows:

$$\mathbf{GI-DC-OPF}:$$

$$\min_{P_{G_i}, P_{S_i}, P_{ij}, \theta, x, y, z} \quad \sum_{i \in I} C_{S_i} P_{S_i} \tag{5–1}$$

$$
s.t.
\begin{cases}
P_{ij} = B_{ij}(\theta_i - \theta_j) z_{ij}, \forall (i,j) \in L \\
-P_{ij_{max}} \leq P_{ij} \leq P_{ij_{max}}, \forall (i,j) \in L \\
P_{G_i} + \sum_{j<i} P_{ji} = (P_{D_i} - P_{S_i}) + \sum_{j>i} P_{ij}, \forall i \in I \\
0 \leq P_{G_i} \leq P_{G_{imax}}, \forall i \in I \\
0 \leq P_{S_i} \leq P_{D_i}, \forall i \in I
\end{cases} \tag{5–2}
$$

$$
\begin{cases}
\sum_{k=1}^{K} x_{ik} = 1, \forall i \in V \\
\sum_{i \in I} \mathbf{1}_i^g x_{ik} \geq 1, \sum_{i \in I} \mathbf{1}_i^d x_{ik} \geq 1, \forall k \\
z_{ij} = \sum_k x_{ik} x_{jk}, \forall (i,j) \in L
\end{cases} \tag{5–3}
$$

$$
\begin{cases}
x_{i^k,k} = 1, \forall k \\
y_k = \sum_{i \in I} x_{ik}, \forall k \\
\sum_j y_{i^k j,k} = y_k - 1, \forall k \\
\sum_{(i,j) \in L'} y_{ij,k} + x_{ik} = \sum_{(j,i) \in L'} y_{ji,k}, \forall k, \forall i \in I, i \neq i^k \\
0 \leq y_{ij,k} \leq y_k x_{ik}, \forall (i,j) \in L', \forall k \\
0 \leq y_{ij,k} \leq y_k x_{jk}, \forall (i,j) \in L', \forall k
\end{cases} \tag{5–4}
$$

$$
x_{ik} \in \{0,1\}, \forall i \in I, \forall k \tag{5–5}
$$

The constraints (5–2) are for DC-OPF, which include (in order): approximate active power flow on transmission lines; the maximum power flow on each line; the power balance at each node, where the served demand at bus i is $(P_{D_i} - P_{S_i})$; the maximum generating output; and the limitation of load shedding by the maximum load. Constraints (5–2) are slightly different from the standard DC-OPF by the addition of z_{ij}. In the standard DC-OPF, $P_{ij} = 0$ implies the phase angles at end buses have to equal. In (5–2), if $z_{ij} = 1$, the constraint is the same as in standard DC-OPF since line (i,j) is inside of an island. If $z_{ij} = 0$, the constraint becomes $P_{ij} = 0$ since line (i,j) is

between two islands and is removed. Because variable z_{ij} relaxes phase angles from the constraint, the phase angle θ_i and θ_j can take any value when $P_{ij} = 0$.

The constraints (5–3) relate to graph partitioning, which include: every node must belong to exactly one island; and every island must have at least one generator and one consumer. The graph partitioning approach has been studied for a long time, but the additional constraints guaranteeing at least one generator and load in each island make our graph partition different from the classical models. Recently in [42], the constraints for each subset (island in our case) has been shown to require loose cardinality requirements instead of equality constraints. Also, if two buses i and j are in the same island, there exists exactly one k' ($1 \leq k' \leq K$) such that $x_{ik'} = x_{jk'} = 1$, $x_{ik} = x_{jk} = 0$ for all other k's, and thus $z_{ij} = 1$. Otherwise, $z_{ij} = 0$.

The constraints (5–4) are to ensure that every island is connected. That is, all buses within one island are connected by transmission lines with both ends in this island. In [32], three ways of enforcing the subnetwork connectivity are summarized via mathematical programming formulations. Here, the method called single commodity flow is used. In this single commodity flow problem, the total number of nodes including i^k in subnetwork k is y_k, and $y_k - 1$ units of flow are supplied at node i^k. Every other node acts as a "sink" by consuming one unit of flow and every node should have a positive incoming flow. The flow out from i^k cannot be distributed to any other subnetwork. The constraints (5–4) are for this single commodity flow problem. If all constraints can be satisfied, every subnetwork should be connected.

The objective (5–1) of **GI-DC-OPF** model is to minimize load shedding cost. In fact, by this objective function, we can minimize the imbalance of generation and load in each island. The total generation of island k is $\sum_{i \in I} P_{G_i} x_{ik}$, which is equal to the total satisfied load demand $\sum_{i \in I} (P_{D_i} - P_{S_i}) x_{ik}$. The unsatisfied load demand or load shedding needed

for island k is $\sum_{i\in I} P_{S_i} x_{ik}$, and summing over all islands, we have

$$\sum_{k=1}^{K}\sum_{i\in I} P_{S_i} x_{ik} = \sum_{i\in I} P_{S_i} \sum_{k=1}^{K} x_{ik} = \sum_{i\in I} P_{S_i},$$

where $\sum_{k=1}^{K} x_{ik} = 1$ is from (5–3). With the penalty cost coefficients, we have the objective function (5–1).

5.3.2 Islanding with Minimum Size

The complete islanding model (5–1)-(5–4) partitions a power grid into K islands. It is possible that the partition consists of some small islands, e.g., an island of one generator and one load bus. An extremely small island may provide little value in practice and may be considered as ill-conditioned in some real applications. One way to control the size of islands is to impose a lower bound $Minsize$. We can add the following constraint

$$\sum_{i\in I} x_{ik} \geq MinSize, \tag{5–6}$$

which controls the number of total buses in each island and call the islanding model with lower bound as **GI-M-DC-OPF**.

5.3.3 Models for Weakly Connected Islanding

In **GI-DC-OPF**, islands are completely separated. In the case of weakly connected islands [111], inter-island flows can be modeled by adding the constraints

$$C(k, k') \leq \varepsilon, \forall k, k', k \neq k', \tag{5–7}$$

where $\varepsilon(> 0)$ is the amount of the flow allowed between different islands. The islanding with weak connection can be formulated by adding constraint (5–7) and removing variables z_{ij}s and the associated constraints from **GI-DC-OPF**. The resulting weakly connected islanding model is **GI-W-DC-OPF**.

The term $C(k, k')$ describes the power crossing between island k and k' and can be measured in different ways. In [111], three methods are used to measure power line

capacities between islands. In this chapter, the actual power flow between two islands is used to measure the inter-island power flows. Assume $F_{kk'}$ is the positive power flow from island k to k', and $F_{k'k}$ is the total positive power flow on lines from k' to k. The three methods for measuring inter-island flows $C(k, k')$ are

I. $C^1(k, k') = F_{kk'} + F_{k'k}$

II. $C^2(k, k') = |F_{kk'} - F_{k'k}|$

III. $C^3(k, k') = \max\{F_{kk'}, F_{k',k}\}$.

Among these three measures, $C^1(k, k')$ is the most conservative measure. Given any feasible partition, the relation is $C^1(k, k') \geq C^3(k, k') \geq C^2(k, k')$.

The measure I. and II. are adopted from [111] by using actual power flow instead of the line capacity. In this chapter, we focus on $C^1(k, k')$ and $C^2(k, k')$ and add them to **GI-DC-OPF** for weakly connected islanding. The measure $C^3(k, k')$ counts maximum directed flow, which can be used to restrict angle differences between two islands [40].

Measure I. counts the absolute amount of flow between two islands in both directions, and we can explicitly express constraints (5–7) as

$$\begin{cases} C^1(k, k') = \sum_{(i,j)\in L} \alpha_{ij,kk'} \leq \varepsilon \\ \alpha_{ij,kk'} \geq P_{ij}(x_{ik}x_{jk'} + x_{ik'}x_{jk}), \quad \forall (i,j) \in L, k \neq k' \\ \alpha_{ij,kk'} \geq -P_{ij}(x_{ik}x_{jk'} + x_{ik'}x_{jk}), \quad \forall (i,j) \in L, k \neq k'. \end{cases} \tag{5–8}$$

Variable $\alpha_{ij,kk'}$ is used to count the absolute amount of power flow on line (i, j) crossing island k and k'. For (i, j) between island k and k', effectively, $\alpha_{ij,kk'} \geq |P_{ij}|$. Effectively, $\sum_{(i,j)\in L} \alpha_{ij,kk'} \leq \varepsilon$ implies $\sum_{(i,j)\in L} |P_{ij}| \leq \varepsilon$, and $\sum_{(i,j)\in L} |P_{ij}| \leq \varepsilon$ implies there exists a feasible assignment of $\alpha_{ij,kk'}$ such that $\sum_{(i,j)\in L} \alpha_{ij,kk'} \leq \varepsilon$. Therefore, constraints (5–8) is equivalent to the constraints $C^1(k, k') \leq \varepsilon$.

In measure II., the difference in power flow between two directions is used to measure the power transferred between two islands. The constraints for this method are

$$C^2(k,k') = |\sum_{(i,j)\in L} P_{ij}(x_{ik}x_{jk'} + x_{ik'}x_{jk})| \leq \varepsilon$$

and equivalent to

$$-\varepsilon \leq \sum_{(i,j)\in L} P_{ij}(x_{ik}x_{jk'} + x_{ik'}x_{jk}) \leq \varepsilon \tag{5–9}$$

for pairs k, k' and $k \neq k'$.

5.3.4 Model Preprocessing

The islanding models discussed in Section 5.3.1 are mixed integer nonlinear programs. In this section, we show that these models can be transformed to linear MIPs which have a wide range of solution techniques.

In the formulation for model **GI-DC-OPF**, there are the following quadratic terms:

$$x_{ik}x_{jk}, \quad y_k x_{ik}, \quad y_k x_{jk}$$

Besides these nonlinear terms, the model **GI-W-DC-OPF** has additional nonlinear terms:

$$x_{ik'}x_{jk}, \quad P_{ij}x_{ik}x_{jk'}, \quad P_{ij}x_{ik'}x_{jk}$$

To linearize these terms, we can define auxiliary variables, for example, $w_{ijkk'} = x_{ik}x_{jk'}$, $u_{ki} = y_k x_{ik}$ and $v_{kj} = y_k x_{jk}$. The variable $w_{ijkk'}$ is the product of two binary ones, and another two u_{ki}, v_{kj} are the product of a binary variable and a continuous one. They can be linearized by the following:

$$\begin{cases} w_{ijkk'} \leq x_{ik} \\ w_{ijkk'} \leq x_{jk'} \\ w_{ijkk'} \geq x_{ik} + x_{jk'} - 1 \\ w_{ijkk'} \geq 0 \end{cases} \tag{5–10}$$

$$\begin{cases} u_{ki} \geq y_k - U(1 - x_{ik}) \\ u_{ki} \geq V x_{ik} \\ u_{ki} \leq y_k - V(1 - x_{ik}) \\ u_{ki} \leq U x_{ik} \end{cases} \tag{5–11}$$

$$\begin{cases} v_{kj} \geq y_k - U(1 - x_{jk}) \\ v_{kj} \geq V x_{jk} \\ v_{kj} \leq y_k - V(1 - x_{jk}) \\ v_{kj} \leq U x_{jk} \end{cases} \tag{5–12}$$

where $V \leq y_k \leq U$ and V, U can be chosen as sufficient small, sufficient large numbers, respectively. For example, we can set $V = 1$ and $U = |I|$.

The terms $x_{ik}x_{jk}, x_{ik'}x_{jk}$ can be linearized analogously to $x_{ik}x_{jk'}$. After the linearization of $x_{ik}x_{jk'}$ by the variable $w_{ijkk'}$, the cubic term $P_{ij}x_{ik}x_{jk'}$ becomes $P_{ij}w_{ijkk'}$ which itself can be linearized in a similar way via new variable $q_{ijkk'}$ by

$$\begin{cases} q_{ijkk'} \geq P_{ij} - P_{ij_{max}}(1 - w_{ijkk'}) \\ q_{ijkk'} \geq -P_{ij_{max}} w_{ijkk'} \\ q_{ijkk'} \leq P_{ij} + P_{ij_{max}}(1 - w_{ijkk'}) \\ q_{ijkk'} \leq P_{ij_{max}} w_{ijkk'}. \end{cases} \tag{5–13}$$

Similarly, we can linearize $P_{ij}x_{ik'}x_{jk}$. With the above linearizations, the islanding problems can now be formulated as integer linear programs. A veriety of commercial software, such as CPLEX, can solve such problems efficiently. In addition, algorithms for large scale optimization, such as branch-and-cut or Benders' decomposition can also be used to solve MIPs.

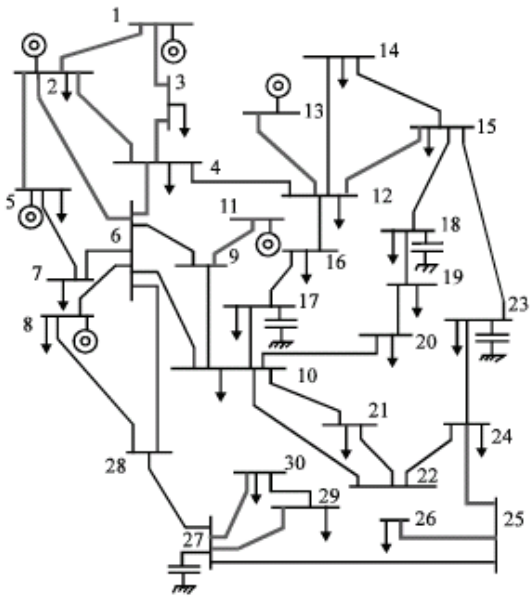


Figure 5-1. IEEE-30-Bus network

5.4 Numerical Experiments

The MIP formulation for the islanding problems were implemented in C++ and using CPLEX 11.0 via ILOG Concert Technology 2.5, and all computations were performed on a SUN UltraSpace-III with a 900 MHz processor and 2.0 GB RAM. The gap was set to be 0.01 in CPLEX.

We use the IEEE-30-Bus Test Case [102] (see Fig. 5-1) as the underlying power grid in the simulation. This network has 30 buses and 41 transmission lines. The data for our **GI-DC-OPF**, **GI-M-DC-OPF**, and **GI-W-DC-OPF** models are shown in Table 5-1 and Table 5-2.

Table 5-1. Generation and load data of IEEE-30-Bus network

Bus No.	**Load** D_i	**Generation** $\bar{G}_i$	**Shedding Cost** r_i
1	0	15	0
2	0	25	0
3	2.4	0	3
4	7.6	0	3
5	0	15	0
6	0	0	0
7	22.8	0	3
8	0	15	0
9	0	0	0
10	5.8	0	3
11	0	30	0
12	11.2	0	3
13	0	30	0
14	6.2	0	3
15	8.2	0	3
16	3.5	0	3
17	9	0	3
18	3.2	0	3
19	9.5	0	3
20	2.2	0	3
21	17.5	0	3
22	0	0	0
23	3.2	0	3
24	8.7	0	3
25	0	0	0
26	3.5	0	3
27	0	0	0
28	0	0	0
29	2.4	0	3
30	10.6	0	3

Note: In this table, if the generation capacity $\bar{G}_i$ for some bus is positive, this bus is a generating bus; similarly, if $D_i > 0$, this bus is a load bus.

By the complete islanding model **GI-DC-OPF** we list the results in Table 5-3 for different cases. In the case of $K = 1$, we consider the whole system as one island. Since there are at most 6 generators, we can obtain at most 6 islands in this system. In Fig. 5-2, the relationship between load shedding cost and the number K of islands is presented. From this figure, we can see that as the number of islands in the same

Table 5-2. Transmission line data of IEEE-30-Bus network

From Bus	To Bus	Capacity	Susceptance
1	2	130	15.65
1	3	130	5.63
2	4	165	5.20
3	4	130	23.53
2	5	130	4.77
2	6	165	5.12
4	6	190	22.31
5	7	170	7.45
6	7	130	11.03
6	8	130	22.01
6	9	165	4.81
6	10	130	1.80
9	11	165	4.81
9	10	165	9.09
4	12	165	3.91
12	13	165	7.14
12	14	130	3.17
12	15	130	6.10
12	16	130	4.10
14	15	160	2.25
16	17	160	4.84
15	18	160	3.69
18	19	160	6.22
19	20	130	11.76
10	20	130	3.99
10	17	130	10.32
10	21	130	10.98
10	22	130	5.40
21	22	130	34.13
15	23	160	3.98
22	24	160	3.95
23	24	160	2.99
24	25	160	2.29
25	26	160	1.82
25	27	160	3.76
28	27	165	2.53
27	29	160	1.88
27	30	160	1.29
29	30	160	1.72
8	28	160	4.54
6	28	130	15.46

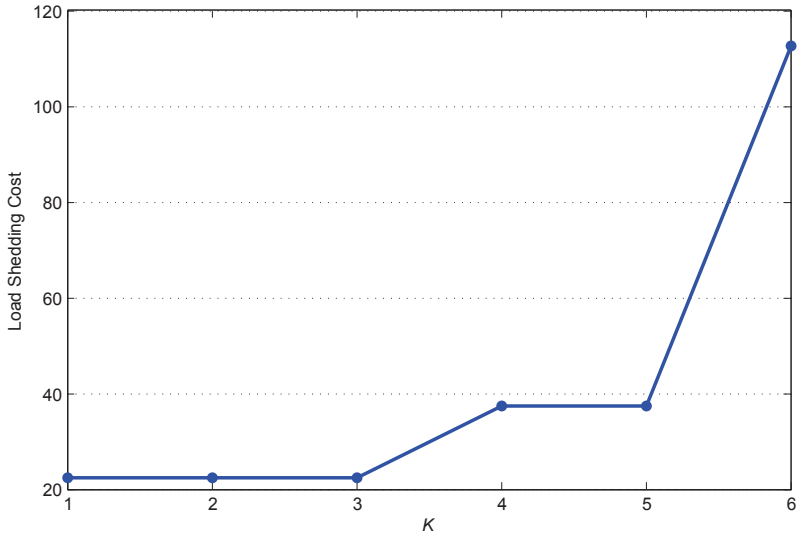


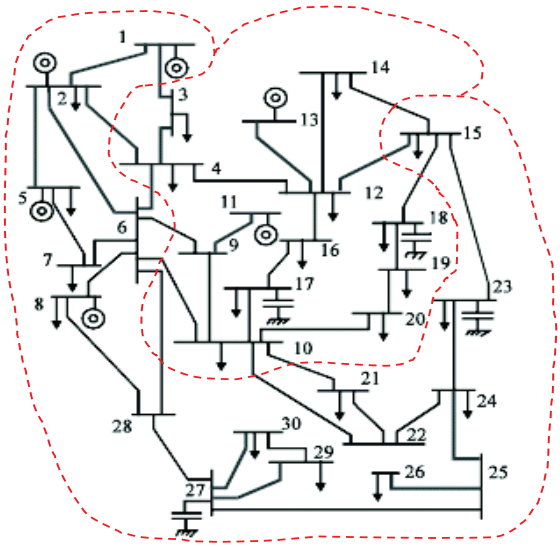
Figure 5-2. Load shedding cost vs. the number K of islands

system increases, the total load shedding cost increases as well. From Table 5-3, most obtained islands have high satisfied demand rates.

Based on the results in Table 5-3, we show three cases of completely divided islands for $K = 2, 3, 4$ in Fig. 5-3, Fig. 5-4, and Fig. 5-5. As shown in these figures, each island is connected and includes at least one generator.

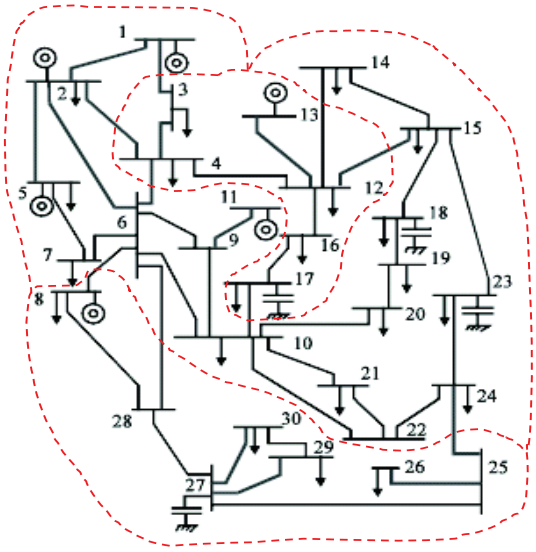
As discussed in Section 5.3.2, to control the size of islands, the contraint (5–6) is added to complete islanding model to form the model **GI-M-DC-OPF**. Here, using same input parameters as Table 5-3, the relationship, obtained by the model **GI-M-DC-OPF**, between load shedding cost and *MinSize* for $K = 2$ and $K = 3$ is presented in Fig. 5-6. For $K = 2$, the maximum *MinSize* is 15, and for $k = 3$, the maximum *MinSize* is 10. As shown in Fig. 5-6, when *MinSize* is small (eg., $MinSize \leq 7$), both costs are the same. When *MinSize* increases, the load shedding cost increases as well but more significantly for $K = 3$.

Next, we tested the model **GI-W-DC-OPF** for islanding in case of weak connection without the *MinSize* constraint. Among islands, as discussed in Section 5.3.3, limited flows are allowed. In the following, we are going to test the model **GI-W-DC-OPF** in case



($K = 2$, Root Buses 1,13)

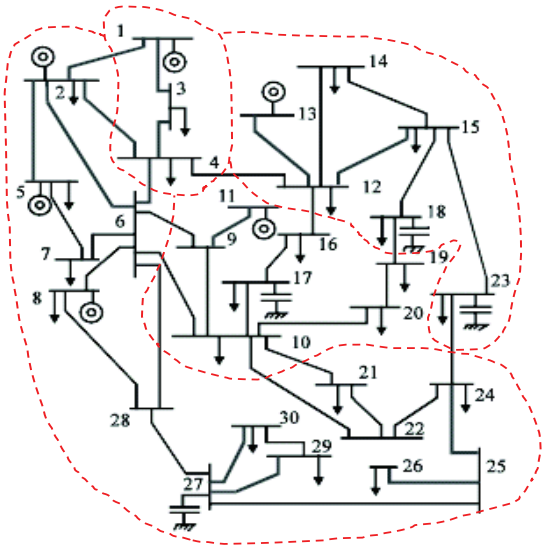
Figure 5-3. IEEE-30-Bus network with 2 islands



($K = 3$, Root Buses 1,8,13)

Figure 5-4. IEEE-30-Bus network with 3 islands

of the most conservative measure $C^1(k, k')$ and the most loose measure $C^2(k, k')$ for weakly connected islanding. In Fig. 5-7, the relationship between load shedding cost and ε under two measures $C^1(k, k')$ and $C^2(k, k')$ is presented for the case of $K = 4$ with root buses $1, 8, 11, 13$, and 4 fixed islands as shown in Fig. 5-5. Additionally, the



($K = 4$, Root Buses 1,8,11,13)

Figure 5-5. IEEE-30-Bus network with 4 islands

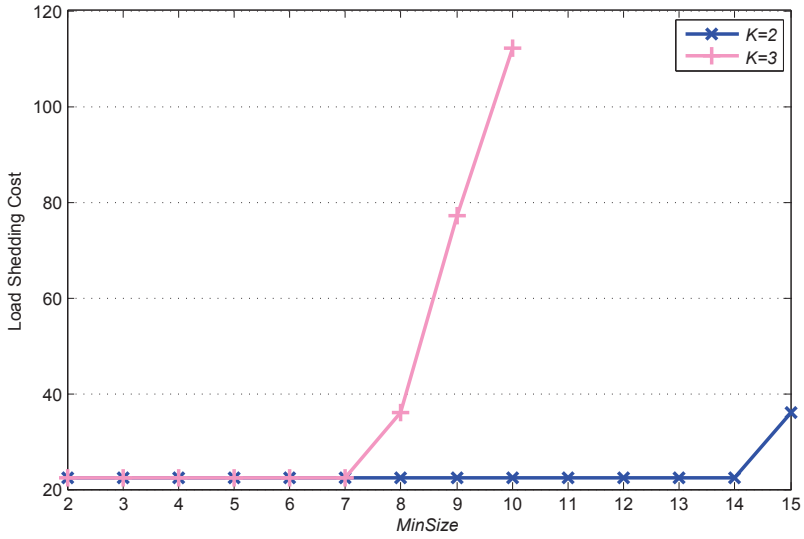
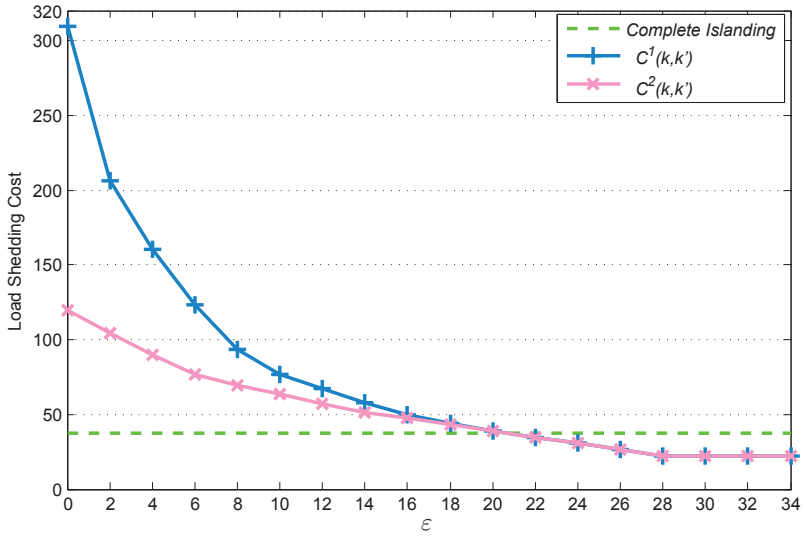


Figure 5-6. Load shedding cost vs. *MinSize*

cost for complete islanding is present as a dashed line. From this figure, we can see load shedding costs for both cuts are monotonically decreasing as ε increases. When ε is large enough to allow large flows among islands, costs are a constant for both cuts. The cost under $C^1(k, k')$ is higher than $C^2(k, k')$, since he cut $C^2(k, k')$ is looser than $C^1(k, k')$ as discussed in 5.3.3.



($K = 4$, Root Buses 1,8,11,13)

Figure 5-7. Load shedding cost vs. ε

In addition, from Fig. 5-7, when $\varepsilon < 20$, the cost under both cuts is larger than the cost of complete islanding. This shows that inter-island flows do not always help power grid performance and in many cases, completely disconnected islands perform much better than weakly connected islands. In complete islanding model **GI-DC-OPF**, there is a constraint $P_{ij} = B_{ij}(\theta_i - \theta_j)$ for line (i, j). When this line is chosen into the cut, $z_{ij} = 0$ and there is no requirement for the difference $\theta_i - \theta_j$. However, in the weekly connected islanding model **GI-W-DC-OPF**, the constraint for line (i, j) becomes $P_{ij} = B_{ij}(\theta_i - \theta_j)$. When this line is chosen into the cut, only limited flow is allowed on this line; when ε becomes closer to 0, this forces $P_{ij} = 0$ and $\theta_i = \theta_j$, which will influence other part of the system and increase the load shedding cost.

5.5 Discussion

In this chapter, we use mixed integer programs to formulate the power grid islanding problem. Our model can form multiple islands simultaneously and can be used to form weakly connected islands. To solve the nonlinear MIPs, we present techniques to transform them into linear MIPs. Our models propose a scheme to form self-sufficient

subnetworks in a large-scale power system and also to minimize the load shedding.

Future research includes following directions by:

- changing the based model from DC-OPF to AC-OPF.
- adding security and stability constraints to our current model, for example, line power flow by $N-1, N-k$ contingency analysis, current thermal limit, and bus voltage stability limit to ensure the system works normally.
- adding the generation and load demand balance constraints. For each island, there is a ratio on load shedding to prevent blackouts.
- adding physical location constraints. For example, physically close buses should be divided into one island to reduce transmission cost.

Table 5-3. Islanding results by complete islanding model

K	Root Buses	Obj.	Islands	Real Generation	Gen. Capacity	Load Demand	Sat. Demand
1	1	22.5	Island 1: 1-30	130.0	130.0	137.5	94.5%
2	1,13	22.5	Island 1: 1,2,5-8,15,21-30	70.0	70.0	76.9	91.0%
			Island 2: 3,4,9-14,16-20	60.0	60.0	60.6	99.0%
3	1,8,13	22.5	Island 1: 1,2,5-7,9-11,14,15,18-24	85.0	85.0	87.3	97.4%
			Island 2: 8,25-30	15.0	15.0	16.5	90.9%
			Island 3: 3,4,12,13,16,17	30.0	30.0	33.7	89.0%
4	1,8,11,13	37.5	Island 1: 1,3,4	10.0	15.0	10.0	100.0%
			Island 2: 2,5-8,21,22,24-30	55.0	55.0	65.5	84.0%
			Island 3: 9-11,16,17,19,20	30.0	30.0	30.0	100.0%
			Island 4: 12-15,18,23	30.0	30.0	32.0	93.8%
5	1,5,8,11,13	37.5	Island 1: 1,3,4	10.0	15.0	10.0	100.0%
			Island 2: 5,7	15.0	15.0	22.8	65.8%
			Island 3: 8,25-30	15.0	15.0	16.5	90.9%
			Island 4: 2,6,9-11,17,19-24	55.0	55.0	55.9	98.4%
			Island 5: 12-16,18	30.0	30.0	32.3	92.9%
6	1,2,5,8,11,13	112.5	Island 1: 1,3	2.4	15.0	2.4	100.0%
			Island 2: 2,4	7.6	25.0	7.6	100.0%
			Island 3: 5,7	15.0	15.0	22.8	65.8%
			Island 4: 8,25-30	15.0	15.0	16.5	90.9%
			Island 5: 6,9-11,21-24	30.0	30.0	35.2	85.2%
			Island 6: 12-20	30.0	30.0	53.0	56.6%

Note: In this table, no flow is allowed among different islands. The value K is the number of islands to form. The Obj. denotes the objective cost of the model **GI-DC-OPF** based on the data in Table 5-1 and Table 5-2. The Root Buses denotes the picked root bus for each island. Real Generation is the total generation at each island. Gen. Capacity and Load Demand are the total generation capacity, total load demand for each divided island. The Sat. Demand denotes the satisfied rate, which is the rate of satisfied demand divided by total load demand for each each island.

CHAPTER 6
CONCLUSIONS

In the dissertation, we mainly studied combinatorial and nonlinear optimization methods and their applications in data clustering, biclustering and power systems for islanding. The mathematical programming approach we used in the dissertation includes integer programming, linear programming, semidefinite programming, quadratically constrained programming, stochastic programming and etc. Spectral, interior point, decomposition and linearization methods are all used for solving these formulations.

In the future, more decomposition methods, for example, Benders decomposition method and L-shaped method, can be used for solving the robust and stochastic optimization models mentioned in the dissertation. The semidefinite programming method can be combined to these decomposition approaches so that more efficient algorithms can be constructed. In the application of power systems islanding, our proposed model is based on DC optimal power flow model. The islanding model based on AC optimal power flow by adding more security and stability constraints, can be constructed as a mixed integer nonlinear program. This requires more efficient algorithms for solving problems arising from large power systems.

REFERENCES

[1] Alpert, C.J., Kahng, A.B.: Recent directions in netlist partitioning: a survey. Integr. VLSI J. **19**(1–2), 1–81 (1995)

[2] Angiulli, F., Cesario, E., Pizzuti, C.: Random walk biclustering for microarray data. Inf. Sci.: Int. J. **178**(6), 1479–1497 (2008)

[3] Armbruster, M., Fugenschuh, M., Helmberg, C., Martin, A.: A comparative study of linear and semidefinite branch-and-cut methods for solving the minimum graph bisection problem. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) Proceedings of the 13th International Integer Programming and Combinatorial Optimization Conference, pp. 112–124. Lecture Notes in Computer Science 5035, Springer (2008)

[4] Barkow, S., et al.: BicAT: A biclustering analysis toolbox. Bioinformatics **22**, 1282–1283 (2006)

[5] Ben-Dor, A., Chor, B., Karp, R., Yakhini, Z.: Discovering local structure in gene expression data: The order-preserving submatrix problem. J. Comput. Biol. **10**, 373–384 (2003)

[6] Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik **4**, 238–252 (1962)

[7] Berkhin, P.: Survey Of clustering data mining techniques. Technical Report. Accrue Software (2002)

[8] Bertsimas, D., Sim, M.: Robust discrete optimization and network flows. Math. Program. Ser. B **98**, 49–71 (2003)

[9] Bertsimas, D., Sim, M.: The price of robustness. INFORMS Oper. Res. **52**(1), 35–53 (2004)

[10] Bezdek, J.C.: Pattern recognition with fuzzy objective function algorithms. Plenum, New York (1981)

[11] Bi, T., Ni, Y., Shen, C.M., Wu, F.F.: Efficient multiway graph partitioning method for fault section estimation in large-scale power networks. IEE Proc.-Gener. Trunsm. Distrib. **149**(3), 289–294 (2002)

[12] Birge, J.R., Louveaux, F.: Introduction to Stochastic Programming, Springer Verlag, New York (1997)

[13] Blumsack, S.: Partitioning of electrical networks. http://cnls.lanl.gov/∼chertkov/SmarterGrids/seminars.html (2009). Accessed 6 June 2011

[14] Boulle, M.: Compact mathematical formulation for graph partitioning. Optim. Eng. **5**, 315–333 (2004)

[15] Burer, S., Monteiro, R.D.C., Zhang, Y.: Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs. SIAM J. Optim. **12**, 503–521 (2001)

[16] Busygin, S., Boyko, N., Pardalos, P., Bewernitz, M., Ghacibehc, G.: Biclustering EEG data from epileptic patients treated with vagus nerve stimulation. In: AIP Conference Proceedings of the Data Mining, Systems Analysis and Optimization in Biomedicine, pp. 220–231 (2007)

[17] Busygin, S., Prokopyev, O.A., Pardalos, P.M.: Feature selection for consistent biclustering via fractional 0–1 programming. J. Comb. Optim. **10**(1), 7-21 (2005)

[18] Busygin, S., Prokopyev, O.A., Pardalos, P.M.: Biclustering in data mining. Comput. Oper. Res. **35**, 2964–2987 (2008)

[19] Califano, A., Stolovitzky, G., Tu, Y.: Analysis of gene expression microarrays for phenotype classification. In: Proceedings of International Conference on Computational Molecular Biology, 75–85 (2000)

[20] Carmona-Saez, P., Pascual-Marqui, R.D., Tirado, F., Carazo, J.M., Pascual-Montano, A.: Biclustering of gene expression data by non-smooth non-negative matrix factorization. BMC Bioinformatics **7**, 78 (2006)

[21] Chan, P.K., Schlag, M., Zien, J.Y.: Spectral k-way ratio-cut partitioning and clustering. IEEE Trans. Comput-Aided Des. Integr. Circuits Syst. **13**, 1088–1096 (1994)

[22] Chaovalitwongse, W.A., Pardalos, P.M., Prokopyev, O.A.: A new linearization technique for multi-quadratic 0–1 programming problems. Oper. Res. Lett. **32**, 517–522 (2004)

[23] Chardaire, P., Sutter, A.: A decomposition method for quadratic zero-one programming. Manag. Sci. **41**(4), 704–712 (1995)

[24] Cheng, K.O., et al.: Bivisu: Software tool for bicluster detection and visualization. Bioinformatics **23**, 2342–2344 (2007)

[25] Cheng, Y., Church, G.M.: Biclustering of expression data. In: Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, pp. 93–103 (2000)

[26] Cho, H., Dhillon, I.S.: Coclustering of human cancer microarrays using minimum sum-squared residue coclustering. IEEE/ACM Trans. Comput. Biol. Bioinform. **5**(3), 385–400 (2008)

[27] Chung, F.R.K.: Spectral graph theory. In: Regional conference series in mathematics by conference board of the mathematical sciences. American Mathematical Society, Providence (1997)

[28] ILOG CPLEX 11.0 Users Manual, (2007)

[29] Data Transformation Steps. http://www.dmg.org/v2-0/Transformations.html (2008). Accessed 6 June 2011

[30] Dhillon, I.S.: Co-clustering documents and words using bipartite spectral graph partitioning. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledging Discovery and Data Mining (KDD), pp. 26–29 (2001)

[31] Dhillon, I.S., Mallela, S., Modha, D.S.: Information theoretic co-clustering. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 89–98 (2003)

[32] Dilkina, B., Gomes, C.P.: Solving connected subgraph problems in wildlife conservation. In: CPAIOR-10: 7th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Bologna, Italy (2010)

[33] DiMaggio, P.A., McAllister, S.R., Floudas, C.A., Feng, X.J., Rabinowitz, J.D., Rabitz, H.A.: Biclustering via optimal re-ordering of data matrices in systems biology: Rigorous methods and comparative studies. BMC Bioinformatics **9**, 458 (2008)

[34] Ding, C.: A Tutorial on Spectral Clustering. http://ranger.uta.edu/∼chqding/Spectral (2004). Accessed 6 June 2011

[35] Ding, C., He, X., Zha, H., Gu, M., Simon, H.: A min-max cut algorithm for graph partitioning and data clustering. In: Proc. IEEE Int'l Conf. Data Mining, (2001)

[36] Du, P., Nelson, J.K.: Two-step solution to optimal load shedding in a micro-grid. IEEE/PES Power Syst. Conf. and Expo. (2009)

[37] Elsner, U.: Graph partitioning–A survey. Technische Universitat Chemnitz, SFB393-Preprint 97–27 (1997)

[38] Fan, N., Boyko, N., Pardalos, P.M.: Recent advances of data biclustering with application in computational neuroscience. In: Chaovalitwongse, W.A., Pardalos, P.M., Xanthopoulos, P. (eds.) Computational Neuroscience, pp. 105–132. Springer, Berlin, (2010)

[39] Fan, N., Chinchuluun, A., Pardalos, P.M. : Integer programming of biclustering based on graph models. In: Chinchuluun, A., Pardalos, P.M., Enkhbat, R., Tseveendorj, I. (eds.) Optimization and Optimal Control: Theory and Applications, pp. 479–498. Springer, Berlin (2010)

[40] Fan, N., Pan, F.: Locating phasor measurements and detecting cutset angles in power systems. In: IEEE PES Innovative Smart Grid Technologies (ISGT 2011) Conference, Anaheim, CA (2011)

[41] Fan, N., Pardalos, P.M., Chinchuluun, A.: Graph partitioning approaches for analyzing biological networks. In: R.P. Mondaini (eds.) BIOMAT 2009-International Symposium on Mathematical and Computational Biology, pp. 250–262. World Scientific, Singapore (2009)

[42] Fan, N., Pardalos, P.M.: Linear and quadratic programming approaches for the general graph partitioning problem. J. Glob. Optim. **48**(1), 57–71 (2010)

[43] Fan, N., Pardalos, P.M.: Multi-way clustering and biclustering by the Ratio cut and Normalized cut in graphs. J. Comb. Optim. (2010). doi:10.1007/s10878-010-9351-5

[44] Fan, N., Pardalos, P.M.: Robust optimization of graph partitioning and critical node detection in analyzing networks. In: Wu, W., Daescu, O. (eds.) COCOA 2010, Part I, pp. 170–183. Lecture Notes in Computer Science Vol. 6508 (2010)

[45] Fan, N., Zheng, Q.P., Pardalos, P.M.: On the two-stage stochastic graph partitioning problem. In: Wang, W., Zhu, X., Du, D.-Z. (eds.) COCOA 2011, pp. 500–509, Lecture Notes in Computer Science Vol. 6831 (2011)

[46] Falkner, J., Rendl, F., Wolkowiz, H.: A computational study of graph partitioning. Math. Program. **66**, 211–240 (1994)

[47] Fiedler, M.: A property of eigenvectors of non-negative symmetric matrices and its application to graph theory. Czech. Math. J. **25**, 619–633 (1975)

[48] Fjallstrom, P.-O.: Algorithms for graph partitioning: a survey. Linkop. Elec. Articles Comput. Inf. Sci. **3**, 10 (1998)

[49] Frieze, A., Flaxman, A., Krivelevich, M.: On the random 2-stage minimum spanning tree. Random Structures and Algorithms **28**, 24–36 (2006)

[50] Frieze, A., Jerrum, M.: Improved approximation algorithms for max k-cut and max bisection. Algorithmica **18**, 67–81 (1997)

[51] Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified NP-complete graph problems. Theor. Comput. Sci. **1**, 237–267 (1976)

[52] Grant, M., Boyd, S.: Graph implementations for nonsmooth convex programs. In: Blondel, V., Boyd, S., Kimura, H. (eds.) Recent Advances in Learning and Control (a tribute to M. Vidyasagar), pp. 95–110. Lecture Notes in Control and Information Sciences. Springer, Berlin (2008)

[53] Grant, M., Boyd, S.: CVX: Matlab software for disciplined convex programming (web page and software). http://stanford.edu/∼boyd/cvx (2009). Accessed 6 June 2011

[54] Gu, J., Liu, J.S.: Bayesian biclustering of gene expression data. BMC Genom. **9**(Suppl 1), S4 (2008)

[55] Gu, M., Zha, H., Ding, C., He, X., Simon, H.: Spectral relaxation models and structure analysis for k-way graph Clustering and bi-clustering. Penn State Univ Tech Report CSE-01-007 (2001)

[56] Hagen, L., Kahng, A.B.: New spectral methods for ratio cut partitioning and clustering. IEEE Trans. on Computer-Aided Des. **11**(9), 1074–1085 (1992)

[57] Hager, W., Krylyuk, Y.: Graph partitioning and continuous quadratic programming. SIAM J. Discret. Math. **12**(4), 500–523 (1999)

[58] Hager, W., Krylyuk, Y.: Multiset graph partitioning. Math. Meth. Oper. Res. **55**, 1–10 (2002)

[59] Hager, W., Phan, D., Zhang, H.: An exact algorithm for graph partitioning. arXiv:0912.1664v1 (2009)

[60] Hartigan, J.A.: Direct clustering of a data matrix. J. Am. Stat. Assoc. **67**, 123–129 (1972)

[61] Hendrickson, B., Kolda, T.G.: Graph partitioning models for parallel computing. Parallel Comp. **26**, 1519–1534 (1999)

[62] Hendrickson, B., Leland, R.: A multilevel algorithm for partitioning graphs. Tech. Rep. SAND93-1301, Sandia National Laboratory (1993)

[63] Hendrickson, B., Leland, R.: An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM J. Sci. Comput. **16**, 452–469 (1995)

[64] Huneault, M., Galiana, F.D.: A survey of the optimal power flow literature. IEEE Trans. Power Syst. **6**(2), 762–770 (1991)

[65] IEEE/CIGRE Joint Task Force on Stability Terms and Definitions.: Definition and classification of power system stability. IEEE Trans. Power Syst. **19**(2), 1387–1401 (2004)

[66] IEEE Std 1547-2003. IEEE Standard for Interconnecting Distributed Resources with Electric Power Systems (2003)

[67] Ihmels, J., Friedlander, G., Bergmann, S., Sarig, O., Ziv, Y., Barkai, N.: Revealing modular organization in the yeast transcriptional network. Nat. Genet. **31**(4), 370–377 (2002)

[68] Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Comput. Sur. **31**(3), 264–323 (1999)

[69] Kaiser, S., Leisch, F.: A toolbox for bicluster analysis in r. Tech. Rep. 028, Ludwing-Maximilians-Universitat Mnchen (2008)

[70] Karisch, S.E., Rendl, F.: Semidefinite programming and graph equipartition. In: Pardalos, P.M., Wolkowicz, H. (eds) Topics in Semidefinite and Interior-Point Methods, pp. 77–95. American Mathmatical Society, Providence (1998)

[71] Karisch, S.E., Rendl, F., Clausen, J.: Solving graph bisection problems with semidefinite programming. INFORMS J. Comp. **12**, 177-191 (2000)

[72] Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S.: Multilevel hypergraph partitioning: applications in VLSI domain. IEEE Trans. Very Larg. Scale Integr. (VLSI) Syst. **7**(1), 69–79 (1999)

[73] Karypis, G., Kumar, V.: Parallel multilevel graph partitioning. In: 10th International Parallel Processing Symposium (IPPS '96), 314 (1996)

[74] Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. **20**, 359–392 (1998)

[75] Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. **49**(1), 291–307 (1970)

[76] Klerk, E.: Aspects of semidefinite programming: Interior point algorithms and selected applications. Kluwer Academic, Dordrecht (2002)

[77] Kluger, Y., Basri, R., Chang, J.T., Gerstein, M.: Spectral biclustering of microarray cancer data: Co-clustering genes and conditions. Genome Res. **13**, 703–716 (2003)

[78] Kochenberger, G.A., Glover, F., Alidaee, B., Rego, C.: An unconstrained quadratic binary programming approach to the vertex coloring problem. Ann. Oper. Res. **139**(1), 229–241 (2005)

[79] Kong, N., Schaefer, A.J.: A factor 1/2 approximation algorithm for two-stage stochastic matching problems. Eur. J. Oper. Res. **172**, 740–746 (2006)

[80] Kouvelis, P., Yu, G.: Robust discrete optimization and its applications. Kluwer Academic Publishers, London (1997)

[81] Lazzeroni, L., Owen, A.: Plaid models for gene expression data. Stat. Sinica **12**, 61–86 (2002)

[82] Lee, D.D., Seung, H.S.: Learning the parts of objets by non-negative matrix factorization. Nature, **401**, 788–791 (1999)

[83] Li, J., Liu, C.-C.: Power system reconfiguration based on multilevel graph partitioning. In: Proceedings of 2009 IEEE Bucharest Power Tech Conference, Burcharest, Romania (2009)

[84] Lisser, A., Rendl, F.: Graph partitioning using linear and semidefinite programming. Math. Program. Ser. B **95**, 91–101 (2003)

[85] Liu, Y., Liu, Y.: Aspects on power system islanding for preventing widespread blackout. In: Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control (ICNSC '06), pp. 1090–1095 (2006)

[86] Lodi, A., Allemand, K., Liebling, T.M.: An evolutionary heuristic for quadratic 0–1 programming. Eur. J. Oper. Res. **119**(3), 662–670 (1999)

[87] Loiola, E.M., deAbreu, N.M.M., Boaventura-Netto, P.O.,Hahn, P.,Querido, T.:A survey for the quadratic assignment problem. Eur. J. Oper. Res. **176**(2), 657–690 (2007)

[88] MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley symposium on mathematical statistics and probability, pp. 281–297. University of California Press, Berkeley (1967)

[89] Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: A survey. IEEE Trans. Comput. Biol. Bioinf. **1**(1), 24–45 (2004)

[90] Madeira, S.C., Oliveira, A.L.: A linear time biclustering algorithm for time series gene expression data. Lect. Notes Comput. Sci. **3692**, 39–52, (2005)

[91] Mitchell, J.: Branch-and-cut for the k-way equipartition problem, Tech. Rep., Department of Mathematical Sciences, Rensselaer Polytechnic Institute (2001)

[92] Montemanni, R., Gambardella, L.M.: An exact algorithm for the robust shortest path problem with interval data. Comput. Oper. Res. **1**(10), 1667–1680 (2004)

[93] Murali, T.M., Kasif, S.: Extracting conserved gene expression motifs from gene expression data. Pacific Symp. Biocomput. **8**, 77–88 (2003)

[94] Overton, M.L., Womersley, R.S.: On the sum of largest eigenvalues of symmetric matrix. SIAM J. Matrix Anal. Appl. **13**, 41–45 (1992).

[95] Pandya, K.S., Joshi, S.K.: A Survey of optimal power flow methods. J. Theor. Appl. Inf. Tech. 450–458 (2008)

[96] Pardalos, P.M., Busygin, S., Prokopyev, O.A.: On biclustering with feature selection for microarray data sets. In: Mondaini, R. (ed.) BIOMAT 2005-international Symposium on Mathematical and Computational Biology, pp. 367–378. World Scientific, Singapore (2006)

[97] Pardalos, P.M., Rodgers, G.P.: Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing **45**, 131–144 (1990)

[98] Pardalos, P.M., Tomaino, V., Xanthopoulos, P.: Optimization and data mining in medicine. Top **17**, 215–236 (2009)

[99] Peiavi, A., Ildarabadi, R.: A fast algorithm for intentional islanding of power systems using the multilevel Kernel k-means approach. J. Appl. Sci. **9**(12), 2247–2255 (2009)

[100] Pascual-Montano, A., Carazo, J.M., Kochi, K., Lehmann, D., Pascual-Marqui, R.D.: Nonsmooth Non-negative matrix factorization (nsNMF). IEEE Trans. Pattern Anal. Mach. Intell. **28**, 403–415 (2006)

[101] Pothen, A., Simon, H.D., Liou, K.P.: Partitioning sparse matrices with eigenvectors of graph. SIAM Journal of Matrix Anal. Appl., **11**, 430–452 (1990)

[102] Power Systems Test Case Archive. http://www.ee.washington.edu/research/pstca (2011). Accessed 6 June 2011

[103] Prelic, A., Bleuler, S., Zimmermann, P., Wille, A., Buhlmann, P., Gruissem, W., Hennig, L., Thiele, L., Zitzler, E.: A systematic comparison and evaluation of biclustering methods for gene expression data. Bioinformatics **22**(9), 1122–1129, (2006)

[104] Rege, M., Dong, M., Fotouhi, F.: Bipartite isoperimetric graph partitioning for data co-clustering. Data Min. Knowl. Discov. **16** 276–312 (2008)

[105] Reiss, D.J., Baliga, N.S., Bonneau, R.: Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. BMC Bioinformatics **7**, 280 (2006)

[106] Richards, A.L., Holmans, P.A., O'Donovan, M.C., Owen, M.J., Jones, L.: A comparison of four clustering methods for brain expression microarray data. BMC Bioinformatics **9**, 490 (2008)

[107] Ropp, M., Bower, W.: Evaluation of islanding detection methods for photovoltaic utility interactive power systems. In: Int. Energy Agency Implementing Agreement on Photovoltaic Power Syst. Tech. rep. IEA PVPS T5-09 (2002)

[108] Salmeron, J., Wood, K., Baldick, R.: Analysis of electric grid security under terrorist threat. IEEE Trans. Power Syst.**19**(2), 905–912 (2004)

[109] Santamaria, R., Theron, R., Quintales, L.: BicOverlapper: A tool for bicluster visualization Rodrigo. Bioinformatics **24**, 1212–1213 (2008)

[110] Schaeffer, S.E.: Survey: graph clustering. Comput. Sci. Rev. **1**, 27–64 (2007)

[111] Sen, A., Ghosh, P., Vittal, V., Yang, B.: A new min-cut problem with application to electric power network partitioning. Eur. Trans. Electr. Power **19**(6), 778–797 (2008)

[112] Senroy, N., Heydt, G.T., Vittal, V.: Decision tree assisted controlled islanding. IEEE Trans. Power Syst. **21**(4), 1790–1797 (2006)

[113] Sheng, Q., Moreau, Y., De Moor, B.: Biclustering microarray data by Gibbs sampling. Bioinformatics **19**, 196–205 (2003)

[114] Sherali, H.D., Smith, J.C.: An improved linearization strategy for zero-one quadratic programming problems. Optim. Lett. **1**, 33–47 (2007)

[115] Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. on Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)

[116] Soyster, A.L.: Convex programming with set-inclusive constraints and applications to inexact linear programming. Oper. Res. **21**, 1154–1157 (1973)

[117] Soper, A.J., Walshaw, C., Cross, M.: A combined multilevel search and multilevel optimization approach to graph-partition. J. Global Optim. **29**, 225–241 (2004)

[118] Sun, J., L.S., Tesfatsion.: DC optimal power flow formulation and solution using QuadProgJ. Economics Department, Iowa State University **6014**, 1–36 (2006)

[119] Supper, J., Strauch, M., Wanke, D., Harter, K., Zell, A.: EDISA: Extracting biclusters from multiple time-series of gene expression profiles. BMC Bioinformatics **8**, 334 (2007)

[120] Tanay, A., Sharan, R., Kupiec, M., Shamir, R.: Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. Proc. Natl. Acad. Sci. USA **101**, 2981–2986 (2004)

[121] Tanay, A., Sharan, R., Shamir, R.: Discovering statistically significant bilcusters in gene expression data. Bioinformatics **18**, S136–S144 (2002)

[122] Tanay, A., Sharan, R., Shamir, R.: Biclustering algorithms: A survey. In: Aluru, S. (ed.) Handbook of Computational Molecular Biology. Chapman Hall, London (2005)

[123] Tiptipakorn, S.: A spectral bisection partitioning method for electric power network applications. PhD Dissertation, Department of Electrical and Computer Engineering, University of Wisconsin (2001)

[124] Torres, F.E.: Linearization of mixed-integer products. Math. Program. **49**, 427–428 (1991)

[125] Vandenberghe, L., Boyd, S.: Semidefinite programming, SIAM Rev. **38**, 49–95 (1996)

[126] Vittal, V., Heydt, G.T., The problem of initiating controlled islanding of a large interconnected power system solved as a Pareto optimization. In: Proceedings of the 2009 IEEE PES Power System Conf. and Expo., Seattle, WA (2009)

[127] Walshaw, C.: Multilevel refinement for combinatorial optimization problems. Ann. Oper. Res. **131**, 325–372 (2004)

[128] Wang, X., Vittal, V.: System islanding using minimal cutsets with minimum net flow. In: Proceeding of the 2004 IEEE PES Power System Conf. and Expo., New York (2004)

[129] Wolkowicz, H., Zhao, Q.: Semidefinite programming relaxations for the graph partitioning problem. Discrete Appl. Math. **96-97**, 461–479 (1996)

[130] Wood, A.J., Wollenberg, B.F.: Power Generation, Operation and Control (2nd ed). Wiley, New York (1996)

[131] Xing, E.P., Jordan, M.I.: On semidefinite relaxation for normalized k-cut and connections to spectral clustering. UC Berkeley Technical Report CSD-03-1265 (2003)

[132] Xu, R., Wunsch, D.II.: Survey of clustering algorithms. IEEE Trans. Neural Netw. **16**(3), 645–678 (2005)

[133] Yaman, H., Karasan, O.E., Pinar, M.C.: The robust shortest spanning tree problem with interval data. Oper. Res. Lett. **29**, 31–40 (2001)

[134] Yang, B., Vittal, V., Heydt, G.T.: Slow-coherency-based controlled islanding - A demonstration of the approach on the August 14, 2003 blackout scenario. IEEE Trans. Power Syst. **21**(4), 1840–1847 (2006)

[135] Yang, J., Wang, W., Wang, H., Yu, P.: δ-Clusters: Capturing subspace correlation in a large data set. In: Proceedings of the 18th IEEE International Conference on Data Engineering, pp. 517–528 (2002)

[136] Yang, J., Wang, W., Wang, H., Yu, P.: Enhanced biclustering on expression data. In: Proceedings of the Third IEEE Conference on Bioinformatics and Bioengineering, pp. 321–327 (2003)

[137] You, H., Vittal, V., Wang, X.: Slow coherency based islanding. IEEE Trans. Power Syst. **19**(1), 483–491 (2004)

[138] You, H., Vittal, V., Yang, Z.: Self-healing in power systems: an approach using islanding and rate of frequency decline based load shedding. IEEE Trans. Power Syst. **18**(1), 174–181 (2003)

[139] Yusof, S.B., Rogers, G.J., Alden, R.T.H.: Slow coherency based network partitioning including load buses. IEEE Trans. Power Syst. **8**(3), 1375–1382 (1993)

[140] Zeineldin, H.H., Bhattacharya, K., El-Saadany, E.F., Salama, M.M.A.: Impact of intentional islanding of distributed generation on electrocity market prices. IEE Proc. Transm. Distrib. **153**(2), 147–154 (2006)

[141] Zha, H., He, X., Ding, C., Simon, H., Gu, M.: Bipartite graph partitioning and data clustering. In: Proceedings of the tenth international conference on Information and knowledge management, pp. 25–32. ACM Press, New York, NY (2001)

[142] Zhao, H., Liew, A.W.-C., Xie, X., Yan, H.: A new geometric biclustering based on the Hough transform for analysis of large-scale microarray data. J. Theor. Biol. **251**, 264–274 (2008)

BIOGRAPHICAL SKETCH

Neng Fan obtained his bachelor's degree in information and computational science from Wuhan University in Wuhan, China in 2004. Then, he received his master's degree in applied mathematics from Nankai University, Tianjin, China in 2007. In May 2009, he obtained a Master of Science in industrial and systems engineering from University of Florida. In August 2011, he graduated from the University of Florida with a PhD degree under the supervision of Dr. Panos M. Pardalos. During August 2010 to May 2011, he was a graduate research assistant at D-6 Risk Analysis and Decision Support Systems, Los Alamos National Laboratory.