# A Study on Relaxation Algorithms for Optimization Problems arising in Data Mining

**March 2016**

**Yoichi Izunaga**

# A Study on Relaxation Algorithms for Optimization Problems arising in Data Mining

**Graduate School of Systems and Information Engineering**

**University of Tsukuba**

**March 2016**

**Yoichi Izunaga**

# Abstract

This thesis is concerned with two major applications in data mining, community detection and ranking. The problems arising in the above applications can be described as a mathematical optimization problem, which, however, often ends up being intractable due to some theoretical difficulties and/or the scale of instances. Solution methods tailored to each application should be strongly desired. In this thesis, we study the solution methods based on relaxation techniques for the three problems, modularity maximization problem, modularity density maximization problem, and ranking problem.

Firstly, we consider the problem of maximizing the modularity in the context of the community detection. This problem, when formulated as the well-known set partitioning problem, has to take into account all nonempty subsets of the node set of the underlying network, resulting in a huge number of variables and difficult set partitioning constraints. To tackle this difficulty we propose two algorithms based on LP relaxation and Lagrangian relaxation. Secondly, we consider the modularity density maximization which is originally formulated as a nonlinear fractional programming. We show that this problem is equivalent to a variant of the semidefinite programming problem called 0-1SDP. We propose to relax 0-1SDP to a semidefinite programming problem with a non-negativity constraint of variables. The relaxed problem has a big advantage that its size depends on neither the number of communities nor the number of edges of the network. Thirdly, we investigate the solution method for the ranking problem. The problem is a problem of maximizing the minimum margin, which is solved in polynomial time theoretically. However, real-world instances of this problem become practically difficult to solve due to their size. We propose a row and column generation algorithm, show that it solves the problem to optimality, and also demonstrate that it mitigates the computational burden by some computational experiment.

# Acknowledgment

This thesis would not have been completed without the help of my supervisor Professor Yoshitsugu Yamamoto. I would like to extend my deep appreciation to him for his enthusiastic guidance, a number of helpful comments, and persistent encouragement. I am also grateful to Professors Akiko Yoshise, Maiko Shigeno, Masahiro Hachimori, and Yusuke Kobayashi for valuable advice. I wish to thank Professor Ayako Shibuya of Yamaguchi University. A great deal of gratitude goes to Professor Tomomi Matsui of Tokyo Institute of Technology, Professor Keiji Tatsumi of Osaka University, and Keisuke Sato of Railway Technical Research Institute, with whom I did joint works. I wish to extend special thanks to my colleagues Kotohumi Inaba, Akihiro Tanaka, and other members of the team *Phil Opt ))))*. Finally, a special word of appreciation goes to my parents.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In recent years, progress in information technology including facility for the data acquisition, cost reduction for the data storage and high-speed data communication, has built up an enormous amount of data. However, thus accumulated data is not organized well, hence much interest has arisen in how to mine the data to elicit useful information. From this background, *knowledge discovery in databases*, KDD for short, and *data mining* have been attracting a great deal of attention from both fields of research and practice. According to Bradley *et al.* [17], KDD and data mining are defined as follows: "*KDD is the process of identifying valid, novel, potentially useful, and ultimately understandable structure in data. This process involves selecting or sampling data from a data warehouse, cleaning or preprocessing it, transforming or reducing it (if needed), applying a data mining component to produce structure, and then evaluating the derived structure, and data mining is a step in the KDD process concerned with the algorithmic means by which patterns or models are enumerated from the data under acceptable computational efficiency limitations.* "

We will consider three problems arising from data mining: modularity maximization problem, modularity density maximization problem and ranking problem. After giving a brief review of typical applications in data mining, we give an outline of these problems together with their backgrounds, and then the outcome of our research.

## 1.1 Typical Applications in Data Mining

The data set dealt with in data mining falls into two types, *labeled* and *unlabeled*. An individual piece of data in the labeled data set is a pair of an attribute and a label. Data mining for this type of data set is called *supervised learning*, and its aim is to infer a hidden functional relation between the set of attributes and the set of labels, and to apply the inferred relation to predict a label of a newly-arrived data. On the other hand, unlabeled data set is composed of data without any significant label, and data mining for this type of data set is called *unsupervised learning*.

Applications in data mining can be divided into four main categories: association rule, clustering, classification, and regression analysis [18].

- Association rule is a task to find frequent patterns (rules) or interesting correlations among objects in the transaction.
- Clustering is a task to divide objects into several groups (clusters) consisting of objects which share a common, possibly yet unknown, characteristics.
- Classification is a task to predict a categorical label (class) of a newly-arrived object based on a relation obtained from a given data set.
- Regression analysis is a task to estimate a relationship between an attribute (independent variable) and a numerical label (dependent variable).

The first two are unsupervised learning, and the remaining two are supervised learning.

## 1.2 Overview of the Thesis

This thesis is concerned with three problems, modularity maximization problem, modularity density maximization problem, and ranking problem. The first two problems are regarded as a clustering problem where the similarity among objects is given by a network structure with those objects as its nodes. Ranking problem is regarded as a classification problem where the label assigned to each data is a categorical value, whereas it can be regarded as a regression problem when the label is a real value.

### 1.2.1 Background of community detection

Network clustering has received growing attention as one of major problems in data mining. One of the most important issues in the network analysis is to find a meaningful structure, which often addresses identifying or detecting community structure. Here communities are the sets of nodes such that each set consists of tightly connected nodes, but loosely connected each other. Community detection is applied to analyze the underlying relationship of a wide variety of networks such as the social network, the biological network, the world-wide-web, and VLSI network.

**Modularity maximization** A variety of approaches to detect communities has been proposed. The *cut size* defined as the number of edges connecting communities is one of the most commonly used quality measure. Minimizing the cut size, known as the minimum cut method, however often results in forming a number of very small communities. Several variants to overcome this drawback have been introduced such as the ratio cut [91], the normalized cut [85], and min-max cut [33].

Some authors presented *centrality* concepts to characterize the important and influential nodes in a network, such as degree centrality, closeness centrality, and betweenness centrality. Girvan and Newman [46] extended the definition of the betweenness centrality from a node to an edge of a network, which represents the ratio between the total number of shortest paths between all pairs of nodes and the number of those that pass through the particular edge. They also proposed a heuristic algorithm based on an edge version of the betweenness centrality. Their algorithm proceeds by removing an edge with the highest betweenness, and then ends up with several connected components each of which is to make a community. Although this algorithm performs well on both real-world and benchmark networks, it places a heavy computational burden due to the necessary calculation of the betweenness centrality in every step.

A novel quality measure, called modularity, has been proposed by Newman and Girvan [78]. The modularity was originally used as a stopping criterion of the hierarchical divisive algorithm [78]. Then Newman [76] suggested an approach of maximizing the modularity due to the observation that a high value of the modularity leads to a good community structure. Then the modularity maximization became one of the central subjects of research.

**Modularity density maximization** The modularity maximization receives criticism

from mainly two viewpoints: *degeneracy* and *resolution limit*. Degeneracy, pointed out in Good *et al.* [47], means the presence of several partitions with high modularity which makes it difficult to find a global optimal partition. Resolution limit, brought up in Fortunato and Barthélemy [42], refers to the sensitivity of modularity to the total number of edges in the network, which leaves small communities not identified and hidden inside larger ones. Even in the schematic case where a network consists of multiple replicas of an identical clique which are connected by a single edge, Fortunato and Barthélemy [42] showed that maximizing the modularity resulted in communities consisting of two or more cliques connected when the number of cliques in the network is larger than the square root of the number of edges. This narrows the application area of the modularity maximization since most of real-world networks contain communities with different scales.

To overcome the resolution limit, there have been extensive studies so far [5,48,74,88]. Arenas *et al.* [5] clarified the topological interpretation of the resolution limit, and then proposed a multiple resolution method which performs community detection at different scales as varying a parameter to specify the scale. However, as mentioned above, real-world networks are characterized by the coexistence of communities with very different scales. Hence, once a resolution parameter is determined, the multiple resolution method tends to merge communities whose scales are smaller than the predetermined parameter and also to split communities whose scales are larger than that [60].

Recently, Li *et al.* [63] have proposed a new measure for community detection, which is called modularity density. Based on a theoretical analysis they show that the modularity density (i) does not divide a clique into separate communities, (ii) can resolve most modular networks correctly, which means maximizing the modularity density results in the partition with each single clique as a community in the schematic case mentioned above, and (iii) can detect communities with different scales. Thanks to these desirable properties, the modularity density maximization increasingly comes into a meaningful problem to comprehend a community structure.

The modularity and the modularity density are defined for a partition of the set of nodes on undirected and unweighted networks. Some authors extended the definitions of the modularity and the modularity density to the directed and/or weighted networks in order to grasp more valuable meaning. In addition, it would be natural to consider

that a node can belong to a number of communities in the real-world networks, thus communities may overlap with each other. See, for instance [25,61,75] for further details. This thesis does not address the above extension and is devoted to only detecting disjoint communities in the undirected and unweighted networks.

**Comparison between Modularity and Modularity density** First, we mention the mathematical programming formulations for the above two problems. The modularity maximization is straightforwardly formulated as an integer linear programming, such as a set partitioning formulation and a clique partitioning formulation. On the other hand, the modularity density maximization is formulated as a nonlinear fractional programming since the modularity density takes account of the number of nodes in each community. This difference in formulation indicates that the modularity density maximization seems to be more intractable than the modularity maximization.

Concerning the computational complexity of the above two problems, the NP-hardness of the modularity maximization problem has been shown by Brandes *et al.* [19], while the NP-hardness of the modularity density maximization still remains open [28], but some computational results reported in [29, 30] imply that maximizing the modularity density is also hard.

### 1.2.2 Background of ranking problem

Development of the world-wide-web has enabled an instantaneous access to enormous information, meanwhile obtaining desirable information is becoming increasingly difficult. Consequently, construction of a search engine that provides effective information retrieval receives growing attention. Given a query, a search engine is expected to retrieve documents together with the ranking showing how relevant the documents are. Thus, reliable and efficient ranking is an important task in many information retrieval applications including collaborative-filtering, text-summarization, and online-advertising.

Given a data set which consists of pairs of an attribute vector and a label chosen from the ordered set of labels, the aim of ranking we consider in this thesis is to infer an underlying model which assigns the label. This problem is regarded as a multi-class classification, which has, besides information retrieval, many other applications, e.g., credit rating and computational biology. Several methods have been proposed, such as PRank [31], Subset Ranking [27], and McRank [62].

Shashua and Levin [83] have proposed a method based on the fixed margin strategy, which can be regarded as a variant of multi-class support vector machine. What distinguishes their approach from other multi-class support vector machines is that the identical normal vector should be shared by all the separating hyperplanes. Their formulation ends up with a convex quadratic programming problem, hence the nonlinear classification method called kernel technique can be naturally applied. Although the convex quadratic programming is solved in polynomial time theoretically, one can hardly secure the computational resource to even hold the problem when the size of the given data set is considerably large.

It would sometimes be desirable that the obtained separating curves have some monotonicity property. A few researchers discussed the monotonicity issue, for example [24, 36]. The main idea of their approaches is to produce a large amount of virtual data which satisfies the desirable monotonic property, and then add the virtual data to the problem. However, this makes the resulting problem difficult, in addition, while it does not guarantee the monotonic separating curves.

### 1.2.3 Contributions of the Thesis

The contribution of this thesis is twofold: community detection in Chapters 2 and 3, and ranking in Chapter 4.

Chapters 2 and 3 are devoted to the development of relaxation algorithms for the modularity maximization and the modularity density maximization problems. Owing to the computational difficulty of these problems, most of researchers have proposed heuristic algorithms so far, while exact algorithms have been proposed only in a few papers [2, 3, 29, 93].

For the modularity maximization, we propose two algorithms each based on LP relaxation and Lagrangian relaxation. Both algorithms have an advantage that they are able to provide the upper bounds on the optimal modularity by solving a small sub-problem. They also enjoy the advantage that multiple cuts (resp., columns) within a cutting plane (resp., a column generation) process significantly reduces the number of iterations. We report some computational results to evaluate the performance of our algorithms.

In Chapter 3 we show that a variant of the semidefinite optimization problem, called 0-1SDP, is equivalent to the modularity density maximization problem. This formulation

has the big advantage that its size is independent of the number of edges of the network. For an upper bound on the optimal modularity density, we propose to solve a doubly non-negative relaxation problem, and for a lower bound we propose an algorithm based on the combination of spectral heuristics and dynamic programming. Some computational results are also reported.

Chapter 4 is devoted to the development of an efficient exact algorithm for the ranking problem. We propose a formulation based on but different from the fixed margin strategy by Shashua and Levin, then develop a row and column generation algorithm. The algorithm starts with a sub-problem which is much smaller than the original problem in both the number of variables and constraints, and increments both of them as the computation goes on. We show that the algorithm solves the problem to optimality. Assuming some conditions, we present a model which necessarily yields the monotonic separating curves.

## 1.3 Organization of the Thesis

This thesis is organized as follows.

In Chapter 2, we first give the definition and some properties of the modularity, and introduce several formulations of the modularity maximization problem. Next, we propose LP relaxation-based and Lagrangian relaxation-based algorithms, and then describe some techniques to accelerate the algorithms, also discuss the convergence issue of our algorithms. Finally, we report the computational experiments of the algorithms.

In Chapter 3, giving the definition of the modularity density and a nonlinear fractional programming formulation of the modularity density maximization problem, we review some properties of the modularity density. We present 0-1SDP formulation for the modularity density maximization and show the equivalence between the two problems. Moreover, we propose to solve a doubly non-negative relaxation problem of 0-1SDP, and explain a heuristic algorithm which constructs a feasible solution by using the solution of the relaxation problem. Finally, we conduct the numerical experiments to evaluate the lower and upper bounds obtained by our algorithm.

In Chapter 4, we first formulate the ranking problem as the problem of maximizing minimum margin, and give its dual problem to apply the kernel technique. Next, we

propose a primal formulation with the dual representation of the normal vector in the objective function, and develop an algorithm based on this formulation. The algorithm takes advantage of the row and column generation technique in order to mitigate the computational burden, while keeping its finite convergence property to optimality. After giving a small illustrative example, we report the computational experiments of our algorithm. Finally, we discuss the monotonicity property of the separating curves.

In Chapter 5, we give some concluding remarks, and mention some future works.

# Chapter 2

# Modularity Maximization

## 2.1 Introduction

As social network services grow, community detection has been attracting a great deal of attention. Since Newman and Girvan [46] proposed the modularity as a quality measure of community detection, the modularity maximization problem became one of the central subjects of research. Most of the solution methods proposed so far are heuristic algorithms due to its NP-hardness, which was shown by Brandes *et al.* [19], while exact algorithms have been proposed only in a few papers.

Aloise *et al.* [3] formulated the problem as a set partitioning problem, which has to take into account all, exponentially many, nonempty subsets of the node set. Therefore, one cannot secure the computational resource to hold the problem when the number of nodes is large. They proposed an algorithm based on the column generation, and applied the stabilized column generation to accelerate the algorithm. Although it is known that the stabilized column generation works well for the problem with set partitioning constraints, several parameters should be controlled dynamically and correctly at every iteration.

In this chapter, for the set partitioning formulation of the modularity maximization, we propose two algorithms based on LP relaxation and Lagrangian relaxation. Both algorithms have an advantage that they are able to derive the upper bounds on the optimal modularity from a small sub-problem. In order to accelerate our algorithms, we incorporate several ideas into the algorithms including modification of stopping criteria, method of multiple cuts or multiple columns, and pegging test.

### 2.1.1 Definitions and notation

Let $G = (V, E)$ be an undirected graph with the set $V$ of $n$ nodes and the set $E$ of $m$ edges. We say that $\Pi = \{C_1, C_2, \ldots, C_k\}$ is a *partition* of $V$ if $V = \bigcup_{p=1}^{k} C_p$, $C_p \cap C_q = \emptyset$ for any distinct $p$ and $q$, and $C_p \neq \emptyset$ for any $p$. Each member $C_p$ of a partition is called a *community*. We denote the set of edges that have both end-nodes in $C$ by $E(C)$. Then *modularity*, denoted by $Q(\Pi)$, of a partition $\Pi$ is defined as

$$Q(\Pi) = \sum_{C \in \Pi} \left( \frac{|E(C)|}{m} - \left( \frac{\sum_{i \in C} d_i}{2m} \right)^2 \right),$$

where $|\cdot|$ denotes the cardinality of the corresponding set and $d_i$ is the degree of node $i$. For $i, j \in V$ let $e_{ij}$ be the $(i, j)$ element of the adjacency matrix of graph $G$, and $\pi(i)$ be the index of community which node $i$ belongs to, i.e., $\pi(i) = p$ means $i \in C_p$. Then $Q(\Pi)$ is rewritten as follows:

$$Q(\Pi) = \frac{1}{2m} \sum_{i \in V} \sum_{j \in V} \left( e_{ij} - \frac{d_i d_j}{2m} \right) \delta(\pi(i), \pi(j)),$$

where $\delta$ is the Kronecker delta, i.e., $\delta(p, q)$ is equal to one when $p = q$ and zero otherwise. *Modularity maximization problem*, MM for short, is the problem of finding a partition of $V$ that maximizes the modularity $Q(\Pi)$. Denoting $(e_{ij} - d_i d_j / 2m)$ by $w_{ij}$, then the problem is formulated as

$$\text{(MM)} \quad \left| \begin{array}{ll} \text{maximize} & \dfrac{1}{2m} \displaystyle\sum_{i \in V} \sum_{j \in V} w_{ij}\, \delta(\pi(i), \pi(j)) \\ \text{subject to} & \Pi \text{ is a partition of } V. \end{array} \right.$$

### 2.1.2 Some properties and complexity

In this subsection, we present some properties of modularity, which will be useful in later discussion.

**Lemma 2.1** (Brandes *et al.* [19], Lemma 1.) *For any partition $\Pi$ of $V$, modularity $Q(\Pi)$ falls between -1/2 and 1.*

The above lemma gives trivial lower and upper bounds on modularity.

**Lemma 2.2** (Brandes *et al.* [19], Corollary 1.) *Isolated nodes have no impact on modularity.*

Thanks to lemma 2.2, we exclude isolated nodes from further consideration in this thesis, hence we can assume that the degree is greater than zero for any node.

**Lemma 2.3** (Brandes *et al.* [19], Lemma 2.) *Let $\Pi^\star$ be a partition with maximum modularity, then $\Pi^\star$ has no community that consists of a single node with degree one.*

**Lemma 2.4** (Brandes *et al.* [19], Lemma 3.) *There exists a partition with maximum modularity, in which each community consists of a connected subgraph.*

From Lemma 2.3 and 2.4, we can conclude that a node with degree one and its neighbor must belong to the same community in the optimal partition.

**Proposition 2.5** *Let $i \in V$ be a node with degree one and $j \in V$ be adjacent to node $i$, then nodes $i$ and $j$ are assigned to the same community.*

Brandes *et al.* [19] also provided the computational complexity for the modularity maximization problem. They proved that the modularity maximization problem is NP-hard. More precisely, they considered the following problem which is the decision version of the modularity maximization.

**Problem 2.6 (MODULARITY)** *Given a graph $G = (V, E)$ and a real number $K \in [-1/2, 1]$, is there a partition $\Pi$ of $V$ such that $Q(\Pi) \geq K$?*

Their complexity result is based on a reduction from the following NP-complete decision problem.

**Problem 2.7 (3-PARTITION)** *Given $3k$ positive integer numbers $A = \{a_1, \ldots, a_{3k}\}$ such that $\sum_{i=1}^{3k} a_i = kb$ and $b/4 < a_i < b/2$ for an integer $b$ and for $i = 1, \ldots, 3k$, is there a partition of $A$ into $k$ sets such that the sum of the numbers in each set is equal to $b$?*

**Theorem 2.8** (Brandes *et al.* [19], Theorem 3.) *MODULARITY is strongly NP-complete.*

Therefore there exists no polynomial time algorithm that finds a partition with maximum modularity unless P=NP. This turned researchers' attention to heuristic algorithms, which resulted in several efficient heuristic algorithms such as the hierarchical agglomerative method by Clauset *et al.* [26], the spectral divisive method by Newman [77], the divisive

method by Cafieri *et al.* [20], the linear programming with rounding procedure [2] and the simulated annealing by Guimerá and Amaral [51].

On the other hand, among exact algorithms three approaches should be mentioned. The first one of them is based on the *quadratic programming* formulation, QP for short. This formulation suffers from symmetry, that is, there exists a large number of equivalent solutions. As a consequence, computation time increases due to expanding the search space of solutions. Xu *et al.* [93] solved instances up to 104 nodes by introducing some symmetric breaking constraints. The second one is based on the formulation of the problem as a clique partitioning problem proposed by Grötschel and Wakabayashi [49]. In this formulation, a binary variable corresponding to each pair of nodes represents whether the two nodes belong to the same community. Then the number of variables and constraints amount to $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$, respectively, both of which grow rapidly with the number of nodes. Based on this formulation Aloise *et al.* [3] solved instances up to 115 nodes by using the cutting plane algorithm. The third one is the set partitioning formulation. Since this formulation has to take into account all nonempty subsets of the node set, it has $\mathcal{O}(2^n)$ variables. One can hardly secure the computational resource to hold the problem when $n$ is large. Aloise *et al.* [3] proposed an algorithm based on the column generation, and solved instances up to 512 nodes exactly.

For the large network whose optimal modularity is unknown, it is also important to obtain an upper bound of the optimal modularity in terms of the accuracy evaluation of the solution provided by heuristic algorithms. Miyauchi and Miyamoto [72] proposed an algorithm computing a nontrivial upper bound, and succeeded in obtaining upper bounds of instances up to 4941 nodes.

## 2.2 Formulations

In this section we introduce three different formulations of the modularity maximization problem. The first one is based on the QP formulation, the second one is based on the clique partitioning formulation, and the third one is based on the set partitioning formulation.

### 2.2.1 QP formulation

Usually, we do not know the optimal number of communities for the modularity maximization problem a priori. Now we consider the problem of partitioning $n$ nodes into $t$ or less communities for a positive integer $t \geq 2$. The QP formulation for this problem has been proposed by Xu *et al.* [93]. Let $T$ be the index set $\{1, 2, \ldots, t\}$ and for each edge $l = \{i, j\} \in E$ and each $p \in T$, let $x_{lp}$ be the binary variable such that

$$x_{lp} = \begin{cases} 1 & \text{when } l \in E(C_p) \\ 0 & \text{otherwise.} \end{cases}$$

For each node $i \in V$ and each $p \in T$, let $y_{ip}$ be defined by

$$y_{ip} = \begin{cases} 1 & \text{when } i \in C_p \\ 0 & \text{otherwise.} \end{cases}$$

Then (MM) is formulated as the following quadratic programming:

$$(QP) \quad \left| \begin{array}{lll} \text{maximize} & \displaystyle\sum_{p \in T} \left( \frac{1}{m} \sum_{l \in E} x_{lp} - \frac{1}{4m^2} \left( \sum_{i \in V} d_i y_{ip} \right)^2 \right) & \\ \text{subject to} & \displaystyle\sum_{p \in T} y_{ip} = 1 & (i \in V) \\ & x_{lp} \leq y_{ip} & (l = \{i, j\} \in E, p \in T) \\ & x_{lp} \leq y_{jp} & (l = \{i, j\} \in E, p \in T) \\ & x_{lp} \in \{0, 1\} & (l \in E, p \in T) \\ & y_{ip} \in \{0, 1\} & (i \in V, p \in T). \end{array} \right.$$

The first set of constraints imposes that each node belongs to exactly one community, and the remaining constraints express that any edge $l = \{i, j\}$ belongs to a set of edges $E(C_p)$ if both end-nodes $i, j$ belong to the community $C_p$. This formulation suffers from *symmetry*, that is, re-indexing some communities yields alternative equivalent solutions. For an optimal solution with $t$ communities, $t\,!$ equivalent solutions exist. Such solutions are called symmetric solutions. As a consequence, branch-and-bound algorithm tends not to work well. Xu *et al.* [93] have proposed some valid inequalities to get rid of the symmetric solutions from the feasible region. In this paper, we will not solve the problem $(QP)$, but make use of this problem to obtain an upper bound on the optimal value of $(P)$. See Subsection 2.3.3 for the detail.

### 2.2.2 Clique partitioning formulation

For each pair of nodes $i$ and $j$, we introduce a binary variable $x_{ij}$, which represents whether the two nodes belong to the same community, i.e., the variable $x_{ij}$ equals 1 when the nodes $i$ and $j$ belong to the same community and 0 otherwise. Then the modularity is expressed as follows:

$$\frac{1}{2m}\sum_{i\in V}\sum_{j\in V} w_{ij}x_{ij},$$

which is rewritten as

$$\frac{1}{m}\sum_{i\in V}\sum_{j\in V\,;\,i<j} w_{ij}x_{ij} + \frac{1}{2m}\sum_{i\in V} w_{ii}$$

due to the symmetry of $x_{ij}$ and $x_{ii} = 1$ for any $i \in V$. Thus (MM) is formulated as the following integer programming:

$$\text{(CPP)} \quad \left| \begin{array}{lll} \text{maximize} & \dfrac{1}{m}\displaystyle\sum_{(i,j)\in V^2_<} w_{ij}x_{ij} + \dfrac{1}{2m}\sum_{i\in V} w_{ii} & \\ \text{subject to} & x_{ij} + x_{jk} - x_{ik} \le 1 & ((i,j,k) \in V^3_<) \\ & x_{ij} - x_{jk} + x_{ik} \le 1 & ((i,j,k) \in V^3_<) \\ & -x_{ij} + x_{jk} + x_{ik} \le 1 & ((i,j,k) \in V^3_<) \\ & x_{ij} \in \{0,1\} & ((i,j) \in V^2_<), \end{array} \right.$$

where

$$V^2_< = \{\, (i,j) \in V \times V \mid i,j \in V; i < j \,\},$$
$$V^3_< = \{\, (i,j,k) \in V \times V \times V \mid i,j,k \in V; i < j < k \,\}.$$

The first three sets of constraints guarantee that if $i$ and $j$ belong to the same community and $j$ and $k$ belong to the same community, then $i$ and $k$ also belong to the same community. These sets of constraints are called *transitivity constraints*. See Grötschel and Wakabayashi [50] for the facial structure of the clique partitioning polytope.

The difficulty of the problem (CPP) is the huge number of constraints, which amounts to $n(n-1)(n-2)/2$. Dinh and Thai [34] proposed the sparse formulations of both (CPP) and its LP relaxation, that is, they clarified redundant constraints at an optimal solution of the problems. More precisely they showed that the following constraints are redundant.

$$x_{ij} + x_{jk} - x_{ik} \le 1 \quad \text{for } (i,j,k) \in V^3_<;\ \{i,j\} \notin E \text{ and } \{j,k\} \notin E,$$
$$x_{ij} - x_{jk} + x_{ik} \le 1 \quad \text{for } (i,j,k) \in V^3_<;\ \{i,j\} \notin E \text{ and } \{i,k\} \notin E,$$
$$-x_{ij} + x_{jk} + x_{ik} \le 1 \quad \text{for } (i,j,k) \in V^3_<;\ \{j,k\} \notin E \text{ and } \{i,k\} \notin E.$$

Recently, the above result was extended by Miyauchi and Sukegawa [73], who presented that the following constraints are also redundant in (CPP) and the LP relaxation of (CPP).

$$\begin{aligned} x_{ij} + x_{jk} - x_{ik} \leq 1 \quad & \text{for } (i,j,k) \in V^3_<;\ w_{ij} < 0 \text{ and } w_{jk} < 0, \\ x_{ij} - x_{jk} + x_{ik} \leq 1 \quad & \text{for } (i,j,k) \in V^3_<;\ w_{ij} < 0 \text{ and } w_{ik} < 0, \\ -x_{ij} + x_{jk} + x_{ik} \leq 1 \quad & \text{for } (i,j,k) \in V^3_<;\ w_{jk} < 0 \text{ and } w_{ik} < 0. \end{aligned}$$

Miyauchi and Miyamoto [72] developed a cutting plane algorithm to solve the LP relaxation problem of the sparse formulation by Dinh and Thai [34], and succeeded in obtaining upper bounds on modularity for large instances.

### 2.2.3 Set partitioning formulation

Let $\mathcal{P}$ denote the family of all nonempty subsets of $V$. Note that $\mathcal{P}$ is composed of $2^n - 1$ subsets of $V$. Introducing a binary variable $z_C$ for each $C \in \mathcal{P}$, a partition $\Pi$ is represented by the $(2^n - 1)$-dimensional binary vector $\boldsymbol{z} = (z_C)_{C \in \mathcal{P}}$ defined as

$$z_C = \begin{cases} 1 & \text{when } C \in \Pi \\ 0 & \text{otherwise.} \end{cases}$$

This enables us to formulate problem (MM) as an integer programming problem. For each $i \in V$ and $C \in \mathcal{P}$ let $a_{iC}$ be defined by

$$a_{iC} = \begin{cases} 1 & \text{when } i \in C \\ 0 & \text{otherwise.} \end{cases}$$

The column $\boldsymbol{a}_C = (a_{1C}, \ldots, a_{nC})^\top$ is the incidence vector of community $C$, i.e., $C = \{\, i \in V \mid a_{iC} = 1 \,\}$. For each $C \in \mathcal{P}$ let $f_C$ be

$$f_C = \frac{1}{2m} \sum_{i \in C} \sum_{j \in C} w_{ij}, \tag{2.1}$$

which is rewritten as

$$= \frac{1}{2m} \sum_{i \in V} \sum_{j \in V} w_{ij} a_{iC} a_{jC}.$$

The constant $f_C$ represents the contribution of community $C$ to the objective function $Q(\Pi)$ when community $C$ is selected as a member of the partition $\Pi$. Thus (MM) is formulated as the integer programming $(P)$:

$$(P) \quad \left| \begin{array}{ll} \text{maximize} & \displaystyle\sum_{C \in \mathcal{P}} f_C z_C \\ \text{subject to} & \displaystyle\sum_{C \in \mathcal{P}} a_{iC} z_C = 1 \quad (i \in V) \\ & z_C \in \{0,1\} \qquad (C \in \mathcal{P}). \end{array} \right.$$

Since the first set of constraints states that the communities adopted form a partition of $V$, this problem is called a *set partitioning problem*. From now on, we will call the first set of constraints set partitioning constraints.

We say that $\Pi = \{C_1, C_2, \ldots, C_k\}$ is a *cover* of $V$ if $V = \bigcup_{p=1}^{k} C_p$ and $C_p \neq \emptyset$ for any $p$. Relaxing the set partitioning constraints in $(P)$ to $\sum_{C \in \mathcal{P}} a_{iC} z_C \geq 1$ for any $i \in V$, the obtained problem is called a *set covering problem* which is to find a cover of $V$. Due to its huge number of variables, these problems easily become computationally intractable as the number of nodes grows. Moreover even if the set partitioning problem as well as the set covering problem has small number of variables, the problems still remain difficult owing to their NP-hardness.

As for these problems given the set of columns explicitly, exact algorithms have been proposed [6, 8, 9, 12]. Balas and Carrera [6] exactly solved the large instances with up to 400 constraints and 4000 variables by branch-and-bound algorithm based on a subgradient optimization. Heuristic and metaheuristic algorithms have been also studied in the literature [10, 11, 21, 23, 52]. See survey papers [22, 89].

## 2.3 LP Relaxation Algorithms

Not only the number of variables but also their integrality makes problem $(P)$ a highly intractable problem. Then it would be a natural and clever strategy to consider relaxation problems for the useful information about the solution of $(P)$. The first choice to consider would be the LP relaxation.

For the modularity maximization, a column generation algorithm based on the LP relaxation was proposed by Aloise *et al.* [3]. Column generation is a common trick to deal with the problems with a huge number of variables, but the auxiliary problem of

determining the entering column ends up as a quadratic programming in binary variables. They applied the stabilized column generation by Du Merle *et al.* [37] to accelerate the algorithm, and used variable neighborhood search heuristics [43] to solve the auxiliary problem. Although the stabilized column generation is a sophisticated technique, the size of the problem to consider is larger than that of original one due to adding the slack and surplus variables to reduce *degeneracy* of the LP problem. In addition, penalty parameters for the slack and surplus variables should be updated dynamically at each iteration in the algorithm. In order to use the stabilized column generation efficiently, one must initialize and update the parameters correctly. However it is difficult to design a specific parameter tuning.

In this section, we propose cutting plane algorithms, which provide the nontrivial upper and lower bounds on the optimal modularity at each iteration. This could enable us to decrease the number of iterations of the algorithms. Our algorithms have the following features: (i) our algorithms do not require additional computational burden in estimating an upper bound on modularity at each iteration, and (ii) have few parameters to control compared with the stabilized column generation, hence it is simple and easy to implement the algorithms.

### 2.3.1 LP relaxation problem and its dual problem

The LP relaxation problem is defined by replacing the binary constraints $z_C \in \{0, 1\}$ by $0 \leq z_C \leq 1$. It is given as

$$(RP) \quad \left| \begin{array}{lll} \text{maximize} & \displaystyle\sum_{C \in \mathcal{P}} f_C z_C & \\ \text{subject to} & \displaystyle\sum_{C \in \mathcal{P}} a_{iC} z_C = 1 & (i \in V) \\ & z_C \geq 0 & (C \in \mathcal{P}). \end{array} \right.$$

The upper bound constraints $z_C \leq 1$ are redundant owing to the first set of constraints, hence omitted. The linear programming dual problem of $(RP)$ is given as the following $(RD)$:

$$(RD) \quad \left| \begin{array}{lll} \text{minimize} & \displaystyle\sum_{i \in V} \lambda_i & \\ \text{subject to} & \displaystyle\sum_{i \in V} a_{iC} \lambda_i \geq f_C & (C \in \mathcal{P}) \\ & \lambda_i \in \mathbb{R} & (i \in V). \end{array} \right.$$

Now let us denote the feasible region and the optimal value of an optimization problem, say $H$, by $\mathcal{F}(H)$ and $\omega(H)$, respectively. Since $(RP)$ is a relaxation problem of $(P)$, it holds that $\mathcal{F}(P) \subseteq \mathcal{F}(RP)$, hence $\omega(P) \leq \omega(RP)$. Applying the linear programming duality theorem to the primal dual pair $(RP)$ and $(RD)$, we see $\omega(P) \leq \omega(RD)$. Namely, solving either $(RP)$ or $(RD)$ we will obtain an upper bound of $\omega(P)$. An optimal solution of $(RP)$ often provides a clue as to possibly a good feasible solution of $(P)$. However the problems $(RP)$ and $(RD)$ still remain difficult owing to exponentially many variables and constraints.

### 2.3.2 Cutting plane method

The constraints of problem $(RD)$ far outnumber the variables, hence most of them should not be binding at an optimal solution. The cutting plane method is one of commonly used methods for LP problem of this kind.

Now we will give a brief review of the cutting plane method which is based on the classical Kelley's algorithm [56]. The key idea of the cutting plane method is to deal with a small subfamily $\mathcal{S}$ of $\mathcal{P}$, and instead of $(RD)$, to solve the following problem with fewer constraints:

$$(RD(\mathcal{S})) \quad \left| \begin{array}{lll} \text{minimize} & \displaystyle\sum_{i \in V} \lambda_i & \\ \text{subject to} & \displaystyle\sum_{i \in V} a_{iC} \lambda_i \geq f_C & (C \in \mathcal{S}) \\ & \lambda_i \in \mathbb{R} & (i \in V). \end{array} \right.$$

Let $\boldsymbol{\lambda}$ denote an optimal solution of $(RD(\mathcal{S}))$. Since the constraints $\sum_{i \in V} a_{iC} \lambda_i \geq f_C$ for $C \in \mathcal{P} \setminus \mathcal{S}$ are not considered, it is not necessarily a feasible solution of $(RD)$. To check the feasibility of $\boldsymbol{\lambda}$, we define a measure of violation $\gamma_C(\boldsymbol{\lambda})$ of the constraint corresponding to $C$ as

$$\gamma_C(\boldsymbol{\lambda}) = f_C - \sum_{i \in V} a_{iC} \lambda_i. \tag{2.2}$$

Note that $\gamma_C(\boldsymbol{\lambda}) \leq 0$ for all $C \in \mathcal{S}$. When $\gamma_C(\boldsymbol{\lambda}) \leq 0$ for all $C \in \mathcal{P} \setminus \mathcal{S}$, $\boldsymbol{\lambda}$ is a feasible solution of the problem $(RD)$, hence an optimal solution of the problem $(RD)$. When

$$\gamma_C(\boldsymbol{\lambda}) > 0$$

holds for some $C \in \mathcal{P} \setminus \mathcal{S}$, adding this $C$ to $\mathcal{S}$ can lead to an improvement of the optimal value of the problem $(RD(\mathcal{S}))$, i.e., $\omega(RD(\mathcal{S} \cup \{C\})) \geq \omega(RD(\mathcal{S}))$. Substituting (2.1)

for $f_C$ of (2.2) yields

$$\gamma_C(\boldsymbol{\lambda}) = \frac{1}{2m}\sum_{i\in V}\sum_{j\in V} w_{ij}a_{iC}a_{jC} - \sum_{i\in V} a_{iC}\lambda_i,$$

hence the problem of maximizing $\gamma_C(\boldsymbol{\lambda})$ over $\mathcal{P}$ is formulated as the problem $(\overline{AP}(\boldsymbol{\lambda}))$ with a quadratic objective function in binary variables:

$$(\overline{AP}(\boldsymbol{\lambda})) \quad \left| \begin{array}{ll} \text{maximize} & \dfrac{1}{2m}\displaystyle\sum_{i\in V}\sum_{j\in V} w_{ij}y_iy_j - \sum_{i\in V}\lambda_i y_i \\ \text{subject to} & y_i \in \{0,1\} \qquad (i \in V). \end{array} \right.$$

An optimal solution $\boldsymbol{y}^*$ of this problem provides the incidence vector of the community that maximizes $\gamma_C(\boldsymbol{\lambda})$ over $\mathcal{P}$. Since $\boldsymbol{y} = \mathbf{0}$ is a feasible solution of this problem, the optimal value is non-negative. According to the quadratic formulation presented in Subsection 2.2.1, $(\overline{AP}(\boldsymbol{\lambda}))$ is equivalently formulated as the following problem with a quadratic concave function:

$$(AP(\boldsymbol{\lambda})): \quad \left| \begin{array}{ll} \text{maximize} & \dfrac{1}{m}\displaystyle\sum_{l\in E} x_l - \frac{1}{4m^2}\left(\sum_{i\in V} d_i y_i\right)^2 - \sum_{i\in V}\lambda_i y_i \\ \text{subject to} & x_l \le y_i \qquad (l = \{i,j\} \in E) \\ & x_l \le y_j \qquad (l = \{i,j\} \in E) \\ & x_l \in \{0,1\} \qquad (l \in E) \\ & y_i \in \{0,1\} \qquad (i \in V). \end{array} \right.$$

For each edge $l = \{i,j\} \in E$, a binary variable $x_l$ is equal to 1 when both end-nodes $i, j$ of edge $l$ belong to a community that maximizes $\gamma_C(\boldsymbol{\lambda})$, and for each $i \in V$ a variables $y_i$ is equal to 1 when node $i$ belongs to the community and 0 otherwise. Having found $\boldsymbol{y}^*$ with positive optimal value, we have only to add the constraint

$$\sum_{i\in V} y_i^*\lambda_i \ge f^*$$

to $(RD(\mathcal{S}))$, where

$$f^* = \frac{1}{2m}\sum_{i\in V}\sum_{j\in V} w_{ij}y_i^*y_j^*.$$

From the above discussion, a prototype of the cutting plane algorithm is given as follows.

**Algorithm Prototype of the Cutting Plane**

Step 1 : Let $\mathcal{S}$ be an initial family of nonempty subsets of $V$.

Step 2 : Solve $(RD(\mathcal{S}))$ to obtain an optimal solution $\boldsymbol{\lambda}$ and the optimal value $\omega(RD(\mathcal{S}))$.

Step 3 : Solve $(AP(\mathcal{S}))$ and set $\boldsymbol{y}^*$ be an optimal solution.

Step 4 : If $\omega(AP(\boldsymbol{\lambda})) \leq 0$, then set $\mathcal{S}^\star \leftarrow \mathcal{S}$ and $\omega^\star \leftarrow \omega(RD(\mathcal{S}))$. Output $\mathcal{S}^\star$ and $\omega^\star$, and terminate.
Otherwise set $C^* \leftarrow \{\, i \in V \mid y_i^* = 1 \,\}$ and increment $\mathcal{S} \leftarrow \mathcal{S} \cup \{C^*\}$. Return to Step 2.

---

Other than (2.2), various cut quality measures have been proposed in the literature, e.g., cut depth, which appears in [7], cut depth variant [92], cut angle, and cut sparsity. The auxiliary problem will be altered in accordance with selection of cut quality measures. See Amaldi *et al.* [4] and the references therein for the detail.

The initial subfamily $\mathcal{S}$ could be empty, but a clever choice may enhance the efficiency of the algorithm. In our experiments, we collected all singletons of $V$ to make the initial family $\mathcal{S}$.

It is also a crucial issue to select the solution method solving the problem $(RD(\mathcal{S}))$. In the simplex method, a warm-start strategy works effectively, that is, the simplex method can re-optimize a nearby problem in a few pivots by exploiting an optimal solution of the preceding problem, then the obtained solution is an extreme point of the optimal face of the polyhedron. Since there exist many extreme points in the case of degenerate LP problem, one would like to cut the whole optimal face. However the cutting plane generated from the extreme point may not cut off much more than this point. On the other hand, the solution retuned by the interior-point method is an interior point of the optimal face, then the cutting plane generated from this point may cut off the whole face although there is no advantage of the warm-start. Bixby *et al.* [16] proposed an approach combined the interior-point method with the simplex method, and showed the combined approach outperformed both of a pure interior-point method and a pure simplex method. Elhedhli and Goffin [39] proposed more elaborate method, which is called *analytic center cutting plane method*, ACCPM for short. ACCPM generates a cut at an analytic center of the feasible region for $(RD(\mathcal{S}))$.

When the algorithm terminates, we have solved $(RD)$, hence also $(RP)$, which usually admits a fractional optimal solution. The final family $\mathcal{S}^\star$ however would yield a set of primal variables that are likely to be positive at an optimal solution of the problem $(P)$. Then we propose to solve the following problem $(P(\mathcal{S}^\star))$ of variables $z_C$ with $C \in \mathcal{S}^\star$.

$$(P(\mathcal{S}^\star)) \quad \left| \begin{array}{lll} \text{maximize} & \displaystyle\sum_{C\in\mathcal{S}^\star} f_C z_C & \\ \text{subject to} & \displaystyle\sum_{C\in\mathcal{S}^\star} a_{iC} z_C = 1 & (i \in V) \\ & z_C \in \{0,1\} & (C \in \mathcal{S}^\star). \end{array} \right.$$

This problem is expected to have much fewer variables than problem $(P)$ does, so that it could be solved within a reasonable time by an IP solver, e.g., CPLEX, Gurobi and Xpress. Lacking variables $z_C$ with $C$ not in $\mathcal{S}^\star$, $(P(\mathcal{S}^\star))$ provides a lower bound of $\omega(P)$. Then

$$\omega(P(\mathcal{S}^\star)) \leq \omega(P) \leq \omega(RD(\mathcal{S}^\star)).$$

The value $\omega(RD(\mathcal{S}^\star)) - \omega(P(\mathcal{S}^\star))$ provides an upper bound of the difference between $\omega(P(\mathcal{S}^\star))$ and $\omega(P)$, hence the quality of the solution of $(P(\mathcal{S}^\star))$ given by an IP solver.

### 2.3.3 Bounding procedures and stopping criterion

The cutting plane algorithm is a powerful technique for large-scale integer programming problem. Generally, this algorithm often suffers from slow convergence, and many iterations may be needed to prove the optimality of $(RD)$ after the optimal value of the problem $(RD(\mathcal{S}))$ has reached the optimal value $\omega(RD)$. In this subsection, we develop methods to obtain nontrivial upper and lower bounds, and propose a stopping criterion.

To obtain an upper bound of $\omega(P)$, we recall the problem $(QP)$ presented in Subsection 2.2.1. We relax the first set of constraints and add them to the objective function as a penalty with Lagrangian multiplier vector $\boldsymbol{\lambda} \in \mathbb{R}^n$, and obtain the following *Lagrangian relaxation* problem:

$$(LRQP(\boldsymbol{\lambda})) \quad \left| \begin{array}{lll} \text{maximize} & \displaystyle\sum_{p\in T}\left(\frac{1}{m}\sum_{l\in E} x_{lp} - \frac{1}{4m^2}\left(\sum_{i\in V} d_i y_{ip}\right)^2\right) + \sum_{i\in V}\lambda_i(1 - \sum_{p\in T} y_{ip}) & \\ \text{subject to} & x_{lp} \leq y_{ip} & (l = \{i,j\} \in E, p \in T) \\ & x_{lp} \leq y_{jp} & (l = \{i,j\} \in E, p \in T) \\ & x_{lp} \in \{0,1\} & (l \in E, p \in T) \\ & y_{ip} \in \{0,1\} & (i \in V, p \in T). \end{array} \right.$$

Owing to the absence of the first set of constraints of the problem $(QP)$, the problem $(LRQP(\boldsymbol{\lambda}))$ is decomposable into $t$ subproblems. For each $p \in T$, let us denote the subproblem by $(LRQP(\boldsymbol{\lambda}, p))$, then the subproblem $(LRQP(\boldsymbol{\lambda}, p))$ is as follows:

$$(LRQP(\boldsymbol{\lambda}, p)) \quad \left| \begin{array}{lll} \text{maximize} & \dfrac{1}{m}\displaystyle\sum_{l\in E} x_{lp} - \dfrac{1}{4m^2}\left(\sum_{i\in V} d_i y_{ip}\right)^2 - \sum_{i\in V}\lambda_i y_{ip} & \\ \text{subject to} & x_{lp} \le y_{ip} & (l = \{i,j\} \in E) \\ & x_{lp} \le y_{jp} & (l = \{i,j\} \in E) \\ & x_{lp} \in \{0,1\} & (l \in E) \\ & y_{ip} \in \{0,1\} & (i \in V). \end{array} \right.$$

The optimal value of $(LRQP(\boldsymbol{\lambda}))$ is given via the optimal value $\omega(LRQP(\boldsymbol{\lambda}, p))$ as follows:

$$\begin{aligned} \omega(LRQP(\boldsymbol{\lambda})) &= \sum_{p\in T} \max_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{F}(LRQP(\boldsymbol{\lambda},p))} \left\{ \frac{1}{m}\sum_{l\in E} x_{lp} - \frac{1}{4m^2}\left(\sum_{i\in V} d_i y_{ip}\right)^2 - \sum_{i\in V}\lambda_i y_{ip} \right\} + \sum_{i\in V}\lambda_i \\ &= \sum_{p\in T} \omega(LRQP(\boldsymbol{\lambda}, p)) + \sum_{i\in V}\lambda_i. \end{aligned}$$

Note that each subproblem $(LRQP(\boldsymbol{\lambda}, p))$ has the same optimal value regardless of index $p \in T$ and the problem $(LRQP(\boldsymbol{\lambda}, p))$ is equivalent to $(AP(\boldsymbol{\lambda}))$. Therefore we have the following equation:

$$\omega(LRQP(\boldsymbol{\lambda})) = t \cdot \omega(AP(\boldsymbol{\lambda})) + \sum_{i\in V}\lambda_i. \tag{2.3}$$

When we set the number $t$ to an upper bound of the number of communities at an optimal solution of $(P)$, the problem $(QP)$ is a relaxation problem of $(P)$, hence the optimal value of the problem $(LRQP(\boldsymbol{\lambda}))$ provides an upper bound of $\omega(P)$ for any $\boldsymbol{\lambda} \in \mathbb{R}^n$.

**Proposition 2.9** *Let $t$ be an upper bound of the number of communities at an optimal solution of $(P)$. Then* (2.3) *is an upper bound of $\omega(P)$ for any $\boldsymbol{\lambda} \in \mathbb{R}^n$.*

**Proof :** Clear from the above discussion. ■

Due to arbitrariness of $\boldsymbol{\lambda} \in \mathbb{R}^n$ in Proposition 2.9, the proposition holds for an optimal solution obtained at each iteration of the algorithm. Thus we can obtain an upper bound of $\omega(P)$ without additional computation.

Next we describe the heuristic algorithm to obtain a feasible solution of the problem $(P)$, which is based on a simple rounding procedure. First, we derive an optimal solution

$\boldsymbol{z}(\mathcal{S})$ of $(RP(\mathcal{S}))$ from the optimal dual solution $\boldsymbol{\lambda}$ obtained at Step 2 of the cutting plane algorithm. Since this primal solution is usually a fractional solution, we construct an integer solution $\bar{\boldsymbol{z}} = (\bar{z}_C)_{C\in\mathcal{S}}$ by rounding $\bar{z}_C = 1$ if $z_C(\mathcal{S}) > 0$, and $\bar{z}_C = 0$ otherwise. Let $\mathcal{C}$ denote a sub-family represented by the solution $\bar{\boldsymbol{z}}$ for succinct notation. The sub-family $\mathcal{C}$ is not necessarily a partition of $V$, but a cover of $V$ due to the feasibility of $\boldsymbol{z}(\mathcal{S})$ for the problem $(RP(\mathcal{S}))$. Thus it is likely that some communities overlap with each other. For a given $\mathcal{C}$, we define the following set

$$M(\mathcal{C}, i) = \{\, C \in \mathcal{C} \mid i \in C \,\}$$

in order to check whether $\mathcal{C}$ is a partition of $V$. When the cardinality of $M(\mathcal{C}, i)$ is equal to 1 for any $i \in V$, the sub-family $\mathcal{C}$ is a partition of $V$, hence we can obtain a lower bound of $\omega(P)$. If $|M(\mathcal{C}, i)| > 1$ holds for some $i \in V$, then we compute the variation $\Delta(i, C)$ of the contribution $f_C$ when node $i$ is removed from a community $C$,

$$\Delta(i, C) = \frac{1}{m} \sum_{j\in C} w_{ij}$$

for each $C \in M(\mathcal{C}, i)$. Let $C^*$ be the community that minimizes $\Delta(i, C)$ over $M(\mathcal{C}, i)$, and we remove the index $i$ from any $C \in M(\mathcal{C}, i) \setminus C^*$. From the above discussion, the rounding heuristics is described as follows.

**Procedure Rounding-Heuristics (RH$(\mathcal{S})$)**

**Input** : a current family $\mathcal{S}$

**Output** : a lower bound $\ell(P)$

Step 1 : Let $\boldsymbol{z}(\mathcal{S})$ be an optimal solution of $(RP(\mathcal{S}))$.

Step 2 : Construct $\bar{\boldsymbol{z}}$ from $\boldsymbol{z}(\mathcal{S})$ by the rounding procedure. Set $\mathcal{C}$ be a sub-family represented by $\bar{\boldsymbol{z}}$.

Step 3 : If $|\, M(\mathcal{C}, i) \,| = 1$ for any $i \in V$, then calculate $f_C$ for any $C \in \mathcal{C}$. Set $\ell(P) \leftarrow \sum_{C\in\mathcal{C}} f_C$, and terminate.

Step 4 : Otherwise select a minimum index $i \in V$ such that $|\, M(\mathcal{C}, i) \,| > 1$.
Compute $\Delta(i, C)$ for any $C \in M(\mathcal{C}, i)$, and set

$$C^* \leftarrow \text{argmin}\{\, \Delta(i, C) \mid C \in M(\mathcal{C}, i) \,\}.$$

For all $C \in M(\mathcal{C}, i) \setminus C^*$, set $C \leftarrow C \setminus \{i\}$.

Step 5 : Update the sub-family $\mathcal{C}$ according to the modification at Step 4, and return to Step 3.

---

Now we consider the stopping criterion of our algorithm. If the difference between the upper bound and $\omega(RD(\mathcal{S}))$ is small, we can stop the algorithm even if $\omega(AP(\boldsymbol{\lambda})) \leq 0$ does not hold. Then we use the following condition as one of the stopping criterion of our algorithm for a predetermined parameter $\varepsilon \geq 0$:

$$\frac{\text{UB} - \omega(RD(\mathcal{S}))}{\text{UB}} \leq \varepsilon,$$

where UB is the smallest upper bound obtained so far.

### 2.3.4 Pegging test

In this subsection we consider the problem reduction method for the problem $(P(\mathcal{S}^\star))$. We can identify the variables which take either one or zero at the optimal solution of $(P(\mathcal{S}^\star))$, this well-known technique is called *pegging test*. In order to explain the pegging test, we first consider the Lagrangian relaxation problem of $(P(\mathcal{S}^\star))$. Introducing a multiplier vector $\boldsymbol{\lambda} \in \mathbb{R}^n$ for the first set of constraints, we obtain the following Lagrangian relaxation problem $(LR(\mathcal{S}^\star, \boldsymbol{\lambda}))$ with only the binary variable constraints:

$$(LR(\mathcal{S}^\star, \boldsymbol{\lambda})) \quad \left| \begin{array}{ll} \text{maximize} & \displaystyle\sum_{C \in \mathcal{S}^\star} f_C z_C + \sum_{i \in V} \lambda_i (1 - \sum_{C \in \mathcal{S}^\star} a_{iC} z_C) \\ \text{subject to} & z_C \in \{0,1\} \qquad (C \in \mathcal{S}^\star). \end{array} \right.$$

From (2.2), the objective function of $(LR(\mathcal{S}^\star, \boldsymbol{\lambda}))$ is written as

$$\sum_{C \in \mathcal{S}^\star} \gamma_C(\boldsymbol{\lambda}) z_C + \sum_{i \in V} \lambda_i.$$

For a given multiplier vector $\boldsymbol{\lambda} \in \mathbb{R}^n$, we can obtain an optimal solution $\boldsymbol{z}(\boldsymbol{\lambda})$ of $(LR(\mathcal{S}^\star, \boldsymbol{\lambda}))$ by simply setting $z_C(\boldsymbol{\lambda}) = 1$ if $\gamma_C(\boldsymbol{\lambda}) > 0$, and $z_C(\boldsymbol{\lambda}) = 0$ otherwise.

**Proposition 2.10** *Let LB be a lower bound of $(P(\mathcal{S}^\star))$ and $\boldsymbol{\lambda}$ be an optimal solution of $(RD(\mathcal{S}^\star))$, respectively. If $\sum_{i \in V} \lambda_i + \gamma_C(\boldsymbol{\lambda}) <$ LB, then $z_C = 0$ for any optimal solution of the problem $(P(\mathcal{S}^\star))$.*

**Proof :** Since the problem $(LR(\mathcal{S}^\star, \boldsymbol{\lambda}))$ is a relaxation problem of $(P(\mathcal{S}^\star))$, we have

$$\omega(P(\mathcal{S}^\star \mid z_C = 1)) \leq \omega(LR(\mathcal{S}^\star, \boldsymbol{\lambda} \mid z_C = 1)). \tag{2.4}$$

For any $C \in \mathcal{S}^\star$, $\gamma_C(\boldsymbol{\lambda})$ is non-positive due to the feasibility of $\boldsymbol{\lambda}$ for the problem $(RD(\mathcal{S}^\star))$, hence $\omega(LR(\mathcal{S}^\star, \boldsymbol{\lambda})) = \sum_{i \in V} \lambda_i$ holds. Moreover we obtain

$$\omega(LR(\mathcal{S}^\star, \boldsymbol{\lambda} \mid z_C = 1)) = \sum_{i \in V} \lambda_i + \gamma_C(\boldsymbol{\lambda}). \tag{2.5}$$

From (2.4), (2.5) and the assumption, we have the following inequality

$$\omega(P(\mathcal{S}^\star \mid z_C = 1)) \leq \sum_{i \in V} \lambda_i + \gamma_C(\boldsymbol{\lambda}) < \text{LB}.$$

The above inequality $\omega(P(\mathcal{S}^\star \mid z_C = 1)) < \text{LB}$ implies the non-existence of the optimal solution that satisfies $z_C = 1$. Therefore $z_C = 0$ for any optimal solution of the problem $(P(\mathcal{S}^\star))$. ■

### 2.3.5 Whole algorithm

Our proposed algorithm is described as follows. Note that $\mathcal{S}$ is incremented by a single set $C$ determined by $\boldsymbol{y}^*$ found in Step 3. We will call this algorithm the *Single-Cutting-Plane-at-a-Time Algorithm*, SCP for short. The algorithm consists of two phases: the first phase is to find binding constraints at an optimal solution of the original $(RD)$ and to obtain upper and lower bounds of $\omega(P)$ (Step 1 to Step 4 in SCP), the second phase is to solve the problem $(P(\mathcal{S}^\star))$ constructed from $\mathcal{S}^\star$ obtained in the first phase (Step 5 in SCP).

**Algorithm Single-Cutting-Plane-at-a-Time (SCP)**

Step 1 : Let $\mathcal{S}$ be an initial family of nonempty subsets of $V$ and $\varepsilon$ be a tolerance parameter, respectively. Initialize an upper bound UB and a lower bound LB by setting UB $\leftarrow 1$ and LB $\leftarrow 0$.

Step 2 : Solve $(RD(\mathcal{S}))$ to obtain an optimal solution $\boldsymbol{\lambda}$ and the optimal value $\omega(RD(\mathcal{S}))$. Call the procedure RH($\mathcal{S}$) to compute a lower bound $\ell(P)$ of $\omega(P)$. If LB $<$ $\ell(P)$, then LB $\leftarrow \ell(P)$.

Step 3 : Solve $(AP(\boldsymbol{\lambda}))$ and set $\boldsymbol{y}^*$ be an optimal solution. Compute an upper bound $u(P)$ according to Proposition 2.9. If UB $> u(P)$, then UB $\leftarrow u(P)$.

Step 4 : If $\omega(AP(\boldsymbol{\lambda})) \leq 0$ or $(\text{UB} - \omega(RD(\mathcal{S})))/\text{UB} \leq \varepsilon$, then set $\mathcal{S}^\star \leftarrow \mathcal{S}, \omega^\star \leftarrow \omega(RD(\mathcal{S}))$, LB$^\star \leftarrow$ LB and UB$^\star \leftarrow$ UB. Go to Step 5.
Otherwise set $C \leftarrow \{i \in V \mid y_i^* = 1\}$, increment $\mathcal{S} \leftarrow \mathcal{S} \cup \{C\}$. Go to Step 2.

Step 5 : Execute the pegging test according to Proposition 2.10. Solve $(P(\mathcal{S}^\star))$ to obtain an optimal integer solution.

---

Carrying out some preliminary experiments by SCP, we frequently observed that the optimal value $\omega(RD(\mathcal{S}))$ stays constant for many iterations even when $\mathcal{S}$ is repeatedly incremented. Take "Karate" for instance, we show in Table 2.1 and Figure 2.1 how slowly $\omega(RD(\mathcal{S}))$ increases as the algorithm SCP progresses. Here "Karate" is the benchmark instance provided by DIMACS. The slow convergence we observed may arise from a

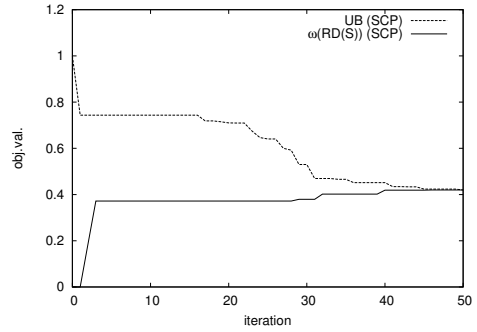



Figure 2.1: Behavior of SCP for Karate

Table 2.1: Plateau situation of SCP

| Karate | |
|---|---|
| iteration | $\omega(RD(\mathcal{S}))$ |
| 0 | 0.00000 |
| 10 | 0.37179 |
| 20 | 0.37179 |
| 30 | 0.37179 |
| 40 | 0.38047 |
| 49 | 0.41979 |

particular structure of $(RD(\mathcal{S}))$ that all coefficients of the objective function are one and all coefficients of the constraints are either zero or one. This makes the contour of the objective function and some face of $\mathcal{F}(RD(\mathcal{S}))$ be parallel, and the whole face be optimal. As a consequence, the optimal value $\omega(RD(\mathcal{S}))$ stays constant although lots of cutting planes are added. To cut off such a face entirely, we propose to simultaneously add multiple cutting planes which may complement well each other.

The first cutting plane is the same as the one defined by $\boldsymbol{y}^*$ and $f^*$ obtained from problem $(AP(\boldsymbol{\lambda}))$. We then fix the variables $y_i$ to zero for all $i$ with $y_i^* = 1$, and consider

$(AP(\boldsymbol{\lambda}))$. More precisely, we let $V^{(1)} = \{\, i \in V \mid y_i^* = 1 \,\}$ and approximately solve the problem

$$(AP(\boldsymbol{\lambda}, V^{(1)})) \quad \left| \begin{array}{lll} \text{maximize} & \dfrac{1}{m}\displaystyle\sum_{l\in E} x_l - \dfrac{1}{4m^2}\left(\sum_{i\in V} d_i y_i\right)^2 - \sum_{i\in V}\lambda_i y_i & \\ \text{subject to} & x_l \le y_i & (l = \{i,j\} \in E) \\ & x_l \le y_j & (l = \{i,j\} \in E) \\ & x_l \in \{0,1\} & (l \in E) \\ & y_i \in \{0,1\} & (i \in V \setminus V^{(1)}) \\ & y_i = 0 & (i \in V^{(1)}) \end{array} \right.$$

to obtain $\boldsymbol{y}^{(1)}$ and $f^{(1)}$, i.e., the second cutting plane. In a general step, with $V^{(h)} = \{\, i \in V \mid y_i^{(l)} = 1 \text{ for some } l < h \,\}$ we approximately solve

$$(AP(\boldsymbol{\lambda}, V^{(h)})) \quad \left| \begin{array}{lll} \text{maximize} & \dfrac{1}{m}\displaystyle\sum_{l\in E} x_l - \dfrac{1}{4m^2}\left(\sum_{i\in V} d_i y_i\right)^2 - \sum_{i\in V}\lambda_i y_i & \\ \text{subject to} & x_l \le y_i & (l = \{i,j\} \in E) \\ & x_l \le y_j & (l = \{i,j\} \in E) \\ & x_l \in \{0,1\} & (l \in E) \\ & y_i \in \{0,1\} & (i \in V \setminus V^{(h)}) \\ & y_i = 0 & (i \in V^{(h)}), \end{array} \right.$$

where $\boldsymbol{y}^{(0)} = \boldsymbol{y}^*$ and $V^{(0)} = \emptyset$. As long as $\omega(AP(\boldsymbol{\lambda}, V^{(h)}))$ is positive, we keep on generating cutting planes. When $\omega(AP(\boldsymbol{\lambda}, V^{(h)}))$ becomes non-positive, we add all the cutting planes obtained so far to $\mathcal{S}$. The Step 4 of the algorithm SCP should be modified as follows. We will call the algorithm with this modification the *Multiple-Cutting-Planes-at-a-Time Algorithm*, MCP for short.

**Step 4 of the Multiple-Cutting-Planes-at-a-Time Algorithm**

Step 4 : If $\omega(AP(\boldsymbol{\lambda})) \le 0$ or $(\text{UB} - \omega(RD(\mathcal{S})))/\text{UB} \le \varepsilon$, set $\mathcal{S}^\star \leftarrow \mathcal{S}$, $\omega^\star \leftarrow \omega(RD(\mathcal{S}))$, $\text{LB}^\star \leftarrow \text{LB}$ and $\text{UB}^\star \leftarrow \text{UB}$. Go to Step 5.
Otherwise generate cutting planes until $\omega(AP(\boldsymbol{\lambda}, V^{(h)}))$ becomes non-positive. Add all the cutting planes generated to $\mathcal{S}$. Go to Step 2.

### 2.3.6 Computational experiments

We report the computational experiments with algorithms SCP and MCP. The experiment was performed on a PC with Intel Core i7, 3.70 GHz processor and 32.0 GB of memory. We implemented the algorithms in Python 2.7. Giving priority to generate stronger cuts, we used the barrier method in Gurobi 6.0.0 as an LP solver to solve the problem $(RD(\mathcal{S}))$. Mitchell [71] pointed out that the barrier method can re-optimize a nearby problem more quickly than the simplex method does in the situation where many cuts are added to the preceding problem at a time, hence it is possibly advantageous for the algorithm MCP to use the barrier method.

In order to evaluate the performance of our algorithms, we solved six benchmark instances; Zachary's karate club dataset [94] representing friendship relationships between members of a club, Lusseau's dolphins dataset [66] representing communications between dolphins in Doubtful Sound New Zealand, Hugo's Les Misèrables dataset [57] representing characters in Victor Hugo's famous novel and their interactions, Krebs' political books dataset [58] representing co-purchasing of the books on U.S. politics by the same buyer, Football dataset representing the schedule of football games between American college teams [46], and s838 [70] dataset representing electronic circuits. The size, the known optimal value $\omega(P)$ and the optimal number of communities $t^\star$ of each instance are given in Table 2.2.

Table 2.2: Solved instances

| ID | name | $n$ | $m$ | $\omega(P)$ | $t^\star$ |
|---|---|---|---|---|---|
| 1 | Karate | 34 | 78 | 0.41979 | 4 |
| 2 | Dolphins | 62 | 159 | 0.52852 | 5 |
| 3 | Les Misèrables | 77 | 254 | 0.56001 | 6 |
| 4 | Books | 105 | 441 | 0.52724 | 4 |
| 5 | Football | 115 | 613 | 0.60457 | 10 |
| 6 | s838 | 512 | 819 | 0.8194 | 12 |

We set $\mathcal{S}$ initially to the family of all singletons, i.e., $\mathcal{S} = \{\{1\}, ..., \{n\}\}$, and set the parameter $t$, which is used in calculating an upper bound, to the optimal number of

communities $t^\star$.

We first conduct experiment with a tolerance parameter $\varepsilon = 0$, that is, SCP and MCP keep on generating cuts as long as $\omega(AP(\boldsymbol{\lambda})) > 0$. Table 2.3 shows the results of both algorithms for each instance. The collected statistics in this table are given as follows:

- iteration : number of solving the problem $(RD(\mathcal{S}))$
- $|\mathcal{S}^\star|$ : cardinality of the final family of subsets $\mathcal{S}^\star$
- $\omega^\star$ : optimal value of $(RD(\mathcal{S}^\star))$ obtained at the end of the first phase
- LB$^\star$ : lower bound at the end of the first phase
- UB$^\star$ : upper bound at the end of the first phase
- $\omega(P(\mathcal{S}^\star))$ : optimal value of $(P(\mathcal{S}^\star))$
- gap (%) : relative gap defined by gap $= \left(\frac{\omega(P)-\omega(P(\mathcal{S}^\star))}{\omega(P)}\right) \times 100$
- time 1 : computation time of the first phase in seconds
- time 2 : computation time of the second phase in seconds
- pegged : number of pegged variables

The symbol "*" in the columns "time 2" and "pegged" represents that the second phase is not executed since the difference between the obtained LB$^\star$ and UB$^\star$ vanishes at the end of the first phase of the algorithm, hence this implies that the corresponding instance is solved to optimality. The symbol "OT" in the column "time 1" represents that the algorithm does not terminate after more than $604{,}800$ seconds, seven days.

From Table 2.3, we observe that the number of generated constraints is much smaller than that of the original problem in both algorithms. Take Karate with $34$ nodes for instance, the generated constraints are less than $1/10^8$ of the original constraints totaling $1.7 \times 10^{10}$. To compare SCP with MCP, we observe that the number of iterations and the computation time for the first phase of MCP are much smaller than those of SCP. We also see that SCP and MCP solve the instances, except for s838 (ID=6), and prove the optimality due to the vanishing gap between the upper and lower bounds. Regarding to the instance s838 (ID=6), SCP does not terminate over $604{,}800$ seconds, while MCP takes

Table 2.3: Computational results of SCP and MCP ($\varepsilon = 0$)

| ID | iteration | | $\lvert\mathcal{S}^\star\rvert$ | | $\omega^\star$ | | LB$^\star$ | | UB$^\star$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SCP | MCP | SCP | MCP | SCP | MCP | SCP | MCP | SCP | MCP |
| 1 | 49 | 24 | 83 | 84 | 0.41979 | 0.41979 | 0.41979 | 0.41979 | 0.41979 | 0.41979 |
| 2 | 100 | 38 | 162 | 170 | 0.52852 | 0.52852 | 0.52852 | 0.52852 | 0.52852 | 0.52852 |
| 3 | 130 | 39 | 207 | 181 | 0.56001 | 0.56001 | 0.56001 | 0.56001 | 0.56001 | 0.56001 |
| 4 | 160 | 63 | 265 | 268 | 0.52724 | 0.52724 | 0.52724 | 0.52724 | 0.52724 | 0.52724 |
| 5 | 472 | 47 | 587 | 238 | 0.60457 | 0.60457 | 0.60457 | 0.60457 | 0.60457 | 0.60457 |
| 6 | NA | 1265 | NA | 9756 | NA | 0.81936 | NA | 0.81936 | NA | 0.81936 |
| ID | $\omega(P(\mathcal{S}^\star))$ | | gap (%) | | time 1 (sec.) | | time 2 (sec.) | | pegged | |
| | SCP | MCP | SCP | MCP | SCP | MCP | SCP | MCP | SCP | MCP |
| 1 | 0.41979 | 0.41979 | 0.000 | 0.000 | 6.176 | 3.809 | * | * | * | * |
| 2 | 0.52852 | 0.52852 | 0.000 | 0.000 | 42.392 | 18.703 | * | * | * | * |
| 3 | 0.56001 | 0.56001 | 0.000 | 0.000 | 83.397 | 29.923 | * | * | * | * |
| 4 | 0.52724 | 0.52724 | 0.000 | 0.000 | 214.049 | 90.278 | * | * | * | * |
| 5 | 0.60457 | 0.60457 | 0.000 | 0.000 | 3173.713 | 178.723 | * | * | * | * |
| 6 | NA | 0.81936 | NA | 0.005 | OT | 37877.979 | NA | 1.809 | NA | 9742 |

less than approximately $40{,}000$ seconds and the gap is less than $0.005\%$. In addition, for the instance s838 (ID=6), the number of pegged variables in MCP is more than $99\%$ of the whole set of variables.

To compere the performance of several existing heuristics and our algorithms, we give the lower bounds on the modularity obtained by existing heuristics and our algorithms in Table 2.4. The columns GN, CNM, SA, SD, CHL, and NR represent the Girvan-Newman algorithm [46], the hierarchical agglomerative method by Clauset *et al.* [26], the simulated annealing by Guimerá and Amaral [51], the Newman's spectral divisive method [77], the divisive method by Cafieri *et al.* [20], and the Noack-Rotta heuristics [79], respectively. From Table 2.4, we confirm that the lower bounds obtained by MCP are equal to or larger than those obtained by the existing heuristics for all instances, which indicates that algorithm MCP is superior to the existing heuristics in terms of accuracy of the solution.

Table 2.4: Lower bounds by several existing heuristics, SCP and MCP

| ID | GN | CNM | SA | SD | CHL | NR | SCP | MCP |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.401 | 0.38067 | 0.420 | 0.39341 | 0.41880 | NA | 0.41979 | 0.41979 |
| 2 | 0.520 | 0.49549 | 0.527 | 0.49120 | 0.52646 | 0.52377 | 0.52852 | 0.52852 |
| 3 | 0.540 | 0.50060 | 0.556 | 0.51383 | 0.54676 | 0.56001 | 0.56001 | 0.56001 |
| 4 | NA | 0.50197 | 0.527 | 0.46718 | 0.52629 | 0.52694 | 0.52724 | 0.52724 |
| 5 | 0.601 | 0.57728 | 0.604 | 0.49261 | 0.60091 | 0.60028 | 0.60457 | 0.60457 |
| 6 | NA | 0.80556 | NA | 0.73392 | 0.81663 | 0.81624 | NA | 0.81936 |

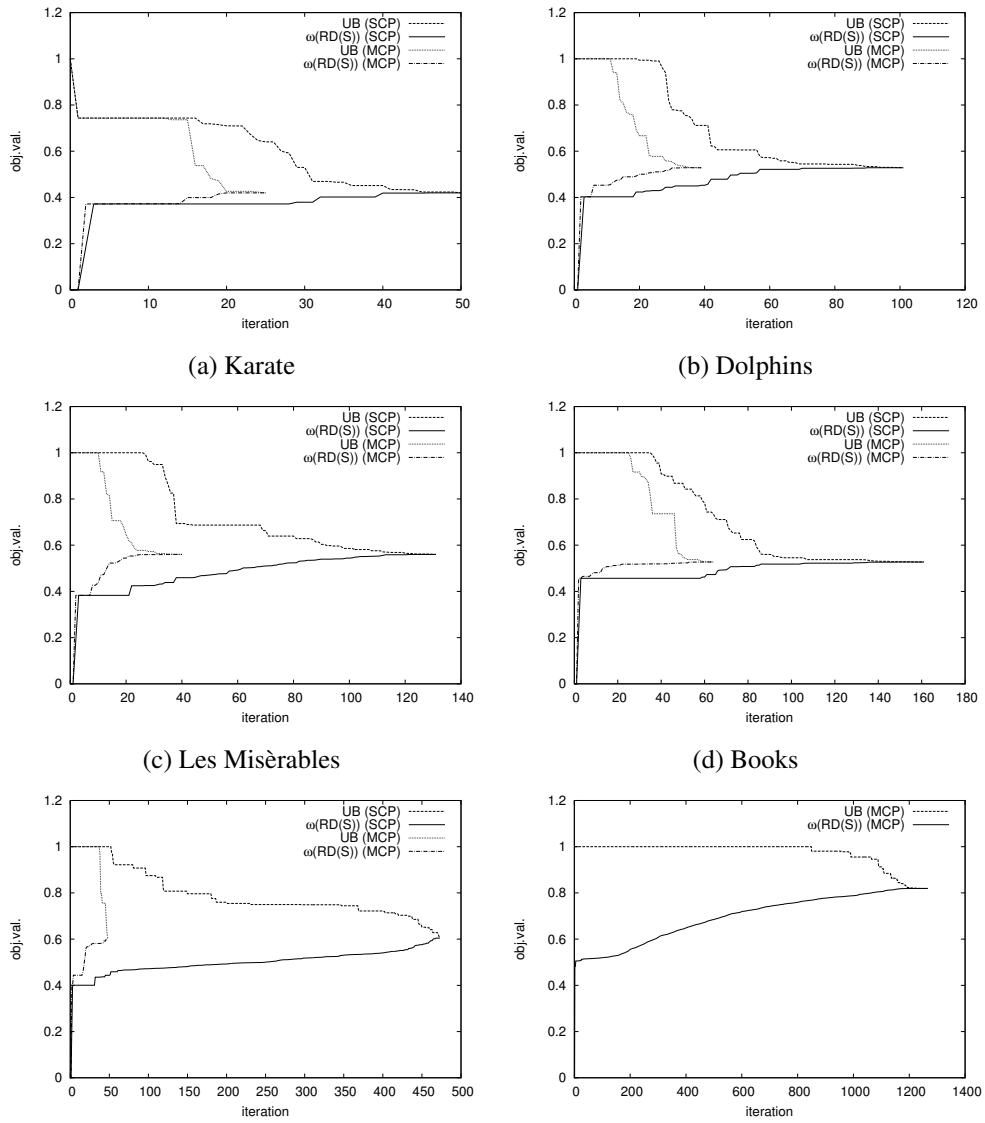
Next, we conduct experiment with a tolerance parameter $\varepsilon = 0.03$, and the computational results are given in Table 2.5. We see that the number of iterations and the computation time for the first phase of this experiment are smaller than those of the experiment with $\varepsilon = 0$, but their reductions are not significant for all instances except for the instance s838 (ID=6). Also, we observe that the predominant portion of the whole computation is spent for the first phase, and the second phase of the algorithm requires a fraction of the whole.

As for the accuracy of the solution, although both of SCP and MCP could not solve the instances, except for the instance Football (ID=5), to optimality, the relative gap was less than $1.5\%$ for each instance.

Table 2.5: Computational results of SCP and MCP ($\varepsilon = 0.03$)

| ID | iteration | | $\lvert\mathcal{S}^\star\rvert$ | | $\omega^\star$ | | LB$^\star$ | | UB$^\star$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SCP | MCP | SCP | MCP | SCP | MCP | SCP | MCP | SCP | MCP |
| 1 | 45 | 20 | 79 | 77 | 0.41880 | 0.41979 | 0.41880 | 0.41979 | 0.42352 | 0.42639 |
| 2 | 86 | 32 | 148 | 159 | 0.52680 | 0.52761 | 0.52632 | 0.52761 | 0.54308 | 0.57244 |
| 3 | 112 | 28 | 189 | 161 | 0.55608 | 0.55909 | 0.54823 | 0.55909 | 0.57105 | 0.57244 |
| 4 | 107 | 52 | 212 | 242 | 0.52184 | 0.52344 | 0.52037 | 0.51797 | 0.53767 | 0.53831 |
| 5 | 471 | 47 | 586 | 238 | 0.60441 | 0.60457 | 0.60441 | 0.60457 | 0.62114 | 0.60457 |
| 6 | NA | 1169 | NA | 9456 | NA | 0.81845 | NA | 0.81677 | NA | 0.84031 |
| ID | $\omega(P(\mathcal{S}^\star))$ | | gap (%) | | time 1 (sec.) | | time 2 (sec.) | | pegged | |
| | SCP | MCP | SCP | MCP | SCP | MCP | SCP | MCP | SCP | MCP |
| 1 | 0.41880 | 0.41979 | 0.235 | 0.000 | 5.590 | 2.853 | 0.232 | 0.230 | 0 | 0 |
| 2 | 0.52632 | 0.52761 | 0.415 | 0.172 | 35.994 | 15.227 | 0.080 | 0.943 | 136 | 0 |
| 3 | 0.55243 | 0.55909 | 1.354 | 0.165 | 71.244 | 21.080 | 0.644 | 1.244 | 105 | 0 |
| 4 | 0.52036 | 0.52262 | 1.305 | 0.875 | 134.659 | 68.383 | 0.306 | 1.974 | 185 | 70 |
| 5 | 0.60441 | 0.60457 | 0.027 | 0.000 | 3131.320 | 178.534 | 7.519 | * | 0 | * |
| 6 | NA | 0.81723 | NA | 0.265 | NA | 33401.686 | NA | 162.054 | NA | 8137 |

Figures 2.2 (a), (b), (c), (d), (e), and (f) show the upper bound and $\omega(RD(\mathcal{S}))$ vs. the number of iterations. For both algorithms, $\omega(RD(\mathcal{S}))$ rapidly increases at an early stage, and then increases slowly as the algorithm progresses.




(a) Karate (b) Dolphins

(c) Les Misèrables (d) Books

(e) Football (f) s838

Figure 2.2: Behaviors of SCP and MCP for each instance

## 2.4 Lagrangian Relaxation Algorithms

As previously mentioned, the LP relaxation provides an upper bound of the optimal value of $(P)$ and the obtained upper bound is often tight. However, the LP relaxation usually suffers high degeneracy due to the set partitioning constraints. To overcome this degeneracy, several techniques have been proposed in the literature. For vehicle-routing and crew-scheduling problems based on the set partitioning formulation, Elhallaoui *et al.* [38] proposed a dynamic constraints aggregation method, which reduces the size of the original set partitioning problem by aggregating some of the constraints. More recently, Benchimol *et al.* [13] developed a stabilized dynamic constraints aggregation which combines the dynamic constraints aggregation and the stabilized column generation by Du Merle [37].

In this section, we propose column generation algorithms which solve the Lagrangian relaxation problem for the restricted problem with fewer variables of $(P)$ instead of the LP relaxation throughout the column generation process. The algorithms we propose here are attractive owing to its low computational burden and low memory consumption compared to the LP relaxation algorithms. In addition, we discuss the convergence issue of our algorithm, then present an improved algorithm which converges a finite number of iterations.

### 2.4.1 Lagrangian relaxation and Lagrangian dual problem

We relax the set partitioning constraints by adding them to the objective function as a penalty with the Lagrangian multiplier vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)^\top \in \mathbb{R}^n$. Then we obtain the following Lagrangian relaxation problem $(LR(\mathcal{P}, \boldsymbol{\lambda}))$ having only binary variable constraints:

$$(LR(\mathcal{P}, \boldsymbol{\lambda})) \quad \left| \begin{array}{ll} \text{maximize} & \displaystyle\sum_{C \in \mathcal{P}} f_C z_C + \sum_{i \in V} \lambda_i (1 - \sum_{C \in \mathcal{P}} a_{iC} z_C) \\ \text{subject to} & z_C \in \{0, 1\} \qquad (C \in \mathcal{P}). \end{array} \right.$$

To write the objective function of $(LR(\mathcal{P}, \boldsymbol{\lambda}))$ in a simple form, we define the coefficient of variable $z_C$ in the objective function as

$$\gamma_C(\boldsymbol{\lambda}) = f_C - \sum_{i \in V} \lambda_i a_{iC}.$$

Using $\gamma_C(\boldsymbol{\lambda})$, the objective function is written as follows:

$$L(\boldsymbol{z}, \boldsymbol{\lambda}) = \sum_{C \in \mathcal{P}} \gamma_C(\boldsymbol{\lambda}) z_C + \sum_{i \in V} \lambda_i.$$

For a given multiplier vector $\boldsymbol{\lambda} \in \mathbb{R}^n$, we can obtain an optimal solution $\boldsymbol{z}(\boldsymbol{\lambda}) = (z_C(\boldsymbol{\lambda}))_{C \in \mathcal{P}}$ of $(LR(\mathcal{P}, \boldsymbol{\lambda}))$ by simply setting $z_C(\boldsymbol{\lambda}) = 1$ if $\gamma_C(\boldsymbol{\lambda}) > 0$, and $z_C(\boldsymbol{\lambda}) = 0$ otherwise. Then the optimal value $\omega(LR(\mathcal{P}, \boldsymbol{\lambda}))$ provides an upper bound of $\omega(P)$ for any $\boldsymbol{\lambda} \in \mathbb{R}^n$. The upper bound given by the problem $(LR(\mathcal{P}, \boldsymbol{\lambda}))$ depends on a choice of the Lagrangian multiplier vector $\boldsymbol{\lambda}$. The problem of finding the best upper bound of $\omega(P)$ is called the Lagrangian dual problem $(LD)$, which is given as:

$$(LD) \quad \left| \begin{array}{ll} \text{minimize} & \omega(LR(\mathcal{P}, \boldsymbol{\lambda})) \\ \text{subject to} & \boldsymbol{\lambda} \in \mathbb{R}^n. \end{array} \right.$$

The objective function $\omega(LR(\mathcal{P}, \boldsymbol{\lambda}))$ of $(LD)$ is a piecewise linear convex with respect to $\boldsymbol{\lambda} \in \mathbb{R}^n$, but sub-differentiable at its breakpoints. One of the most commonly used method for this problem is the *subgradient method*.

In the remainder of this subsection, we see some theoretical properties between the problems $(RD)$ and $(LD)$. Now we consider LP relaxation problem of $(LR(\mathcal{P}, \boldsymbol{\lambda}))$, i.e., replacing the binary constraint $z_C \in \{0, 1\}$ by $0 \leq z_C \leq 1$, and denote it by $(\overline{LR}(\mathcal{P}, \boldsymbol{\lambda}))$.

$$(\overline{LR}(\mathcal{P}, \boldsymbol{\lambda})) \quad \left| \begin{array}{ll} \text{maximize} & L(\boldsymbol{z}, \boldsymbol{\lambda}) = \displaystyle\sum_{C \in \mathcal{P}} \gamma_C(\boldsymbol{\lambda}) z_C + \sum_{i \in V} \lambda_i \\ \text{subject to} & 0 \leq z_C \leq 1 \qquad (C \in \mathcal{P}). \end{array} \right.$$

Clearly, any optimal solution of $(\overline{LR}(\mathcal{P}, \boldsymbol{\lambda}))$ is also optimal for $(LR(\mathcal{P}, \boldsymbol{\lambda}))$, which is called *integrality property*. Therefore the optimal value of $(LD)$ coincides with that of the problem $(RD)$. See Geoffrion [44].

### 2.4.2 Column generation method

As we have discussed in the previous subsection, the optimal solution $\boldsymbol{z}(\boldsymbol{\lambda})$ of $(LR(\mathcal{P}, \boldsymbol{\lambda}))$ can be obtained by checking the sign of $\gamma_C(\boldsymbol{\lambda})$. In practice, it is hard to compute all $\gamma_C(\boldsymbol{\lambda})$'s owing to the huge number of variables. However, the number of variables which are positive at an optimal solution of $(P)$ is at most the number of nodes, hence we need only a small number of variables. Based on this fact, we use the column generation technique in order to alleviate the computation burden. In this subsection, we first introduce

restricted problems and explain the subgradient method. Second, we apply the column generation technique.

We choose a small subfamily $\mathcal{S}$ of $\mathcal{P}$ and consider the following restricted problem $(P(\mathcal{S}))$ with fewer variables:

$$(P(\mathcal{S})) \quad \left| \begin{array}{lll} \text{maximize} & \displaystyle\sum_{C\in\mathcal{S}} f_C z_C & \\ \text{subject to} & \displaystyle\sum_{C\in\mathcal{S}} a_{iC} z_C = 1 & (i \in V) \\ & z_C \in \{0,1\} & (C \in \mathcal{S}). \end{array} \right.$$

We denote the Lagrangian relaxation problem of $(P(\mathcal{S}))$ by $(LR(\mathcal{S},\boldsymbol{\lambda}))$. The problem $(LR(\mathcal{S},\boldsymbol{\lambda}))$ is given as

$$(LR(\mathcal{S},\boldsymbol{\lambda})) \quad \left| \begin{array}{lll} \text{maximize} & \displaystyle\sum_{C\in\mathcal{S}} \gamma_C(\boldsymbol{\lambda}) z_C + \sum_{i\in V} \lambda_i & \\ \text{subject to} & z_C \in \{0,1\} & (C \in \mathcal{S}). \end{array} \right.$$

In the same way as in the discussion in Subsection 2.4.1, an optimal solution $\boldsymbol{z}(\mathcal{S},\boldsymbol{\lambda}) = (z_C(\boldsymbol{\lambda}))_{C\in\mathcal{S}}$ of the problem $(LR(\mathcal{S},\boldsymbol{\lambda}))$ can be obtained by

$$z_C(\boldsymbol{\lambda}) = \begin{cases} 1 & \text{when } \gamma_C(\boldsymbol{\lambda}) > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{2.6}$$

for a given $\boldsymbol{\lambda} \in \mathbb{R}^n$. We denote the Lagrangian dual problem corresponding to $(P(\mathcal{S}))$ by $(LD(\mathcal{S}))$:

$$(LD(\mathcal{S})) \quad \left| \begin{array}{ll} \text{minimize} & \omega(LR(\mathcal{S},\boldsymbol{\lambda})) \\ \text{subject to} & \boldsymbol{\lambda} \in \mathbb{R}^n. \end{array} \right.$$

We use the subgradient method to solve the problem $(LD(\mathcal{S}))$.

**Definition 2.11** *$\boldsymbol{g}(\boldsymbol{\lambda}_0) \in \mathbb{R}^n$ is a subgradient of $\omega(LR(\mathcal{S},\boldsymbol{\lambda}))$ at $\boldsymbol{\lambda}_0$ when $\boldsymbol{g}^\top(\boldsymbol{\lambda}_0)(\boldsymbol{\lambda} - \boldsymbol{\lambda}_0) \leq \omega(LR(\mathcal{S},\boldsymbol{\lambda})) - \omega(LR(\mathcal{S},\boldsymbol{\lambda}_0))$ holds for any $\boldsymbol{\lambda} \in \mathbb{R}^n$.*

The subgradient method uses the subgradient $\boldsymbol{g}(\boldsymbol{\lambda})$ at a current multiplier vector $\boldsymbol{\lambda}$, and updates the Lagrangian multiplier vector to the direction of $-\boldsymbol{g}(\boldsymbol{\lambda})$ with a step-size. The following lemma is well known.

**Lemma 2.12** *Let $\boldsymbol{z}(\mathcal{S},\boldsymbol{\lambda}) = (z_C(\boldsymbol{\lambda}))_{C\in\mathcal{S}}$ be an optimal solution of the Lagrangian relaxation problem $(LR(\mathcal{S},\boldsymbol{\lambda}))$. Then $\boldsymbol{g}(\boldsymbol{\lambda}) = (1 - \sum_{C\in\mathcal{S}} a_{iC} z_C(\boldsymbol{\lambda}))_{i\in V}$ is a subgradient of $\omega(LR(\mathcal{S},\boldsymbol{\lambda}))$ at $\boldsymbol{\lambda} \in \mathbb{R}^n$.*

Various methods to determine the step-size and their convergence properties have been discussed. See, for example [40, 53]. We use the following rule to update the current Lagrangian multiplier vector $\boldsymbol{\lambda}$ to the next multiplier vector $\hat{\boldsymbol{\lambda}}$.

$$\hat{\lambda}_i = \lambda_i - \mu \frac{\omega(LR(\mathcal{S}, \boldsymbol{\lambda})) - \mathrm{LB}}{\|\boldsymbol{g}(\boldsymbol{\lambda})\|^2} g_i(\boldsymbol{\lambda}) \quad \text{for all } i \in V, \tag{2.7}$$

where $\mu$ is a step-size control parameter and LB is a lower bound of $\omega(P)$. When the multiplier vector is updated, the value $\omega(LR(\mathcal{S}, \boldsymbol{\lambda}))$ does not necessarily decrease. We count the number of consecutive failures to decrease the value, and when it reaches a predetermined number, we halve the step-size control parameter. When the step-size control parameter falls below $0.005$, we stop the subgradient algorithm. The subgradient method to solve the problem $(LD(\mathcal{S}))$ is described as follows:

**Procedure Sub-Gradient (SG($\boldsymbol{\lambda}$, LB))**

**Input** : a current multiplier $\boldsymbol{\lambda}$ and a lower bound LB

**Output** : a near optimal multiplier $\boldsymbol{\lambda}$, the objective value $\omega_{\text{up}}$, and a lower bound LB

Step 1 : Initialize a parameter $\mu \leftarrow 2.0$, and set a counter $k \leftarrow 0$, $\hat{\boldsymbol{\lambda}} \leftarrow \boldsymbol{\lambda}$ and $\omega_{\text{up}} \leftarrow +\infty$.

Step 2 : Solve the problem $(LR(\mathcal{S}, \hat{\boldsymbol{\lambda}}))$ by setting $z_C(\hat{\boldsymbol{\lambda}})$ according to (2.6).
If $\omega(LR(\mathcal{S}, \hat{\boldsymbol{\lambda}})) < \omega_{\text{up}}$, then update $\omega_{\text{up}}$ to $\omega(LR(\mathcal{S}, \hat{\boldsymbol{\lambda}}))$, and set $k \leftarrow 0$. Otherwise $k \leftarrow k + 1$.

Step 3 : If $(\omega_{\text{up}} - \mathrm{LB})/\omega_{\text{up}} < \epsilon$, then terminate.

Step 4 : Compute the subgradient $\boldsymbol{g}(\boldsymbol{\lambda})$ from $\boldsymbol{z}(\mathcal{S}, \boldsymbol{\lambda})$.
If $\boldsymbol{g}(\boldsymbol{\lambda}) = \mathbf{0}$, then set LB $\leftarrow \omega_{\text{up}}$, and terminate.

Step 5 : If $\mu \leq 0.005$, then terminate. If $k \geq 30$, set $\mu \leftarrow \mu/2$.

Step 6 : Update $\boldsymbol{\lambda}$ to $\hat{\boldsymbol{\lambda}}$ according to (2.7), and go to Step 2.

Note that the solution obtained by this procedure is not necessarily an optimal solution of $(LD(\mathcal{S}))$. Now we denote the obtained objective value of $(LD(\mathcal{S}))$ by $\bar{\omega}(LD(\mathcal{S}))$.

Since the variables $z_C$ for $C \in \mathcal{P} \setminus \mathcal{S}$ are not considered in the problem $(LR(\mathcal{S}, \boldsymbol{\lambda}))$, the optimal solution $z(\mathcal{S}, \boldsymbol{\lambda})$ is not necessarily optimal to $(LR(\mathcal{P}, \boldsymbol{\lambda}))$. Therefore there is

no guarantee that $\omega(LR(\mathcal{S}, \boldsymbol{\lambda}))$ as well as $\bar{\omega}(LD(\mathcal{S}))$ reaches an upper bound of $\omega(P)$. When $\gamma_C(\boldsymbol{\lambda}) \leq 0$ for all $C \in \mathcal{P} \setminus \mathcal{S}$, $z(\mathcal{S}, \boldsymbol{\lambda})$ is an optimal solution of $(LR(\mathcal{S}, \boldsymbol{\lambda}))$ due to the setting by means of (2.6), thus $\omega(LR(\mathcal{S}, \boldsymbol{\lambda}))$ and $\bar{\omega}(LD(\mathcal{S}))$ are upper bounds of $\omega(P)$. On the other hand, if there exists $C \in \mathcal{P} \setminus \mathcal{S}$ which satisfies

$$\gamma_C(\boldsymbol{\lambda}) > 0,$$

adding this $C$ to $\mathcal{S}$ can lead to an improvement of the optimal value of $(LR(\mathcal{S}, \boldsymbol{\lambda}))$, i.e., $\omega(LR(\mathcal{S}', \boldsymbol{\lambda})) > \omega(LR(\mathcal{S}, \boldsymbol{\lambda}))$ where $\mathcal{S}' = \mathcal{S} \cup \{C\}$. It should be noted that $\boldsymbol{\lambda}$ is an appropriate solution for the problem $(LD(\mathcal{S}))$ but no longer for $(LD(\mathcal{S}'))$. Then we solve the problem $(LD(\mathcal{S}'))$ again to obtain a near optimal Lagrangian multiplier by the subgradient method.

The problem of finding a community that maximizes $\gamma_C(\boldsymbol{\lambda})$ ends up as the problem $(AP(\boldsymbol{\lambda}))$ with a quadratic concave function. In the same manner as in the algorithm based on the LP relaxation, finding $\boldsymbol{y}^*$ with positive optimal value, we add the variable $z_C$ corresponding to the community $C = \{\, i \in V \mid y_i^* = 1 \,\}$ to the problem $(LR(\mathcal{S}, \boldsymbol{\lambda}))$. From the above discussion, a prototype of the column generation algorithm based on the Lagrangian relaxation is described as follows.

**Algorithm Prototype of Lagrangian-based-Column-Generation**

Step 1 : Let $\mathcal{S}$ and $\boldsymbol{\lambda}$ be an initial family of nonempty subsets of $V$ and an initial multiplier vector, respectively. Initialize a lower bound by setting LB $\leftarrow 0$.

Step 2 : Call the procedure SG($\boldsymbol{\lambda}$, LB) to solve $(LD(\mathcal{S}))$.
Set $\boldsymbol{\lambda}$ and $\bar{\omega}(LD(\mathcal{S}))$ be a solution and the objective value of $(LD(\mathcal{S}))$ returned by the procedure.

Step 3 : Solve $(AP(\boldsymbol{\lambda}))$ to obtain an optimal solution $\boldsymbol{y}^*$.

Step 4 : If $\omega(AP(\boldsymbol{\lambda})) \leq 0$, then set $\mathcal{S}^* \leftarrow \mathcal{S}$ and $\omega^* \leftarrow \bar{\omega}(LD(\mathcal{S}))$. Output $\mathcal{S}^*$ and $\omega^*$, and terminate.
Otherwise set $C^* \leftarrow \{i \in V \mid y_i^* = 1\}$ and increment $\mathcal{S} \leftarrow \mathcal{S} \cup \{C^*\}$. Return to Step 2.

Similarly to the algorithms based on the LP relaxation in the previous section, we construct the problem $(P(\mathcal{S}^*))$ from obtained $\mathcal{S}^*$. The optimal value $\omega(P(\mathcal{S}^*))$ provides a lower bound of $\omega(P)$. Furthermore we obtain an upper bound $\bar{\omega}(LD(\mathcal{S}^*))$ of $\omega(P)$ when $\omega(AP(\boldsymbol{\lambda})) \leq 0$ is satisfied. Therefore we have the following inequality:

$$\omega(P(\mathcal{S}^*)) \leq \omega(P) \leq \bar{\omega}(LD(\mathcal{S}^*)).$$

The value $\bar{\omega}(LD(\mathcal{S}^*)) - \omega(P(\mathcal{S}^*))$ provides an upper bound of the difference between $\omega(P)$ and $\omega(P(\mathcal{S}^*))$.

### 2.4.3 Bounding procedures and stopping criteria

The column generation algorithm often suffers from slow convergence. In this algorithm, the objective value of $(LD(\mathcal{S}))$ increases slowly as the solution $\boldsymbol{\lambda}$ is close to the optimal solution, namely long-tail convergence process is observed in latter iteration. This slow convergence process is called *tailing-off effect* [32, 65]. When the difference between an upper bound and $\omega(LD(\mathcal{S}))$ is small, we stop the algorithm even if $\omega(AP(\boldsymbol{\lambda})) \leq 0$ does not hold since the difference between the upper bound and $\omega(LD(\mathcal{S}))$ provides the quality of the current objective value of $(LD(\mathcal{S}))$. In addition, the most time-consuming step in the algorithm is the step of solving the problem $(AP(\boldsymbol{\lambda}))$, hence decreasing the number of iterations of the algorithm could considerably contribute to reduce the total computing time. We put off the description of the bounding procedures until the last in this subsection, and first present the stopping criteria. Now we consider the following stopping criterion

$$\frac{\text{UB} - \omega(LD(\mathcal{S}))}{\text{UB}} \leq \varepsilon,$$

where UB is the best upper bound obtained so far. Namely, we stop the algorithm when $\omega(LD(\mathcal{S}))$ is sufficiently close to an upper bound of $\omega(P)$. Note that the objective value $\bar{\omega}(LD(\mathcal{S}))$ we obtain at each iteration is not necessarily optimal for $(LD(\mathcal{S}))$. In other words, the closeness of UB and $\bar{\omega}(LD(\mathcal{S}))$ does not imply the closeness of UB and $\omega(LD(\mathcal{S}))$. To resolve this invalidity, we propose two stopping criteria.

The first one uses a lower bound LB of $\omega(P)$ . When the difference of between UB and LB is small, we can claim that this lower bound fully approximates the optimal value of $\omega(P)$. The second one uses the optimal value of $(LD(\mathcal{S}))$. As we have seen in Subsection 2.4.1, the optimal value of the Lagrangian dual problem equals to that

of the LP relaxation problem. We exploit this property for the purpose of computing $\omega(LD(\mathcal{S}))$. Denoting the LP relaxation problem and its dual problem corresponding to $(P(\mathcal{S}))$ by $(RP(\mathcal{S}))$ and $(RD(\mathcal{S}))$, we have $\omega(LD(\mathcal{S})) = \omega(RD(\mathcal{S}))$. The optimal value of $(RD(\mathcal{S}))$ provides the optimal value of $(LD(\mathcal{S}))$. Therefore we solve the problem $(RD(\mathcal{S}))$ to obtain $\omega(RD(\mathcal{S}))$ only if $(\mathrm{UB} - \bar{\omega}(LD(\mathcal{S})))/\mathrm{UB} \leq \varepsilon$, and check whether the following condition holds.

$$\frac{\mathrm{UB} - \omega(RD(\mathcal{S}))}{\mathrm{UB}} \leq \varepsilon.$$

To obtain an upper bound of $\omega(P)$, we also make use of Proposition 2.9 thanks to the arbitrariness of $\boldsymbol{\lambda}$ in this proposition. Next, we describe the Lagrangian heuristics to obtain a feasible solution of the problem $(P)$, which is based on a simple greedy algorithm. For given a family $\mathcal{S}$ and a multiplier vector $\boldsymbol{\lambda}$, this algorithm iteratively select a community $C \in \mathcal{S}$ with the maximum value of $\gamma_C(\boldsymbol{\lambda})$ as a member of a partition. Note that we have to select a community which is disjoint from the previously selected communities at each iteration. Then removing the nodes in the previously selected communities from the node set $V$, we update $\mathcal{S}$ to the family whose element consists of a subset of the remaining nodes. From the above discussion, the greedy heuristics to obtain a lower bound of $\omega(P)$ is given as follows.

**Procedure Greedy-Heuristics (GH($\mathcal{S}, \boldsymbol{\lambda}$))**

**Input** : a current family $\mathcal{S}$, and a current multiplier $\boldsymbol{\lambda}$

**Output** : a lower bound $\ell(P)$

Step 1 : Let $\mathcal{S}$ and $\boldsymbol{\lambda}$ be a current subfamily and a Lagrangian multiplier vector, respectively. Set $\ell(P) \leftarrow 0$.

Step 2 : $C^* \leftarrow \mathrm{argmax}\,\{\gamma_C(\boldsymbol{\lambda}) \mid C \in \mathcal{S}\}$, and set $\ell(P) \leftarrow \ell(P) + f_{C^*}$.

Step 3 : Update $V' \leftarrow V \setminus C^*$, and $S' \leftarrow \{C \in \mathcal{S} \mid C \subseteq V'\}$.

Step 4 : If $V' = \emptyset$, then terminate.
Otherwise set $V \leftarrow V'$ and $S \leftarrow S'$. Go to Step 2.

### 2.4.4 Pegging test

In this section we consider the problem reduction technique for the problem $(P(\mathcal{S}^*))$, i.e., fixing some variables without loss of the optimality of the solution. This problem reduction technique is called *pegging test*, which makes use of a lower bound and $\omega(LR(\mathcal{S}^*, \boldsymbol{\lambda}))$.

**Proposition 2.13** *Let LB be a lower bound of* $(P(\mathcal{S}^*))$ *and* $\alpha \in \{0, 1\}$. *If* $\omega(LR(\mathcal{S}^*, \boldsymbol{\lambda} \mid z_C = \alpha)) <$ *LB, then* $z_C = 1 - \alpha$ *for any optimal solution of the problem* $(P(\mathcal{S}^*))$.

**Proof :** Since the problem $(LR(\mathcal{S}^*, \boldsymbol{\lambda}))$ is a relaxation problem of $(P(\mathcal{S}^*))$, it is clear that

$$\omega(P(\mathcal{S}^* \mid z_C = \alpha)) \leq \omega(LR(\mathcal{S}^*, \boldsymbol{\lambda} \mid z_C = \alpha)). \tag{2.8}$$

Suppose that $\omega(LR(\mathcal{S}^*, \boldsymbol{\lambda} \mid z_C = \alpha)) <$ LB, we can obtain $\omega(P(\mathcal{S}^* \mid z_C = \alpha)) <$ LB from (2.8). This inequality implies the non-existence of the optimal solution that satisfies $z_C = \alpha$. Hence we conclude $z_C = 1 - \alpha$ for any optimal solution of $(P(\mathcal{S}^*))$ ■

In addition, suppose that we have an optimal solution $\boldsymbol{z}(\mathcal{S}^*, \boldsymbol{\lambda})$ of $LR(\mathcal{S}^*, \boldsymbol{\lambda})$ and that $z_C(\boldsymbol{\lambda}) = 0$. Since $z_C = 0$ implies $\gamma_C(\boldsymbol{\lambda}) \leq 0$ from (2.6), we can calculate $\omega(LR(\mathcal{S}^*, \boldsymbol{\lambda} \mid z_C = 1))$ as follows:

$$\omega(LR(\mathcal{S}^*, \boldsymbol{\lambda} \mid z_C = 1)) = \omega(LR(\mathcal{S}^*, \boldsymbol{\lambda})) + \gamma_C(\boldsymbol{\lambda}).$$

In the same way, when $z_C = 1$, we can calculate $\omega(LR(\mathcal{S}^*, \boldsymbol{\lambda} \mid z_C = 0))$ as follows:

$$\omega(LR(\mathcal{S}^*, \boldsymbol{\lambda} \mid z_C = 0)) = \omega(LR(\mathcal{S}^*, \boldsymbol{\lambda})) - \gamma_C(\boldsymbol{\lambda})$$

since $z_C = 1$ implies $\gamma_C(\boldsymbol{\lambda}) > 0$. Thus we can fix some variables by simple calculation.

### 2.4.5 Convergence issue

Now we discuss the convergence property of the column generation algorithm based on the Lagrangian relaxation. If the Lagrangian multiplier vector $\boldsymbol{\lambda}$ obtained by the subgradient method satisfies the dual feasibility condition $f_C - \sum_{i \in V} \lambda_i a_{iC} \leq 0$ for any $C \in \mathcal{S}$ for the problem $(RD(\mathcal{S}))$, then the community constructed by an optimal solution $\boldsymbol{y}^*$ is not in $\mathcal{S}$. However the obtained multiplier $\boldsymbol{\lambda}$ does not necessarily satisfy the dual feasibility,

hence we cannot conclude that the obtained community is not in $\mathcal{S}$, i.e., a community that is not considered yet. Thus there is no guarantee that our column generation algorithm converges since the algorithm may repeatedly generate a community which is already in $\mathcal{S}$. To resolve this issue, we incorporate the multiplier adjustment procedure based on Balas and Carrera [6] into our algorithm, and propose the following procedure.

**Procedure Multiplier-Adjustment (MA($\mathcal{S}, \boldsymbol{\lambda}$))**

**Input** : a current family $\mathcal{S}$ and a current multiplier $\boldsymbol{\lambda}$

**Output** : a dual feasible multiplier $\boldsymbol{\lambda}$

Step 1 : If there exists a subset $C' \in \mathcal{S}$ such that $\gamma_{C'}(\boldsymbol{\lambda}) > 0$, then Go to Step 2.
Otherwise terminate.

Step 2 : Choose $i \in C'$, and set $\lambda_i \leftarrow \lambda_i + \gamma_{C'}(\boldsymbol{\lambda})$. Go to Step 1.

---

The procedure MA produces a dual feasible multiplier $\boldsymbol{\lambda}$ without increasing the associated objective value $\omega(LR(\mathcal{S}, \boldsymbol{\lambda}))$. Namely, the following lemma holds.

**Lemma 2.14** *For any $\mathcal{S}$ and $\bar{\boldsymbol{\lambda}} \in \mathbb{R}^n$, let $\boldsymbol{\lambda}$ be a multiplier vector returned by the procedure MA($\mathcal{S}, \bar{\boldsymbol{\lambda}}$). Then $\gamma_C(\boldsymbol{\lambda}) \leq 0$ for any $C \in \mathcal{S}$, and $\omega(LR(\mathcal{S}, \boldsymbol{\lambda})) \leq \omega(LR(\mathcal{S}, \bar{\boldsymbol{\lambda}}))$ holds.*

**Proof :** Suppose that $\gamma_{C'}(\bar{\boldsymbol{\lambda}}) > 0$ for some $C'$ and let $i^* \in C'$ be the node chosen at Step 2 of the procedure. First, we show the returned multiplier vector satisfies the dual feasibility. The multiplier vector $\boldsymbol{\lambda} = (\lambda_i)_{i \in V}$ at the end of Step 2 is given as

$$\lambda_i = \begin{cases} \bar{\lambda}_i & (i \neq i^*) \\ \bar{\lambda}_{i^*} + \gamma_{C'}(\bar{\boldsymbol{\lambda}}) & (i = i^*), \end{cases} \tag{2.9}$$

then we have the following equality

$$\begin{aligned}
\gamma_{C'}(\boldsymbol{\lambda}) &= f_{C'} - \sum_{i \in V} a_{iC}\lambda_i \\
&= f_{C'} - \sum_{i \in V \setminus \{i^*\}} a_{iC'}\lambda_i - a_{i^*C'}\lambda_{i^*} \\
&= f_{C'} - \sum_{i \in V \setminus \{i^*\}} a_{iC'}\bar{\lambda}_i - (\bar{\lambda}_{i^*} + \gamma_{C'}(\bar{\boldsymbol{\lambda}})) \\
&= f_{C'} - \sum_{i \in V} a_{iC'}\bar{\lambda}_i - \gamma_{C'}(\bar{\boldsymbol{\lambda}}) \\
&= 0
\end{aligned}$$

due to the construction of $\boldsymbol{\lambda}$ and $a_{i^*C'} = 1$. Moreover $\boldsymbol{\lambda} \geq \bar{\boldsymbol{\lambda}}$ holds from the construction of $\boldsymbol{\lambda}$, which and the non-negativity of $a_{iC}$ imply that $\gamma_C(\bar{\boldsymbol{\lambda}}) \geq \gamma_C(\boldsymbol{\lambda})$ for any $C \in \mathcal{S}$. Therefore the multiplier vector ends up being dual feasible for $(RD(\mathcal{S}))$ after at most $|\mathcal{S}|$ iterations. Next, we show the multiplier vector does not increases the associated objective value. The optimal values of the Lagrangian relaxation problems at each multiplier vector $\boldsymbol{\lambda}$ and $\bar{\boldsymbol{\lambda}}$ are given by (2.9) as follows:

$$\begin{aligned}
\omega(LR(\mathcal{S}, \boldsymbol{\lambda})) &= \sum_{C \in \mathcal{S}(\boldsymbol{\lambda})^+} \gamma_C(\boldsymbol{\lambda}) + \sum_{i \in V} \bar{\lambda}_i + \gamma_{C'}(\bar{\boldsymbol{\lambda}}), \\
\omega(LR(\mathcal{S}, \bar{\boldsymbol{\lambda}})) &= \sum_{C \in \mathcal{S}(\bar{\boldsymbol{\lambda}})^+ \setminus \{C'\}} \gamma_C(\bar{\lambda}) + \gamma_{C'}(\bar{\boldsymbol{\lambda}}) + \sum_{i \in V} \bar{\lambda}_i,
\end{aligned}$$

where $\mathcal{S}(\boldsymbol{\lambda})^+ = \{\, C \in \mathcal{S} \mid \gamma_C(\boldsymbol{\lambda}) > 0 \,\}$. Hence, we have

$$\omega(LR(\mathcal{S}, \boldsymbol{\lambda})) - \omega(LR(\mathcal{S}, \bar{\boldsymbol{\lambda}})) = \sum_{C \in \mathcal{S}(\boldsymbol{\lambda})^+} \gamma_C(\boldsymbol{\lambda}) - \sum_{C \in \mathcal{S}(\bar{\boldsymbol{\lambda}})^+ \setminus \{C'\}} \gamma_C(\bar{\boldsymbol{\lambda}}). \tag{2.10}$$

Obviously, the inclusion relation $\mathcal{S}(\boldsymbol{\lambda})^+ \subseteq \mathcal{S}(\bar{\boldsymbol{\lambda}})^+$ holds from the definition of $\mathcal{S}(\boldsymbol{\lambda})^+$ and $\gamma_C(\boldsymbol{\lambda}) \leq \gamma_C(\bar{\boldsymbol{\lambda}})$ for any $C \in \mathcal{S}$. Thus this and $C' \notin \mathcal{S}(\boldsymbol{\lambda})^+$ imply that (2.10) is non-positive. ■

The convergence of the algorithm directly follows Lemma 2.14.

**Theorem 2.15** *The Lagrangian relaxation-based algorithm with the procedure MA terminates within a finite number of iterations.*

### 2.4.6 Whole algorithm

The algorithm based on the Lagrangian relaxation is described as follows. We will call this algorithm the *Lagrangian-based-Single-Column-Generation-at-a-Time algorithm*, LSCG for short.

**Algorithm Lagrangian-based-Single-Column-Generation-at-a-Time (LSCG)**

Step 1 : Determine a tolerance parameter $\varepsilon$, and set UB $\leftarrow 1$ and LB $\leftarrow 0$.
Let $\mathcal{S}$ and $\boldsymbol{\lambda}$ be an initial family of nonempty subsets of $V$ and an initial multiplier vector, respectively.

Step 2 : Call the procedure SG($\boldsymbol{\lambda}$, LB) to solve the problem $(LD(\mathcal{S}))$. Let $\bar{\boldsymbol{\lambda}}$ and $\bar{\omega}(LD(\mathcal{S}))$ be a solution and its objective value returned by the procedure.
Compute a lower bound $\ell(P)$ of $\omega(P)$ by the procedure GH($\mathcal{S}, \bar{\boldsymbol{\lambda}}$).
If LB $< \ell(P)$, then set LB $\leftarrow \ell(P)$.

Step 3 : Call the procedure MA($\mathcal{S}, \bar{\boldsymbol{\lambda}}$) in order to convert $\bar{\boldsymbol{\lambda}}$ to dual feasible solution $\boldsymbol{\lambda}$.
Solve $(AP(\boldsymbol{\lambda}))$ and set $\boldsymbol{y}^*$ be an optimal solution. Compute an upper bound $u(P)$ according to Proposition 2.9. If UB $> u(P)$, then UB $\leftarrow u(P)$.

Step 4 : If $\omega(AP(\boldsymbol{\lambda})) \leq 0$ or (UB$-$LB)/UB $\leq \varepsilon$, then set $\mathcal{S}^\star \leftarrow \mathcal{S}, \omega^\star \leftarrow \bar{\omega}(LD(\mathcal{S}))$, LB$^\star \leftarrow$ LB and UB$^\star \leftarrow$ UB. Go to Step 8.

Step 5 : If $(\text{UB} - \bar{\omega}(LD(\mathcal{S})))/\text{UB} \leq \varepsilon$, then solve $(RD(\mathcal{S}))$ to obtain $\omega(RD(\mathcal{S}))$. Otherwise go to Step 7.

Step 6 : If $(\text{UB} - \omega(RD(\mathcal{S})))/\text{UB} \leq \varepsilon$, then set $\mathcal{S}^\star \leftarrow \mathcal{S}, \omega^\star \leftarrow \bar{\omega}(LD(\mathcal{S}))$, LB$^\star \leftarrow$ LB and UB$^\star \leftarrow$ UB. Go to Step 8.

Step 7 : Set $C^* \leftarrow \{\, i \in V \mid y_i^* = 1 \,\}$ and increment $\mathcal{S} \leftarrow \mathcal{S} \cup \{C^*\}$. Return to Step 2.

Step 8 : Execute the pegging test according to Proposition 2.13. Solve $(P(\mathcal{S}^\star))$ by an IP solver. Output $\mathcal{S}^\star$, LB$^\star$, UB$^\star$ and $\omega^\star$, and terminate.

---

The algorithm consists of two phases: the first phase is to collect a set of variables that are likely to be positive at an optimal solution and to obtain lower and upper bounds

of $\omega(P)$ (Step 1 to Step 7 in LSCG), the second phase is to solve the problem $(P(\mathcal{S}^\star))$ (Step 8 in LSCG). Similarly to the LP relaxation algorithm, we extend the algorithm based on the Lagrangian relaxation to an algorithm of adding multiple columns which may complement well each other, and call the algorithm the *Lagrangian-based-Multiple-Column-Generation-at-a-Time algorithm*, LMCG for short.

### 2.4.7 Computational experiments

In this subsection, we report the computational experiments with algorithms LSCG and LMCG. The experiment was performed on a PC with Intel Core i7, 3.70 GHz processor and 32.0 GB of memory. Using Gurobi 6.0.0 as an IP solver to solve the auxiliary problem, we implemented the algorithms in Python 2.7. We solved the same instances introduced in the Subsection 2.3.6. Throughout the experiments, we set an initial family $\mathcal{S}$ to the family of all singletons, i.e., $\mathcal{S} = \{\,\{1\}, \ldots, \{n\}\,\}$, and an initial multiplier vector $\boldsymbol{\lambda}$ to $\mathbf{0}$, respectively.

First, we conduct the experiment with a tolerance parameter $\varepsilon = 0$, and Table 2.6 shows the results of both algorithms for each instance. The collected statistics in this table are given as follows:

- iteration : number of solving the problem $(LD(\mathcal{S}))$
- $|\mathcal{S}^\star|$ : cardinality of the final family of subsets $\mathcal{S}^\star$
- $\omega^\star$ : objective value of $(LD(\mathcal{S}^\star))$ obtained at the end of the first phase
- LB$^\star$ : lower bound at the end of the first phase
- UB$^\star$ : upper bound at the end of the first phase
- $\omega(P(\mathcal{S}^\star))$ : optimal value of $(P(\mathcal{S}^\star))$
- gap (%) : relative gap defined by gap $= \left(\frac{\omega(P)-\omega(P(\mathcal{S}^\star))}{\omega(P)}\right) \times 100$
- time 1 : computation time of the first phase in seconds
- time 2 : computation time of the second phase in seconds
- pegged : number of pegged variables

Table 2.6: Computational results of LSCG and LMCG ($\varepsilon = 0$)

| ID | iteration | | $\lvert\mathcal{S}^\star\rvert$ | | $\omega^\star$ | | LB$^\star$ | | UB$^\star$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG |
| 1 | 27 | 13 | 61 | 61 | 0.41979 | 0.41979 | 0.41979 | 0.41979 | 0.41979 | 0.41979 |
| 2 | 63 | 22 | 125 | 121 | 0.52852 | 0.52852 | 0.52852 | 0.52852 | 0.52852 | 0.52852 |
| 3 | 67 | 19 | 144 | 124 | 0.56001 | 0.56001 | 0.56001 | 0.56001 | 0.56001 | 0.56001 |
| 4 | 109 | 29 | 214 | 187 | 0.52724 | 0.52724 | 0.52724 | 0.52724 | 0.52724 | 0.52724 |
| 5 | 63 | 13 | 178 | 160 | 0.60457 | 0.60457 | 0.60457 | 0.60457 | 0.60457 | 0.60457 |
| 6 | 1150 | 123 | 1662 | 1299 | 0.81937 | 0.81937 | 0.81936 | 0.81936 | 0.81937 | 0.81937 |
| ID | $\omega(P(\mathcal{S}^\star))$ | | gap (%) | | time 1 (sec.) | | time 2 (sec.) | | pegged | |
| | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG |
| 1 | 0.41979 | 0.41979 | 0.000 | 0.000 | 3.721 | 2.678 | * | * | * | * |
| 2 | 0.52852 | 0.52852 | 0.000 | 0.000 | 32.736 | 12.464 | * | * | * | * |
| 3 | 0.56001 | 0.56001 | 0.000 | 0.000 | 50.389 | 15.285 | * | * | * | * |
| 4 | 0.52724 | 0.52724 | 0.000 | 0.000 | 170.039 | 51.655 | * | * | * | * |
| 5 | 0.60457 | 0.60457 | 0.000 | 0.000 | 470.289 | 178.593 | * | * | * | * |
| 6 | 0.81936 | 0.81936 | 0.005 | 0.005 | 36727.123 | 4844.363 | 5.149 | 3.308 | 1622 | 1273 |

The symbol "*" in the columns "time 2" and "pegged" represents that the second phase is not executed since the difference between the obtained $LB^\star$ and $UB^\star$ vanishes at the end of the first phase.

Similarly to the results of the algorithms based on LP relaxation, we observe that the number of generated variables is much smaller than that of the original problem in both LSCG and LMCG from Table 2.6. To compare LSCG with LMCG, we see that LMCG performs better than LSCG in terms of the number of iterations and the computation time for the first phase. Especially, for the instance s838 (ID=6), the number of iterations (resp., the computation time) is reduced by a factor of approximately 9.34 (resp., 7.58). On the other hand, regarding the accuracy of the solution, LSCG provides the lower and upper bounds as good as LMCG does, indeed, both algorithms solve the instances, except for the instance s838 (ID=6), to optimality. The algorithms fail to solve the instance s838, but the remaining gap is less than 0.005%.

To compere the performance of several existing heuristics and algorithms LSCG and LMCG, we give the lower bounds on the modularity obtained by the same existing heuristics [20, 26, 46, 51, 77, 79] used in Subsection 2.3.6 and our algorithms in Table 2.7. We confirm that LSCG and LMCG outperform other heuristic algorithms in terms of accuracy from the table.

Table 2.7: Lower bounds by several existing heuristics, LSCG and LMCG

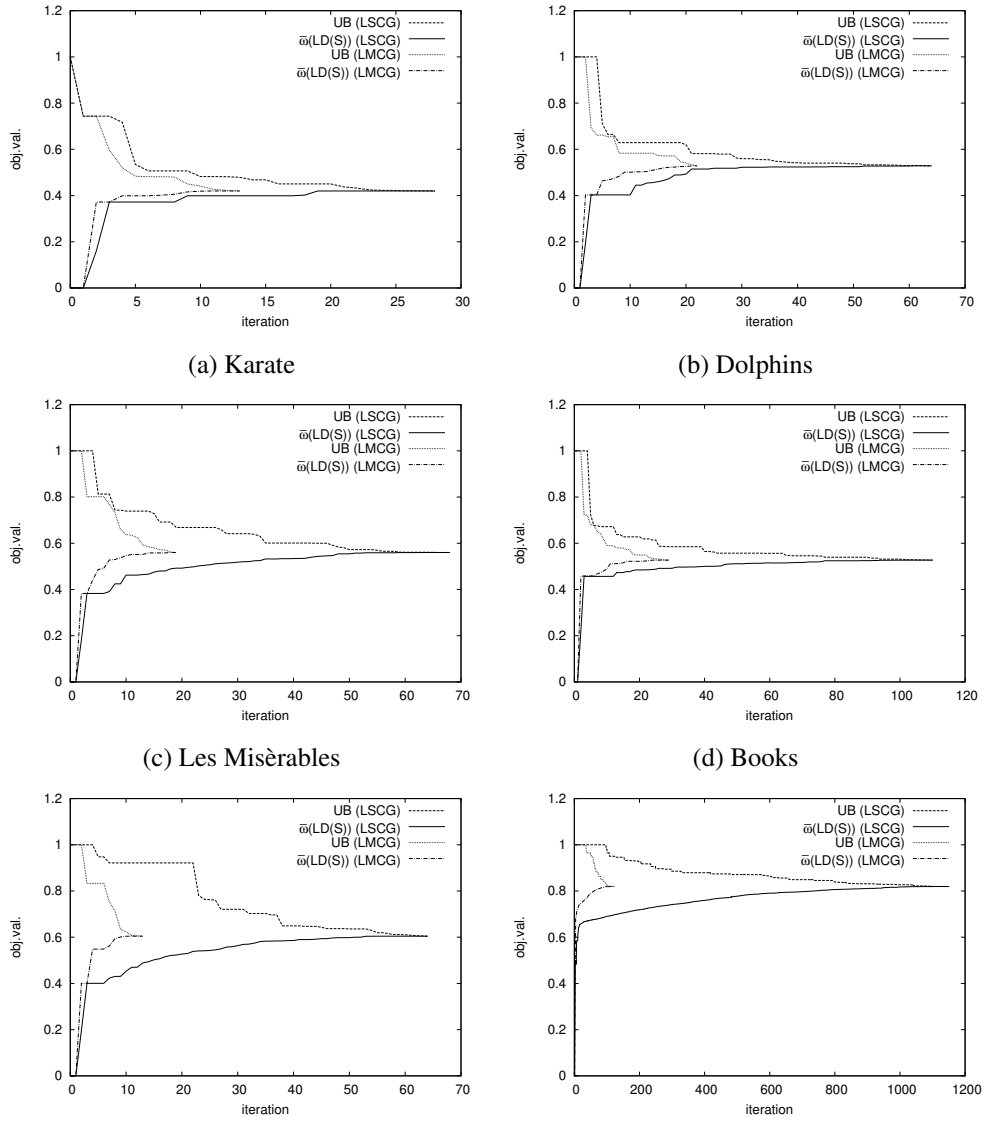
| ID | GN | CNM | SA | SD | CHL | NR | LSCG | LMCG |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.401 | 0.38067 | 0.420 | 0.39341 | 0.41880 | NA | 0.41979 | 0.41979 |
| 2 | 0.520 | 0.49549 | 0.527 | 0.49120 | 0.52646 | 0.52377 | 0.52852 | 0.52852 |
| 3 | 0.540 | 0.50060 | 0.556 | 0.51383 | 0.54676 | 0.56001 | 0.56001 | 0.56001 |
| 4 | NA | 0.50197 | 0.527 | 0.46718 | 0.52629 | 0.52694 | 0.52724 | 0.52724 |
| 5 | 0.601 | 0.57728 | 0.604 | 0.49261 | 0.60091 | 0.60028 | 0.60457 | 0.60457 |
| 6 | NA | 0.80556 | NA | 0.73392 | 0.81663 | 0.81624 | 0.81936 | 0.81936 |

Next, we conduct experiment with a tolerance parameter $\varepsilon = 0.03$, and the computational results of the algorithms LSCG and LMCG are given in Table 2.8. From the table, we observe that the number of iterations and the computation time for the first phase in this experiment are smaller than those in the experiment with $\varepsilon = 0$ for all instances, and

Table 2.8: Computational results of LSCG and LMCG ($\varepsilon = 0.03$)

| ID | iteration | | $\|\mathcal{S}^\star\|$ | | $\omega^\star$ | | LB$^\star$ | | UB$^\star$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG |
| 1 | 22 | 11 | 56 | 59 | 0.41979 | 0.41979 | 0.41979 | 0.41979 | 0.43208 | 0.42710 |
| 2 | 46 | 20 | 108 | 117 | 0.52487 | 0.52541 | 0.51820 | 0.52207 | 0.54031 | 0.54095 |
| 3 | 51 | 16 | 128 | 121 | 0.55731 | 0.55827 | 55731 | 0.55827 | 0.57266 | 0.57193 |
| 4 | 77 | 23 | 182 | 178 | 0.52440 | 0.52527 | 0.52118 | 0.52278 | 0.53942 | 0.53774 |
| 5 | 55 | 11 | 170 | 156 | 0.60443 | 0.60443 | 0.60443 | 0.60443 | 0.61951 | 0.60697 |
| 6 | 863 | 93 | 1375 | 1194 | 0.80925 | 0.81750 | 0.75784 | 0.78826 | 0.83175 | 0.84132 |
| ID | $\omega(P(\mathcal{S}^\star))$ | | gap (%) | | time 1 (sec.) | | time 2 (sec.) | | pegged | |
| | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG | LSCG | LMCG |
| 1 | 0.41979 | 0.41979 | 0.000 | 0.000 | 2.926 | 2.286 | 0.014 | 0.015 | 51 | 55 |
| 2 | 0.51820 | 0.52207 | 1.953 | 1.220 | 23.744 | 11.408 | 0.329 | 0.214 | 54 | 81 |
| 3 | 0.55731 | 0.55827 | 0.482 | 0.310 | 39.130 | 13.144 | 0.059 | 0.050 | 120 | 115 |
| 4 | 0.52118 | 0.52278 | 1.149 | 0.846 | 115.422 | 40.353 | 0.680 | 0.619 | 122 | 124 |
| 5 | 0.60443 | 0.60443 | 0.023 | 0.023 | 310.270 | 126.620 | 0.186 | 0.138 | 155 | 146 |
| 6 | 0.78317 | 0.81694 | 4.421 | 0.300 | 25698.797 | 3236.463 | 170.764 | 143.462 | 7 | 21 |

their reductions are significant as the instance size grows unlike the computational results of the algorithms based on LP relaxation. To be specific, for the instance s838 (ID=6), the computation time of both LSCG and LMCG in the experiment with $\varepsilon = 0.03$ is reduced by a factor approximately 1.5 compared with that in the experiment with $\varepsilon = 0$. Concerning the accuracy of the solution, for the instances except for the instance Karate (ID=1), we see that both algorithms fail to solve to optimality, but the gap is less than 5%.




(a) Karate

(b) Dolphins

(c) Les Misèrables

(d) Books

(e) Football

(f) s838

Figure 2.3: Behaviors of LSCG and LMCG for each instance

Figures 2.3 (a), (b), (b), (d), (e), and (f) show $\bar{\omega}(LD(\mathcal{S}))$ and the upper bound vs. the number of iterations. Similarly to the behaviors of the algorithms based on LP relaxation,

$\bar{\omega}(LD(\mathcal{S}))$ rapidly increases at an early stage, and increases slowly as the algorithm goes on for both LSCG and LMCG.

# Chapter 3

# Modularity Density Maximization

## 3.1 Introduction

Modularity proposed by Newman and Girvan [78] is the most commonly used measure when the nodes of a network are grouped into communities consisting of tightly connected nodes. However, some authors pointed out drawbacks of the modularity, the main issue of which is resolution limit. Resolution limit refers to the sensitivity of the modularity to the total number of edges in the graph, which leaves small communities not identified and hidden inside larger ones. To overcome this drawback, Li *et al.* [63] have proposed a new measure for community detection, which is called modularity density, and the problem of maximizing the modularity density can be straightforwardly formulated as a nonlinear binary programming.

As for the mathematical optimization approaches for the modularity density maximization, Costa [29] has presented some *mixed integer linear programming* formulations, MILP for short, which enables an application of general-purpose solvers, e.g., CPLEX, Gurobi and Xpress, to the problem. However, the number of communities must be fixed in advance, and a difficult auxiliary problem need be solved in their formulations. More recently, a hierarchical divisive heuristics has been proposed by Costa *et al.* [30] to obtain a good lower bound on the modularity density.

In this chapter, for the modularity density maximization, we give a new formulation based on a variant of a semidefinite programming called 0-1SDP. One of the advantages of this formulation is that the size of the problem is independent of the number of edges

of the graph. In order to obtain an upper bound on the modularity density, we propose to relax 0-1SDP to a semidefinite programming problem with non-negative constraints. The relaxation problem obtained can be solved in polynomial time, and also does not require the number of communities in contrast to MILP formulations. Moreover, we develop a method based on the combination of spectral heuristics and dynamic programming to construct a feasible solution from the solution obtained by the relaxation problem.

### 3.1.1 Definitions and notation

Let $G = (V, E)$ be an undirected graph with the set $V$ of $n$ nodes and the set $E$ of undirected $m$ edges. We assume that $V$ has at least two nodes. We denote the set of edges that have one end-node in $C$ and the other end-nodes in $C'$ by $E(C, C')$. When $C = C'$, we abbreviate $E(C, C')$ to $E(C)$ for the sake of simplicity. Then modularity density, denoted by $D(\Pi)$, for a partition $\Pi$ is defined as

$$D(\Pi) = \sum_{C \in \Pi} \left( \frac{2|E(C)| - \sum_{C' \in \Pi} |E(C, C')|}{|C|} \right),$$

where $|\cdot|$ denotes the cardinality of the corresponding set. We refer to each term of the summation as the contribution of community to the modularity density.

*Modularity density maximization problem*, MD for short, is to find a partition of $V$ that maximizes the modularity density $D(\Pi)$, then the problem is formulated as

$$\text{(MD)} \quad \left| \begin{array}{ll} \text{maximize} & \displaystyle\sum_{C \in \Pi} \left( \frac{2|E(C)| - \sum_{C' \in \Pi} |E(C, C')|}{|C|} \right) \\ \text{subject to} & \Pi \text{ is a partition of } V. \end{array} \right.$$

A nonlinear binary programming formulation for (MD) have been proposed in Li *et al.* [63]. Although the optimal number of communities for the modularity density maximization is a priori unknown similarly to the modularity maximization problem, we suppose it is known for the time being, and denote it by $t$, and let $T$ be the index set $\{ 1, 2, \ldots, t \}$. Introducing a binary variables $x_{ip}$ indicating whether node $i$ belongs to

community $C_p$, we have the following formulation:

$$\text{(MD)} \quad \left| \begin{array}{lll} \text{maximize} & \displaystyle\sum_{p\in T} \left( \frac{2\sum_{i\in V}\sum_{j\in V} a_{ij}x_{ip}x_{jp} - \sum_{i\in V} d_i x_{ip}}{\sum_{i\in V} x_{ip}} \right) & \\ \text{subject to} & \displaystyle\sum_{p\in T} x_{ip} = 1 & (i \in V) \\ & \displaystyle\sum_{i\in V} x_{ip} \geq 1 & (p \in T) \\ & x_{ip} \in \{\, 0, 1 \,\} & (i \in V, p \in T), \end{array} \right.$$

where $a_{ij}$ is the $(i, j)$ element of the adjacency matrix $A$ of graph $G$, and $d_i$ is the degree of node $i \in V$. The first set of constraints states that each node belongs to exactly one community, and the second set of constraints imposes that each community should be a nonempty subset of $V$. The objective function in this problem is the sum of fractional functions with a quadratic numerator and a linear denominator. One of the widely used solution approaches for the problem of this kind is a parametric algorithm by Dinkelbach [35]. Another approach is a branch-and-bound algorithm [14, 59] in global optimization area.

### 3.1.2 Some properties

In this subsection, we give some properties of the modularity density. Now suppose that there exist several isolated nodes in a graph $G$. After removing the isolated nodes from $G$, we find a partition $\Pi^\star$ that maximizes the modularity density on the reduced graph of $G$. If the contribution of a community is non-negative for any community of $\Pi^\star$, then $\Pi^\star \cup \{\bar{C}\}$ is an optimal partition on the original graph $G$, where $\bar{C}$ consists of the isolated nodes once deleted. If there exist some communities with a negative contribution, then the contribution increases by adding the isolated nodes to these communities since the denominator of the contribution increases. Therefore we have the following lemma.

**Lemma 3.1** (Costa [29], Lemma 1.) *The isolated nodes can be assigned to communities a posteriori.*

Due to Lemma 3.1, we have only to consider graphs with no isolated nodes.

**Proposition 3.2** (Costa [29], Proposition 1, Corollary 1 and Corollary 2.) *Let* $\Pi^\star$ *be a partition with maximum modularity density, then the size of each community is between* 2 *and*

$n - 2(|\Pi^\star| - 1)$, *i.e.,*

$$2 \leq |C| \leq n - 2(|\Pi^\star| - 1) \text{ for any } C \in \Pi^\star.$$

## 3.2 Formulations

In this section, we first present two different reformulations of the modularity density maximization, which are based on MILP formulation according to Costa [29]. Next, we show that the modularity density maximization can be equivalently formulated as 0-1SDP, a variant of the semidefinite programming.

Hereafter, $\mathcal{S}_n, \mathcal{S}_n^+$ and $\mathcal{N}_n$ denote the set of $n \times n$ symmetric matrices, the positive semidefinite cone, and the symmetric non-negative cone, i.e.,

$$\begin{aligned}
\mathcal{S}_n &= \{\, Y \in \mathbb{R}^{n \times n} \mid Y = Y^\top \,\}, \\
\mathcal{S}_n^+ &= \{\, Y \in \mathcal{S}_n \mid \boldsymbol{u}^\top Y \boldsymbol{u} \geq 0 \text{ for all } \boldsymbol{u} \in \mathbb{R}^n \,\}, \\
\mathcal{N}_n &= \{\, Y = (y_{ij})_{i,j \in \{1,2,\ldots,n\}} \in \mathcal{S}_n \mid y_{ij} \geq 0 \text{ for all } i, j \in \{1, 2, \ldots, n\} \,\}.
\end{aligned}$$

For a given vector $\boldsymbol{u}$, Diag$(\boldsymbol{u})$ is the diagonal matrix with $u_i$ as the $i$-th diagonal element, and vec$(U)$ is the vector obtained by stacking columns of a given matrix $U$.

### 3.2.1 MILP formulations

We can rewrite the objective function in the problem (MD) as follows:

$$\sum_{p \in T} \left( \frac{4 \sum_{\{i,j\} \in E} x_{ip} x_{jp} - \sum_{i \in V} d_i x_{ip}}{\sum_{i \in V} x_{ip}} \right), \tag{3.1}$$

due to the definition of adjacency matrix $A = (a_{ij})_{ij \in V}$ and its symmetry. The quadratic terms $x_{ip}x_{jp}$ can be linearized by replacing them with new variable $y_{ijp}$ and adding the following *Fartet inequalities* [41]:

$$y_{ijp} \leq x_{ip}, \quad y_{ijp} \leq x_{jp}, \quad x_{ip} + x_{jp} \leq y_{ijp} + 1 \quad \text{for } p \in T. \tag{3.2}$$

Note that the last inequality in (3.2) is redundant owing to the objective function of maximizing with respect to the variable $y_{ijp}$, hence can be omitted. Next, we introduce a continuous variable $\alpha_p$ defined as:

$$\alpha_p = \frac{4 \sum_{\{i,j\} \in E} y_{ijp} - \sum_{i \in V} d_i x_{ip}}{\sum_{i \in V} x_{ip}}.$$

This equality constraint can be relaxed to

$$\alpha_p \leq \frac{4\sum_{\{i,j\}\in E} y_{ijp} - \sum_{i\in V} d_i x_{ip}}{\sum_{i\in V} x_{ip}}, \tag{3.3}$$

without overlooking an optimal solution due to the objective function. Since the denominator in (3.3) is positive, we can rewrite it as

$$\alpha_p \sum_{i\in V} x_{ip} \leq 4 \sum_{\{i,j\}\in E} y_{ijp} - \sum_{i\in V} d_i x_{ip}.$$

Finally, to linearize the product $\alpha_p x_{ip}$, we then introduce a continuous variable $\gamma_{ip}$ to replace $\alpha_p x_{ip}$ and make use of the following *McCormick inequalities* [67]:

$$\begin{aligned}
&L_\alpha x_{ip} \leq \gamma_{ip} \leq U_\alpha x_{ip} \quad \text{for } i \in V, p \in T, \\
&\alpha_p - U_\alpha(1 - x_{ip}) \leq \gamma_{ip} \leq \alpha_p - L_\alpha(1 - x_{ip}) \quad \text{for } i \in V, p \in T,
\end{aligned}$$

where $L_\alpha$ and $U_\alpha$ are lower and upper bounds of $\alpha_p$, respectively. From the above discussion, MILP formulation is given as

$$(\text{MILP}_1) \quad \left| \begin{array}{lll}
\text{maximize} & \displaystyle\sum_{p\in T} \alpha_p & \\
\text{subject to} & \displaystyle\sum_{p\in T} x_{ip} = 1 & (i \in V) \\
& 2 \leq \displaystyle\sum_{i\in V} x_{ip} \leq n - 2\,(t-1) & (p \in T) \\
& y_{ijp} \leq x_{ip}, \quad y_{ijp} \leq x_{jp} & (\{i,j\} \in E, p \in T) \\
& \displaystyle\sum_{i\in V} \gamma_{ip} \leq 4 \sum_{\{i,j\}\in E} y_{ijp} - \sum_{i\in V} d_i x_{ip} & (p \in T) \\
& L_\alpha x_{ip} \leq \gamma_{ip} \leq U_\alpha x_{ip} & (i \in V, p \in T) \\
& \alpha_p - U_\alpha(1 - x_{ip}) \leq \gamma_{ip} \leq \alpha_p - L_\alpha(1 - x_{ip}) & (i \in V, p \in T) \\
& x_{ip} \in \{0,1\} & (i \in V, p \in T) \\
& y_{ijp} \in \mathbb{R} & (\{i,j\} \in E, p \in T) \\
& L_\alpha \leq \alpha_p \leq U_\alpha & (p \in T) \\
& \gamma_{ip} \in \mathbb{R} & (i \in V, p \in T).
\end{array} \right.$$

From the inequality (3.3) and Proposition 3.2, a valid lower bound on the variables $\alpha_p$ is attained when the corresponding community consists of only two nodes with the largest degree, thus $L_\alpha$ is given as $L_\alpha = -(d_{\max_1} + d_{\max_2})/2$ where $d_{\max_1}$ and $d_{\max_2}$ are the largest and the second largest degrees, respectively. On the other hand, in order to obtain the

upper bound on $\alpha_p$, we need to solve the following auxiliary problem:

$$(\mathrm{AP}_1)\quad \left|\begin{array}{ll} \text{maximize} & \dfrac{4\sum_{\{i,j\}\in E} x_i x_j - \sum_{i\in V} d_i x_i}{\sum_{i\in V} x_i} \\ \text{subject to} & 2 \le \displaystyle\sum_{i\in V} x_i \le n - 2(t-1) \\ & x_i \in \{0,1\} \quad (i \in V). \end{array}\right.$$

Another MILP formulation is obtained by splitting the two terms in the numerator of (3.1), that is, rewriting the objective function as follows:

$$\frac{4\sum_{\{i,j\}\in E} x_{ip}x_{jp}}{\sum_{i\in V} x_{ip}} - \frac{\sum_{i\in V} d_i x_{ip}}{\sum_{i\in V} x_{ip}}.$$

Introducing continuous variables $\alpha_p$ and $\beta_p$ in order to linearize the two terms, we have

$$\alpha_p \le \frac{\sum_{\{i,j\}\in E} x_{ip}x_{jp}}{\sum_{i\in V} x_{ip}}, \quad \beta_p \ge \frac{\sum_{i\in V} d_i x_{ip}}{\sum_{i\in V} x_{ip}}.$$

In the same manner as in ($\mathrm{MILP}_1$), we can reformulate the problem as follows:

$$(\mathrm{MILP}_2)\quad \left|\begin{array}{lll} \text{maximize} & \displaystyle\sum_{p\in T}(4\alpha_p - \beta_p) & \\ \text{subject to} & \displaystyle\sum_{p\in T} x_{ip} = 1 & (i \in V) \\ & 2 \le \displaystyle\sum_{i\in V} x_{ip} \le n - 2\,(t-1) & (p \in T) \\ & y_{ijp} \le x_{ip}, \quad y_{ijp} \le x_{jp} & (\{i,j\} \in E, p \in T) \\ & \displaystyle\sum_{i\in V} \gamma_{ip} \le \sum_{\{i,j\}\in E} y_{ijp} & (p \in T) \\ & \displaystyle\sum_{i\in V} \mu_{ip} \ge \sum_{i\in V} d_i x_{ip} & (p \in T) \\ & L_\alpha x_{ip} \le \gamma_{ip} \le U_\alpha x_{ip} & (i \in V, p \in T) \\ & \alpha_p - U_\alpha(1-x_{ip}) \le \gamma_{ip} \le \alpha_p - L_\alpha(1-x_{ip}) & (i \in V, p \in T) \\ & L_\beta x_{ip} \le \mu_{ip} \le U_\beta x_{ip} & (i \in V, p \in T) \\ & \beta_p - U_\beta(1-x_{ip}) \le \mu_{ip} \le \beta_p - L_\beta(1-x_{ip}) & (i \in V, p \in T) \\ & x_{ip} \in \{0,1\} & (i \in V, p \in T) \\ & y_{ijp} \in \mathbb{R} & (\{i,j\} \in E, p \in T) \\ & L_\alpha \le \alpha_p \le U_\alpha, \quad L_\beta \le \beta_p \le U_\beta & (p \in T) \\ & \gamma_{ip}, \mu_{ip} \in \mathbb{R} & (i \in V, p \in T), \end{array}\right.$$

where $L_\alpha$ ($L_\beta$) and $U_\alpha$ ($U_\beta$) are lower and upper bounds of $\alpha_p$ ($\beta_p$), respectively. As for the upper and lower bounds on the variables $\beta_p$, we can readily obtain $U_\beta = (d_{\max_1} + d_{\max_2})/2$

and $L_\beta = (d_{\min_1} + d_{\min_2})/2$, where $d_{\min_1}$ and $d_{\min_2}$ are the smallest and the second smallest degrees, respectively. From the definition of $\alpha_p$, we can set $L_\alpha$ to zero, while we need to solve the following problem to derive the upper bound $U_\alpha$.

$$(\mathrm{AP}_2) \quad \left| \begin{array}{ll} \text{maximize} & \dfrac{\sum_{\{i,j\}\in E} x_i x_j}{\sum_{i\in V} x_i} \\ \text{subject to} & 2 \le \displaystyle\sum_{i\in V} x_i \le n - 2(t-1) \\ & x_i \in \{0,1\} \quad (i \in V). \end{array} \right.$$

The problem $(\mathrm{AP}_1)$ as well as $(\mathrm{AP}_2)$ is a problem of maximizing a nonlinear objective function with binary variables, thus difficult to optimize globally. Using a nonlinear programming solver SCIP [1], Costa [29] solved the continuous relaxation problems of $(\mathrm{AP}_1)$ and $(\mathrm{AP}_2)$. In our experiment which is done for the purpose of comparing the solutions obtained by $(\mathrm{MILP}_1)$ formulations and 0-1SDP formulation introduced in Subsection 3.2.2, we solve the problem $(\mathrm{AP}_1)$ to optimality by Dinkelbach's parametric algorithm to derive the upper bound $U_\alpha$ on $\alpha_p$.

### 3.2.2 0-1SDP reformulation

Let $X$ denote the $n \times t$ matrix whose elements are the binary variables $x_{ip}$ in the problem (MD), i.e.,

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1t} \\ x_{21} & x_{22} & \cdots & x_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nt} \end{bmatrix}.$$

The $p$-th column $(x_{1p}, x_{2p}, \ldots, x_{np})^\top$ of the matrix represents the incidence vector of the community $C_p$. Then the constraints of (MD) are concisely expressed as follows:

$$X\boldsymbol{e}_t = \boldsymbol{e}_n, \tag{3.4}$$

$$X^\top \boldsymbol{e}_n \ge \boldsymbol{e}_t, \tag{3.5}$$

$$X \in \{0,1\}^{n\times t}, \tag{3.6}$$

where $\boldsymbol{e}_k$ is the $k$-dimensional vector of 1's.

For a matrix $X$ which satisfies (3.4), (3.5) and (3.6), we define the matrix $Z$ as

$$Z = (z_{ij})_{i,j\in V} = X(X^\top X)^{-1} X^\top. \tag{3.7}$$

Note that the inverse of $X^\top X$ exists since the matrix $X^\top X$ is a nonsingular diagonal matrix whose diagonal entry is the number of 1's of each column of $X$ by (3.4), (3.5) and (3.6). Then we can write the objective function in (MD) as $\mathrm{Tr}((2A-D)Z)$ by means of the matrix $Z$, where $D$ is a diagonal matrix whose $i$-th diagonal element is the degree of node $i$, i.e., $D = \mathrm{Diag}(d_1, d_2, \ldots, d_n) \in \mathbb{R}^{n\times n}$. Clearly $Z$ satisfies

$$Z\boldsymbol{e}_n = ZX\boldsymbol{e}_t = X\boldsymbol{e}_t = \boldsymbol{e}_n, \text{ and } \mathrm{Tr}(Z) = t$$

from (3.4) and (3.7). Moreover, it is symmetric and idempotent, i.e., $Z^2 = Z$ due to (3.7), hence $Z$ is a orthogonal projection matrix. Now we consider the following problem, which we call 0-1SDP

$$\text{(0-1SDP)} \quad \left| \begin{array}{ll} \text{maximize} & \mathrm{Tr}((2A-D)Z) \\ \text{subject to} & Z\boldsymbol{e}_n = \boldsymbol{e}_n \\ & \mathrm{Tr}(Z) = t \\ & Z^2 = Z \\ & Z \in \mathcal{N}_n. \end{array} \right.$$

because Peng and Xia stated "we call it 0-1SDP owing to the similarity of the constraint $Z^2 = Z$ to the classical 0-1 requirement in integer programming" in [80].

From the discussion so far, we have seen that we can construct a feasible solution of (0-1SDP) from a feasible solution of (MD). Furthermore, we can also construct a feasible solution $X$ of (MD) satisfying (3.7) from a feasible solution of (0-1SDP).

**Lemma 3.3** *For any feasible solution $Z$ of (0-1SDP), we can construct a feasible solution $X$ for (MD) which satisfies $Z = X(X^\top X)^{-1}X^\top$.*

**Proof :** Let $Z$ be a feasible solution of (0-1SDP). Then it clearly satisfies the positive semi-definiteness due to the idempotence constraint. Then there exists an index $i_1 \in V$ which satisfies $z_{i_1 i_1} = \max\{\, z_{ij} \mid i, j \in V \,\}$, which is positive owing to the non-negativity of $Z$ and the constraint $Z\boldsymbol{e}_n = \boldsymbol{e}_n$. Let us define the index set $\mathcal{I}_1 = \{\, j \in V \mid z_{i_1 j} > 0 \,\}$, then we readily obtain the following equalities

$$\sum_{j\in\mathcal{I}_1} (z_{i_1 j})^2 = z_{i_1 i_1} \text{ and } \sum_{j\in\mathcal{I}_1} z_{i_1 j} = 1,$$

due to the constraints $Z^2 = Z$, $Z\boldsymbol{e}_n = \boldsymbol{e}_n$, and the symmetry of $Z$. Since $z_{i_1 i_1}$ is positive, the first equality reduces to

$$\sum_{j\in\mathcal{I}_1} \left( \frac{z_{i_1 j}}{z_{i_1 i_1}} \right) z_{i_1 j} = 1.$$

By using the second equality, this yields

$$\sum_{j\in\mathcal{I}_1}\left(\frac{z_{i_1j}}{z_{i_1i_1}}\right)z_{i_1j}=\sum_{j\in\mathcal{I}_1}z_{i_1j},$$

which is rewritten as

$$\sum_{j\in\mathcal{I}_1}\left(\frac{z_{i_1j}}{z_{i_1i_1}}-1\right)z_{i_1j}=0.$$

From the non-negativity of $z_{ij}$ and the maximality of $z_{i_1i_1}$, we have $(z_{i_1j}/z_{i_1i_1}-1)=0$, which implies $z_{i_1j}=z_{i_1i_1}$ for any $j\in\mathcal{I}_1$. Then we have

$$z_{i_1i_1}=z_{i_1j}=\frac{1}{|\mathcal{I}_1|}\quad\text{for any } j\in\mathcal{I}_1. \tag{3.8}$$

For an index $j\in\mathcal{I}_1$, we consider the $(i_1,j)$ element, denoted by $Z^2_{i_1j}$, of the matrix $Z^2$, which is given as

$$Z^2_{i_1j}=\sum_{k\in V}z_{i_1k}z_{kj}=\sum_{k\in\mathcal{I}_1}z_{i_1k}z_{kj}=z_{i_1i_1}\left(\sum_{k\in\mathcal{I}_1}z_{kj}\right).$$

Note that the last equality is due to (3.8). The above equality, $Z^2=Z$ and (3.8) yield

$$z_{i_1i_1}=z_{i_1i_1}\left(\sum_{k\in\mathcal{I}_1}z_{kj}\right),$$

which is reduced to

$$1=\sum_{k\in\mathcal{I}_1}z_{kj}.$$

This implies that $z_{jk}=1/|\mathcal{I}_1|$ for any $j,k\in\mathcal{I}_1$ owing to the maximality of $z_{i_1i_1}$ and the constraint $Z\boldsymbol{e}_n=\boldsymbol{e}_n$. Denote the sub-matrix $(z_{ij})_{i,j\in\mathcal{I}_1}$ by $Z_{\mathcal{I}_1}$. By permuting the rows and columns of $Z$ simultaneously, we obtain

$$P^\top ZP=\begin{bmatrix}Z_{\mathcal{I}_1} & O\\ O & Z_{\bar{\mathcal{I}}_1}\end{bmatrix}$$

where $P$ is an appropriate permutation matrix, and $\bar{\mathcal{I}}_1=V\setminus\mathcal{I}_1$. Then it is clear that the sub-matrix $Z_{\bar{\mathcal{I}}_1}$ satisfies the following

$$Z_{\bar{\mathcal{I}}_1}\boldsymbol{e}_{|\bar{\mathcal{I}}_1|}=\boldsymbol{e}_{|\bar{\mathcal{I}}_1|},\mathrm{Tr}(Z_{\bar{\mathcal{I}}_1})=t-1,Z^2_{\bar{\mathcal{I}}_1}=Z_{\bar{\mathcal{I}}_1}\text{ and }Z_{\bar{\mathcal{I}}_1}\in\mathcal{N}_{|\bar{\mathcal{I}}_1|}.$$

Thus repeating the process described above, we can convert $Z$ to a block diagonal matrix as follows

$$P^\top ZP = \begin{bmatrix} Z_{\mathcal{I}_1} & & & \\ & Z_{\mathcal{I}_2} & & \\ & & \ddots & \\ & & & Z_{\mathcal{I}_t} \end{bmatrix},$$

where each block diagonal element $Z_{\mathcal{I}_p}$ is the $|\mathcal{I}_p| \times |\mathcal{I}_p|$ matrix whose elements are all $1/|\mathcal{I}_p|$. Now, we construct a matrix $X = (x_{ip})$ such that

$$x_{ip} = \begin{cases} 1 & \text{when } i \in \mathcal{I}_p, \\ 0 & \text{otherwise,} \end{cases}$$

then $X$ is clearly feasible for the problem (MD) and we can confirm $Z = X(X^\top X)^{-1}X^\top$ by simple calculation. ■

The equivalence between (MD) and (0-1SDP) directly follows the above lemma.

**Theorem 3.4** *Solving the problem (0-1SDP) is equivalent to finding an optimal solution of (MD).*

The size of (0-1SDP) depends on neither the number of edges nor the number of communities. Moreover, we need not solve the auxiliary problem unlike the case of MILP formulations. These features make (0-1SDP) more attractive than MILP formulations. The objective function in (0-1SDP) is linear with respect to the matrix $Z$, but the idempotence constraint makes the problem difficult. We will discuss how to deal with this difficult part in the next section.

## 3.3 Conic Programming Relaxation

As stated in Subsection 3.2.2, what makes (0-1SDP) difficult to solve is the idempotence constraint, which imposes the condition that each eigenvalue of $Z$, denoted by $\lambda_i$, is either $0$ or $1$. Then it would be a natural strategy to relax the constraint to a more tractable constraint. The first choice is to relax the binary constraint to $0 \leq \lambda_i \leq 1$, which is expressed as $Z \in \mathcal{S}_n^+$ and $I - Z \in \mathcal{S}_n^+$, where $I$ is the identity matrix. The latter constraint $I - Z \in \mathcal{S}_n^+$ which represents the upper bound constraint of $\lambda_i$ is redundant owing to the

presence of $Z \in \mathcal{N}_n$ and $Z\boldsymbol{e}_n = \boldsymbol{e}_n$, hence can be omitted. Since the optimal number $t$ is unknown a priori, we further relax the constraint $\mathrm{Tr}(Z) = t$. Then we obtain the following relaxation problem over the *doubly non-negative cone* $\mathcal{N}_n \cap \mathcal{S}_n^+$:

$$\text{(DNN)} \quad \left| \begin{array}{ll} \text{maximize} & \mathrm{Tr}((2A - D)Z) \\ \text{subject to} & Z\boldsymbol{e}_n = \boldsymbol{e}_n \\ & Z \in \mathcal{N}_n \cap \mathcal{S}_n^+. \end{array} \right.$$

The optimization problems over a symmetric cone are solved efficiently, e.g., linear programming, second-order cone programming, and semidefinite programming problems. Indeed, the primal-dual-interior-point method solves the problems in polynomial time. On the other hand, since the doubly non-negative cone is not symmetric, we cannot directly apply the primal-dual-interior-point method to solve the problem (DNN). Representing the doubly non-negative cone as a symmetric cone embedded in a higher dimension, we could apply the primal-dual-interior-point method to the embedded problem which is described as follows:

$$\left| \begin{array}{ll} \text{maximize} & \mathrm{Tr}((2A - D)Z) \\ \text{subject to} & Z\boldsymbol{e}_n = \boldsymbol{e}_n \\ & \begin{bmatrix} Z & O \\ O & \mathrm{Diag}(\mathrm{vec}(Z)) \end{bmatrix} \in \mathcal{S}^+_{n+n^2}. \end{array} \right.$$

Although the above problem can be solved in polynomial time theoretically, we have to solve a quite large optimization problem over the positive semidefinite cone and it is too computationally expensive in practice. Nevertheless it is worthwhile to solve the problem (DNN) due to the fact that the doubly non-negative relaxation often provides significantly tight bound for some combinatorial optimization problems.

Now we introduce valid inequalities for (0-1SDP) in order to strengthen the bound obtained by the relaxation problem. From the proof of Lemma 3.3, any feasible solution $Z$ of (0-1SDP) can be transformed to a block diagonal matrix. It is easy to see that the maximum value in the $i$-th row of $Z$ is located on the $(i, i)$ element for each $i \in V$, hence we have the following result.

**Lemma 3.5** *The following inequalities are valid for (0-1SDP).*

$$z_{ii} \geq z_{ij} \quad \textit{for } i, j \in V.$$

We denote the problem (DNN) with the above valid inequalities added by ($\overline{\text{DNN}}$).

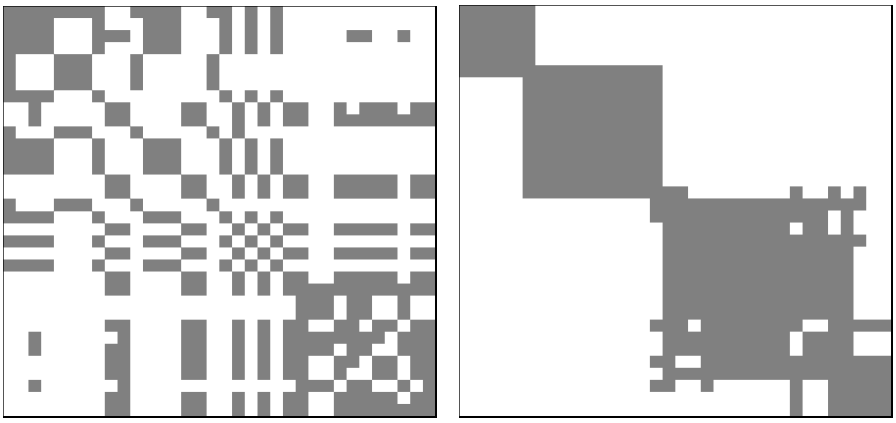
## 3.4 Heuristics based on Dynamic Programming

In this section, we will develop an algorithm to construct a feasible solution for (MD) from the solution obtained by solving the relaxation problem presented in Section 3.3. The algorithm we propose is based on the combination of spectral heuristics and dynamic programming.

### 3.4.1 Permutation based on spectrum

From the proof of Lemma 3.3, we have seen that an optimal solution of (0-1SDP) forms a matrix with block diagonal structure by applying an appropriate simultaneous permutation of the rows and columns, and each block corresponds to a community. Unless otherwise stated, we refer to the simultaneous permutation of the rows and columns simply as a permutation. The optimal solution of the problem (DNN) or ($\overline{\text{DNN}}$) is not necessarily transformed to a block diagonal matrix by any permutation since we relaxed some constraints in the problems. The solution however may provide a clue as to possibly a good solution of the original problem (MD). Thus, it would be helpful to transform the solution matrix to a matrix which is close to a block diagonal one. To this end, we exploit the spectrum of the optimal solution.

Let $\lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{R}$ be the eigenvalues of an optimal solution $Z^*$ for the relaxation problem and $\boldsymbol{u}^1, \boldsymbol{u}^2, \ldots, \boldsymbol{u}^n \in \mathbb{R}^n$ be their corresponding eigenvectors. We assume that the eigenvalues are sorted in the non-increasing order, that is, $1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. Focusing on the eigenvector $\boldsymbol{u}^2 = (u_1^2, u_2^2, \ldots, u_n^2)^\top$ corresponding to the second largest eigenvalue, we permutate the rows and columns of the matrix $Z^*$ consistent with the non-decreasing order of values of components of $\boldsymbol{u}^2$. Take a benchmark instance Karate for example, we show an optimal solution $Z^*$ of (DNN) and the matrix obtained in the manner described above in Figures 3.1 (a) and (b). From these figures, we observe that the permuted matrix is considerably close to a block diagonal matrix. However, there is no theoretical validity of using the eigenvector corresponding to the second largest eigenvalue. Here still remains room for further research.



(a) Original matrix (b) Permuted matrix

Figure 3.1: Comparison with two matrices

### 3.4.2 Dynamic programming

Next we discuss how to construct a feasible solution of (MD) from the permuted matrix. Let $\bar{V}$ be a sequence of nodes consistent with the non-decreasing order of components of $\boldsymbol{u}^2$. For the sake of notational simplicity, we renumber the nodes in $V$ and denote the sequence $\bar{V}$ by $(1, 2, \ldots, n)$. Now, we try to find a partition with maximum modularity density of $\bar{V}$ under the constraint that each member consists of consecutive indices of $\bar{V}$. For this problem, we propose an algorithm using the dynamic programming. We define $q(k, l)$ by

$$q(k,l) = \frac{2\sum_{i=k}^{l}\sum_{j=k}^{l} a_{ij} - \sum_{i=k}^{l} d_i}{l-(k-1)}$$

for $k$ and $l$ of $\bar{V}$ with $k \leq l$. The value $q(k, l)$ represents the contribution of the community $(k, \ldots, l)$ of $\bar{V}$ to the modularity density when it is selected as a member of the partition. For any index $s$ of $\bar{V}$, let $\mu(s)$ be the maximum modularity density that is achieved by partitioning the sequence $(1, \ldots, s)$ into several consecutive subsequences. If we define $\mu(0) = 0$ for notational convenience, then $\mu(s)$ satisfies the recursive equation

$$\mu(s) = \max\{\, \mu(h) + q(h+1, s) \mid h \in \{0, 1, \ldots, s-1\}\,\}. \tag{3.9}$$

Owing to (3.9), starting from $\mu(1) = q(1, 1)$, we first obtain

$$\mu(2) = \max\{\, q(1,2),\ \mu(1) + q(2,2)\,\},$$

then obtain $\mu(3)$ by means of $\mu(1)$ and $\mu(2)$, and so on. Our algorithm is given as follows.

**Algorithm DP** (Dynamic Programming)

Step 1 : Solve the relaxation problem to obtain an optimal solution $Z^*$.

Step 2 : Find the eigenvector $\boldsymbol{u}^2$ corresponding to the second largest eigenvalue of $Z^*$. Let $\bar{V} = (1, 2, \ldots, n)$ be a sequence of nodes obtained by rearranging them in non-decreasing order of corresponding components in $\boldsymbol{u}^2$.

Step 3 : Set $\mu(0) := 0$.

`for` $s = 1$ `to` $n$ `do`

Compute $\mu(s)$ according to (3.9).

`end-do`

## 3.5 Computational Experiments

To evaluate the lower and upper bounds obtained by our algorithm, we conducted the computational experiments. The experiments were performed on a computer with Intel Core i7, 3.70 GHz processor and 32.0 GB of memory. Using SeDuMi 1.2 as an SDP solver, we implemented the algorithm in MATLAB 2010.

Table 3.1: Solved instances

| ID | name | $n$ | $m$ | LB$^\star$ | $t^\star$ |
|---|---|---|---|---|---|
| 1 | Strike | 24 | 38 | 8.8611 | 4 |
| 2 | Karate | 34 | 78 | 7.8451 | 3 |
| 3 | Mexico | 35 | 117 | 8.7180 | 3 |
| 4 | Sawmill | 36 | 62 | 8.6233 | 4 |
| 5 | Dolphins | 62 | 159 | 12.1252 | 5 |
| 6 | Les Misérables | 77 | 254 | 24.5339 | 9 |
| 7 | Books | 105 | 441 | 21.9652 | 7 |

In the experiments, we used seven instances; Michael's strike dataset [68], Zachary's karate dataset [94], Gil-Mendieta and Schmidt's Mexico dataset [45], Michael and Massey's

sawmill dataset [69], Lusseau's dolphins dataset [66], Hugo's Les Misérables dataset [57], and Krebs' books dataset [58]. Details of each instance are listed in Table 3.1, where the columns "LB$^\star$" and "$t^\star$" represent the known lower bound and the corresponding number of communities, respectively. Since the first four instances in the table were solved to optimality in Costa [29], the optimal values and the optimal numbers of communities are listed. For the remaining instances, we list the lower bounds and the number of communities reported in Costa *et al.* [30] since the instances were not solved so far to our knowledge.

Table 3.2 shows the computational results of the algorithm described in Subsection 3.4.2, where the columns "UB", "LB", "gap" and "time" represent the optimal value of the relaxation problem, the lower bound obtained by the algorithm DP, the duality gap defined by $100(\text{UB} - \text{LB})/\text{LB}$, and the computation time in seconds, respectively. For each instance, we observed that the predominant portion of the computation time was spent for solving a relaxation problem, and the remaining parts of the algorithm require a fraction of time, specifically less than one second.

Table 3.2: Computational results of our algorithm

| ID | (DNN) | | | | $(\overline{\text{DNN}})$ | | | |
|---|---|---|---|---|---|---|---|---|
| | UB | LB | gap(%) | time(sec.) | UB | LB | gap(%) | time(sec.) |
| 1 | 9.5808 | 8.8611 | 8.122 | 1.05 | 9.3049 | 8.8611 | 5.008 | 3.54 |
| 2 | 8.9548 | 7.8424 | 14.184 | 5.83 | 8.4141 | 7.8451 | 7.253 | 36.04 |
| 3 | 10.3151 | 8.5580 | 20.532 | 7.64 | 9.9570 | 8.5227 | 16.829 | 43.48 |
| 4 | 10.5048 | 7.0486 | 49.034 | 7.75 | 10.0311 | 7.3587 | 36.316 | 54.21 |
| 5 | 15.0218 | 9.8286 | 52.838 | 316.61 | 14.3552 | 11.4610 | 25.253 | 1681.81 |
| 6 | 28.0957 | 22.2680 | 35.279 | 658.28 | 27.4276 | 23.3416 | 17.505 | 7018.03 |
| 7 | 26.5387 | 20.2470 | 31.075 | 4626.11 | 24.7749 | 20.3150 | 21.953 | 60437.45 |

From Table 3.2, we observe that the upper bounds UB provided by $(\overline{\text{DNN}})$ are tighter than those provided by $(\text{DNN})$ for all instances, which indicates the effectiveness of the valid inequalities in Lemma 3.5. Moreover, we also confirm that the lower bounds obtained for $(\overline{\text{DNN}})$ are equal to or larger than those obtained for $(\text{DNN})$ for all instances

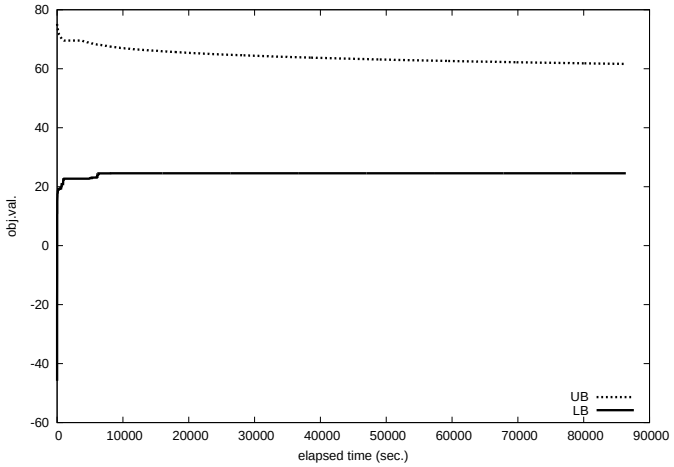
except Mexico (ID=3). However, solving the problem $(\overline{\text{DNN}})$ requires a rather long computation time compared with solving $(\text{DNN})$ as the instance size grows. To be specific, for the instance Books (ID=7), $(\text{DNN})$ takes approximately 4600 seconds, whereas $(\overline{\text{DNN}})$ takes over 60000 seconds, approximately 16 hours.

Table 3.3: Computational results of the branch-and-bound algorithm for $(\text{MILP}_1)$

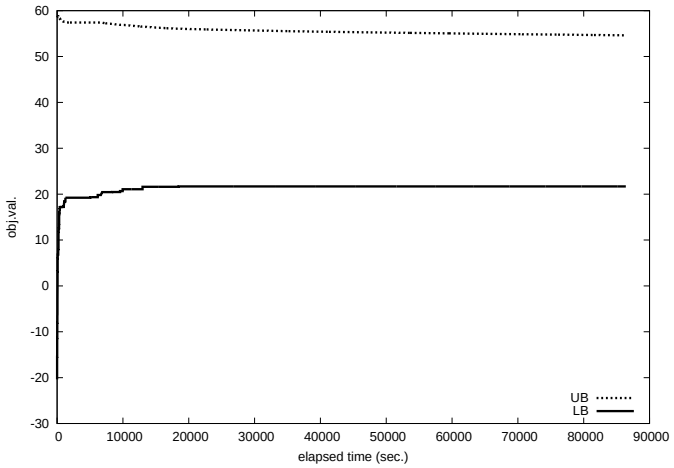
| ID | $(\text{MILP}_1)$ | | | |
|---|---|---|---|---|
| | UB | LB | gap(%) | time(sec.) |
| 1 | 8.8611 | 8.8611 | 0 | 0.50 |
| 2 | 7.8451 | 7.8451 | 0 | 0.74 |
| 3 | 8.7180 | 8.7180 | 0 | 7.84 |
| 4 | 8.6233 | 8.6233 | 0 | 6.10 |
| 5 | 13.8466 | 12.1252 | 14.196 | 86400.00 |
| 6 | 61.6302 | 24.5339 | 151.204 | 86400.00 |
| 7 | 54.6234 | 21.6803 | 151.949 | 86400.00 |

Next, we solve the problem $(\text{MILP}_1)$ by branch-and-bound algorithm in Gurobi 6.0.0 to compare the quality of the solutions obtained by our algorithm for 0-1SDP formulation. The computational results are given in Table 3.3. In our experiments, we impose a time limit of 86400 seconds, 24 hours, on the computation time of the branch-and-bound algorithm.

In Table 3.3, we see that the first four instances are solved to optimality within a short computation time, while the remaining instances cannot be solved within the time limit. Especially, for the instances Les Misérables (ID=6) and Books (ID=7), a quite large duality gap remains. Figures 3.2 (a) and (b) show the upper and lower bounds vs. the elapsed time. From the figures, we observe the following behavior: (i) the branch-and-bound algorithm gives a good lower bound in early stage, and (ii) the improvement of upper bound rarely occurs throughout the computation. Owing to the latter of these, the duality gap still remains large even though a good feasible solution has been found. This suggests that deriving a tight upper bound enables us to estimate the accuracy of an incumbent solution obtained by the branch-and-bound algorithm more precisely.




(a) Instance Les Misérables




(b) Instance Books

Figure 3.2: The upper and lower bounds vs. elapsed time in the branch-and-bound

# Chapter 4

# Ranking Problem

## 4.1 Introduction

We are concerned with a multi-class classification problem of $n$ objects, each of which is endowed with an $m$-dimensional attribute vector $\boldsymbol{x}^i = (x_1^i, x_2^i, \ldots, x_m^i)^\top \in \mathbb{R}^m$ and a label $\ell_i$. The underlying statistical model assumes that object $i$ receives label $k$, i.e., $\ell_i = k$, when latent variable $y_i$ determined by

$$y_i = \boldsymbol{w}^\top \boldsymbol{x}^i + \epsilon_i$$

falls between two thresholds $p_k$ and $p_{k+1}$, where $\epsilon_i$ represents a random noise whose probabilistic property is not known. Namely, attribute vectors of objects are loosely separated by hyperplanes $H(\boldsymbol{w}, p_k) = \{\, \boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{w}^\top \boldsymbol{x} = p_k \,\}$ for $k = 1, 2, \ldots, l$ which share a common normal vector $\boldsymbol{w}$, then each object is given a label according to the layer it is located in. Note that neither $y_i$'s, $w_j$'s nor $p_k$'s are observable. Our problem is to find the vector $\boldsymbol{w} \in \mathbb{R}^m$ as well as the thresholds $p_1, p_2, \ldots, p_l$ that best fit the input data $\{\, (\boldsymbol{x}^i, \ell_i) \mid i = 1, 2, \ldots, n \,\}$.

This problem is known as the *ranking problem* and frequently arises in social sciences and operations research. See, for instance Crammer and Singer [31], Herbrich *et al.* [54], Liu [64], Shashua and Levin [83] and Chapter 8 of Shawe-Taylor and Cristianini [84]. It is a variation of the multi-class classification problem, for which several learning algorithms of the *support vector machine* (*SVM* for short) have been proposed. We refer the reader to Chapters 4.1.2 and 7.1.3 of Bishop [15], Chapter 10.10 of Vapnik [90] and Tatsumi *et al.* [87] and references therein. What distinguishes the problem from other multi-class

classification problems is that the identical normal vector should be shared by all the separating hyperplanes. In this chapter, based on the formulation *fixed margin strategy* by Shashua and Levin [83], we propose a row and column generation algorithm to maximize the minimum margin for the ranking problems.

### 4.1.1 Definitions and notation

Throughout this chapter, $N = \{1, 2, \ldots, i, \ldots, n\}$ denotes the set of $n$ objects and $\boldsymbol{x}^i = (x_1^i, x_2^i, \ldots, x_m^i)^\top \in \mathbb{R}^m$ denotes the attribute vector of object $i$. The predetermined set of labels is $L = \{0, 1, \ldots, k, \ldots, l\}$ and the label assigned to object $i$ is denoted by $\ell_i$. Let $N(k) = \{\, i \in N \mid \ell_i = k \,\}$ be the set of objects with label $k \in L$, and for notational convenience we write $n(k) = |N(k)|$ for $k \in L$, and $N(k..k') = N(k) \cup N(k+1) \cup \cdots \cup N(k')$ for $k, k' \in L$ such that $k < k'$. We can assume that $i < j$ holds when $\ell_i < \ell_j$ for $i, j \in N$, by rearranging the objects if necessary, hence $N(0) = \{\, 1, 2, \ldots, n(0) \,\}$, $N(1) = \{\, n(0)+1, \ldots, n(0)+n(1) \,\}$, and so forth. For succinct notation we define

$$X = \begin{bmatrix} & \cdots & \boldsymbol{x}^i & \cdots & \\ & & & & \end{bmatrix}_{i \in N} \in \mathbb{R}^{m \times n} \tag{4.1}$$

$$X_W = \begin{bmatrix} \cdots & \boldsymbol{x}^i & \cdots \\ & & \end{bmatrix}_{i \in W} \in \mathbb{R}^{m \times |W|}$$

for $W \subseteq N$, and the corresponding Gram matrices

$$\begin{aligned} K &= X^\top X \in \mathbb{R}^{n \times n}, \\ K_W &= X_W^\top X_W \in \mathbb{R}^{|W| \times |W|}. \end{aligned} \tag{4.2}$$

We denote the $k$-dimensional zero vector and the $k$-dimensional vector of 1's by $\mathbf{0}_k$ and $\mathbf{1}_k$, respectively. Given a subset $W \subseteq N$ and a vector $\boldsymbol{\alpha} = (\alpha_i)_{i \in W}$ we use the notation $(\boldsymbol{\alpha}_W, \mathbf{0}_{N \setminus W})$ to denote the $n$-dimensional vector $\bar{\boldsymbol{\alpha}}$ such that

$$\bar{\alpha}_i = \begin{cases} \alpha_i & \text{when } i \in W \\ 0 & \text{otherwise.} \end{cases}$$

## 4.2 Maximization of Minimum Margin for Separable Case

### 4.2.1 Primal of hard margin problem

Henceforth we assume that $N(k) \neq \emptyset$ for all $k \in L$ for the sake of simplicity, and adopt the notational convention that $p_0 = -\infty$ and $p_{l+1} = +\infty$. We say that an instance $\{ (\boldsymbol{x}^i, \ell_i) \mid i \in N \}$ is *separable* if there exist $\boldsymbol{w} \in \mathbb{R}^m$ and $\boldsymbol{p} = (p_1, p_2, \ldots, p_l)^\top \in \mathbb{R}^l$ such that

$$p_{\ell_i} < \boldsymbol{w}^\top \boldsymbol{x}^i < p_{\ell_i+1} \quad \text{for } i \in N.$$

Clearly an instance is separable if and only if there are $\boldsymbol{w}$ and $\boldsymbol{p}$ such that

$$p_{\ell_i} + 1 \leq \boldsymbol{w}^\top \boldsymbol{x}^i \leq p_{\ell_i+1} - 1 \quad \text{for } i \in N.$$

For each $k \in L \setminus \{0\}$ we see that

$$\max_{i \in N(k-1)} (\boldsymbol{w})^\top \boldsymbol{x}^i \leq p_k - 1 < p_k < p_k + 1 \leq \min_{j \in N(k)} (\boldsymbol{w})^\top \boldsymbol{x}^j,$$

implying

$$\min_{j \in N(k)} \frac{\boldsymbol{w}^\top}{\|\boldsymbol{w}\|} \boldsymbol{x}^j - \max_{i \in N(k-1)} \frac{\boldsymbol{w}^\top}{\|\boldsymbol{w}\|} \boldsymbol{x}^i \geq \frac{2}{\|\boldsymbol{w}\|}.$$

Then the margin between $\{ \boldsymbol{x}^i \mid i \in N(k-1) \}$ and $\{ \boldsymbol{x}^j \mid j \in N(k) \}$ is at least $2/\|\boldsymbol{w}\|$ for $k = 2, \ldots, l$. Hence the maximization of the minimum margin is formulated as the quadratic programming

$$\left| \begin{array}{ll} \text{minimize} & \dfrac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} & p_{\ell_i} + 1 \leq (\boldsymbol{x}^i)^\top \boldsymbol{w} \leq p_{\ell_i+1} - 1 \quad (i \in N), \end{array} \right.$$

more explicitly with the notation introduced in Section 4.1

$$(H) \quad \left| \begin{array}{ll} \text{minimize} & \dfrac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} & 1 - (\boldsymbol{x}^i)^\top \boldsymbol{w} + p_{\ell_i} \leq 0 \qquad (i \in N(1..l)) \\ & 1 + (\boldsymbol{x}^i)^\top \boldsymbol{w} - p_{\ell_i+1} \leq 0 \quad (i \in N(0..l-1)). \end{array} \right.$$

The constraints therein are called *hard margin* constraints, and we name this problem $(H)$. The leading coefficient $1/2$ of the objective function is for the sake of notational simplicity in further discussion.

### 4.2.2 Dual of hard margin problem

Applying the kernel technique to the problem $(H)$ in later discussion, we give the dual problem of $(H)$ in this subsection. The Lagrangian function for the hard margin problem $(H)$ introduced in the previous subsection is

$$L(\boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i \in N(1..l)} \alpha_i(1 - (\boldsymbol{x}^i)^\top \boldsymbol{w} + p_{\ell_i}) + \sum_{i \in N(0..l-1)} \beta_i(1 + (\boldsymbol{x}^i)^\top \boldsymbol{w} - p_{\ell_i+1}),$$

where $\alpha_i$ and $\beta_i$ are nonnegative Lagrangian multipliers. Denoting

$$\boldsymbol{\alpha} = (\alpha_i, \ldots, \alpha_n)^\top \in \mathbb{R}^n \text{ and } \boldsymbol{\beta} = (\beta_i, \ldots, \beta_n)^\top \in \mathbb{R}^n$$

with the convention that

$$\alpha_i = 0 \text{ for all } i \in N(0) \text{ and } \beta_i = 0 \text{ for all } i \in N(l),$$

the Lagrangian function is compactly written as

$$L(\boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - (X(\boldsymbol{\alpha} - \boldsymbol{\beta}))^\top \boldsymbol{w} + (A\boldsymbol{\alpha} - B\boldsymbol{\beta})^\top \boldsymbol{p} + \boldsymbol{\alpha}^\top \mathbf{1}_n + \boldsymbol{\beta}^\top \mathbf{1}_n,$$

where the matrix $X$ is defined in (4.1), and

$$A = \begin{bmatrix} \mathbf{0}_{n(0)}^\top & \mathbf{1}_{n(1)}^\top & & & \\ & & \mathbf{1}_{n(2)}^\top & & \\ & & & \ddots & \\ & & & & \mathbf{1}_{n(l)}^\top \end{bmatrix} \in \mathbb{R}^{l \times n},$$

$$B = \begin{bmatrix} \mathbf{1}_{n(0)}^\top & & & & \\ & \mathbf{1}_{n(1)}^\top & & & \\ & & \ddots & & \\ & & & \mathbf{1}_{n(l-1)}^\top & \mathbf{0}_{n(l)}^\top \end{bmatrix} \in \mathbb{R}^{l \times n}.$$

Let

$$\omega(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \min\{\, L(\boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \mid (\boldsymbol{w}, \boldsymbol{p}) \in \mathbb{R}^{m+l} \,\},$$

then the *Lagrangian dual of the hard margin problem* is

$$(dH) \quad \left| \begin{array}{ll} \text{minimize} & \omega(\boldsymbol{\alpha}, \boldsymbol{\beta}) \\ \text{subject to} & (\boldsymbol{\alpha}, \boldsymbol{\beta}) \geq \mathbf{0}_{2n}. \end{array} \right.$$

Since $L$ is a convex function with respect to $(\boldsymbol{w}, \boldsymbol{p})$, a point $(\boldsymbol{w}^*, \boldsymbol{p}^*)$ attains the minimum $\omega(\boldsymbol{\alpha}, \boldsymbol{\beta})$ for a given $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ if and only if the partial derivatives of $L$ with respect to $(\boldsymbol{w}, \boldsymbol{p})$ vanish at $(\boldsymbol{w}^*, \boldsymbol{p}^*)$, i.e.,

$$\frac{\partial L}{\partial(\boldsymbol{w}, \boldsymbol{p})}(\boldsymbol{w}^*, \boldsymbol{p}^*, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathbf{0}_{n+l}.$$

This condition reduces to

$$\boldsymbol{w}^* - X(\boldsymbol{\alpha} - \boldsymbol{\beta}) = \mathbf{0}_n$$

and

$$A\boldsymbol{\alpha} - B\boldsymbol{\beta} = \mathbf{0}_l. \tag{4.3}$$

Plugging these equalities into $L$, we obtain

$$\omega(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\left(\frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\beta})^\top K(\boldsymbol{\alpha} - \boldsymbol{\beta}) - \mathbf{1}_n^\top(\boldsymbol{\alpha} + \boldsymbol{\beta})\right),$$

where the matrix $K$ is defined in (4.2). Note that the variable $\boldsymbol{p}$ disappears due to the equality condition (4.3). Deleting the leading negative coefficient $-1$, the Lagrangian dual of the hard margin problem is given as

$$(dH) \quad \left| \begin{array}{ll} \text{minimize} & \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\beta})^\top K(\boldsymbol{\alpha} - \boldsymbol{\beta}) - \mathbf{1}_n^\top(\boldsymbol{\alpha} + \boldsymbol{\beta}) \\ \text{subject to} & A\boldsymbol{\alpha} - B\boldsymbol{\beta} = \mathbf{0}_l \\ & \boldsymbol{\alpha}_{N(0)} = \mathbf{0}_{n(0)} \\ & \boldsymbol{\beta}_{N(l)} = \mathbf{0}_{n(l)} \\ & \boldsymbol{\alpha} \geq \mathbf{0}_n \\ & \boldsymbol{\beta} \geq \mathbf{0}_n, \end{array} \right.$$

where $\boldsymbol{\alpha}_{N(0)}$ is a vector of $\alpha_i$'s for $i \in N(0)$ and $\boldsymbol{\beta}_{N(l)}$ is a vector of $\beta_i$'s for $i \in N(l)$. This is a convex quadratic minimization problem since the matrix $K$ is nonnegative definite. Let $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ denote an optimal solution of $(dH)$. Then an optimal normal vector $\boldsymbol{w}^*$ and an optimum threshold vector $\boldsymbol{p}^*$ of the primal problem are given by

$$\begin{aligned} \boldsymbol{w}^* &= X(\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*), \\ p_k^* &= \frac{1}{2}\left(\max_{i \in N(k-1)}(\boldsymbol{w}^*)^\top \boldsymbol{x}^i + \min_{i \in N(k)}(\boldsymbol{w}^*)^\top \boldsymbol{x}^i\right) \text{ for } k \in L \setminus \{0\}. \end{aligned} \tag{4.4}$$

### 4.2.3 Kernel technique for hard margin problem

The matrix $K$ of the Lagrangian dual of the hard margin problem $(dH)$ is composed of the inner products $(\boldsymbol{x}^i)^\top \boldsymbol{x}^j$ for $i, j \in N$, which enables us to apply the *kernel technique* simply by replacing them by $\kappa(\boldsymbol{x}^i, \boldsymbol{x}^j)$ for some appropriate kernel function $\kappa$.

Let $\phi : \mathbb{R}^m \to \mathbb{F}$ be a function, possibly unknown, from $\mathbb{R}^m$ to some higher dimensional inner product space $\mathbb{F}$, so-called the *feature space* such that

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle$$

holds for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$, where $\langle \cdot, \cdot \rangle$ is the inner product defined on $\mathbb{F}$. In the sequel, we denote $\tilde{\boldsymbol{x}} = \phi(\boldsymbol{x})$. The kernel technique considers the vectors $\tilde{\boldsymbol{x}}^i \in \mathbb{F}$ instead of $\boldsymbol{x}^i \in \mathbb{R}^m$, and finds the normal vector $\tilde{\boldsymbol{w}} \in \mathbb{F}$ and thresholds $p_1, \ldots, p_l$. Therefore the matrices $X$ and $K$ should be replaced by $\tilde{X}$ consisting of vectors $\tilde{\boldsymbol{x}}^i$ and $\tilde{K} = \tilde{X}^\top \tilde{X}$, respectively. Note that the latter matrix is given as

$$\tilde{K} = \left[ \kappa(\boldsymbol{x}^i, \boldsymbol{x}^j) \right]_{i,j \in N} \in \mathbb{R}^{n \times n}$$

by the kernel function $\kappa$. Then the kernel version of the hard margin problem is given as

$$(d\tilde{H}) \quad \left| \begin{array}{ll} \text{minimize} & \dfrac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\beta})^\top \tilde{K}(\boldsymbol{\alpha} - \boldsymbol{\beta}) - \mathbf{1}_n^\top(\boldsymbol{\alpha} + \boldsymbol{\beta}) \\ \text{subject to} & A\boldsymbol{\alpha} - B\boldsymbol{\beta} = \mathbf{0}_l \\ & \boldsymbol{\alpha}_{N(0)} = \mathbf{0}_{n(0)} \\ & \boldsymbol{\beta}_{N(l)} = \mathbf{0}_{n(l)} \\ & \boldsymbol{\alpha} \geq \mathbf{0}_n \\ & \boldsymbol{\beta} \geq \mathbf{0}_n. \end{array} \right.$$

Solving the kernel hard margin problem $(d\tilde{H})$ to find $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$, the optimal normal vector $\tilde{\boldsymbol{w}}^* \in \mathbb{F}$ of the primal problem would be given as

$$\tilde{\boldsymbol{w}}^* = \sum_{i \in N} (\alpha_i^* - \beta_i^*) \tilde{\boldsymbol{x}}^i,$$

which is in general not available due to the absence of an explicit representation of $\tilde{\boldsymbol{x}}$. However the value of $\langle \tilde{\boldsymbol{w}}^*, \tilde{\boldsymbol{x}}^j \rangle$ necessary to compute the thresholds $p_k^*$ according to (4.4)

is obtained as the inner product of $\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*$ and the $j$-th column of $\tilde{K}$. In fact

$$\begin{aligned}\langle \tilde{\boldsymbol{w}}^*, \tilde{\boldsymbol{x}}^j \rangle &= \langle \sum_{i \in N} (\alpha_i^* - \beta_i^*) \tilde{\boldsymbol{x}}^i, \tilde{\boldsymbol{x}}^j \rangle = \sum_{i \in N} (\alpha_i^* - \beta_i^*) \langle \tilde{\boldsymbol{x}}^i, \tilde{\boldsymbol{x}}^j \rangle \\ &= (\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*)^\top \begin{pmatrix} \vdots \\ \langle \tilde{\boldsymbol{x}}^i, \tilde{\boldsymbol{x}}^j \rangle \\ \vdots \end{pmatrix} = (\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*)^\top \begin{pmatrix} \vdots \\ \kappa(\boldsymbol{x}^i, \boldsymbol{x}^j) \\ \vdots \end{pmatrix}.\end{aligned}$$

Suppose we are given a newly arrived object with attribute vector $\boldsymbol{x} \in \mathbb{R}^m$ to assign a label. In the same way as above we have

$$\langle \tilde{\boldsymbol{w}}^*, \tilde{\boldsymbol{x}} \rangle = (\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*)^\top \begin{pmatrix} \vdots \\ \kappa(\boldsymbol{x}^i, \boldsymbol{x}^j) \\ \vdots \end{pmatrix}.$$

Then by locating the threshold interval into which this value falls, we can assign a label to the newly arrived object.

## 4.3 Maximization of Minimum Margin for Non-Separable Case

### 4.3.1 Primal of soft margin problem

Similarly to the binary SVM, introducing nonnegative slack variables $\xi_{-i}$ and $\xi_{+i}$ for $i \in N$ relaxes the hard margin constraints to *soft margin* constraints:

$$p_{\ell_i} + 1 - \xi_{-i} \leq \boldsymbol{w}^\top \boldsymbol{x}^i \leq p_{\ell_i+1} - 1 + \xi_{+i} \quad \text{for } i \in N.$$

Positive values of variables $\xi_{-i}$ and $\xi_{+i}$ mean misclassification, hence they should be as small as possible. If we penalize positive $\xi_{-i}$ and $\xi_{+i}$ by adding $\sum_{i \in N} (\xi_{-i} + \xi_{+i})$ to the objective function, we have the following *primal soft margin problem* with *1-norm penalty*.

$$(S_1) \quad \left| \begin{array}{ll} \text{minimize} & \dfrac{1}{2} \|\boldsymbol{w}\|^2 + \dfrac{1}{2} c \, \mathbf{1}_n^\top (\boldsymbol{\xi}_- + \boldsymbol{\xi}_+) \\ \text{subject to} & p_{\ell_i} + 1 - \xi_{-i} \leq (\boldsymbol{x}^i)^\top \boldsymbol{w} \leq p_{\ell_i+1} - 1 + \xi_{+i} \quad (i \in N) \\ & \boldsymbol{\xi}_-, \boldsymbol{\xi}_+ \geq \mathbf{0}_n, \end{array} \right.$$

where $\boldsymbol{\xi}_- = (\xi_{-1}, \ldots, \xi_{-n}), \boldsymbol{\xi}_+ = (\xi_{+1}, \ldots, \xi_{+n})$ and $c$ is a *penalty parameter*. When *2-norm penalty* is employed, we have

$$(S_2) \quad \left| \begin{array}{ll} \text{minimize} & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{2}c\,(\|\boldsymbol{\xi}_-\|^2 + \|\boldsymbol{\xi}_+\|^2) \\ \text{subject to} & p_{\ell_i} + 1 - \xi_{-i} \leq (\boldsymbol{x}^i)^\top \boldsymbol{w} \leq p_{\ell_i+1} - 1 + \xi_{+i} \quad (i \in N) \\ & \boldsymbol{\xi}_-, \boldsymbol{\xi}_+ \geq \mathbf{0}_n, \end{array} \right.$$

**Lemma 4.1** *The non-negativity constraints on variables $\xi_{-i}$ and $\xi_{+i}$ of problem $(S_2)$ are redundant.*

**Proof :** Let $(\boldsymbol{w}, \boldsymbol{\xi}_-, \boldsymbol{\xi}_+)$ be a feasible solution of $(S_2)$ with the non-negativity constraints removed. If $\xi_{-i} < 0$ for some $i \in N$, replacing it with zero will reduce the objective function value. Therefore $\boldsymbol{\xi}_-$ and $\boldsymbol{\xi}_+$ are nonnegative at any optimum solution of $(S_2)$. ■

Thus our problem with 2-norm penalty reduces to

$$(S_2) \quad \left| \begin{array}{ll} \text{minimize} & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{2}c\,(\|\boldsymbol{\xi}_-\|^2 + \|\boldsymbol{\xi}_+\|^2) \\ \text{subject to} & p_{\ell_i} + 1 - \xi_{-i} \leq (\boldsymbol{x}^i)^\top \boldsymbol{w} \leq p_{\ell_i+1} - 1 + \xi_{+i} \quad (i \in N). \end{array} \right.$$

Naturally, we could add the constraints

$$p_{k'} + 1 - \xi_{-i} \leq \boldsymbol{w}^\top \boldsymbol{x}^i \leq p_{k''} - 1 + \xi_{+i} \quad (k', k'' \in L;\ k' \leq k < k'')$$

to each of the above formulations. It would, however, inflate the problem size and most of those constraints would be likely redundant. Therefore we will not discuss this formulation.

### 4.3.2 Dual of soft margin problem

In this subsection we will make the Lagrangian dual of the soft margin problems.

We begin by looking at the soft margin problem with 1-norm penalty $(S_1)$. The Lagrangian function for $(S_1)$ is

$$\begin{aligned} L(\boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\xi}_-, \boldsymbol{\xi}_+, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - X((\boldsymbol{\alpha} - \boldsymbol{\beta}))^\top \boldsymbol{w} + (A\boldsymbol{\alpha} - B\boldsymbol{\beta})^\top \boldsymbol{p} \\ + \boldsymbol{\alpha}^\top \mathbf{1}_n + \boldsymbol{\beta}^\top \mathbf{1}_n + (c\mathbf{1}_n - \boldsymbol{\alpha})^\top \boldsymbol{\xi}_- + (c\mathbf{1}_n - \boldsymbol{\beta})^\top \boldsymbol{\xi}_+ \\ - \boldsymbol{\lambda}^\top \boldsymbol{\xi}_- - \boldsymbol{\mu}^\top \boldsymbol{\xi}_+, \end{aligned}$$

and the Lagrangian relaxation problem for a given nonnegative multiplier vector $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \in \mathbb{R}_+^{4n}$ is

$$\left| \begin{array}{ll} \text{minimize} & L(\boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\xi}_-, \boldsymbol{\xi}_+, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ \text{subject to} & (\boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\xi}_-, \boldsymbol{\xi}_+) \in \mathbb{R}^{m+l+2n}. \end{array} \right.$$

Thank to the convexity of the problem, the optimality condition of $(\boldsymbol{w}^*, \boldsymbol{p}^*, \boldsymbol{\xi}_-^*, \boldsymbol{\xi}_+^*)$ is simply given as

$$\frac{\partial L}{\partial(\boldsymbol{w}, \boldsymbol{p})}(\boldsymbol{w}^*, \boldsymbol{p}^*, \boldsymbol{\xi}_-^*, \boldsymbol{\xi}_+^*, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{0}_{n+l},$$
$$\frac{\partial L}{\partial(\boldsymbol{\xi}_-, \boldsymbol{\xi}_+)}(\boldsymbol{w}^*, \boldsymbol{p}^*, \boldsymbol{\xi}_-^*, \boldsymbol{\xi}_+^*, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{0}_{2n},$$

each of which reduces to

$$\boldsymbol{w}^* = X(\boldsymbol{\alpha} - \boldsymbol{\beta}) \tag{4.5}$$
$$A\boldsymbol{\alpha} - B\boldsymbol{\beta} = \mathbf{0}_l \tag{4.6}$$
$$\boldsymbol{\alpha} \leq c\,\mathbf{1}_n \tag{4.7}$$
$$\boldsymbol{\beta} \leq c\,\mathbf{1}_n \tag{4.8}$$

by virtue of the non-negativity of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$. The *complementarity condition*

$$(c\mathbf{1}_n - \boldsymbol{\alpha}^*)^\top \boldsymbol{\xi}_-^* = (c\mathbf{1}_n - \boldsymbol{\beta}^*)^\top \boldsymbol{\xi}_+^* = 0 \tag{4.9}$$

holds for a primal optimal solution $(\boldsymbol{\xi}_-^*, \boldsymbol{\xi}_+^*)$ and a dual optimal solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$. Denoting the optimum objective function value of the above relaxation problem by $\omega(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ the *Lagrangian dual of the soft margin problem* is

$$\left| \begin{array}{ll} \text{maximize} & \omega(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ \text{subject to} & (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \geq \mathbf{0}_{4n}. \end{array} \right.$$

Plugging (4.5), (4.6) and (4.9) into the Lagrangian function, the Lagrangian dual problem is rewritten as

$$(dS_1) \quad \left| \begin{array}{ll} \text{minimize} & \dfrac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\beta})^\top K(\boldsymbol{\alpha} - \boldsymbol{\beta}) - \mathbf{1}_n^\top(\boldsymbol{\alpha} + \boldsymbol{\beta}) \\ \text{subject to} & A\boldsymbol{\alpha} - B\boldsymbol{\beta} = \mathbf{0}_l \\ & \boldsymbol{\alpha}_{N(0)} = \mathbf{0}_{n(0)} \\ & \boldsymbol{\beta}_{N(l)} = \mathbf{0}_{n(l)} \\ & \mathbf{0}_n \leq \boldsymbol{\alpha} \leq c\,\mathbf{1}_n \\ & \mathbf{0}_n \leq \boldsymbol{\beta} \leq c\,\mathbf{1}_n, \end{array} \right.$$

This differs from the dual of the hard margin problem $(dH)$ only in the additional upper bound constraints (4.7) and (4.8) of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. From the complementarity condition (4.9), we have

$$\alpha_i^* < 1 \text{ implies } \xi_{-i}^* = 0, \text{ i.e., } p_{\ell_i}^* + 1 \leq (\boldsymbol{w}^*)^\top \boldsymbol{x}^i$$
$$\beta_i^* < 1 \text{ implies } \xi_{+i}^* = 0, \text{ i.e., } (\boldsymbol{w}^*)^\top \boldsymbol{x}^i \leq p_{\ell_i+1}^* - 1,$$

hence an optimum normal vector $\boldsymbol{w}^*$ and an optimum thresholds vector $\boldsymbol{p}^*$ of the primal problem $(S_1)$ are given as follows:

$$\boldsymbol{w}^* = X(\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*)$$
$$p_k^* = \frac{1}{2}\left(\max_{i \in N(k-1); \beta_i^* < 1} (\boldsymbol{w}^*)^\top \boldsymbol{x}^i + \min_{i \in N(k); \alpha_i^* < 1} (\boldsymbol{w}^*)^\top \boldsymbol{x}^i\right) \text{ for } k \in L \setminus \{0\}.$$

First equation follows from (4.5).

Next we consider the dual of the soft margin problem with 2-norm penalty $(S_2)$. Since the derivation of the optimality condition and the dual problem is technical, we defer the complete derivation to Appendix A. The dual of $(S_2)$ is

$$(dS_2) \quad \left| \begin{array}{ll} \text{minimize} & \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\beta})^\top K(\boldsymbol{\alpha} - \boldsymbol{\beta}) + \frac{1}{2c}(\boldsymbol{\alpha}^\top \boldsymbol{\alpha} + \boldsymbol{\beta}^\top \boldsymbol{\beta}) - \mathbf{1}_n^\top (\boldsymbol{\alpha} + \boldsymbol{\beta}) \\ \text{subject to} & A\boldsymbol{\alpha} - B\boldsymbol{\beta} = \mathbf{0}_l \\ & \boldsymbol{\alpha}_{N(0)} = \mathbf{0}_{n(0)} \\ & \boldsymbol{\beta}_{N(l)} = \mathbf{0}_{n(l)} \\ & \boldsymbol{\alpha} \geq \mathbf{0}_n \\ & \boldsymbol{\beta} \geq \mathbf{0}_n. \end{array} \right.$$

With the quadratic term $(\boldsymbol{\alpha}^\top \boldsymbol{\alpha} + \boldsymbol{\beta}^\top \boldsymbol{\beta})$ added, the objective function becomes strictly convex, which may lighten the computational burden. Furthermore,

$$c\,\boldsymbol{\xi}_-^* = \boldsymbol{\alpha}^* \text{ and } c\,\boldsymbol{\xi}_+^* = \boldsymbol{\beta}^* \tag{4.10}$$

hold between a pair of primal and dual optimum solutions. From (4.10), we have

$$\alpha_i^* = 0 \text{ implies } p_{\ell_i}^* + 1 \leq (\boldsymbol{w}^*)^\top \boldsymbol{x}^i$$
$$\beta_i^* = 0 \text{ implies } (\boldsymbol{w}^*)^\top \boldsymbol{x}^i \leq p_{\ell_i+1}^* - 1.$$

Therefore the optimal threshold $\boldsymbol{p}^*$ of the primal problem $(S_2)$ is determined by

$$p_k^* = \frac{1}{2}\left(\max_{i \in N(k-1);\, \beta_i^* = 0} (\boldsymbol{w}^*)^\top \boldsymbol{x}^i + \min_{i \in N(k);\, \alpha_i^* = 0} (\boldsymbol{w}^*)^\top \boldsymbol{x}^i\right).$$

### 4.3.3 Kernel technique for soft margin problem

Kernel technique can apply to the soft margin problems in the same way as discussed in the hard margin problem. Take the primal soft margin problem $(S_1)$, then kernel version of the problem is given by the dual problems $(dS_1)$ in the previous subsection with $K$ replaced by $\tilde{K}$. Namely, the problem to solve is

$$(d\tilde{S}_1) \quad \left| \begin{array}{ll} \text{minimize} & \frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\beta})^\top \tilde{K}(\boldsymbol{\alpha}-\boldsymbol{\beta}) - \mathbf{1}_n^\top(\boldsymbol{\alpha}+\boldsymbol{\beta}) \\ \text{subject to} & A\boldsymbol{\alpha} - B\boldsymbol{\beta} = \mathbf{0}_l \\ & \boldsymbol{\alpha}_{N(0)} = \mathbf{0}_{n(0)} \\ & \boldsymbol{\beta}_{N(l)} = \mathbf{0}_{n(l)} \\ & \mathbf{0}_n \leq \boldsymbol{\alpha} \leq c\mathbf{1}_n \\ & \mathbf{0}_n \leq \boldsymbol{\beta} \leq c\mathbf{1}_n. \end{array} \right.$$

For an optimal solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ of the problem $(d\tilde{S}_1)$, the optimum thresholds vector $\boldsymbol{p}^*$ should be determined by

$$p_k^* = \frac{1}{2}\left( \max_{i \in N(k-1); \beta_i^* < 1} \langle \tilde{\boldsymbol{w}}^*, \tilde{\boldsymbol{x}}^i \rangle + \min_{i \in N(k); \alpha_i^* < 1} \langle \tilde{\boldsymbol{w}}^*, \tilde{\boldsymbol{x}}^i \rangle \right) \text{ for } k \in L \setminus \{0\}.$$

In the same manner as in the hard margin problem, we have

$$\langle \tilde{\boldsymbol{w}}^*, \tilde{\boldsymbol{x}} \rangle = (\boldsymbol{\alpha}^* - \boldsymbol{\beta}^*)^\top \begin{pmatrix} \vdots \\ \kappa(\boldsymbol{x}^i, \boldsymbol{x}) \\ \vdots \end{pmatrix}$$

for $\boldsymbol{x} \in \mathbb{R}^m$. Thus $p_k^*$'s can be obtained without knowing $\tilde{\boldsymbol{w}}^*$.

## 4.4 Reformulations based on Dual Representation

### 4.4.1 Dual representation for hard margin problem

A close look at the primal problem $(H)$ shows that the following property holds for an optimum solution $\boldsymbol{w}^*$. See, for example Chapter 6 of Bishop [15], Shashua and Levin [83], and Theorem 1 in Scholkopf *et al.* [82].

**Lemma 4.2** *Let $(\boldsymbol{w}^*, \boldsymbol{p}^*) \in \mathbb{R}^{m+l}$ be an optimum solution of $(H)$. Then $\boldsymbol{w}^*$ lies in the range space of $X$, i.e., $\boldsymbol{w}^* = X\boldsymbol{\lambda}$ for some $\boldsymbol{\lambda} \in \mathbb{R}^n$.*

**Proof :** Let $\boldsymbol{w}_i$ be the orthogonal projection of $\boldsymbol{w}^*$ onto the range space of $X$ and let $\boldsymbol{w}_2 = \boldsymbol{w}^* - \boldsymbol{w}_1$. Then we obtain

$$(\boldsymbol{x}^i)^\top \boldsymbol{w}^* = (\boldsymbol{x}^i)^\top (\boldsymbol{w}_1 + \boldsymbol{w}_2) = (\boldsymbol{x}^i)^\top \boldsymbol{w}_1 \quad \text{for } i \in N,$$

meaning that $(\boldsymbol{w}_1, \boldsymbol{p}^*)$ is feasible to $(H)$, and

$$\frac{1}{2}\|\boldsymbol{w}^*\|^2 = \frac{1}{2}(\|\boldsymbol{w}_1\|^2 + \|\boldsymbol{w}_2\|^2) \geq \frac{1}{2}\|\boldsymbol{w}_1\|^2.$$

Hence by the optimality of $\boldsymbol{w}^*$, we conclude that $\boldsymbol{w}_2 = \boldsymbol{0}_m$. ■

The representation $\boldsymbol{w} = X\boldsymbol{\lambda}$ is called the *dual representation* of the normal vector. Substituting $X\boldsymbol{\lambda}$ for $\boldsymbol{w}$ yields another formulation of the primal hard margin problem $(\bar{H})$:

$$(\bar{H}) \quad \left| \begin{array}{ll} \text{minimize} & \dfrac{1}{2}\boldsymbol{\lambda}^\top K \boldsymbol{\lambda} \\ \text{subject to} & p_{\ell_i} + 1 \leq (\boldsymbol{k}^i)^\top \boldsymbol{\lambda} \leq p_{\ell_i+1} - 1 \quad (i \in N), \end{array} \right.$$

where $(\boldsymbol{k}^i)^\top = ((\boldsymbol{x}^i)^\top \boldsymbol{x}^1, (\boldsymbol{x}^i)^\top \boldsymbol{x}^2, \ldots, (\boldsymbol{x}^i)^\top \boldsymbol{x}^n)$ is the $i$th row of the matrix $K$. Since $n$ is typically by far larger than $m$, problem $(\bar{H})$ might be less interesting than problem $(H)$. However the fact that this formulation only requires the matrix $K$ will enable an application of the kernel technique to the problem.

Next we apply the kernel technique to the hard margin problem $(\bar{H})$ with the dual representation. The matrix $K$ in the problem $(\bar{H})$ is composed of the inner products $(\boldsymbol{x}^i)^\top \boldsymbol{x}^j$ for $i, j \in N$. This enables us to apply the kernel technique simply by replacing it by $\tilde{K}$. Denote the $i$th row of the matrix $\tilde{K}$ by $(\tilde{\boldsymbol{k}}^i)^\top$, then the problem to solve is

$$(\tilde{H}) \quad \left| \begin{array}{ll} \text{minimize} & \dfrac{1}{2}\boldsymbol{\lambda}^\top \tilde{K} \boldsymbol{\lambda} \\ \text{subject to} & p_{\ell_i} + 1 \leq (\tilde{\boldsymbol{k}}^i)^\top \boldsymbol{\lambda} \leq p_{\ell_i+1} - 1 \quad (i \in N). \end{array} \right.$$

Solving $(\tilde{H})$ to find $\boldsymbol{\lambda}^*$, an optimal normal vector $\tilde{\boldsymbol{w}}^* \in \mathbb{F}$ would be given as

$$\tilde{\boldsymbol{w}}^* = \sum_{i \in N} \lambda_i^* \tilde{\boldsymbol{x}}^i,$$

which is not available in general due to the absence of an explicit representation of $\tilde{\boldsymbol{x}}^i$'s. However, the value of $\langle \tilde{\boldsymbol{w}}^*, \tilde{\boldsymbol{x}} \rangle$ can be computed for the attribute vector $\boldsymbol{x} \in \mathbb{R}^n$ of a newly-arrived object in the following way:

$$\langle \tilde{\boldsymbol{w}}^*, \tilde{\boldsymbol{x}} \rangle = \langle \sum_{i \in N} \lambda_i^* \tilde{\boldsymbol{x}}^i, \tilde{\boldsymbol{x}} \rangle = \sum_{i \in N} \lambda_i^* \langle \tilde{\boldsymbol{x}}^i, \tilde{\boldsymbol{x}} \rangle = \sum_{i \in N} \lambda_i^* \langle \phi(\boldsymbol{x}^i), \phi(\boldsymbol{x}) \rangle = \sum_{i \in N} \lambda_i^* \kappa(\boldsymbol{x}^i, \boldsymbol{x}).$$

Then by locating the threshold interval determined by $\boldsymbol{p}^*$ into which this value falls, we can assign a label to the new object.

### 4.4.2 Dual representation for soft margin problem

Obviously we can replace $\|\boldsymbol{w}\|^2$ and $(\boldsymbol{x}^i)^\top \boldsymbol{w}$ in the primal problem given in the preceding subsection by $\boldsymbol{\lambda}^\top K \boldsymbol{\lambda}$ and $(\boldsymbol{k}^i)^\top \boldsymbol{\lambda}$, respectively, to obtain the primal problem with the dual representation of the normal vector. Take $(S_1)$ for instance, and we have

$$(\bar{S}_1) \quad \left| \begin{array}{ll} \text{maximize} & \dfrac{1}{2}\,\boldsymbol{\lambda}^\top K \boldsymbol{\lambda} + \dfrac{1}{2}\,c\,\mathbf{1}_n^\top (\boldsymbol{\xi}_- + \boldsymbol{\xi}_+) \\ \text{subject to} & p_{\ell_i} + 1 - \xi_{-i} \le (\boldsymbol{k}^i)^\top \boldsymbol{\lambda} \le p_{\ell_i+1} - 1 + \xi_{+i} \quad (i \in N) \\ & \boldsymbol{\xi}_-, \boldsymbol{\xi}_+ \ge \mathbf{0}_n. \end{array} \right.$$

The kernel technique can apply to the soft margin problems in the same way as discussed in Subsection 4.4.1. For the kernel version of soft margin problems with the dual representation of the normal vector, we have only to replace $K$ by $\tilde{K}$ given by some kernel function $\kappa$. Then the kernel version of $(\bar{S}_1)$ is given as

$$(\tilde{S}_1) \quad \left| \begin{array}{ll} \text{minimize} & \dfrac{1}{2}\,\boldsymbol{\lambda}^\top \tilde{K} \boldsymbol{\lambda} + \dfrac{1}{2}\,c\,\mathbf{1}_n^\top (\boldsymbol{\xi}_- + \boldsymbol{\xi}_+) \\ \text{subject to} & p_{\ell_i} + 1 - \xi_{-i} \le (\tilde{\boldsymbol{k}}^i)^\top \boldsymbol{\lambda} \le p_{\ell_i+1} - 1 + \xi_{+i} \quad (i \in N) \\ & \boldsymbol{\xi}_-, \boldsymbol{\xi}_+ \ge \mathbf{0}_n. \end{array} \right.$$

Since the reformulation with the dual representation as well as the kernel technique can apply to the soft margin problem $(S_2)$ with 2-norm penalty in the same manner as the above discussions, we omit the description of the problem $(S_2)$.

## 4.5 Row and Column Generation Algorithms

### 4.5.1 Algorithms for hard margin problem

We start with proposing an algorithm for the problem $(\bar{H})$ arising from separable instances. Note that the separability makes $(\bar{H})$ feasible. Since the dimension $m$ of the attribute vector is much smaller than the number $n$ of objects, we need a small number of attribute vectors for the dual representation, that is, a small number of $\lambda_i$'s are positive in the dual representation $\boldsymbol{w} = X\boldsymbol{\lambda}$. Furthermore it is likely that most of the constraints are redundant at an optimal solution of the problem, i.e., a small number of support vectors is expected. Then we propose to start the algorithm with a small number of attribute vectors

as $W$ and then increment it as the computation goes on. The sub-problem to solve is

$$(\bar{H}(W)) \quad \left| \begin{array}{ll} \text{minimize} & \frac{1}{2}\boldsymbol{\lambda}_W^\top K_W \boldsymbol{\lambda}_W \\ \text{subject to} & p_{\ell_i}+1 \le (\boldsymbol{k}_W^i)^\top \boldsymbol{\lambda}_W \le p_{\ell_i+1}-1 \quad (i \in W), \end{array} \right.$$

where $(\boldsymbol{k}_W^i)^\top$ is the row vector consisting $(\boldsymbol{x}^i)^\top \boldsymbol{x}^j$ for $j \in W$. Note that the dimension of $\boldsymbol{\lambda}_W$ varies when the size of $W$ changes as the computation goes on.

**Algorithm RC$\bar{\text{H}}$** (Row and Column Generation Algorithm for $(\bar{H})$)

Step 1 : Let $W^0$ be an initial working set, and let $\nu = 0$.

Step 2 : Solve $(\bar{H}(W^\nu))$ to obtain $\boldsymbol{\lambda}_{W^\nu}$ and $\boldsymbol{p}^\nu$.

Step 3 : Let $\Delta = \{\, i \in N \setminus W^\nu \mid (\boldsymbol{\lambda}_{W^\nu}, \boldsymbol{p}^\nu)$ violates $p_{\ell_i}+1 \le (\boldsymbol{k}_{W^\nu}^i)^\top \boldsymbol{\lambda}_W \le p_{\ell_i+1}-1 \,\}$.

Step 4 : If $\Delta = \emptyset$, terminate.

Step 5 : Otherwise choose $\Delta^\nu \subseteq \Delta$, let $W^{\nu+1} = W^\nu \cup \Delta^\nu$, increment $\nu$ by 1 and go to Step 2.

---

The following lemma shows that Algorithm RC$\bar{\text{H}}$ solves problem $(\bar{H})$ upon termination.

**Lemma 4.3** *Let $(\hat{\boldsymbol{\lambda}}_W, \hat{\boldsymbol{p}}) \in \mathbb{R}^{|W|+l}$ be an optimum solution of $(\bar{H}(W))$. If*

$$\hat{p}_{\ell_i}+1 \le (\boldsymbol{k}_W^i)^\top \hat{\boldsymbol{\lambda}}_W \le \hat{p}_{\ell_i+1}-1 \quad \textit{for all } i \in N \setminus W, \tag{4.11}$$

*then $(\hat{\boldsymbol{\lambda}}_W, \mathbf{0}_{N\setminus W}) \in \mathbb{R}^n$ together with $\hat{\boldsymbol{p}}$ forms an optimum solution of $(\bar{H})$.*

**Proof :** Note that $((\hat{\boldsymbol{\lambda}}_W, \mathbf{0}_{N\setminus W}), \hat{\boldsymbol{p}})$ is a feasible solution of $(\bar{H})$ since $(\boldsymbol{k}^i)^\top \begin{pmatrix} \hat{\boldsymbol{\lambda}}_W \\ \mathbf{0}_{N\setminus W} \end{pmatrix} = (\boldsymbol{k}_W^i)^\top \hat{\boldsymbol{\lambda}}_W$, $(\hat{\boldsymbol{\lambda}}_W, \hat{\boldsymbol{p}})$ is feasible to $(\bar{H}(W))$ and satisfies (4.11).

For an optimum solution $(\boldsymbol{\lambda}^*, \boldsymbol{p}^*)$ of $(\bar{H})$ let $\boldsymbol{w}^* = X\boldsymbol{\lambda}^*$, $\boldsymbol{w}_1$ be its orthogonal projection onto the range space of $X_W$ and $\boldsymbol{w}_2 = \boldsymbol{w}^* - \boldsymbol{w}_1$. Then $\boldsymbol{w}_1 = X_W \boldsymbol{\mu}_W^*$ for some $\boldsymbol{\mu}_W^* \in \mathbb{R}^{|W|}$ and

$$\frac{1}{2}(\boldsymbol{\lambda}^*)^\top K \boldsymbol{\lambda}^* = \frac{1}{2}\|\boldsymbol{w}^*\|^2 \ge \frac{1}{2}\|\boldsymbol{w}_1\|^2 = \frac{1}{2}(\boldsymbol{\mu}_W^*)^\top K_W \boldsymbol{\mu}_W^* \tag{4.12}$$

by the orthogonality between $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. For $i \in N \cap W$ it holds that

$$(\boldsymbol{k}_W^i)^\top \boldsymbol{\mu}_W^* = (\boldsymbol{x}^i)^\top X_W \boldsymbol{\mu}_W^* = (\boldsymbol{x}^i)^\top \boldsymbol{w}_1 = (\boldsymbol{x}^i)^\top (\boldsymbol{w}_1 + \boldsymbol{w}_2)$$
$$= (\boldsymbol{x}^i)^\top \boldsymbol{w}^* = (\boldsymbol{x}^i)^\top X \boldsymbol{\lambda}^* = (\boldsymbol{k}^i)^\top \boldsymbol{\lambda}^*,$$

which is between $p_{\ell_i}^* + 1$ and $p_{\ell_i+1}^* - 1$ since $(\boldsymbol{\lambda}^*, \boldsymbol{p}^*)$ is feasible to $(\bar{H})$. Then $(\boldsymbol{\mu}_W^*, \boldsymbol{p}^*)$ is feasible to $(\bar{H}(W))$. This and the optimality of $\hat{\boldsymbol{\lambda}}_W$ yield the inequality

$$\frac{1}{2}(\boldsymbol{\mu}_W^*)^\top K_W \boldsymbol{\mu}_W^* \geq \frac{1}{2}\hat{\boldsymbol{\lambda}}_W^\top K_W \hat{\boldsymbol{\lambda}}_W = \frac{1}{2}\begin{pmatrix} \hat{\boldsymbol{\lambda}}_W \\ \mathbf{0}_{N\setminus W} \end{pmatrix}^\top K \begin{pmatrix} \hat{\boldsymbol{\lambda}}_W \\ \mathbf{0}_{N\setminus W} \end{pmatrix}. \tag{4.13}$$

The two inequalities (4.12) and (4.13) prove the optimality of $((\hat{\boldsymbol{\lambda}}_W, \mathbf{0}_{N\setminus W}), \hat{\boldsymbol{p}})$. ■

Theorem 4.4 follows directly from the above lemma.

**Theorem 4.4** *The Algorithm RC$\bar{H}$ solves problem* $(\bar{H})$.

Next we present the algorithm for the kernel version of the problem $(\bar{H})$. In the same way as for the hard margin problem $(\bar{H})$ we consider the sub-problem

$$(\tilde{H}(W)) \quad \left| \begin{array}{ll} \text{minimize} & \frac{1}{2}\boldsymbol{\lambda}_W^\top \tilde{K}_W \boldsymbol{\lambda}_W \\ \text{subject to} & p_{\ell_i} + 1 \leq (\tilde{\boldsymbol{k}}_W^i)^\top \boldsymbol{\lambda}_W \leq p_{\ell_i+1} - 1 \quad (i \in W), \end{array} \right.$$

where $\tilde{K}_W$ is the sub-matrix consisting of the rows and columns of $\tilde{K}$ with indices in $W$, and $(\tilde{\boldsymbol{k}}_W^i)^\top$ is the row vector of $\kappa(\boldsymbol{x}^i, \boldsymbol{x}^j)$ for $j \in W$.

**Algorithm RC$\tilde{\text{H}}$** (Row and Column Generation Algorithm for $(\tilde{H})$)

Step 1 : Let $W^0$ be an initial working set, and let $\nu = 0$.

Step 2 : Solve $(\tilde{H}(W^\nu))$ to obtain $\boldsymbol{\lambda}_{W^\nu}$ and $\boldsymbol{p}^\nu$.

Step 3 : Let $\Delta = \{\, i \in N \setminus W^\nu \mid (\boldsymbol{\lambda}_{W^\nu}, \boldsymbol{p}^\nu)$ violates $p_{\ell_i} + 1 \leq (\tilde{\boldsymbol{k}}_{W^\nu}^i)^\top \boldsymbol{\lambda}_W \leq p_{\ell_i+1} - 1 \,\}$.

Step 4 : If $\Delta = \emptyset$, terminate.

Step 5 : Otherwise choose $\Delta^\nu \subseteq \Delta$, let $W^{\nu+1} = W^\nu \cup \Delta^\nu$, increment $\nu$ by 1 and go to Step 2.

---

The validity of the algorithm is straightforward from the following lemma, which is proved in exactly the same way as Lemma 4.3

**Lemma 4.5** *Let* $(\hat{\boldsymbol{\lambda}}_W, \hat{\boldsymbol{p}}) \in \mathbb{R}^{|W|+l}$ *be an optimum solution of* $(\tilde{H}(W))$. *If*

$$\hat{p}_{\ell_i} + 1 \leq (\tilde{\boldsymbol{k}}^i_W)^\top \hat{\boldsymbol{\lambda}}_W \leq \hat{p}_{\ell_i+1} - 1 \quad \textit{for all } i \in N \setminus W,$$

*then* $(\hat{\boldsymbol{\lambda}}_W, \mathbf{0}_{N\setminus W}) \in \mathbb{R}^n$ *together with* $\hat{\boldsymbol{p}}$ *forms an optimum solution of* $(\tilde{H})$.

**Theorem 4.6** *The Algorithm RC*$\tilde{H}$ *solves problem* $(\tilde{H})$.

### 4.5.2 Algorithms for soft margin problem

The algorithms for the soft margin problems may not differ substantially from those for the hard margin problems. The sub-problem $(\bar{S}_1(W))$ of $(\bar{S}_1)$ for the working set $W \subseteq N$ is

$$(\bar{S}_1(W)) \quad \left| \begin{array}{ll} \text{minimize} & \dfrac{1}{2}\boldsymbol{\lambda}_W^\top K_W \boldsymbol{\lambda}_W + \dfrac{1}{2} c\, \mathbf{1}_{|W|}^\top (\boldsymbol{\xi}_{-W} + \boldsymbol{\xi}_{+W}) \\ \text{subject to} & p_{\ell_i} + 1 - \xi_{-i} \leq (\boldsymbol{k}^i_W)^\top \boldsymbol{\lambda}_W \leq p_{\ell_i+1} - 1 + \xi_{+i} \quad (i \in W) \\ & \boldsymbol{\xi}_{-W},\ \boldsymbol{\xi}_{+W} \geq \mathbf{0}_{|W|}. \end{array} \right.$$

**Algorithm RC$\bar{\mathbf{S}}$** (Row and Column Generation Algorithm for $(\bar{S})$)

Step 1 : Let $W^0$ be an initial working set, and let $\nu = 0$.

Step 2 : Solve $(\bar{S}(W^\nu))$ to obtain $(\boldsymbol{\lambda}_{W^\nu}, \boldsymbol{p}^\nu, \boldsymbol{\xi}_{-W^\nu}, \boldsymbol{\xi}_{+W^\nu})$.

Step 3 : Let $\Delta = \{\, i \in N \setminus W^\nu \mid (\boldsymbol{\lambda}_{W^\nu}, \boldsymbol{p}^\nu)$ violates $p_{\ell_i} + 1 \leq (\boldsymbol{k}^i_{W^\nu})^\top \boldsymbol{\lambda}_W \leq p_{\ell_i+1} - 1 \,\}$.

Step 4 : If $\Delta = \emptyset$, terminate.

Step 5 : Otherwise choose $\Delta^\nu \subseteq \Delta$, let $W^{\nu+1} = W^\nu \cup \Delta^\nu$, increment $\nu$ by 1 and go to Step 2.

---

**Lemma 4.7** *Let* $(\hat{\boldsymbol{\lambda}}_W, \hat{\boldsymbol{p}}, \hat{\boldsymbol{\xi}}_{-W}, \hat{\boldsymbol{\xi}}_{+W})$ *be an optimum solution of* $(\bar{S}(W))$. *If*

$$\hat{p}_{\ell_i} + 1 \leq (\boldsymbol{k}^i_W)^\top \hat{\boldsymbol{\lambda}}_W \leq \hat{p}_{\ell_i+1} - 1 \quad \textit{for all } i \in N \setminus W,$$

*then* $((\hat{\boldsymbol{\lambda}}_W, \mathbf{0}_{N\setminus W}), \hat{\boldsymbol{p}}, (\boldsymbol{\xi}^\nu_{-W}, \mathbf{0}_{N\setminus W}), (\boldsymbol{\xi}^\nu_{+W}, \mathbf{0}_{N\setminus W}))$ *is an optimum solution of* $(\bar{S})$.

**Proof :** First note that $((\hat{\boldsymbol{\lambda}}_W, \mathbf{0}_{N\setminus W}), \hat{\boldsymbol{p}}, (\hat{\boldsymbol{\xi}}_{-W}, \mathbf{0}_{N\setminus W}), (\hat{\boldsymbol{\xi}}_{+W}, \mathbf{0}_{N\setminus W}))$ is feasible to $(\bar{S})$. Let $(\boldsymbol{\lambda}^*, \boldsymbol{p}^*, \boldsymbol{\xi}^*_{-W}, \boldsymbol{\xi}^*_{+W})$ be an optimum solution of $(\bar{S})$, let $\boldsymbol{w}^* = X\boldsymbol{\lambda}^*$ and $\boldsymbol{w}_1$ be its orthogonal projection onto the range space of $X_W$. Then we see that the coefficient vector $\boldsymbol{\mu}^*_W$ such that $\boldsymbol{w}_1 = X_W\boldsymbol{\mu}^*_W$ together with $(\boldsymbol{p}^*, \boldsymbol{\xi}^*_{-W}, \boldsymbol{\xi}^*_{+W})$ forms a feasible solution of $(\bar{S}(W))$, and

$$\frac{1}{2}\|\boldsymbol{w}^*\| \geq \frac{1}{2}\|\boldsymbol{w}_1\|.$$

Therefore in the same manner as Lemma 4.3, we obtain the desired result. ■

The validity of the algorithm directly follows the above lemma.

**Theorem 4.8** *The Algorithm RC$\bar{S}$ solves problem* $(\bar{S})$.

Similarly to the hard margin problem, we present the algorithm for the kernel version of the problem $(\bar{S}_1)$. In the same way as in the previous subsection we consider the sub-problem of $(\tilde{S}_1)$,which is given as

$$(\tilde{S}_1(W)) \quad \left| \begin{array}{ll} \text{minimize} & \frac{1}{2}\boldsymbol{\lambda}_W^\top \tilde{K}_W \boldsymbol{\lambda}_W + \frac{1}{2} c\, \mathbf{1}_{|W|}^\top (\boldsymbol{\xi}_{-W} + \boldsymbol{\xi}_{+W}) \\ \text{subject to} & p_{\ell_i} + 1 - \xi_{-i} \leq (\tilde{\boldsymbol{k}}^i_W)^\top \boldsymbol{\lambda}_W \leq p_{\ell_i+1} - 1 + \xi_{+i} \quad (i \in W) \\ & \boldsymbol{\xi}_{-W},\ \boldsymbol{\xi}_{+W} \geq \mathbf{0}_{|W|}. \end{array} \right.$$

**Algorithm RC$\tilde{\mathbf{S}}$** (Row and Column Generation Algorithm for $(\tilde{S})$)

Step 1 : Let $W^0$ be an initial working set, and let $\nu = 0$.

Step 2 : Solve $(\tilde{S}(W^\nu))$ to obtain $(\boldsymbol{\lambda}_{W^\nu}, \boldsymbol{p}^\nu, \boldsymbol{\xi}_{-W^\nu}, \boldsymbol{\xi}_{+W^\nu})$.

Step 3 : Let $\Delta = \{\, i \in N\setminus W^\nu \mid (\boldsymbol{\lambda}_{W^\nu}, \boldsymbol{p}^\nu)$ violates $p_{\ell_i} + 1 \leq (\tilde{\boldsymbol{k}}^i_{W^\nu})^\top \boldsymbol{\lambda}_W \leq p_{\ell_i+1} - 1 \,\}$.

Step 4 : If $\Delta = \emptyset$, terminate.

Step 5 : Otherwise choose $\Delta^\nu \subseteq \Delta$, let $W^{\nu+1} = W^\nu \cup \Delta^\nu$, increment $\nu$ by 1 and go to Step 2.

---

**Theorem 4.9** *The Algorithm RC$\tilde{S}$ solves problem* $(\tilde{S})$.

## 4.6 Illustrative Example

We show with a small instance how different models result in different classifications. The instance is the grades in calculus of 44 undergraduates. Each student is given one of the four possible grades $A, B, C$ and $D$ according to his/her total score of mid-term exam, end-of-term exam and a number of in-class quizzes. We take the scores of student's mid-term and end-of-term exams to form the attribute vector, and his/her grade as the label.

Since the score of quizzes is not considered as an attribute, the instance is not separable, hence the hard margin problem $(H)$ is infeasible. The solution of the soft margin problem $(S)$ with $c = 15$ is given in Figure 4.1, where the students of different grades are loosely separated by three straight lines. Each value on the lines in the figure represents the corresponding threshold $p_k$.

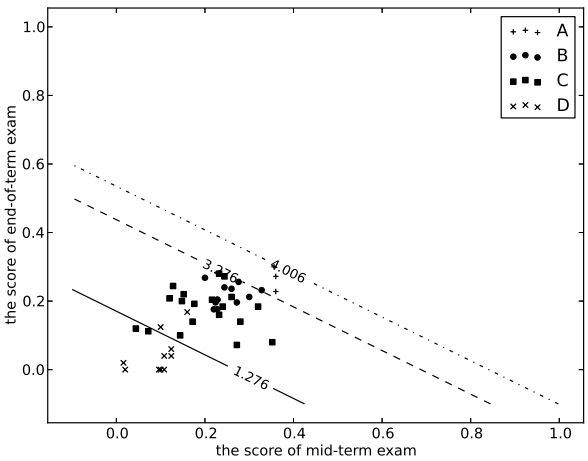



Figure 4.1: Classification by $(S)$

Using the following two kernel functions defined as

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{y}\|^2\right) \quad \text{(Gaussian kernel)},$$

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x}^\top \boldsymbol{y})^d \quad \text{(Polynomial kernel)}$$

with several different values of $\sigma$ and $d$, we solved $(\tilde{S})$. The result of the Gaussian kernel with $c = 10$ and $\sigma = 0.5$ is given in Figure 4.2, where one can observe that the problem $(\tilde{S})$ with the Gaussian kernel is exposed to the risk of over-fitting. This issue will be discussed in Section 4.8. The result of the polynomial kernel with $c = 15$ and $d = 4$ is

given in Figure 4.3. From Figure 4.3, we observe that the students of different grades are separated by three gentle curves.

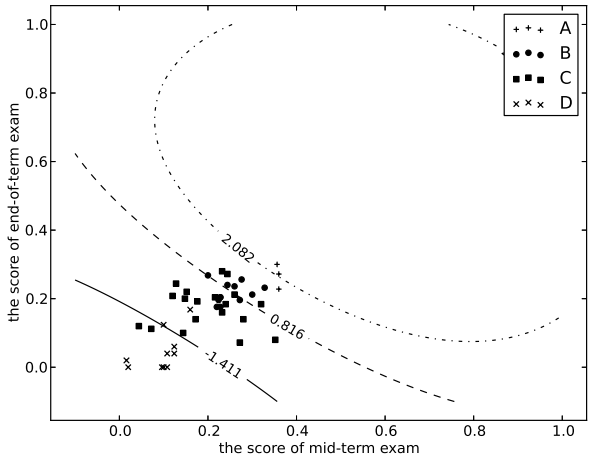



Figure 4.2: Classification by $(\tilde{S})$ with the Gaussian kernel

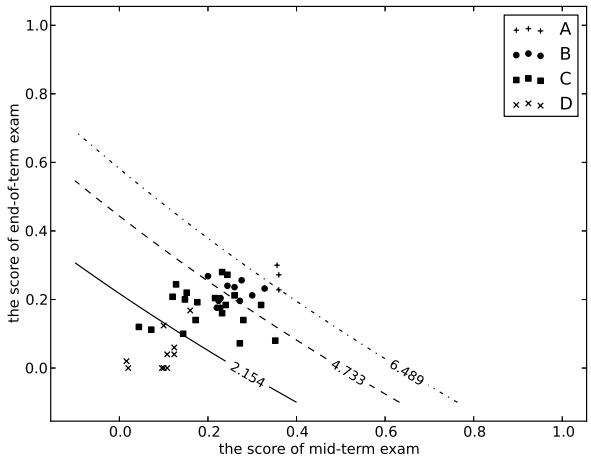



Figure 4.3: Classification by $(\tilde{S})$ with the polynomial kernel

## 4.7 Computational Experiments

We report on the computational experiments with our proposed algorithms. Implementing the algorithms in Python 2.7, using Gurobi 6.0.0 as a QP solver, we performed the

experiments on a PC with Intel Core i7, 3.70 GHz processor and 32.0 GB of memory.

The instances we tested were randomly generated and fall into two types: separable instances and non-separable instances. First, we generate $n$ attribute vectors $\boldsymbol{x}^i = (x_1^i, x_2^i)$ of two dimension, each component of which is drawn uniformly from the unit interval $[0, 1]$. Then object $i$ is assigned the label $\ell_i$ defined as

$$\ell_i = \max_{\ell \in L} \{\, \ell \in L \mid x_1^i + x_2^i > p_\ell \,\}$$

for the fixed thresholds $(p_0, p_1, p_2, p_3) = (-\infty, 0.5, 1.0, 1.5)$. The instances thus generated are of the first type, i.e., separable. Non-separable instances are generated by altering the labels of objects. Namely, adding a random noise to each element of the attribute vector to make a perturbed attribute vector $(x_1^i + \varepsilon_1^i, x_2^i + \varepsilon_2^i)$, where $\varepsilon_1^i$ and $\varepsilon_2^i$ follow the normal distribution with a zero mean and a standard deviation of $0.03$, we give the label $\ell_i$ to object $i$ according to the sum $x_1^i + \varepsilon_1^i + x_2^i + \varepsilon_2^i$ instead of $x_1^i + x_2^i$. Due to the presence of the random noise, the instances thus generated are not necessarily separable.

We generate five datasets for each instance with several different numbers of objects since the results may change owing to the random variables used in the instance generation. We name the separable type dataset (resp., the non-separable dataset) "S.$n$.q" (resp., "NS.$n$.q"), where $n$ is the number of objects and q is the dataset ID.

We solved the separable instances by the algorithm RC$\tilde{\text{H}}$ and the non-separable instances by RC$\tilde{\text{S}}$ with $c = 10$. In all experiments we used the polynomial kernel with $d = 4$. To make the initial working set $W^0$, we collect three objects for each label that have the highest, the lowest and the median values of $x_1^i + x_2^i$ among the objects assigned the same label. At Step 5 in the algorithms, we add to the current working set $W^\nu$ at most two objects corresponding to the most violated constraints at $(\boldsymbol{\lambda}_{W^\nu}, \boldsymbol{p}^\nu)$, more precisely, we add the objects $i$ and $j \in N \setminus W^\nu$ such that

$$i = \text{argmax} \left\{ 1 - (\tilde{\boldsymbol{k}}_{W^\nu}^i)^\top \boldsymbol{\lambda}_{W^\nu} + p_{\ell_i}^\nu > 0 \mid i \in N \setminus W^\nu \right\},$$
$$j = \text{argmax} \left\{ 1 + (\tilde{\boldsymbol{k}}_{W^\nu}^i)^\top \boldsymbol{\lambda}_{W^\nu} - p_{\ell_i+1}^\nu > 0 \mid i \in N \setminus W^\nu \right\}.$$

Table 4.1 shows the computational results of applying RC$\tilde{\text{H}}$ (resp., RC$\tilde{\text{S}}$) to separable instances (resp., non-separable instances), where the columns "# iter.", "# added obj." and "time" represent the number of sub-problems solved, the number of added objects and the computation time in seconds, respectively. In order to assess the efficiency of

our algorithm, we added the column "GRB" showing the computation time when the whole problems $(\tilde{H})$ and $(\tilde{S})$ were directly input and solved by Gurobi, and the column "PRank" showing the computation time of applying PRank algorithm [31], which is an on-line learning algorithm motivated by the perceptron. The entries "ave." and "st.dev." show the average and the standard deviation across the five datasets.

We begin by looking at the results for the separable instances. From Table 4.1, we observe that the number of added objects is much smaller than that of the original problem. Specifically, only from 0.4% to 8.6% of the whole set of variables of the problems suffices in order to obtain an optimal solution. Thus RC$\tilde{\text{H}}$ requires a small memory capacity, and it helps RC$\tilde{\text{H}}$ be applicable to further larger instances. Gurobi Optimizer for solving $(\tilde{H})$ requires a rather long computation time as the instance size grows. To be specific, Gurobi takes over $10{,}000$ seconds, approximately 2.8 hours, on average to solve the instances with $n = 10{,}000$, while our algorithm takes less than $300$ seconds on average. Concerning PRank algorithm, it also requires a long computation time for large instances since the algorithm needs to compute and store an $n$-dimensional row vector of $\tilde{K}$ at every iteration. From these observations, RC$\tilde{\text{H}}$ is superior to applying Gurobi and PRank directly to the whole problem in terms of both computation time and memory consumption.

Turning now to the results for the non-separable instances, we observe that the number of iterations as well as the number of added objects is larger than that for the separable instances. Nevertheless the number of added objects is approximately 20% of the whole set of variables, that is, RC$\tilde{\text{S}}$ also requires a small memory capacity. In contrast, RC$\tilde{\text{S}}$ takes much longer computation time than applying Gurobi and PRank directly to the whole problem. This drawback is due to the way of our implementation. Whenever a working set $W$ is updated in our algorithm, we generate a sub-problem corresponding to $W$ by adding not only constraints but also variables as well. The vast majority of computation time was spent for Gurobi to carry out sub-problem generation repeatedly every time when the working set was updated. Take "NS.$1000$.5" for instance, we give a breakdown of the computation time at each iteration in Figure 4.4. The legends in the figure are as follows: "GenTime" is the time of generating a sub-problem, "RunTime" is the time of solving a sub-problem, and "FeasTime" is the time required for searching violated constraints at Step 3. We observe that the predominant portion of the computation time was devoted to the problem generation and the run time and feasibility checking time

Table 4.1: Computational results

| instance | $n$ | # iter. | | # added obj. | | time (sec.) | | GRB (sec.) | | PRank (sec.) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ave. | st.dev. | ave. | st.dev. | ave. | st.dev. | ave. | st.dev. | ave. | st.dev. |
| S.100.1 – S.100.5 | 100 | 5.800 | 1.789 | 8.600 | 3.050 | 0.336 | 0.174 | 0.004 | 0.000 | 0.028 | 0.001 |
| S.500.1 – S.500.5 | 500 | 8.200 | 1.643 | 13.600 | 2.702 | 1.018 | 0.313 | 0.099 | 0.006 | 0.562 | 0.032 |
| S.1000.1 – S.1000.5 | 1000 | 12.200 | 0.837 | 20.800 | 1.095 | 3.342 | 0.353 | 0.499 | 0.015 | 2.644 | 0.259 |
| S.5000.1 – S.5000.5 | 5000 | 19.400 | 3.975 | 35.600 | 7.301 | 65.724 | 14.942 | 1139.662 | 389.168 | 207.425 | 1.027 |
| S.10000.1 – S.10000.5 | 10000 | 23.400 | 6.189 | 42.600 | 11.349 | 286.731 | 80.009 | 10639.340 | 3839.160 | 1472.864 | 2.497 |
| NS.100.1 – NS.100.5 | 100 | 16.400 | 7.403 | 25.200 | 6.058 | 2.592 | 1.859 | 0.022 | 0.002 | 0.027 | 0.002 |
| NS.500.1 – NS.500.5 | 500 | 65.600 | 31.777 | 109.800 | 24.448 | 107.200 | 106.311 | 1.601 | 0.066 | 0.544 | 0.004 |
| NS.1000.1 – NS.1000.5 | 1000 | 108.400 | 39.855 | 193.600 | 36.184 | 493.781 | 444.807 | 15.470 | 4.984 | 2.462 | 0.018 |



Table 4.2: Average rank losses

| instance | $n$ | Our algorithm | | PRank | |
|---|---|---|---|---|---|
| | | ave. | st.dev. | ave. | st.dev. |
| S.100.1 – S.100.5 | 100 | 0.000 | 0.000 | 0.458 | 0.390 |
| S.500.1 – S.500.5 | 500 | 0.000 | 0.000 | 0.250 | 0.166 |
| S.1000.1 – S.1000.5 | 1000 | 0.000 | 0.000 | 0.244 | 0.180 |
| S.5000.1 – S.5000.5 | 5000 | 0.000 | 0.000 | 0.101 | 0.045 |
| S.10000.1 – S.10000.5 | 10000 | 0.000 | 0.000 | 0.065 | 0.023 |
| NS.100.1 – NS.100.5 | 100 | 0.058 | 0.015 | 0.292 | 0.060 |
| NS.500.1 – NS.500.5 | 500 | 0.059 | 0.009 | 0.192 | 0.088 |
| NS.1000.1 – NS.1000.5 | 1000 | 0.063 | 0.007 | 0.260 | 0.255 |

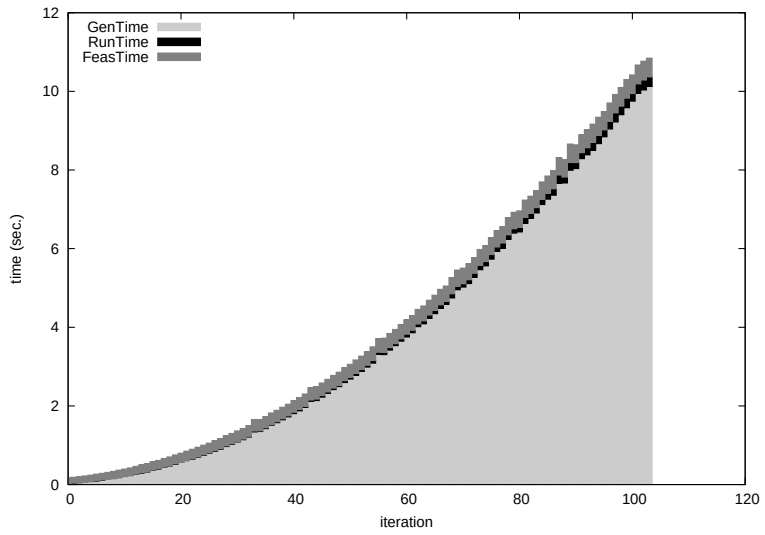



Figure 4.4: Breakdown of the computation time for "NS.1000.5"

together are fringe. This suggests that a finely tuned application of a QP solver could dramatically reduce the total computation time. This would merit further research.

Next, we compare our algorithms with PRank algorithm with respect to the accuracy of solutions. We use the following average rank loss as a measure of accuracy:

$$\frac{1}{n}\sum_{i\in N}|\hat{\ell}_i - \ell_i|,$$

where $\hat{\ell}_i$ is a label predicted from the solution $(\boldsymbol{\lambda}^*, \boldsymbol{p}^*)$ obtained by the algorithms. Namely, $\hat{\ell}_i = \max_{\ell\in L}\{\, \ell \in L \mid \sum_{j\in N} \lambda_j^* \kappa(\boldsymbol{x}^i, \boldsymbol{x}^j) > p_\ell^* \,\}$. Since a positive value of the average rank loss indicates misclassification, it is preferable to be close to zero. Table 4.2 summarizing the average rank losses of our algorithms and PRank algorithm shows that our algorithms outperform PRank algorithm in all instances. As a matter of course, we confirm that the average rank losses obtained by our algorithm are zero for the separable instances.

## 4.8 Monotonicity Issue

In some situations it would be desirable that the separating curves have some monotonicity property, namely an object with attribute vector $\boldsymbol{x}$ be ranked higher than an object with $\boldsymbol{y}$ such that $\boldsymbol{y} \leq \boldsymbol{x}$.

Let $P$ be a hyperplane in $\mathbb{F}$ defined by

$$P = \{\, \tilde{\boldsymbol{x}} \in \mathbb{F} \mid \langle \tilde{\boldsymbol{w}}^*, \tilde{\boldsymbol{x}} \rangle = b \,\}$$

for some constant $b \in \mathbb{R}$ and let $C$ denote its inverse image under the unknown function $\phi$, i.e.,

$$C = \{\, \boldsymbol{x} \in \mathbb{R}^m \mid \phi(\boldsymbol{x}) \in P \,\}.$$

Then $\boldsymbol{x} \in C$ if and only if $\langle \tilde{\boldsymbol{w}}^*, \phi(\boldsymbol{x}) \rangle = b$. Since $\tilde{\boldsymbol{w}}^* = \sum_{i \in N} \lambda_i^* \tilde{\boldsymbol{x}}^i = \sum_{i \in N} \lambda_i^* \phi(\boldsymbol{x}^i)$, we obtain

$$\langle \sum_{i \in N} \lambda_i^* \phi(\boldsymbol{x}^i), \phi(\boldsymbol{x}) \rangle = b.$$

Due to the bi-linearity of the inner product $\langle \cdot, \cdot \rangle$, we have

$$\langle \sum_{i \in N} \lambda_i^* \phi(\boldsymbol{x}^i), \phi(\boldsymbol{x}) \rangle = \sum_{i \in N} \lambda_i^* \langle \phi(\boldsymbol{x}^i), \phi(\boldsymbol{x}) \rangle = \sum_{i \in N} \lambda_i^* \kappa(\boldsymbol{x}^i, \boldsymbol{x}),$$

and then an expression of the inverse image

$$C = \{\, \boldsymbol{x} \in \mathbb{R}^m \mid \sum_{i \in N} \lambda_i^* \kappa(\boldsymbol{x}^i, \boldsymbol{x}) = b \,\}$$

by the kernel function $\kappa$.

Suppose that the kernel function $\kappa(\boldsymbol{x}^i, \cdot)$ is nondecreasing for $i \in N$, in the sense that

$$\boldsymbol{x} \leq \boldsymbol{x}' \;\Rightarrow\; \kappa(\boldsymbol{x}^i, \boldsymbol{x}) \leq \kappa(\boldsymbol{x}^i, \boldsymbol{x}'),$$

and $\lambda_i^* \geq 0$ for $i \in N$. Then $\sum_{i \in N} \lambda_i^* \kappa(\boldsymbol{x}^i, \boldsymbol{x})$ is nondecreasing as a whole.

**Lemma 4.10** *The kernel function $\kappa(\boldsymbol{x}^i, \cdot)$ is nondecreasing and $\lambda_i^* \geq 0$ for $i \in N$. Then the contours are nondecreasing.*

The polynomial kernel

$$\kappa(\boldsymbol{x}^i, \boldsymbol{x}) = (1 + (\boldsymbol{x}^i)^\top \boldsymbol{x})^d$$

is nondecreasing with respect to $\boldsymbol{x}$ if $\boldsymbol{x}^i \geq \boldsymbol{0}$ for $i \in N$. Therefore it would be appropriate to use the polynomial kernel when all the attribute vectors are nonnegative including those

of potential objects, and the monotonicity is desirable. In this case the kernel hard margin problem $(\tilde{H})$ should be added non-negativity constraints of variables $\lambda_i$'s:

$$(\tilde{H}_+) \quad \left| \begin{array}{ll} \text{minimize} & \dfrac{1}{2}\boldsymbol{\lambda}^\top \tilde{K}\boldsymbol{\lambda} \\ \text{subject to} & p_{\ell_i} + 1 \leq (\tilde{\boldsymbol{k}}^i)^\top \boldsymbol{\lambda} \leq p_{\ell_i+1} - 1 \quad (i \in N) \\ & \boldsymbol{\lambda} \geq \mathbf{0}. \end{array} \right.$$

The problem remains an ordinary convex quadratic optimization, and the additional non-negativity constraints do not make it more difficult to solve.

# Chapter 5

# Concluding Remarks

In this thesis, we have considered three problems in data mining: the modularity maximization, the modularity density maximization, and the ranking problem. When we are faced with difficult optimization problems, it would be an effective way to consider their relaxation problems. Relaxation technique contributes not only to ease inherent difficulties of the problems but also to reduce memory consumption. With this intention, we have proposed solution methods based on relaxation for the three problems. We summarize the results obtained and point out further possible studies for each problem.

## 5.1 Modularity Maximization

Based on the set partitioning formulation of the modularity maximization, we proposed the algorithms: LP relaxation-based and Lagrangian relaxation-based. In order to alleviate the computational burden, we applied a cutting plane method and a column generation method to both of LP and Lagrangian relaxation problems. One of the advantages of the algorithms is that they are able to provide the upper bounds on the optimal modularity by solving a small sub-problem of the original problem. This bounding procedure enables us to evaluate the quality of the optimal value of the sub-problem at every iteration of the algorithm.

Through the computational experiments, we observed that the Lagrangian relaxation-based algorithm required much fewer iterations than the LP relaxation-based algorithm. The method of multiple cuts or multiple columns that we proposed also performs fairly well. This method could apply to other clustering problems of finding a partition of a

given set, however, it should need further investigation from both theoretical and computational viewpoints. The computational results also showed that both upper bound and lower bound obtained by our algorithms were significantly close to the optimal value. Indeed, the algorithms solved most of the instances we tested to optimality, which was proved by the vanishing gap between the upper and lower bounds.

A direction of further research is to incorporate a heuristics providing an initial partition into our algorithms. The exact algorithm [3], for the modularity maximization, first finds a partition by Noack and Rotta's heuristics [79] before starting the column generation process, and uses members of the partition as an initial set of columns. This scheme would be expected to reduce the number of iterations of our algorithms.

## 5.2 Modularity Density Maximization

In Chapter 3, for the modularity density maximization problem, we presented 0-1SDP formulation which was originally introduced by Peng and Xia [80] for minimum sum-of-squares clustering problem. We showed that 0-1SDP is equivalent to the modularity density maximization, and proposed to solve a doubly non-negative relaxation of the problem 0-1SDP in order to obtain an upper bound on the modularity density. The advantage of the relaxation problem is twofold: the problem does not require the optimal number of communities be known in advance, and the size of the problem depends on neither the number of communities nor the number of edges of the underlying network. In addition, we developed a lower bounding algorithm based on the combination of spectral heuristics and dynamic programming. Our formulation was numerically compared to MILP formulation, with the results that it provides upper bounds competitive with MILP formulation.

The future work addresses the following two issues. The first issue is how to strengthen the upper bound obtained by the relaxation problem. To this end, one of the possible ways is to add valid inequalities. It is well known that the triangle inequalities are effective as the valid inequalities for several clustering problems, while the number of them grows rapidly with the number of nodes. Since most of the inequalities however would be likely redundant at an optimal solution, we need only a small number of them. Hence, it should need a well-organized strategy of selecting a small fraction of inequalities via a preprocessing which utilizes information obtained from the original network.

The second issue arises from the phase of permuting the rows and columns of the solution matrix. We observed that the solution matrix showed a block-diagonal-like structure when its rows and columns are rearranged simultaneously in accordance with the magnitude of the components of the eigenvector corresponding to the second largest eigenvalue. Theoretical study should be carried out about whether this procedure functions effectively, and why if it does. The alternative to form a block-diagonal-like matrix is to develop a heuristics based on the numerical linear algebraic computation such as the algorithms of Sargent and Westerberg [81], and Tarjan [86].

## 5.3 Ranking Problem

In Chapter 4, we proposed to apply the dual representation of the normal vector to the ranking problem based on the fixed margin strategy by Shashua and Levin [83]. The problem obtained has the drawback that it has the number of variables as well as constraints being proportional to the number of objects unlike the case of the original formulation by Shashua and Levin [83]. However it enables an application of the kernel technique, which outweighs the drawback. We proposed a row and column generation algorithm in order to alleviate the computational burden. Furthermore, we showed that the algorithm solves the problem to optimality within a finite number of iterations.

Through some preliminary experiments, the number of both variables and constraints added in our proposed algorithm are much smaller than those of original problem, thus our algorithm requires a small memory capacity. However it should need further research such as the setting of the initial working set and the choice of variables and constraints to add at every iteration of the algorithm since a clever choice of these may enhance the efficiency of the algorithm. In addition, we should clarify the class of kernel functions which satisfies the nondecreasing property discussed in Section 4.8.

# Appendix

## A Dual of Soft Margin Problem $(S_2)$

The Lagrangian function for the problem $(S_2)$ is

$$L(\boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\xi}_-, \boldsymbol{\xi}_+, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{w}\boldsymbol{w}^\top - (X(\boldsymbol{\alpha} - \boldsymbol{\beta}))^\top \boldsymbol{w} + (A\boldsymbol{\alpha} - B\boldsymbol{\beta})^\top \boldsymbol{p} + \boldsymbol{\alpha}^\top \mathbf{1}_n + \boldsymbol{\beta}^\top \mathbf{1}_n + (\frac{1}{2}c\,\boldsymbol{\xi}_- - \boldsymbol{\alpha})^\top \boldsymbol{\xi}_- + (\frac{1}{2}c\,\boldsymbol{\xi}_+ - \boldsymbol{\beta})^\top \boldsymbol{\xi}_+,$$

and the Lagrangian relaxation problem for a given $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathbb{R}^{2n}_+$ is

$$\left| \begin{array}{ll} \text{minimize} & L(\boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\xi}_-, \boldsymbol{\xi}_+, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\ \text{subject to} & (\boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\xi}_-, \boldsymbol{\xi}_+) \in \mathbb{R}^{m+l+2n}. \end{array} \right.$$

Since $L$ is a convex function with respect to $(\boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\xi}_-, \boldsymbol{\xi}_+)$, the optimality condition of $(\boldsymbol{w}^*, \boldsymbol{p}^*, \boldsymbol{\xi}_-^*, \boldsymbol{\xi}_+^*)$ for a given Lagrangian multiplier vector $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is

$$\frac{\partial L}{\partial(\boldsymbol{w}, \boldsymbol{p}, \boldsymbol{\xi}_-, \boldsymbol{\xi}_+)}(\boldsymbol{w}^*, \boldsymbol{p}^*, \boldsymbol{\xi}_-^*, \boldsymbol{\xi}_+^*) = \mathbf{0}_{m+l+2n},$$

which reduces to

$$\boldsymbol{w}^* = X(\boldsymbol{\alpha} - \boldsymbol{\beta}), \tag{5.1}$$

$$A\boldsymbol{\alpha} - B\boldsymbol{\beta} = \mathbf{0}_l, \tag{5.2}$$

$$c\,\boldsymbol{\xi}_-^* - \boldsymbol{\alpha} = c\,\boldsymbol{\xi}_+^* - \boldsymbol{\beta} = \mathbf{0}_n. \tag{5.3}$$

Substituting (5.1), (5.2) and (5.3) for $L$, we obtain the optimal value $\omega(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of the Lagrangian relaxation problem

$$\omega(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\left(\frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\beta})^\top K(\boldsymbol{\alpha} - \boldsymbol{\beta}) - \mathbf{1}_n^\top(\boldsymbol{\alpha} + \boldsymbol{\beta}) + \frac{1}{2c}(\boldsymbol{\alpha}^\top\boldsymbol{\alpha} + \boldsymbol{\beta}^\top\boldsymbol{\beta})\right).$$

Then the Lagrangian dual problem of $(S_2)$ forms the following $(dS_2)$:

$$
(dS_2) \quad \left| \begin{array}{ll} \text{minimize} & \frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\beta})^\top K(\boldsymbol{\alpha}-\boldsymbol{\beta}) + \frac{1}{2c}(\boldsymbol{\alpha}^\top\boldsymbol{\alpha}+\boldsymbol{\beta}^\top\boldsymbol{\beta}) - \mathbf{1}_n^\top(\boldsymbol{\alpha}+\boldsymbol{\beta}) \\ \text{subject to} & A\boldsymbol{\alpha} - B\boldsymbol{\beta} = \mathbf{0}_l \\ & \boldsymbol{\alpha}_{N(0)} = \mathbf{0}_{n(0)} \\ & \boldsymbol{\alpha}_{N(l)} = \mathbf{0}_{n(l)} \\ & \boldsymbol{\alpha} \geq \mathbf{0}_n \\ & \boldsymbol{\beta} \geq \mathbf{0}_n. \end{array} \right.
$$

# Bibliography


[1] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] G. Agarwal and D. Kempe. Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B*, 66(3):409–418, 2008.

[3] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, S. Perron, and L. Liberti. Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, 82(4):046112, 2010.

[4] E. Amaldi, S. Coniglio, and S. Gualandi. Coordinated cutting plane generation via multi-objective separation. *Mathematical Programming*, 143(1):87–110, 2014.

[5] A. Arenas, A. Fernández, and S. Gómez. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(5):053039, 2008.

[6] E. Balas and M.C. Carrera. A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research*, 44(6):875–890, 1996.

[7] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58(1-3):295–324, 1993.

[8] E. Balas and A. Ho. Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study. *Mathematical Programming*, 12:37–60, 1980.

[9] J.E. Beasley. An algorithm for set covering problem. *European Journal of Operational Research*, 31(1):85–93, 1987.

[10] J.E. Beasley. A Lagrangian heuristic for set-covering problems. *Naval Research Logistics*, 37(1):151–164, 1990.

[11] J.E. Beasley and P.C. Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):392–404, 1996.

[12] J.E. Beasley and K. Jörnsten. Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58(2):293–300, 1992.

[13] P. Benchimol, G. Desaulniers, and J. Desrosiers. Stabilized dynamic constraint aggregation for solving set partitioning problems. *European Journal of Operational Research*, 223(2):360–371, 2012.

[14] H.P. Benson. Global optimization algorithm for the nonlinear sum of ratios problem. *Journal of Optimization Theory and Applications*, 112(1):1–29, 2002.

[15] C.M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[16] R.E. Bixby, J.W. Gregory, I.J. Lustig, R.E. Marsten, and D.F. Shanno. Very large-scale linear programming: a case study in combining interior point and simplex methods. *Operations Research*, 40(5):885–897, 1992.

[17] P.S. Bradley, U.M. Fayyad, and O.L. Mangasarian. Mathematical programming for data mining: formulations and challenges. *INFORMS Journal on Computing*, 11(3):217–238, 1999.

[18] M. Bramer. *Principles of data mining*. Springer, 2007.

[19] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[20] S. Cafieri, P. Hansen, and L. Liberti. Locally optimal heuristic for modularity maximization of networks. *Physical Review E*, 83(5):056105, 2011.

[21] A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. *Operations Research*, 47(5):730–743, 1999.

[22] A. Caprara, P. Toth, and M. Fischetti. Algorithms for the set covering problem. *Annals of Operations Research*, 98(1):353–371, 2000.

[23] S. Ceria, P. Nobili, and A. Sassano. A Lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming*, 81(2):215–228, 1998.

[24] C-C. Chen and S-T. Li. Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications*, 41(16):7235–7247, 2014.

[25] M. Chen, K. Kuzmin, and B.K. Szymanski. Extension of modularity density for overlapping community structure. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 856–863. IEEE, 2014.

[26] A. Clauset, M.E. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[27] D. Cossock and T. Zhang. Subset ranking using regression. In G. Lugosi and H. Simon, editors, *Learning Theory*, volume 4005 of *Lecture Notes in Computer Science*, pages 605–619. Springer, 2006.

[28] A. Costa. Some remarks on modularity density. *arXiv preprint arXiv:1409.4063*, 2014.

[29] A. Costa. MILP formulations for the modularity density maximization problem. *European Journal of Operational Research*, 245(1):14–21, 2015.

[30] A. Costa, S. Kushnarev, L. Liberti, and Z. Sun. Divisive heuristic for modularity density maximization. Optimization Online, 2015. `http://www.optimization-online.org/DB_HTML/2015/09/5116.html`.

[31] K. Crammer and Y. Singer. Pranking with ranking. In *Proceedings Advances in Neural Information Procession Systems (NIPS)*, pages 641–647, 2001.

[32] G. Desaulniers, J. Desrosiers, and M.M. Solomon. *Column generation*. Springer, 2005.

[33] C.H.Q. Ding, X. He, H. Zha, M. Gu, and H.D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings IEEE International Conference on Data Mining (ICDM)*, pages 107–114. IEEE, 2001.

[34] T.N. Dinh and M.T. Thai. Towards optimal community detection: From trees to general weighted networks. *Internet Mathematics*, 11(3):181–200, 2015.

[35] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967.

[36] M. Doumpos and C. Zopounidis. Monotonic support vector machines for credit risk rating. *New Mathematics and Natural Computation*, 5(3):557–570, 2009.

[37] O. Du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1):229–237, 1999.

[38] I. Elhallaoui, D. Villeneuve, F. Soumis, and G. Desaulniers. Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research*, 53(4):632–645, 2005.

[39] S. Elhedhli and J.L. Goffin. The integration of an interior-point cutting plane method within a branch-and-price algorithm. *Mathematical Programming*, 100(2):267–294, 2004.

[40] M.L. Fisher. The Lagrangian relaxation method for solving integer programming problem. *Management Science*, 27(1):1–18, 2004.

[41] R. Fortet. Applications de lalgebre de boole en recherche opérationelle. *Revue Française de Recherche Opérationelle*, 4(14):17–26, 1960.

[42] S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[43] M. Gendreau and J.Y. Potvin. *Handbook of metaheuristics*. Springer, 2010.

[44] A.M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Studies*, 2:82–114, 1974.

[45] J. Gil-Mendieta and S. Schmidt. The political network in mexico. *Social Networks*, 18(4):355–381, 1996.

[46] M. Girvan and M.E. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[47] B.H. Good, Y.A. de Montjoye, and A. Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.

[48] C. Granell, S. Gomez, and A. Arenas. Hierarchical multiresolution method to overcome the resolution limit in complex networks. *International Journal of Bifurcation and Chaos*, 22(07):1250171, 2012.

[49] M. Grötschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45(1):59–96, 1989.

[50] M. Grötschel and Y. Wakabayashi. Facets of the clique partitioning polytope. *Mathematical Programming*, 47(1):367–387, 1990.

[51] R. Guimerá and L.A Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005.

[52] S. Haddadi. Simple Lagrangian heuristic for the set covering problem. *European Journal of Operational Research*, 97(1):200–204, 1997.

[53] M. Held, P. Wolfe, and H.P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6(1):62–88, 1974.

[54] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

[55] Y. Izunaga, K. Sato, K. Tatsumi, and Y. Yamamoto. Row and column generation algorithms for minimum margin maximization of ranking problems. *Journal of the Operations Research Society of Japan*, 58(4):394–409, 2015.

[56] J.E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial & Applied Mathematics*, 8(4):703–712, 1960.

[57] D.E. Knuth. *The Stanford GraphBase: A platform for combinatorial computing*. Addison-Wesley Reading, 1993.

[58] V. Krebs. `http://www.orgnet.com/`.(last visited october 13, 2015.).

[59] T. Kuno. A branch-and-bound algorithm for maximizing the sum of several linear ratios. *Journal of Global Optimization*, 22(1-4):155–174, 2002.

[60] A. Lancichinetti and S. Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6):066122, 2011.

[61] E.A. Leicht and M.E. Newman. Community structure in directed networks. *Physical Review Letters*, 100(11):118703, 2008.

[62] P. Li, Q. Wu, and C.J. Burges. McRank: Learning to rank using multiple classification and gradient boosting. In *Proceedings Advances in Neural Information Procession Systems (NIPS)*, pages 897–904, 2007.

[63] Z. Li, S. Zhang, R.S. Wang, X.S. Zhang, and L. Chen. Quantitative function for community detection. *Physical Review E*, 77(3):036109, 2008.

[64] T-Y. Liu. *Learning to rank for information retrieval*. Springer, 2011.

[65] M.E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[66] D. Lusseau, K. Schneider, O.J. Boisseau, P. Haase, E. Slooten, and S.M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[67] G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part i convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976.

[68] J.H. Michael. Labor dispute reconciliation in a forest products manufacturing facility. *Forest Products Journal*, 47(11/12):41–45, 1997.

[69] J.H. Michael and J.G. Massey. Modeling the communication network in a sawmill. *Forest Products Journal*, 47(9):25–30, 1997.

[70] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon. Superfamilies of evolved and designed networks. *Science*, 303(5663):1538–1542, 2004.

[71] J. E. Mitchell. Cutting plane methods and subgradient methods. *Tutorials in Operations Research*, pages 34–61, 2009.

[72] A. Miyauchi and Y. Miyamoto. Computing an upper bound of modularity. *The European Physical Journal B*, 86(7):302, 2013.

[73] A. Miyauchi and N. Sukegawa. Redundant constraints in the standard formulation for the clique partitioning problem. *Optimization Letters*, 9(1):199–207, 2015.

[74] S. Muff, F. Rao, and A. Caflisch. Local modularity measure for network clusterizations. *Physical Review E*, 72(5):056107, Nov 2005.

[75] M.E. Newman. Analysis of weighted networks. *Physical Review E*, 70(5):056131, 2004.

[76] M.E. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[77] M.E. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[78] M.E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[79] A. Noack and R. Rotta. Multi-level algorithms for modularity clustering. In J. Vahrenhold, editor, *Experimental Algorithms*, volume 5526 of *Lecture Notes in Computer Science*, pages 257–268. Springer, 2009.

[80] J. Peng and Y. Xia. A new theoretical framework for $k$-means-type clustering. In W. Chu and T.Y. Lin, editors, *Foundations and Advances in Data Mining*, volume 180 of *Studies in Fuzziness and Soft Computing*, pages 79–96. Springer, 2005.

[81] R.W.H. Sargent and A.W. Westerberg. Speed-up in chemical engineering design. *Transactions of the Institution of Chemical Engineers*, 42:190–197, 1964.

[82] B. Schölkopf, R. Herbrich, and A.J. Smola. A generalized representer theorem. In D. Helmbold and B. Williamson, editors, *Computational Learning Theory*, volume 2111 of *Lecture Notes in Computer Science*, pages 416–426. Springer, 2001.

[83] A. Shashua and A. Levin. Ranking with large margin principle: Two approaches. In *Proceedings Advances in Neural Information Procession Systems (NIPS)*, pages 937–944, 2002.

[84] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[85] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[86] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[87] K. Tatsumi, R. Kawachi, K. Hayashida, and T. Tanino. Multiobjective multiclass support vector machines maximizing geometric margins. *Pacific Journal of Optimization*, 6(1):115–141, 2010.

[88] V.A. Traag, P. Van Dooren, and Y. Nesterov. Narrow scope for resolution-limit-free community detection. *Physical Review E*, 84(1):016114, 2011.

[89] S. Umetani and M. Yagiura. Relaxation heuristics for the set covering problem. *Journal of the Operations Research Society of Japan*, 50(4):350–375, 2007.

[90] V.N. Vapnik. *Statistical learning theory*. Wiley, 1998.

[91] Y-C. Wei and C-K. Cheng. Ratio cut partitioning for hierarchical designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(7):911–921, 1991.

[92] F. Wesselmann and U.H. Suhl. Implementation techniques for cutting plane management and selection. Technical report, Technical Report Universität Paderborn, 2007.

[93] G. Xu, S. Tsoka, and L.G. Papageorgiou. Finding community structures in complex networks using mixed integer optimisation. *The European Physical Journal B*, 60(2):231–239, 2007.

[94] W.W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

# List of the Author's Work

## Refereed Journals

1. Yoichi Izunaga, Keisuke Sato, Keiji Tatsumi, and Yoshitsugu Yamamoto, "Row and column generation algorithms for minimum margin maximization of ranking problems," *Journal of The Operations Research Society of Japan*, Vol.58, No.4, pp.394–409, 2015.

## Refereed Conference Proceedings

1. Kotohumi Inaba, Yoichi Izunaga, and Yoshitsugu Yamamoto, "A Lagrangian relaxation algorithm for modularity maximization problem," *Operations Research Proceedings 2014*, to appear.

2. Yoichi Izunaga, Keisuke Sato, Keiji Tatsumi, and Yoshitsugu Yamamoto, "Row and column generation algorithm for maximization of minimum margin for ranking problems," *Operations Research Proceedings 2014*, to appear.

## Unpublished Papers

1. Yoichi Izunaga, and Yoshitsugu Yamamoto, "A cutting plane algorithm for modularity maximization problem," submitted for publication.